THE GOOD DRAWINGS D_n OF THE COMPLETE GRAPH K_n

by

Nabil H. Rafla

School of Computer Science

McGill University, Montreal

March, 1988

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Montreal, Quebec

March, 1988

Abstract

This thesis treats some of the problems related to the good drawings D_n of the complete graph K_n. The first of these problems is obtaining all the non-isomorphic good drawings D_n of K_n. After conjecturing that any good drawing D_n of K_n has at least one crossing-free Hamiltonian Circuit, an algorithm generating all the non-isomorphic good drawings D_n of K_n is developed. The second problem, determining the existence of a rectilinear drawing D_n of K_n with a given set of crossings, is solved by finding a characteristic of the rectilinear drawings D_n of K_n. An algorithm using this characteristic determines whether a given set of crossings defines a rectilinear drawing D_n of K_n. The last problem, to generate all the non-isomorphic rectilinear drawings D_n of K_n, is solved by an algorithm using a set of rectilinear drawings D_{n-1} of K_{n-1}.

Résumé

Cette thèse traite quelques problèmes ayant rapport aux bons dessins, "good drawings", D_n de K_n du graphe complet K_n. Le premier de ces problèmes est d'obtenir les bons dessins non-isomorphes D_n de K_n. Après avoir conjecturé que tout bon dessin D_n de K_n a au moins un cycle hamiltonien sans aucun croisement, un algorithme produisant tous les bons dessins D_n de K_n est développé. Le deuxième problème, déterminer l'existence d'un dessin D_n de K_n ayant un ensemble donné de croisements, est résolu en établissant une caractéristique des dessins rectilignes D_n de K_n. Un algorithme qui utilise cette caractéristique détermine si un ensemble donné de croisements définit un dessin rectiligne D_n de K_n. Le dernier problème, produire tous les dessins rectilignes non-isomorphes D_n de K_n, est résolu à l'aide d'un algorithme qui utilise un ensemble de dessins rectilignes D_{n-1} de K_{n-1}.

This work is dedicated to the memory of my father, Helmi Rafla.

Acknowledgements

This thesis would not have existed without the help of my Thesis Supervisor, Professor M. Newborn. His advice, suggestions and criticisms have been most valuable to this dissertation. He has always found time for discussing problems and eliminating obstacles no matter how onerous his other duties and activities have been. His enthusiastic encouragement, kindness and cheerful character have been a source of comfort and inspiration during what has often been a lonely and difficult journey.

I am very grateful to Professor R. K. Guy of the University of Calgary for his generosity in providing me with some of the important and valuable work which was accomplished by the late Professor A. Uytterhœven of Baal, Belgium and by Mr. J. Backelin of the University of Stockholm.

I am thankful to Professor P. Erdős for the time he spent discussing certain aspects of this paper.

I also thank Professor V. Chvátal of McGill University for his useful suggestions.

To Professor S. Zlobec of McGill University I express my gratitude for his encouragement in embarking on this project.

Finally, special thanks are due to Mr. Jeremy Tallboy for his many suggestions and for his work preparing the manuscript and drawing the numerous graphics.

Table of Contents

Part VI

Chapter 1

Introduction

This thesis treats some of the problems related to the good drawings D_n of the complete graph K_n. The first of these problems is to obtain all the non-isomorphic good drawings D_n of K_n. After conjecturing that any good drawing D_n of K_n has at least one Crossing-Free Hamiltonian Circuit, an algorithm generating all the non-isomorphic good drawings D_n of K_n is developed. The second problem, to determine the existence of a rectilinear drawing D_n of K_n with a given set of crossings, is solved by finding a characteristic of the rectilinear drawings D_n of K_n. An algorithm which uses this characteristic determines whether a given set of crossings defines a rectilinear drawing D_n of K_n. The last problem, to generate all the non-isomorphic rectilinear drawings D_n of K_n, is solved by an algorithm using a set of rectilinear drawings D_{n-1} of K_{n-1}.

Before proceeding however, let us examine the definitions which are required throughout the thesis. A *graph* G is a set V of *vertices* and a subset E of the *unordered pairs of vertices*, called *edges*. A *drawing* D of a graph G is a mapping of G into a surface, which in this thesis will be the Euclidean plane [8,10,14,15]. The vertices are mapped into distinct points, called *nodes*. An edge is mapped into an open *arc* which is closed by its two defining nodes. A *good drawing* is a drawing in which (see Fig. 1.1):

i) no arc intersects itself,

ii) no two arcs incident with a common node have a common point (the tangents to the arcs at the point are distinct)

iii) no two arcs have more than one point in common.

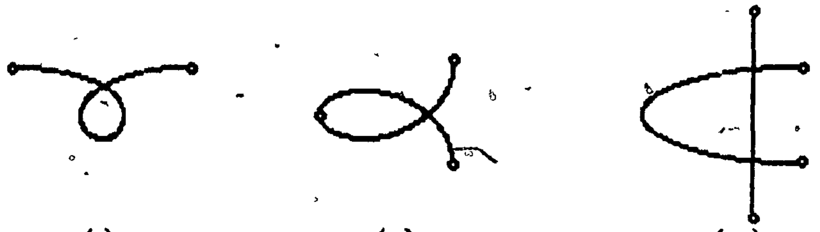


Fig 1.1: The three cases which must be avoided to obtain a good drawing.

A *good arc* of a drawing D is an arc which does not intersect itself. A common point of two arcs is called a *crossing*. We assume that no point of the plane belongs to more than two arcs. The *complete graph*, K_n, has n vertices and all $\binom{n}{2}$ possible edges. A *crossing optimal* drawing D_n of K_n is one which has, among all possible drawings of K_n, the minimum number of crossings. This least number of crossings is called the *crossing number* of K_n and is denoted by $\nu(K_n)$. If we consider only the rectilinear drawings of K_n then the minimum number of crossings is called the *rectilinear crossing number* of K_n. The *node responsibility* is the total number of crossings on all arcs incident with this node. The *arc responsibility* is the total number of times this arc is crossed. For the purposes of this thesis two drawings D and D' are called *isomorphic* when there is a one-to-one correspondence between the nodes of D and the nodes of D' such that if any pair of arcs in D crosses then the corresponding pair in D' also crosses. This is a weak definition of isomorphism. Examples are given in Fig. 1.2. The usual definition of isomorphism would distinguish between drawings in which the same crossings on an arc were made in different orders.

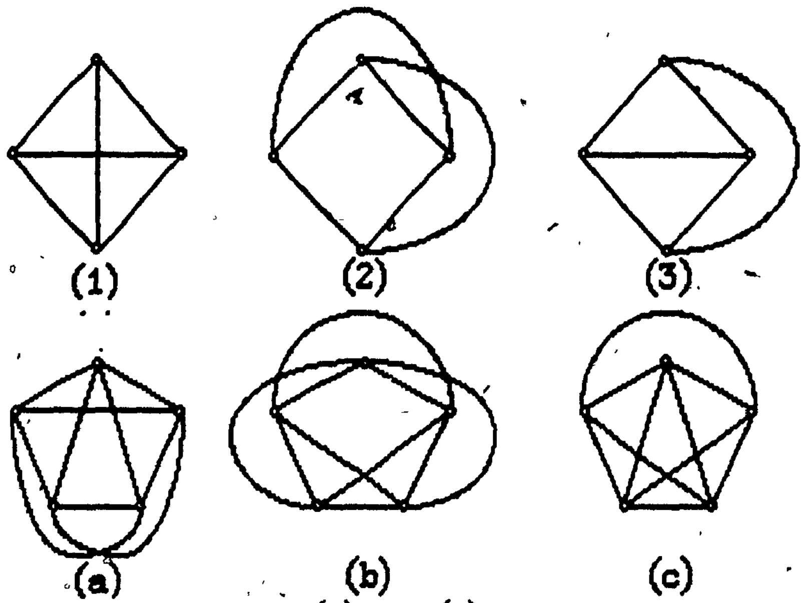


Fig.1.2: The drawings (1) and (2) are isomorphic; while (1) and (3) are non-isomorphic. Similarly (a) and (b) are isomorphic but (a) and (c) are non-isomorphic.

If all the arcs of a drawing D are restricted to straight line segments then D is a *rectilinear drawing,* as in the drawing of Fig. 1.3.

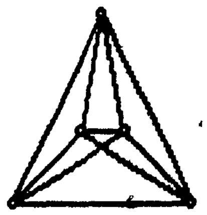


Fig. 1.3: A rectilinear drawing.

A *k-circuit* of a drawing D_n is a sequence of arcs $(a_1, a_2), (a_2, a_3), \ldots, (a_i, a_{i+1}), \ldots, (a_{k-1}, a_k), (a_k, a_1)$, such that $a_i \neq a_j$ whenever $i \neq j$. A *Hamiltonian Circuit,* HC, of a drawing D_n is an n-circuit. If no two arcs in an HC cross one another, then it is called a *crossing-free Hamiltonian Circuit* or C-F HC. Examples of C-F HC's are given in Fig.1.4. An *n-circuit optimal* drawing D_n of K_n is one which has, among all possible drawings of K_n, the maximum number of C-F HC's. If only the rectilinear drawings D_n of K_n are considered then the term k-circuit is replaced by *k-gon*. An *n-gon optimal* rectilinear drawing D_n of K_n is one which has, among all possible rectilinear drawings of K_n, the maximum number of C-F HC's.

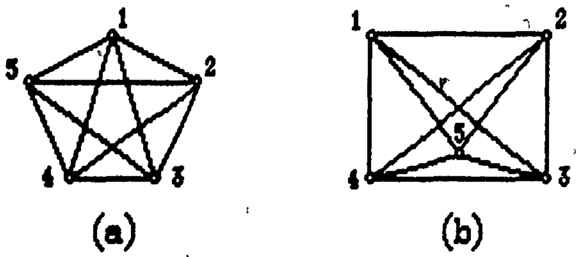


Fig.1.4: Drawing (a) has only one C-F HC, namely (1,2,3,4,5,1) Drawing (b) has four different C-F HC's: (1,2,3,4,5,1), (1,2,3,5,4,1), (1,2,5,3,4,1) and (1,4,3,2,5,1).

Although studies related to the complete graph K_n started at least a few decades ago, only partial results have been reported thus far. Many of these results concern the crossings and the crossing number of K_n [3 - 5,7 - 9,16 -20]. The upper bound for the crossing number of K_n,

$$\nu(K_n) \leq \tfrac{1}{4}[\tfrac{1}{2}n][\tfrac{1}{2}(n-1)][\tfrac{1}{2}(n-2)][\tfrac{1}{2}(n-3)],$$

where brackets denote *greatest integer not greater than*, has been obtained by R. K. Guy [3]. It has been conjectured that the exact value of the crossing number of K_n is given by this bound. Guy confirmed this conjecture for $n \leq 7$ [8], then later he confirmed it for $n \leq 10$ [13]. Lower bounds for $\nu(K_n)$ have also been obtained by Guy [13]. For the rectilinear crossing number of K_n, an upper bound has been ob–

tained by H. F. Jensen [12] and independently by R. B. Eggleton. This bound was then independently improved by D. Singer and H. F. Jensen. Guy [14] has confirmed that the rectilinear crossing number for K_8 is 19. R. B. Eggleton [16] obtained all the non-isomorphic drawings D_5 of K_5. Most of the drawings D_6 of K_6 can be found in the correspondence between R. Guy and both A. Uytterhœven and J. Backelin [18]. The first part of the thesis, Chapters 2, 3 and 4, treats the problem of generating, for a given n, all the non-isomorphic drawings D_n of the complete graph K_n. An algorithm that generates all the non-isomorphic good drawings D_n of K_n is developed and results are obtained by the corresponding computer program for $n \leq 7$. The only input to this algorithm is the value of n. In the third part of the thesis, Chapters 8, 9 and 10, the non-isomorphic rectilinear drawings D_n of K_n are obtained using a set of rectilinear drawings D_{n-1} of K_{n-1}. An algorithm is written to obtain these drawings D_n. Using the corresponding computer program, the non-isomorphic rectilinear drawings D_7 are obtained using a set of rectilinear drawings D_6.

The problem concerning straight line representation of graphs has been considered for many years [1,2,11,16]. A necessary and sufficient condition for some graphs to be drawn rectilinearly was presented by Eggleton [16]. A similar condition for the complete graph K_n is studied in the second part of this thesis, Chapters 5, 6 and 7. A computer program is written to determine whether there exists a rectilinear drawing D_n satisfying a given set of crossings of K_n.

We summarize the three main contributions of the thesis in the following:

1. **Generating all the non-isomorphic drawings D_n of K_n having at least one C-F HC.** In Chapter 2, along with the relevant theory, an algorithm is developed to generate, for a given n, all the drawings D_n. The results of this algorithm for $3 < n \leq 7$ are presented in Chapter 3 and Appendices A.2 and A.3. Also in Chapter 3, having on hand the entire set of all drawings D_7, we are able to determine all those with the largest number of C-F HC's, confirming the results obtained by Newborn and Moser[19]. Chapter 4 includes an analysis of the algorithm.
2. **Determining whether a drawing D_n of K_n (as specified by a set of crossings) is rectilinear.** In Chapter 5 we prove that all non-rectilinear drawings D_n of K_n always have a specific sub-drawing which cannot be a sub-drawing of a rectilinear drawing. This characteristic can be determined by just knowing the set of crossings of the drawing. In Chapters 6 and 7 we present the corresponding results and analysis.
3. **Generating the non-isomorphic rectilinear drawings D_n given the rectilinear drawings D_{n-1}.** In Chapter 8, the relevant theory is presented to show that we can obtain the rectilinear drawings D_n using the rectilinear drawings D_{n-1}. The corresponding algorithm is presented in Chapter 8. In Chapter 9, the summary of the results for n = 7 is given; while the actual drawings are shown in Appendix C.2. An analysis of the algorithm is provided in Chapter 10.

Chapter 2

AN ALGORITHM FOR FINDING ALL THE NON-ISOMORPHIC GOOD DRAWINGS D_n OF K_n HAVING A CROSSING-FREE HAMILTONIAN CIRCUIT.

Work related to obtaining all the non-isomorphic good drawings D_n of K_n started a few years ago, [15,16,18]. In this chapter an algorithm is developed to generate all the non-isomorphic good drawings having at least one C-F HC, **C**. It can be shown that each rectilinear drawing D_n of K_n has a C-F HC. A proof of the existence of a C-F HC in each non-rectilinear good drawing D_n of K_n has not been obtained yet. The large number of unsuccessful trials to produce a good drawing D_n of K_n with no C-F HC created the belief that there are no such good drawings. This belief is strengthened by examining both sets of drawings D_6 of K_6 obtained by the late Professor Uytterhœven and by Mr. Backelin and finding that each of the drawings has a C-F HC. From these facts we obtain the following conjecture.

Conjecture

Every good drawing D_n of K_n has at least one C-F HC.

We note that if D_n is not a good drawing it might still have a C-F HC as shown in Fig. 2.1. Only the drawings that have a C-F HC are considered in this chapter.

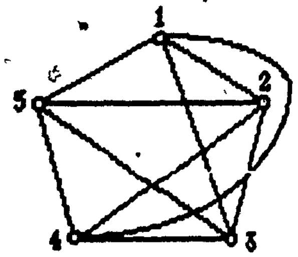


Fig.2.1: A drawing D_5 in which the two arcs (1,3) and (1,4) cross. D_5 is not a good drawing. however it has a C-F HC, **C**=(1,2,3,4,5,1).

General Description of the Algorithm

The algorithm begins by determining the list of all the edges (α,β) of K_n different from the edges of $\mathbf{C}=(1,2,3,\ldots,n-1,n,1)$. For each edge (α,β), the algorithm generates all the arcs (a,b) into which (α,β) could be mapped, such that (a,b) crosses any of the arcs of $\mathbf{C}$ at most once and does not cross any of the four arcs of $\mathbf{C}$ which are incident to either node a or node b. If (a,b) is a good arc, i.e. does not cross itself, then it is retained by the algorithm; otherwise it is discarded. At this point the algorithm has the list of good arcs (a,b) related to each of the edges (α,β).

Next, to the arcs of $\mathbf{C}$, the algorithm adds arcs (a,b), one arc per edge and one arc at a time. Before adding an arc, the algorithm verifies whether the crossings occurring between this arc and each of the arcs which were previously added to $\mathbf{C}$ do not violate conditions (ii) and (iii) (on page 1) for a good drawing. If for each edge (α,β) an arc (a,b) is added to $\mathbf{C}$ then the algorithm has obtained a good drawing D_n of K_n.

The algorithm compares this drawing D_n to the non-isomorphic drawings which were previously obtained. If D_n is non-isomorphic to each of these, then it is added to the set of non-isomorphic good drawings of K_n.

Detailed Description of the Algorithm

Consider n edges of K_n and map them into a C-F HC $\mathbf{C}=(1,2,3,\ldots,n-1,n,1)$. We call the area bounded by $\mathbf{C}$, **Int C**, and the remainder of the plane, **Ext C**. Consider an edge (α,β) different from the edges of $\mathbf{C}$. The possible mappings of an edge (α,β) with respect to $\mathbf{C}$ are illustrated in Fig. 2.1.a, b, c, d and e for n = 5, 6, and 7. In

Theorem 2.1, we show how we determine that a particular mapping (a, b) of (α, β) is a good arc.

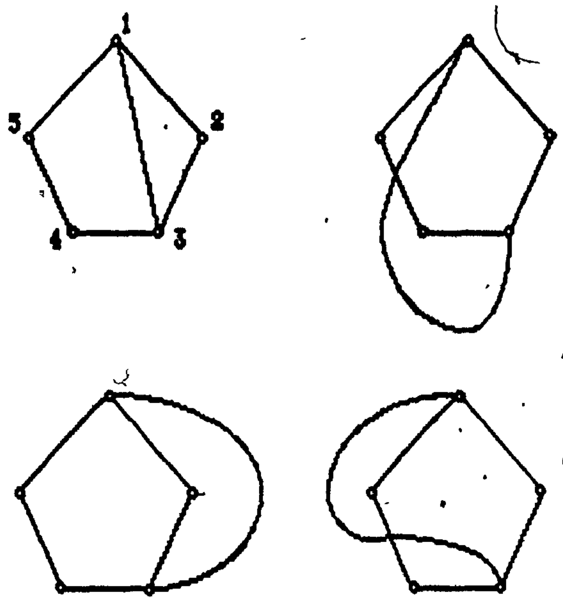


Fig.2.1.a: Each of the edges (1,3), (2,4), (3,5), (4,1,) and (5,2) of K_5 can be mapped into exactly 4 arcs.

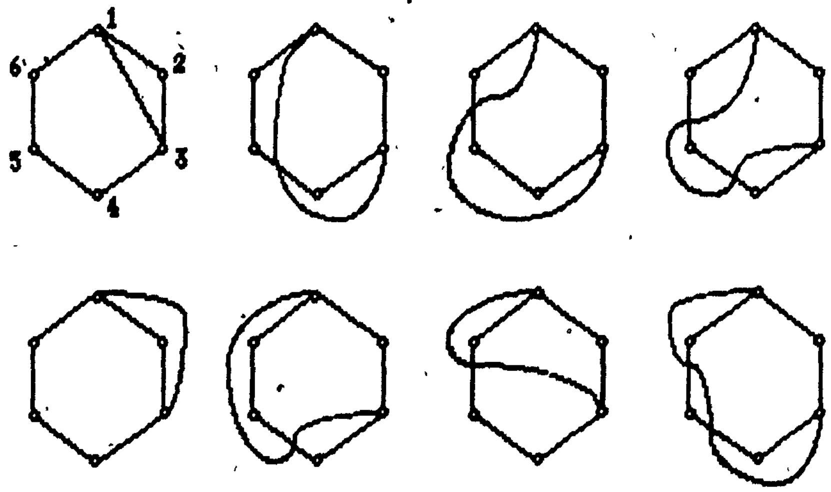


Fig.2.1.b: Each of the edges (1,3), (2,4), (3,5), (4,6), (5,1) and (6,2) of K_6 can be mapped into exactly 8 arcs.

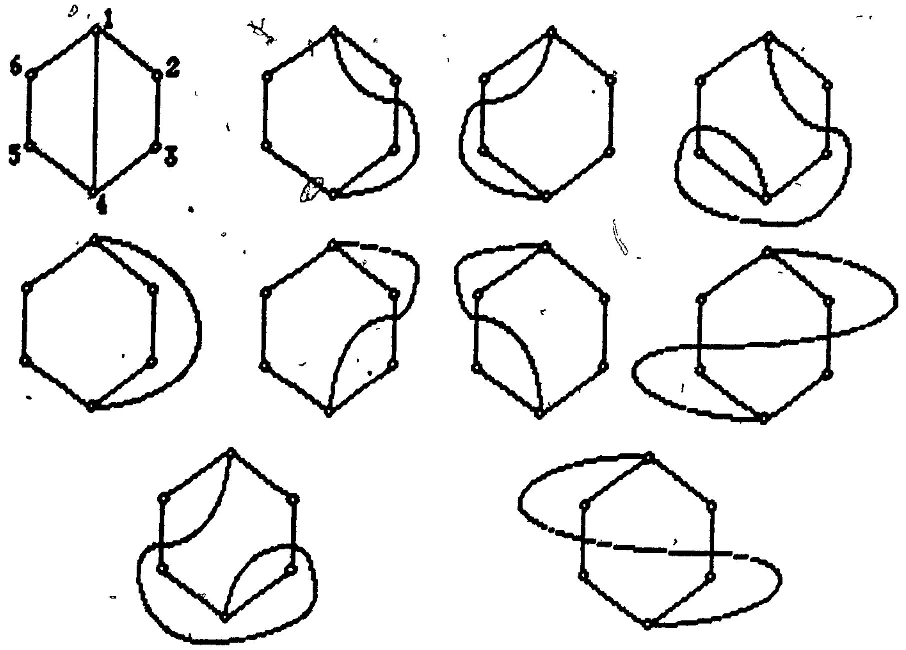
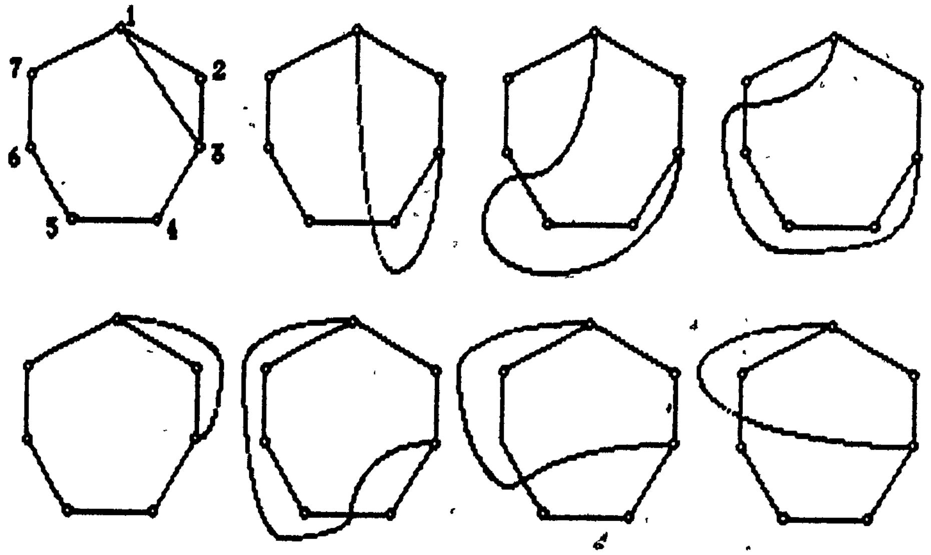


Fig. 2.1.c: Each of the three edges (1,4), (2,5), and (3,6) of K_6 can be mapped into exactly 10 arcs.



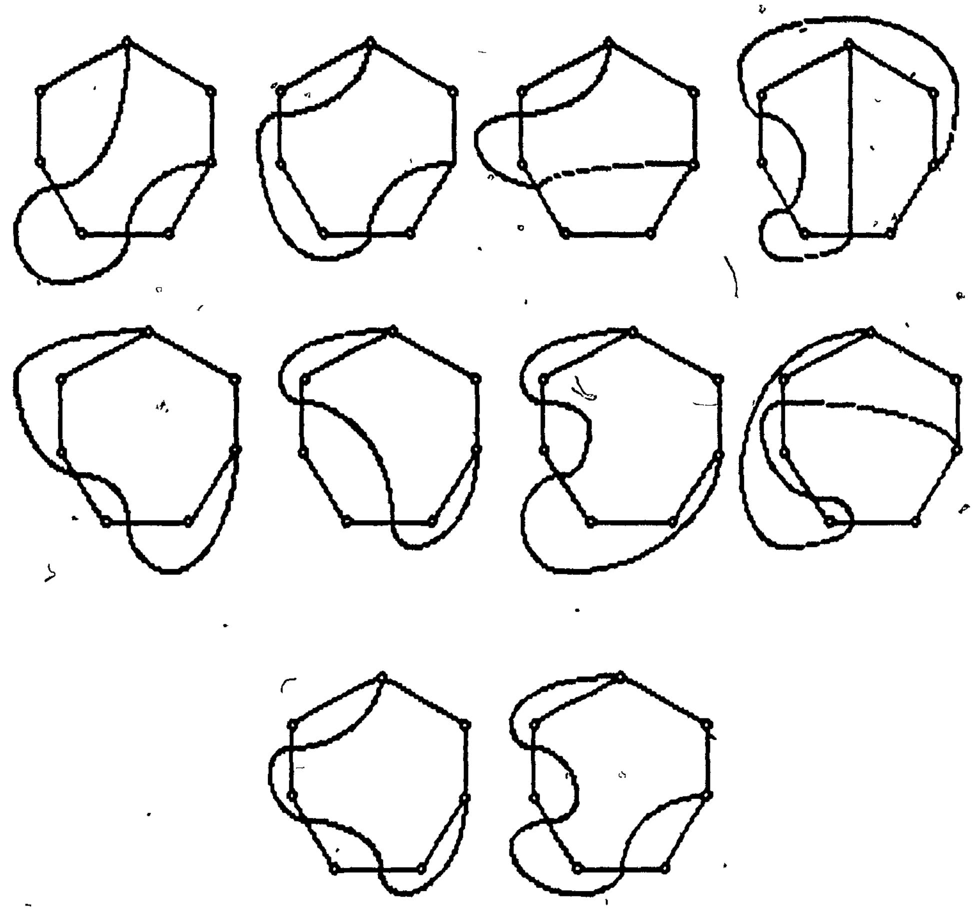
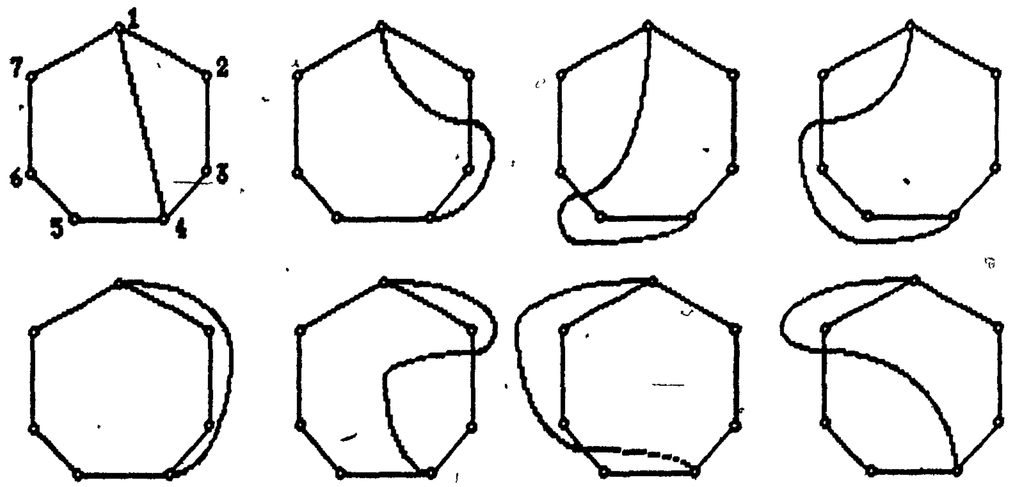


Fig.2.1.d: Each of the edges (1,3),(2,4), (3,5), (4,6), (5,7), (6,1) and (7,2) of K_7 can be mapped into exactly 18 arcs.



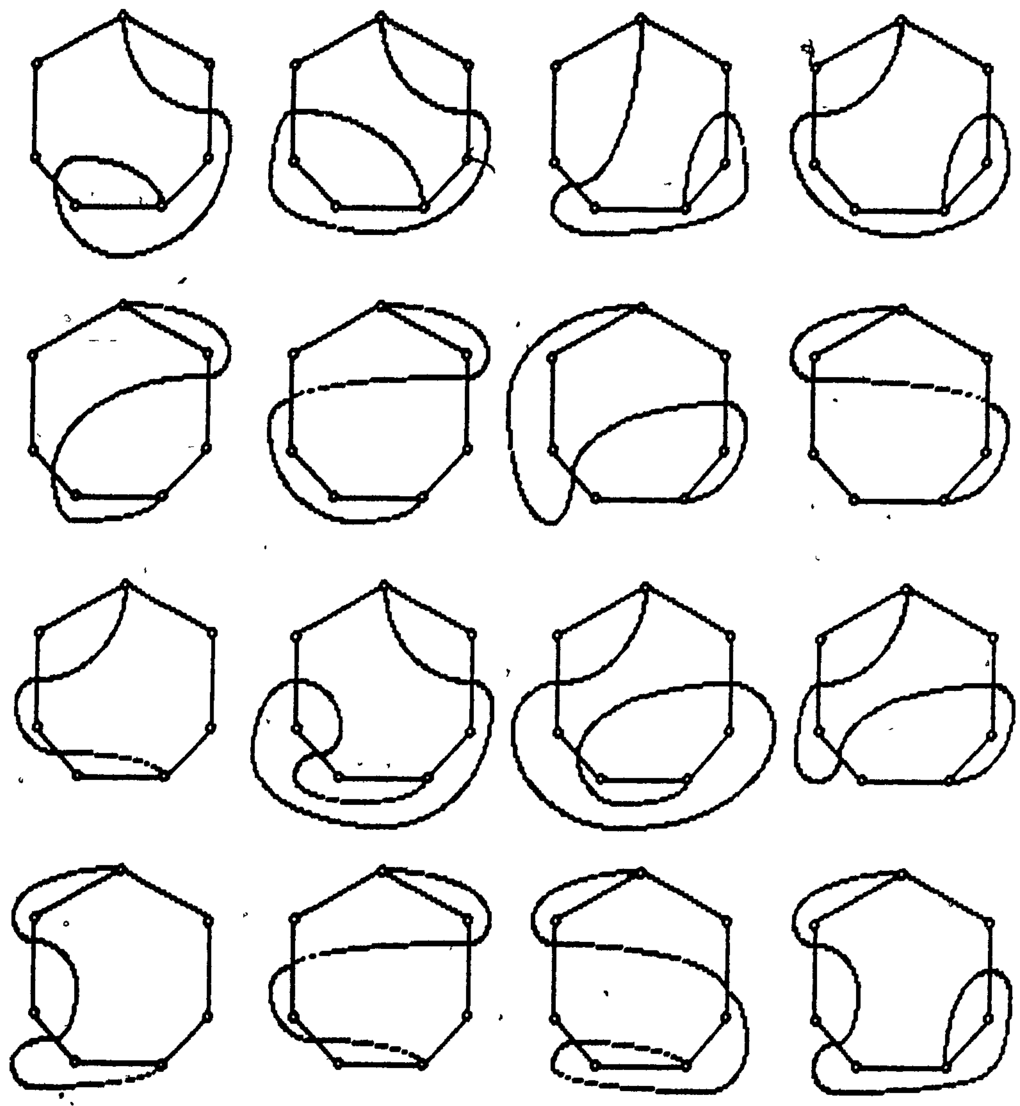


Fig.2.1.e: Each of the arcs(1,4), (2,5), (3,6), (4,7), (5,1), (6,2) and (7,3) of K_7 can be mapped into exactly 24 arcs.

Theorem 2.1

Let D_n be a drawing (not necessarily a good drawing) of the complete graph K_n, $\mathbf{C}$ be a C-F HC of D_n and (a,b) be an arc of D_n different from the arcs of $\mathbf{C}$. In addition, if $\mathbf{C}$, the arcs of $\mathbf{C}$ crossed by (a,b) and the location of each of the segments of (a,b) with respect to $\mathbf{C}$ are given (i.e. whether it is in $\mathrm{Int}\ \mathbf{C}$ or $\mathrm{Ext}\ \mathbf{C}$), then, one can determine whether (a,b) is a good arc.

Proof

As defined in Chapter 1, a good arc is one which does not intersect itself. We prove Theorem 2.1 by constructing an algorithm that determines whether an arc intersects itself.

Let D_n be a drawing of K_n with at least one C-F HC, **C**. Label the nodes of D_n such that **C** becomes $(1,3,5,\ldots,2n-1,1)$. We label the arcs of **C** with the integers 2, 4, 6, ..., 2n as shown in Fig.2.2.

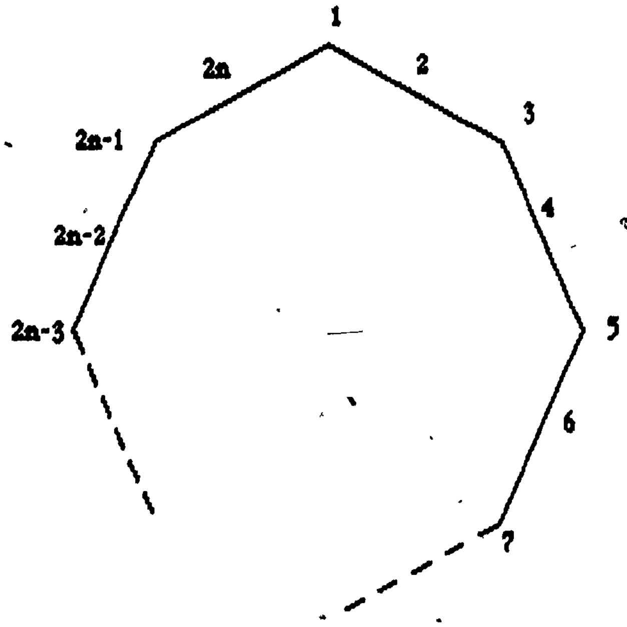


Fig.2.2: A C-F HC of a drawing D_n.

The arcs of **C** crossed by (a, b) will be denoted by $p_1, p_2, \ldots, p_{r-1}$. The notation $(p_0, p_1, p_2, \ldots, p_{r-1}, p_r)$ will mean that starting from node $a \equiv p_0$ and going to node $b \equiv p_r$, the arcs $p_1, p_2, \ldots, p_{r-1}$ are crossed by (a, b) in this order. The arc (a, b) is said to be composed of segments $(p_0, p_1), (p_1, p_2), \ldots, (p_{r-1}, p_r)$. For example, in Fig.2.3, the arc (a, b) is composed of three segments $(p_0, p_1), (p_1, p_2)$ and (p_2, p_3).

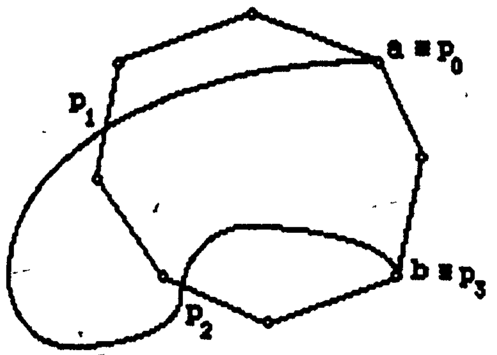


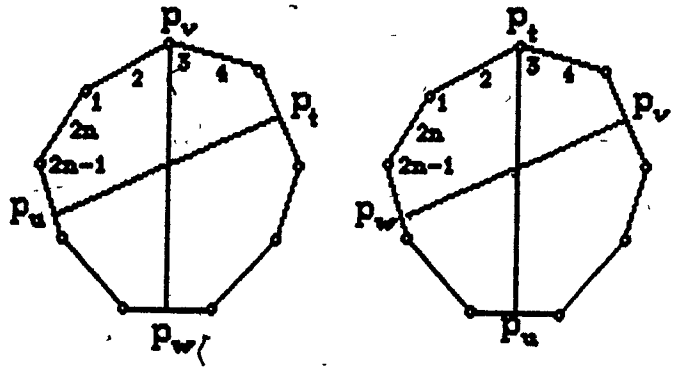
Fig.2.3: The arcs p_1 and p_2 are crossed by the arc (a, b). Starting from node a, p_1 is crossed first, followed by p_2.

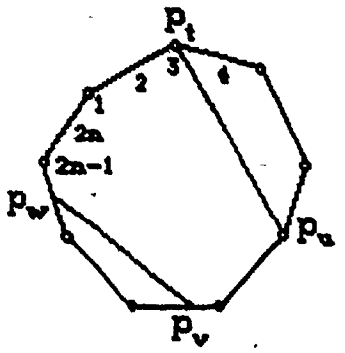
Now form a list of all the segments of the arc (a, b) lying in **Int C**. Let these segments be denoted by (p_i, p_j). Each pair (p_t, p_u) and (p_v, p_w) is considered separately. Assuming that $p_t < p_u$ and $p_v < p_w$, we have the following:

$$\left.\begin{array}{c} p_v < p_t < p_w < p_u, \\ \text{or} \\ p_t < p_v < p_u < p_w \end{array}\right\} \Leftrightarrow (p_t, p_u) \times (p_v, p_w)$$

(Fig.2.4)

We note that if there are no segments in **Int C** then (a, b) does not cross itself in **Int C**. In an analogous way, the segments of the arcs (a, b) lying in **Ext C** are considered. □





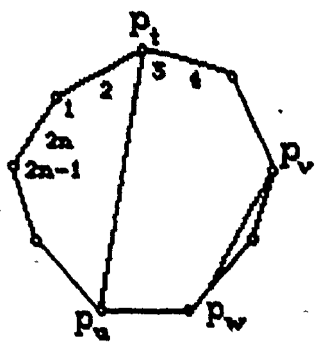


Fig.2.4: There is a crossing only in the first two diagrams.

We have explained previously that the algorithm adds arcs one at a time to $\mathbb{C}$ to form a good drawing of K_n. Before adding each arc, the algorithm verifies whether the arc crosses any of the arcs already added in a way that violates conditions (ii) or (iii) for a good drawing. Two examples of such violations are given in Fig. 2.5. In the following theorem we show that given two arcs (a,b) and (c,d), we can determine whether (a,b) crosses (c,d) and whether they cross more than once. The ability to determine how (a,b) and (c,d) cross is necessary because the algorithm considers, with respect to $\mathbb{C}$, each good arc (a,b) of an edge (α,β) along with each good arc (c,d) of an edge (σ,δ), where (α,β) and (σ,δ) are different from the edges of $\mathbb{C}$. The algorithm stores information about each pair (a,b), (c,d) reflecting whether they cross and if so, whether there is any violation of conditions (ii) or (iii).

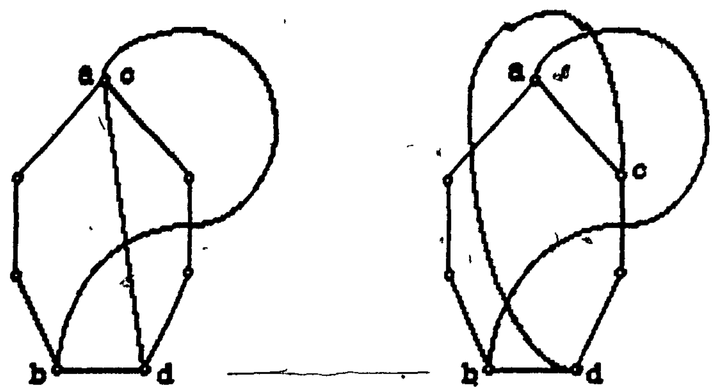


Fig.2.5: Two pairs of arcs (a,b) and (c,d) resulting in drawings which are not good drawings. The first violates condition (ii) and the second violates condition (iii).

In this theorem we demonstrate that, given $\mathfrak{C}$, two arcs (a,b) and (c,d) of D_n, the arcs of $\mathfrak{C}$ crossed by each of (a,b) and (c,d), and the location, $\mathfrak{Int}\ \mathfrak{C}$ or $\mathfrak{Ext}\ \mathfrak{C}$, of each of the segments of (a,b) and (c,d) with respect to $\mathfrak{C}$, we can determine whether (a,b) crosses (c,d) and whether the two arcs cross more than once. This theorem applies to drawings D_n of K_n which have at least one C-F HC.

Theorem 2.2

Let D_n be a drawing (not necessarily a good drawing) of the complete graph K_n, $\mathfrak{C}$ be a C-F HC of D_n, and (a,b) and (c,d) be two arcs of D_n which are different from the arcs of $\mathfrak{C}$. Let the sub-drawing consisting of $\mathfrak{C}$ and (a,b) and the one consisting of $\mathfrak{C}$ and (c,d) be good drawings. In addition, if $\mathfrak{C}$, the arcs of $\mathfrak{C}$ crossed by (a,b), the arcs of $\mathfrak{C}$ crossed by (c,d) and the location of each of the segments of (a,b) and (c,d) with respect to $\mathfrak{C}$ are known; then, we can determine whether (a,b) and (c,d) cross and whether they cross more than once.

Proof

Let D_n be a drawing of K_n with at least one C–F HC, $\mathbf{C}$. We label the nodes of D_n such that $\mathbf{C}$ becomes $(1,5,9,\ldots,4n-3,1)$. An arc $(i,i+4)$ of $\mathbf{C}$ could be crossed by both (a,b) and (c,d). To avoid the meeting of (a,b) and (c,d) at the same point of $(i,i+4)$, we consider two distinct points on $(i,i+4)$ where it could be crossed by (a,b) and (c,d). We label the point closer to node i with the integer $i+1$ and the one closer to $i+4$ with the integer $i+3$, as shown in Fig.2.6.

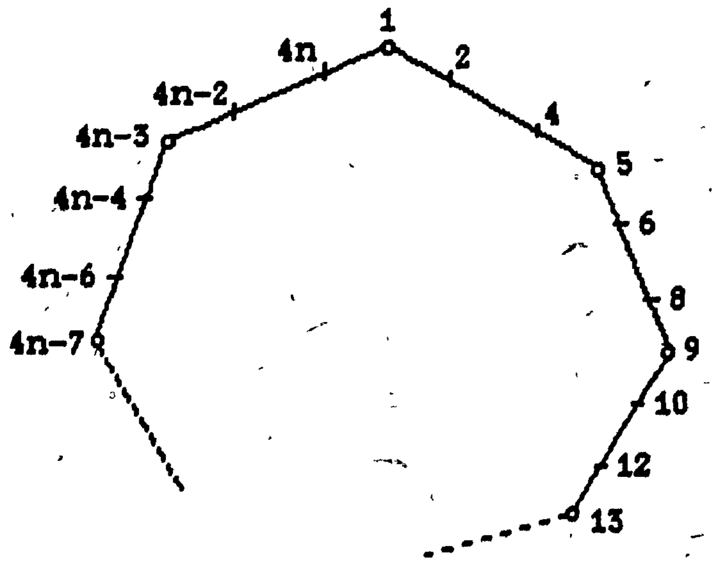


Fig.2.6: A C-F HC of a drawing D_n.

We assume that $a < b$, $c < d$ and $a \le c$. The arcs of $\mathbf{C}$ crossed by (a,b) will be denoted by p_1, $p_2,\ldots,p_{r-1}$ and the arcs crossed by (c,d) will be denoted by q_1, $q_2,\ldots,q_{s-1}$. The notation $(p_0,p_1,p_2,\ldots,p_{r-1},p_r)$ will mean that starting from node $a = p_0$ and going to node $b = p_r$, the arcs labelled with the integers $p_1,p_2,\ldots,p_{r-1}$ are crossed by (a,b) in this order. The equivalent notation $(q_0,q_1,q_2,\ldots,q_{s-1},q_s)$ is used to denote the arcs of $\mathbf{C}$ crossed by (c,d) and their order.

When both (a,b) and (c,d) cross an arc (i,i+4) of **C**, we have two ways of drawing (a,b) and (c,d):

1) (a,b) crossing at point i+1 and
   (c,d) crossing at point i+3;
2) (a,b) crossing at point i+3 and
   (c,d) crossing at point i+1.

If k arcs of **C** are crossed by both (a,b) and (c,d), then we get 2^k ways of drawing them. The number of times these two arcs cross is determined in the following manner:

Consider all the segments of the two arcs (a,b) and (c,d) lying in **Int C**. Let these segments be denoted by (p_i, p_j) when they belong to (a,b) and by (q_i, q_j) when they belong to (c,d), as shown in Fig.2.7.

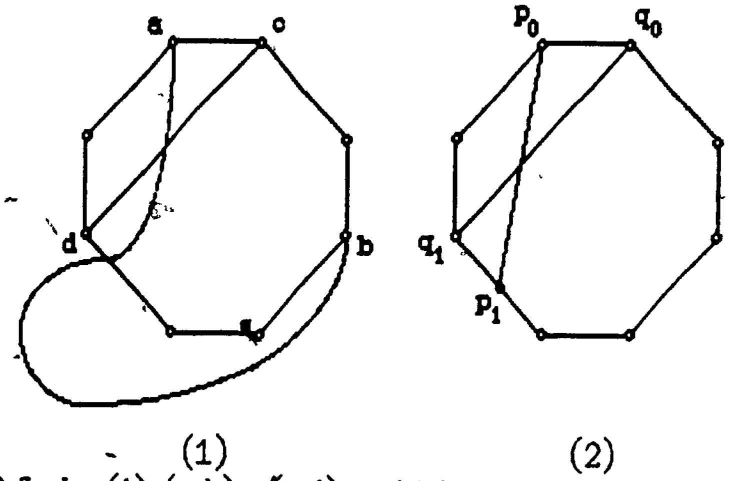


Fig.2.7: In (1) (a,b) x (c,d), which is reflected also in (2) by the two segments (p_0, p_1) and (q_0, q_1) crossing.

Each pair (p_t, p_u) and (q_v, q_w) is considered separately. Assuming that $p_t < p_u$ and $q_v < q_w$, we have the following:

$$\left.\begin{array}{c} q_v < p_t < q_w < p_u \\ \text{or} \\ p_t < q_v < p_u < q_w \end{array}\right\} \Longleftrightarrow (p_t, p_u) \times (q_v, q_w)$$

(Fig. 2.8)

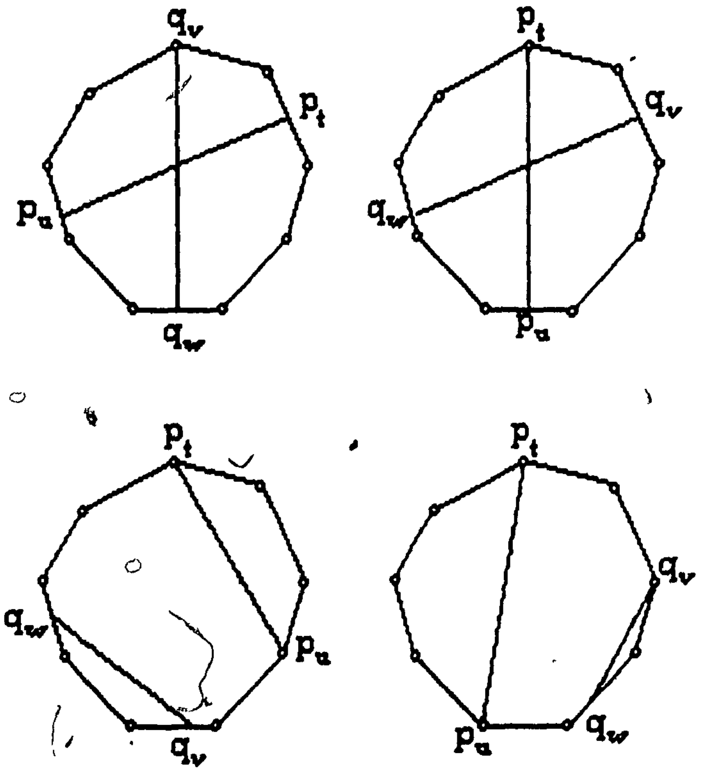
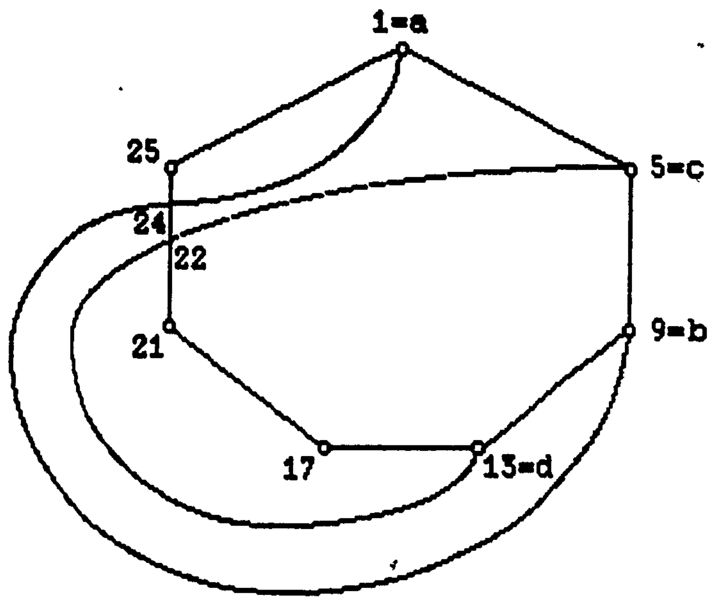


Fig. 2.8: There is a crossing only in the first two diagrams.

In an analogous way, the segments of the arcs of (a,b) and (c,d) lying in $\mathfrak{Ext}\ \mathfrak{C}$ are considered.

□

We note that whenever k arcs of $\mathbf{C}$ are crossed by both arcs (a, b) and (c, d), these two arcs can be drawn in 2^k possible ways; but only the drawings with the least number of crossings are good drawings, as shown in the two examples of Figs. 2.9 and 2.10.



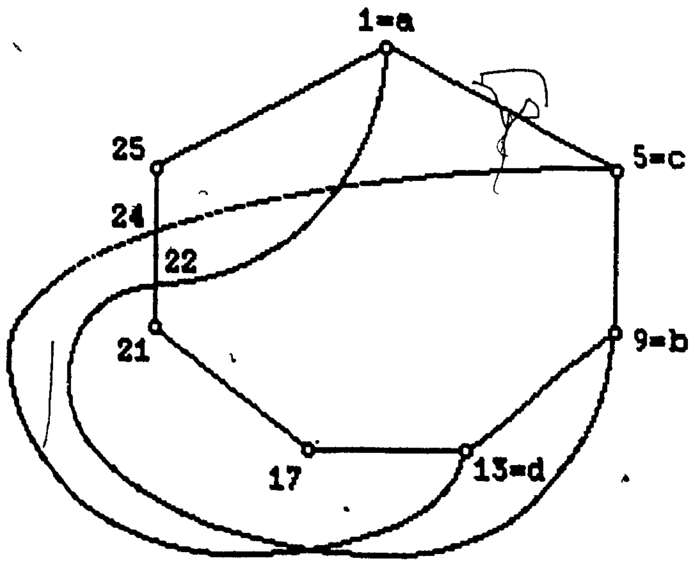


Fig. 2.9: In the two drawings (a, b) and (c, d) cross the arc (21, 25) but only the first drawing is good.

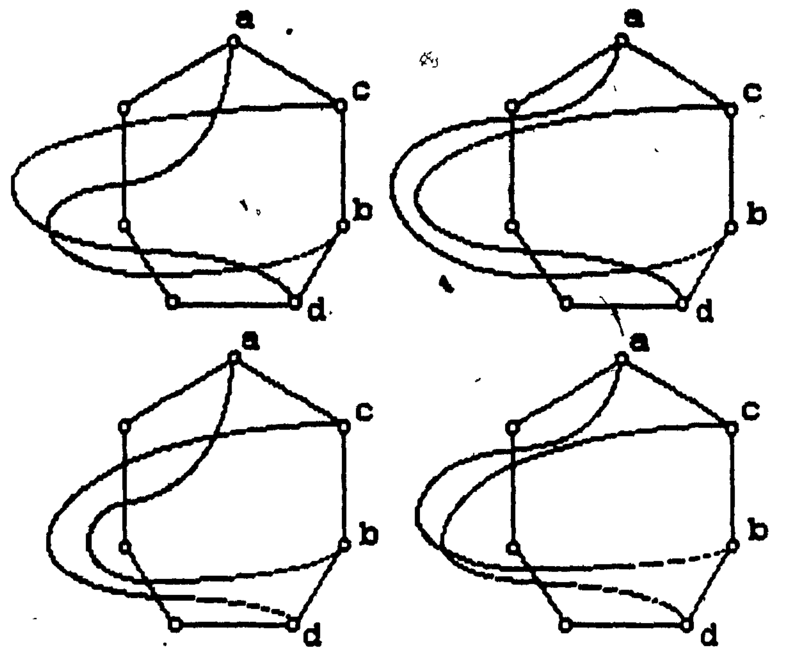


Fig.2.9: The arcs (a,b) and (c,d) cross two arcs of C in four different ways. The first drawing is not a good drawing since (a,b) crosses (c,d) more than once.

BACKGROUND TO THE ALGORITHM

The purpose of the algorithm is to generate all the good drawings D_n of the complete graph K_n, having a C-F HC. We let $\mathbf{C} \equiv (1,2,3,\ldots,n-1, n,1)$ be a C-F HC of all the drawings D_n of K_n.

Now, suppose that two edges (α, β) and (γ, δ) of K_n, which are different from the edges of $\mathbf{C}$, can be mapped into p arcs and q arcs respectively. Then each of the p arcs is considered along with each of the q arcs to generate $p \times q$ sub-drawings, (many of which will not be good drawings).

Hence, the idea behind the algorithm is to determine all the different arcs of each of the edges of K_n (with respect to a C-F HC, $\mathbf{C}$, of K_n) and to generate the drawings consisting of the different combinations of these arcs, as shown in Figs.2.10 and 2.11.

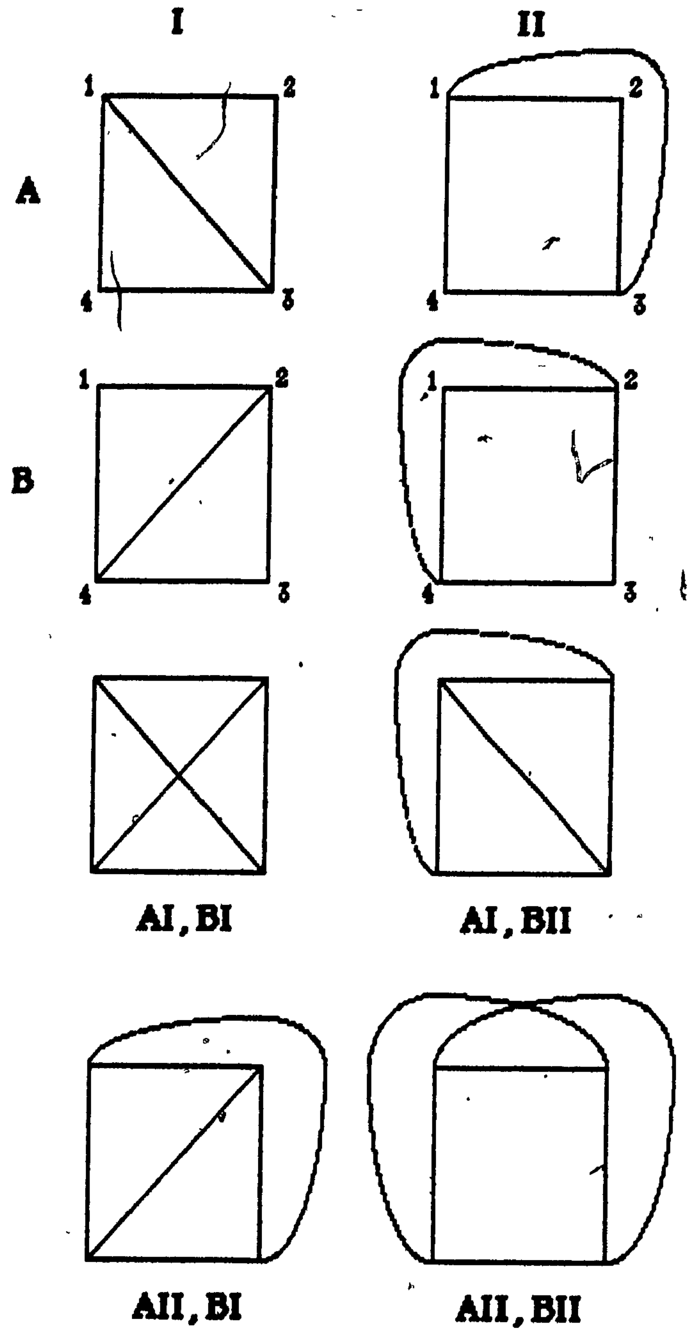


Fig.2.10: For K_4, the edges (1,2),(2,3),(3,4) and (4,1) are mapped to form a C-F HC. The algorithm will generate all the drawings obtained by combining one sub-drawing from row A with another from row B.

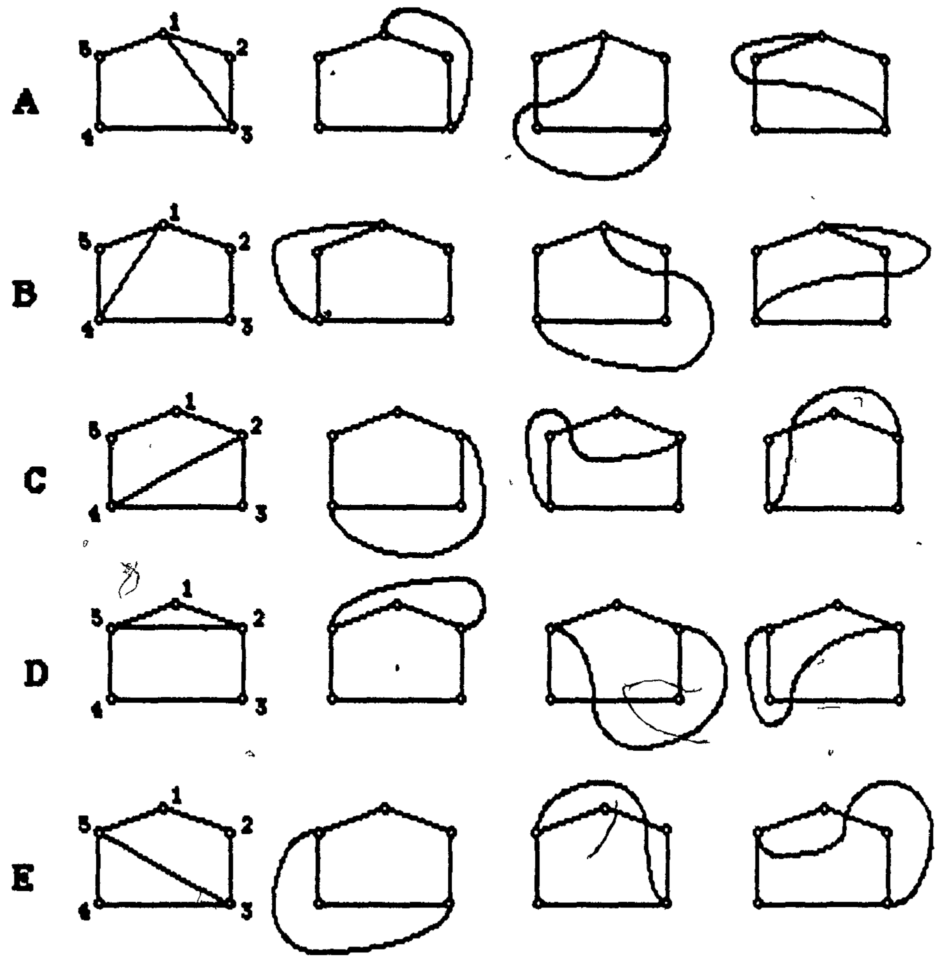


Fig.2.11: A C-F HC (1,2,3,4,5,1) is used for K_5. The algorithm will generate all possible drawings by combining exactly one sub-drawing from each of the rows A, B, C, D and E.

Let R be a set of arcs obtained by mapping each of the edges of K_n different from the edges of $\mathbf{C}$ into an arc. For example, considering the drawings in Fig. 2.15, if we take one of the arcs (1,3) from row A, an arc (1,4) from row B, an arc (2,4) from row C, an arc (2,5) from row D, and an arc (3,5) from row E; then we have a set of arcs R.

In general, there are $m = \binom{n}{2} - n$ edges of K_n apart from the edges of $\mathbf{C}$. If we denote the i-th edge of these by a_i and the j-th good mapping of a_i by a_{ij}, then a set $A_p = \{a_{ij}\}$ will consist of m arcs each of which is a mapping of an edge a_i of the m edges of K_n. If the number of good mappings of a_i is r_i, then the number of sets A_p will equal the product of all r_i where $i = 1, 2, \ldots, m$.

After generating the edges a_i and determining the good arcs into which they could be mapped, the algorithm generates a matrix $MX(\cdot,\cdot)$, reflecting the relation between any two arcs α and β of the different sets A_i (i.e. whether they cross, and, in case they cross, whether their crossing violates the rules of a good drawing). If a pair of arcs α and β in A_i does not cross then $MX(\alpha,\beta)$ is set to zero. If α and β cross without violating the rules of a good drawing then $MX(\alpha,\beta)$ is set to one. Otherwise $MX(\alpha,\beta)$ is set to two.

From the information generated in the matrix $MX(\cdot,\cdot)$, the algorithm then checks the relationship between each pair of arcs of a set A_i. If for any pair α and β, $MX(\alpha,\beta)$ equals 2 then this set A_i is discarded and the algorithm starts checking the pairs of arcs of a new set. If there exists no such pair of arcs in A_i then a set of crossings, X, corresponding to a good drawing D_n of K_n is obtained. If the drawing corresponding to X is isomorphic to a drawing corresponding to any of the previously generated sets of crossings then X is discarded by the algorithm. Otherwise the algorithm adds X to a set D consisting of the sets of crossings corresponding to the non-isomorphic good drawings D_n of K_n. The algorithm stops when all the sets A_i are checked.

THE ALGORITHM

Step 1 Generate a list of the m edges $a_1, a_2, \ldots, a_p, \ldots, a_m$ of K_n where a_p is not an edge of $\mathbf{C}$ and $m = \binom{n}{2} - n$.

Step 2 For each a_p, determine its corresponding good arcs according to condition (i).

Step 3 For each pair of arcs α and β, determine whether they cross, and whether any of the conditions (ii) and (iii) is met. The existence or non-existence of such crossings is reflected in a matrix $MX(\cdot,\cdot)$,

$$\textbf{where } MX(\alpha,\beta) = \begin{cases} 0 & \text{if arcs } \alpha \text{ and } \beta \text{ do not cross} \\ 1 & \text{if arcs } \alpha \text{ and } \beta \text{ cross such that the sub-drawing } \mathfrak{C} \cup \alpha \cup \beta \text{ is a good drawing} \\ 2 & \text{otherwise} \end{cases}$$

```
Step 4.0   i ← 1.
           $\mathfrak{D} \leftarrow \emptyset$   ($\mathfrak{D}$ will be composed of the sets of crossings
                  corresponding to the non-isomorphic drawings
                  $D_n$).
Step 4.1   Form a set $A_i$ where $A_i$ consists of m arcs, each of which
           is a mapping of one of the m edges of $K_n$ different from
           the edges of $\mathfrak{C}$.
           $X \leftarrow \emptyset$   (X will be a set of crossings)
           $j \leftarrow 1$
Step 4.2   IF $MX(\alpha,\beta)_j = 2$, where $(\alpha,\beta)_j$ is the j-th pair of arcs
           belonging to $A_i$.
           THEN STEP 4.3
           $X \leftarrow X \cup x$      (where $x$ is a crossing of an arc of $\mathfrak{C}$
                                   and $\alpha$ or $\beta$, if any)
           IF $MX(\alpha,\beta)_j = 1$
           THEN $X \leftarrow X \cup (\alpha \times \beta)$
           IF $j = \binom{m}{2}$ where $m = \binom{n}{2} - n$
           THEN IF X corresponds to a drawing which is non-
                   isomorphic to each of the drawings
                   corresponding to the sets of crossings in $\mathfrak{D}$
                THEN $\mathfrak{D} \leftarrow \mathfrak{D} \cup X$
```

```
            STEP 4.3
       ELSE j ← j+1
            STEP 4.2
STEP 4.3  IF $i < r_1 \times r_2 \times \ldots \times r_m$ where $r_p$ is the number of
          good mappings of the edge $a_p$ of $K_n$
          THEN  i ← i +1
                STEP 4.1
          ELSE  STOP
```

Chapter 3

RESULTS OF THE ALGORITHM

A computer program is written to implement the algorithm presented in Chapter 2. This program is listed in Appendix A.1, and the drawings D_n having a C-F HC for $n \leq 6$ are presented in Appendix A.2, while in Appendix A.3 all the drawings D_7 are listed. In addition, results related to the n-circuit optimal and n-gon optimal drawings D_7 are obtained [19].

DRAWINGS D_5

By implementing the algorithm for $n = 5$, the number of drawings generated by the computer program is one hundred twelve (112) D_5, as per Table 3.1.

Number of Crossings	1	2	3	4	5
Number of D_5	20	0	70	0	22

Table 3.1: The number of drawings D_5 generated by the algorithm.

These 112 drawings yielded the five non-isomorphic drawings which are displayed in Fig.3.1. These drawings are essentially the same as the ones of Diagram 37 in [16].

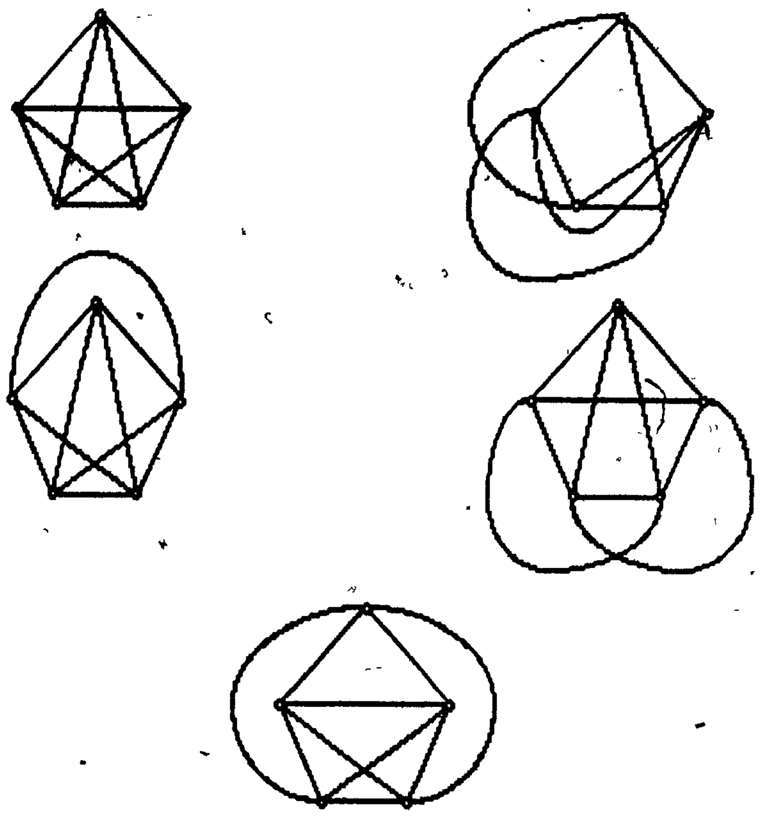
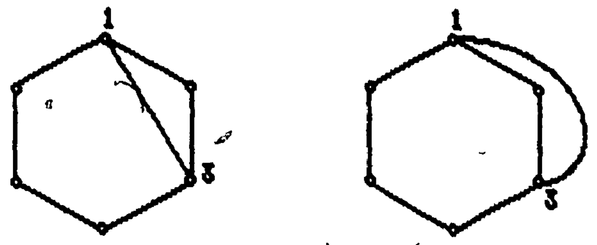


Fig.3.1: The five non-isomorphic drawings of K_5: two drawings with 5 crossings, two drawings with 3 crossings, and one drawing with 1 crossing.

DRAWINGS D_6

The number of good drawings D_6 of K_6 generated by the algorithm is fourteen thousand four hundred and sixty (14,460). However, we need only generate half of these drawings (7,230) since the drawings generated using the last four arcs (1,3) shown in Fig.2.1.b are isomorphic to the drawings generated using the first four arcs (1,3) as shown in Fig.3.2.



(a) (a')

Fig.3.2: All the drawings generated when arc (1,3) is as in (a') will be isomorphic to all the drawings generated when arc (1,3) is as in (a).

We get the one hundred and two (102) non-isomorphic drawings which are displayed in Appendix A.2. Most of these 102 drawings have already been found by Professor Uytterhœven and Mr. J. Backelin in the early seventies. These drawings are shown in their correspondence with Professor Guy [18].

Backelin's census consists of one hundred and twenty-three (123) drawings D_6. When the crossings of these drawings are input to a computer program to identify isomorphism, only ninety-six (96) are shown to be non-isomorphic. This discrepancy can most probably be explained by the fact that it is extremely difficult to detect instances of isomorphism between any two of the drawings visually.

Professor Uytterhœven's findings are presented in a Table 3.2 which can be compared to the computer's findings.

Number of Crossings	Number of Drawings found by	
	Uytterhœven	Computer Program
3	1	1
4	1	1
5	3	3
6	3	3
7	9	9
8	13	13
9	16	17
10	9	9
11	20	21
12	15	15
15	10	10
Number of D_6	100	102

Table 3.2: The number of drawings D_6 found by Professor Uytterhœven compared with those generated by the computer program.

Similarly, in Table 3.3, Backelin's figures can be compared to the program's results.

Number of Crossings	Number of Drawings found by	
	Backelin	Computer Program
3	1	1
4	1	1
5	3	3
6	3	3
7	9	9
8	14	13
9	19	17
10	14	9
11	25	21
12	19	15
15	15	10
Number of D_6	123	102

Table 3.3: The number of drawings obtained by Mr. Backelin and the number of drawings generated by the computer program.

Table 3.4 reflects the number of non-isomorphic drawings found by both Uytterhœven and Backelin.

Number of Crossings	Number of Drawings found by		
	Uytterhœven	Backelin	Computer Program
3	1	1	1
4	1	1	1
5	3	3	3
6	3	3	3
7	9	7	9
8	13	11	13
9	16	15	17
10	9	9	9
11	20	21	21
12	15	15	15
15	10	10	10
Number of D_6	100	96	102

Table 3.4: The number of non-isomorphic drawings D_6 as obtained by Professor Uytterhœven, Mr. Backelin and the computer program.

It should be noted that the computer program substantially confirms Professor Uytterhœven's results.

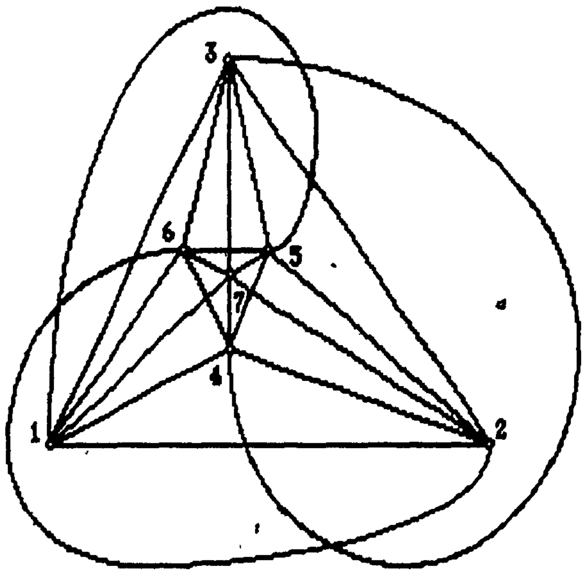
DRAWINGS D_7

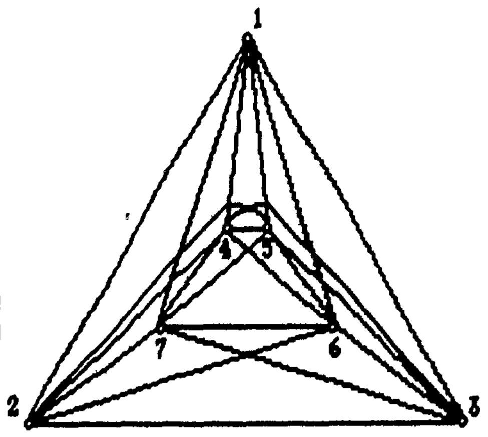
All the non-isomorphic drawings D_7 of K_7 are generated. Table 3.5 reflects the number of drawings D_7 distributed according to their number of crossings. Note that there are no drawings D_7 having an even number of crossings. All these drawings are listed in Appendix A.3.

Number of Crossings	Number of Drawings
9	5
11	27
13	103
15	363
17	937
19	1653
21	2259
23	2344
25	1769
27	1030
29	633
31	318
35	115
Number of D_7	11556

Table 3.5: The number of the non-isomorphic drawings D_7 having at least one C-F HC.

The C-F HC's are counted for each D_7, to find the ones with the largest number. The results obtained confirm Newborn and Moser's results related to optimal C-F HC drawings of K_7 [19].





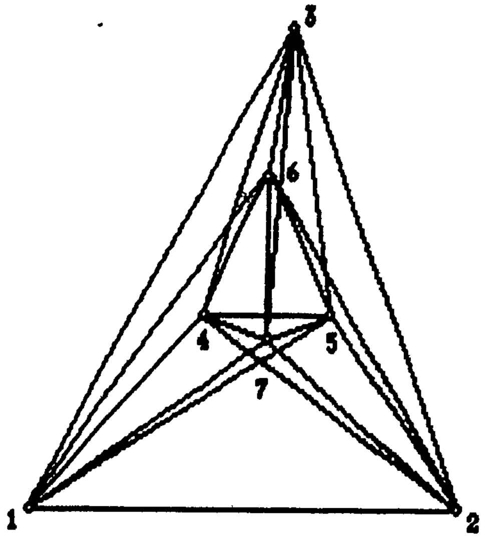


Fig.3.3: The first two drawings have the greatest number of 7-circuits in any drawing of K_7. The third one has the greatest number of 7-gons in any rectilinear drawing of K_7. These two numbers are 96 and 92 respectively.

Chapter 4
Analysis of the Algorithm

Before counting the number of operations that are performed by the algorithm, we first determine the number of arcs into which the edges of K_n could be mapped.

There are n edges of K_n which are mapped into n arcs to form a C-F HC, **C**. Consider the remaining $k = \binom{n}{2} - n$ edges (α, β) of K_n. Any arc (a, b) which is a mapping of (α, β) might cross i arcs of **C**, where $o \leq i \leq n-4$, as shown in Fig.4.1.

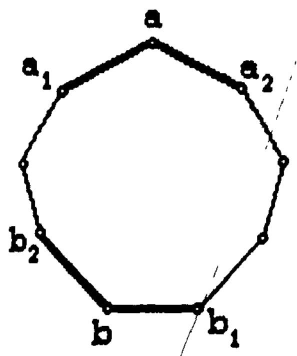


Fig.4.1: A C-F HC, C. The arc (a,b) might cross any arcs of C except (a,a_1), (a,a_2), (b,b_1) and (b,b_2).

Suppose (a, b) crosses some of the arcs of **C**. From Fig.4.2 we can see that the arcs of **C** and (a, b) can be redrawn without changing the crossings occurring between (a, b) and the arcs of **C**, or the order of these crossings.

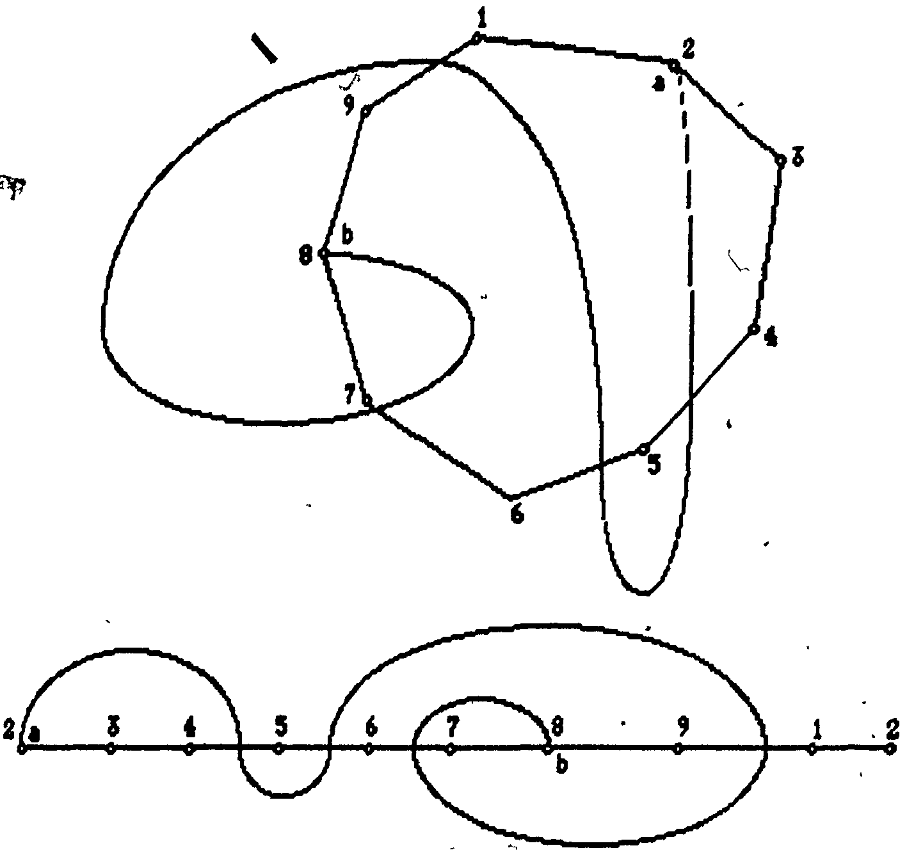


Fig.4.2: Two drawings of the arcs of $\mathbf{C}$ and an arc (a,b). The crossings and their order did not change from one drawing to the other.

Consider the segments of the arc (a,b) falling above the arcs of $\mathbf{C}$. Put an open parenthesis above each of the arcs which meet a left end of these segments and put a closed parenthesis above each of the arcs which meet a right end of these segments. Now, consider the segments of (a,b) which are below the arcs of $\mathbf{C}$, and in a similar manner open and closed parentheses are placed below the arcs of $\mathbf{C}$ as shown in Fig.4.3. Now, we note that for each of the arcs c of $\mathbf{C}$, we have one of the following five possibilities displayed in Fig.4.4.

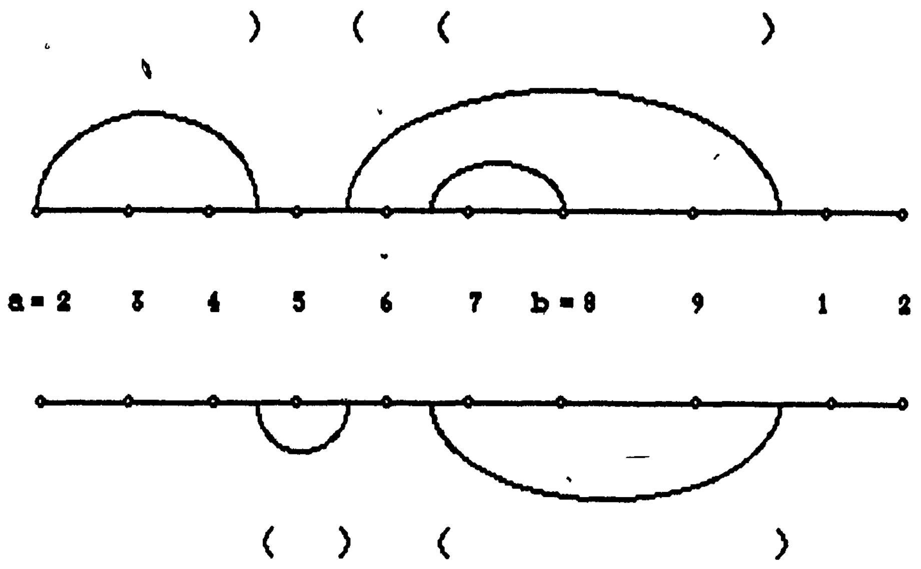


Fig.4.3: A parenthesis is placed above and below each of the arcs of $\mathbf{C}$ which are crossed by (a,b).

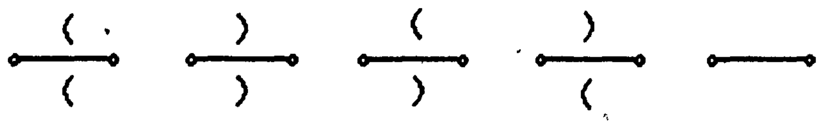


Fig.4.4: Any arc of $\mathbf{C}$ will have exactly two parentheses or no parentheses at all.

Let p be the number of good arcs (a, b) into which an edge (α, β) could be mapped, then

$$p \leq 5^{n-4}+1$$

Now we obtain a lower bound for p. Let α, β be two vertices of K_n, and let a, b be their corresponding nodes. Let (a, a_1) and (b, b_1) be two arcs of $\mathbf{C}$. If $a_1 = b_1$, then the number of mappings of (α, β), will be as small as possible.

The edge (α, β) might cross up to m=n−4 of the arcs of $\mathbf{C}$ which we draw as in Fig.4.5. The nodes are labeled above $\mathbf{C}$; while the m arcs of $\mathbf{C}$ are labeled below $\mathbf{C}$.

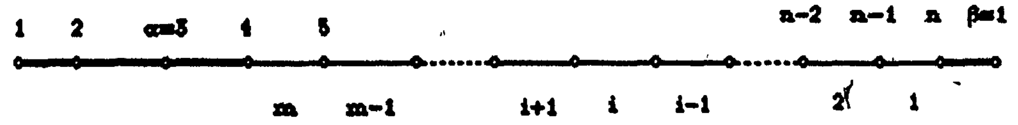


Fig.4.5: The heavy arcs cannot be crossed by (α,β).

Starting from $\alpha = 3$ going to $\beta = 1$, suppose the first arc of **C** to be crossed by (α,β) is arc i, then there are $i-1$ arcs on the right side of arc i. Each of these may or may not be crossed by (α,β) leading to 2^{i-1} possible combinations of arcs to be crossed, where $1 \le i \le m$. On the left side of i there are m−i arcs. The edge (α,β) may cross only an even number of these arcs, hence there are 2^{m-i-1} possible combinations of arcs that (α,β) may cross where $1 \le i \le m-1$. The total number of arcs into which (α,β) could be mapped is then at least

$$2^{m-1} + \sum_{i=1}^{m-1} 2^{i-1} \bullet 2^{m-i-1} = (m+1) \bullet 2^{m-2}$$

To this, the arc which does not cross any of the m arcs, is added; then the total is multiplied by 2 since arc i could be crossed from either side of **C**. Hence a lower bound for p is

$$2 + (m+1) \bullet 2^{m-1}$$

Therefore, lower and upper bounds for the number of good arcs into which an edge (α,β) could be mapped are given by the following inequalities:

$$\mathbf{2 + (n-3) \bullet 2^{n-5} \le p \le 5^{n-4}+1}$$

The following table reflects the number of arcs into which each of the m edges of K_n could be mapped without violating the rules of good drawings for $3 < n \le 7$.

Edges \ n	4	5	6	7
(1,3)	2	4	8	18
(2,4)	2	4	8	18
(3,5)	-	4	8	18
(4,6)	-	-	8	18
(5,7)	-	-	-	18
(6,1)	-	-	-	18
(7,2)	-	-	-	18
(1,4)	-	4	10	24
(2,5)	-	4	10	24
(3,6)	-	-	10	24
(4,7)	-	-	-	24
(5,1)	-	-	8	24
(6,2)	-	-	8	24
(7,3)	-	-	-	24

Table 4.1: The number of the possible good mappings for each of the m edges of K_n. The C-F HC $(1,2,3\ldots,n,1)$ is assumed in all cases.

The number of drawings that the algorithm generates is bounded by the product of the number of good arcs into which each of the k edges of K_n could be mapped. However this bound is extremely high, since the larger number of drawings will not be good drawings, as reflected in Table 4.2. This is due, obviously, to the fact that many arcs when considered in pairs yield drawings which are not good drawings.

n	Product of the number of arcs	Generated good drawings
5	$4^5 = 1024$	112
6	$8^6 \times 10^3 = 262144000$	14460

Table 4.2: The number of good drawings generated by the algorithm is small with respect to the product of the numbers of arcs.

Now, we calculate the number of operations taken by each of the steps of the algorithm.

Step#	Step Description	Number of Operations
1	Generate a list of the m edges $a_1, a_2, \ldots, a_p, \ldots, a_m$ of K_n where a_p is not an edge of $\mathbf{C}$ and $m=\binom{n}{2}-n$.	$O(n^2)$
2	For each a_p, determine its corresponding good arcs according to condition (i).	$O(n^4 \bullet 5^n)$
3	For each pair of arcs α and β, determine whether they cross, and whether any of the conditions (ii) and (iii) is met. The existence or non-existence of such crossings is reflected in a matrix $MX(\cdot,\cdot)$, where	$O(n^2 \bullet 5^{2n})$

$$MX(\alpha,\beta) = \begin{cases} 0 & \text{if arcs } \alpha \text{ and } \beta \text{ do not cross} \\ 1 & \text{if arcs } \alpha \text{ and } \beta \text{ cross such that the sub-drawing } \mathbf{C} \cup \alpha \cup \beta \text{ is a good drawing} \\ 2 & \text{otherwise} \end{cases}$$

```
4.0     $i \leftarrow 1$
        $\mathfrak{D} \leftarrow \emptyset$ ($\mathfrak{D}$ will be composed of the
        sets of crossings corresponding to
        the non-isomorphic drawings $D_n$)

4.1     Form a set $A_i$ where $A_i$ consists
        of m arcs, each of which is a
        mapping of one of the m edges of
        $K_n$ different from the edges of $\mathfrak{C}$.
        $X \leftarrow \emptyset$ (X is a set of crossings)
        $j \leftarrow 1$

4.2     IF $MX(\alpha,\beta)_j = 2$, where $(\alpha,\beta)_j$ is
        the j-th pair of arcs belonging to $A_i$
        THEN STEP 4.3
        $X \leftarrow X \cup \mathbf{x}$ (where $\mathbf{x}$ is a crossing of
        an arc of $\mathfrak{C}$ and $\alpha$ or $\beta$, if any)
        IF $MX(\alpha,\beta)_j = 1$
        THEN $X \leftarrow X \cup (\alpha \times \beta)$
        IF $j = \binom{m}{2}$ where $m = \binom{n}{2} - n$
        THEN

4.2.1   Obtain nodes responsibility          $O(n^5)$

4.2.2   Obtain arcs responsibility           $O(n^6)$

4.2.3   Determine whether $D_n$ is           $O(pn!)$
        isomorphic to any of the             where p is the
        drawings in $\mathfrak{D}$                     number of non-
                                             isomorphic
                                             drawings in $\mathfrak{D}$
                                             which have the
                                             same nodes and
                                             arcs responsibilities
                                             as $D_n$
```

```
4.2.4      Store $D_n$ if it is non-isomorphic      $O(n^4)$
           to each of the stored drawings
           ELSE  $j \leftarrow j+1$
                 STEP 4.2

4.3        IF $i = r_1 \times r_2 \times \ldots \times r_m$ where $r_p$ is the
           number of good mappings of an edge
           $\alpha$ of $K_n$
           THEN  $i \leftarrow i+1$
                 STEP 4.1
           ELSE  STOP
```

An upper bound on the number of times Step 4.1 is performed is

$$O(5^{n^3})$$

For each of these, Step 4.2 is performed at most $O(n^2)$ times. The number of operations shown for Step 2 is also an upper bound for this step. Neither of these two bounds is realized for the reasons previously given; see Table 4.2. Indeed, determining whether the drawing in hand is isomorphic to any of the previously obtained drawings takes the bulk of the algorithm's time. This is due to the fact that the nodes in many drawings have equal responsibilities, in addition to the fact that arcs responsibilities are equal in many drawings, as shown in Table 4.3.

Number of Crossings	Number of D_6	Largest number of nodes having equal responsibilities
3	58	6
4	288	5
5	492	4
	48	5
6	652	3
7	1638	4
8	1176	3
	1956	5
9	1632	3
	716	6
10	1320	5
11	2358	4
12	1680	3
15	446	6

Table 4.3: The largest number of nodes with equal responsibility is six for all the drawings having 15 crossings. For many of these drawings, $k \times 6!$ comparisons are required to determine isomorphism, where $k \leq 10$.

To obtain the 102 drawings of K_6, an IBM microcomputer model AT ran for about three days. Since, in addition, the product of the good arcs into which the 14 arcs of K_7 could be mapped can be calculated to be $18^7 \times 24^7$; it becomes clear that obtaining all the drawings of K_7, using the same computer, would require so long as to be impractical. The SUN computer, model 3/280S at McGill Computer Science School was used for n=7. Although it performs 4×10^6 instructions/second, it required about 18 days to produce the 11,556 non-isomorphic drawings D_7.

Chapter 5

AN ALGORITHM TO DETERMINE WHETHER THERE EXISTS A RECTILINEAR DRAWING D_n OF K_n HAVING A GIVEN SET OF CROSSINGS

Background

A necessary and sufficient condition for some graphs to be drawn rectilinearly is given by Eggleton[16]. In this chapter we present some propositions and theorems by which we show that given a set of crossings of K_n, it is possible to determine whether there exists a rectilinear drawing D_n which has exactly this set of crossings.

First, the necessary definitions and explanation of the terminology that we are using are given.

A *trigon* T of a drawing D_n of K_n consists of three nodes α, β, γ and three arcs (α,β), (α,γ), (β,γ) of D_n. Obviously any D_n has $\binom{n}{3}$ trigons. The *contents* of a trigon T refer to the nodes contained in the area bounded by the arcs of T. Except for the vertices of T, we say that a node υ is *contained* in T whenever υ is inside T, and we write $\upsilon \in \mathbf{Int}\ T$. If υ is not contained in T, we write $\upsilon \in \mathbf{Ext}\ T$. Two drawings of K_n are *equivalent* if there is a one-to-one correspondence between their nodes and their trigons such that if a node υ is inside a trigon T in one drawing, then, in the other drawing, the node corresponding to υ is inside the trigon corresponding to T. Counter examples are given in Fig. 5.0.1. Finally, if a set of arcs, segments of arcs and nodes of D_n form a boundary, B, such that all the remaining nodes and arcs fall in the interior of B, then we call B the *outer boundary* of D_n.

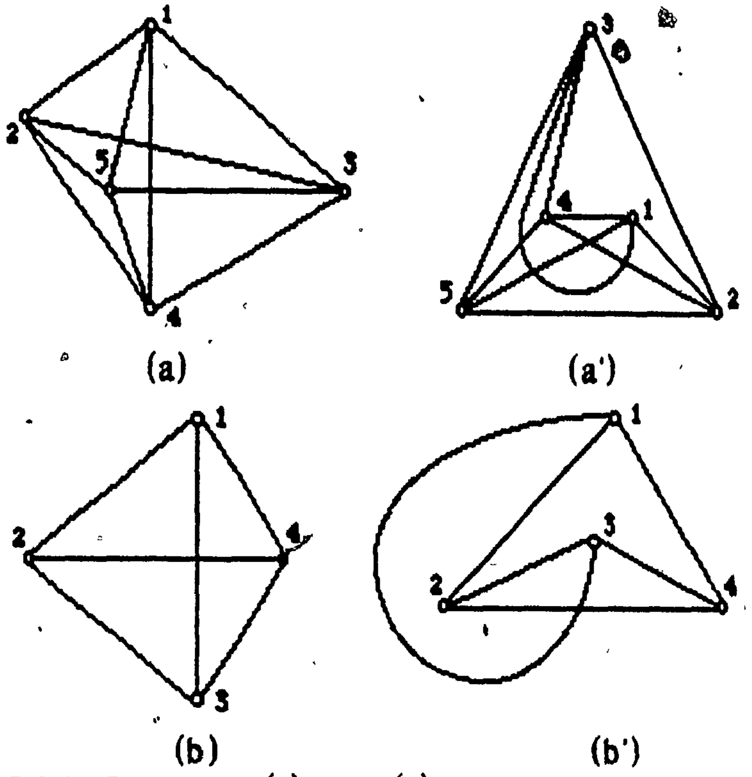
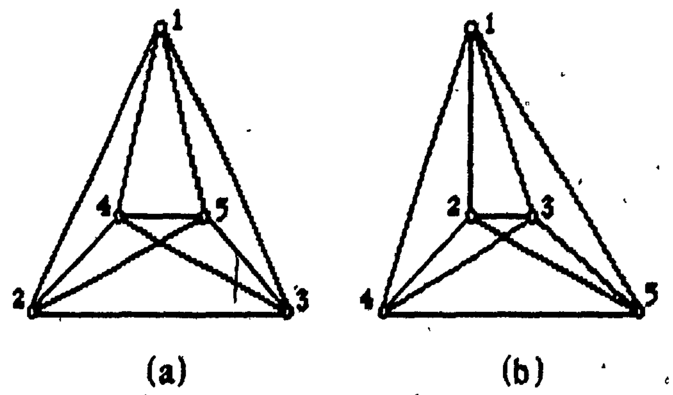
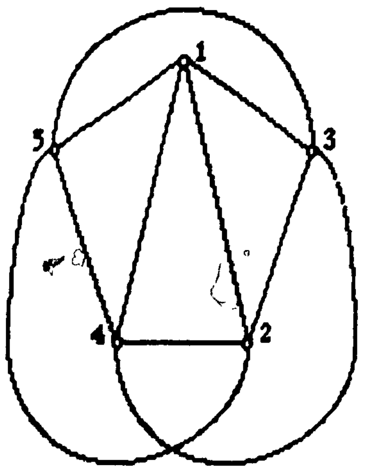


Fig.5.0.1: Drawings (a) and (b) are isomorphic but non-equivalent to drawings (a') and (b') respectively.

More than one drawing D_n of K_n may have the same set of crossings. Some of these will be equivalent, while others will be non-equivalent, as shown in Fig.5.0.2.





(c)
Fig.5.0.2: Three drawings (a), (b) and (c) having one crossing (2,5) × (3,4). Only (a) and (b) are equivalent.

If the outer boundary B of D_n does not have any crossed arcs, as in Fig.5.0.3(b), then B is a *convex hull* of D_n and we denote it by *CH*. On the other hand if any of these arcs is crossed then B is not a convex hull of D_n, as in Fig.5 0.3(a).

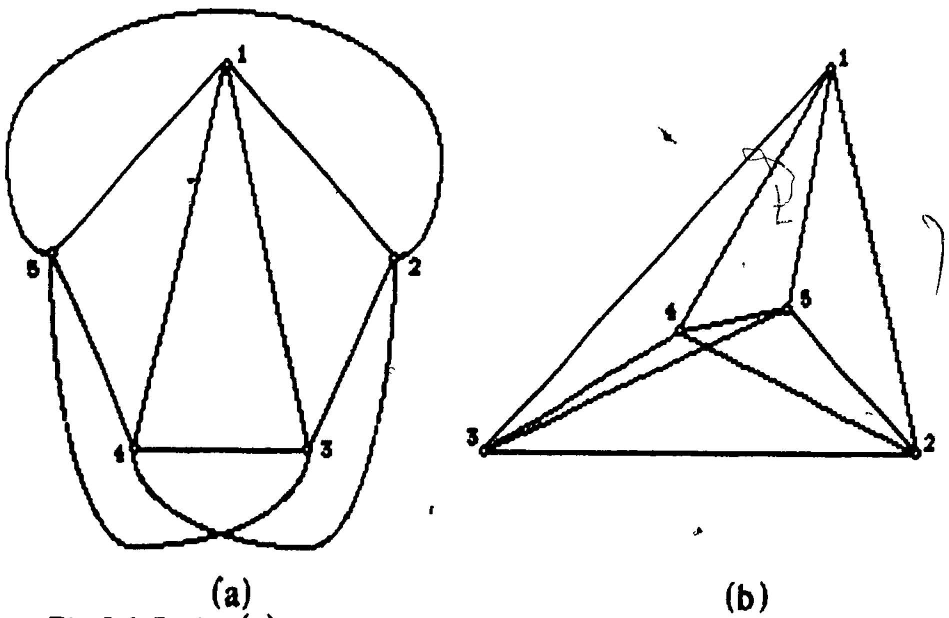


Fig.5.0.3: In (a) the outer boundary of D_5 is not a convex hull, while in (b) the outer boundary of D_5 is a CH.

Let $C_1, C_2, \ldots, C_p, \ldots, C_k$ be the CH's of the set of drawings D_n of K_n and let us draw a drawing D_n having the CH C_p.

Consider the arcs (α,β), (α,γ), (β,γ) and (α,δ) of a drawing D_n. We say that (α,β) and (α,γ) are *adjacent with respect to (α,δ)* if, and only if, neither (α,δ) nor a segment (α,δ_0) of (α,δ) falls in the area T bounded by (α,β), (α,γ) and (β,γ). If (α,δ) or (α,δ_0) is in T we say that (α,δ) is *located between* (α,β) and (α,γ), as shown in the drawings (c) and (d) in Fig.5.0.4.

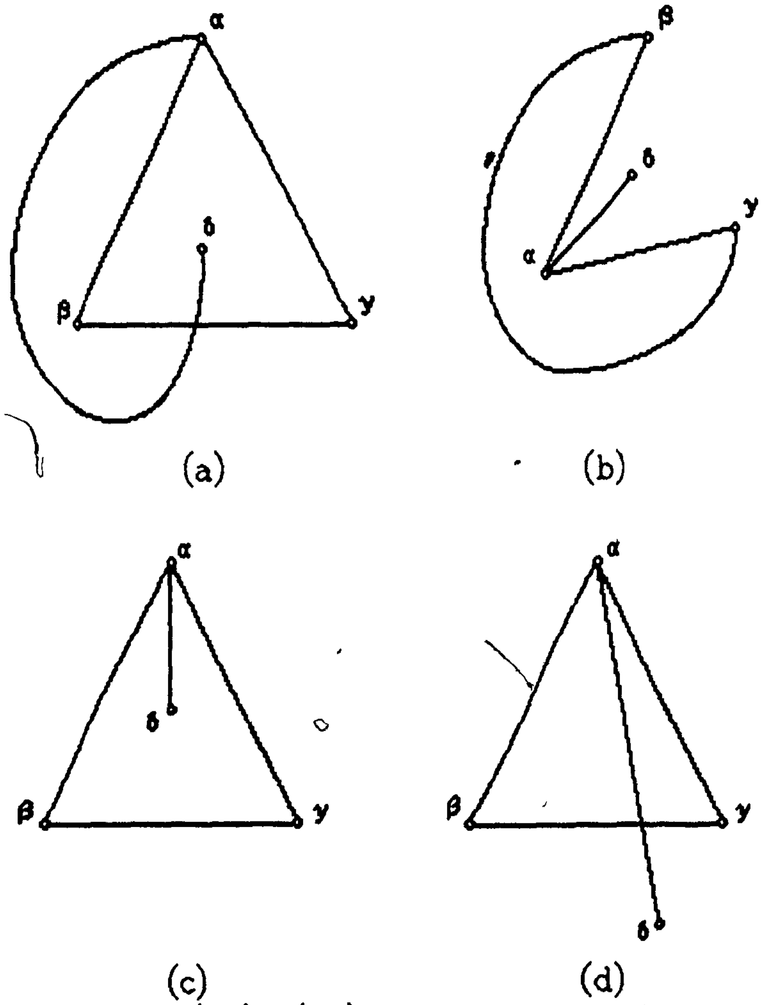


Fig.5.0.4: Arcs (α,β), (α,γ) are adjacent with respect to (α,δ) in (a) and (b) only. In (c) and (d) (α,δ) is between (α,β) and (α,γ).

With the above definitions, the solution to the problem of determining whether K_n can be drawn rectilinearly with a given set of crossings is presented in this chapter. First we show that there exists a drawing, A, shown in Fig.5.0.5, which is always a sub-drawing of any non-rectilinear drawing D_n. We also show that the drawing A cannot be a sub-drawing of a rectilinear drawing D_n.

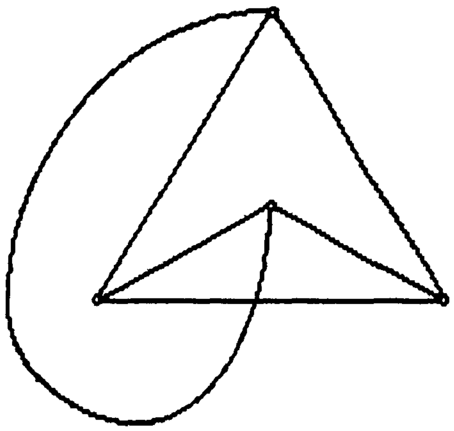


Fig.5.0.5: The drawing A.

The drawing A is a good drawing D_4 of K_4, drawn such that the area bounded by a trigon T, consisting of three nodes a_i of D_4 and their corresponding three arcs, contains the fourth node υ of D_4, and such that one of the arcs (υ, a_i) crosses an arc of T.

To determine whether A is a sub-drawing of D_n, we must know the content of each trigon T of D_n, in other words, for each T we must know whether υ is contained in T, where υ is any node of D_n. This matter is resolved when we show that, given a set of crossings of D_n along with its CH, we can determine whether an arbitrary node of D_n falls in an arbitrary trigon of D_n.

In the above, we have assumed that a CH of D_n is given along with a set of crossings of K_n. However, we want to be able to determine whether K_n can be drawn rectilinearly just by being given a set of its crossings. For this purpose we produce a proposition showing that, given a set of crossings of K_n, we can determine all of the CH's of the drawings D_n of K_n having this set of crossings.

Finally an algorithm is developed. With a set of crossings of K_n as input, this algorithm determines whether there exists a rectilinear drawing D_n of K_n having exactly this set of crossings.

A Characteristic of the Rectilinear Drawings

The complete graph K_4 has exactly three non-equivalent good drawings D_4, as shown in Fig.5.1.1.

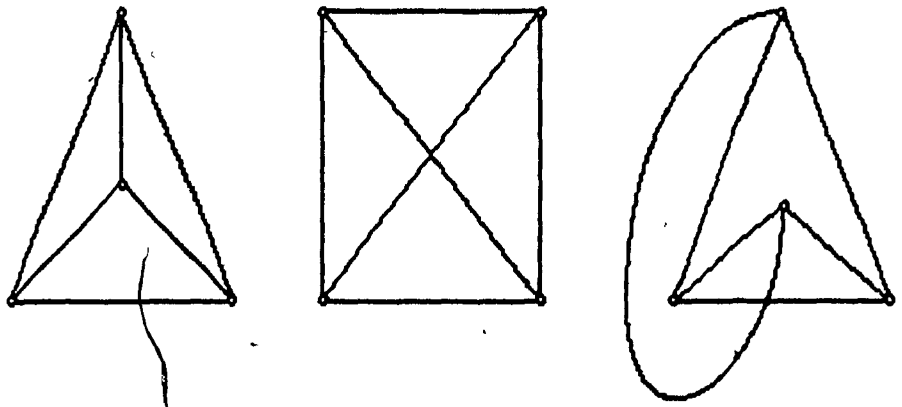


Fig.5.1.1: The three non-equivalent drawings of K_4.

The third of these, the drawing A, is of great importance and is used extensively in this chapter. We refer to the second drawing by X This section is actually developed to show that A is always a sub-drawing of any non-rectilinear drawing D_n.

We prove that given the crossings of D_n and its convex hull C, it is possible to determine whether the arcs of D_n can be realized by straight line segments. Essentially, we prove in the following theorem that D_n can be realized by straight line segments; if, and only if, A is not a sub-drawing of D_n.

Before proceeding with the theorem, we present three propositions. In the first one it is shown that if A is a sub-drawing of D_n, by redrawing the arcs constituting A such that they form a sub-drawing equivalent to X instead of A, then the new drawing D_n will have a CH, C which will necessarily be different from C. The second proposition is a generalization of the first, whereby we show that by determining the CH of D_n, the containment of each node of D_n is also determined with respect to each trigon of D_n. Finally, in the third proposition, given a triangle $T = (\alpha, \beta, \gamma, \alpha)$ containing some arcs (α, x_i), (α, y_i) and (x_i, y_i), all of which being straight line segments, we show that an arc (α, δ) can be represented by a straight line segment if (α, δ) is located between each pair of arcs (α, x_i), (α, y_i).

Each of these propositions is important in proving the theorem.

Proposition 5.1

Let D_n be a drawing of K_n, C be the CH of D_n, and A be a sub-drawing of D_n. Denote the nodes of A by 1, 2, 3 and 4.

If D'_n is a drawing isomorphic to D_n in which the sub-drawing consisting of the nodes 1, 2, 3, 4 and their corresponding arcs is equivalent to an X drawing instead of an A drawing; then C cannot be the CH of D'_n.

Proof

Let υ be a node of C, as in Fig.5.1.2 (a),(b). The arc $(\upsilon,4)$ crosses either $(1,2)$ or $(2,3)$ but not both. When K_n is redrawn such that the nodes 1, 2, 3, 4 and their corresponding arcs form a sub-drawing equivalent to X, and in order to maintain the crossing $(\upsilon,4) \times (1,2)$ or $(\upsilon,4) \times (2,3)$, node υ must fall inside the area bounded by the arcs (1,2), (2,4), (4,3) and (3,1); this leads to a CH different from C.

□

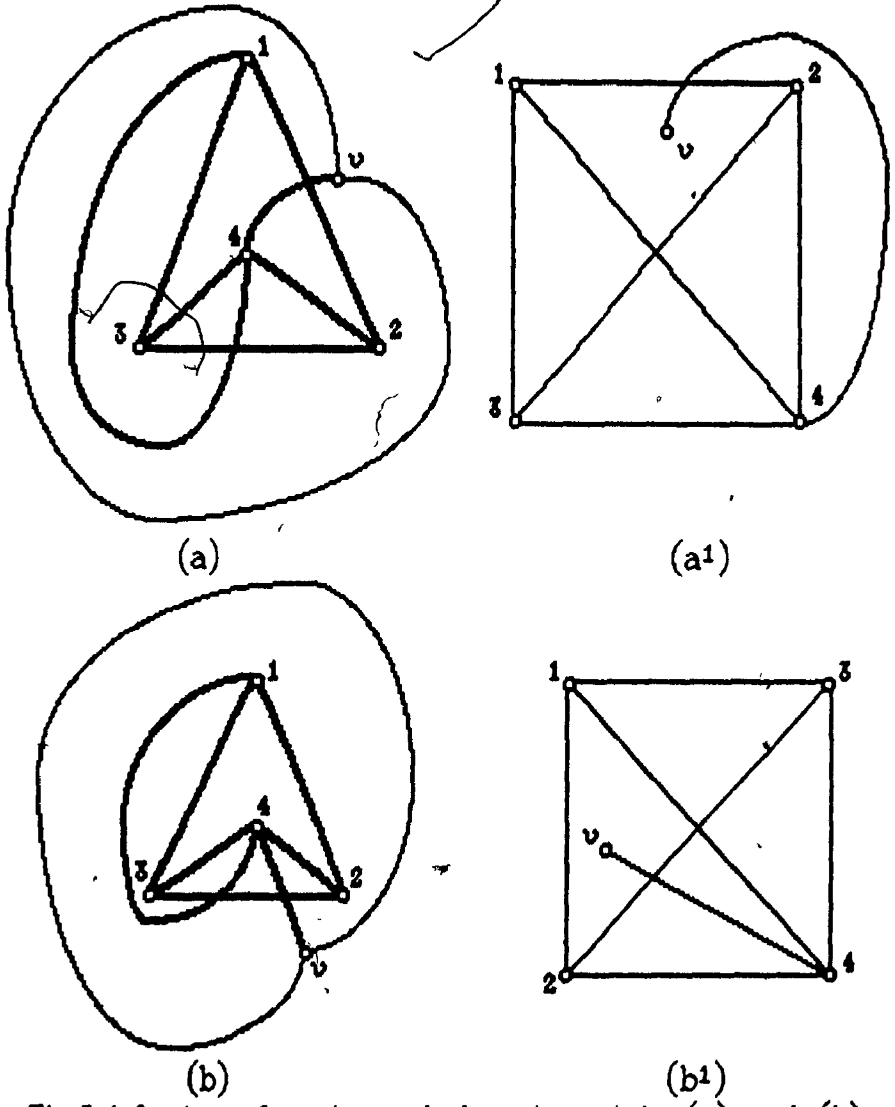


Fig.5.1.2: Arcs forming sub-drawings A in (a) and (b). The same arcs are redrawn in (a^1) and (b^1) to form sub-drawings X.

In Fig.5.1.3 - 5.1.5. we present some examples illustrating the existence of a sub-drawing equivalent to the drawing A in every non-rectilinear drawing D_n. For the purposes of this chapter, a drawing is considered rectilinear whenever its arcs are restricted to straight line segments while preserving its CH. The two drawings in Fig.5.1.3(i) are equivalent; only the second one is rectilinear. The drawing in Fig.5.1.3(ii) is non-rectilinear.

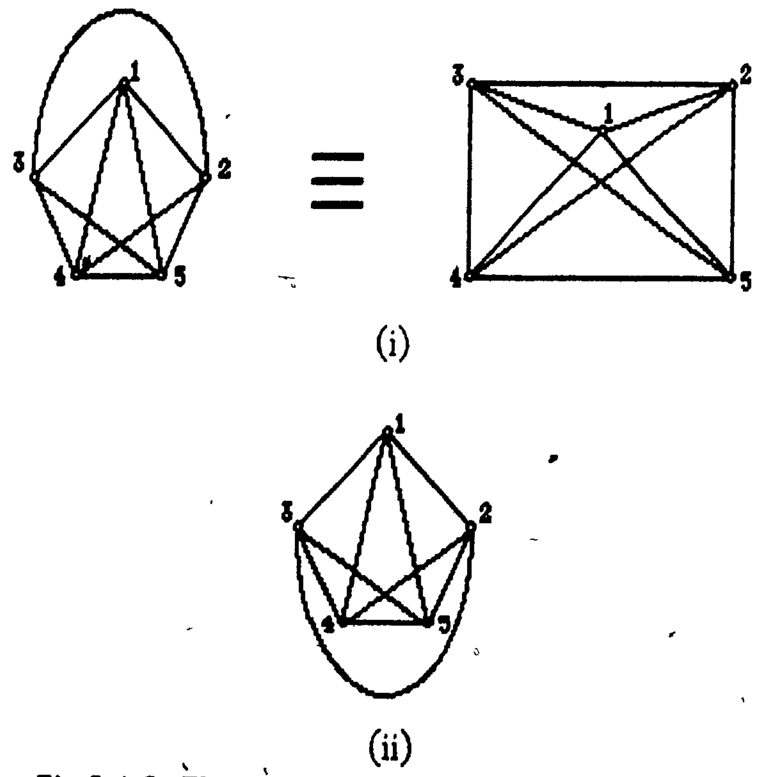
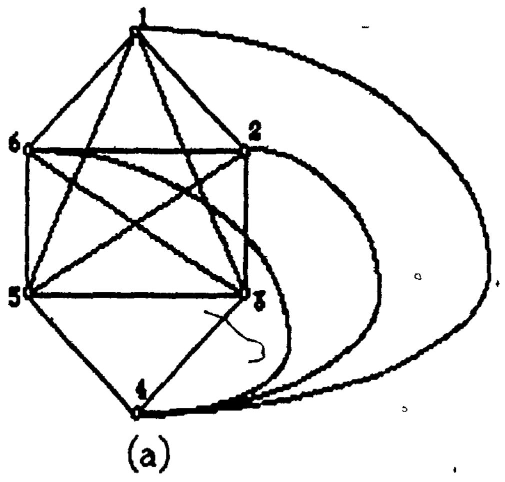
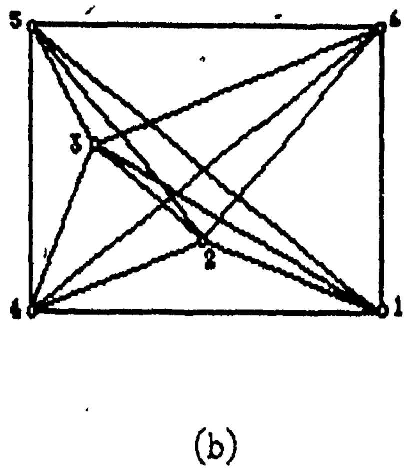


Fig.5.1.3: Three isomorphic drawings of K_5. In (i) the first drawing can be realized using strictly straight line segments to look like the second drawing with the same CH. In (ii), (2,4) × (3,5) while 4 is inside the trigon (2,3,5,2).





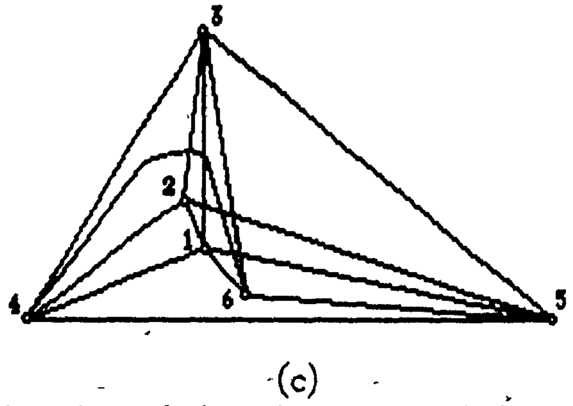


Fig.5.1.4: A is not a sub-drawing of any of the sub-drawings in (a) and (b). In (c) A is a sub-drawing of D_6, (4,6) × (2,5) while node 6 is inside the trigon (2,4,5,4).

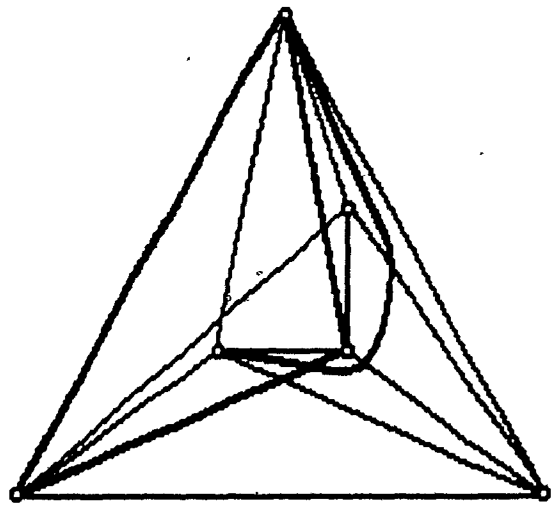


Fig.5.1.5: The drawing A, in heavy lines, is a sub-drawing of a drawing D_6 of K_6.

Proposition 5.2

Let T be a triangle with vertices α, β, and γ. Let S be a drawing consisting of T, some arcs (α, x_i), (α, y_i), (x_i, y_i) and one arc (α, δ) located in T, as shown in Fig.5.1.6. Suppose that (α, x_i), (α, y_i) and (x_i, y_i) are straight line segments. If (α, δ) is located between each pair (α, x_i), (α, y_i), then (α, δ) can be realized by a straight line segment.

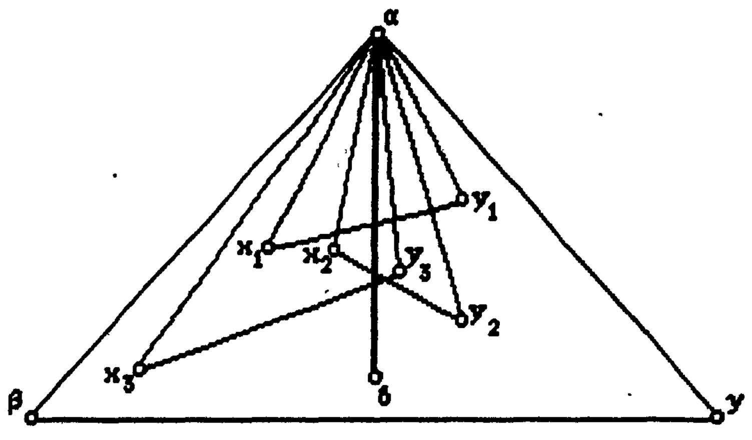


Fig.5.1.6: A drawing consisting of a triangle and arcs (α, x_i), (α, y_i) and (x_i, y_i). All arcs (α, x_i), (α, y_i) and (x_i, y_i) are straight line segments.

Proof

For convenience we assume that all (α, x_i) are on one side of (α, δ), and all (α, y_i) are on the other side of (α, δ). Among all (α, x_i), let (α, x_s) be the closest arc from (α, δ); and among all (α, y_i), let (α, y_s) be the closest arc from (α, δ). We extend (α, x_s) and (α, y_s) to meet (β, γ) at the points β_0 and γ_0, as shown in Fig.5.1.7. The interior of the triangle with vertices $\alpha, \beta_0, \gamma_0$ contains nothing but a segment of each of the arcs (x_i, y_i) the arc (α, δ) and the node δ, hence (α, δ) can be realized rectilinearly.

□

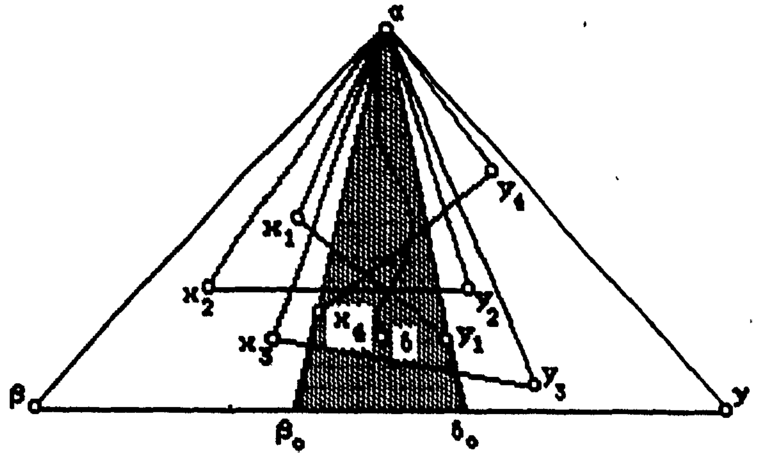


Fig.5.1.7: The shaded triangle contains no nodes except δ. The arc (α,δ) can always be realized by a straight line segment.

Theorem 5.3

Let D_n be a good drawing of K_n. D_n is non-rectilinear $\Leftrightarrow$ A is a sub-drawing of D_n.

Proof

(i) First we prove that:

A is a sub-drawing of D_n $\Rightarrow$ D_n is non-rectilinear.

All the arcs of A cannot be realized rectilinearly unless they are re-drawn to form a sub-drawing equivalent to X instead of A; obtaining a drawing D'_n. By Proposition 5.1, $D_n \neq D'_n$.

(ii) Secondly, we prove that:

D_n is non-rectilinear $\Rightarrow$ A is a sub-drawing of D_n.

Let C be the CH of D_n and let x_i denote the nodes of C. Let the arcs (x_i, x_j) be straight line segments, as in Fig.5.1.8. If any arc (x_p, x_q) is not a straight line, then by *pulling* x_p and x_q in the appropriate directions, (x_p, x_q) will become a straight line segment without affecting any of the crossings of D_n, as shown in Fig.5.1.8.0. Of course, some arcs (x_i, x_j) will be *pushing* and *pulling* some of the other arcs, however the crossings will be maintained.

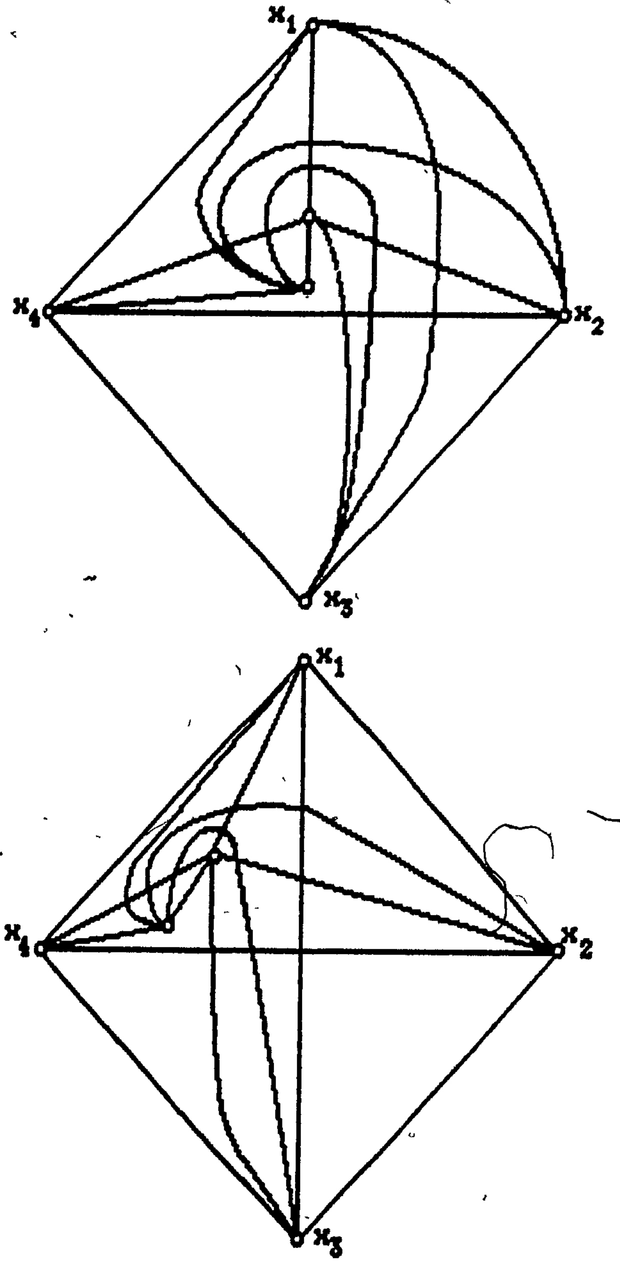


Fig.5.1.8.0: Two equivalent drawings D_6, all the arcs (x_i, x_j) of the second one are straight line segments.

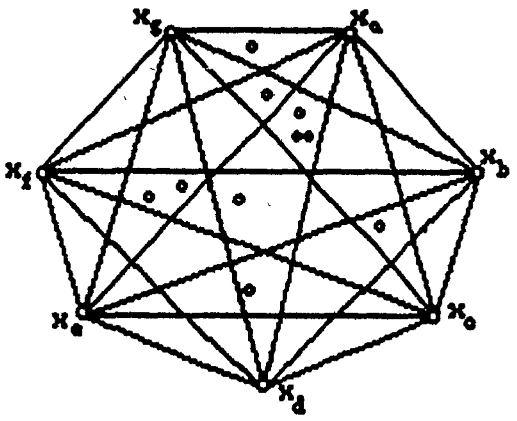


Fig.5.1.8: The arcs (x_i, x_j) of a drawing D_n realized using straight line segments.

Hence, any arc which we cannot realize by a straight line segment has either:

(1) a node x_o belonging to C and a node v_o located in the interior of C, as shown in Fig.5.1.9.

or

(2) two nodes located in the interior of C, as shown in Fig.5.1.10.

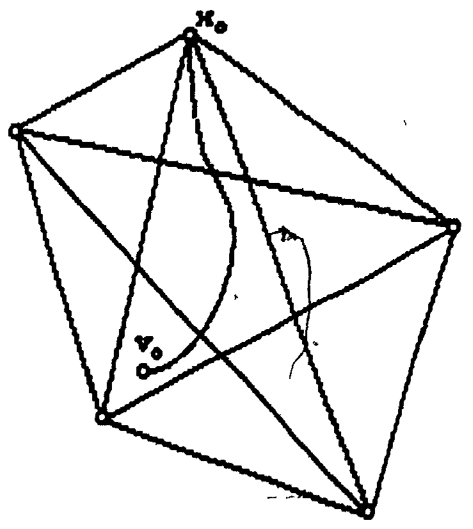


Fig.5.1.9: The arc (x_o, v_o) has only one node located inside C.

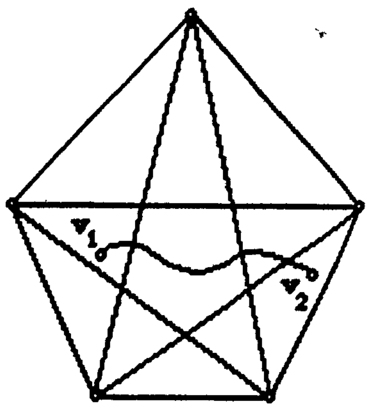


Fig.5.1.10: The arc (v_1, v_2) has its two nodes inside C.

Consider the first case. Arc (x_o, v_o) falls entirely in a triangle T with vertices x_0, x_1 and x_2. In order not to be able to represent (x_o, v_o) rectilinearly, (x_o, v_o) must be crossing some arcs (v_i, v_j).

Now consider the triangle T and all the arcs (x_o, v_i), as shown in Fig.5.1.11.

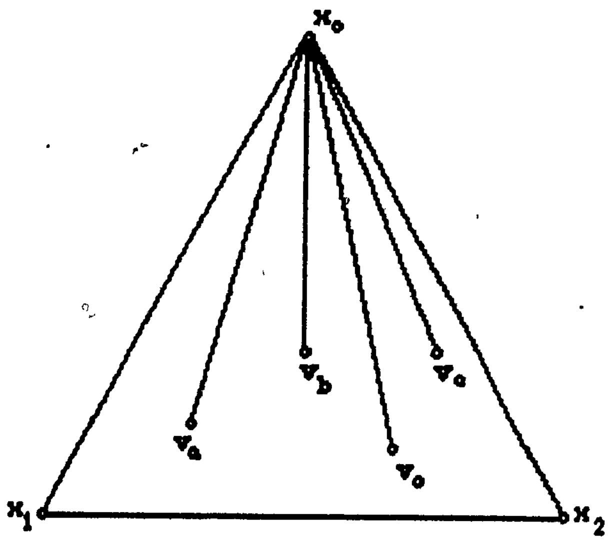


Fig.5.1.11: The triangle T, the arc (x_o, v_o) and the arcs (x_o, v_i).

By Proposition 5.2, there is a pair of arcs (x_o, v_a), (x_o, v_b) which is adjacent with respect to (x_o, v_o), where (v_a, v_b) is crossed by (x_o, v_o), as shown in Fig.5.1.12.

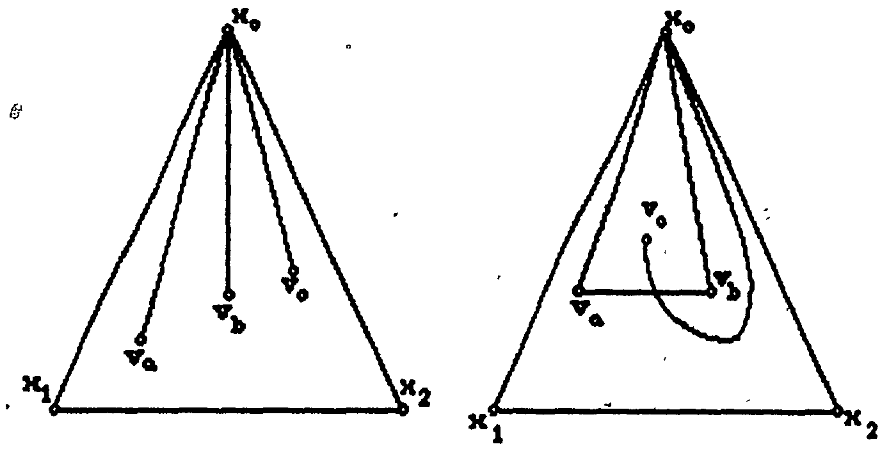


Fig.5.1.12: (x_o,v_a) and (x_o,v_b) are adjacent with respect to (x_o,v_0), and (x_o,v_0) crosses (v_a,v_b).

Now, by considering only T and the arcs (x_o,v_o), (x_o,v_a), (x_o,v_b) and (v_a,v_b), we see that A is a sub-drawing of D_n, as shown in Fig.5.1.12. Looking at the second case, we suppose that (v_a,v_b) cannot be drawn rectilinearly, as shown in Fig.5.1.13.

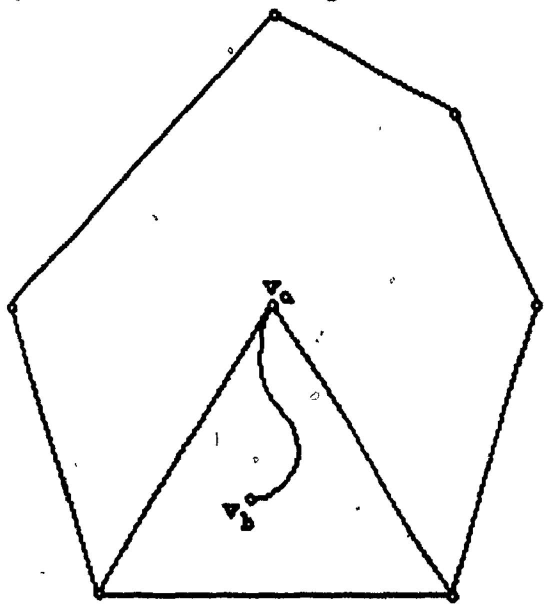


Fig.5.1.13: An arc (v_a,v_b), with its two nodes inside C, falls in the interior of a triangle having two nodes belonging to C.

The arc (v_a, v_b) is located in a trigon with vertices v_a and two nodes of C. By changing the label v_a to x_o and v_b to v_o we obtain a case similar to the first case. □

Determining the Location of a Node

In the preceding section we have concluded that the existence of a rectilinear drawing D_n depends on whether A is a sub-drawing of D_n. To be able to find out if A is a sub-drawing of a given drawing D_n, it is necessary then to know:

1. the crossings of D_n, and
2. the location of each of the nodes with respect to each of the trigons of D_n.

Here in a theorem, we show how the location of an arbitrary node υ of D_n with respect to an arbitrary triangle T of D_n, can be determined if the CH, C of D_n is known along with the crossings of D_n While in this section we assume that C is given, in the next section we demonstrate that all the CH's of the drawings D_n of K_n with a set of crossings can be obtained just by knowing these crossings.

First, we produce a proposition to be used in the proof of the above mentioned theorem.

Proposition 5.4

Let $\{1,2,3,\alpha,\beta\}$ be the set of nodes of a drawing D_5 of K_5. Let T denote the triangle formed by (1,2), (2,3) and (1,3). If the crossings of D_5 are given and if we know the location of α with respect to T, then we can determine the location of β with respect to T.

Proof

Assume that $\alpha \in \text{Int } T$.

$$\left.\begin{array}{l}(\alpha,\beta) \text{ crosses exactly one of} \\ \text{the arcs of } T \text{ or all its arcs,} \\ \text{Fig.5.2.1.}\end{array}\right\} \Leftrightarrow \beta \in \text{Ext } T$$

$$\left.\begin{array}{l}(\alpha,\beta) \text{ does not cross any of the} \\ \text{arcs of } T \text{ or crosses exactly two} \\ \text{of its arcs, Fig.5.2.2}\end{array}\right\} \Leftrightarrow \beta \in \text{Int } T$$

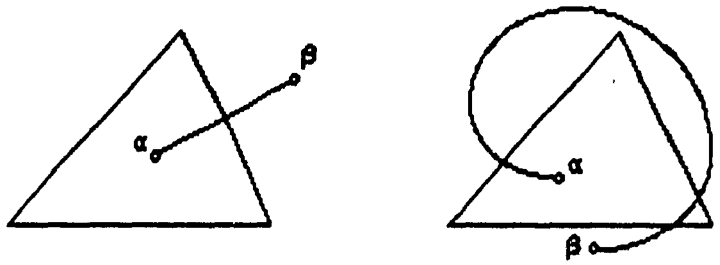


Fig.5.2.1: Node α is inside T and β is outside T.

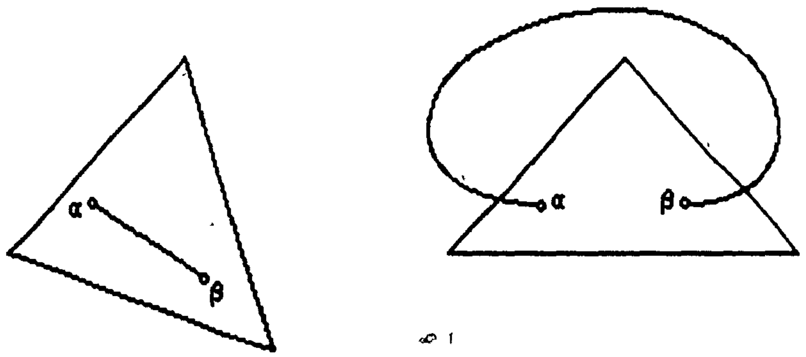


Fig.5.2.2: Nodes α and β are both inside T.

A similar argument is used when α is in Ext T.

□

From the above proposition it is obvious that in order to determine the location of a node α with respect to a triangle T, the location of a node β with respect to T has to be known. This does not represent an obstacle in locating the nodes of D_n as long as C is known. Any of the nodes of C can be used as a reference node β, since none of these is located in the interior of any of the triangles of D_n, as shown in Fig.5.2.3.

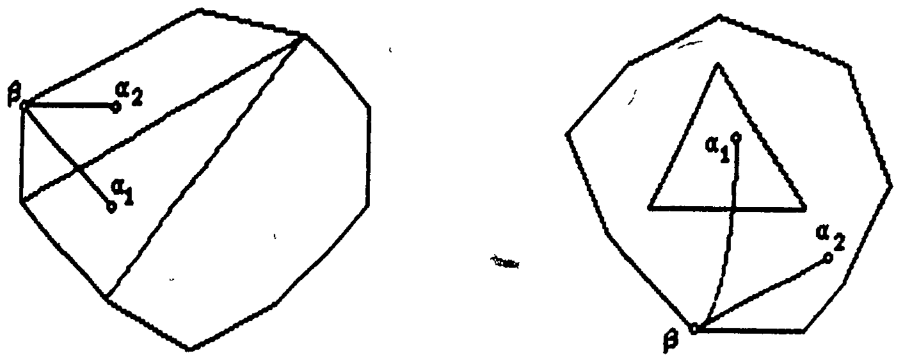


Fig.5.2.3: A node β belonging to the CH is used as a reference to determine the location of α_1 and α_2 with respect to a triangle.

Theorem 5.5

Let D_n be a drawing of K_n, and let C be the CH of D_n. Denote the nodes of C by x and the remaining nodes of D_n by v, as in Fig.5.2.4. Knowing the nodes of C and the crossings of D_n, we can determine whether a node v is contained in a trigon T of D_n.

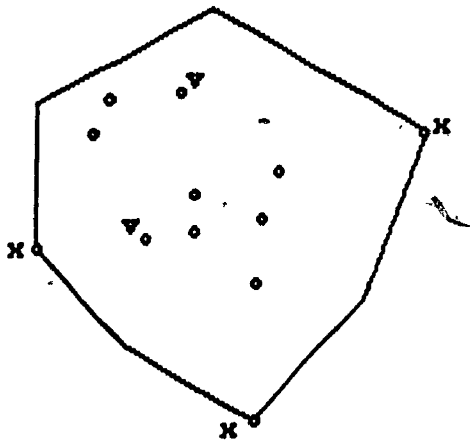


Fig.5.2.4: A CH, C with nodes x, and nodes v inside C.

Proof

The nodes of a trigon T of D_n can be

- three x's, as shown in Fig.5.2.5.a,
- three v's, as shown in Fig.5.2.5.b,
- two x's and one v, as shown in Fig.5.2.5.c,
- one x and two v's, as shown in Fig.5.2.5.d.

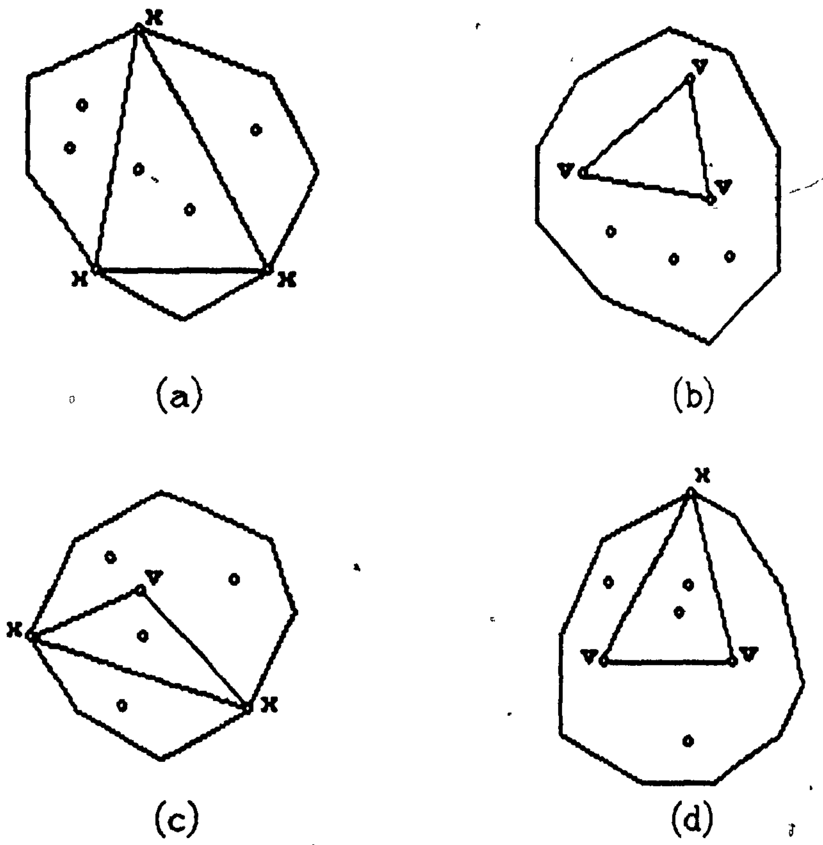


Fig.5.2.5: Trigons of D_n.

To determine whether $v \in$ Int T or $v \in$ Ext T, we consider, along with T and v, any node of the x's which does not belong to T, say x_o. We know that $x_o \in$ Ext T, hence by Proposition 5.4 we can determine whether v is inside T. □

Determining the Convex Hulls

More than one non-equivalent drawing D_n of K_n might share the same set of crossings X. In this section we show that the CH's of these drawings can be determined by examining this set of crossings X.

Suppose we are given an uncrossed C-F HC, C of a drawing D_k along with the crossings of D_k. The arcs of D_k, different from the arcs of C, might be on either side of C, as in Fig.5.3.1.

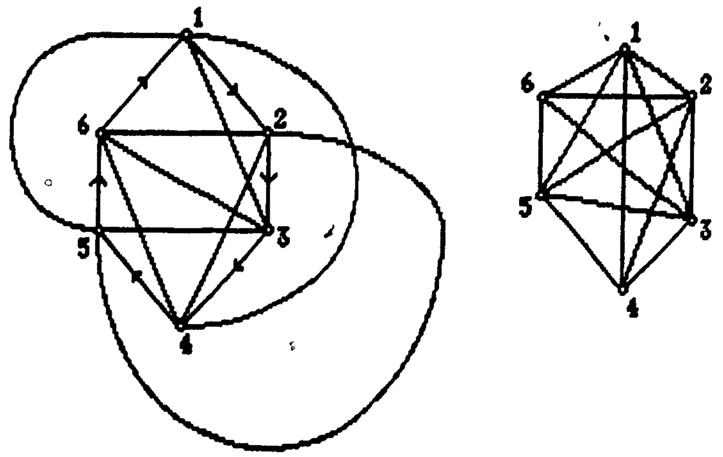


Fig.5.3.1: An uncrossed C-F HC = C = (1,2,3,4,5,6,1) of two drawings D_6. Only in the second drawing, all the arcs fall on one side of C, namely Int C.

If C is also a CH of D_k, then all the arcs, different from the arcs of C, must be in Int C. This happens only when the number of crossings of D_k is $\binom{k}{4}$. If k = 3, then all the arcs, different from the arcs of C, are on either side of C.

In the next proposition, given a CH, C of D_k and the crossings of D_n, we show how we can determine whether C is a CH of D_n.

Proposition 5.6

Let D_k be a sub-drawing of D_n, and C be an uncrossed CH of D_k. Let υ be a node of D_n but not of D_k and $k > 3$.

υ is in Int C if and only if there exists an arc (υ, r) which crosses (p, q) of D_k where p, q and r are nodes of C.

Proof

(i) First we prove that

$$\upsilon \in \text{Int } C \Rightarrow (\upsilon, r) \times (p, q)$$

The arc (p, q) divides D_k into two sub-drawings. One of them contains the node υ. Let r be on the convex hull of the other sub-drawing as shown in Fig. 5.3.2. The arcs of C are uncrossed, Hence $(\upsilon, r) \times (p, q)$.

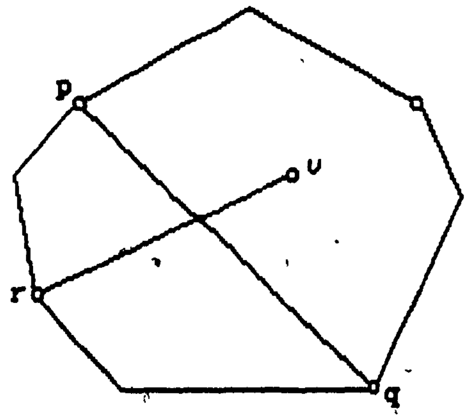


Fig. 5.3.2: $(\upsilon, r) \times (p, q)$ when υ is in Int C.

(ii) Now we have to show that

$$(\upsilon, r) \times (p, q) \Longrightarrow \upsilon \in \text{Int } C$$

Suppose υ is in Ext C. Since C is uncrossed then any arc (υ, i) must fall completely in Ext C, as in Fig. 5.3.3. Hence we can write:

$$\upsilon \in \text{Ext } C \Rightarrow \text{there exists no arc } (\upsilon, r) \text{ crossing } (p, q),$$

or equivalently:

$$\text{there exists an arc } (\upsilon, r) \text{ crossing } (p, q) \Rightarrow \upsilon \in \text{Int } C.$$

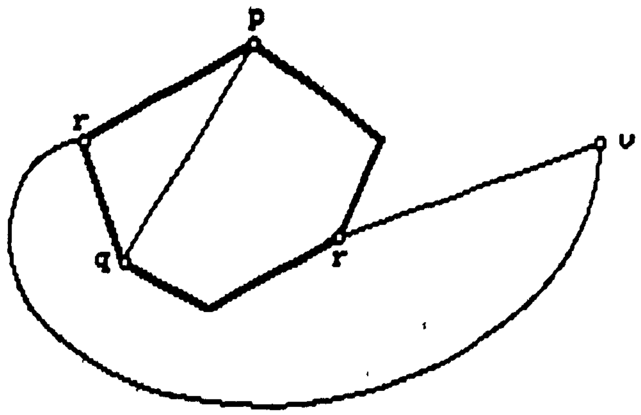


Fig.5.3.3: $\nu \in$ Ext C (C in heavy lines) implies that (ν, r) cannot cross any arc (p,q) where p, q and r are nodes of C.

□

The Algorithm

From the preceding sections we know that D_n is non-reetilinear if and only if A is a sub-drawing of D_n. Here we present an algorithm, which on being given the dimension n of K_n and a set of crossings X as its only input, determines whether K_n has a rectilinear drawing D_n with the set of crossings X.

The crossings are input to the algorithm, which first finds all the uncrossed k-circuits C_i of K_n using the Depth First Search method [14]. From these C_i's the algorithm determines and retains the ones which are convex hulls for drawings D_n of K_n. For each of these D_n the location of each node v_j with respect to each of the trigons of D_n is then determined. For each v_j located inside a trigon T, the algorithm checks to see whether $v_j \cup T$ is A. If no sub-drawing A is found, then it is concluded that D_n is rectilinear. On the other hand, if each D_n, has a sub-drawing A, then it is concluded that there is no rectilinear drawing D_n of K_n having the set of crossings X.

```
Step 1.0    Input the dimension $n$ and the crossings of $K_n$

Step 2.0    Find the set $\mathbf{C}$ of all uncrossed k-circuits $C_i$ of $K_n$
            If $\mathbf{C}$ is empty
            Then Output:   *There is no rectilinear drawing $D_n$
            having the given set of crossings*
                  STOP

Step 3.0    $i \leftarrow 1$

Step 3.1    Consider an uncrossed k-circuit $C_i$ of $\mathbf{C}$
            IF $C_i$ is not a CH of a drawing $D_n$
            Then eliminate $C_i$ from $\mathbf{C}$
            IF $i = m$      (where $m$ is the number of
            uncrossed k-circuits $C_i$)
            THEN  GOTO Step 3.2
            ELSE  $i \leftarrow i + 1$
                  GOTO Step 3.1

Step 3.2    $p \leftarrow$ the number of CH's in $\mathbf{C}$
            IF $p = 0$
            THEN Output:  *There is no rectilinear drawing $D_n$
            having the given set of crossings*
                  STOP

Step 4.0    $i \leftarrow 1$

Step 4.1    Consider a CH, $C_i$ of $\mathbf{C}$

Step 4.2    $j \leftarrow 1$

Step 4.3    Consider a trigon T of $D_n$ (**the drawing with
            the CH, $C_i$**), and consider the nodes $\nu$ falling
            in **Int** T
            IF there exists a crossing $(\nu, \alpha) \times (\beta, \delta)$
            where $\alpha$, $\beta$ and $\delta$ are the nodes of T
```

```
                THEN Output: *This drawing $D_n$ cannot be*
                *realized rectilinearly*
                     GOTO Step 4.5

Step 4.4        IF    $j = \binom{n}{3}$   (the number of trigons)
                THEN Output: $C_1$. *$D_n$ having CH $C_1$ is rectilinear*
                     STOP
                ELSE $j \leftarrow j + 1$
                     GOTO Step 4.3

Step 4.5        IF    $i = p$

                THEN Output: *There is no rectilinear drawing $D_n$*
                *having the given set of crossings*
                     STOP
                ELSE $i \leftarrow i + 1$
                     GOTO Step 4.1
```

In Appendix B, the corresponding computer program is presented. The next chapter gives some of the results obtained by running the computer program, followed by an analysis in Chapter 7.

Chapter 6

RESULTS OF THE ALGORITHM

An algorithm which determines whether there exists a rectilinear drawing D_n of K_n is developed in the preceding chapter. In Appendix B a computer program to implement this algorithm for $n \leq 10$ is presented.

All non-isomorphic drawings D_6 obtained by using the algorithm of Chapter 2 and presented in Appendix A.2 were input to the computer program to determine the rectilinear ones. The results are given below, along with the rectilinear drawings D_6. Some drawings D_8, D_9 and D_{10} are also shown here.

Results related to the non-isomorphic drawings D_6

By examining the one hundred and two non-isomorphic drawings obtained in Chapter 3, it was found that:

- 15 drawings are rectilinear,
- 21 drawings are non-rectilinear and have a CH,
- 66 drawings are non-rectilinear and do not have a CH.

Their distribution related to their number of crossings is as follows:

Number of Crossings	Rectilinear D_6	Non-rectilinear		Total number D_6
		Having no CH	Having a CH	
3	1	-	-	1
4	1	-	-	1
5	2	-	1	3
6	1	-	2	3
7	2	6	1	9
8	2	6	5	13
9	2	10	5	17
10	1	8	-	9
11	1	17	3	21
12	1	10	4	15
15	1	9	-	10
Total number of D_6	15	66	21	102

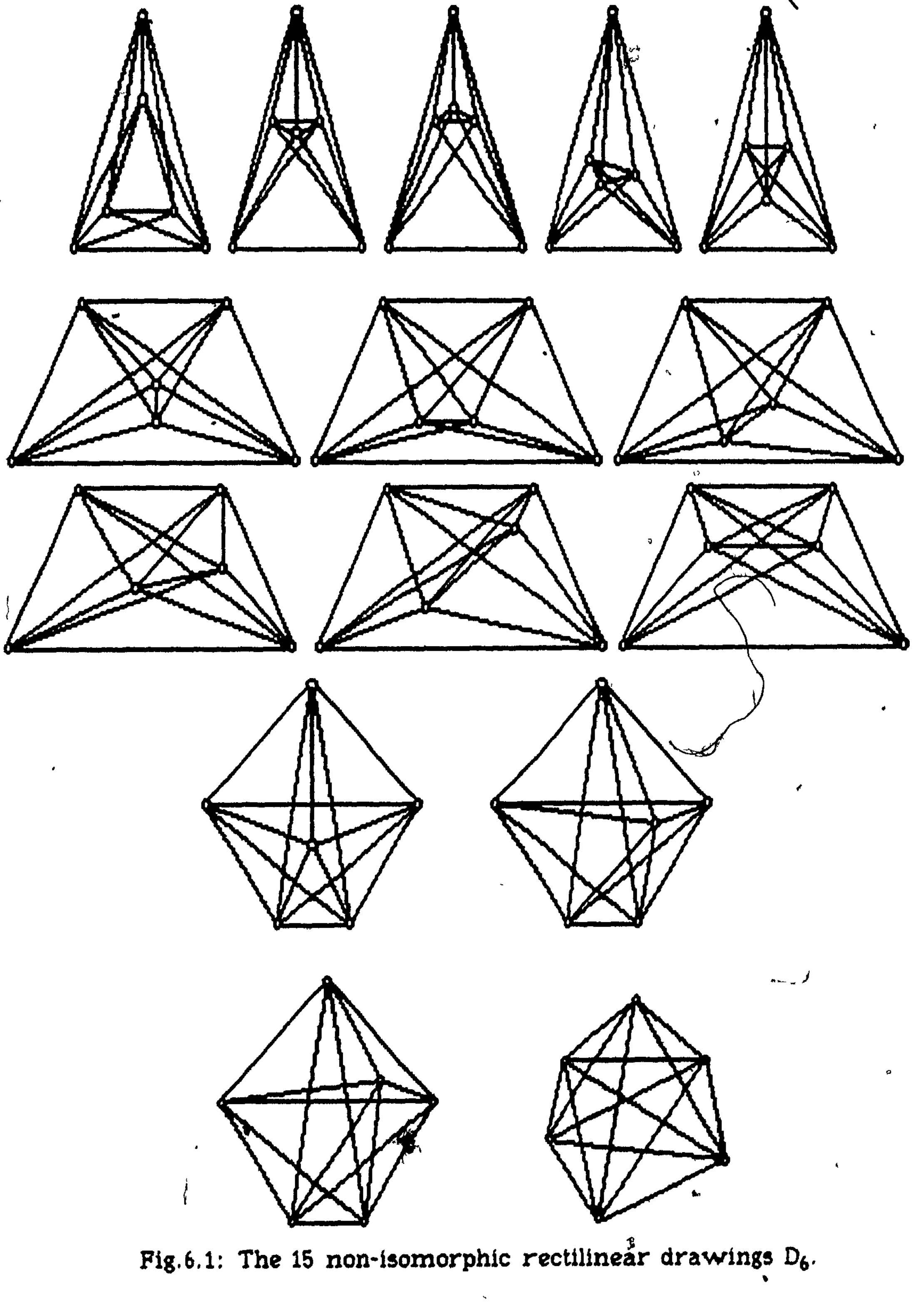


Fig.6.1: The 15 non-isomorphic rectilinear drawings D_6.

Results for some drawings D_8, D_9 and D_{10}

In order to illustrate the possible results which can be reached by the computer program, we present four examples corresponding to four different sets of crossings.

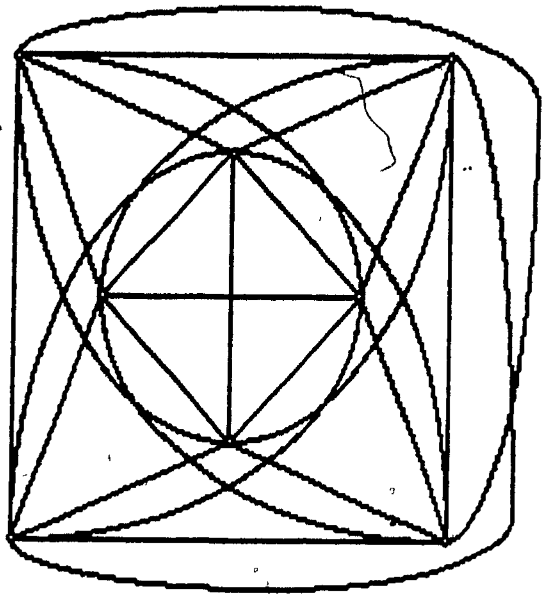
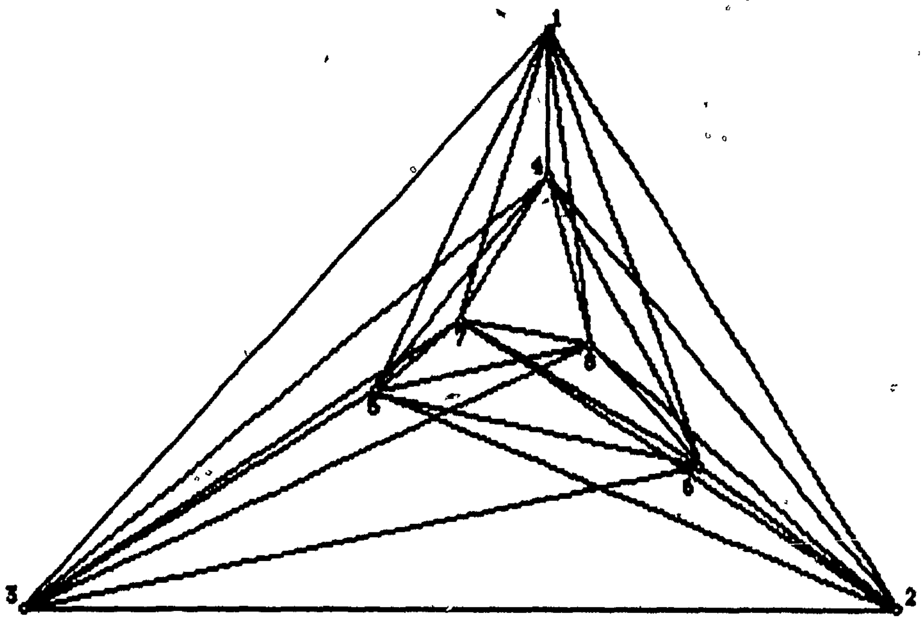


Fig.6.2: A drawing D_8 with no CH's.

The corresponding computer output for the set of crossings defining the drawing in Fig.6.2 is *There is no rectilinear drawing D_n having the given set of crossings,* which is obviously the case.

The two drawings shown in Fig.6.3 share the same set of crossings. The first one has the CH (1,2,3,1) while the second has the CH (4,7,8,4). Both are rectilinear drawings.



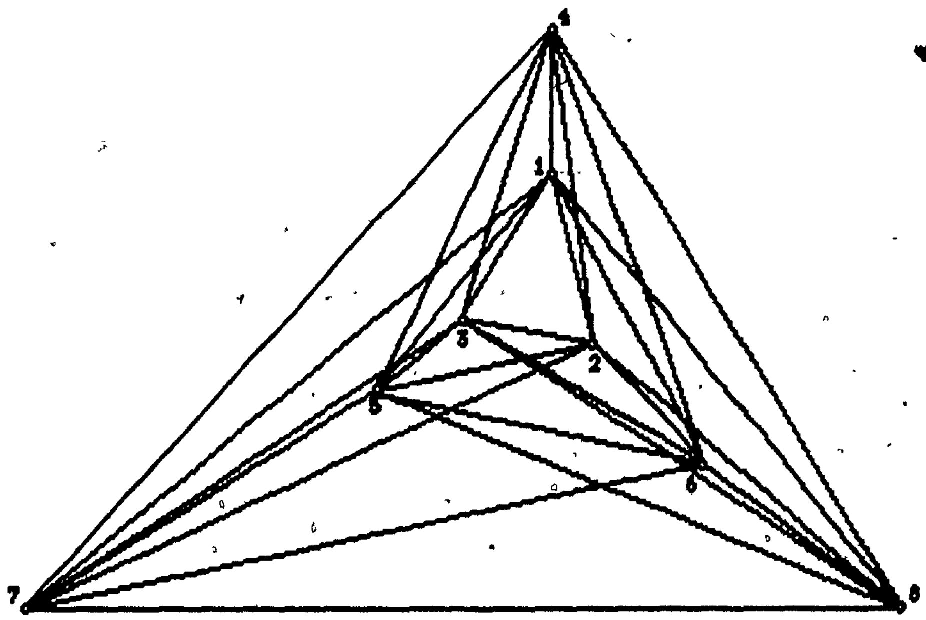


Fig.6.3: Two drawings D_8 with the same set of crossings.

Similarly, the two rectilinear drawings shown in Fig.6.4 share the same set of crossings. The first one has the CH (1,2,3,1), while the second has the CH (7,8,9,7).

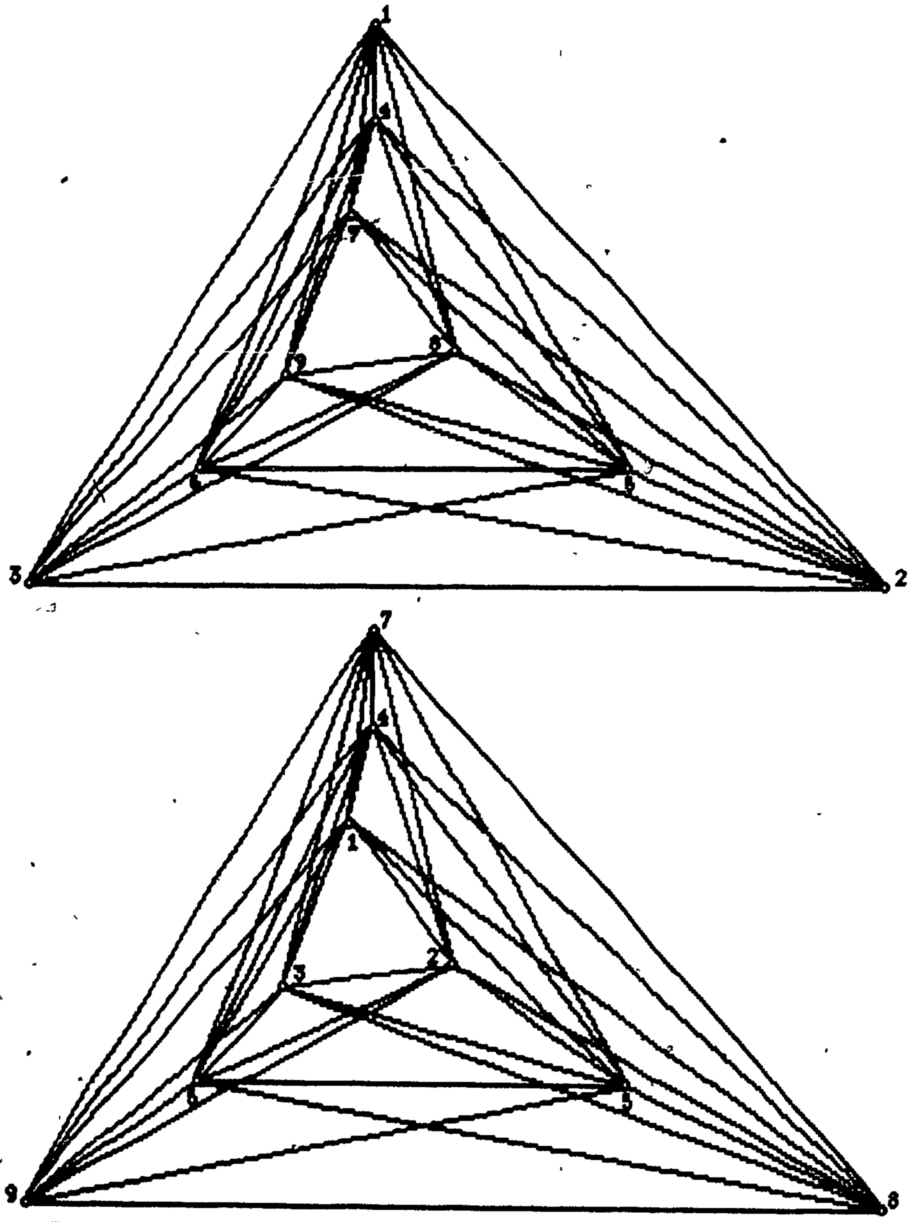


Fig.6.4: Two drawings D_9 sharing one set of crossings.

A set of crossings X of K_{10} is read by the algorithm which determines that there are three drawings D_{10} of K_{10} with the set of crossings X while having different convex hulls, (1,2,3,1), (2,3,10,2) . mand (7,8,9,7). The drawing with CH (1,2,3,1) is rectilinear since it does not have a sub-drawing equivalent to drawing A, as presented in Fig.6.5 .The two other drawings are non-rectilinear.

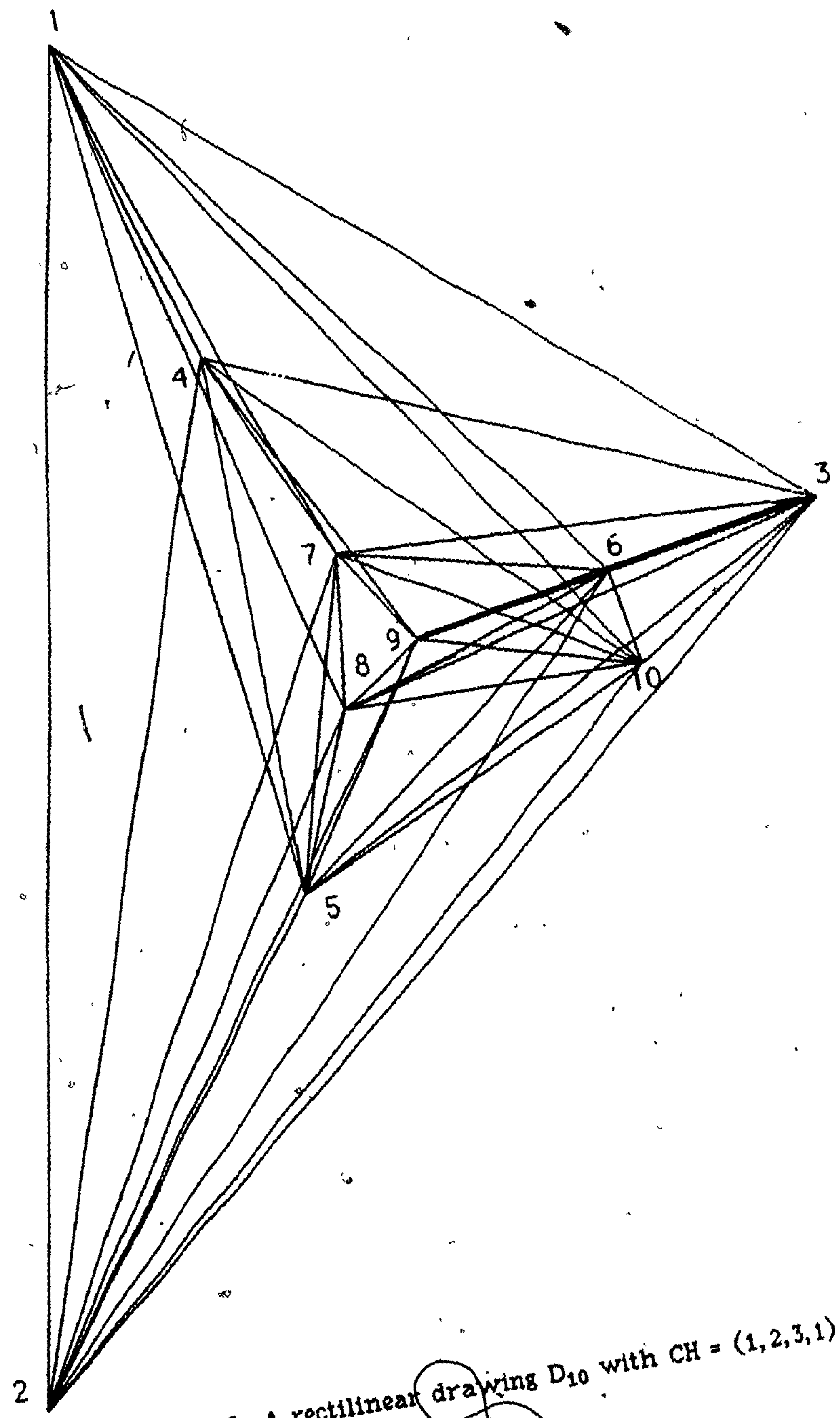


Fig.6.5: A rectilinear drawing D_{10} with CH = (1,2,3,1).

In the drawing with CH (2,3,10,2), consider the trigon T = (2,3,5,2). It can be determined that, since (6,10) crosses only one side of T, node 6 is in the interior of T, as shown in Fig.6.6(a). But (2,6) crosses (3,5), hence we get A as a sub-drawing of the drawing having the CH (2,3,10,2), as shown in Fig.6.6(b).

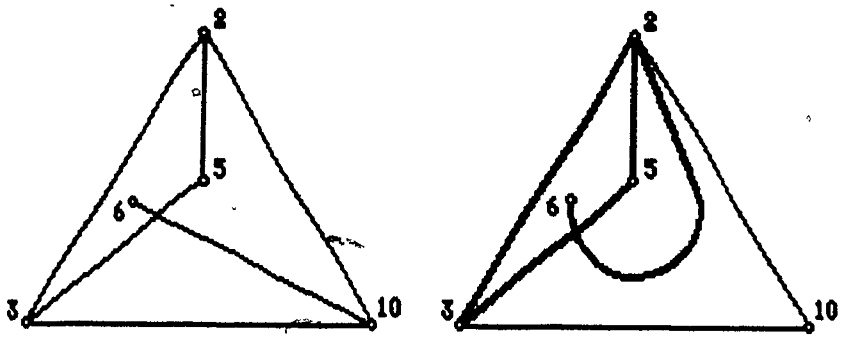
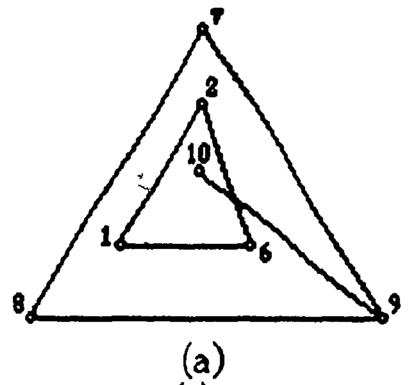


Fig.6.6: In (a), node 6 is in the interior of the trigon (2,3,5,2). In (b), (2,6) × (3,5) producing the sub-drawing A shown in heavy line.

In the drawing with CH (7,8,9,7) the crossing (2,6) × (9,10) implies that node 10 is in the interior of the trigon T = (1,2,6,1), since (9,10) does not cross any other side of T, as in Fig.6.7(a). In addition (1,10) crosses (2,6) resulting in the sub-drawing A, as shown in Fig.6.7(b).



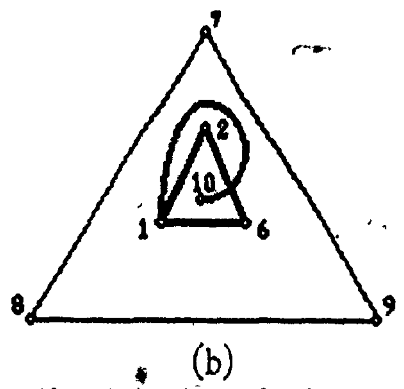


Fig.6.7: In (a), node 10 is in the interior of the trigon (1,2,6,1). In (b), (1,10) × (2,6) producing the sub-drawing A shown in heavy line.

Chapter 7

Analysis of the Algorithm

In Chapter 5 an algorithm that determines whether K_n with a given set of crossings can be drawn rectilinearly is developed. Here we prove two theorems which are used in determining the amount of time required by the algorithm. Then we present an analysis of the algorithm itself.

Background

Let D_n be a good drawing of the complete graph K_n and let C be an uncrossed k-circuit of K_n. If the interior of C does not contain any nodes, arcs or segments of arcs of D_n, then C is the convex hull of a drawing D'_n having the same crossings as D_n. For convenience we call it CH-circuit, as in Fig.7.1.0.

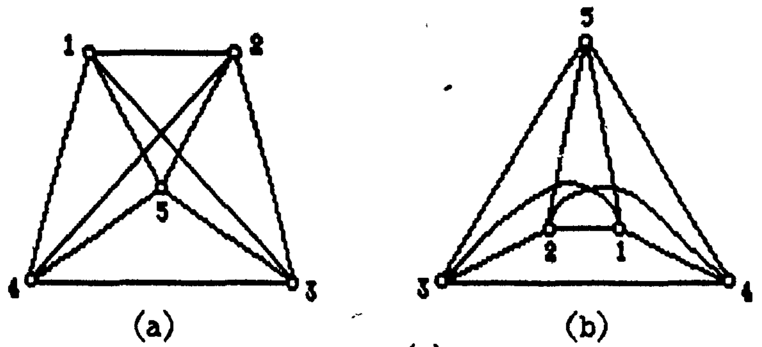


Fig.7.1.0: The drawing D_5 in (a) has three uncrossed k-circuits: (1,2,3,4,1) which is its CH, (1,2,3,5,4,1) and (3,4,5,3). Only the third one is a CH-circuit. It is also the convex hull of D_5 in (b).

We prove in theorem 7.3 that the largest number of CH-circuits in any drawing D_n of K_n is less than n. In Theorem 7.4 we show that we cannot have more than $2(n-1)$ uncrossed arcs in any drawing D_n of K_n. To prove these two theorems, the following two propositions are needed.

Proposition 7.1

Let D_n be a drawing of the complete graph K_n. If D_n has the maximum number of CH-circuits which can occur in a drawing of K_n, then all these CH-circuits are trigons.

Proof (By Contradiction)

Let D_n be a drawing of K_n having the maximum number of circuits. Suppose D_n has a CH-circuit C which is not a trigon, say C=(1,2,3,4,1), then both (1,3), and (2,4) are in **Ext** C as in Fig.7.1(i).

By removing only one of (1,3) and (2,4) from **Ext** C to **Int** C, we obtain a drawing of K_n non-isomorphic to D_n which has at least one CH-circuit more than those of D_n, namely (1,2,3,1) or (1,2,4,1), as shown in Fig.7.1(ii).

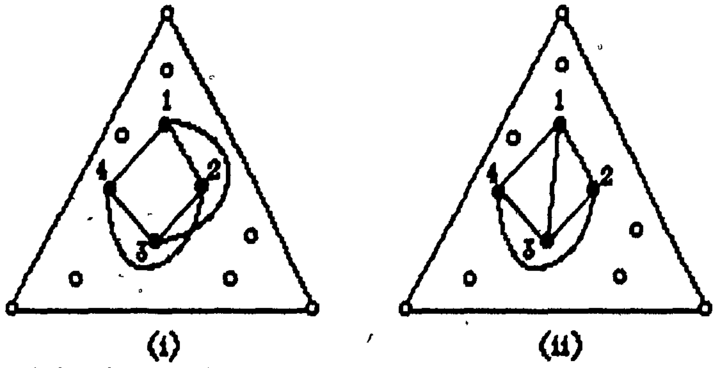


Fig.7.1: In (ii), (1,2,3,1) is a CH-circuit which does not exist in (i).

Proposition 7.2

Let S_n be a sub-drawing of D_n consisting of the largest number of uncrossed trigons, S_n has at most $2n - 5$ trigons containing no nodes in their interior.

Proof (By Induction)

In D_4, the maximum number of uncrossed trigons having no nodes in their interior is three as shown in Fig.7.2.

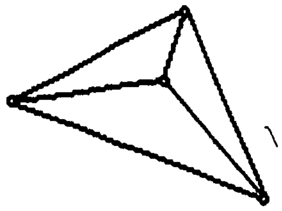
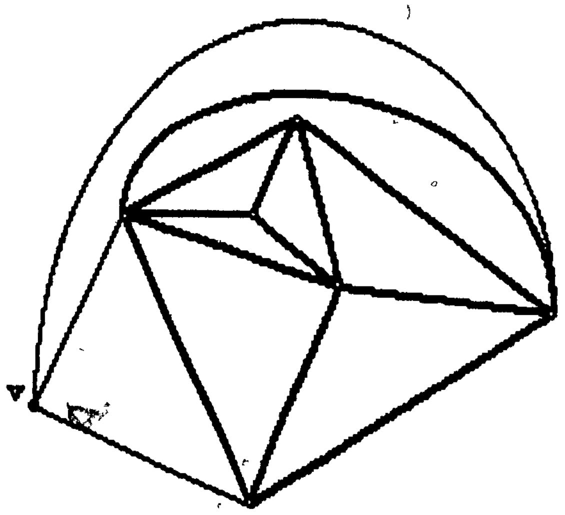


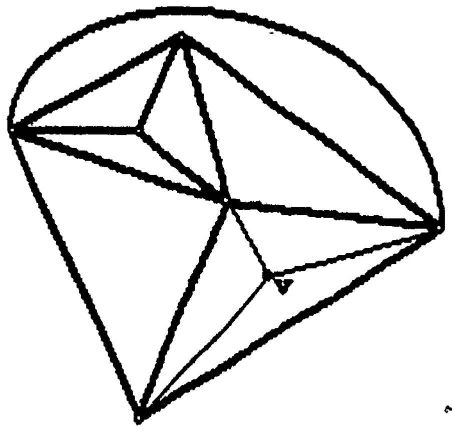
Fig.7.2: A drawing D_4 with three uncrossed trigons containing no nodes in their interior.

Suppose that for S_{n-1}, the maximum number of such trigons is $2(n-1) - 5$. We insert the n-th node v, either in the interior of any of the trigons of S_{n-1}, or in the exterior of all of these trigons. In either case, at most three uncrossed arcs (v, .) can be inserted, hence increasing the number of trigons by 2, to $2n - 5$ trigons as shown in Fig.7.3.

□



(I)



(II)

Fig.7.3: In (i) a node is inserted in the exterior of all trigons. In (ii) a node is inserted in the interior of a trigon.

Next, we present two theorems which are useful in analyzing the algorithm. The first one establishes an upper bound on the number of CH-circuits in D_n, while the second theorem provides an upper bound on the number of uncrossed arcs in D_n.

Theorem 7.3

The largest number of CH-circuits in a drawing D_n of K_n does not exceed $n-1$.

Proof

From Proposition 7.1 and 7.2 we know that the maximum number of the uncrossed trigons of D_n having no nodes in their interior cannot exceed $2n-5$; however, some of these $2n-5$ uncrossed trigons are not CH-circuits because of the following.

Suppose a node α is in the interior of a trigon T, then (α,β) must cross one of the arcs of T whenever β is in **Ext** T, as shown in Fig.7.4.

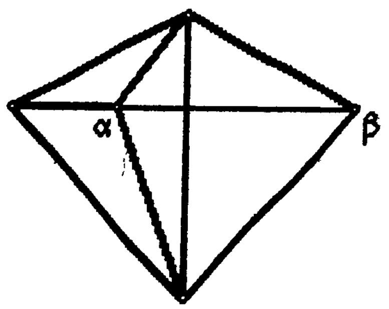


Fig.7.4: Node α in Int T and node β in Ext T implies that (α,β) crosses an arc of T.

Hence, either no nodes are located in the interior of any of the trigons, or exactly one trigon contains all nodes. This way we have a maximum of $n - 1$ CH-circuits as shown in Fig.7.5.

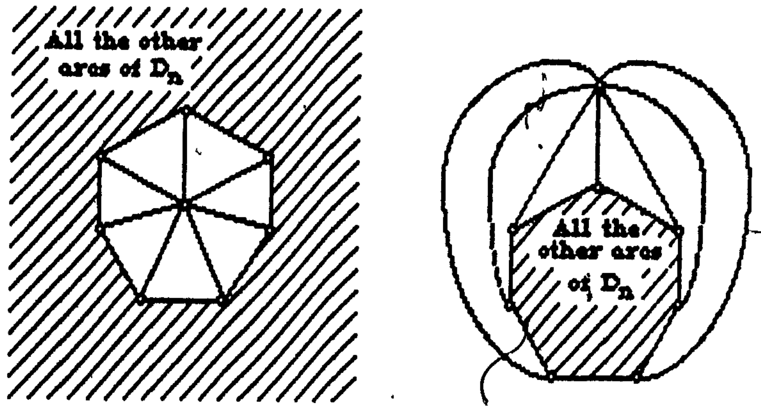


Fig.7.5: A drawing D_n can have at most $n - 1$ CH-circuits.

Theorem 7.4

The largest number of uncrossed arcs in any drawing D_n of K_n does not exceed $2(n - 1)$.

Proof (By Induction)

In D_4, the largest number of uncrossed arcs is six as shown in Fig.7.6.

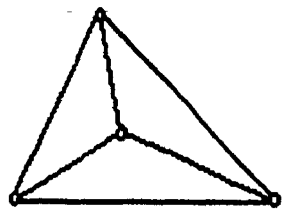


Fig.7.6: A D_4 with the maximum number of uncrossed arcs.

Suppose that for D_{n-1}, the largest number of uncrossed arcs is $2[(n-1)-1]$. We add an n-th node, $\mathbf{v}$, to D_{n-1}. If $\mathbf{v}$ is inserted in the interior of a trigon $(\alpha_1, \alpha_2, \alpha_3, \alpha_1)$, then at most three arcs $(\mathbf{v}, \bullet)$ are uncrossed. But at least one arc (α_i, α_j) is crossed, hence at most only two uncrossed arcs are added. A similar argument can be used when $\mathbf{v}$ is inserted in the exterior of D_{n-1}, as shown in Fig.7.7 □

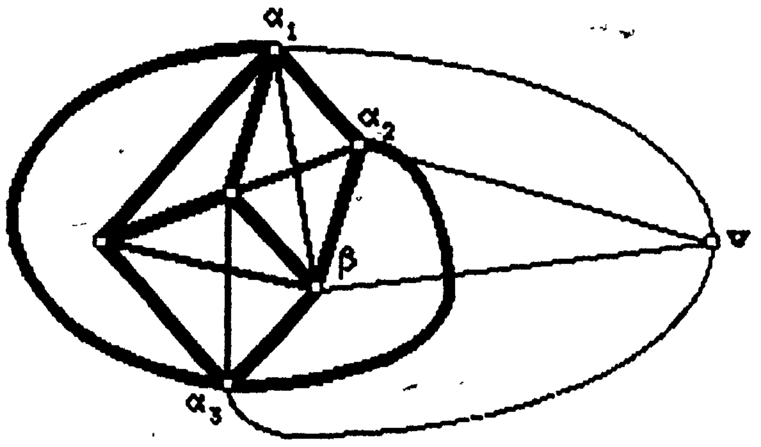


Fig.7.7: Three uncrossed arcs $(\mathbf{v}, \alpha_1)$ are added, while an arc (α_i, α_j) which was uncrossed is now crossed by $(\mathbf{v}, \beta)$.

Analysis

The four steps of the algorithm presented in Chapter 5 are considered and described with some details in the next three pages.

Step #	Step Description	Number of Operations
1.0	* Input	
1.1	- Input dimension and number of crossings of K_n	constant
1.2	- Input crossings, sort them in ascending order and reorder a crossing $(a,b) \times (c,d)$ such that $a < b$, $c < d$ and $a < c$	$O(n^4)$
2.0	* Find all uncrossed k-circuits C_i of K_n	
2.1	- Find all uncrossed edges of K_n	$O(n^4)$
2.2	- Initialize arrays to be used in the Depth First Search procedure	$O(n^2)$
2.3	- Depth First Search procedure to determine all C_i. If none exists Output *K_n has no uncrossed circuits, there is no rectilinear drawing D_n having this set of crossings*, Stop	$O(n!)$
2.4	- Order each C_i, such that $C_i(1) < C_i(j)$ for $j=2,3,\ldots,k$ and $C_i(2) < C_i(k)$ where k is the number of vertices of C_i	$O(n^3)$
2.5	- Eliminate duplicate C_i's and form a set $\mathfrak{C}$ of the remaining C_i's	$O(n^3)$
3.0	* Find all CH's of K_n by considering each C_i of $\mathfrak{C}$ and determining whether it is a CH-circuit	$O(n^7)$
3.1	- Determine all the nodes v different from the nodes of C_i	$O(n^2)$

3.2	- Determine the number of crossings x involving only the nodes of C_i. If $x \neq \binom{k}{4}$, then C_i is not a CH. Eliminate C_i from $\mathfrak{C}$. If $\mathfrak{C}$ is empty and no CH has been found, then Output *K_n has no CH, there is no rectilinear drawing D_n having this set of crossings* and Stop, else Step 3.0	$O(n^5)$
3.3	- If $k=3$ then C_i is a CH, and Step 3.0 - If $x = \binom{k}{4}$, then for each node v determine whether there is a crossing $(v,a) \times (b,c)$ where b and c are nodes of C_i. If there is a node v which has no such crossings then C_i is not a CH. Eliminate C_i from $\mathfrak{C}$. If $\mathfrak{C}$ becomes empty and no CH has been found, then Output *K_n has no CH, there is no rectilinear drawing D_n having this set of crossings* and Stop; else Step 3.0	$O(n^6)$
4.0	* Consider a CH, C_i of $\mathfrak{C}$ - For each trigon T of D_n, determine whether there is a crossing $(v,a) \times (b,c)$ such that v is in **int** T and a, b, c are nodes of T	$O(n^8)$
4.1	- Determine all nodes v located in **int** T	$O(n^5)$
4.2	- For each v determine whether $(v,a) \times (b,c)$ - If there is no T and v for which such crossings occur then Output *D_n having CH, C_i is Rectilinear,* else Output *D_n cannot be realized rectilinearly with CH C_i.*	$O(n^5)$

Eliminate C_1 from **C**. If
C is not empty then
Step 4.0, else Stop.

Although the bound used in Step 2.3 is never reached, due to the existence of crossings when $n > 4$, we know that this step might require as much as $O(2^{f(n)})$ where $f(n)$ is a function of n as illustrated in Fig.7.8.

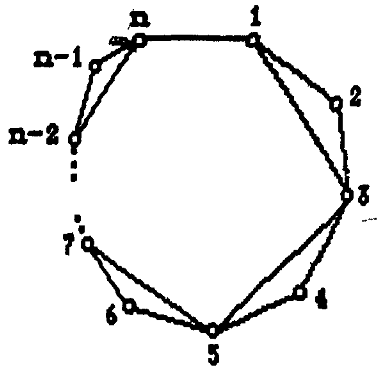


Fig.7.8: Only the uncrossed arcs of a drawing D_n are shown. The remaining arcs of D_n would be in Ext C, where $C = (1,2,3,\ldots,n,1)$. The number of uncrossed k-circuits in this drawing is 2^r where $r = (n-1)/2$; while the number of CH-circuits is just $(n+1)/2$.

On the other hand, from Theorem 7.3 the number of CH-circuits in any drawing D_n of K_n is less than n. Hence, if in addition to the set of crossings of K_n we are given the possible CH-circuits then Step 2.3 will be skipped and the time required by the algorithm will drop to $O(n^9)$. The drawing in Fig.7.8 illustrates the fact that the number of uncrossed circuits is not proportional to the number of CH-circuits.

Chapter 8

AN ALGORITHM FOR FINDING ALL NON-ISOMORPHIC RECTILINEAR DRAWINGS D_n OF K_n

Background

In this chapter we present an algorithm to find all the non-isomorphic rectilinear drawings D_n of K_n. The input to the algorithm is the set of crossings corresponding to each of the non-equivalent rectilinear drawings D_{n-1} of K_{n-1}. For each of these drawings, the algorithm generates a set of rectilinear drawings D_n. The set of all the sets of these drawings D_n contains all the non-isomorphic rectilinear drawings D_n.

First we produce a theorem on which the algorithm is based Then we present the algorithm.

Before proceeding with the theorem, we introduce the following definition which is required for the theorem.

Definition

Let (ν,α) and (ν,β) be two arcs of D_n. If the area bounded by the triangle having nodes ν, α and β, does not contain any arcs or segment of arcs $(\nu,\bullet)$, then (ν,α) and (ν,β) are *adjacent.*

Theorem 8.1

Let D_n be a rectilinear drawing of the complete graph K_n. Let the nodes of D_n be denoted by $1,2,3,\ldots,n$ such that the arcs (n,i) and $(n,i+1)$ be adjacent and such that the node n be on the CH of D_n as shown in Fig.8.1.1.

If all the crossings involving the node n are given, then all the remaining crossings of D_n can be determined.

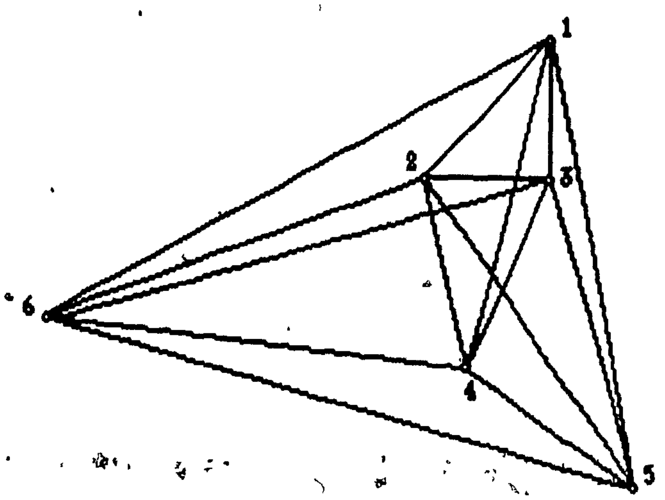


Fig.8.1.1: Knowing that all the crossings, involving node 6, are $(6,3)\times(2,4)$, $(6,3)\times(2,5)$ and $(6,3)\times(1,4)$, we can determine that the remaining crossings of D_6 are $(1,4)\times(2,3)$, $(1,4)\times(2,5)$ and $(2,5)\times(3,4)$.

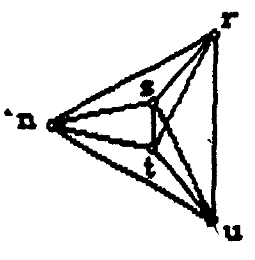
Proof

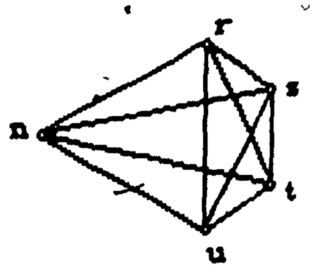
Let $1 \le r < s < t < u \le n-1$ (1)

then (r,t) may cross (n,s), (s,u) may cross (n,t),

and (r,u) may cross any of (n,s) and (n,t).

By considering all the possible sub-drawings formed by n, r, s, t and u and their corresponding arcs, we get eight sub-drawings represented in Fig.8.1.2.





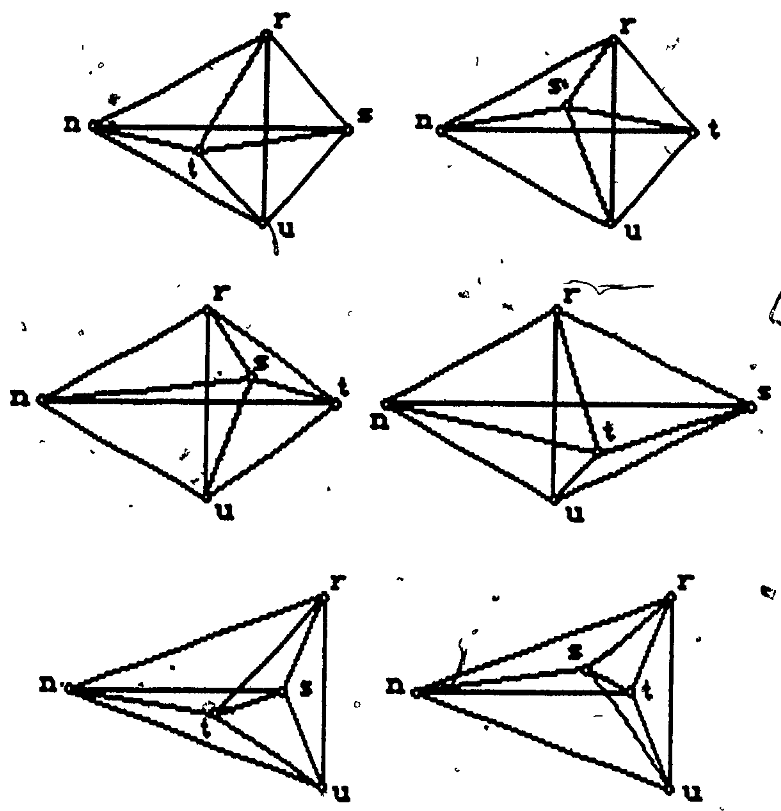


Fig.8.1.2: The eight possible sub-drawings with nodes n, r, s, t and u, and their corresponding arcs.

From these sub-drawings we obtain the following:

$$\left.\begin{array}{l}(r,t) \times (s,u) \Leftrightarrow \begin{cases}[(r,t) \,||\, (n,s) \text{ and } (s,u) \,||\, (n,t)] \\ \text{or} \\ [(r,t) \times (n,s) \text{ and } (s,u) \times (n,t)]\end{cases} \\ (r,u) \times (s,t) \Leftrightarrow \begin{cases}[(r,u) \times (n,s) \text{ and } (r,u) \,||\, (n,t)] \\ \text{or} \\ [(r,u) \,||\, (n,s) \text{ and } (r,u) \times (n,t)]\end{cases}\end{array}\right\} \quad (2)$$

We consider every possible combination of four nodes r, s, t and u such that the inequalities (1) are satisfied. From (2), knowing all the crossings involving the arcs (n, s) and (n, t) is equivalent to knowing whether (r, t)×(s, u) and whether (r, u)×(s, t).

□

Background to the Algorithm

Let D_{n-1} be a rectilinear drawing with k nodes on its CH, **C**. Consider one of the nodes of **C** and label it n−1. Label the remaining nodes such that (n, i) and (n, i+1) become adjacent, for i = 1, 2, ..., n−2 as shown in Fig. 8.1.3.

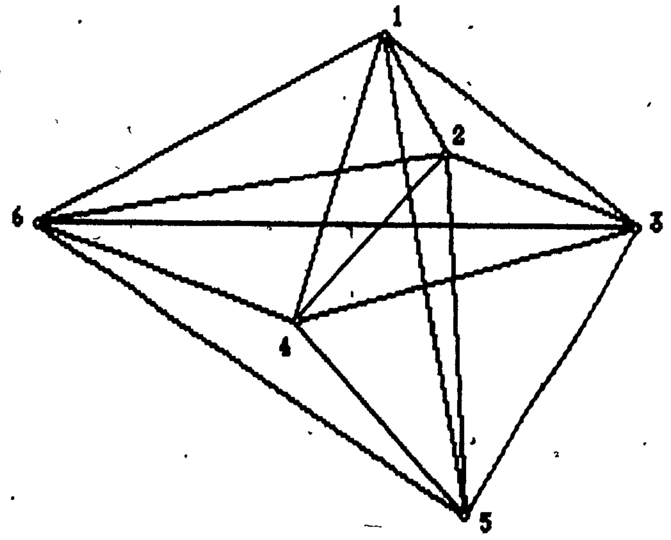


Fig. 8.1.3: A drawing D_6 being labeled such that (6, i) and (6, i+1) are adjacent, (i = 1, 2, 3, 4).

We call the area bounded by **C**, **Int C**; and we call the remainder of the plane, **Ext C**. We add a node n to D_{n-1}, such that (n−1, n) be in **Ext C** and (n−1, n) and (n−1, n−2) be adjacent as shown in Fig. 8.1.4.

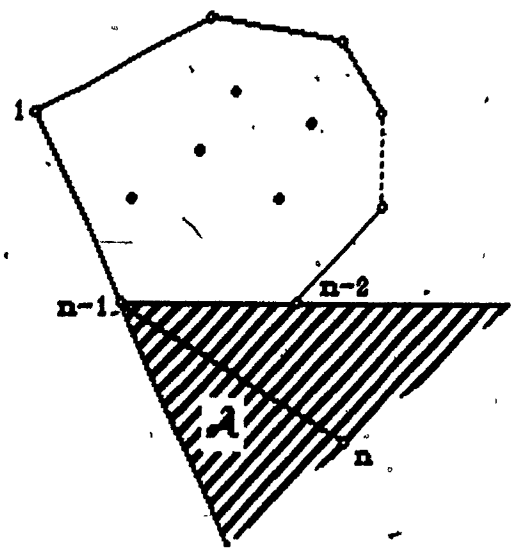
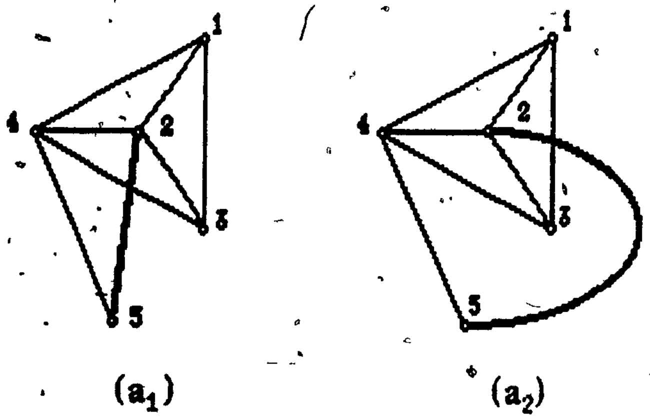


Fig.8.1.4: The shaded area $\mathcal{A}$ indicates the possible locations for node n.

By considering all the possible mappings of an edge (n,i), we note that they cannot exceed 2^{n-i-2}. This is of course due to the fact that when mapping (n,i), only one of the following two situations can occur as shown in Fig.8.1.5:

$$\left.\begin{array}{l}(n,i)\times(n-1,j)\\(n,i)\,||\,(n-1,j)\end{array}\right\}\begin{array}{l}i=1,2,\dots,n-3\\i<j\leq n-2\end{array}.$$



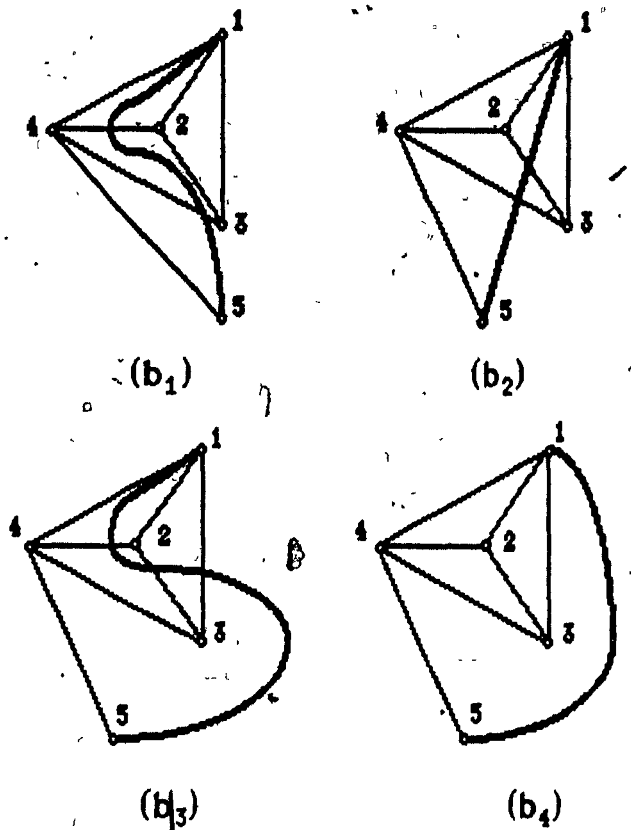


Fig.8.1.5: Two mappings are possible for (5,2), (a_1) & (a_2); and the number of mappings of (5,1) cannot exceed 2^2. We notice that (b_3) is not a good drawing.

We input the crossings of D_{n-1} to the algorithm which considers each of the possible mappings of each edge (n,i). Some of these mappings produce rectilinear drawings. Only these rectilinear drawings are retained by the algorithm.

The same process is repeated with each of the nodes of $\mathfrak{C}$; hence the algorithm generates a set of drawings containing all the non-isomorphic rectilinear drawings D_n which have D_{n-1} as a sub-drawing and a node n located in $\mathfrak{Ext}\,\mathfrak{C}$ as shown in Fig.8.1.6.

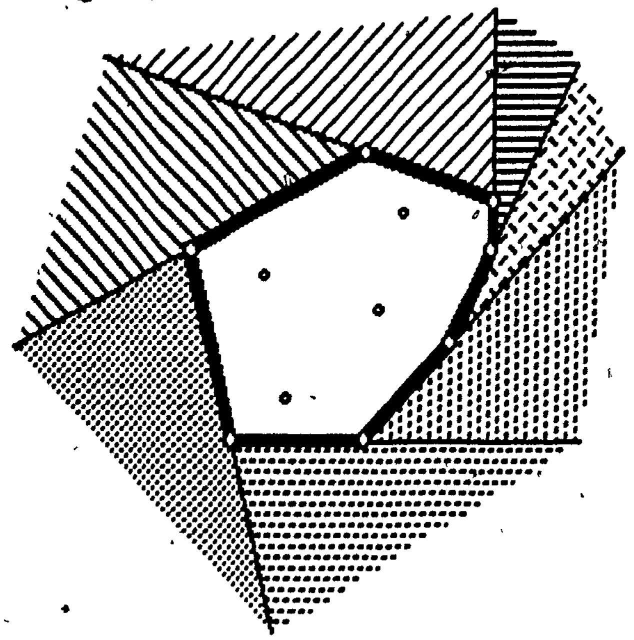


Fig.8.1.6: The CH, $\mathfrak{C}$ of a drawing D_{n-1} is shown in heavy line. $\mathfrak{C}$ has k nodes, and Ext $\mathfrak{C}$ is divided in k regions (shaded areas). A node n is placed in one of the k areas and all rectilinear drawings D_n consisting of the union of D_{n-1} and the arcs (n,i) are obtained. The node n is then placed in another shaded area to obtain another set of rectilinear drawings D_n. The process is repeated for each of the k shaded areas. The result is the set of all rectilinear drawings D_n satisfying the following:

1. D_{n-1} is a sub-drawing of D_n,
2. node n is on the CH of D_n.

Let $\mathfrak{D}_{n-1}$ be the set of all non-equivalent rectilinear drawings with n-1 nodes,

$$\mathfrak{D}_{n-1}=\{D_{n-1}{}^1, D_{n-1}{}^2, D_{n-1}{}^3, \ldots, D_{n-1}{}^q\}$$

If $D_{n-1}{}^j$ has a CH with k nodes then the nodes of $D_{n-1}{}^j$ might have to be re-labeled k times. We denote the set of crossings using the i-th labeling of the nodes $D_{n-1}{}^j$ by $X_i{}^j$.

Finally, let $\mathfrak{P}$ be the set of all values of

$\{p_{1,1}, p_{2,1}, p_{2,2}, \ldots, p_{k,1}, p_{k,2}, \ldots, p_{k,k}, \ldots, p_{n-3,1}, p_{n-3,2}, \ldots, p_{n-3,n-3}\}$, where $p_{k,l}$ takes the value 0 or 1 depending on whether the two arcs (n,n-k-2) and (n-1,n-l-1) cross.

The Algorithm

Step 1: Input:
1. the dimension n
2. the number of crossings of D_{n-1}^j, where D_{n-1}^j belong to $\mathfrak{D}_{n-1}$
3. the number of nodes on the CH of D_{n-1}^j
4. the crossings X_1^j obtained with the first labeling
5. the q-1 sets of labels to be used in generating $X_2^j, X_3^j, \ldots, X_q^j$

```
Step 2      j ← 1
Step 2.1    i ← 1
Step 2.2:   𝐗 ← X_i^j
            m ← 1
Step 2.3:   k ← 1
Step 2.4:   l ← 1
Step 2.5:   IF     p^m_{k,l} = 1 where p^m_{k,l} = p_{k,l} of the m-th set of 𝒫
            THEN   𝐗 ← 𝐗 ∪ (n, n-k-2) × (n, n-l-1)
            IF     l = k
            THEN   IF      k = n-3
                   THEN    Step 3
                   ELSE    k ← k+1
                              Step 2.4
            ELSE   l ← l+1
                   Step 2.5
```

Step 3: Consider each combination of nodes r, s, t along with the two nodes n and n−1, where $r < s < t < n-1$; and let $S_{r,s,t}$ be the sub-drawing consisting of the nodes r, s, t, n, n−1 and their corresponding arcs.

```
Step 4:   IF $S_{r,s,t}$ is rectilinear for each of the
            combinations of r,s and t
          THEN using Theorem 8.1, determine the remaining
            crossings of $D_n$, and
            Output the set of crossings $\mathbf{X}$ of $D_n$
Step 5:   IF     $m < M$ (where M is the number of sets P of $\mathbf{P}$)
          THEN   $m \leftarrow m+1$
                 Step 2.3
          Output the crossings of $D_n$
Step 6:   IF $i < q$
          THEN    $i \leftarrow i+1$
                  Step 2.2
Step 7:   IF $j < J$ (where J is the number of drawings $D_{n-1}$ to be
                  read by the program)
          THEN    $j \leftarrow j+1$
                  Step 2.1
          ELSE    STOP.
```

The set of drawings D_n produced by the algorithm contains all the non-isomorphic rectilinear drawings D_n. A computer program to obtain all D_n from the non-equivalent drawings D_{n-1} has been developed and its listing is produced in Appendix C.1. All non-isomorphic rectilinear drawings D_7 are given in Appendix C.2. Results which are obtained by implementing the algorithm for D_4 and D_5 and by using the corresponding computer programs for D_6 and D_7 are presented in the next chapter.

Chapter 9

Results of the Algorithm

In this chapter, we find all the non-isomorphic rectilinear drawings D_n using D_{n-1} when $n \leq 7$. Using the rectilinear drawing D_3, we obtain exactly two drawings D_4 as shown in Fig. 9.1.

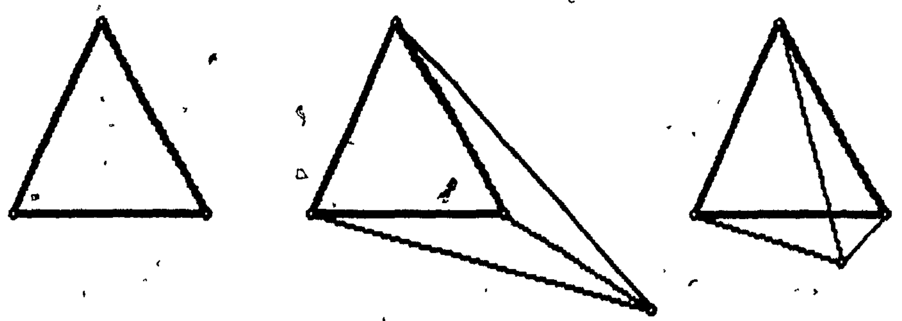
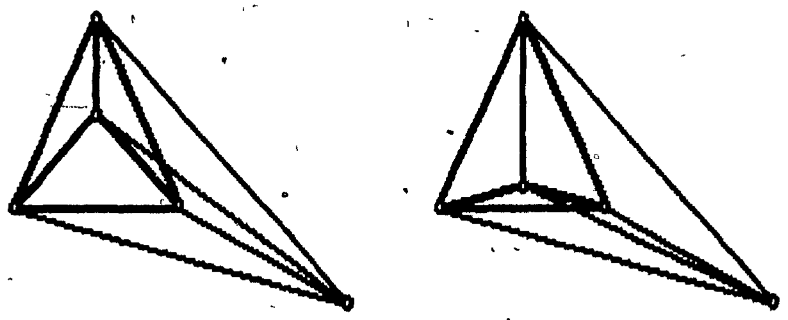


Fig. 9.1: Two drawings D_4 are obtained from the drawing D_3.

We call the first drawing D_4, (a), and the second one, (b). Using (a), we obtain the four drawings D_5, (a.1), (a.2), (a.3) and (a.4); using (b), we obtain the four drawings (b.1), (b.2), (b.3) and (b.4) as shown in Fig. 9.2. The three non-isomorphic rectilinear drawings D_5 are among these eight drawings.



(a.1) (a.2)

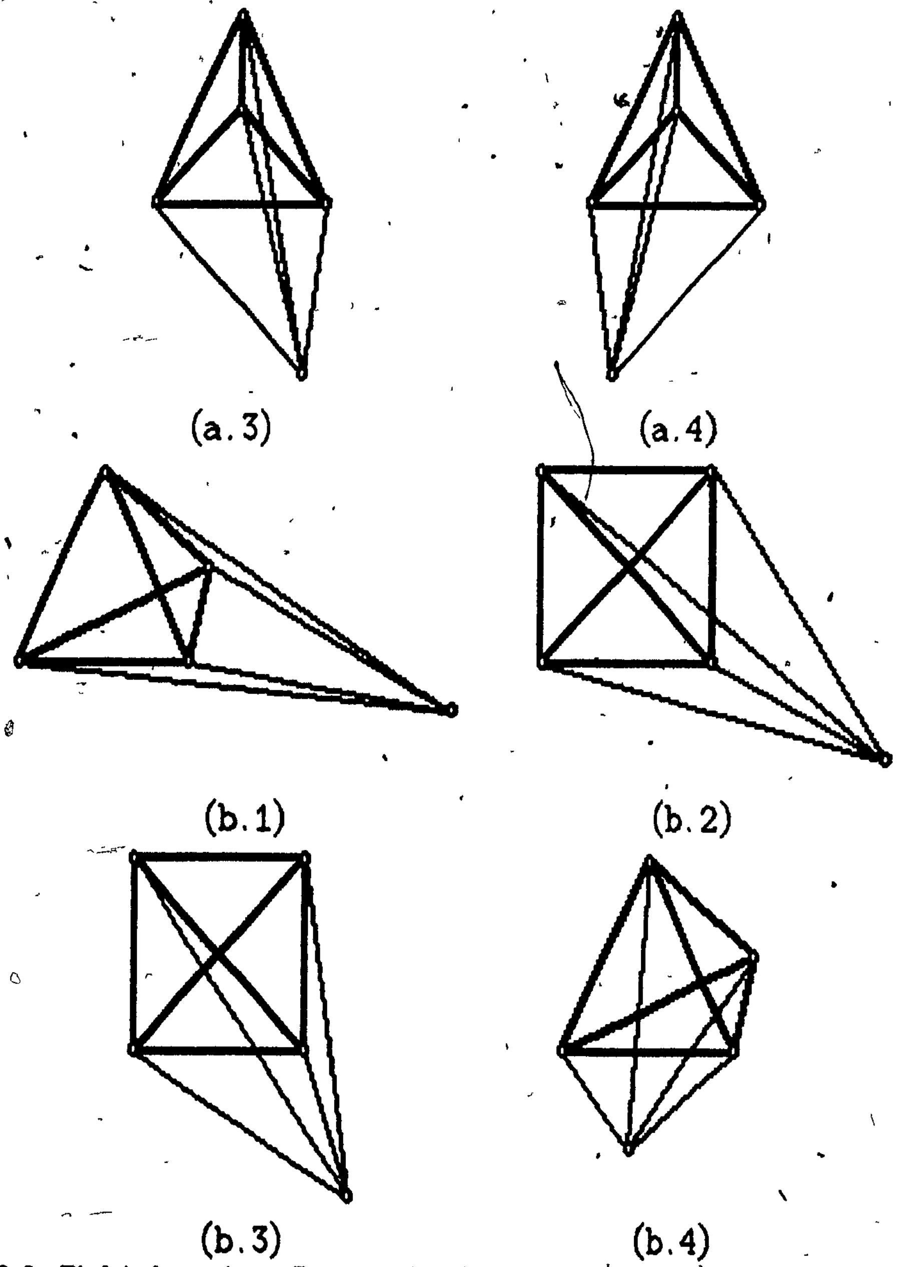


Fig. 9.2: Eight drawings D_5 are obtained from the two drawings D_4.

We consider one of the three drawings D_6 as shown in Figs. 9.3 and 9.4.

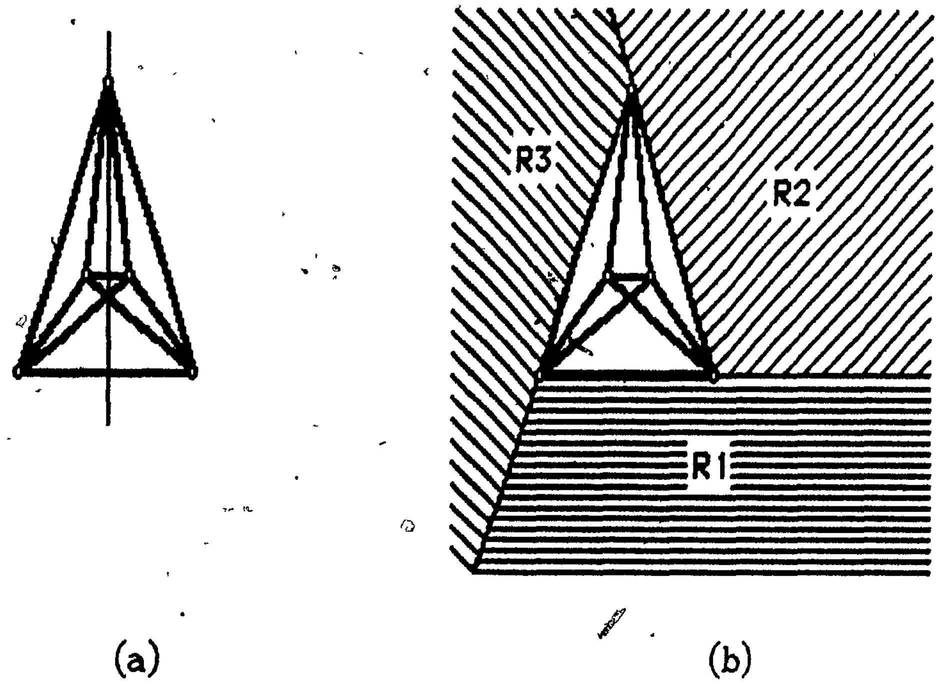


Fig. 9.3: In (a), the same drawings D_6 are obtained by placing the sixth node to the right or to the left of the light line. We then note that it suffices to consider the sixth node in the regions R1 and R2 in (b) since these two regions will cover all the half plane on the right of the light line.

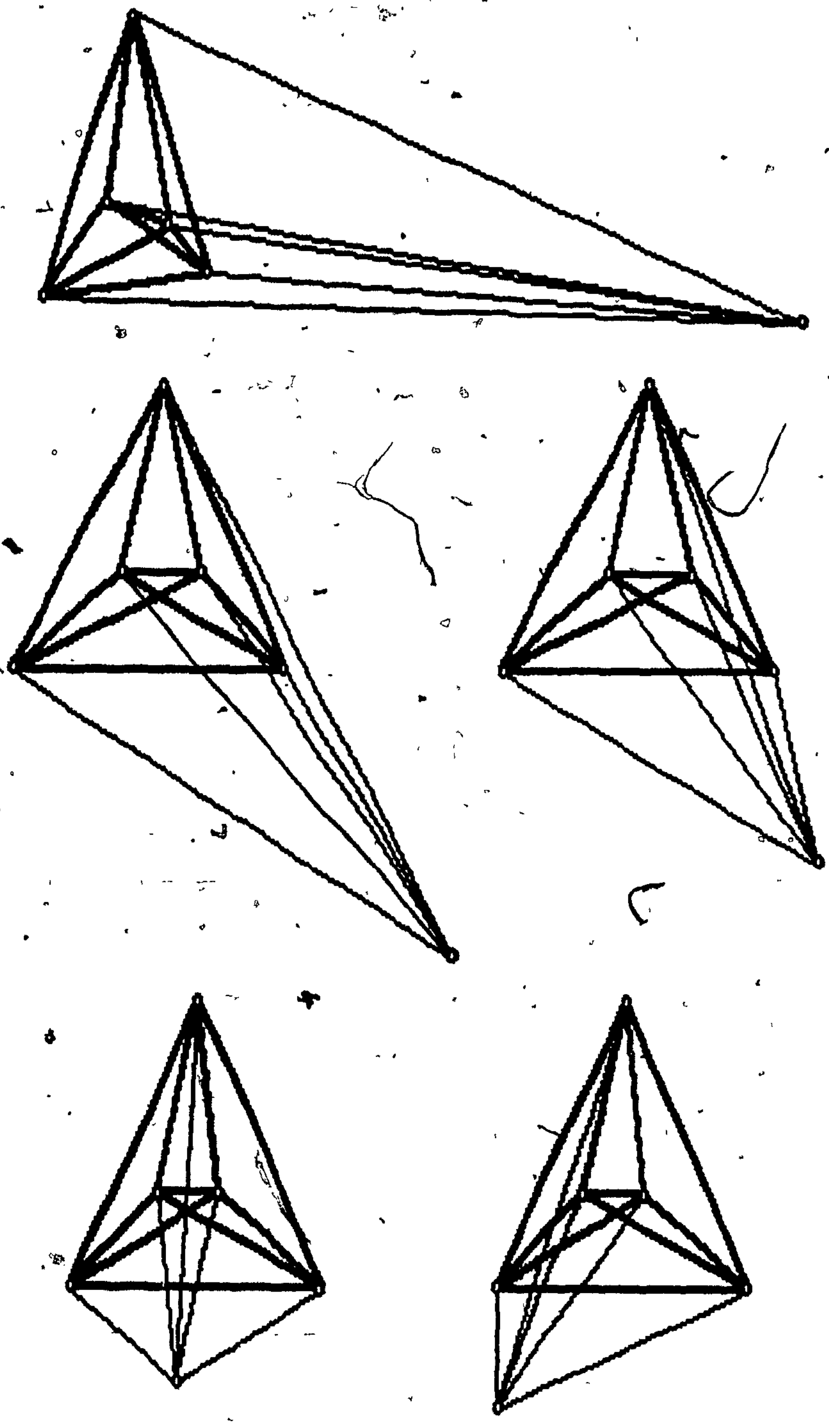


Fig.9.4: All rectilinear drawings D_6 obtained by considering the sixth node in region R1.

First, we place the sixth node in the region R1 to obtain the rectilinear drawings shown in Fig. 9.4. In a similar fashion we consider the sixth node in the region R2 to obtain a set of rectilinear drawings D_6 as shown in Fig. 9.5.

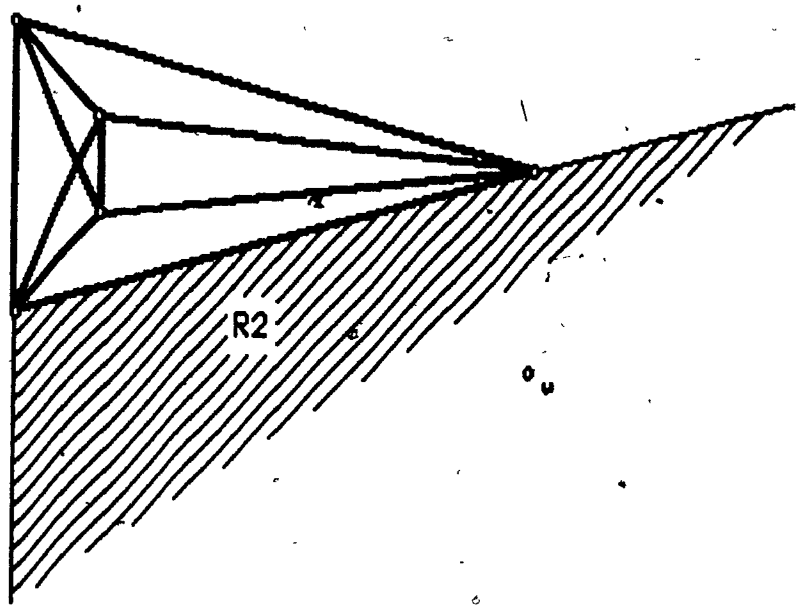
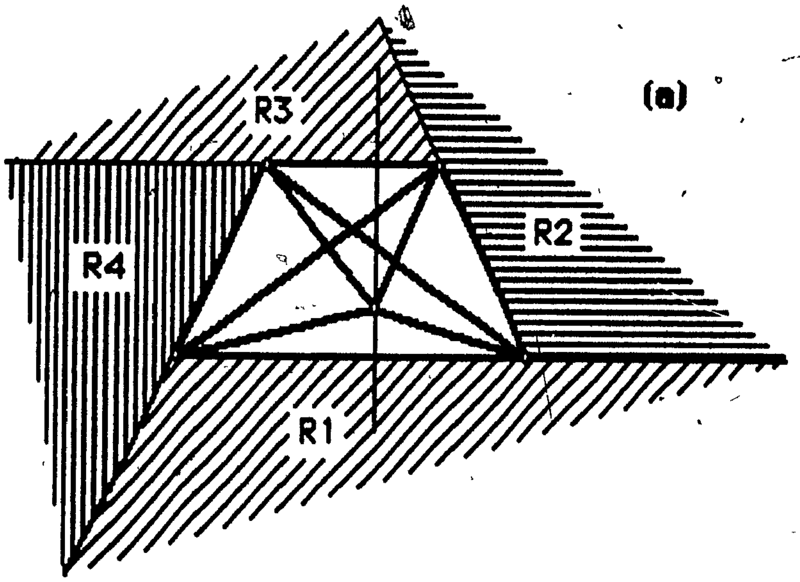


Fig. 9.5. A node v is placed in the region R2 and drawings D_6 are obtained by connecting v to the nodes of D_5 in every possible way.

The two other drawings D_5 as shown in Fig. 9.6, are treated similarly to obtain all the rectilinear drawings D_6, as shown in Fig. 9.7.



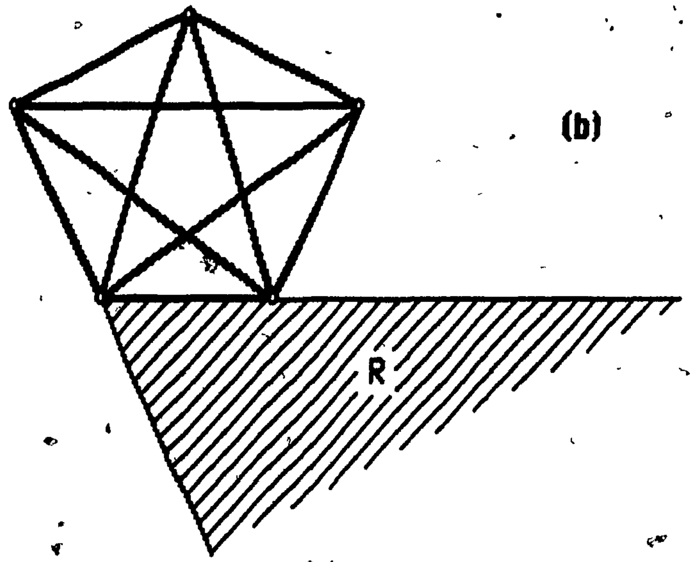
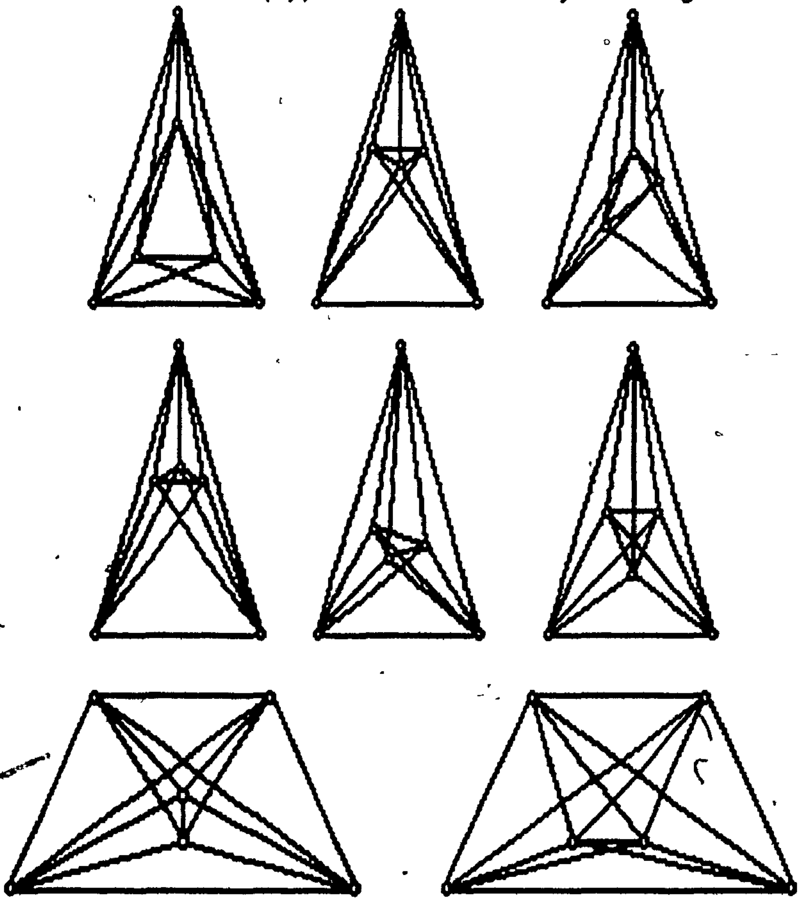


Fig.9.6: The drawing (a) is symmetric about the thin line, only the regions R1, R2 and R3 need to be considered. In (b), we consider only the region R.



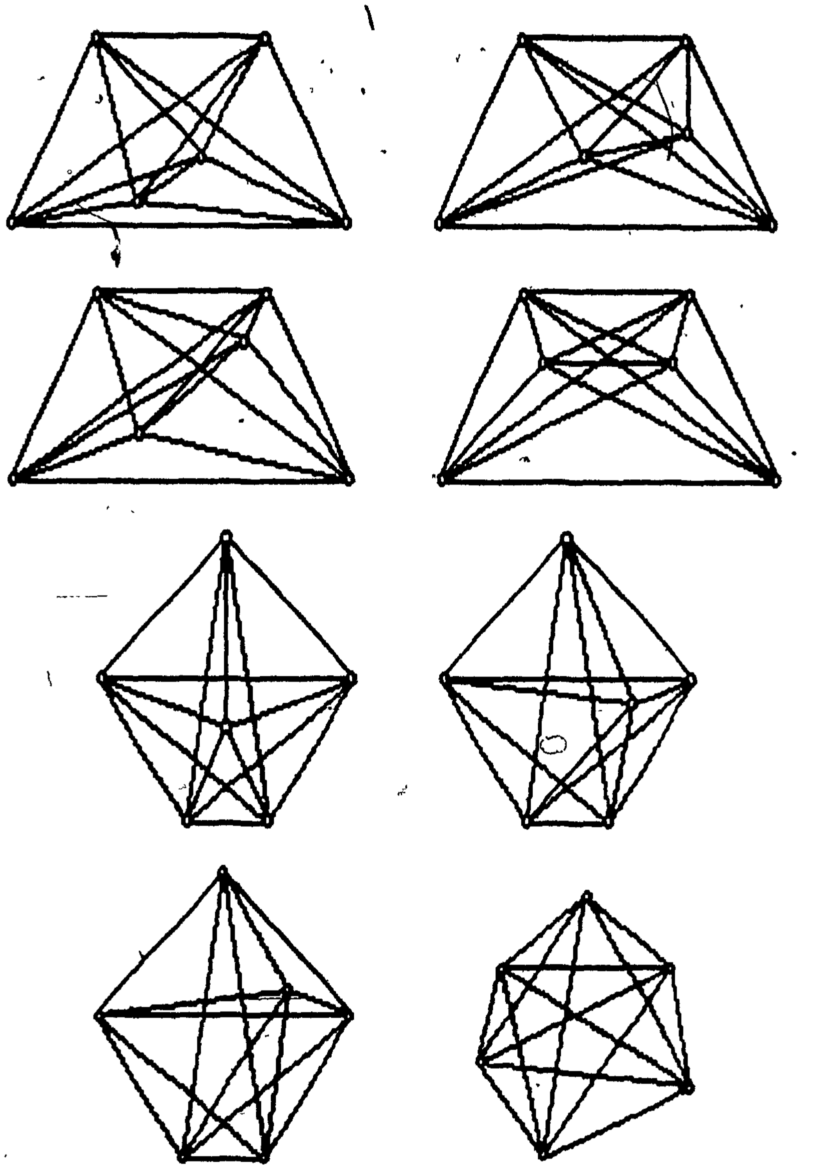


Fig.9.7: All the non-equivalent rectilinear drawings D_6.

The sixteen drawings D_6, given in Fig.9.7, are used to obtain all the non-isomorphic rectilinear drawings D_7. The total number of these drawings is one hundred and twenty-two (122). The following table represents their distribution by the number of crossings and by the number of arcs of their convex hulls. A listing of the computer program which has generated the one hundred and twenty-two D_7 is given in Appendix C.1.

Number of Non-isomorphic Rectilinear D_7
(By number of crossings and by number of arcs of CH)

No. of Crossings \ No. of Arcs of CH	3	4	5	6	7	No. of Drawings
9	3					3
11	11					11
13	12					12
15	9	10				19
17	1	21				22
19		21	1			22
21		6	4			10
23		1	10			11
25			5			5
27			2	2		4
29				1		1
31				1		1
35					1	1
No. of Drawings	36	59	22	4	1	122

Chapter 10

Analysis of the Algorithm

An analysis of the preceding algorithm is presented in this chapter, whereby we show that the number of operations required by the algorithm is

$$O(p \times n \times 2^{\frac{n^2}{2}})$$

p being the number of drawings D_{n-1} used to generate the drawings D_n

For a given rectilinear drawing D_{n-1} with arcs (n−1,n−2), (n−1, n−3), . . . , (n−1,1) as shown in Fig.10.1, we generate the arcs (n,i), where i = 1, 2, . . . , n−2. A node n is placed in the area bounded by the extension of the arc (1,n−1) and by the arc (n−1,n−2) and its extension as shown in Fig.10.2. An arc (n,i) may cross any of the arcs (n−1,j) where $i < j \leq n-2$ as shown in Fig.10.3.

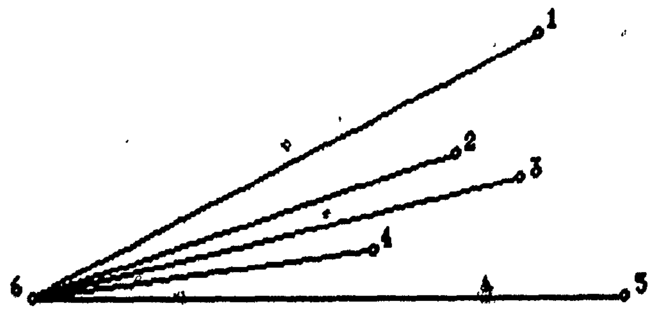


Fig.10.1: The arcs (6,i), i = 1,2,3,4,5 of a rectilinear drawing D_6.

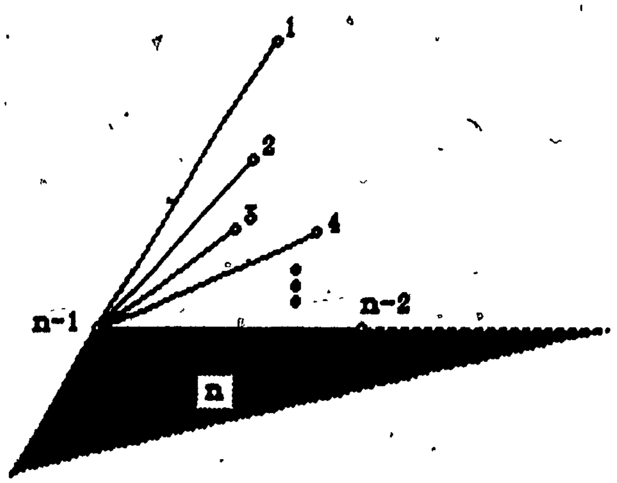


Fig.10.2: A node n is added in the shaded area.

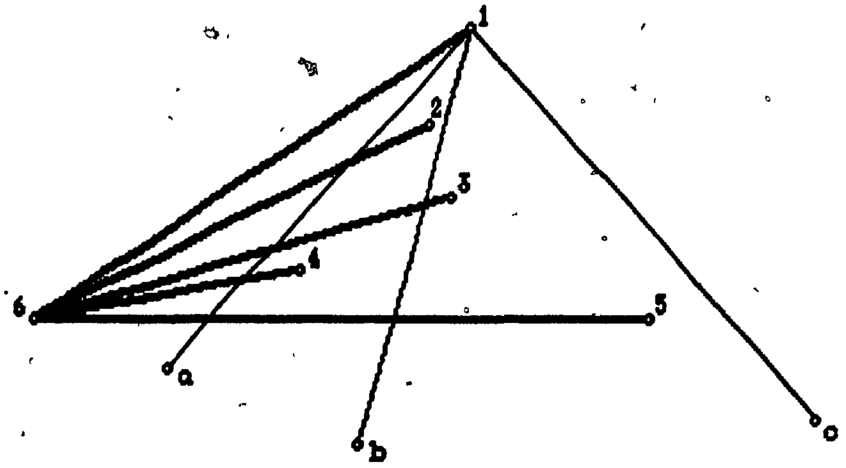


Fig.10.3: An arc (7,1) may cross all of the arcs (6,j) as in (a,1), or some of the arcs (6,j) as in (b,1), or none of them as in (c,1).

An upper bound on the number of drawings D_n which could be generated using a rectilinear drawing D_{n-1} is

$$k \times 2^{\frac{(m^2+m)}{2}}$$

where $m=n-3$ and k is the number of arcs of the convex hull of D_{n-1} as shown in Table 10.1.

n	$2^{\frac{(m^2+m)}{2}}$	D_n generated by Algorithm	Non-isomorphic rectilinear D_n
	(1)	(2)	
4	2	2	2
5	8	7	3
6	64	46	15
7	1024	608	122

Table 10.1:

Column (1) = the largest number of drawings D_n which could be generated using one rectilinear drawing D_{n-1}, $(m = n-3)$.

Column (2) = the number of drawings generated by the algorithm using all the non-equivalent rectilinear drawings D_{n-1}.

The number of D_n in column (2) are obtained when we take advantage of obvious symmetry which exists in some drawings D_{n-1}. The number of generated D_n is generally higher if symmetry is discarded, Table 10.2.

n	D_{n-1}	SYMMETRY DISCARDED		SYMMETRY TAKEN INTO ACCOUNT	
		D_{n-1} INPUT	GENERATED D_n	D_{n-1} INPUT	GENERATED D_n
	(1)	(2)	(3)	(4)	(5)
4	1	3	6	1	2
5	2	7	24	2	7
6	3	12	94	6	46
7	16	59	887	41	608

Table 10.2:

Column (1) =	Number of the non-equivalent rectilinear drawings D_{n-1}.
Column (2) =	Number of sets of crossings input to the algorithm, without taking into consideration the possible symmetry in a drawing. If a CH of a drawing has k arcs, then the crossings of this drawing will be input k times to the algorithm using k appropriate different sets of labels.
Column (3) =	Number of generated drawings D_n when possible symmetry in drawings is not considered.
Column (4) =	Number of sets of crossings input to the algorithm when obvious symmetry in a drawing is taken into account.
Column (5) =	Number of generated drawings D_n when obvious symmetry is taken into consideration.

We have shown that the potential number of drawings D_n to be generated for each D_{n-1} is

$$k \times 2^{\frac{(m^2+m)}{2}}$$

k being the number of arcs of the convex hull of D_{n-1} and $m = n-3$. In the following we look at the details of the algorithm to determine the number of operations required to generate each drawing D_n.

Step #	Step Description	Number of Operations
1	Input the data related to D_{n-1} which is used to generate the drawings D_n. This data includes the crossings of D_{n-1}	$O(n^4)$
2	Set the relationship between each pair of arcs $(n, n-k-2)$ and $(n-1, n-l-1)$, where $n-k-2 < n-l-1$	$O(2^m)$ $m = \frac{(n-3)^2+(n-3)}{2}$
3	Check whether the sub-drawing consisting of the nodes $r, s, t, n-1$, n and their corresponding arcs is rectilinear, where $1 \le r < s < t < n-1$	$O(n^3)$
4	Determine the remaining crossings of D_n	$O(n^7)$
5	Output the crossings of D_n	$O(n^4)$
6	Repeat steps 2 to 5 for each set of labels of D_{n-1}	
7	Repeat steps 1 to 6 for each drawing D_{n-1}.	

We go to Step 6 k times for each D_{n-1}, where k is the number of arcs of the convex hull of D_{n-1}. Step 7 is repeated p times where p is the number of drawings D_{n-1}. Hence, based on this and the number of operations at Step 2, we can conclude that the algorithm requires no more than $p \times n \times 2^m$.

Chapter 11

Conclusion

In this chapter we first summarize the results obtained in the thesis, then we shed light on some problems which could be considered for future research.

In the first part of the thesis, related to generating all the non-isomorphic drawings D_n of the complete graph K_n, we conjecture that any good drawing D_n of K_n has at least one C-F HC, C. An algorithm is developed to obtain all such drawings. The only input required by the algorithm is the dimension n. Upon reading n, the algorithm generates all the edges e_i of K_n different from the edges of C. For each edge e_i, the algorithm constructs all the arcs a_{ij} into which e_i can be mapped without violating any of the rules of a good drawing. The good drawings are then obtained and only the non-isomorphic good ones are kept by the algorithm. A computer program is written and results are obtained for n = 6 (102 drawings) and for n = 7 (11556 drawings). The drawings for n = 4,5,6 are produced manually in Appendix A.2. The complete set of all the non-isomorphic drawings D_7 of K_7 is generated by the program in list form in Appendix A.3. Results obtained by Newborn and Moser [19] for the n-circuit and the n-gon optimal drawings are confirmed for n = 7. Exactly two non-rectilinear drawings D_7 have the largest number of C-F HC's in any drawing D_7 of K_7. This number is 96. Only one rectilinear drawing D_7 has 92 C-F HC's which is the largest number of C-F HC's in any rectilinear drawing D_7 of K_7.

A characteristic of the rectilinear drawings D_n of K_n is obtained in the second part of the thesis. This characteristic is based on the existence of a specific drawing D_4 of K_4 as a sub-drawing in only the non-rectilinear drawings D_n of K_n. For convenience we call this D_4, A. This drawing A has a node, say v, located in the area bounded by the arcs (a,b), (b,c) and (a,c), such that (a,v) x (b,c), where a, b, and c are the three other nodes of A. Using this characteristic, an algorithm is developed to determine whether there exists a rectilinear drawing D_n of K_n which has a given set of crossings, X. The only inputs required by the algorithm are the dimension n and the set of crossings X. Upon reading n and X, the algorithm finds the uncrossed edges of K_n, all uncrossed k-circuits and then all the convex hulls, each of which has a corresponding drawing D_n of K_n having the set of crossings X. For each of these drawings D_n it verifies whether D_n has a sub-drawing equivalent to the drawing A. If one of these drawings has the characteristic of rectilinear drawings, i.e. does not have the drawing A as a sub-drawing, then the algorithm stops after printing the convex hull corresponding to this rectilinear drawing; otherwise it concludes that the set of crossings X does not determine any rectilinear drawing D_n of K_n. By applying this algorithm to the set of the non-isomorphic drawings D_6 obtained earlier in the thesis, we can determine that 15 out of the 102 drawings D_6 are rectilinear and 87 are non-rectilinear. We note that 66 of the 87 non-rectilinear drawings D_6 do not have a convex hull.

In the last part of the thesis, an algorithm is written to generate all the non-isomorphic rectilinear drawings D_n of K_n, using the non-equivalent rectilinear drawings D_{n-1} of K_{n-1}. The corresponding computer program is implemented for $n=7$. The sets of crossings of the 16 non-equivalent drawings D_6 of K_6 are used as input to the computer program. The complete set of all the non-isomorphic rectilinear drawings D_7, which consists of 122 drawings, is obtained. The actual drawings are displayed in Appendix C.2.

In the following paragraphs, we direct the reader's attention to some of the problems which could be viewed as possible research problems.

The first of these problems is to confirm the conjecture of Chapter 2 stating that: *Every good drawing D_n of K_n has at least one C-F HC.* While this conjecture can be easily confirmed for the rectilinear drawings D_n of K_n, it is apparently an extremely difficult problem to resolve for the case of non-rectilinear good drawings.

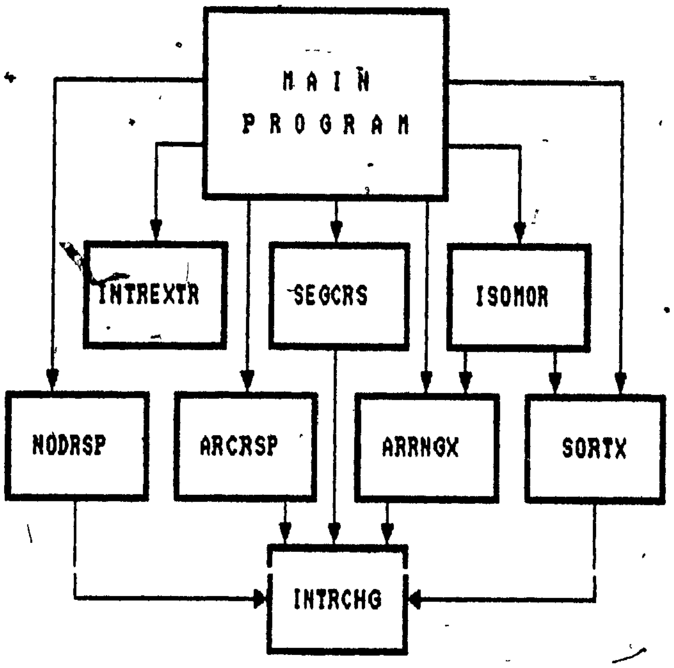
The algorithm generating all the non-isomorphic drawings D_n of K_n requires a large amount of time when implemented for $n=7$. Improvement in the efficiency of the algorithm, coupled with the increased speed of the new generation computers, might pave the way to implement the algorithm for larger values of n. Such improvement might be obtained if we can find a way to determine isomorphism for larger numbers of drawings without actually generating these drawings. We might then be able to obtain, for example, all the non-isomorphic drawings D_8 of K_8 or the complete set of all crossing optimal drawings D_9 of K_9 and D_{10} of K_{10}.

To determine whether there is a rectilinear drawing D_n of K_n having a given set of crossings, the algorithm of Chapter 5 must first find all the uncrossed k-circuits in order to obtain all the possible convex hulls. The large number of these uncrossed k-circuits reduces the efficiency of the algorithm. One way to eliminate this might be to obtain the convex hulls without requiring all the uncrossed k-circuits.

APPENDIX A.1

PROGRAM TO GENERATE ALL NON-ISOMORPHIC GOOD DRAWINGS D_n OF K_n

I. GENERAL DIAGRAM



MAIN : Generates all the non-isomorphic good drawings D_n of K_n.

INTREXTR: Selects the good arcs into which an edge e of E could be mapped, where E is the set of edges of K_n different from the edges of C and where C is a C-F HC of K_n.
(The set of these good arcs will be denoted by A.)

SEGCRS : Determines whether two arcs of A cross and whether their crossing will lead to a good drawing.

ISOMOR : Determines whether two drawings are isomorphic.

NODRSP : Calculates nodes' responsibilities.

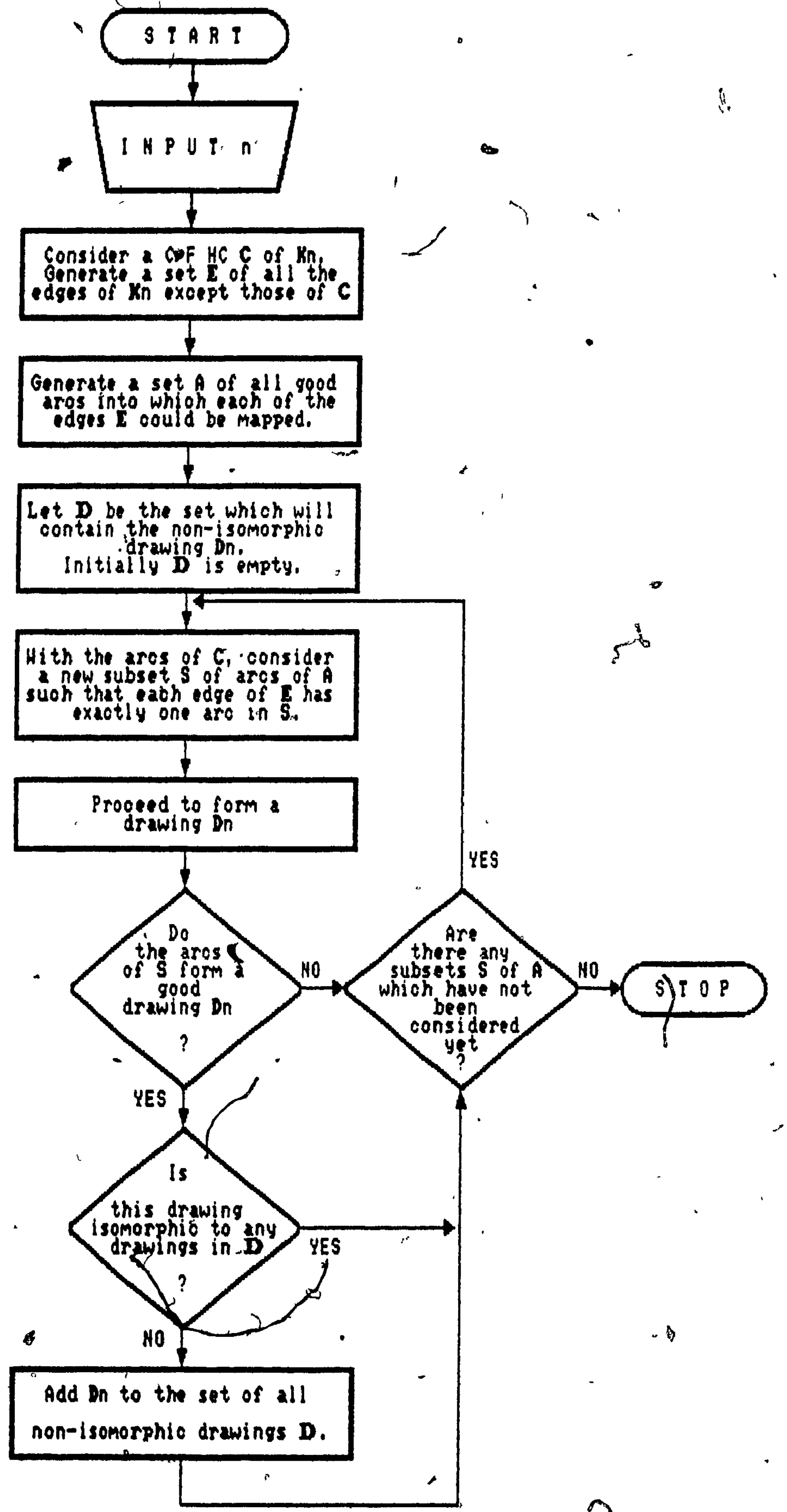
ARCRSP : Calculates arcs' responsibilities.

ARRNGX : Arranges the nodes corresponding to the crossing (a,b)x(c,d) such that: a<b , c<d , a<b.

SORTX : Sorts a set of crossings in ascending order such that if x_1=(a,b)x(c,d) and x_2=(e,f)x(g,h) then x_1 is listed before x_2 whenever the number ~~abcd~~ is smaller than the number efgh.
We note that n is assumed to be smaller than 10.
For n≥10, the subroutine will need some modifications.

INTRCHG : Interchanges the values of two variables p and q, such that p takes the value of q , and q takes the value of p.

II. GENERAL FLOWCHART


START
INPUT n
Consider a C≠F HC C of Kn. Generate a set E of all the edges of Kn except those of C
Generate a set A of all good arcs into which each of the edges E could be mapped.
Let D be the set which will contain the non-isomorphic drawing Dn. Initially D is empty.
With the arcs of C, consider a new subset S of arcs of A such that each edge of E has exactly one arc in S.
Proceed to form a drawing Dn
Do the arcs of S form a good drawing Dn ?
NO
YES
Are there any subsets S of A which have not been considered yet ?
YES
NO
STOP
Is this drawing isomorphic to any drawings in D ?
YES
NO
Add Dn to the set of all non-isomorphic drawings D.

III- COMPUTER PROGRAM (FORTRAN 77)

```
*************************************************************
*                                                           *
*                                                           *
*  Given sufficient space and time, this program will       *
*  generate all the good drawings Dn of Kn having at least  *
*  one C-F H C, C.  The only input required by the program  *
*  is n.                                                    *
*                                                           *
*                                                           *
*  The program consists of the following four parts :      *
*                                                           *
*  PART  I :    Given n, the program will generate all the  *
*               edges (a,b) which are different from the    *
*               edges of C.                                 *
*                                                           *
*  PART II :    For each edge (a,b), the program will       *
*               generate  all the good arcs into which      *
*               (a,b) could be mapped.                      *
*                                                           *
*  In Part I and Part II, the nodes of Kn are denoted by    *
*  the odd numbers 1,3,5,.... ,2n-3,2n-1 and the edges of   *
*  Kn are denoted by the even numbers 2,4,6, ... ,2n        *
*  where  i  is the edge (i-1,i+1) for 1 < i < 2n-1 , and   *
*  2n is the edge (2n-1,1). In both parts, the program      *
*  assumes that the vertices of Kn are labelled such        *
*  that the C-F HC, C, is (1,3,5, ... ,2n-3,2n-1, 1).       *
*                                                           *
*                                                           *
*  PART III:    For each pair of arcs corresponding to two  *
*               distinct edges (a,b) and (c,d), the program *
*               will determine whether (a,b) and (c,d)      *
*               cross each other, and whether they cross    *
*               more than once.                             *
*                                                           *
*  In this part of the program, the nodes  of Kn are        *
*  denoted by the odd numbers 1,5,9, ... ,4n-7,4n-3 such    *
*  such that C becomes (1,5,9, ... ,4n-7,4n-3,1).  Each arc *
*  c = (i,i+4)  has two points labelled i+1 and i+3, such   *
*  that whenever c is crossed, it will be crossed in either *
*  i+1 or i+3.                                              *
*                                                           *
*                                                           *
*  PART IV :    In this part of the program, all the        *
*               non-isomorphic drawings are generated.      *
*                                                           *
*                                                           *
*************************************************************
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                       *
*                                                       *
      IMPLICIT INTEGER*2 (A-Z)
*                                                       *
*                                                       *
*   EDGE(.,1) , EDGE(.,2) will contain the edges of Kn  *
*                         different from the edges of C *
*                         when the vertices of Kn are   *
*                         labeled 1,2,3,...,n           *
*                                                       *
*   EDGE2(.,1), EDGE2(.,2 ) will contain the same edges,*
*                           but when the vertices are   *
*                           labeled 1,5,7,...,2*n-1     *
*                                                       *
*   NEDGE(.) will refer to the edge number              *
*                                                       *
*   ADDT(i+1) = the number of good arcs into which all the *
*               i  edges could be mapped                *
*                                                       *
*   NUMARC(i+1) = the number of arcs related to any edge *
*                 belonging to the i-th group of edges. *
*                                                       *
      DIMENSION EDGE(14,2),EDGE2(14,2),NEDGE(294),ADDT(15)
      DIMENSION NUMARC(4)
*                                                       *
*                                                       *
*   V(.) will contain a good arc of Kn                  *
*                                                       *
*   ARCS(.) will contain an arc of Kn, usually not a good *
*        arc                                            *
*                                                       *
*   MARCS(.,.) will contain all good arcs of Kn ,       *
*              different from the arcs of C.            *
*                                                       *
      DIMENSION V(6),ARCS(6),MARCS(294,6)
*                                                       *
*   INDX(.) and IARC(.)  will be used to generate the   *
*                        drawings Dn                    *
*                                                       *
*   INDX(i) = the i-th arc of the drawing on hand       *
*                                                       *
      DIMENSION INDX(14),IARC(14)
*                                                       *
*   INTR(.,1) , INTR(.,2) will contain the segments of arc *
*                         located in Int C              *
*                                                       *
*   EXTR(.,1) , EXTR(.,2) will contain the segments of arc *
*                         located in Ext C              *
*                                                       *
      DIMENSION INTR(24,2),EXTR(24,2)
```

```
*                                                                    *
*   ARC1(.) , ARC2(.) = a pair of good arcs of Dn                   *
*                                                                    *
*   Each pair of arcs of Dn is considered in order to               *
*   determine whether Dn is a good drawing.                          *
*                                                                    *
      DIMENSION AR1(2,7),AR2(2,7),AR3(2,7),AR4(2,7)
      DIMENSION INDX1(7),INDX2(7),NXARCS(7)
      DIMENSION ARC1(7),ARC2(7)
*                                                                    *
*                                                                    *
*   MX(i,j) will contain a value indicating whether the             *
*           i-th arc crosses the j-th arc, and whether              *
*           they cross more than once                               *
*                                                                    *
      DIMENSION MX(294,294)
*                                                                    *
*                                                                    *
*   NODES(.) = nodes of the drawing on hand                         *
*                                                                    *
*   X1(.),X2(.),X3(.),X4(.) =  crossings of the drawing             *
*                              on hand                              *
*                                                                    *
*   RSPND(.) = nodes responsibilities                               *
*                                                                    *
*   RSPAR(.) = arcs responsibilities                                *
*                                                                    *
      DIMENSION NODES(7),X1(35),X2(35),X3(35),X4(35)
      DIMENSION RSPND(7),RSPAR(21)
*                                                                    *
*                                                                    *
*   SNODES(.) = nodes of the drawing to be compared against         *
*               the drawing on hand                                 *
*                                                                    *
*   SX1(.),SX2(.),SX3(.),SX4(.) =  crossings of the drawing         *
*                                  to be compared against           *
*                                  the drawing on hand              *
*                                                                    *
*   SRSPND(.) = nodes responsibilities of the drawing to be         *
*               compared against the drawing on hand                *
*                                                                    *
*   SRSPND(.) = nodes responsibilities of the drawing to be         *
*               compared against the drawing on hand                *
*                                                                    *
*   SINDX(i) = the i-th arc of the drawing to be compared           *
*              against the drawing on hand                          *
*                                                                    *
      DIMENSION SNODES(7),SX1(35),SX2(35),SX3(35),SX4(35)
      DIMENSION SRSPND(7),SRSPAR(21)
      DIMENSION SINDX(14)
```

```
*                                                                 *
*                                                                 *
*  MNX(i) = number of crossings of the i-th non-isomorphic        *
*           drawing                                               *
*                                                                 *
*  X1(i,.), ..., X4(i,.) = crossings of the i-th                  *
*                          non-isomorphic drawing                 *
*                                                                 *
*  MNODES(i,.),MRSPND(i,.),MRSPAR(i,.) = nodes, nodes and         *
*                          arcs responsibilities of the           *
*                          i-th non-isomorphic drawing            *
*  MINDX(i,j) = the j-th arc of the i-th non-isomorphic           *
*               drawing                                           *
*                                                                 *
      DIMENSION MNX(5000),
     *     MX1(35,5000),MX2(35,5000),
     *     MX3(35,5000),MX4(35,5000),
     *     MRSPND(7,5000),MNODES(7,5000),MRSPAR(21,5000),
     *     MINDX(14,5000)
*                                                                 *
*                                                                 *
*                                                                 *
*                                                                 *
*                                                                 *
*                                                                 *
*  NNX(i) = number of drawings with i crossings                   *
*                                                                 *
      DIMENSION NNX(35)
*                                                                 *
*                                                                 *
*    ISO1(.) will be used in comparing the first n                *
*            elements of INDX(.) against those of                 *
*            MINDX(.,.)                                           
*                                                                 *
      DIMENSION ISO1(7),ISO2(7),ISO3(7),ISO4(7),SYM(18)
*
      DATA SYM/1,2,8,7,6,5,4,3,13,14,11,12,9,10,16,15,18,17/
*
*
      WRITE(*,'(1X,'' I N P U T    N'')')
      READ(*,'(I2)') N
      NM1 = N-1
      NT2 = 2*N
      NM4 = N-4
```

```
*************************************************************
*************************************************************
********                                             ********
********              P A R T  I   -                 ********
********                                             ********
********    G E N E R A T E   T H E  E D G E S       ********
********                                             ********
*************************************************************
*************************************************************
*                                                           *
*                                                           *
*                                                           *
*     Given n, all the edges (a,b) different from the       *
*                                                           *
*     edges of C,  are generated.                           *
*                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
      NA = 0
      NAMAX = N*(N-1)/2 - N
      KA = 2
1000  DO 1100 I=1, N
         NA = NA+1
         EDGE(NA,1) = I
         EDGE(NA,2) = I+KA
         IF (EDGE(NA,2).GT.N) EDGE(NA,2) = EDGE(NA,2)-N
         EDGE2(NA,1) = EDGE(NA,1)*2-1
         EDGE2(NA,2) = EDGE(NA,2)*2-1
         IF (NA.EQ.NAMAX) GOTO 2000
1100  CONTINUE
      KA = KA+1
      GOTO 1000
```

```
**************************************************************
**************************************************************
*******                                                *******
*******                                                *******
*******                 P A R T   II                   *******
*******                                                *******
*******  G E N E R A T E   A L L   T H E   G O O D     *******
*******                                                *******
*******  A R C S   I N T O   W H I C H   T H E         *******
*******                                                *******
*******  E D G E S   ( A , B )   O F   K n             *******
*******                                                *******
*******  C O U L D   B E   M A P P E D ,   W H E R E   *******
*******                                                *******
*******  A   A N D   B   A R E   T W O   N O D E S     *******
*******                                                *******
*******  O F   C.                                      *******
*******                                                *******
*******                                                *******
**************************************************************
**************************************************************
*                                                            *
* In this part, the program generates all the good arcs     *
*                                                            *
* into which (a,b) could be mapped, by indicating the arcs  *
*                                                            *
* of C which are crossed by (a,b) and their order as        *
*                                                            *
* follows:                                                   *
*                                                            *
*          (a,c1,c2, ... , cp-1,cp,b)                        *
*                                                            *
* meaning that the segment of arc (a,c1) is in Int C, or    *
*                                                            *
*          (0,a,c1,c2, ... , cp-1,cp,b)                      *
*                                                            *
* meaning that the segment of arc (a,c1) is in Ext C.       *
*                                                            *
*                                                            *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
*
2000  ZERO = 0
      II = 0
      DO 2400 IK=1, KA-1
         MAXEDGE = N
         NADD = NADD + NUMARC(IK) * MAXEDGE
         IF (IK.EQ.KA-1) MAXEDGE = NAMAX-N*(KA-2)
         II = II + 1
         ADDT(II) = NARC
         A = EDGE2(II,1)
         B = EDGE2(II,2)
```

```
          WRITE(*,'(//)')
          WRITE(*,'(1X,'' EDGE ( '',I2,'','',I2,'')'')') A,B
          WRITE(*,'(1X,'' ===================='')')
          WRITE(*,'(1X,/'' CORRESPONDING ARCS '')')
          WRITE(*,'(1X,'' ------------------ '')')
          NAR = 0
          NAR = NAR+1
          WRITE(*,'(1X,I2,5X,2I3)') NAR,A,B
          NAR = NAR+1
          WRITE(*,'(1X,I2,5X,3I3)') NAR,ZERO,A,B
          DO 2010 I=1, NM1
             V(I) = 0
2010      CONTINUE
          NC = 0
          DO 2020 I=2, NT2, 2
             IF (IABS(A-I).NE.1 .AND. IABS(B-I).NE.1 .AND.
     *               IABS(A-I).NE.NT2-1 .AND.
     *               IABS(B-I).NE.NT2-1)    THEN
                NC = NC+1
                ARCS(NC) = I
             ENDIF
2020      CONTINUE
*
*
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                       *
* The edge (a,b) is mapped into an arc which crosses p  *
*                                                       *
* of the arcs of C.  These p arcs are stored in         *
*                                                       *
* V(2),V(3), ... ,V(p+1), while  a  and  b  are stored  *
*                                                       *
* in V(1) and V(p+2) respectively.                      *
*                                                       *
*                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
*
          NARC = NARC+1
          NUMARC(IK+1) = NUMARC(IK+1) + 1
          NEDGE(NARC) = II
          MARCS(NARC,1) = A
          MARCS(NARC,2) = B
          NARC = NARC+1
          NUMARC(IK+1) = NUMARC(IK+1) + 1
          NEDGE(NARC) = II
          MARCS(NARC,1) = 0
          MARCS(NARC,2) = A
          MARCS(NARC,3) = B
```

```
                DO 2300 M2=1, NM4
                   DO 2030 I=1, NM4
                      INDX(I) = 1
                      IARC(I) = NM4
          2030     CONTINUE
          2035     DO 2050 I=1, M2-1
                      DO 2040 J=I+1, M2
                         IF (INDX(I).EQ.INDX(J)) GOTO 2105
          2040        CONTINUE
          2050     CONTINUE
                   V(1) = A
                   DO 2060 I=2, M2+1
                      V(I) = ARCS(INDX(I-1))
          2060     CONTINUE
                   V(M2+2) = B
          *
          * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
          *                                                         *
          *        Checking whether V(.) is a good arc              *
          *                                                         *
          * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                   NX = 0
                   IN = 0
                   EX = 0
                   NSEG = N/2
                   DO 2070 K=1,NSEG+1
                      INTR(K,1) = 0
                      INTR(K,2) = 0
                      EXTR(K,1) = 0
                      EXTR(K,2) = 0
          2070     CONTINUE
          *                                                         *
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                             *
*    Splitting the arc V(.) into two sets of                  *
*    segments of arcs:                                        *
*                                                             *
*                                                             *
*    1.   segments falling in Int C, which are                *
*         represented by INTR(.,1) , INTR(.,2)  and           *
*                                                             *
*    2.   segments falling in Ext C, which are                *
*         represented by EXTR(.,1) , EXTR(.,2).               *
*                                                             *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
*
            DO 2080 K=1, NSEG+1, 2
               IN = IN+1
               INTR(IN,1) = V(K)
               INTR(IN,2) = V(K+1)
               EX = EX+1
               EXTR(EX,1) = V(K+1)
               EXTR(EX,2) = V(K+2)
2080        CONTINUE
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                             *
*  Each pair of segments of (a,b) located in                  *
*  Int C is considered in order to determine the              *
*  existence of a crossing.  If a crossing is                 *
*  found, then V(.) is rejected.                              *
*                                                             *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
            CALL INTREXTR(NT2,NSEG,INTR,NX)
            IF (NX.GT.0) GOTO 2105
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                             *
*  Each pair of segments of (a,b) located in                  *
*  Ext C is considered in order to determine the              *
*  existence of a crossing.  If a crossing is                 *
*  found, then V(.) is rejected.                              *
*                                                             *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
            CALL INTREXTR(NT2,NSEG,EXTR,NX)
            IF (NX.GT.0) GOTO 2105
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                         *
*  Here V(i) is a good arc.  It is then stored in the     *
*  matrix MARCS(.,.) which contains all the good arcs     *
*  of (a,b), and which will be used as input to the part  *
*  of the program generating all the non-isomorphic good  *
*  drawings of the complete graph, having at least one    *
*  C-F H C.                                               *
*                                                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
               NAR = NAR+1
               NARC = NARC+1
               NUMARC(IK+1) = NUMARC(IK+1) + 1
               NEDGE(NARC) = II
               DO 2090 I=1, M2+2
                  MARCS(NARC, I) = V(I)
2090           CONTINUE
               WRITE(*,'(1X,I2,5X,10I3)')
     *                    NAR,(V(I), I=1,M2+2)
               NARC = NARC+1
               NUMARC(IK+1) = NUMARC(IK+1) + 1
               NEDGE(NARC) = II
               MARCS(NARC,1) = 0
               NAR = NAR+1
               DO 2100 I=2, M2+3
                  MARCS(NARC, I)=V(I-1)
2100           CONTINUE
               WRITE(*,'(1X,I2,5X,11I3)')
     *                    NAR,ZERO,(V(I-1), I=2,M2+3)
2105           IF (INDX(M2).LT.IARC(M2)) GOTO 2110
               LAST = M2
               GOTO 2120
2110           INDX(M2) = INDX(M2)+1
               GOTO 2035
2115           IF (INDX(LAST).LT.IARC(LAST)) GOTO 2130
2120           IF (LAST.EQ.1) GOTO 2300
               LAST = LAST-1
               GOTO 2115
2130           INDX(LAST) = INDX(LAST)+1
               DO 2200 I=LAST+1, NM4
                  INDX(I) = 1
2200           CONTINUE
               GOTO 2035
2300        CONTINUE
            DO 2350 I =1,MAXEDGE-1
               II = II + 1
               ADDT(II) = NARC
               A = EDGE2(II,1)
               B = EDGE2(II,2)
               WRITE(*,'(//)')
             WRITE(*,'(1X,'' EDGE ( '',I2,'','',I2,'')'')') A,B
             WRITE(*,'(1X,'' ===================='')')
             WRITE(*,'(1X,/'' CORRESPONDING ARCS '')')
             WRITE(*,'(1X,'' ------------------ '')')
```

```
                 NAR = 0
                 DO 2340 I1 =1, NUMARC(IK+1)
                    NARC = NARC+1
                    NEDGE(NARC) = II
                    NAR = NAR+1
                    DO 2330 J1 =1, N
                       IF (MARCS(I1+NADD,J1).NE.0) THEN
                          MARCS(NARC,J1) = MARCS(I1+NADD,J1)+2*I
                          IF (MARCS(NARC,J1).GT.N*2)
     *                        MARCS(NARC,J1) = MARCS(NARC,J1)-N*2
                       ENDIF
 2330               CONTINUE
                    WRITE(*,'(1X,I2,5X,10I3)')
     *                    NAR, (MARCS(NARC,J1), J1=1,NM1)
 2340            CONTINUE
 2350         CONTINUE
 2400      CONTINUE
```

```
************************************************************
************************************************************
************                                     ***********
************           P A R T  III              ***********
************                                     ***********
************     G E N E R A T E  T H E          ***********
************                                     ***********
************     M A T R I X   M X( . , . )      ***********
************                                     ***********
************************************************************
************************************************************
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                         *
*  Given all the arcs corresponding to each of the edges, *
*  a matrix MX(.,.) with entries 0 , 1 or 2 , is produced.*
*                                                         *
*                              (  0  if arc i and arc j   *
*                              (     do not cross         *
*                              (                          *
*        M X ( i, j )   =      (  1  if arc i and arc j   *
*                              (     cross exactly once,  *
*                              (     and if they do not   *
*                              (     have a common node   *
*                              (                          *
*                              (  2  otherwise            *
*                                                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      DO 2999 I=1,294
         DO 2998 J=1,294
            MX(I,J) = 0
2998     CONTINUE
2999  CONTINUE
      DO 3800 II=1, NARC
         DO 3700 JJ=1, NARC
            IF (NEDGE(II).EQ.NEDGE(JJ)) GOTO 3700
            DO 3000 K=1, NM1
               AR1(1,K)  = 0
               AR1(2,K)  = 0
               AR2(1,K)  = 0
               AR2(2,K)  = 0
               INDX1(K)  = 1
               INDX2(K)  = 1
               NXARCS(K) = 1
3000        CONTINUE
```

```
           DO 3002 K=1, NM1
              A = MARCS(II,K)
              IF (A.EQ.0) GOTO 3001
              IF (A/2*2.NE.A) AR1(1,K) = A*2 - 1
              IF (A/2*2.EQ.A) THEN
                 AR1(1,K) = A*2
                 AR1(2,K) = AR1(1,K)-2
              ENDIF
3001          A = MARCS(JJ,K)
              IF (A.EQ.0) GOTO 3002
              IF (A/2*2.NE.A) AR2(1,K) = A*2 - 1
              IF (A/2*2.EQ.A) THEN
                 AR2(1,K) = A*2
                 AR2(2,K) = AR2(1,K)-2
              ENDIF
3002       CONTINUE
           DO 3004 I=1, NM1
              IF (AR1(2,I).NE.0) THEN
                 DO 3003 J=1, NM1
                    IF (AR2(2,J).NE.0) THEN
                       IF (AR1(2,I).EQ.AR2(2,J)) GOTO 3004
                    ENDIF
3003             CONTINUE
                 AR1(2,I) = 0
              ENDIF
3004       CONTINUE
           DO 3006 I=1, NM1
              IF (AR2(2,I).NE.0) THEN
                 DO 3005 J=1, NM1
                    IF (AR1(2,J).NE.0) THEN
                       IF (AR2(2,I).EQ.AR1(2,J)) GOTO 3006
                    ENDIF
3005             CONTINUE
                 AR2(2,I) = 0
              ENDIF
3006       CONTINUE
```

```
          MINX = 100
          DO 3007 K=1, NM1
             IF (AR1(2,K).NE.0) NXARCS(K) = 2
3007      CONTINUE
3008      DO 3010 I=1, NM1
             A = AR1(INDX1(I),I)
             IF (A.NE.0 .AND. A/2*2.EQ.A) THEN
                DO 3009 J=1, NM1
                   IF (A.EQ.AR2(INDX2(J),J)) THEN
                      INDX2(J) = INDX2(J) +1
                      IF (INDX2(J).EQ.3) INDX2(J) = 1
                      GOTO 3010
                   ENDIF
3009            CONTINUE
             ENDIF
3010      CONTINUE
3012      DO 3020 K=1, NM1
             ARC1(K) = AR1(INDX1(K),K)
             ARC2(K) = AR2(INDX2(K),K)
3020      CONTINUE
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                           *
*    Given two arcs corresponding to two edges,             *
*    each of these is divided into two sets of              *
*    segments.                                              *
*    The matrices INTR(.,1), INTR(.,2) will contain         *
*    the segments located in Int C, while EXTR(.,1),        *
*    EXTR(.,2) will contain the segments located in         *
*    Ext C.                                                 *
*                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
            NX = 0
            IN = 0
            EX = 0
            DO 3300 K=1, 2*NM4+1
               INTR(K,1) = 0
               INTR(K,2) = 0
               EXTR(K,1) = 0
               EXTR(K,2) = 0
3300        CONTINUE
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                           *
*         Storing the segments of the first arc             *
*                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*

            DO 3400 K=1, N-2, 2
               IN = IN+1
               INTR(IN,1) = ARC1(K)
               INTR(IN,2) = ARC1(K+1)
               EX = EX+1
               EXTR(EX,1) = ARC1(K+1)
               EXTR(EX,2) = ARC1(K+2)
3400        CONTINUE
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                           *
*         Storing the segments of the second arc            *
*                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
            DO 3410 K=1, N-2, 2
               IN = IN+1
               INTR(IN,1) = ARC2(K)
               INTR(IN,2) = ARC2(K+1)
               EX = EX+1
               EXTR(EX,1) = ARC2(K+1)
               EXTR(EX,2) = ARC2(K+2)
3410        CONTINUE
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                           *
*    Determining whether the segments in Int C cross        *
*                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
            CALL SEGCRS(N,INTR,NX)
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
*       Determining whether the segments in Ext C cross               *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
            CALL SEGCRS(N, EXTR, NX)
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
            IF (NX.LT.MINX) MINX = NX
            IF (MINX.EQ.0) GOTO 3650
            IF (INDX1(NM1).EQ.NXARCS(NM1)) THEN
               LAST = NM1
               GOTO 3425
            ENDIF
            INDX1(NM1) = INDX1(NM1)+1
            GOTO 3008
3420        IF (INDX1(LAST).LT.NXARCS(LAST)) GOTO 3435
            IF (LAST.EQ.0) GOTO 3600
3425        LAST = LAST-1
            GOTO 3420
3435        INDX1(LAST) = INDX1(LAST)+1
            DO 3437 I=LAST+1, NM1
               INDX1(I) = 1
3437        CONTINUE
            GOTO 3008
3600        IF (MINX.EQ.0) GOTO 3650
            IF (MINX.GT.1) THEN
               MINX = 2
               GOTO 3650
            ENDIF
            NII = NEDGE(II)
            NJJ = NEDGE(JJ)
            IF (EDGE(NII,1).EQ.EDGE(NJJ,1) .OR.
     *          EDGE(NII,1).EQ.EDGE(NJJ,2) .OR.
     *          EDGE(NII,2).EQ.EDGE(NJJ,1) .OR.
     *          EDGE(NII,2).EQ.EDGE(NJJ,2))  MINX = 2
3650        MX(II,JJ) = MINX
3700     CONTINUE
3800  CONTINUE
```

```
****************************************************************
****************************************************************
****************************************************************
******                                                    ******
******                    P A R T   IV                    ******
******                                                    ******
******    G E N E R A T E   T H E   D R A W I N G S       ******
******                                                    ******
******    U S I N G   T H E   I N F O R M A T I O N       ******
******                                                    ******
******    I N   M A T R I X   M X ( . , . ) .             ******
******                                                    ******
****************************************************************
****************************************************************
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
*  M = number of arcs of Kn = n*(n-1)/2                        *
*  M1= number of arcs different from the arcs of C             *
*    = M - n                                                   *
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
      M   = N*(N-1)/2
      M1  = N*(N-1)/2-N
      ADDT(M1+1) = NARC
      ND = 0
      DO 4000 I=1, M1
         INDX(I) = 1
         IARC(I) = ADDT(I+1)-ADDT(I)
4000  CONTINUE
4005  NX = 0
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
*                                                              *
*  Forming the crossings occuring between the edges (a,b)     *
*                                                              *
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
      DO 4040 K1=1, M1
         I1 = INDX(K1)+ADDT(K1)
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                         *
*        Forming the crossings occuring between           *
*        the edges (a,b) and the edges of C.              *
*                                                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         DO 4020 K=1, NM1
            A = MARCS(I1,K)
            IF (A.EQ.0 .OR.  A/2*2.NE.A) GOTO 4020
            NX = NX+1
            IF (NX.GT.MAXCRS) THEN
               LAST = K1
               GOTO 4115
            ENDIF
            X1(NX) = EDGE(K1,1)
            X2(NX) = EDGE(K1,2)
            X3(NX) = A/2
            X4(NX) = A/2 + 1
            IF (X4(NX).EQ.N+1) X4(NX) = 1
4020     CONTINUE
         DO 4030 K2=1, K1-1
            I2 = INDX(K2)+ADDT(K2)
            IF (MX(I1,I2).EQ.2) THEN
               LAST = K1
               GOTO 4115
            ENDIF
            IF (MX(I1,I2).EQ.0)GOTO 4030
            NX = NX+1
            IF (NX.GT.MAXCRS) THEN
               LAST = K1
               GOTO 4115
            ENDIF
            X1(NX) = EDGE(K1,1)
            X2(NX) = EDGE(K1,2)
            X3(NX) = EDGE(K2,1)
            X4(NX) = EDGE(K2,2)
4030     CONTINUE
4040  CONTINUE
*
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                         *
*      Arranging each crossing (a,b) x (c,d) such         *
*      that a < b , c < d , and a < c ;  and              *
*      sorting the crossings in ascending order.          *
*                                                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
      CALL ARRNGX(NX,X1,X2,X3,X4)
      CALL SORTX(NX,X1,X2,X3,X4)
*
```

```
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
*                                                                                  *
*      Obtaining nodes and arcs responsibilities                                   *
*                                                                                  *
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
*
         CALL NODRSP(N,NX,X1,X2,X3,X4,RSPND,NODES)
         CALL ARCRSP(N,M,NX,X1,X2,X3,X4,RSPAR)
*
*
         IF (ND.GT.O) THEN
            ISO = 0
            DO 4070 I=1, ND
               SNX = MNX(I)
               DO 4042 J=1,SNX
                  SX1(J) = MX1(J,I)
                  SX2(J) = MX2(J,I)
                  SX3(J) = MX3(J,I)
                  SX4(J) = MX4(J,I)
 4042          CONTINUE
               DO 4044 J=1, N
                  SRSPND(J) = MRSPND(J,I)
                  SNODES(J) = MNODES(J,I)
 4044          CONTINUE
               DO 4046 J=1, M
                  SRSPAR(J) = MRSPAR(J,I)
 4046          CONTINUE
               DO 4048 J=1, M1
                  SINDX(J) = MINDX(J,I)
 4048          CONTINUE
               IF (SNX.EQ.NX ) THEN
                  DO 4050 J=1, N
                     IF (SRSPND(J).NE.RSPND(J)) GOTO 4070
 4050             CONTINUE
                  DO 4060 J=1, M
                     IF (SRSPAR(J).NE.RSPAR(J)) GOTO 4070
 4060             CONTINUE
                  CALL ISOMOR
     *                (N,NX,RSPND,NODES,SNODES,X1,X2,X3,X4,
     *                                SX1,SX2,SX3,SX4,ISO)
                  IF (ISO.EQ.1) GOTO 4105
               ENDIF
 4070       CONTINUE
         ENDIF
         NNX(NX) = NNX(NX) + 1
         ND = ND + 1
         MNX(ND) = NX
         DO 4080 J=1, NX
            MX1(J,ND) = X1(J)
            MX2(J,ND) = X2(J)
            MX3(J,ND) = X3(J)
            MX4(J,ND) = X4(J)
 4080    CONTINUE
```

```
        DO 4082 J=1, N
           MRSPND(J,ND) = RSPND(J)
           MNODES(J,ND) = NODES(J)
 4082   CONTINUE
        DO 4084 J=1, M
           MRSPAR(J,ND) = RSPAR(J)
 4084   CONTINUE
        DO 4086 J=1, M1
           MINDX(J,ND) = INDX(J)
 4086   CONTINUE
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                       *
*        Displaying the crossings of the                *
*        non-isomorphic drawings                        *
*                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
        WRITE(*,'(1X,'' DRAWING #  '',I5,4X,15I3)') ND,
     *               (INDX(I), I=1, M1),NX
        DO 4100 I=1, NX
           WRITE(*,'(1X,I3,5X,''('',I2,'','',I2,'') X ('',
     *                      I2,'','',I2,'')'')')
     *               I,X1(I),X2(I),X3(I),X4(I)
 4100   CONTINUE
        WRITE(*,'(1X,14I4)') (NNX(I), I=9, 35 ,2)
 4105   IF (INDX(M1).LT.IARC(M1)) GOTO 4110
        LAST = M1
        GOTO 4120
 4110   INDX(M1) = INDX(M1)+1
        GOTO 4005
 4115   IF (INDX(LAST).LT.IARC(LAST)) GOTO 4200
 4120   IF (LAST.EQ.1) GOTO 9999
        LAST = LAST-1
        GOTO 4115
 4200   INDX(LAST) = INDX(LAST)+1
        DO 4205 I=LAST+1, M1
           INDX(I) = 1
 4205   CONTINUE
        IF (LAST.LT.3) WRITE(*,'(1X,'' I1  I2 '',2I3)')
     *          INDX(1),INDX(2)
        IF (LAST.LE.N) THEN
 4210      MAX1 = INDX(N)
           DO 4250 J=1, N-1
              IF (MAX1.LT.INDX(J)) MAX1 = INDX(J)
 4250      CONTINUE
           INDX(N) = MAX1
           DO 4300 J=1, N
              P = INDX(J)
              ISO1(J) = P
              POSNEG = 1
              IF (P/2*2.EQ.P) POSNEG = -1
              ISO2(J) = P + POSNEG
 4300      CONTINUE
           MAX2 = ISO2(N)
```

```
            DO 4304 J=1,N-1
               IF (MAX2.LT.ISO2(J)) MAX2 = ISO2(J)
 4304       CONTINUE
            DO 4350 KJ=1, N-1
               T = ISO1(1)
               DO 4310 J=1, N-1
                  ISO1(J) = ISO1(J+1)
 4310          CONTINUE
               ISO1(N) = T
               IF (T.EQ.MAX1) THEN
                  DO 4322 J=1, ND
                     DO 4320 J1=1, N
                        IF (ISO1(J1).NE.MINDX(J1,J)) GOTO 4322
 4320                CONTINUE
                     LAST = N
 4321                IF (INDX(LAST).LT.IARC(LAST)) THEN
                        INDX(LAST) = INDX(LAST) +1
                        GOTO 4210
                     ENDIF
                     IF (LAST.EQ.1) GOTO 9999
                     LAST = LAST - 1
                     GOTO 4321
 4322             CONTINUE
               ENDIF
               T = ISO2(1)
               DO 4323 J=1, N-1
                  ISO2(J) = ISO2(J+1)
 4323          CONTINUE
               ISO2(N) = T
               IF (T.EQ.MAX2) THEN
                  DO 4328 J=1, ND
                     DO 4325 J1=1, N
                        IF (ISO2(J1).NE.MINDX(J1,J)) GOTO 4328
 4325                CONTINUE
                     LAST = N
 4326                IF (INDX(LAST).LT.IARC(LAST)) THEN
                        INDX(LAST) = INDX(LAST) +1
                        GOTO 4210
                     ENDIF
                     IF (LAST.EQ.1) GOTO 9999
                     LAST = LAST - 1
                     GOTO 4326
 4328             CONTINUE
               ENDIF
 4350       CONTINUE
*.......................................................
            DO 4400 J=1, N
               P = SYM(INDX(J))
               ISO3(N+1-J) = P
               POSNEG = 1
               IF (P/2*2.EQ.P) POSNEG = -1
               ISO4(N+1-J) = P + POSNEG
 4400       CONTINUE
            MAX3 = ISO3(N)
```

```
           DO 4402 J=1, N-1
              IF (MAX3.LT.ISO3(J)) MAX3 = ISO3(J)
 4402      CONTINUE
           MAX4 = ISO4(N)
           DO 4404 J=1, N-1
              IF (MAX4.LT.ISO4(J)) MAX4 = ISO4(J)
 4404      CONTINUE
           DO 4450 KJ=1, N
              T = ISO3(1)
              DO 4410 J=1, N-1
                 ISO3(J) = ISO3(J+1)
 4410         CONTINUE
              ISO3(N) = T
              IF (T.EQ.MAX3) THEN
                 DO 4422 J=1, ND
                    DO 4420 J1=1, N
                       IF (ISO3(J1).NE.MINDX(J1,J)) GOTO 4422
 4420               CONTINUE
                    LAST = N
 4421               IF (INDX(LAST).LT.IARC(LAST)) THEN
                       INDX(LAST) = INDX(LAST) +1
                       GOTO 4210
                    ENDIF
                    IF (LAST.EQ.1) GOTO 9999
                    LAST = LAST - 1
                    GOTO 4421
 4422            CONTINUE
              ENDIF
              T = ISO4(1)
              DO 4423 J=1, N-1
                 ISO4(J) = ISO4(J+1)
 4423         CONTINUE
              ISO4(N) = T
              IF (T.EQ.MAX4) THEN
                 DO 4428 J=1, ND
                    DO 4425 J1=1, N
                       IF (ISO4(J1).NE.MINDX(J1,J)) GOTO 4428
 4425               CONTINUE
                    LAST = N
 4426               IF (INDX(LAST).LT.IARC(LAST)) THEN
                       INDX(LAST) = INDX(LAST) +1
                       GOTO 4210
                    ENDIF
                    IF (LAST.EQ.1) GOTO 9999
                    LAST = LAST - 1
                    GOTO 4426
 4428            CONTINUE
              ENDIF
 4450      CONTINUE
        ENDIF
        GOTO 4005
 9999   WRITE(*,'(1X,''TOTAL NUMBER OF DRAWINGS =  '',I5)') ND
        END
```

```
*****************************************************************
******************  S U B R O U T I N E  ************************
*****************************************************************
* This subroutine determines whether an arc is good.  It        *
* considers the segments of the arcs located in Int C and       *
* the segments in Ext C.  If a crossing is found in either      *
* set of segments then the arc on hand is not a good arc.       *
*****************************************************************
      SUBROUTINE INTREXTR(NT2,NSEG,INEX,NX)
      IMPLICIT INTEGER*2 (A-Z)
      DIMENSION INEX(24,2)
      DO 20 K=1, NSEG
         C1 = INEX(K,1)
         C2 = INEX(K,2)
         IF (C1.NE.0 .AND. C2.NE.0) THEN
            DO 10 K1=K+1, NSEG
               A1 = C1
               A2 = C2
               B1 = INEX(K1,1)
               B2 = INEX(K1,2)
               IF (B1.NE.0 .AND. B2.NE.0) THEN
                  A1 = 0
                  A2 = A2-C1
                  IF (A2.LT.0) A2 = A2+NT2
                  B1 = B1-C1
                  IF (B1.LT.0) B1 = B1+NT2
                  B2 = B2-C1
                  IF (B2.LT.0) B2 = B2+NT2
                  IF ((A1.LT.B1 .AND. B1.LT.A2
     *                             .AND. A2.LT.B2) .OR.
     *                (A1.LT.B2 .AND. B2.LT.A2
     *                             .AND.A2.LT.B1))
     *                NX = NX+1
               ENDIF
10          CONTINUE
         ENDIF
20    CONTINUE
      RETURN
      END
*****************************************************************
*****************  S U B R O U T I N E  *************************
*****************************************************************
*          Interchange the values of two variables              *
*****************************************************************
      SUBROUTINE INTRCHG(A,B)
      INTEGER*2 A,B,T
      T = A
      A = B
      B = T
      RETURN
      END
```

```
************************************************************
*****************  S U B R O U T I N E  ********************
************************************************************
*                                                          *
*  This subroutine calculates the number of crossings      *
*  occuring between the segments of arcs located on one    *
*  side of C.                                              *
*                                                          *
************************************************************
      SUBROUTINE SEGCRS(N,INEX,NX)
      IMPLICIT INTEGER*2 (A-Z)
      DIMENSION INEX(24,2)
      NT4 = 4*N
      DO 20 K=1, 2*(N-3)-1
         C1 = INEX(K,1)
         C2 = INEX(K,2)
         IF (C1.NE.0 .AND. C2.NE.0) THEN
            IF (C2.LT.C1) CALL INTRCHG(C1,C2)
            DO 10 K1=K+1, 2*(N-3)
               A1 = C1
               A2 = C2
               B1 = INEX(K1,1)
               B2 = INEX(K1,2)
               IF (B1.NE.0 .AND. B2.NE.0) THEN
                  A1 = 0
                  A2 = A2-C1
                  B1 = B1-C1
                  B2 = B2-C1
                  IF (A2.LT.0) A2 = A2+NT4
                  IF (B1.LT.0) B1 = B1+NT4
                  IF (B2.LT.0) B2 = B2+NT4
                  IF ((A1.LT.B1 .AND. B1.LT.A2
     *                    .AND. A2.LT.B2)        .OR.
     *                 (A1.LT.B2 .AND. B2.LT.A2
     *                    .AND.  A2.LT.B1))   NX = NX+1
               ENDIF
10          CONTINUE
         ENDIF
20    CONTINUE
      RETURN
      END
```

6

```
**********************************************************
*****************  S U B R O U T I N E  ******************
**********************************************************
*                                                        *
* Nodes responsibilities are calculated here.  These are *
* sorted in descending order and their corresponding nodes *
* are rearranged accordingly.                            *
*                                                        *
**********************************************************
      SUBROUTINE NODRSP(N,NX,X1,X2,X3,X4,RSPND,NODES)
      IMPLICIT INTEGER*2 (A-Z)
      DIMENSION X1(NX),X2(NX),X3(NX),X4(NX)
      DIMENSION RSPND(N),NODES(N)
      DO 10 J=1, N
         RSPND(J) = 0
10    CONTINUE
      DO 30 I=1, NX
         DO 20 J=1, N
            IF (X1(I).EQ.J .OR.
     *           X2(I).EQ.J .OR.
     *           X3(I).EQ.J .OR.
     *           X4(I).EQ.J)    RSPND(J) = RSPND(J)+1
20       CONTINUE
30    CONTINUE
      DO 40 I=1, N
         NODES(I) = I
40    CONTINUE
      DO 60 I=1, N-1
         DO 50 J=I+1, N
            IF (RSPND(I).LT.RSPND(J)) THEN
               CALL INTRCHG(RSPND(I),RSPND(J))
               CALL INTRCHG(NODES(I),NODES(J))
            ENDIF
50       CONTINUE
60    CONTINUE
      RETURN
      END
```

```
************************************************************
*****************  S U B R O U T I N E  *******************
************************************************************
*                                                          *
*  Arcs responsibilities are calculated then arranged in   *
*  descending order.                                       *
*                                                          *
************************************************************
      SUBROUTINE ARCRSP(N,M,NX,X1,X2,X3,X4,RSPAR)
      IMPLICIT INTEGER*2 (A-Z)
      DIMENSION X1(NX),X2(NX),X3(NX),X4(NX)
      DIMENSION RSPAR(M)
      DO 10 J=1, M
         RSPAR(J) = 0
10    CONTINUE
      DO 40 I=1, NX
         JK = 0
         DO 30 J=1, N-1
            DO 20 K=J+1, N
               JK = JK+1
               IF ((X1(I).EQ.J .AND. X2(I).EQ.K) .OR.
     *              (X3(I).EQ.J .AND. X4(I).EQ.K))
     *                RSPAR(JK) = RSPAR(JK)+1
20          CONTINUE
30       CONTINUE
40    CONTINUE
      DO 60 I=1, M -1
         DO 50 J=I+1, M
            IF (RSPAR(I).LT.RSPAR(J))
     *              CALL INTRCHG(RSPAR(I),RSPAR(J))
50       CONTINUE
60    CONTINUE
      RETURN
      END
*************************************************************
*****************  S U B R O U T I N E  ********************
*************************************************************
*      A crossing (a,b)x(c,d) is arranged such that         *
*             a < b ,  c < d  and  a < c                    *
*************************************************************
      SUBROUTINE ARRNGX(NX,X1,X2,X3,X4)
      IMPLICIT INTEGER*2 (A-Z)
      DIMENSION X1(NX),X2(NX),X3(NX),X4(NX)
      DO 10 I=1, NX
         IF (X1(I).GT.X2(I)) CALL INTRCHG(X1(I),X2(I))
         IF (X3(I).GT.X4(I)) CALL INTRCHG(X3(I),X4(I))
         IF (X1(I).GT.X3(I)) THEN
            CALL INTRCHG(X1(I),X3(I))
            CALL INTRCHG(X2(I),X4(I))
         ENDIF
10    CONTINUE
      RETURN
      END
```

```
**************************************************************
*****************  S U B R O U T I N E  **********************
**************************************************************
*                                                            *
* The crossings of a drawing are sorted in ascending order   *
*                                                            *
**************************************************************
      SUBROUTINE SORTX(NX,X1,X2,X3,X4)
      IMPLICIT INTEGER*2 (A-Z)
      DIMENSION X1(NX),X2(NX),X3(NX),X4(NX)
      DO 30 I=1, NX-1
         DO 20 J=I+1, NX
            IF (X1(I).LT.X1(J)) GOTO 20
            IF (X1(I).GT.X1(J)) GOTO 10
            IF (X2(I).LT.X2(J)) GOTO 20
            IF (X2(I).GT.X2(J)) GOTO 12
            IF (X3(I).LT.X3(J)) GOTO 20
            IF (X3(I).GT.X3(J)) GOTO 14
            IF (X4(I).LT.X4(J)) GOTO 20
            IF (X4(I).GT.X4(J)) GOTO 16
10          CALL INTRCHG(X1(I),X1(J))
12          CALL INTRCHG(X2(I),X2(J))
14          CALL INTRCHG(X3(I),X3(J))
16          CALL INTRCHG(X4(I),X4(J))
20       CONTINUE
30    CONTINUE
      RETURN
      END
```

```
*************************************************************
*****************  S U B R O U T I N E  *******************
*************************************************************
*                                                           *
*     Two drawings are compared for isomorphism.  Node      *
*     responsibilities are used to reduce the number        *
*     of comparisons.                                       *
*                                                           *
*     Variable ISO takes the value 1 whenever the two       *
*     drawings are isomorphic, otherwise it retains its     *
*     original value of zero.                               *
*                                                           *
*************************************************************
      SUBROUTINE ISOMOR(N,NX,RSPND,NODES,SNODES,
     *                           X1,X2,X3,X4,
     *                           SX1,SX2,SX3,SX4,ISO)
      IMPLICIT INTEGER*2 (A-Z)
      DIMENSION RSPND(N),NODES(N),SNODES(N)
      DIMENSION X1(NX),X2(NX),X3(NX),X4(NX)
      DIMENSION SX1(NX),SX2(NX),SX3(NX),SX4(NX)
*                                                           *
* PR(.),PRM(.),MP(.,.),MINI(.),MAXI(.) are used to generate*
*                                        new nodes' labels  *
*                                                           *
*
      DIMENSION PR(7),PRM(7),MP(7,7),MINI(14),MAXI(14)
*                                                           *
*  Y1(.),Y2(.),Y3(.),Y4(.) = crossings after relabelling    *
*                            the nodes                      *
*                                                           *
      DIMENSION Y1(35),Y2(35),Y3(35),Y4(35)
      DO 100 L=1, N
         PR(L) = 0
100   CONTINUE
      J = 1
      PR(1) = 1
      DO 110 L=2, N
         IF (RSPND(L-1).GT.RSPND(L)) THEN
            J = J+1
            PR(J) = 1
            GOTO 110
         ENDIF
         PR(J) = PR(J)+1
110   CONTINUE
      S = 0
      R = 0
      DO 130 I=1, N
         DO 120 J=1, N
            MP(I,J) = 0
120      CONTINUE
130   CONTINUE
      DO 220 J=1, N
         S = S+PR(J-1)
         IF (PR(J).NE.0) THEN
```

```
                 DO 210 K=1, PR(J)
                    R = R+1
                    DO 200 L=1, PR(J)
                       MP(NODES(L+S),K) = SNODES(R)
 200                CONTINUE
 210             CONTINUE
              ENDIF
 220       CONTINUE
           DO 320 I=1, N
              PI = 0
              DO 300 J=1, N
                 IF (MP(I,J).EQ.0) GOTO 310
                 PI = PI+1
 300          CONTINUE
 310          MINI(I) = 1
              MAXI(I) = PI
 320       CONTINUE
           IS = 1
 330       DO 410 RW=IS, N
              PRMI = MP(RW,MINI(RW))
              PRM(RW) = PRMI
              DO 400 J=1, RW-1
                 IF (PRM(J).EQ.PRMI) GOTO 700
 400          CONTINUE
 410       CONTINUE
           DO 510 K=1, NX
              DO 500 J=1, N
                 IF (X1(K).EQ.J) Y1(K) =PRM(J)
                 IF (X2(K).EQ.J) Y2(K) =PRM(J)
                 IF (X3(K).EQ.J) Y3(K) =PRM(J)
                 IF (X4(K).EQ.J) Y4(K) =PRM(J)
 500          CONTINUE
 510       CONTINUE
           CALL ARRNGX(NX,Y1,Y2,Y3,Y4)
           CALL SORTX(NX,Y1,Y2,Y3,Y4)
           DO 600 I=1, NX
              IF (SX1(I).NE.Y1(I) .OR. SX2(I).NE.Y2(I) .OR.
     *            SX3(I).NE.Y3(I) .OR. SX4(I).NE.Y4(I))
     *                                    GOTO 700
 600       CONTINUE
           ISO = 1
           GOTO 900
 700       IS = RW
           IF (MINI(RW).LT.MAXI(RW)) GOTO 710
           LST = RW
           GOTO 730
 710       MINI(RW) = MINI(RW)+1
           GOTO 330
 720       IF (MINI(LST).LT.MAXI(LST)) GOTO 740
 730       IF (LST.EQ.1) GOTO 900
           LST = LST-1
           IS = IS-1
           GOTO 720
 740       MINI(LST) = MINI(LST)+1
```

```
      DO 750 L=LST+1, N
         MINI(L) =1
750   CONTINUE
      GOTO 330
900   RETURN
      END
```

Note 1

The program generates all possible sets of arcs corresponding to the edges of K_n different from the edges of C, where $C = (1,2,3,\ldots,n-1,n,1)$. The first n arcs generated correspond to the edges $(1,3),(2,4),(3,5),\ldots,(n-1,1),(n,2)$.

Let $A = \{a, a_2, a_3, \ldots, a_n\}$ be a set of arcs corresponding to the first n edges and suppose that all the non-isomorphic drawings having A have been obtained. Whenever the program generates a new set of arcs $B = \{b, b_2, b_3, \ldots, b_n\}$ corresponding to the first n edges, it starts checking in the non-isomorphic drawings on hand whether

$$B_i = A \qquad i = 2,3,\ldots,n$$

where $B_i = \{b_i, b_{i+1}, \ldots b_n, b_1, b_2, \ldots, b_{i-1}\}$.

If there is a $B_i = A$ then the program drops this set B and generates another set, hence avoiding a large number of drawings which would have turned out to be isomorphic to previously generated drawings.

The program also checks whether

$$B_i^* = A \qquad i = 2,3,\ldots,n$$

where $B_i^* = \{b_i^*, b_{i+1}^*, \ldots, b_n^*, b_i^*, b_2^*, \ldots, b_{i-1}^*\}$

$$\text{and } b_j^* = \begin{cases} b_j + 1 & \text{if } b_j \text{ is odd} \\ b_j - 1 & \text{if } b_j \text{ is even} \end{cases}$$

To reduce the number of comparisons between A's and B's the program retains only the drawings in which

$$a_n = \max\{a_i\}, i = 1,2,\ldots,n.$$

Note 2

The program segment between the two dashed lines was added to the algorithm to benefit from possible symmetric properties in the drawings D_n when drawn using a C-F HC as a basis. Figures A.1.1 and A.1.2 provide examples illustrating how symmetry can be used to speed up the process of obtaining the non-isomorphic drawings.

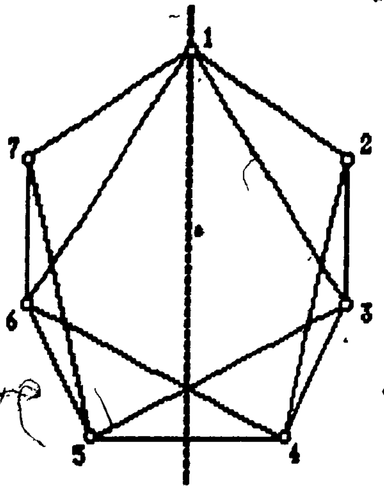


Fig.A.1.1: The dashed line splits the C-F HC into two parts to reflect a possible symmetry in drawings D_7.

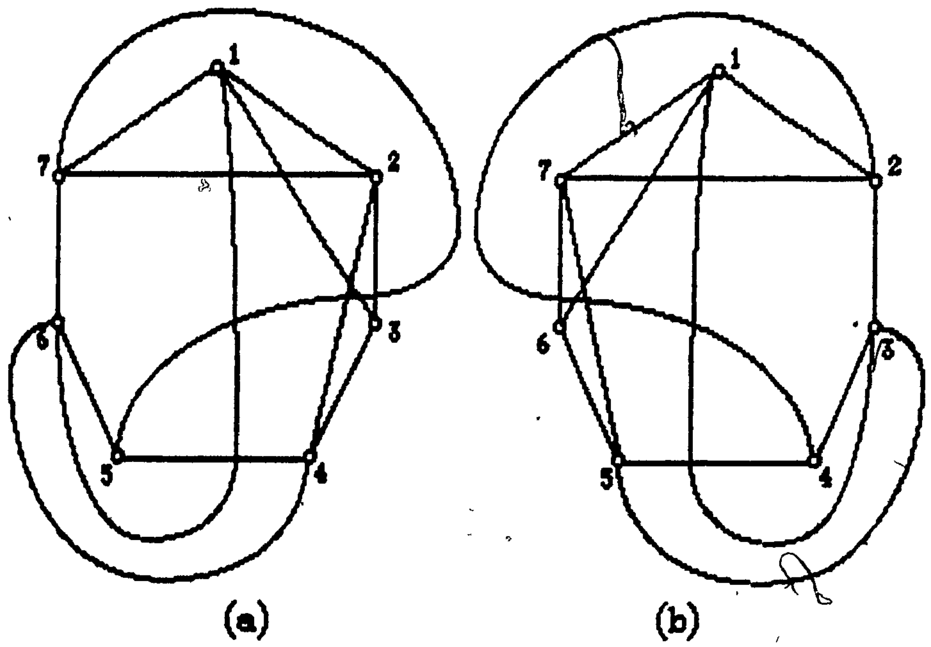


Fig.A.1.2: Taking symmetry into consideration, we can readily see that (b) is a mirror image of (a).

Any drawing having its first seven arcs as in Fig. A.1.2 (a) will be isomorphic to a drawing having its first seven arcs as in Fig. A.1.2 (b). Hence, if all the non-isomorphic drawings having (a) as a sub-drawing are obtained, then we can ignore all the drawings having (b) as a sub-drawing.

So, in a similar manner to the explanation in Note 1, whenever a new set of arcs B is generated, the program checks in the non-isomorphic drawings on hand whether

$$B_1 = A$$

and whether

$$B_1^* = A$$

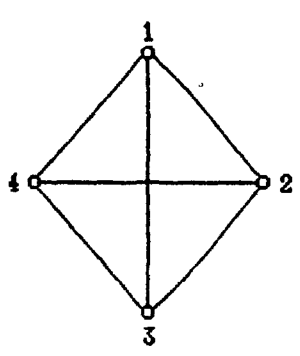
where A, B, B_1 and B_1^* are as defined in Note 1.

APPENDIX A.2

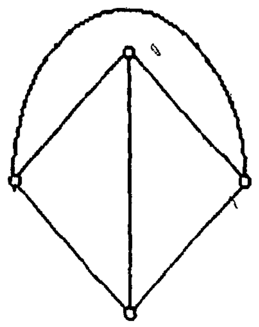
ALL NON-ISOMORPHIC GOOD DRAWINGS D_n OF K_n HAVING AT LEAST ONE C-F HC, FOR $n = 4,5,6$.

Appendix A.2

n = 4

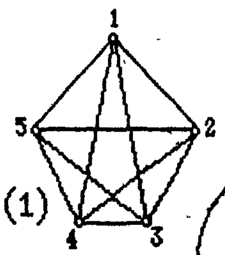


(1)

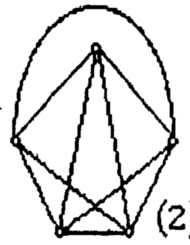


(2)

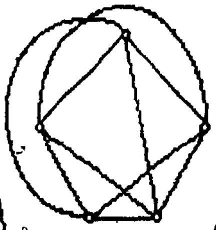
n = 5



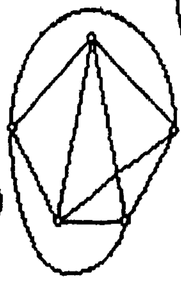
(1)



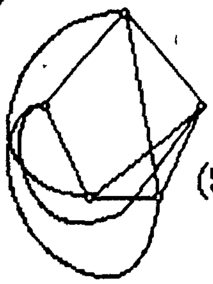
(2)



(3)

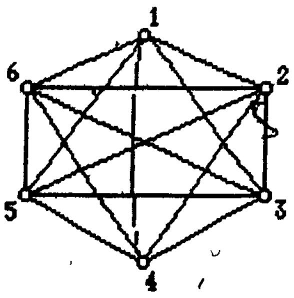


(4)

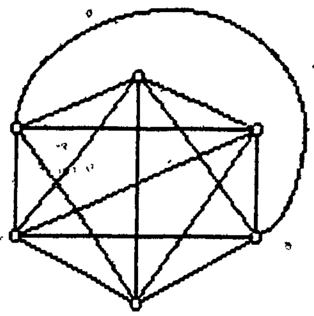


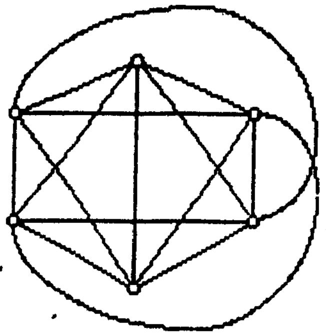
(5)

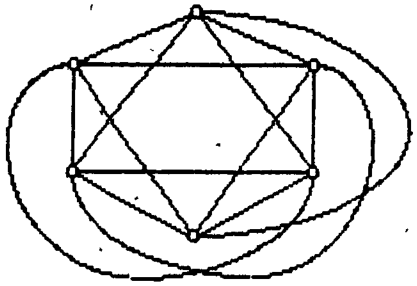
n = 6

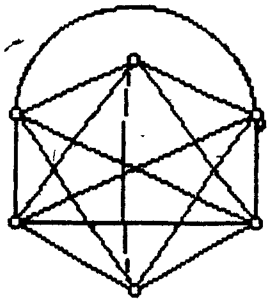


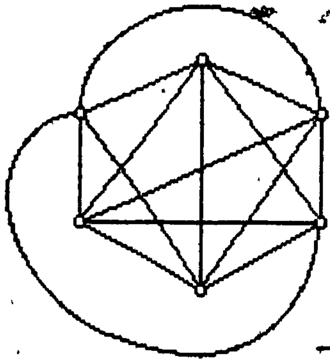
(1)

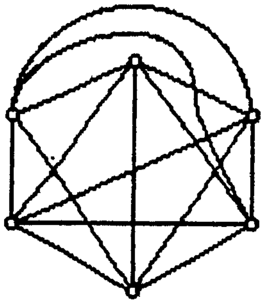

(2)

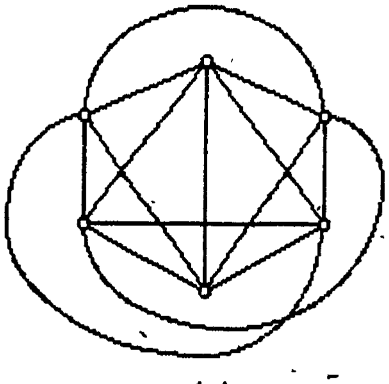

(3)

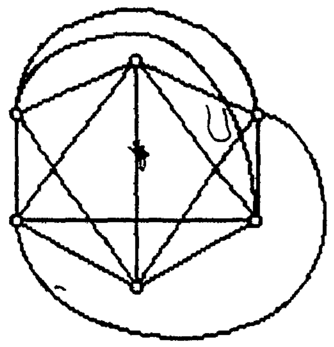

(4)

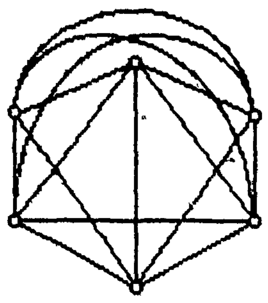

(5)

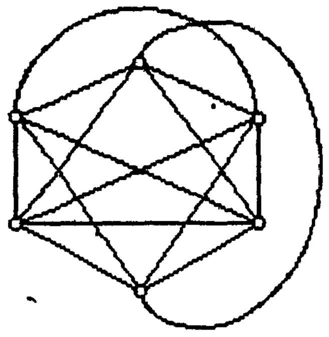

(6)

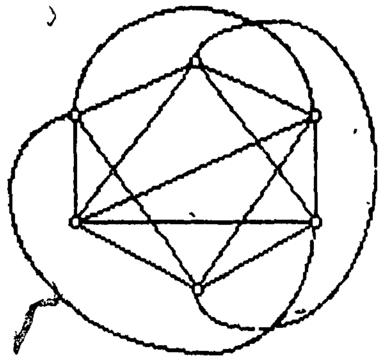

(7)

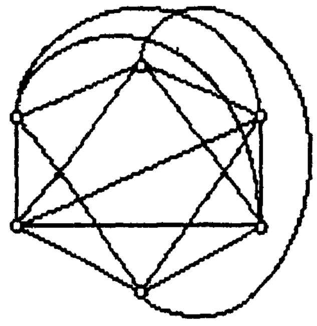

(8)

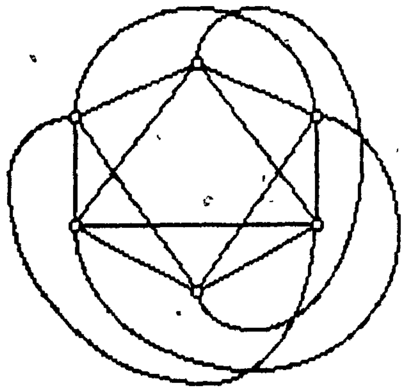

(9)

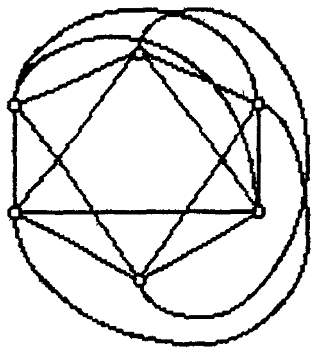

(10

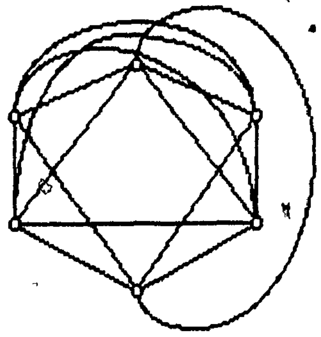

(11)

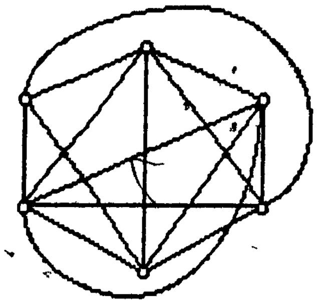

(12)

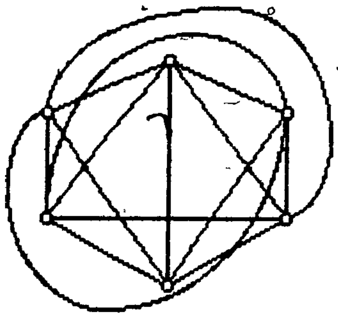

(13)

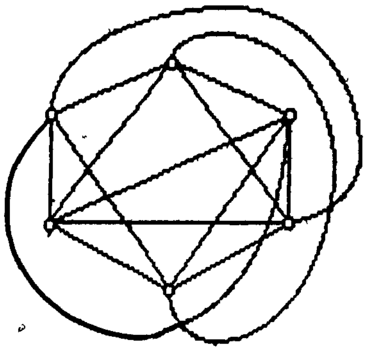

(14)

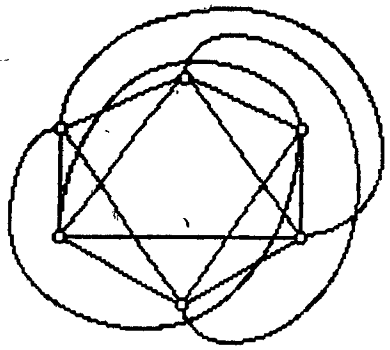

(15)

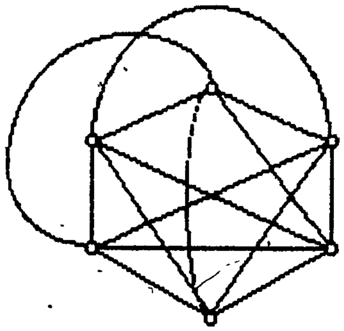

(16)

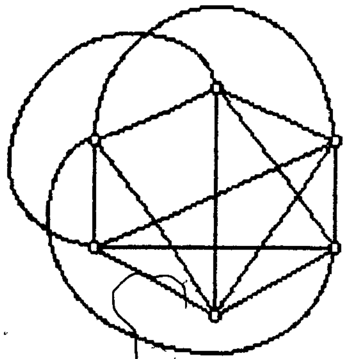

(17)

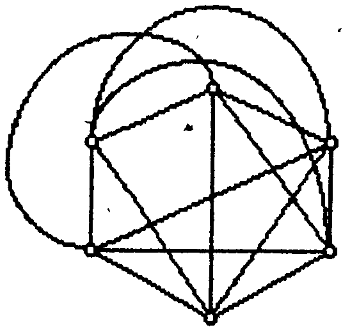

(18)

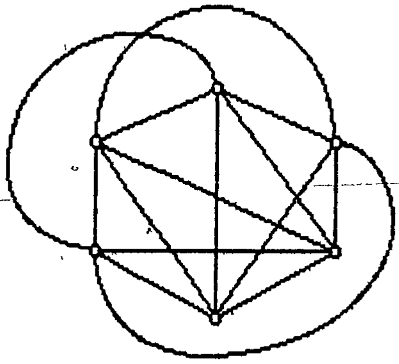

(19)

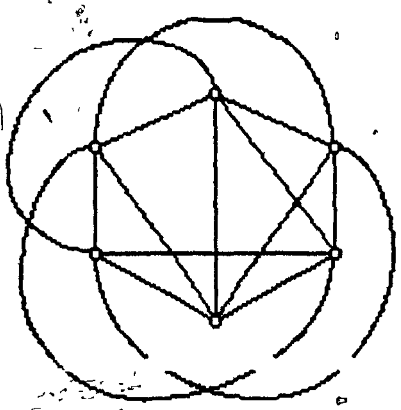

(20)

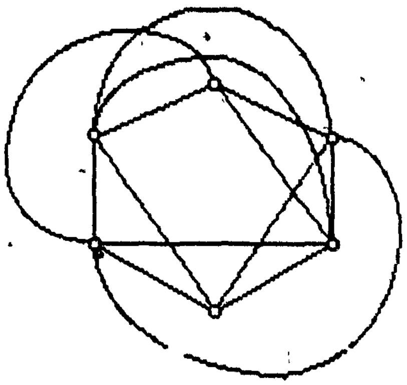

(21)

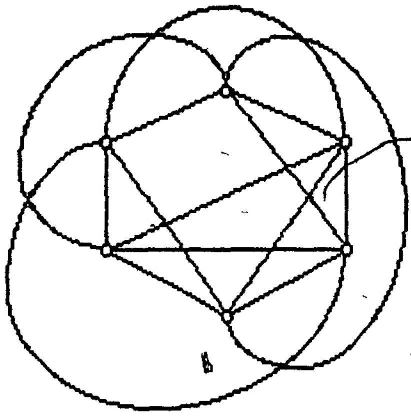

(22)

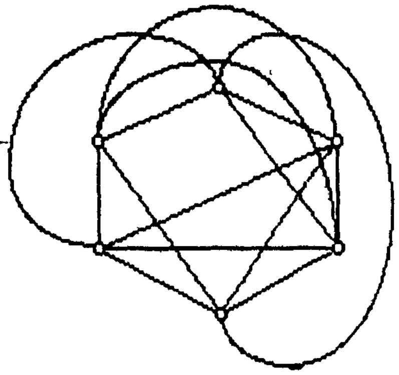

(23)

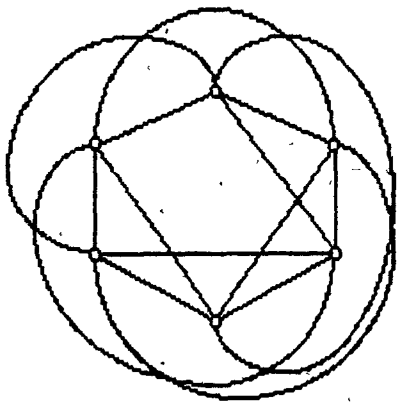

(24)

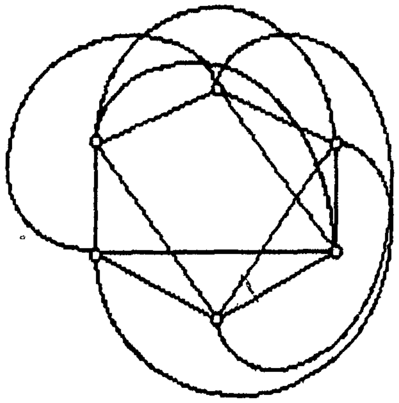

(25)

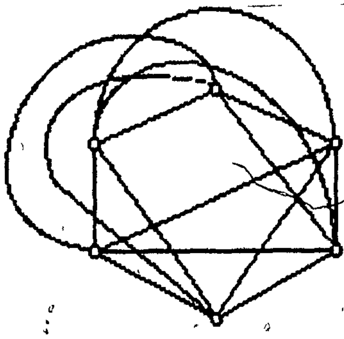

(26)

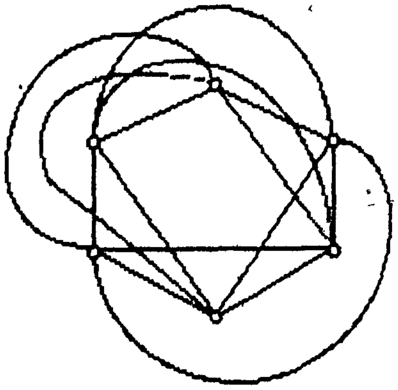

(27)

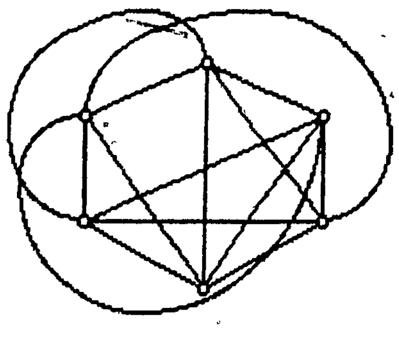

(28)

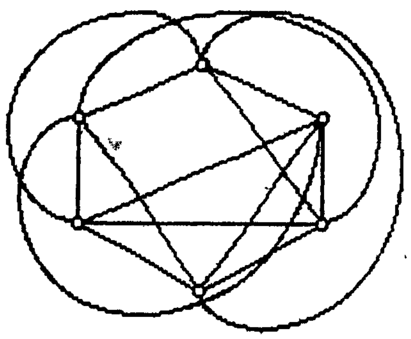

(29)

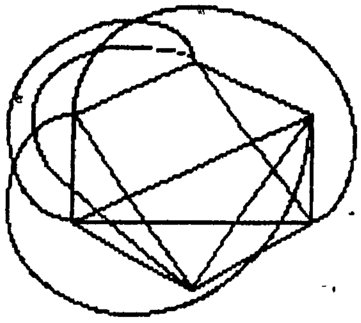

(30)

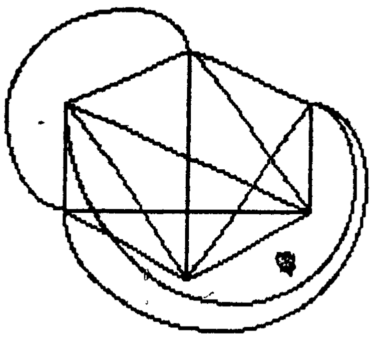

(31)

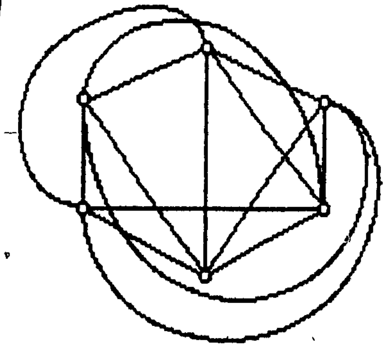

(32)

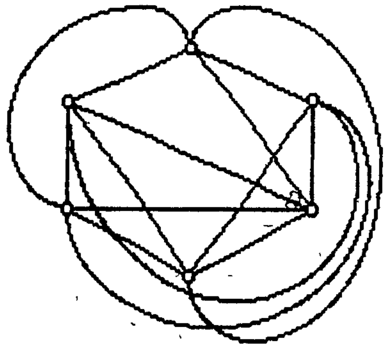

(33)

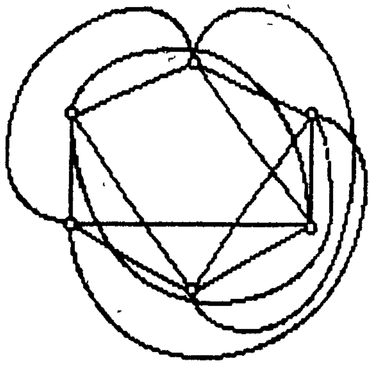

(34)

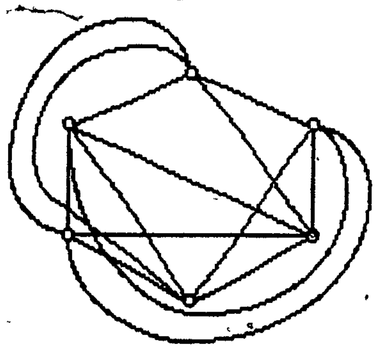

(35)

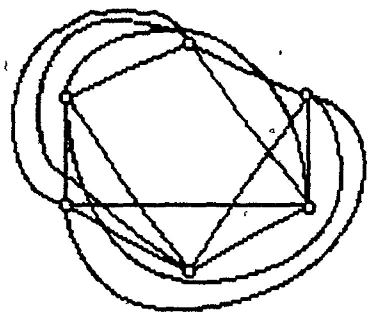

(36

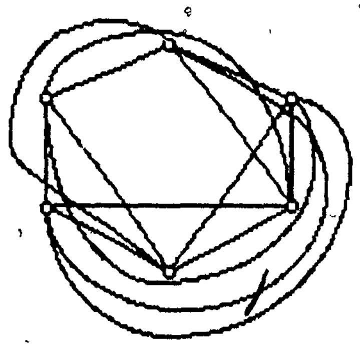

(37

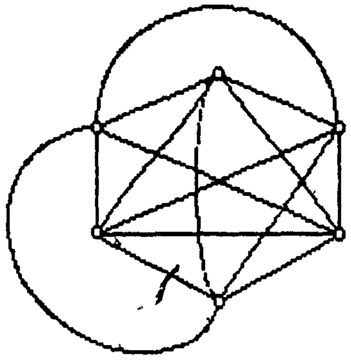

(38)

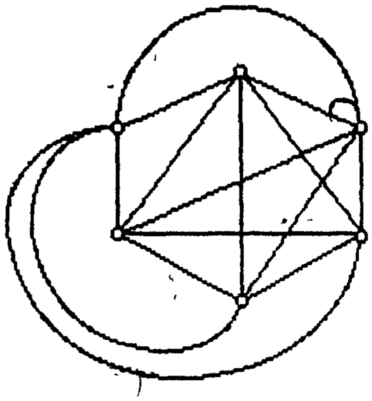

(39)

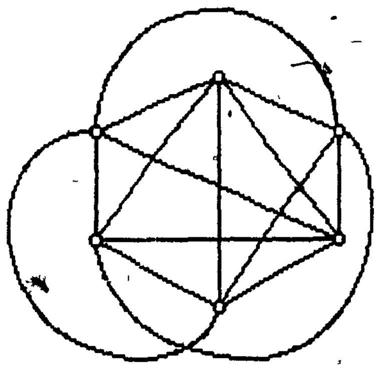

(40)

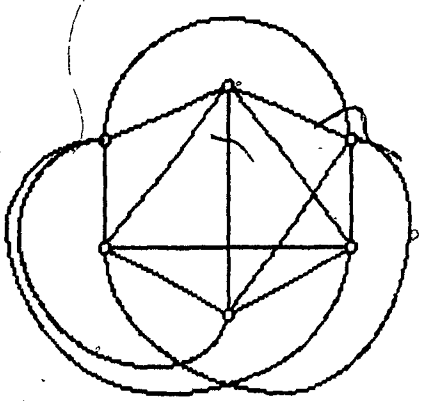

(41)

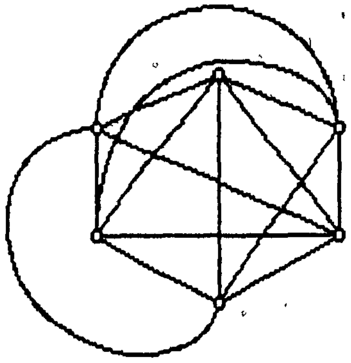

(42)

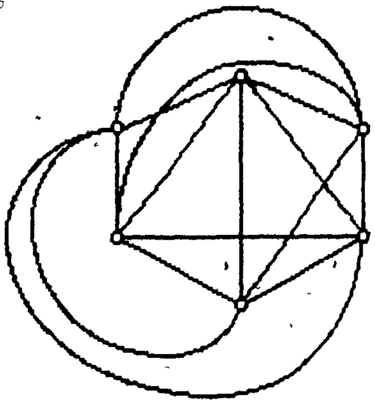

(43)

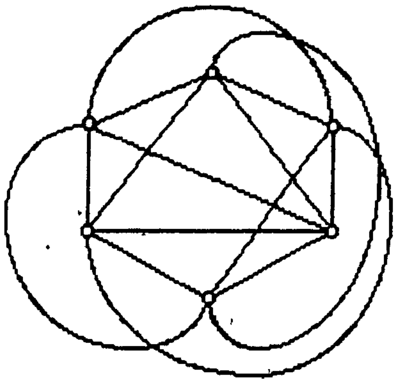

(44)

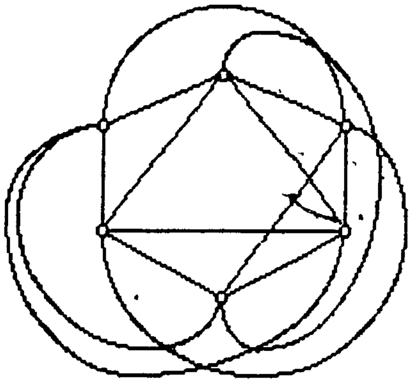

(45)

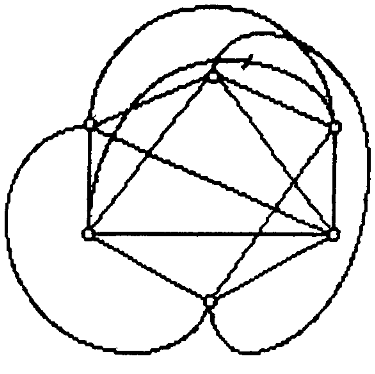

(46)

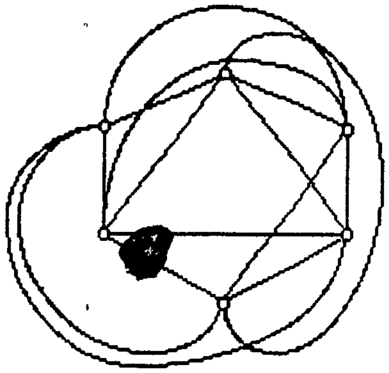

(47)

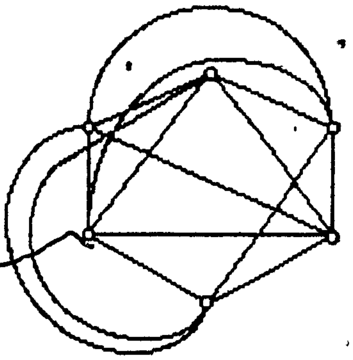

(48)

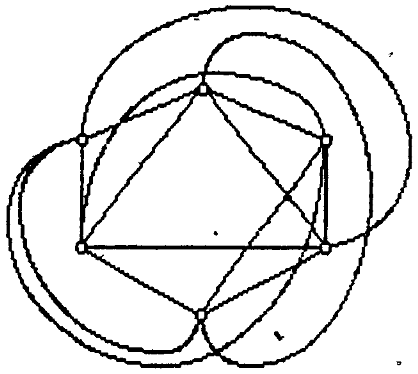

(49)

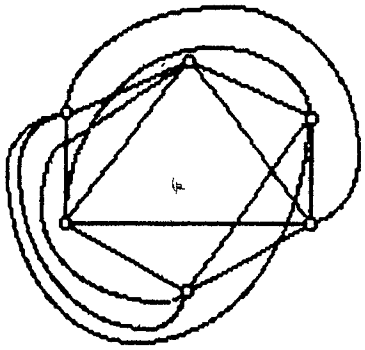

(50)

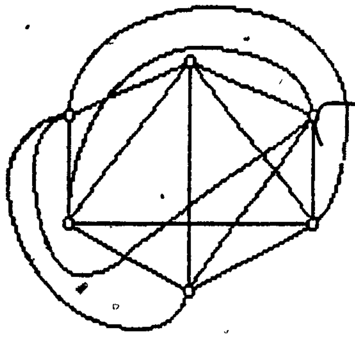

(51)

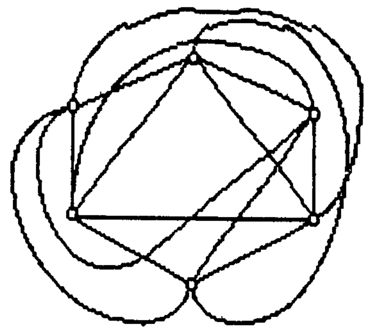

(52)

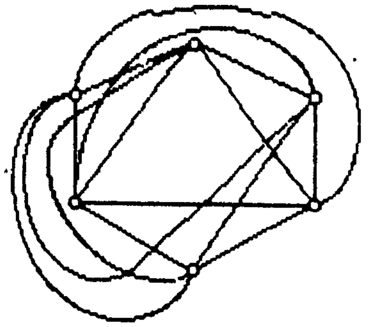

(53)

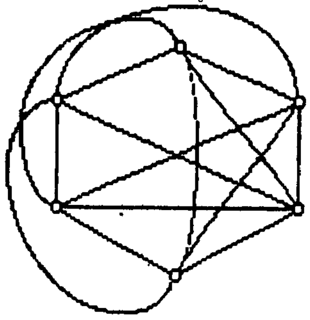

(54)

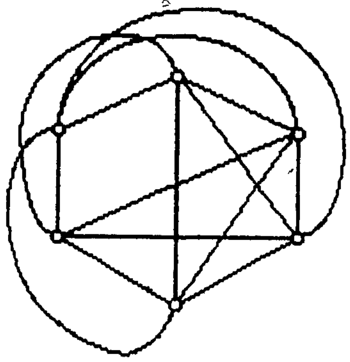

(55)

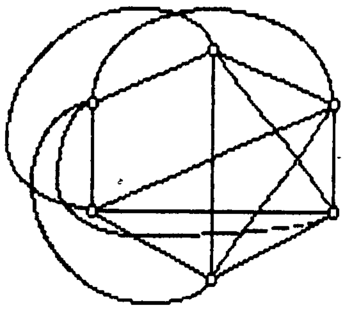

(56)

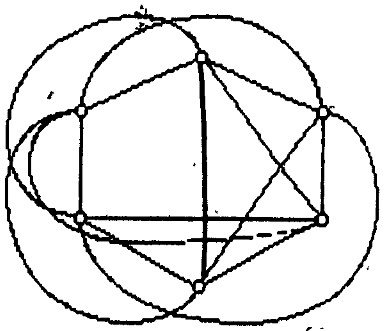

(57)

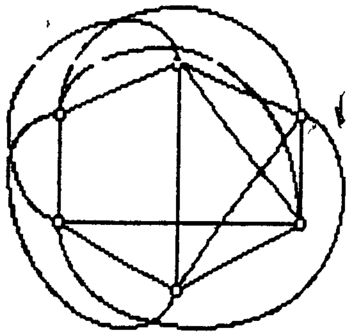

(58)

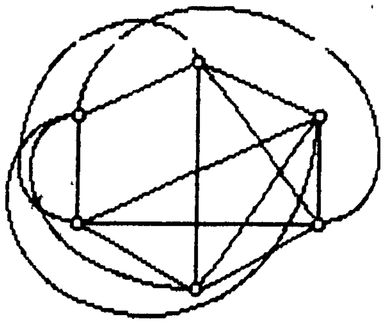

(59)

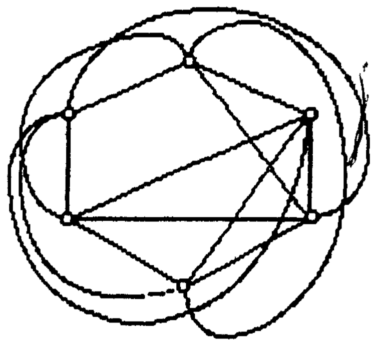

(60)

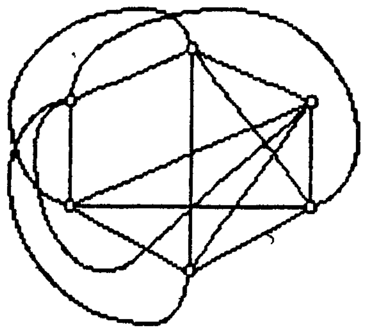

(61)

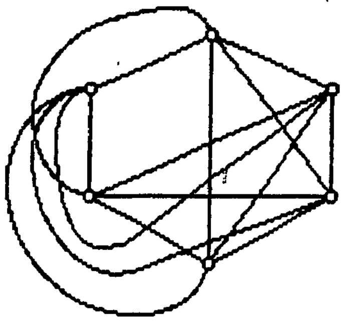

(62)

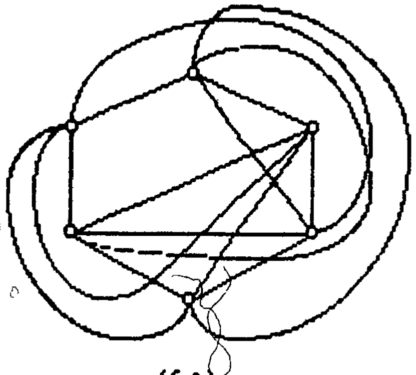

(63)

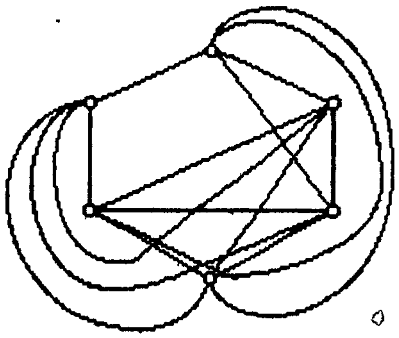

(64)

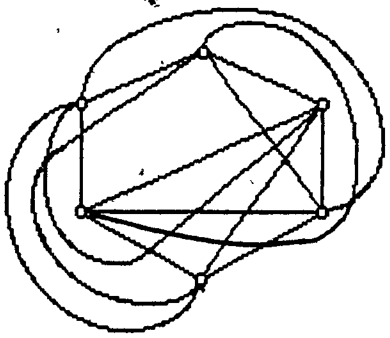

(65)

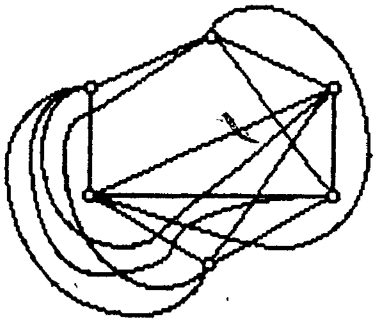

(66)

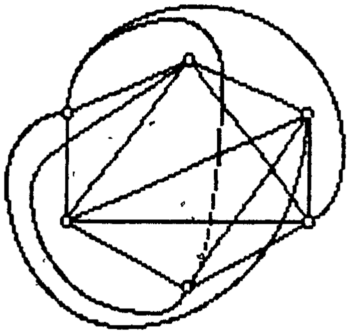

(67)

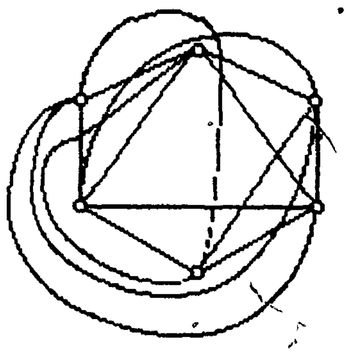

(68)

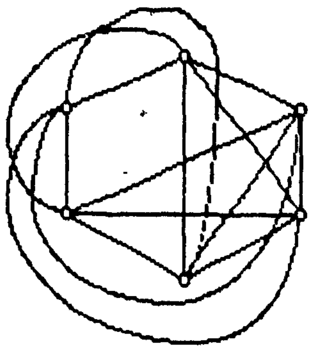

(69)

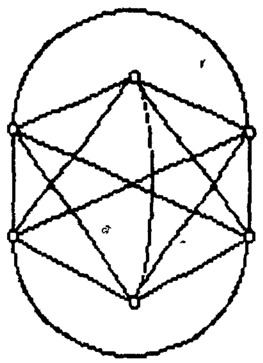

(70)

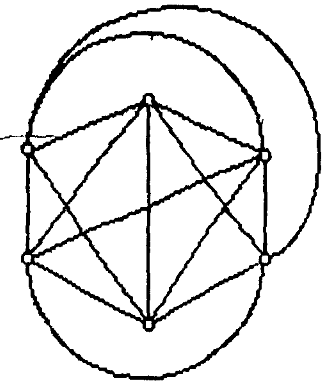

(71)

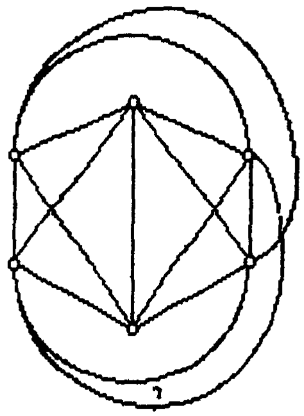

(72)

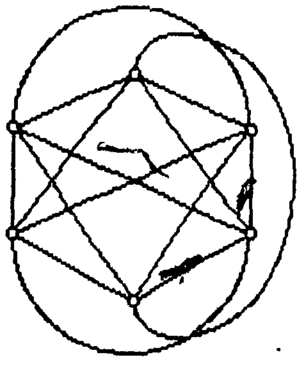

(73)

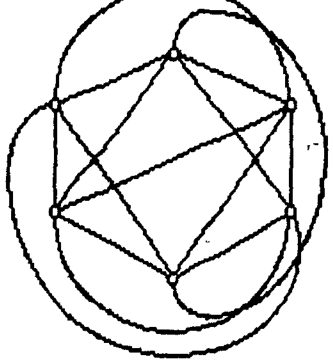

(74)

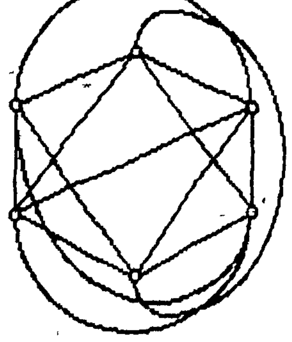

(75)

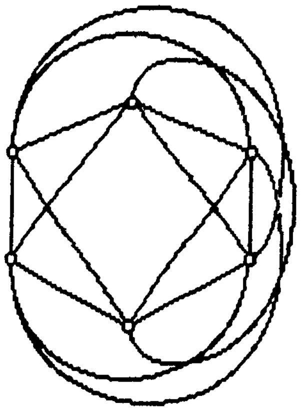

(76)

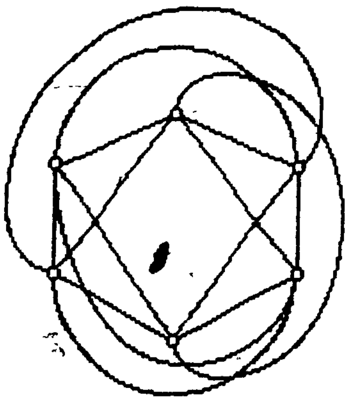

(77)

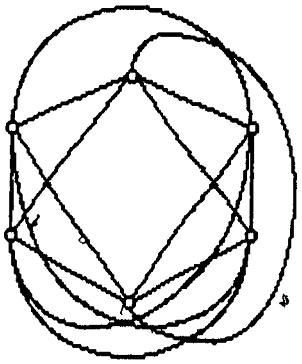

(78)

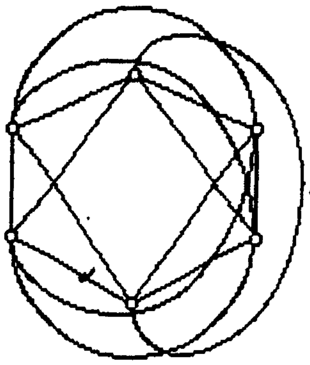

(79)

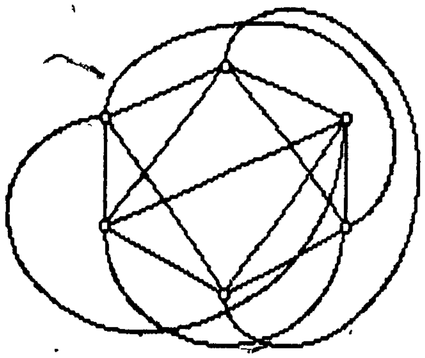

(80)

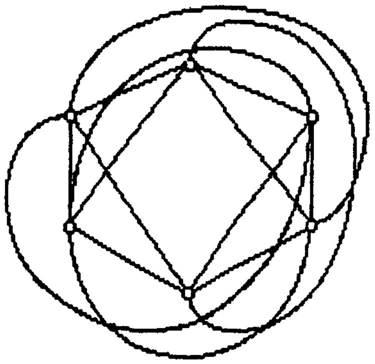

(81)

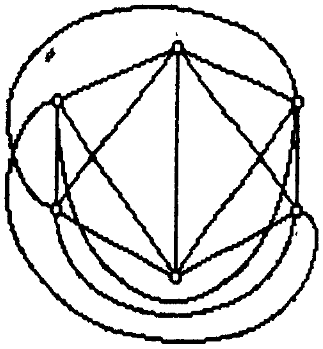

(82)

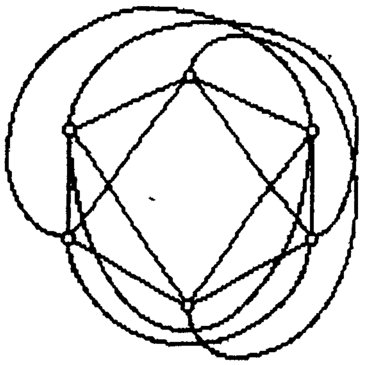

(83)

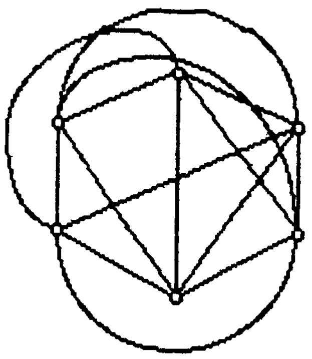

(84)

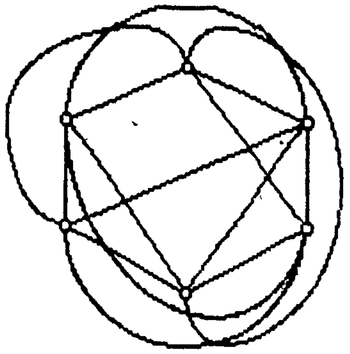

(85)

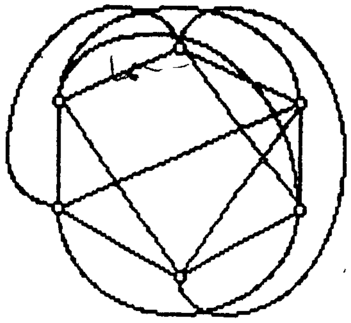

(86)

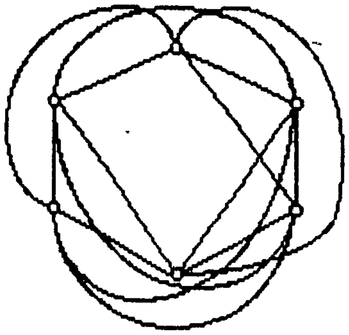

(87)

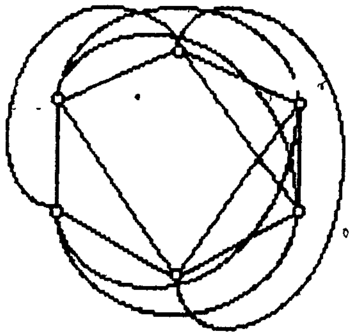

(88)

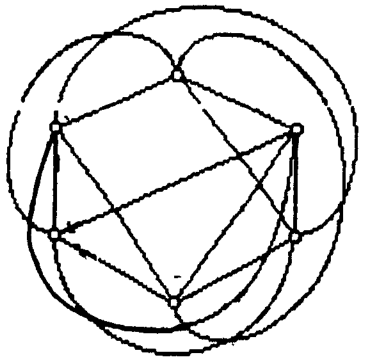

(89)

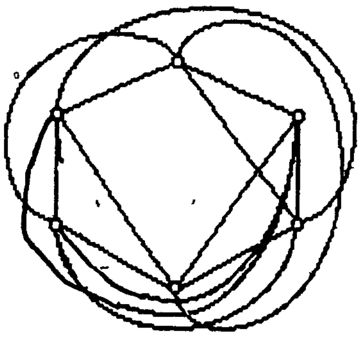

(90)

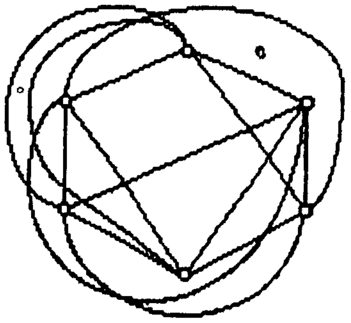

(91)

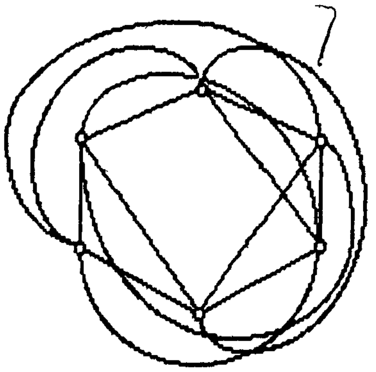

(92)

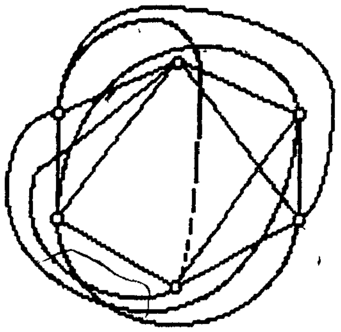

(93)

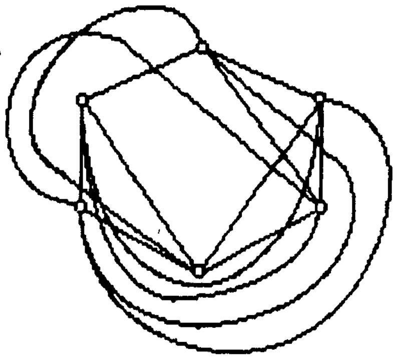

(94)

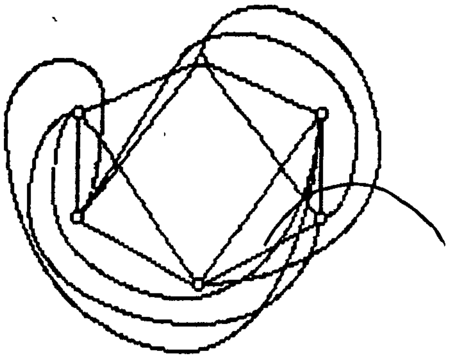

(95)

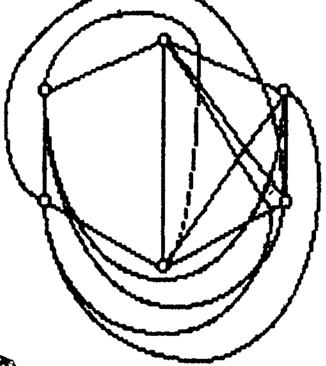

(96)

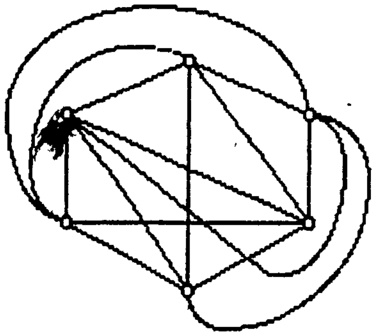

(97)


(98)


(99)


(100)


(101)


(102)

CROSSINGS OF THE 102 NON-ISOMORPHIC GOOD DRAWINGS D. OF K.

DRAWING	CROSSINGS														
NUMBER	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	13x24	13x25	13x26	14x25	14x26	14x35	14x36	15x26	15x36	15x46	24x35	24x36	25x36	25x46	35x46
2	13x24	13x25	13x26	14x25	14x26	14x35	15x26	15x46	24x35	25x46	35x46				
3	13x24	13x26	14x26	14x35	15x26	15x46	24x35	25x36	35x46						
4	13x24	13x26	14x25	14x36	15x26	15x46	24x35	25x36	35x46						
5	13x24	13x25	14x25	14x35	14x36	15x36	15x46	24x35	24x36	25x36	25x46	35x46			
6	13x24	13x25	14x25	14x35	15x46	24x35	25x46	35x46							
7	12x36	13x24	13x25	14x25	14x35	15x46	24x35	24x36	25x36	25x46	35x46				
8	13x24	14x35	15x46	24x35	25x36	35x46									
9	12x36	13x24	14x35	15x46	24x35	24x36	35x46								
10	12x36	13x24	14x35	15x46	16x25	24x35	24x36	25x36	25x46	35x46					
11	13x24	13x25	14x26	15x36	15x46	24x35	24x36	25x36	25x46	35x46					
12	13x24	13x25	14x26	14x36	15x46	24x35	25x46	35x46							
13	12x36	13x24	13x25	14x26	14x36	15x46	24x35	24x36	25x36	25x46	35x46				
14	13x24	14x25	14x26	14x36	15x46	24x35	25x36	35x46							
15	12x36	13x24	14x25	14x26	14x36	15x46	24x35	24x36	35x46						
16	12x36	13x24	14x25	14x26	14x36	15x46	16x25	24x35	24x36	25x36	25x46	35x46			
17	13x24	13x25	13x26	14x25	14x35	15x46	24x35	25x46	26x34	26x35	35x46				
18	13x24	13x26	14x35	15x46	16x25	24x35	25x46	26x34	26x35	35x46					
19	13x24	13x25	13x26	14x26	14x36	15x46	24x35	25x46	26x34	26x35	35x46				
20	13x24	13x26	14x25	14x26	14x36	15x46	16x25	24x35	25x46	26x34	26x35	35x46			
21	13x24	13x25	14x25	14x35	14x36	15x26	24x35	24x36	25x36	25x46	35x46				
22	13x24	13x25	14x25	14x35	15x26	15x36	24x35	25x46	35x46						
23	12x36	13x24	13x25	14x25	14x35	15x26	15x36	24x35	24x36	25x36	25x46	35x46			
24	13x24	14x35	14x36	15x26	24x35	24x36	35x46								
25	13x24	14x35	15x26	15x36	24x35	25x36	35x46								
26	12x36	13x24	14x35	15x26	15x36	24x35	24x36	35x46							
27	13x24	13x25	14x26	14x36	15x26	15x36	24x35	25x46	35x46						
28	12x36	13x24	13x25	14x26	14x36	15x26	15x36	24x35	24x36	25x36	25x46	35x46			
29	13x24	14x25	14x26	14x36	15x26	15x36	24x35	25x36	35x46						
30	12x36	13x24	14x25	14x26	14x36	15x26	15x36	24x35	24x36	35x46					
31	12x36	13x24	13x25	14x25	14x26	14x35	14x36	14x56	15x26	15x36	24x35	24x36	25x36	25x46	35x46
32	12x36	13x24	14x26	14x35	14x36	14x56	15x26	15x36	24x35	24x36	35x46				
33	13x24	13x25	13x26	14x25	14x35	15x26	15x36	24x35	25x46	26x34	26x35	35x46			
34	13x24	13x25	13x26	14x26	14x36	15x26	15x36	24x35	25x46	26x34	26x35	35x46			
35	13x24	13x25	13x26	14x25	14x26	14x35	14x36	14x56	15x26	15x36	24x35	25x46	26x34	26x35	35x46
36	13x24	14x35	14x36	24x35	24x36	26x35	26x45	35x46							
37	12x36	13x24	14x35	15x36	24x35	24x36	26x35	26x45	35x46						
38	13x24	14x25	14x26	24x35	24x36	26x35	26x45	35x46							
39	12x36	13x24	14x25	14x26	14x36	15x36	24x35	24x36	26x35	26x45	35x46				
40	13x24	14x26	14x35	14x56	24x35	24x36	26x35	26x45	35x46						
41	12x36	13x24	14x26	14x35	14x36	14x56	15x36	24x35	24x36	26x35	26x45	35x46			
42	12x36	13x24	14x26	14x35	14x36	14x56	15x23	15x24	15x26	15x36	24x35	24x36	26x35	26x45	35x46
43	13x24	13x25	14x25	14x35	14x36	15x36	24x35	24x36	25x36						
44	13x24	13x25	14x25	14x35	24x35										
45	13x24	14x35	14x36	15x36	24x35	24x36	25x46								
46	13x24	14x35	24x35	25x36	25x46										
47	13x24	14x35	14x36	15x36	16x25	24x35	24x36	25x36							
48	13x24	14x35	16x25	24x35											
49	13x24	14x25	14x26	15x36	24x35	24x36	25x46								
50	13x24	14x25	14x26	14x36	24x35	25x36	25x46								

DRAWING NUMBER	CROSSINGS 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
51	13x24	14x25	14x26	15x36	16x25	24x35	24x36	25x36							
52	13x24	14x25	14x26	14x36	16x25	24x35									
53	13x24	14x25	14x36	14x56	15x36	16x25	24x35	24x36	25x36						
54	13x24	13x26	14x25	14x26	14x36	16x25	24x35	26x34	26x35						
55	13x24	13x26	14x25	14x56	16x25	24x35	26x34	26x35							
56	13x24	13x26	14x26	14x35	16x25	24x35	26x35	26x45							
57	13x24	13x26	14x25	14x36	16x25	24x35	26x35	26x45							
58	13x24	13x26	14x25	14x26	14x56	16x25	24x35	26x35	26x45						
59	13x24	13x25	14x25	14x35	14x36	15x26	15x46	24x35	24x36	25x36					
60	13x24	13x25	14x25	14x35	15x26	15x36	15x46	24x35							
61	13x24	13x25	14x25	14x35	14x36	15x26	15x36	15x46	24x35	24x36	36x45				
62	13x24	14x35	14x36	15x26	15x36	15x46	24x35	24x36	25x36	25x46	36x45				
63	12x36	13x24	14x35	15x26	15x36	15x46	24x35	24x36	25x46						
64	13x24	13x25	13x26	14x25	14x35	15x26	15x36	15x46	24x35	26x34	26x35				
65	13x24	13x25	13x26	14x26	14x36	15x26	15x36	15x46	24x35	26x34	26x35				
66	13x24	13x25	13x26	14x25	14x26	14x35	15x26	15x36	15x46	24x35	26x35	26x45			
67	13x24	13x25	13x26	14x25	14x26	14x35	14x36	15x26	15x36	15x46	24x35	24x36	26x35	26x45	36x45
68	13x24	13x25	13x26	14x36	15x24	15x26	15x34	15x36	24x35	26x35	26x45				
69	13x24	13x25	13x26	15x24	15x26	15x34	15x36	24x35	24x36	26x35	26x45	36x45			
70	13x24	13x25	13x26	14x26	14x56	15x24	15x26	15x34	15x36	24x35	26x35	26x45			
71	13x24	13x25	13x26	14x26	14x36	14x56	15x24	15x26	15x34	15x36	24x35	24x36	26x35	26x45	36x45
72	12x46	13x24	13x25	13x26	13x46	14x56	24x35	25x46	26x34	26x35	35x46				
73	12x46	13x24	13x26	13x46	14x25	14x56	16x25	24x35	25x46	26x34	26x35	35x46			
74	12x46	13x24	13x25	13x26	13x46	14x25	14x35	15x26	15x36	15x46	24x35	25x46	26x34	26x35	35x46
75	13x24	13x25	14x25	14x36	15x36	15x46	24x36	25x36	25x46						
76	13x24	13x25	14x25	15x46	25x46										
77	13x24	15x46	24x36												
78	13x24	13x25	14x26	14x35	15x36	15x46	24x36	25x36	25x46						
79	13x24	13x25	14x26	14x35	14x36	15x46	25x46								
80	13x24	13x25	14x26	14x35	14x36	15x36	15x46	25x36	25x46	36x45					
81	13x24	14x25	14x26	14x35	14x36	15x46	25x36								
82	13x24	14x25	14x26	14x35	14x36	15x36	15x46	36x45							
83	13x24	13x25	14x25	14x26	14x35	14x36	15x36	15x46	25x34	25x36	36x45				
84	12x36	13x24	13x25	14x25	14x26	14x35	14x36	15x46	24x36	25x34	25x36				
85	13x24	13x25	13x26	14x26	14x35	14x36	15x46	25x46	26x34	26x35					
86	13x24	13x26	14x25	14x26	14x35	14x36	15x46	16x25	25x46	26x34	26x35				
87	13x24	13x26	15x26	15x46	25x36	26x34	26x45								
88	13x24	13x26	14x25	14x26	14x35	14x36	15x26	15x46	25x36	26x34	26x45				
89	12x36	13x24	13x25	14x25	15x26	15x36	24x36	25x36	25x46						
90	13x24	13x25	14x26	14x35	14x36	15x26	25x36	25x46	36x45						
91	12x36	13x24	13x25	14x26	14x35	14x36	15x26	15x36	24x36	25x36	25x46				
92	13x24	13x25	14x25	14x26	14x35	14x36	15x26	25x34	25x36	36x45					
93	12x36	13x24	13x25	14x25	14x26	14x35	14x36	15x26	15x36	24x36	25x34	25x36			
94	13x24	13x25	13x26	14x26	14x35	14x36	15x26	15x36	25x46	26x34	26x35				
95	13x24	13x25	13x26	14x25	14x26	14x35	14x36	15x26	15x36	25x34	26x34	26x35			
96	13x24	13x25	13x26	14x25	14x26	14x36	14x56	15x26	15x36	25x46	26x34	26x35			
97	12x36	13x24	14x25	14x26	14x35	14x36	15x36	24x36	26x35	26x45					
98	12x46	13x24	13x26	13x46	14x25	14x35	14x56	16x25	25x46	26x34	26x35				
99	13x24	13x26	14x26	14x35	14x36	14x56	15x23	15x24	15x26	16x35	24x35	26x34	26x35	26x45	36x45
100	13x24	13x25	13x26	14x25	14x26	14x35	14x36	15x46	16x25	16x35	25x34	25x36	25x46	26x34	35x46
101	12x35	12x46	13x24	13x26	13x46	15x24	15x26	15x34	15x36	15x46	24x35	26x34	26x35	26x45	36x45
102	14x26	14x35	14x36	26x34	26x35	35x46									

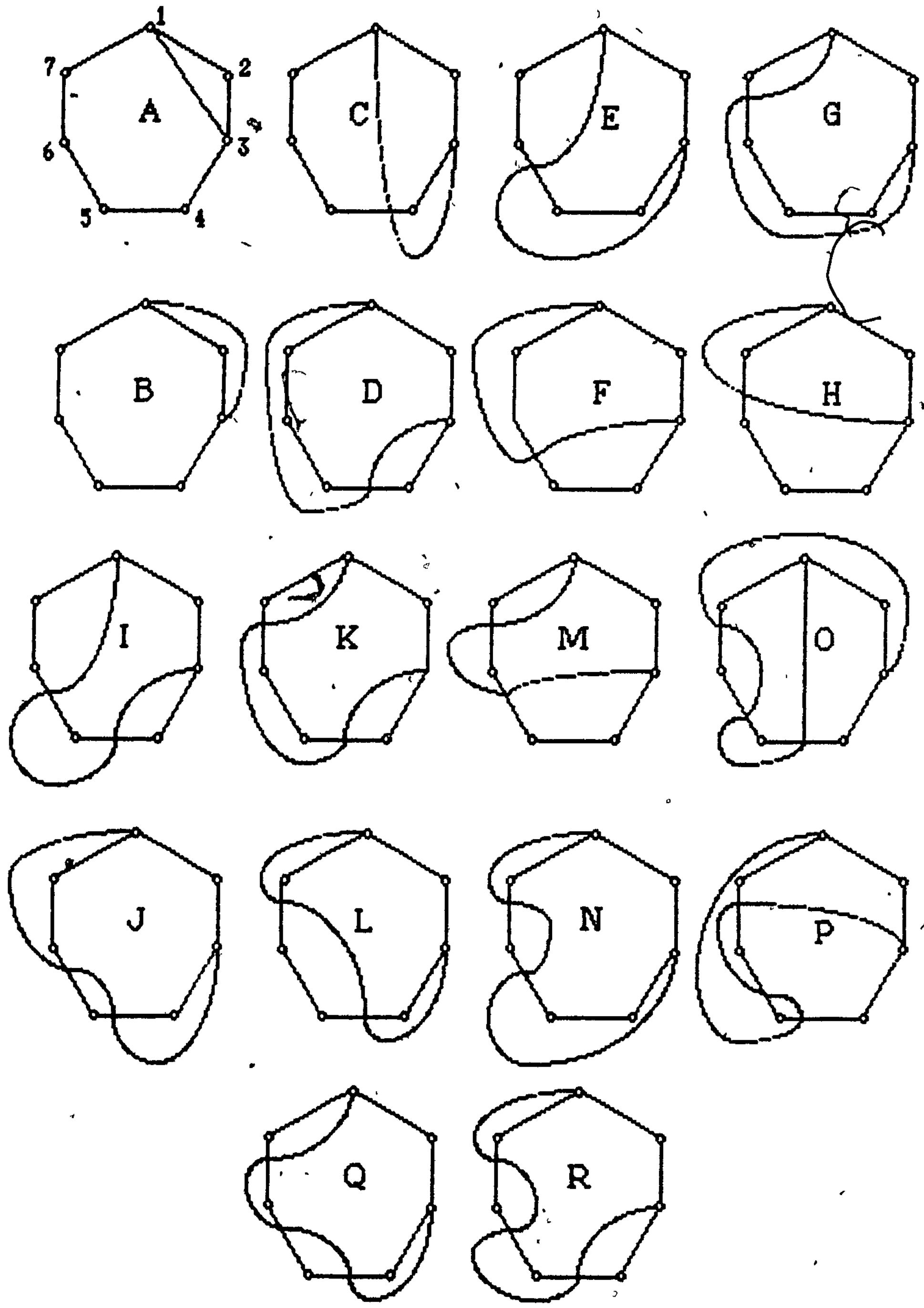
APPENDIX A.3

LIST OF THE NON-ISOMORPHIC GOOD DRAWINGS

D_7 OF K_7

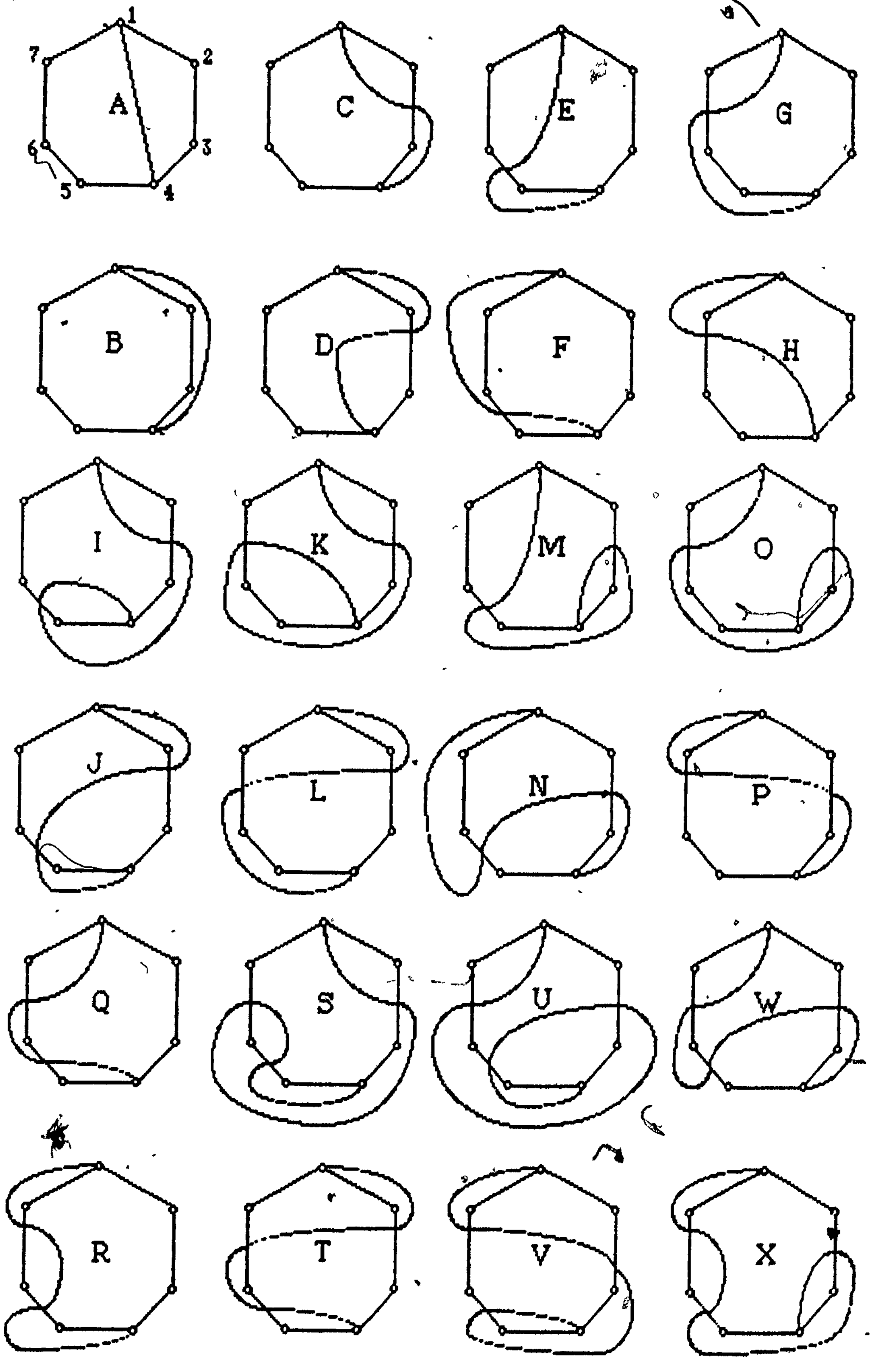
HAVING AT LEAST ONE C-F HC

LETTERS SYMBOLIZING ARCS



See Fig.2.1.d

LETTERS SYMBOLIZING ARCS



See Fig.2.1.e

DRAWINGS WITH 9 CROSSINGS:

```
AAABAABAABBABB AAABAABAHBBABB AAABAABAHBDABB AABAABBABBAABA AABAABBABBAABE
```

DRAWINGS WITH 11 CROSSINGS:

```
AAAABABAABBACB AAAABABAAFBACB AAAABABABABDCB AAAABABABBBACB AAAABABABFBACB AAAABABABGBACB AAAABABAHFBACB AAAABABAFABDCB
AAAABABAFBACB  AAABAABAABAABB AAABAABAABBAHB AAABAABABBAABB AAABAABABBBABB AAABAABABBBACB AAABAABABBBDABB AAABAABABBEABB
AAABAABAFBAABB AAABAABAFBBABB AAABAABAFBDABB AAABAABAFBEABB AAABAABAHBAABB AAABAABAHBDABG AAABAABAHBEABB AABAABBAABAABG
AABAABBABBABBB AABAABBABBABBD AABAABBABBACBB
```

DRAWINGS WITH 13 CROSSINGS:

```
AAAAAABAABBABB AAAAAABAABBACB AAAAAABABBBABB AAAAAABABBBABD AAAAAABABBEABB AAAAABBABBAABA AAAAABBABBAABB AAAAABBABBAABD
[illegible]    AAAABAAAABACB  AAAABABAAABDCB AAAABABAABBAAB AAAABABAABBABB AAAABABAABBDAB AAAABABAABBDCB AAAABABAAFBDCB
[illegible]    AAAABABABBBCB  [illegible]    AAAABABABBAABB AAAABABABBAACB AAAABABABBBAAB AAAABABABBBABB AAAABABABBBDAB
[illegible]    AAAABABAFBAAB  AAAABABABFBAAB AAAABABAFBDCB  AAAABABABDBABB AAAABABABDBDCB AAAABABAHBBACB AAAABABABABDGB
[illegible]    AAAABABABBDCB  AAAABABABFBDAB AAAABABABFBDCB AAAABABABBDCB  AAAABADAAFBACB AAAABADAHFBACB AAAABADABFBACB
[illegible]    AAABAABAAABACB AAABAABAABAACB AAABAABAABAAFB AAABAABAABBABD AAABAABAABEAFB AAABAABAABGAFB AAABAABABABABB
[illegible]    AAABAABABBAABB AAABAABABBBABD AAABAABABBDABB AAABAABABBEABB AAABAABABBGABB AAABAABABDBABB AAABAABAFBAABG
[illegible]    AAABAABAFBAFB  AAABAABAFBAHB  AAABAABAFBDABA AAABAABAFBDABB AAABAABAFBDAFB AAABAABAFBEAFB AAABAABAFBGABB
[illegible]    AAABAABAHBAABB AAABAABAHBBABA AAABAABAHBBABD AAABAABAHBDABA AAABAABAHBDABD AAABAABAHBDAHB AAABAABAHBEABB
[illegible]    AAABAABAHBEABB AAABAABAHBDABB AAABAABBHBDABB AAABABBAHBBAFB AAABABBAHBBAHB AAABABBAHBDAHB AAABABBBBBDABA
AAABABBBFBDABA AAABABBBHBDABA AAABABBBHBDABB AAABABBEABEABB AAABABBEABGABB AAABABBEFBDABA AAABABBEFBDABB AAABABBEHBDABA
AAABABBAFBDABA AABAABBAABAABA AABAABBAABAABE AABAABBAABABBE AABAABBAABABBB AABAABBAABACBE AABAABBAABACBB AABAABBABBABAB
AABAABBABBBABA AABAABBABBGABE AABAABBABBGABB AABAABBABEAABA AABAABBABEAABE AABAABBABEABBG AABAABBABGABBG
```

DRAWINGS WITH 15 CROSSINGS:

```
AAAAAABAABAABB AAAAAABAABBAAB AAAAAABAABBABD AAAAAABAABBAFB AAAAAABAABEABB AAAAAABAABGABB AAAAAABAABGACB AAAAAABAAFBACB
[illegible]
```

DRAWINGS WITH 17 CROSSINGS:

[illegible]

AAABAABEFBBAFB	AAABAABEFBBAHB	AAABAABEFDBABB	AAABAABGFABABB	AAABAABGFBBAFB	AAABAABGFBEAFB	AAABAABGFBEAHB	AAABAADAHABACB
[illegible]	[illegible]	AAABAADGHBBACB	AAABAADGHBBAHB	AAABAAEABBAABD	AAABAAEABBEABD	AAABAAEAFBAABD	AAABAAFAABBAAB
[illegible]	AAABAAFAAFBAAB	AAABAAFAAFBAHB	AAABAAFAHBBAHB	AAABAAFBHBBACB	AAABAAFBHBBAHB	AAABAAFEABBAAB	AAABAAFEABBAHB
AAABAAFEAFBAAB	AAABAAFEAFBAHB	AAABAAFGABBAAB	AAABAAFGABBACB	AAABAAFGAFBAAB	AAABAAKAABAABB	AAABAAKAABEABB	AAABAAKAHBAABB
AAABAAKAHBEABB	AAABAAKEABEABB	AAABAAMAABGABB	AAABAAMAHBBAFB	AAABAAMAHBBAHB	AAABAAMBABBABQ	AAABAAMBABBAFB	AAABAAMBAFBAHB
[illegible]	[illegible]	[illegible]	AAABAAMEABBAHB	AAABAAMEABGABB	AAABAAMEHBBABB	AAABAAMGABBABB	AAABABBAAABAAB
[illegible]	AAABABBAABBCBB	AAABABBABABAAB	AAABABBABABBBA	AAABABBABABCBA	AAABABBABAGFBA	AAABABBABADBBA	AAABABBABADCBA
[illegible]	AAABABBABBBAAA	AAABABBABBBAAD	[illegible]	AAABABBABBBCBB	AAABABBABBBCBD	AAABABBABBBFBA	AAABABBABBDAAA
[illegible]	AAABABBABBDBBB	[illegible]	AAABABBABBDFBB	AAABABBABBDFBD	AAABABBABBEAAB	[illegible]	AAABABBABEDBBA
[illegible]	[illegible]	AAABABBABGDBBA	AAABABBABGDFBA	AAABABBAFBBAFA	AAABABBAFBBAHA	AAABABBAFBBAHD	AAABABBAFBDAHA
AAABABBAFBDAHD	AAABABBAFBDFBB	AAABABBAFBDFBD	AAABABBAFBEAHB	AAABABBAFBGAFB	AAABABBAFBGAHB	AAABABBAHBBAFA	AAABABBAHBBAFQ
[illegible]	AAABABBAHBBAHD	AAABABBAHBBAHD	AAABABBAHBBFBA	AAABABBAHBBFBB	AAABABBAHBDAHA	AAABABBAHBDAHD	AAABABBAHBDAHQ
[illegible]	AAABABBAHBDFBD	AAABABBAHBEAHB	AAABABBAHBGAHG	AAABABBBABDBAA	AAABABBBABDCAA	AAABABBBABDCHA	AAABABBBABDGAA
[illegible]	AAABABBBBABAAB	AAABABBBBABABB	AAABABBBBABABD	AAABABBBBABBAA	AAABABBBBABBBA	AAABABBBBABCBA	AAABABBBBADABB
[illegible]	[illegible]	AAABABBBBADCBA	AAABABBBBBBAAA	AAABABBBBBBAAB	AAABABBBBBBABB	AAABABBBBBBABD	AAABABBBBBBABQ
[illegible]	AAABABBBBBBAAA	[illegible]	[illegible]	AAABABBBBEBABA	AAABABBBFABABA	AAABABBBFADABA	AAABABBBFBBABB
[illegible]	AAABABBBFBBABD	AAABABBBFBBAHA	AAABABBBFBDAHB	AAABABBBFBDAHA	AAABABBBFBDAHB	AAABABBBFEDABA	AAABABBBHABABA
[illegible]	AAABABBBHAHABA	AAABABBBHAHABB	AAABABBBHBBABD	AAABABBBHBBAFB	AAABABBBHBBAHA	AAABABBBHBBAHQ	AAABABBBHBDAHA
[illegible]	AAABABBBHBDAHD	AAABABBBHEDABA	AAABABBEABGAAG	AAABABBEABGAHG	AAABABBEBABABA	AAABABBEBABABB	AAABABBEBABABD
[illegible]	AAABABBEBBBABD	AAABABBEBBDAAB	AAABABBEBBGABG	AAABABBEFBBABD	AAABABBEFBBAFB	AAABABBEFBBAHB	AAABABBEFBDAHB
[illegible]	[illegible]	AAABABGEABEABD	AAABABGEBBEABD	AAABABGEBBEABG	AAABABGEHBEABG	AAABABKAHBEABB	AAABABKEABEABG
[illegible]	AAABABMAHBBAHB	AAABABMAHBDAHB	AAABABMBHBBAHB	AAABABMBHBDAHB	AAABADEHFBACBA	AAABADERFBACBA	AAABADMAHBBABB
[illegible]	AAABADMRHBDABB	AAABADMRHBDABG	AAABADMRHBDABD	AAABAFKAHBAABB	AAABAFMRHBDABB	AAABAHKAHBAABB	AAABAHKAHBEABB
AAABAHMAHBBAFB	AAABAHMAHBBAHB	[illegible]	AAABAHMBHBBABB	AAABAHMBHBBAHB	AAABAHMBHBDABB	AAABCAFBABBAAB	AAABCAFBABBACB
AAABCAFBAFBAAB	AAABCAFEABBAAB	AAABCAFEABBAHB	AAABCAFEAFBAAB	AAABCAFEAFBAHB	AAABCAFGABBAAB	AAABCAFGABBACB	AAABCAFGAFBAAB
AAABCAKEABEABB	AAABCAKEABJABB	AAABCAMBABBABB	AAABCAMEABBAFB	AAABCAMEABBAHB	AAABCAMEABGABB	AAABCAMGABBABB	AAABEAHBABBACB
AAABEAHBACBACB	AAABEAHEACBACB	AAABGAHBABBACB	AAABGAHBACBACB	AAABGAHEACBACB	AAAECBGEBBDABG	AAAECBKEABDABG	AABAABBAAAABBB
AABAABBAABABAE	AABAABBAABBBBA	AABAABBAABGBBE	AABAABBAABBBBG	AABAABBAAEAABG	AABAABBAAEABBE	AABAABBAAEABBG	AABAABBABABCBA
AABAABBABAHFBB	AABAABBABAEHBA	AABAABBABBACAD	AABAABBABBACAG	AABAABBABBACBD	AABAABBABBBFBB	AABAABBABBGBAB	AABAABBABBGBAE
AABAABBABBGBBB	AABAABBABBGBBB	AABAABBABBGFBB	AABAABBABEBABA	AABAABBACBBBBA	AABAABBACBGBBE	AABAABBAHBBFBA	AABAABBAHBBFBB
[illegible]	AABAABBBBBBABA	AABAABEABAAABA	AABAABEABAAABD	AABAABEABAACBD	AABAABEABAAFBD	AABAABEABAAHBA	AABAABEABAAHBD
AABAABEABEAABD	AABAABEABGAFBD	AABAABEBBBAABA	AABAABEBBEAABA	AABAABEFBBAHBA	AABAABEFBEAHBA	AABAABEFBGAHBA	AABAABEHBDAHBA
AABAABEHBEAHBA	AABAABGBABAABA	AABAABGBABAABG	AABAABGBBBAABD	AABAABGBBBAABG	AABAABGBBDAABA	AABAABGBBEAABG	AABAABGBBEABBG
AABAABEFBEABBB	AABAABKAHBEABB	AABABABAAAAAAB	AABABABAAFABFB	AABABABAAFBBFB	AABABABAADABCB	AABABABBAAAAAB	AABABABBAAAABB
AABABABBAAAACB	AABABABBAABABA	AABABABBABAABA	AABABABBABAAFB	AABABABBABABBA	AABABABBABABBB	AABABABBABABCB	AABABABBABBAFB
AABABABBADAAAB	AABABABBADAABB	AABABABBADAACB	AABABABBADBABB	AABABABBAFAAAB	AABABABBAFABAB	AABABABBAFBAFB	AABABABBAQBABB
AABABABBBBAAAR	AABABABBBBAABR	AABABABBBBAAGB	AABABABBBBABAH	AABABABBBBABGB	AABABABBBBAABE	AABABABBBBAAQB	AABABABBBBABAD
AABABABBBBABAE	AABABABBBBABBB	AABABABBBBABBD	AABABABBBBABBE	AABABABBBBABCB	AABABABBBBABQB	AABABABBBFABAR	AABABABFABABAB

DRAWINGS WITH 19 CROSSINGS:

AAAAAAAAADAABB	AAAAAAAAADABBB	AAAAAAAAAGBAFB	AAAAAAAAABBBBB	AAAAAAAABABABB	AAAAAAAABABABB	AAAAAAAABBABBG	AAAAAAAABBBBBB
AAAAAABAAABAAB	AAAAAABAAABABD	AAAAAABAAABBAB	AAAAAABAAABBBB	AAAAAABAAABBCB	AAAAAABAABAAAB	AAAAAABAABABAB	AAAAAABAABABBA
AAAAAABAABABBD	AAAAAABAABABBB	AAAAAABAABABCB	AAAAAABAABABFB	AAAAAABAABBAAD	AAAAAABAABBACA	AAAAAABAABBAFD	AAAAAABAABBBBA
AAAAAABAABBBBD	AAAAAABAABBBFB	AAAAAABAABEAAB	AAAAAABAABEBBB	AAAAAABAABGACB	AAAAAABAABGBAB	AAAAAABAABGBBB	AAAAAABAABGBCB
AAAAAABAAFBACF	AAAAAABAAFBBAB	AAAAAABAAFBBCB	AAAAAABAAFGACB	AAAAAABAAGBABD	AAAAAABAAQBABD	AAAAAABABAABAB	AAAAAABABAABCB
AAAAAABABAABGA	AAAAAABABAABGB	AAAAAABABAABBD	AAAAAABABABABF	AAAAAABABABBAA	AAAAAABABABBAD	AAAAAABABABBCA	AAAAAABABABBCD
AAAAAABABABBGA	AAAAAABABABBGB	AAAAAABABABBBD	AAAAAABABABFAB	AAAAAABABABFCA	AAAAAABABABFCD	AAAAAABABAEBBB	AAAAAABABAEBBD
AAAAAABABAEFBB	AAAAAABABAEFBD	AAAAAABABAGABB	AAAAAABABBABAA	AAAAAABABBBABD	AAAAAABABBAFAB	AAAAAABABBBAAA	AAAAAABABBBBAA
AAAAAABABBBBAD	AAAAAABABBBBCA	AAAAAABABBBBCD	AAAAAABABBBFAB	AAAAAABABBEACB	AAAAAABABBEDAB	AAAAAABABBEBBG	AAAAAABABBEBCB
AAAAAABABBEFBB	AAAAAABABBEFCB	AAAAAABABBGACB	AAAAAABABBGBAB	AAAAAABABBGBBB	AAAAAABABBGBCB	AAAAAABABBGFBB	AAAAAABABBGFCB
[illegible]	AAAAAABABFEACB	AAAAAABABFGACB	AAAAAABABGABBG	AAAAAABABGABGA	AAAAAABABGABGD	AAAAAABABGAFBG	AAAAAABABGBBBA
[illegible]	[illegible]	AAAAAABABGBFBD	AAAAAABABGEBBD	AAAAAABABGEFBD	AAAAAABABQABBB	AAAAAABABQABBD	AAAAAABABQAFBB
[illegible]	[illegible]	[illegible]	AAAAAABABDBBBD	AAAAAABABDBFBB	AAAAAABABDBFBD	AAAAAABABDEACB	AAAAAABABDGABB
[illegible]	[illegible]	AAAAAABAFBBFBD	AAAAAABAFBBFBD	AAAAAABAFBEFBB	AAAAAABAFBEFBD	AAAAAABAFBGABG	AAAAAABAHABFBB
[illegible]	[illegible]	[illegible]	AAAAAABAHBBFBA	AAAAAABAHBBFBD	AAAAAABAHBEFBB	AAAAAABAHBGFBB	AAAAAABBAABBAA
[illegible]	[illegible]	[illegible]	AAAAAABBAABBCB	AAAAAABBAABGAB	AAAAAABBABABAB	AAAAAABBABABBB	AAAAAABBABBBAA
[illegible]	[illegible]	[illegible]	AAAAAABBABBBAD	AAAAAABBAFBBAB	AAAAAABBABBBAD	AAAAAABBBABBAF	AAAAAABBBABBBB
[illegible]	[illegible]	[illegible]	AAAAAABFBAAFBA	AAAAAABFBBAFBA	AAAAAADAABBAAB	AAAAAADAABBAFB	AAAAAADAAFBAFB
[illegible]	[illegible]	[illegible]	AAAAAADBABBACB	AAAAAADBAFBAAB	AAAAAADBAFBACB	AAAAAADGABBACB	AAAAAADGAFBACB
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAAAEABGABBA	AAAAAAEABGABBD	AAAAAAEABGEABD
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABBAABABFB	AAAAABBAABACBD	AAAAABBAABACFB
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABBAABEAFB	AAAAABBAABGAAB
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABBABABGGA	AAAAABBABABCBA
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABBABBAAAB	AAAAABBABBABAD
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABBABBBCBB
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABBABBEBBB	AAAAABBABBECBB
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABBABEBBBA
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABBABGBFBA
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABBAHBBFBB	AAAAABBAHBEABD
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABBBABBAAA	AAAAABBBABBABD
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABBBBBBAAB	AAAAABBBBBBABB
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABEABAABGA	AAAAABEABAACBA	AAAAABEABAAFBA
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABEABGABGD	AAAAABEABGACGA	AAAAABEABGAFBD
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABEABBABBA	AAAAABEABBACBA	AAAAABEBBAABGA
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABEFBAABBA
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABEHBBAFBA	[illegible]	AAAAABEHBEACBA
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABGAABACBA	AAAAABGAABACBG
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABGABEAABD	AAAAABGABEABBD
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAAABGAHBAFBA	AAAAABGAHBAFBG	AAAAABGAHBEABG

```
AAAAABGBABAABA AAAAABGBABAABB AAAAABGBABABBA [illegible]
AAAAABGBBBAABG AAAAABGBBEAABG AAAAABGBBEABBG [illegible]
AAAAABGHBAAFBA AAAAABGHBEACBA AAAAABKAABAABB [illegible]
AAAAADGABBAABA AAAAADGABBAABD AAAAADGABBAABG [illegible]
AAAABABAAAABCB AAAABABAAAADAB AAAABABAAAADBB [illegible]
AAAABABAAABDBF AAAABABAAABDCH AAAABABAABAABA [illegible]
AAAABABAABBBBA AAAABABAABBBCA AAAABABAABBBFB [illegible]
AAAABABAABEBCB AAAABABAABEDEB AAAABABAABGBBB [illegible]
[illegible]
```

```
AAABAABABCABBD
[illegible]

AAABAABABCAFBB
[illegible]

AAABAABABCAFBD
AAABAABABGAFBB
AAABAABABQAFBD
AAABAABAFAAFBB
AAABAABAFBDAFD
AAABAABAFCAFBB
AAABAABAHADABA
AAABAABAHBEFBD
AAABAABBBAABAA
AAABAABBBABBCA
AAABAABBFAAABA
AAABAABBFBDAHA
AAABAABBHBDAFD
AAABAABEFBBAFA
AAABAABEHAGACB
AAABAADAAFAACB
AAABAADBHBDACB
AAABAAEABBABBD
AAABAAFAHDBACB
AAABAAFBHABACB
AAABAAFGAABACB
AAABAAGABBAFBD
[illegible]

AAABAABABCBBAB
AAABAABABGBBBD
AAABAABABQBBBB
AAABAABAFAAFBD
AAABAABAFBDAHA
AAABAABAFCAFBD
AAABAABAHADFBB
AAABAABAHBEFBG
AAABAABBBAABAB
AAABAABBBABBCB
AAABAABBFAAABB
AAABAABBFQBABB
AAABAABBHBDAHA
AAABAABEFBBAHA
AAABAABEHBBAFQ
AAABAADAAFAAFB
AAABAADBHBDAFB
AAABAAEABGABBD
AAABAAFBABBBAB
AAABAAFEAABACB
AAABAAFGABGAAB
AAABAAGBBGAABD
[illegible]

AAABAABABCBBBB
[illegible]

AAABAABABCBBBD
AAABAABABGDBBD
AAABAABABQBFBB
AAABAABAFAGABB
AAABAABAFBDFBG
AAABAABAFCBFBD
AAABAABAHBBFBQ
AAABAABAHCAFBB
AAABAABBBAABBD
AAABAABBBBABAB
AAABAABBFADABB
AAABAABBHABACA
AAABAABBHBDAHQ
AAABAABEFQDABB
AAABAABEHBDAFQ
AAABAADAAFEACB
AAABAADEAFBGCB
AAABAAFAAABACB
AAABAAFBACBBAB
AAABAAFEABGAHB
AAABAAFGHBBACB
AAABAAKAHBAABG
AAABAAMAABGAHB
AAABAAMBAFBAFB
AAABAAMEABGAHB
AAABAAMGHBBAHB
AAABABBAAADCBA
AAABABBAABDCHG
AAABABBABABAAA
AAABABBABABFBB
AAABABBABBBAAQ
AAABABBABBBFBD
AAABABBABBEAAD
AAABABBABCDFBA
AAABABBABGDBBG
AAABABBAFBDFBQ
AAABABBAHBBFBD
AAABABBAHEDFBG
AAABABBBABDBAQ
AAABABBBABDGAQ
AAABABBBBABAAF
AAABABBBBBBAAQ
AAABABBBBBDCAA
AAABABBBFABABD
AAABABBBFBBAHQ
AAABABBBHABABD
AAABABBBHEBABQ
AAABABBEBABAAB
AAABABBEBBGAAB
AAABABBEFBBAHD
AAABABBEHABABD
AAABABBEHBDAHQ
AAABABBRHBDAHQ
AAABABKEABEAFB
AAABABMBHBBAHQ
AAABABMRHBDAHB
AAABADMEHBBABB
AAABAFMAHBDABQ
AAABAHMAHBBABQ
AAABAHMRHBDABG
AAABCAFBACBDAB
AAABCAMEABBABQ
AAABCBGEBBEABB
AAACAFKAHBDABB
AAAECBMEABDABB
AABAABBAABABAD
AABAABBAAEACBE
AABAABBABAGFBB
AABAABBABBGCAB
AABAABBACABBBA
AABAABBAHABFBA
AABAABBBBDBABA
AABAABEBBDABBD
AABAABEHBBAHBA
AABAABGBAEAABG
AABAABGBBDAABD
AABAABFFBDABBB
AABAABIFBBABBG
AABAAFKAHBEABB
AABABABBABAABD
AABABABBAFAAFB

AAABAABABCBFBB
AAABAABABGDFBD
AAABAABABQBFBD
AAABAABAFBAAHA
AAABAABAFBEAHB
AAABAABAFDGABB
AAABAABAHBDAHA
AAABAABAHCBFBD
AAABAABBBABBAD
AAABAABBBBABBB
AAABAABBFBAAHA
AAABAABBHADABA
AAABAABBHQDABB
AAABAABEFQEABB
AAABAABGFABABA
AAABAADAAFEAFB
AAABAADEAFEACB
AAABAAFAABGAAB
AAABAAFBAFBAIB
AAABAAFEAFBAIB
AAABAAFGHBBAHB
AAABAAKAHBEABG
AAABAAMAAFBAFB
AAABAAMBHBBABQ
AAABAAMEAFBAFB
AAABABBAAABBAB
AAABABBAAADCBB
AAABABBAABEBBG
AAABABBABABAAD
AAABABBABABFBD
AAABABBABBBBAA
AAABABBABBDAAQ
AAABABBABBEAAG
AAABABBABEBBBA
AAABABBABGDFBD
AAABABBAFBEAHD
AAABABBAHBBFBQ
AAABABBAHGDFBA
AAABABBBABDBFA
AAABABBBABDGFA
AAABABBBBABABF
AAABABBBBBBBAA
AAABABBBBCBBAA
AAABABBBFABCBA
AAABABBBFBDAHD
AAABABBBHABABF
AAABABBEAAEABB
AAABABBEBAEABB
AAABABBEBEBABA
AAABABBEFBDAHA
AAABABBEHADABA
AAABABBRBBDAAA
AAABABDBHBBAHB
AAABABKEHBEABB
AAABABMBHBDAHG
AAABADEAFBAFBA
AAABADMEHBDABQ
AAABAFMEHBBABB
AAABAHMAHBDABQ
AAABAHMBHBDABQ
AAABCAFBAFBDAB
AAABCAMEABGAFB
AAABCFKEABDABB
AAAEAAKAABDABB
AAABCBGEBBDABB
AABAABBAABACAE
AABAABBAAEACBG
AABAABBABBBCAA
AABAABBABBGCAE
AABAABBACABBBB
AABAABBAHABFBB
AABAABBBBEBABA
AABAABEBBEABBD
AABAABGAAEAABA
AABAABGBAEABBG
AABAABGBBDABBD
AABAABGFBDABBD
AABAABIFBBABBG
AABABABAAFEBFB
AABABABBABAAAE
AABABABBAFABAB

AAABAABABCBFBD
AAABAABABQABBB
AAABAABAFAAFBB
AAABAABAFBAAHG
AAABAABAFBEFBG
AAABAABAHAAFBB
AAABAABAHBDAHQ
AAABAABAHQBFBB
AAABAABBBABBAF
AAABAABBBBABBD
AAABAABBFBBAFA
AAABAABBHBBAFA
AAABAABEFABABA
AAABAABEHABABA
AAABAABGFAEABB
AAABAADBAFDAAB
AAABAADEHABACB
AAABAAFAABGAHB
AAABAAFBAFBBAB
AAABAAFEAQBACB
AAABAAGABAAABD
AAABAAKAHBEAFB
AAABAAMAAFBAHB
AAABAAMBHBBAFB
AAABAAMEAFBAHB
AAABABBAAABBBA
AAABABBAABDBAA
AAABABBAABGBBG
AAABABBABABBAA
AAABABBABADAAB
AAABABBABBBBAB
AAABABBABBDBAA
AAABABBABBGAAG
AAABABBABEBCBA
AAABABBABGDFBG
AAABABBAFBEAHG
AAABABBAHBEAHG
AAABABBAHGDFBG
AAABABBBABDBHA
AAABABBBABDGHA
AAABABBBBABBAB
AAABABBBBBBBBA
AAABABBBBCBBBA
AAABABBBFADABB
AAABABBBFBDAHG
AAABABBBHADABD
AAABABBEAAGABB
AAABABBEBAEABD
AAABABBEFABABA
AAABABBEFBDAHG
AAABABBEHBBAFA
AAABABBRBBDAAB
AAABABDBHBDAHB
AAABABKEHBEABG
AAABABMBHBDAHQ
AAABADEAFBAIBA
AAABAFGRHBDABA
AAABAFMRHBDABG
AAABAHMAHBGAFB
AAABAHMBHBDAHB
AAABCAFEABGAHB
AAABCAMEABGAHB
AAABCFMEABDABB
AAAEAAKAHBDABB
AABAABBAAAAABA
AABAABBAABBBAA
AABAABBABABAGA
AABAABBABBBCAB
AABAABBABBGFAB
AABAABBACBBBAA
AABAABBAHBBFBD
AABAABBBBGBABA
AABAABEFBAABBD
AABAABGAHEAABA
AABAABGBBAAABD
AABAABGBBEAABD
AABAABIBBBAABB
AABAABIFBBAHBB
AABABABAAFGBFB
AABABABBABABBD
AABABABBAQAABB
```

```
AABABABBADAACB AABABABBBAAAGB AABABABBBABAGH AABABABBBBAAAE AABABABBBBAACE AABABABBBBABCD AABABABBBBABCE AABABABBBBABGD
AABABABBDBBABDE AABABABBBBBAQB AABABABBBDAAAE AABABABBBDAACE AABABABBBFAAAB AABABABBBFAAQB AABABABBBFABQD AABABABBBFABGR
AABABABBBFBAQB AABABABBBGAABE AABABABBBGABBD AABABABBBGABBE AABABABBBGABGE AABABABBBGAABR AABABABBBGABGR AABABABBBGABGR
AABABABBBQBAAH AABABABFABABAA AABABABFABABBA AABABABFABABBB AABABABFABABCB AABABABFAFABAB AABABABFEBABAB AABABABQBABBBB
AABABABQABEBBB AABABABQABGBBB AABABABQAFEBAB AABABABQBFEFAR AABABABQBFEHAR AABABABQBFGFAR AABABABQBFGHAR AABABBEBBAAABA
AABABBEBBDAABA AABABBEBBEAABA AABABBEFBAAHBA AABABBEFBDAHBA
```

DRAWINGS WITH 21 CROSSINGS:

```
AAAAAAAAABABB AAAAAAAAABBBB AAAAAAAAABAABG AAAAAAAAABABAB AAAAAAAAABABBG AAAAAAAAABABFB AAAAAAAAABBBFB AAAAAAAAABGABG
[illegible]
AAAABABAABGDFB AAAABABAAFABFB AAAABABAAFADFB AAAABABAAFEBAB AAAABABAAFEBCB AAAABABAAFEDFB AAAABABAADABBB AAAABABAADABCB
```

```
AAAABABAAQADBB
AAAABABAAQEDCB
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
AAAABADAAFABCB
AAAABADAAFEDCB
AAAABADARAEDAB
AAAABADARBAACB
AAAABADBABADAB
AAAABADBAFBDIB
AAAABADBRAADAB
AAAABADGAFEDAB
AAAABADGEAEDAB
AAAABADQAFBDCB
AAAABAEABAAFBA
AAAABAEABBAFBA
AAAABAEABGAFBA
AAAABAEABQABBD
AAAABAFAAABDCB
[illegible]

AAAABABAAQADCB
AAAABABAAQGBCB
AAAABABARQBBGF
AAAABABAEABBGF
AAAABABAEFEBAB
AAAABABAHQBFCA
AAAABABAHQEFCB
AAAABABARBEFAA
AAAABABARQBDGH
AAAABABBABABFB
AAAABABBAQBBBB
AAAABABBBFEACB
AAAABABBBGBBGA
AAAABABBBQADCB
AAAABABBBQBDGH
AAAABABBEFADAB
AAAABABBRFBDCH
AAAABABGAFBDCH
AAAABABGBQBDCF
AAAABABGRFBDCF
AAAABABQRQBDCF
AAAABADAAFADAB
AAAABADAAQBDCB
AAAABADARAEDCB
AAAABADARQEACB
AAAABADBABBAFB
AAAABADBAFBGCB
AAAABADBRBBDCB
AAAABADGAFBDFB
AAAABADGHFBAFB
AAAABADQAFBBCB
AAAABAEABAAFBD
AAAABAEABBAFBD
AAAABAEABGAFBD
AAAABAEABQABGD
AAAABAFAAFBDCF
AAAABAFARFBDCB
AAAABAFBRFBDIB
AAAABAFGRQBACB
AAAABAHAAFBACH
[illegible]

AAAABABAAQBABH
AAAABABAAQGDBB
AAAABABARQBFGF
AAAABABAEABDGH
AAAABABAEFABAB
AAAABABAHBBAFA
AAAABABARAADGA
AAAABABARBEFAB
AAAABABARQEFCB
AAAABABBAFABCB
AAAABABBAQBBCF
AAAABABBBFADCB
AAAABABBBGBDBA
AAAABABBBQADGB
AAAABABBBQBDGF
AAAABABBHFAACB
AAAABABBRQAACB
AAAABABGAQBACH
AAAABABGRQBDCH
AAAABABGRQBACH
AAAABABQBQBDCH
AAAABADAAFADCB
AAAABADAHFAAFB
AAAABADARFAAAB
AAAABADBAAABAB
AAAABADBABBBCB
AAAABADBAQBDCB
AAAABADBRQBDCB
AAAABADGAFBDIB
AAAABADGHFBAIB
AAAABADQRAEDAB
AAAABAEABAEABD
AAAABAEABBEBBD
AAAABAEABGEAGD
AAAABAEABQEABD
AAAABAFAAFBDIB
AAAABAFARQBACB
AAAABAFBAFBGCB
AAAABAFBRQBDCB
AAAABAFGRFBDCB
AAAABAHAAFBBCF
AAAABAHABABDCF
AAAABAHABBBDCB
AAAABAHABQBACH
AAAABAHBABQBCB
AAAABAHBBAFDAF
AAAABAHBRQBACB
AAAABAHGAQBDCB
AAAABAHQBABFCH
AAAABANABAAFCD
AAAABANABGEAGD
AAAABANABQEDGD
AAAABBBAAABBBB
AAAABBBAABCAAB
AAAABBBABAAAAA
AAAABBBABABFAD
AAAABBBABBBFAB
AAAABBBABEAAAD
[illegible]
AAAABABAAABBCD

AAAABABAAQBBCF
AAAABABARAAAAA
AAAABABAHQBFGH
AAAABABAEABDGH
AAAABABAEFADAB
AAAABABAHFEFCB
AAAABABARAAFGA
AAAABABARFAFAB
AAAABABBABAAAA
AAAABABBAFADCB
AAAABABBAQBBCH
AAAABABBBFBBCF
AAAABABBBGBDGA
AAAABABBBQBAGF
AAAABABBBQBDGH
AAAABABBHQBABF
AAAABABBRQBAGF
AAAABABGAQBGCB
AAAABABGBQEDBB
AAAABABGRQBDCF
AAAABABQBQBFCH
AAAABADAAFBBFB
AAAABADAHFEAFB
AAAABADARFADAB
AAAABADBAAADAB
AAAABADBAFAAAB
AAAABADBEAABAB
AAAABADGAABGCB
AAAABADGAFBGCB
AAAABADGHQBACB
AAAABADQRBBDCB
AAAABAEABAEBBD
AAAABAEABBEDBD
AAAABAEABGEBBD
AAAABAEBBBABBA
AAAABAFAAQBDCB
AAAABAFBAABBCF
AAAABAFBAQBDCB
AAAABAFGABBDCB
AAAABAFGRQBACB
AAAABAHAAFBBCH
AAAABAHABABDCH
AAAABAHABFBBCF
AAAABAHABQBFCB
AAAABAHBAFBBIB
AAAABAHBBAFDAH
AAAABAHBBQBBCH
AAAABAHGBBBACB
AAAABAHQAABFCB
AAAABANABAAFBD
AAAABANABGEDGD
AAAABANQBAADGA
AAAABBBAABABAA
AAAABBBAABCABB
AAAABBBABAAAAD
AAAABBBABABFGB
AAAABBBABBBFBB
[illegible]
[illegible]
AAAABBBAHBBFBB
AAAABBBBABABBB
AAAABBBBBBABBD
AAAABBBBBGAABD
[illegible]
AAAABBEABBAFBD
AAAABBEFBGAABA
AAAABBEHBBABDA
AAAABBNFBEABAB
[illegible]
AAAABENQBACBAH
AAAABGNBBACBAH
AAAACADAAFBDCB
AAAACAFBAFBDCB
AAAACAFGAFBACB
AAAADANABACBAH
AAAADBEABEACBA
AAAADBEBBGABGA
[illegible]
AAAADBNFBACBBA
AAABAABAAAABBA
AAABAABAAAABCF

AAAABABAAQBBCH
AAAABABARABFGF
AAAABABARQEFGB
AAAABABAEAEBAB
AAAABABAEFBBAH
AAAABABAHQAFBB
AAAABABARABAGH
AAAABABARFBFAH
AAAABABBABAACA
AAAABABBAFBBCB
AAAABABBAQBDBF
AAAABABBBBAAGA
AAAABABBBQAAGB
AAAABABBBQBAGH
AAAABABBBQBDGF
AAAABABBHQBACH
AAAABABBRQBDCH
AAAABABGAQGDCB
AAAABABGBQEDCB
AAAABABGRQGDCB
AAAABABQRQBDCF
AAAABADAAFBBIB
AAAABADARAAACB
AAAABADARFADCB
AAAABADBAABGCB
AAAABADBAFAACB
AAAABADBEAADAB
AAAABADGAAEDAB
AAAABADGAFEAAB
AAAABADGRAEDAB
AAAABADQRQBDCB
AAAABAEABAEDBD
AAAABAEABGAABA
AAAABAEABGEBGD
AAAABAEBBGABBA
AAAABAFARABDCF
AAAABAFBAABGCB
AAAABAFBEAFBAF
AAAABAFGABBACB
AAAABAFQAABDCB
AAAABAHAAFBDCF
AAAABAHABABFCH
AAAABAHABFBDCF
AAAABAHABQGACB
AAAABAHBAFBDCH
AAAABAHBBBBBCB
AAAABAHBBQBDCB
AAAABAHGBFBDCB
AAAABAHQBFBBCB
AAAABANABAEACD
AAAABANABQAACD
AAAABANGBAADGD
AAAABBBAABABBD
AAAABBBAABCBAB
AAAABBBABAAFAD
AAAABBBABABFED
AAAABBBABBBFBD
AAAABBBABHGBAB
AAAABBBAEABBAA
AAAABBBAHBCABB
AAAABBBBABBAAD
AAAABBBBBBBAAD
AAAABBBBBGAAGA
AAAABBDNAABAAB
AAAABBEABEAABA
AAAABBEFBAAFBA
AAAABBEHBBAFBA
AAAABBHBBABAAB
AAAABEFBAABDAB
AAAABENFBACBAA
AAAABGNBBACBGA
AAAACADBABBACB
AAAACAFAABBACB
AAAACAFBAFBDAB
AAAADANABACBGA
AAAADBEABGACGA
AAAADBEBBEACBA
AAAADBEHBEABDA
AAAADBNFBEABGA
AAABAABAAAABBD
AAABAABAAADBAB

AAAABABAAQBDBF
AAAABABARFEFAB
AAAABABARQGBBB
AAAABABAEAEBGB
AAAABABAEFBDAH
AAAABABAHQAFCB
AAAABABARABFGF
AAAABABARFEFCB
AAAABABBABABCA
AAAABABBAFBGCF
AAAABABBAQBDCH
AAAABABBBGADBA
AAAABABBBQABBB
AAAABABBBQBBBB
AAAABABBRQBDGH
AAAABABBRBADCB
AAAABABBRQBDGF
AAAABABGBFBDCF
AAAABABGBQEDGB
AAAABABQAQBBCB
AAAABADAAFAAAB
AAAABADAAFEAAB
AAAABADARAADAB
AAAABADARFEAAB
AAAABADBABAAAB
AAAABADBAFABAB
AAAABADBHFBAFB
AAAABADGABBAFB
AAAABADGAFEACB
AAAABADGRBBDCB
AAAABAEABAAABD
AAAABAEABAEDGD
AAAABAEABGAAGD
AAAABAEABGEDBD
AAAABAEBBGABGA
AAAABAFARAGACB
AAAABAFBAAFBAF
AAAABAFBEAFDAF
AAAABAFGAFBAIB
AAAABAFQAFBDCB
AAAABAHAAFBDIB
AAAABAHABAGACB
AAAABAHABFBDCH
AAAABAHBAABACH
AAAABAHBAFBDIB
AAAABAHBBBBDCB
AAAABAHGAABGCB
AAAABAHGBQBACB
AAAABAHQBFBDCB
AAAABANABAEAGD
AAAABANABQAAGD
AAAABANGBGABGA
AAAABBBAABABFB
AAAABBBAABEAAB
AAAABBBABABAAF
AAAABBBABBAFAD
AAAABBBABBCAAA
AAAABBBABEAAGA
AAAABBBAEABBGA
AAAABBBAHBEABD
AAAABBBBABBABD
AAAABBBBBBBABB
AAAABBBBBGBABA
AAAABBDBAABBAB
AAAABBEABGAAGA
AAAABBEFBBABBA
AAAABBEHBEABDA
AAAABBNABACBAH
AAAABEFBEABBAB
AAAABENQBACBAA
AAAABGNBBEABGA
AAAACADBAFBACB
AAAACAFAAFBACB
AAAACAFGAABDAB
AAAADANEBACBGA
AAAADBEBBACBBA
AAAADBEFBBABBA
AAAADBEHBEACBA
AAAADBNHBAAGAA
AAABAABAAABACB
AAABAABAAADBBD

AAAABABAAQBDCH
AAAABABARFGFAB
AAAABABARQBRGB
AAAABABAEAGBGB
AAAABABAEFEDAB
AAAABABAHQBABH
AAAABABARABFGH
AAAABABARFGFAB
AAAABABBABABFB
AAAABABBAFBGCH
AAAABABBAQBGCB
AAAABABBBGADGA
AAAABABBBQABCB
AAAABABBBQBBBH
AAAABABBERABAA
AAAABABBRFAAAB
AAAABABFBFABAB
AAAABABGBFBDCH
AAAABABGBQGDCB
AAAABABQBQBBBB
AAAABADAAFAAFB
AAAABADAAFEAFB
AAAABADARAADCB
AAAABADARFEDAB
AAAABADBABAACB
AAAABADBAFADAB
AAAABADBHFBAIB
AAAABADGABEAAB
AAAABADGAFEDAB
AAAABADGRQBDCB
AAAABAEABAADBD
AAAABAEABBAABA
AAAABAEABGADBD
AAAABAEABGEDGD
AAAABAFAAABBCB
AAAABAFARAGDCB
AAAABAFBAAFDAF
AAAABAFBHFBAIB
AAAABAFGAQBACB
AAAABAFQRFBDCB
AAAABAHAAQBBCB
AAAABAHABAGBCB
AAAABAHABFBFCH
AAAABAHBAABDCH
AAAABAHBAFBGCF
AAAABAHBBFBBCF
AAAABAHGABBDCB
AAAABAHQAABBCB
AAAABANABAA[illegible]CD
AAAABANABAEBCD
AAAABANABQABCD
AAAABANQBAEBGD
AAAABBBAABBAAA
AAAABBBAABEABD
AAAABBBABABAAH
AAAABBBABBBBAD
AAAABBBABBCBBB
AAAABBBABEBBGA
AAAABBBAEABBGB
AAAABBBAHBEAFB
AAAABBBBABBAFB
AAAABBBBBBBABD
AAAABBBBEBABAA
AAAABBDBAABAAB
AAAABBEABGAAGD
AAAABBEFBBABDA
AAAABBEHBEAFBA
AAAABBNABFABGD
AAAABEFBEABDAF
AAAABGHBAABDAB
AAAABKNABACBCA
AAAACADGAABACB
AAAACAFBAABDAB
AAAACAFGABBACB
AAAADANFBACBGA
AAAADBEBBBABBA
AAAADBEFBBABDA
AAAADBNABAABAA
AAAADBNBBEABGA
AAABAABAAABBAD
AAABAABAAADBCB

AAAABABAAQEDBB
AAAABABARGBFGA
AAAABABARQGFBB
AAAABABAEBBBAA
AAAABABAEFGBAB
AAAABABAHQBFCF
AAAABABARBAACA
AAAABABARQAFGB
AAAABABBABADFB
AAAABABBAGBABA
AAAABABBBBADGA
AAAABABBBGBAGA
AAAABABBBQABGB
AAAABABBBQBBCF
AAAABABBEBBGAA
AAAABABBRFAACB
AAAABABFBGABBA
AAAABABGBQBABF
AAAABABGEFEDAB
AAAABABQBQBBCH
AAAABADAAFABAB
AAAABADAAFEDAB
AAAABADARAEACB
AAAABADARFEDCB
AAAABADBABABAB
AAAABADBAFBDFB
AAAABADBHQBACB
AAAABADGABEACB
AAAABADGAQBDCB
AAAABADQAABBCB
AAAABAEABAADGD
AAAABAEABBADBD
AAAABAEABGADGD
AAAABAEABQAABD
AAAABAFAABBBCB
AAAABAFARBGDCB
AAAABAFBABBBCB
AAAABAFBHQBACB
AAAABAFGHABACB
AAAABAHAABGBCB
AAAABAHAAQBDCB
AAAABAHABAGDCB
AAAABAHABFGBCB
AAAABAHBAABGCF
AAAABAHBAFBGCH
AAAABAHBBFBDCF
AAAABAHGABGACB
AAAABAHQAABDCB
AAAABANABAAAGD
AAAABANABAEDCD
AAAABANABQADGD
AAAABANQBAEDGD
AAAABBBAABBAAD
AAAABBBAABEAFB
AAAABBBABABABF
AAAABBBABBBBBB
AAAABBBABBCFAA
AAAABBBABGAFGD
AAAABBBAEACBAB
AAAABBBAHBGAFB
AAAABBBBAEABBA
AAAABBBBBEAAGA
AAAABBBBEBABAB
AAAABBDBEABBAB
AAAABBEABGAFBD
AAAABBEFBEAFBA
AAAABBEHBGAFBA
AAAABBNBBGABGA
AAAABEFBRABDAB
AAAABGHBBABDAB
AAAABKNABACBGA
AAAACADGAABDCB
AAAACAFBABBAAB
AAAACAFGABBDAB
AAAADBEABAABDA
AAAADBEBBBABDA
AAAADBEFBGABBA
AAAADBNABACBAA
AAAADBNBBGABGA
AAABAABAAABBAF
AAABAABAAAEAAB
```

```
AAABAABAAAEBBB AAABAABAAAGBAB AAABAABAAAGBBB AAABAABAAAGBCB AAABAABAABABAD AAABAABAABABAG AAABAABAABABCD AAABAABAABABCB
AAABAABAABABFD AAABAABAAGBBAD AAABAABAABBBCD AAABAABAABBBFD AAABAABAABBBHD AAABAABAABDBAD AAABAABAABDBAG AAABAABAABDBCD
AAABAABAABDBCG AAABAABAABDBFD AAABAABAABEBHB AAABAABAABGBAG AAABAABAABGBCG AAABAABAACABBD AAABAABAACBBAF AAABAABAACBBBF
AAABAABAACBBBQ AAABAABAACBBCD AAABAABAACBBCF AAABAABAAFBBAF AAABAABAAFBBCF AAABAABABAAACD AAABAABABAABAA AAABAABABAABAD
AAABAABABAABCA AAABAABABAABCD AAABAABABAAFAB AAABAABABABBAF AAABAABABABBBF AAABAABABABBCF AAABAABABABFAA AAABAABABABFAD
AAABAABABABFBF AAABAABABABFCF AAABAABABADBAB AAABAABABADBCB AAABAABABADFCB AAABAABABAEBAB AAABAABABAEBCB AAABAABABAEFCB
AAABAABABAGBAB AAABAABABAGBBB AAABAABABAGBCB AAABAABABAGFBB AAABAABABAGFCB AAABAABABBABAG AAABAABABBAFAA AAABAABABBAFAD
AAABAABABBBBBQ AAABAABABBBFAA AAABAABABBBFAD AAABAABABBBFBQ AAABAABABBDAAQ AAABAABABBDBAA AAABAABABBDBAD AAABAABABBDBBQ
AAABAABABBDBCA AAABAABABBDBCD AAABAABABBDFAB AAABAABABBDFBQ AAABAABABBDFCA AAABAABABBDFCD AAABAABABBEBAD AAABAABABBEBCD
AAABAABABBEFAB AAABAABABBEFCD AAABAABABBGBBG AAABAABABBGFAB AAABAABABBGFBG AAABAABABCABAB AAABAABABCBBAA AAABAABABCBBAD
AAABAABABCBFAB AAABAABABCDBBB AAABAABABCDBBD AAABAABABCDFBB AAABAABABCDFBD AAABAABABCEBBB AAABAABABCEBBD AAABAABABCEFBB
AAABAABABCEFBD AAABAABABFABAB AAABAABABFBFAB AAABAABABGDBBG AAABAABABGDFBG AAABAABABQDBBB AAABAABABQDBBD AAABAABABQDFBB
AAABAABABDDFBD AAABAABABQEBBB AAABAABABQEBBD AAABAABABQEFBB AAABAABABQEFBD AAABAABAFADFBA AAABAABAFADFBB AAABAABAFADFBD
AAABAABAFAEFBB AAABAABAFAEFBD AAABAABAFBBFBQ AAABAABAFBDAHQ AAABAABAFBDFBQ AAABAABAFBGFBG AAABAABAFCDFBB AAABAABAFCDFBD
AAABAABAFCEFBB AAABAABAFCEFBD AAABAABAFGBFBA AAABAABAFGBFBD AAABAABAFGAFBB AAABAABAFQAFBD AAABAABAFQBFBB AAABAABAFQBFBD
AAABAABAHAAABA AAABAABAHAAFBA AAABAABAHAAFBD AAABAABAHABFBF AAABAABAHADFBA AAABAABAHADFBD AAABAABAHAEFBB AAABAABAHAGFBB
AAABAABAHBAAHA AAABAABAHCAFBA AAABAABAHCAFBD AAABAABAHCBFBF AAABAABAHCBFBQ AAABAABAHCDFBD AAABAABAHBAFBA AAABAABAHGAFBD
AAABAABAHGAFBG AAABAABAHGBFBA AAABAABAHGBFBD AAABAABAHGBFBA AAABAABAHGDFBD AAABAABAHBDFBG AAABAABAHQAFBB AAABAABAHQBFBD
AAABAABAHDDFBB AAABAABBBAAAAA AAABAABBBAAACA AAABAABBBAABAD AAABAABBBAABCA AAABAABBBAABCB AAABAABBBABBBF AAABAABBBABBCD
AAABAABBBABBCF AAABAABBBADBAB AAABAABBBADBBB AAABAABBBADBBD AAABAABBBBAAAB AAABAABBBBAACB AAABAABBBBABAD AAABAABBBBABBG
AAABAABBBABBCB AAABAABBBBBBAD AAABAABBBBBABB AAABAABBBBBBBD AAABAABBBBBBCB AAABAABBBBDAAB AAABAABBBBDAAB AAABAABBBBDBBD
AAABAABBBCABBD AAABAABBBCBBAF AAABAABBBCBBBD AAABAABBFBAAHG AAABAABBFBBAFQ AAABAABBFBDAFQ AAABAABBFBDAHB AAABAABBFBDAHQ
AAABAABBFQAABB AAABAABBFQDABB AAABAABBHAAABA AAABAABBHAAACB AAABAABBHBAACA AAABAABBHBAAHA AAABAABBHBAAHB AAABAABBHQAABB
AAABAABEFADABA AAABAABEFAGABB AAABAABEFBBAFQ AAABAABEFBBAHQ AAABAABEFBDAFQ AAABAABEFBDAHA AAABAABEFQGABB AAABAABEHABACA
AAABAABEHAEACB AAABAABEHBDAHQ AAABAABGFAGABB AAABAABGFBBAHQ AAABAADAAABBCB AAABAADAACBBCB AAABAADAAFAAAB AAABAADAAFAAHB
AAABAADAAFBBFB AAABAADAAFBBHB AAABAADAAFEAAB AAABAADAAFEAHB AAABAADAHADACB AAABAADAHQDACB AAABAADBAADAAB AAABAADBAADACB
AAABAADBABABAB AAABAADBABBBCB AAABAADBABBBHB AAABAADBABBGCB AAABAADBABBGHB AAABAADBABDBAB AAABAADBABDGAB AAABAADBACBBCB
AAABAADBACBBHB AAABAADBACBGCB AAABAADBACBGHB AAABAADBAFAAAB AAABAADBAFAACB AAABAADBAFAAFB AAABAADBAFAAHB AAABAADBAFBBCB
AAABAADBAFBBHB AAABAADBAFBGCB AAABAADBAFBGHB AAABAADBHBAACB AAABAADBHBAAFB AAABAADBHBAAHB AAABAADEAABGCB AAABAADEAADACB
AAABAADEACBGCB AAABAADEAFBGFB AAABAADEAFBGHB AAABAADEAFEAAB AAABAADEAFEAHB AAABAADEAQDACB AAABAADEHBEAHB AAABAADGABBBCB
AAABAADGACBGCB AAABAADGAFBGCB AAABAADGHBEACB AAABAADGHBEAHB AAABAAEABAAABD AAABAAEABAABBD AAABAAEABAEABD AAABAAEABBAFBD
AAABAAEABBEBBD AAABAAEABGAFBD AAABAAEABQAABD AAABAAEABQEABD AAABAAEAFBAFBA AAABAAEAFBAFBD AAABAAEBBAABBA AAABAAEBBAAABA
AAABAAEBBBAABD AAABAAEBBBABBA AAABAAEBBGAABA AAABAAEBBGAABD AAABAAEBBGABBA AAABAAEBFBAABA AAABAAEBFBAABD AAABAAFAAABAAD
AAABAAFAABBBAB AAABAAFAABBBCB AAABAAFAAFBBAB AAABAAFAAFBBCB AAABAAFBABBBCB AAABAAFBABBBHB AAABAAFBABBGCB AAABAAFBABBGHB
AAABAAFBACBBAF AAABAAFBACBBCB AAABAAFBACBBHB AAABAAFBAFBBAF AAABAAFBAFBBCB AAABAAFBAFBBHB AAABAAFBAFBGAF AAABAAFBAFBGCB
AAABAAFBAFBGHB AAABAAFBHGBACB AAABAAFEAABAAB AAABAAFEABBGAB AAABAAFEABBGCB AAABAAFEAFBGAB AAABAAFEAFBGCB AAABAAFEHABACB
AAABAAFEHBBACB AAABAAFGAABGAB AAABAAFGABBGAB AAABAAFGAFBGAB AAABAAFGADBACB AAABAAFGHABACB AAABAAGAABABBA AAABAAGAABABBD
AAABAAGAABABBG AAABAAGABAABBD AAABAAGABBABBG AAABAAGABBAFBG AAABAAGABBEBBD AAABAAGABBEFBD AAABAAGABGABBB AAABAAGABGAFBG
AAABAAGAHADABD AAABAAGAHBAFBA AAABAAGAHBAFBD AAABAAGAHBAFBG AAABAAGAHQDABD AAABAAGBABABBA AAABAAGBAGAABA AAABAAGBAGAABD
AAABAAGBAGAABG AAABAAGBBAAABD AAABAAGBBADABD AAABAAGBBGAABG AAABAAGBBGDABG AAABAAGBBGDABQ AAABAAGBHGDABA AAABAAGBHGDABG
AAABAAGBHGDABQ AAABAAKAABABBB AAABAAKAABDBBB AAABAAKBHBAABB AAABAAKBHBDABG AAABAAKBHBDAFB AAABAAHAABBBBB AAABAAHAABDBBB
AAABAAHBABBAFQ AAABAAHBABBBBB AAABAAHBABBBHB AAABAAHBABBGBB AAABAAHBABBGHB AAABAAHBABDBBB AAABAAHBABFABQ AAABAAHBAGBABQ
AAABAAHBHBBAHQ AAABAAHEABBGBB AAABAAHEHBBABQ AAABAAHEHBDABQ AAABAAHEHBDAHB AAABAAHGABBABQ AAABAAHGABGAFB AAABAAHGAFBAFB
AAABAAHGHBBAFB AAABABBAAABAAA AAABABBAAABAAD AAABABBAAABBAA AAABABBAAABBBD AAABABBAAABCAB AAABABBAAABCBD AAABABBAAADBAB
AAABABBAAADBBD AAABABBAAAGAAB AAABABBAABDBAD AAABABBAABDBAQ AAABABBAABDBFD AAABABBAABDBFQ AAABABBAABDBHA AAABABBAABDBHG
AAABABBAABDCAD AAABABBAABDCAQ AAABABBAABDCHD AAABABBAABDCHQ AAABABBAABEAAD AAABABBAABEAHD AAABABBAABEBBD AAABABBAABGBAG
AAABABBAABGCAG AAABABBAABGCHG AAABABBAACBBAA AAABABBAACBBBD AAABABBAACBBBQ AAABABBAACBBHB AAABABBAACDBAB AAABABBAACDBBD
AAABABBAACDBBQ AAABABBAAEBBBA AAABABBAAEBCBA AAABABBABABAAF AAABABBABABBAD AAABABBABABCAA AAABABBABABCAB AAABABBABABFAA
AAABABBABABFAB AAABABBABADAAA AAABABBABADAAD AAABABBABADBAA AAABABBABADBAB AAABABBABAEAAB AAABABBABAGAAB AAABABBABBBBAD
AAABABBABBBBBQ AAABABBABBBCAD AAABABBABBBFAA AAABABBABBBFAB AAABABBABBBFBQ AAABABBABBDBAD AAABABBABBDBAG AAABABBABBDCAD
AAABABBABBDCAG AAABABBABBDFAA AAABABBABBDFAB AAABABBABBEBAB AAABABBABBGBAB AAABABBABBGBBB AAABABBABBGBBB AAABABBABBGCAB
AAABABBABBGFBB AAABABBABBGFBG AAABABBABCBBAA AAABABBABCBBAB AAABABBABCBBBB AAABABBABCBBBD AAABABBABCBFBB AAABABBABCBFBD
AAABABBABCDBBB AAABABBABCDBBD AAABABBABCDFBB AAABABBABCDFBD AAABABBABEDBBQ AAABABBABEDFBQ AAABABBABGDBBD AAABABBABGDFBQ
AAABABBAFABCBB AAABABBAFABCBD AAABABBAFABFBB AAABABBAFABFBD AAABABBAFADCBB AAABABBAFADFBB AAABABBAFADFBD AAABABBAFBBFBQ
AAABABBAFBGFBB AAABABBAFBGFBG AAABABBAFCBFBB AAABABBAFCBFBD AAABABBAFCDFBD AAABABBAFEDCBG AAABABBAFEDFBB AAABABBAHAGFBD
AAABABBAHADFBD AAABABBAHBEAHD AAABABBAHBEFBD AAABABBAHEBFBA AAABABBAHEDFBQ AAABABBAHGDFBD AAABABBAHGDFBD AAABABBBAADCAA
AAABABBBAADCAB AAABABBBABDBAD AAABABBBABDBFG AAABABBBABDBFQ AAABABBBABDBHB AAABABBBABDBHD AAABABBBABDCAD AAABABBBABDCHD
AAABABBBABDGAD AAABABBBABDGFG AAABABBBABDGFQ AAABABBBABDGHB AAABABBBABDGHD AAABABBBACDBAQ AAABABBBACDBHA AAABABBBACDBHB
AAABABBBACDGAQ AAABABBBACDGHA AAABABBBACDGHB AAABABBBBAHBAD AAABABBBBABBAF AAABABBBBABBBB AAABABBBBABBBD AAABABBBBABCAB
AAABABBBBABCBB AAABABBBBABCBD AAABABBBBADAAD AAABABBBBADBAB AAABABBBBADBBB AAABABBBBADBBD AAABABBBBADCAA AAABABBBBADCBB
AAABABBBBBBBAB AAABABBBBBBCAB AAABABBBBBBCBB AAABABBBBBBCBD AAABABBBBBDBAB AAABABBBBBDBBB AAABABBBBBDBBD AAABABBBBBDCAB
AAABABBBBCBBAB AAABABBBBCDBAA AAABABBBFBBAFD AAABABBBFBBAFQ AAABABBBFEBABQ AAABABBBFEBCBA AAABABBBHBBAFD AAABABBEAADGAB
AAABABBEAAEABD AAABABBEAAGAAB AAABABBEABDGAD AAABABBEABDGAQ AAABABBEABDGFD AAABABBEABDGFQ AAABABBEABDGHA AAABABBEABDGHB
AAABABBEABEAAD AAABABBEABEAHD AAABABBEACBGHB AAABABBEACDBAB AAABABBEBABAAA AAABABBEBABAAD AAABABBEBABABF AAABABBEBADAAB
AAABABBEBAGABB AAABABBEBDBAAQ AAABABBEBBDAAQ AAABABBEBBEAAD AAABABBEBBEAAB AAABABBEBBGAAB AAABABBEBEBABD AAABABBEFAEABB
AAABABBEFAEABD AAABABBEFBBAFQ AAABABBEFBBAHQ AAABABBEFBDAHQ AAABABBEFBEAHD AAABABBEFBEAHG AAABABBEFBBAHB AAABABBEFEBABA
AAABABBEFEDABQ AAABABBEHABABF AAABABBEHAEABB AAABABBEHAGABB AAABABBEHEBABA AAABABBEHEBABD AAABABBGABDGAB AAABABBGABDBAD
AAABABBGBBDAAD AAABABBGBBDAAG AAABABBGBBDAAQ AAABABBGFADABB AAABABBGFADABD AAABABBGFADCBA AAABABBGFBDAHD AAABABBGFBDAHG
AAABABBGFBDAHQ AAABABBGHBDAHD AAABABDBABBBAB AAABABDBABBCAB AAABABDBABBGAB AAABABDBABDBAB AAABABDBABDCAB AAABABDBAHDCHB
AAABABDBABDGAB AAABABDBACBBAB AAABABDBACBBAB AAABABDBHBBAFB AAABABDEABBGAB AAABABDEABDGAB AAABABDEABDGFB AAABABDGHBDAHB
AAABABFAABGAAB AAABABFAABGAHB AAABABFAHBBAHB AAABABFBABBBAB AAABABFBABBCAB AAABABFBABBGAB AAABABFBACBBAB AAABABFEABGAAB
AAABABFEABGAHB AAABABFEHBBAHB AAABABGAAEEBBG AAABABGAHBEFBB AAABABGEBAEABD AAABABKEABDGFB AAABABKEHBEAFB AAABABHAHBBAFD
AAABABHAHBBAHQ AAABABHAHBDAHQ AAABABHGHBBAFQ AAABABHEABDGFB AAABABHEABBAHB AAABABHEHBDAHB AAABABHEHBGAFB AAABABHEHBGAHB
AAABABHGHBDAHG AAABABHGHBDAHQ AAABADEAFBAFBD AAABADEAFBAIBD AAABADEAFEAABA AAABADEAFEAFBA AAABADEHFBAFBA AAABADEHFBAIBA
AAABADEHFEACBA AAABADEGFBACBD AAABADERFEACBA AAABADKAABACBB AAABADKAABACBB AAABADKAABEABB AAABADKAHEDABB AAABADKEABEABB
AAABADKEABEABG AAABADKEHBEABB AAABADKEHBEABG AAABADHAABGABB AAABADHAHEDABG AAABADHEABGABB AAABADHEABGABB AAABADHEHBGABD
AAABADHEHBGABB AAABADHEHBGABG AAABAFGAABACBA AAABAFGAABACBD AAABAFGAABACBB AAABAFGABBACBB AAABAFGEABEABD AAABAFGEABEABB
AAABAFGB[illegible]EABD AAABAFGEBGDABA AAABAFGEBGDABG AAABAFGEBGEABD AAABAFGEHBEABD AAABAFGEHBEABB AAABAFGEHGDABA AAABAFGEHGDABQ
AAABAFGBHGACBA AAABAFGBABAABA AAABAFGBABAABD AAABAFGBABAABB AAABAFGBABACBA AAABAFGBBBAABA AAABAFGBBBAABD AAABAFGBHBACBA
AAABAFGBGBDABA AAABAFGBGBDABD AAABAFGBGBGABA AAABAFGBGBAABD AAABAFGBGBACBA AAABAFGBGBDABA AAABAFGBHBAABA AAABAFGBHBAABD
AAABAFGBHBAABG AAABAFGBHGDABA AAABAFGBHGDABQ AAABAFKAABACBB AAABAFKAABEABB AAABAFKAHBEAFB AAABAFKEABEABB AAABAFKEHBEABB
```

```
AAABAFKRABAABB AAABAFKRHBAABB AAABAFKRHBDABB AAABAFMAHBBABQ AAABAFMEABGABB AAABAFMEHBDABQ AAABAFMEHBGABB AAABAHKAABACBB
AAABAHKAABEAFB AAABAHKRABAABB AAABAHKBHBAABB AAABAHKBHBDABB AAABAHMAABBCBB AAABAHMBABBCBB AAABAHMBABBCHB AAABAHMBHBBAHQ
AAABAHMGHFBAFB AAABAHMBABGABB AAABAHMBABGAHB AAABAHMGHBBAFB AAABAHMGHBGABB AAABAHMGHBGAHB AAABAHMGHFBAHB AAABBADBAAAACB
AAABBADBACAACB AAABBADBAFAAFB AAABBADBAFAAHB AAABBADBHFAACB AAABBADEHFEAFB AAABBADERFEACB AAABBADGHBEACB AAABBADGHFEACB
AAABBAEABAAABD AAABBAEABAABBD AAABBAEABGABBD AAABBAEABQAABD AAABBAEBBAABBA AAABBAFEABGAHB AAABBAFGABGAHB AAABBBEBBAAABA
AAABBBEBBEAABA AAABCEFBEABDAB AAABCAFAABBDAB AAABCAFAABBDCB AAABCAFAABGAHB AAABCAFAAFBDAB AAABCAFBABBDCB AAABCAFBACBDCB
AAABCAFBAFBAIB AAABCAKBADBACB AAABCAFGAQBACB AAABCAGEBADABD AAABCAGEBCEABD AAABCAMAABBABQ AAABCAMBAFB[illegible]FB AAABCAMGABBABQ
AAABCBGEBADABD AAABCBGEBBJABD AAABCBGEBBJABG AAABCBGEBCDABD AAABCFGEBGDABD AAABCFKEABDABG AAABCFMEABDABG AAABCHMBABDABB
AAABDBEBBAACBA AAABDBEBBEABDB AAABDBEBBEACBA AAABDBEBBGACBA AAABDBEHBAACBA AAABDBEHBEACBA AAABDBERBGACBA AAABEAHBAQBACB
AAABEAHEABBDAB AAABEAHEABBDCB AAABEAHBABBDAB AAABEAHGACBDAB AAABEAHGAFBDAB AAABFBGBBAACBA AAABFBGBBEACBA AAABFBGBBGACBA
AAABFBGHBAACBA AAABFBGHBEACBA AAABFBGHBEAFBA AAABFBGHBGAFBA AAABGAHBABBGCB AAABGAHBACBGCB AAABGAHBAFBGCH AAABGAHEABBDCB
AAABGAHEACBGCB AAABGAHEAFBGCH AAABGAHEAQBGCB AAACABGABBMALB AAACADEABBAABD AAACADEABBACBD AAACADEAFBAABA AAACADEAFBAABD
AAACADEAFBAFBA AAACADKAHBAABB AAACAFGABBACBD AAACAFGABBACBD AAACAFGABBACBG AAACAFKAABDABB AAACAFKAHBAABB AAACAFKAHBAABG
AAACAFDAHBDABB AAADAAFBAFBBAB AAADAAFBAFBBAF AAADBEFBEABDAB AAADBEFBEABDAF AAADCAFBAABDAB AAADCAFBACBDAB AAADCAFBAFBDAB
AAAEAAFEABBABD AAAEAAEABGABBD AAAEAAKAABEABB AAAEAAKAHBEABB AAAEAAKAHBEABG AAAEAAMAABDABB AAAEAAMAHBDABB AAAEAHKAABDABB
AAAEAHKAHBAABB AAAEAHKAHBAABG AAAEAHKAHBEABB AAAEAHKAHBEABG AAAEAHMAHBDABB AAAECAKEABEABB AAAECAKEABEABG AAAECAKEABJABB
AAAECAKEABJABG AAAECAHEABDABB AAAECBGEABMABG AAAECBGEBBDABD AAAECBGEBBEABD AAAECBGEBBJABG AAAECBGEBBMABG AAAECBGEBBDABG
AAAECBGRBBDABG AAAECBKEABEABG AAAEDBEHBBACBA AAAEDBEHB[illegible]CBA AAAFCAFBAABDAB AAAFCAFBACBDAB AAAFEAHBAABDAB AAAFEAHBACBDAB
AAAGAHKAHBAABB AAAGCAKEABEABG AAAGCAKEABJABG AAAGCAHE[illegible]BG AAAGCBGEBBDABD AAAGCBGEBBEABG AAAGCDMEABDABG AABAABBAAAACBD
AABAABBAAABBAA AABAABBAAABBBD AABAABBAAABCBA AABAABBAAABCBB AABAABBAAAGBAB AABAABBAAAGBBB AABAABBAABACAD AABAABBAABBBAD
AABAABBAABBCAA AABAABBAABEBAE AABAABBAABEBAG AABAABBAABGBFG AABAABBAABGCAE AABAABBAAEBBBA AABAABBABABCAA AABAABBABABCAB
AABAABBABABCGA AABAABBABABFGB AABAABBABABHAA AABAABBABABHGA AABAABBABABHGB AABAABBABAEFAB AABAABBABAEHAB AABAABBABAGCBB
AABAABBABAGFAB AABAABBABAGHAB AABAABBABBBCQA AABAABBABBBCQB AABAABBABBBFQA AABAABBABBBFQB AABAABBABBGBQG AABAABBABBGCAG
AABAABBABBGCDB AABAABBABBGCDE AABAABBABBGFAG AABAABBABBGFQB AABAABBABEBAGA AABAABBABGBCGA AABAABBABGBFGA AABAABBABGBHGA
AABAABBACABBAA AABAABBACABCBA AABAABBACABCBB AABAABBACABHAB AABAABBACAEBAB AABAABBACAGBAB AABAABBACAGBBB AABAABBACBBCAA
AABAABBACBGCAE AABAABBACBGCAG AABAABBACEBABA AABAABBACEBBBA AABAABBAFABFBB AABAABBAFBBCFB AABAABBAHABFBD AABAABBAHABHBA
AABAABBAHABHBD AABAABBAHEBABA AABAABBBAABBBB AABAABBBADBBAA AABAABBBADBBAB AABAABBBADBBBA AABAABBBAEBBBA AABAABBBBDBAAA
AABAABBBBDBAAB AABAABBBBDBABB AABAABBBBDBCBA AABAABBBBEBCBA AABAABBBBGBCBA AABAABDBABAAAB AABAABDBABAAFB AABAABDBABABAB
AABAABEBBAAABD AABAABEBBAABGD AABAABEBBAACBD AABAABEBBDAABD AABAABEBBDABGD AABAABEBBDACBD AABAABEBBEAABD AABAABEBBEACBD
AABAABEBBGABGD AABAABEFBAAHBD AABAABEFBDAHBD AABAABEFBEAHBD AABAABEFBGAHBD AABAABGAAAAABA AABAABGAAEAABD AABAABGAAEABBD
AABAABGAAAAABA AABAABGAHEAABD AABAABGBABABBD AABAABGBABACBD AABAABGBADAABA AABAABGBAEAABE AABAABGBAEACBG AABAABGBAGABBG
AABAABGBBAACBD AABAABGBBDACBD AABAABGBBEACBD AABAABGBHBAABD AABAABGBHBAABE AABAABGBHEAABA AABAABGBHEAABG AABAABGFABABBG
AABAABIBBBABQB AABAABIBBDAABB AABAABIBBDABGB AABAABIBBEAABG AABAABIBFBAABB AABAABIBFBAABG AABAABIFBBABQB AABAABIFBBAFBB
AABAABIFBBAFBG AABAABIFBDABGB AABAABIFBDAFBB AABAABIFBDAHBB AABAABIFBEAFBG AABAABIFBEAHBG AABAABKBABABBB AABAABKBABABBG
AABAABKBHBAABB AABAABKBHBAABG AABABABAAAAACA AABABABAEFEBQR AABABABAEFGBQR AABABABBAAAAAA AABABABBAAAAAR AABABABBAAAABR
AABABABBAAAACA AABABABBAAAACR AABABABBABAAAE AABABABBABAACE AABABABBABAAFA AABABABBABABCD AABABABBABABCE AABABABBABABFA
AABABABBABBAFA AABABABBABBBBD AABABABBABBBFB AABABABBADAACA AABABABBAFAAAR AABABABBAFAACR AABABABBAFABCR AABABABBAFABFB
AABABABBAFBBFB AABABABBAQABBB AABABABBAQABCB AABABABBAQBBBB AABABABBBBAAQE AABABABBBBBBBB AABABABBBBBBBD AABABABBBBBBQB
AABABABBBDAAGE AABABABBBFAAQR AABABABBBFBBQB AABABABBBGAAGE AABABABBBQAAGR AABABABBBQBAGH AABABABBBQBBBH AABABABBBQBBGH
AABABABBCABABD AABABABBEBABAE AABABABBEFABAR AABABABBHBAABE AABABABBRAAAAR AABABABFABABAD AABABABFABABAE AABABABFABABBD
AABABABFABABBE AABABABFABABCA AABABABFABABFB AABABABFAFABAR AABABABFEBABQB AABABABFEFABAB AABABABQABBBBA AABABABQABBBBD
AABABABBDABBFB AABABABQABEBAD AABABABQABEBBD AABABABQABEBFB AABABABQABGBFB AABABABQAFBBFB AABABABQAQBBBB AABABABQBABHGH
AABABAEBBGAABA AABABAEBBEAABD AABABAEBBBABBD AABABAEBBBABBE AABABAEBBDAABA AABABAEBBDAABD AABABAEBBDADBA AABABAEBBDADBD
AABABAEBBDABBD AABABAEBBGAABA AABABAEBBGAABD AABABAEBBGABBD AABABAEBBGABBE AABABAEBBGABGD AABABAEFBBABBD AABABAEFBBABBE
AABABAEFBBAHBA AABABAEFBBAHBD AABABAEFBDAHBA AABABAEFBDAHBD AABABAEFBGAHBA AABABAFBAFBDIB AABABAFGRAGAAB AABABAFGRBGAAB
AABABAFGRDAAAB AABABAFGRFGAAB AABABAHBAFBBIB AABABAHBAFBDIB AABABAHGBBGAAB AABABAHGBDGAAB AABABAHGBFGAAB AABABAHQBBGHAB
AABABAHQBDBGHB AABABAHQBFGHAB AABABAIBBBAABB AABABAIBBBABBB AABABAIBBBABBE AABABAIBBBABQB AABABAIBBDAABB AABABAIBBDADBB
AABABAIBBDADBB AABABANBBDAAAA AABABANBBDAAAD AABABANBBDAACA AABABANBBDAACD AABABANBBDAACR AABABANBBDAAGD AABABARQBBEHAB
AABABARQBBEFAB AABABARQBDEHAB AABABARQBFEHAB AABABBIBBBAABB AABABBIBBDAABB AABABBIFBDAFBB AABABBIFBDAFBR AABABBIFBDAHBB
AABACADGABEAAB AABACADGABEACB
```

DRAWINGS WITH 23 CROSSINGS:

```
AAAAAAAAAAABBB AAAAAAAAAABAAB AAAAAAAAABGBBG AAAAAAAAABGBFB AAAAAAAAABABAF AAAAAAAAABABBAF AAAAAAAAABABBBF AAAAAAAAABBGBBG
AAAAAAAABGGFBB AAAAAAAABGAFBG AAAAAABAAAAABD AAAAAABAAAABAB AAAAAABAAAABBA AAAAAABAAAABBD AAAAAABAAAABCB AAAAAABAAAABAAD
AAAAAABAAABAAF AAAAAABAAABACA AAAAAABAAABBAA AAAAAABAAABBAD AAAAAABAAABBAF AAAAAABAAABBBF AAAAAABAAABBCA AAAAAABAAABBCD
AAAAAABAAABBCF AAAAAABAAAEACB AAAAAABAAAEBBB AAAAAABAAAGAAB AAAAAABAAAGBAB AAAAAABAAAGBBB AAAAAABAAAGBCB AAAAAABAABAAAD
AAAAAABAABAAAG AAAAAABAABABAA AAAAAABAABABAD AAAAAABAABABAG AAAAAABAABABCA AAAAAABAABABCD AAAAAABAABABCG AAAAAABAABABFA
AAAAAABAABABFD AAAAAABAABABFG AAAAAABAABBBFA AAAAAABAABBBFD AAAAAABAABEAAG AAAAAABAABGBAG AAAAAABAABGBCG AAAAAABAAFAAAB
AAAAAABAAFABAB AAAAAABAAFABCB AAAAAABAAFABFB AAAAAABAAFBBAF AAAAAABAAFBBCF AAAAAABAAFEAAB AAAAAABAAFGBAB AAAAAABAAFGBCB
AAAAAABAAGABBA AAAAAABAAGABBD AAAAAABAAGABBG AAAAAABAAGBBBA AAAAAABAAGBBBD AAAAAABAAQAABD AAAAAABAAQABBB AAAAAABAAQBBBD
AAAAAABAAGEACB AAAAAABABAAFAA AAAAAABABAAFAD AAAAAABABABAGF AAAAAABABABBGF AAAAAABABABFAF AAAAAABABAEBAD AAAAAABABAEBCD
AAAAAABABAEFAB AAAAAABABAEFCD AAAAAABABAEFGB AAAAAABABAEFGD AAAAAABABAGASB AAAAAABABAGBGB AAAAAABABAGFAB AAAAAABABBBAFAG
AAAAAABABBEBAG AAAAAABABBEBCG AAAAAABABBEFAD AAAAAABABBEFCG AAAAAABABBGBAB AAAAAABABBGBCB AAAAAABABBGFCB AAAAAABABEAFAB
AAAAAABABFBBAF AAAAAABABFEBAB AAAAAABABFEBCB AAAAAABABFEFCB AAAAAABABFGBAB AAAAAABABFGBCB AAAAAABABGBFGA AAAAAABABGBFGD
AAAAAABABGEFGD AAAAAABABQAFGB AAAAAABABQAFGD AAAAAABABQBAGF AAAAAABABQBBBF AAAAAABABQBFBF AAAAAABABQBFGB AAAAAABABQBFGD
AAAAAABABQEBCB AAAAAABABQEBGB AAAAAABABQEBGD AAAAAABABQGAGB AAAAAABABQGBBB AAAAAABABQGFBB AAAAAABAFAEFBB AAAAAABAFAEFBD
AAAAAABAFGGABB AAAAAABAFBGAFG AAAAAABAFBGFBG AAAAAABAFGAFBB AAAAAABAFGBFBA AAAAAABAFGBFBD AAAAAABAFGEFBD AAAAAABAFQAFBB
AAAAAABAFQAFBD AAAAAABAFQBFBB AAAAAABAFQBFBD AAAAAABAFQGABB AAAAAABAHAAFBA AAAAAABAHAAFBD AAAAAABAHABFBF AAAAAABAHAEFBB
AAAAAABAHAGFBB AAAAAABAHGAFBA AAAAAABAHGAFBD AAAAAABAHGAFBG AAAAAABAHGBFBA AAAAAABAHGBFBD AAAAAABAHGAFBB AAAAAABAHQBFBD
AAAAAABBAAABAA AAAAAABBAAABBD AAAAAABBAABBCD AAAAAABBAABGAD AAAAAABBAABGBD AAAAAABBAABGBF AAAAAABBAABGCA AAAAAABBAABGCF
AAAAAABBABABAD AAAAAABBABABAG AAAAAABBABAGAB AAAAAABBABAGBA AAAAAABBABAGBB AAAAAABBABBBCD AAAAAABBABBBFA AAAAAABBABBGAD
AAAAAABBABBGBD AAAAAABBABBGCA AAAAAABBABBGFB AAAAAABBAFABAB AAAAAABBAFBBCF AAAAAABBAFBBFB AAAAAABBAFBGAF AAAAAABBAFBGCB
AAAAAABBAGBBBA AAAAAABBAQBBBB AAAAAABBAQBBCB AAAAAABBABBGF AAAAAABBBBBBCD AAAAAABBBFBBAF AAAAAABBBFBBCB AAAAAABBBQBBBB
AAAAAABBBQBBBD AAAAAABFBAAFAA AAAAAABFBAAFAB AAAAAABFBAAFCA AAAAAABFBAAFCB AAAAAABFBAAFGA AAAAAABFBBAFAA AAAAAABFBBAFAB
AAAAAABFBBAFBG AAAAAABFBBAFCA AAAAAABFBBAFCB AAAAAABFBGAFBD AAAAAADAAABAAB AAAAAADAAABBAB AAAAAADAAABBCB AAAAAADAAABAAAB
AAAAAADAABABAB AAAAAADAABABCB AAAAAADAABABFB AAAAAADAABBBFB AAAAAADAABEAAB AAAAAADAAFAACB AAAAAADAAFAAFB AAAAAADAAFBBFB
AAAAAADAAFEACB AAAAAADAAFEAFB AAAAAADAHFAACB AAAAAADAHFAAFB AAAAAADAHFEACB AAAAAADAHFEAFB AAAAAADBAABBCB AAAAAADBAABGAB
AAAAAADBABAAAB AAAAAADBABAACB AAAAAADBABAAFB AAAAAADBABABAB AAAAAADBABBBCB AAAAAADBABBGAB AAAAAADBAFBAIB AAAAAADBAFBBCB
AAAAAADBAFBGAB AAAAAADBAQBACB AAAAAADBHABACB AAAAAADBHBBAFB AAAAAADBHFBAFB AAAAAADGAABGAB AAAAAADGABBGAB AAAAAADGABEAAB
AAAAAADGABEACB AAAAAADGABEAFB AAAAAADGAFBAIB AAAAAADGAFBGAB AAAAAADGAQBACB AAAAAADGHABACB AAAAAADGHBBAFB AAAAAADGHFBAFB
AAAAAAEABAAABD AAAAAAEABAABGA AAAAAAEABAABGD AAAAAAEABAAFBA AAAAAAEABAAFBD AAAAAAEABAEABD AAAAAAEABAEBBD AAAAAAEABBEFBD
AAAAAAEABGAABD AAAAAAEABGAAGD AAAAAAEABGABBD AAAAAAEABGEABD AAAAAAEABGEAGD AAAAAAEABGEBGD AAAAAAEABGEFBD AAAAAAEABQAABD
AAAAAAEABQABBD AAAAAAEABQEABD AAAAAAEAFBAABA AAAAAAEAFBAFBA AAAAAAEAFBAFBD AAAAAAEBBAABBD AAAAAAEBBAABGA AAAAAAEBBBAABA
AAAAAAEBBBAABD AAAAAAEBBBABBD AAAAAAEBBGAABA AAAAAAEBBGAABD AAAAAAEBBGABBD AAAAAAEBBGABBA AAAAAAEFBAABBA AAAAAAEFBBABBA
AAAAAAEFBGABBA AAAAAAKAABAABG AAAAAAKAABABBB AAAAAAKAABEABG AAAAAAKAABEAFB AAAAAAKAHBEABG AAAAABBAAAAABA AAAAABBAAAAABD
AAAAABBAAAABAB AAAAABBAAAABBD AAAAABBAAABABF AAAAABBAAABBAA AAAAABBAAABBBD AAAAABBAAABCBA AAAAABBAAABCBB AAAAABBAAAEABB
AAAAABBAAAGABB AAAAABBAABAAAA AAAAABBAABAAAD AAAAABBAABABAD AAAAABBAABABFD AAAAABBAABACAA AAAAABBAABACAG AAAAABBAABACFD
AAAAABBAABBBAD AAAAABBAABBBFA AAAAABBAABBCAA AAAAABBAABBCFA AAAAABBAABEAAB AAAAABBAABEAFD AAAAABBAABEBAB AAAAABBAABEBBD
AAAAABBAABEBFB AAAAABBAABECBD AAAAABBAABECFB AAAAABBAABGBAG AAAAABBAABGBFB AAAAABBAABGCAB AAAAABBAABGCFB AAAAABBAAEAABA
AAAAABBAAEACBA AAAAABBAAEACBG AAAAABBAAEBABA AAAAABBAAEBBBA AAAAABBAAGABBD AAAAABBAAGACBD AAAAABBAAGBBBA AAAAABBAAGBCBA
AAAAABBABABAAF AAAAABBABABBAF AAAAABBABABBBF AAAAABBABABCAA AAAAABBABABCAB AAAAABBABABCGA AAAAABBABABFAD AAAAABBABABFBF
AAAAABBABABFGB AAAAABBABABFGD AAAAABBABAEAAB AAAAABBABAEAGB AAAAABBABAEAGD AAAAABBABAEBAB AAAAABBABAEBBB AAAAABBABAEBGD
AAAAABBABAECBB AAAAABBABAECBD AAAAABBABAGAAB AAAAABBABAGBAB AAAAABBABAGBBB AAAAABBABAGFBB AAAAABBABBBCAD AAAAABBABBBFAD
AAAAABBABBEBAD AAAAABBABBEBAG AAAAABBABBECAB AAAAABBABBEFAB AAAAABBABBGBAG AAAAABBABBGCAB AAAAABBABBGFAB AAAAABBABEBAGA
AAAAABBABEBFGA AAAAABBABGBBGD AAAAABBABGBCGA AAAAABBABGBFGA AAAAABBABGEBGD AAAAABBAFABEBB AAAAABBAFABFBD AAAAABBAFAEABB
AAAAABBAFBBCFA AAAAABBAFBBCFB AAAAABBAFBECFB AAAAABBAFBGAFG AAAAABBAFBGCBB AAAAABBAFBGCBB AAAAABBAFBGFBB AAAAABBAFBGFBG
AAAAABBAFEACBG AAAAABBAHABABF AAAAABBAHABFBD AAAAABBAHAEABB AAAAABBAHAGABB AAAAABBAHBEAFD AAAAABBAHBEFBD AAAAABBAHEAABA
AAAAABBAHEBABA AAAAABBAHEBFBA AAAAABBAHGBFBA AAAAABBBAABAAD AAAAABBBAABBAD AAAAABBBAABBBD AAAAABBBAABBBF AAAAABBBAABGAF
AAAAABBBABBAFD AAAAABBBABBBAD AAAAABBBABBBBD AAAAABBBABBBFA AAAAABBBABBBFB AAAAABBBABBCBD AAAAABBBABBCFA AAAAABBBABBCFB
AAAAABBBABBGBB AAAAABBBAEBCBA AAAAABBBAEBGBA AAAAABBBAGBABD AAAAABBBAGBBBA AAAAABBBAGBCBA AAAAABBBBBBBAD AAAAABBBBBBBBB
AAAAABBBEBBBBD AAAAABBBBBBCBD AAAAABBBBEBAGA AAAAABBBBGBAGA AAAAABBBBGBBGA AAAAABBBFEBABA AAAAABBBFGBABA AAAAABBHEBABA
AAAAABDAABAAAB AAAAABDAABABAB AAAAABDAABABFB AAAAABDAABACFB AAAAABDAABBBAB AAAAABDAABEAFB AAAAABDAHBEAFB AAAAABDBAABAAB
AAAAABDBAABBAB AAAAABDBABAAAB AAAAABDBABAAFB AAAAABDBABABAB AAAAABDBABBAFB AAAAABDBABBBAB AAAAABEABAACGA AAAAABEABAEABD
AAAAABEABAEBBD AAAAABEABBEFBD AAAAABEABEAAGA AAAAABEABGAAGD AAAAABEABGACGD AAAAABEABBAFBD AAAAABEABGEAGD AAAAABEABGEBBD
AAAAABEABGEBGD AAAAABEABGEFBD AAAAABEAFAACBA AAAAABEAFAAFBA AAAAABEAFEAABA AAAAABEAQAABBA AAAAABEAQEACBA AAAAABEBBAAABD
AAAAABEBBAABGD AAAAABEBBAACBD AAAAABEBBAACGA AAAAABEBBEAABD AAAAABEBBEAAGA AAAAABEBBEABGD AAAAABEBBEACBD AAAAABEBBGAAGA
AAAAABEBBGABBD AAAAABEBBGABGD AAAAABEBFBAABD AAAAABEBFBACBD AAAAABEBFEAABA AAAAABEBQAACBA AAAAABEFBAABBD AAAAABEFBBAFBD
AAAAABEFBEABBD AAAAABEFBEAFGA AAAAABEFBGABBD AAAAABEFQBABBA AAAAABEFQEABBA AAAAABEHBEABBD AAAAABEHBEACGA AAAAABEHBEAFGA
AAAAABEHBGABBD AAAAABEHBGABGD AAAAABEHBGAFGA AAAAABEHQEABBA AAAAABEHQGABBA AAAAABEAABBAAB AAAAABFBAABAAB AAAAABFBAAEBAB
AAAAABFBABBBAB AAAAABGAAAABBA AAAAABGAABEBBG AAAAABGAAEAABA AAAAABGAAEACBA AAAAABGAAEACBD AAAAABGAABABBD AAAAABGAAGEABD
AAAAABGABAEBBD AAAAABGABAEFBD AAAAABGABGEBBD AAAAABGABBEFBD AAAAABGAHAAFBA AAAAABGAHBEFBB AAAAABGAHEAABA AAAAABGAHBEABD
AAAAABGBAAAABA AAAAABGBAAACBA AAAAABGBABABBD AAAAABGBABACBD AAAAABGBABABBA AAAAABGBAEACBB AAAAABBBAEAGBA AAAAABGBAGABBB
AAAAABGBBAACBD AAAAABGBBEACBD AAAAABGHBGABBD AAAAABGHBAAABD AAAAABGBHEAABA AAAAABGBHEAABB AAAAABGFABABBA AAAAABGFAEABBA
AAAAABGFBAABBD AAAAABGFBAAFBD AAAAABGFBBABBD AAAAABGFBBABBG AAAAABGFBBAFBD AAAAABGFBBAFBB AAAAABGFBEABBD AAAAABGFBEAFBD
AAAAABGFBGABBD AAAAABGHABABBB AAAAABGHABACBG AAAAABGHAEABBA AAAAABGHBEABBD AAAAABGHBEACBB AAAAABGHBEAFBD AAAAABKAABABFB
AAAAABKAABACFB AAAAABKAABEAFB AAAAABKAHBEAFB AAAAABKBABAAFB AAAAABKBABABBB AAAAABKBABABBB AAAAABKBABACBB AAAAABKBABACBB
AAAAABKBHBAABB AAAAABKBHBAABG AAAAADEABAACBA AAAAADEABAAFBA AAAAADEABBAFBD AAAAADEABBEABD AAAAADEAFEAABA AAAAADEHFBAFBD
AAAAADEAFBAIBA AAAAADEAFBEABD AAAAADEAFEACBA AAAAADEAFEAFBA AAAAADEAQBACBA AAAAADEHBBAFBA AAAAADEHBEACBA AAAAADEHBEAFBA
AAAAADEHFBAFBA AAAAADGAABAABA AAAAADGAABAABD AAAAADGAABACBA AAAAADGAABACBB AAAAADGABEACBB AAAAADGAHBEABD AAAAADGAHBEAAB
AAAAADGAHEAFBB AAAAADKAABAABB AAAAADKAABAABB AAAAADKAHBEABB AAAAADKAHBEABB AAAAAFKAABAABB AAAAAFKAHBAABB AAAAAFKAHBAGFB
AAAAAFKAHBEABB AAAABABAAAAABA AAAABABAAAABCA AAAABABAABAAAA AAAABABAABAAFA AAAABABAABABFA AAAABABAABADAA AAAABABAABADCA
AAAABABAABBBFA AAAABABAABBDFA AAAABABAAFEBFB AAAABABAAFGBFB AAAABABAAGABBA AAAABABAAGBBBA AAAABABAAGBDBA AAAABABAAQBBBF
AAAABABAAQBBBA AAAABABAAQBDBA AAAABABAAQEBBB AAAABABAAQEBCB AAAABABAAQBBBB AAAABABABBBBGF AAAABABABBGFGB AAAABABAEAABAA
AAAABABAEAADGA AAAABABAEBADAA AAAABABAEFEBAB AAAABABAHBAAFA AAAABABAHBEAFBA AAAABABAHBGFBA AAAABABAHQBFBF AAAABABAHQBFBA
AAAABABAHQEFBB AAAABABAHQEFCB AAAABABAHQGFBB AAAABABABAAACA AAAABABABAAAAA AAAABABABAADCA AAAABABABBAAAA AAAABABABBADAA
AAAABABABBADCA AAAABABABBAFAA AAAABABABFEFAB AAAABABABQBFGF AAAABABABQBFGB AAAABABABQEFGB AAAABABABQGFGB AAAABABABBAAFA
AAAABABBABAGFA AAAABABBABADCA AAAABABBABAGBB AAAABABBABAGCB AAAABABBABABFB AAAABABBABAFAFB AAAABABBAFAFB AAAABABBAFAGAB
AAAABABBAFBGFB AAAABABBAGAABA AAAABABBAGABBA AAAABABBAGBBBA AAAABABBAGABBA AAAABABBAGAABB AAAABABBAQABCB AAAABABBAQAQBB
AAAABABBQAADCB AAAABABBAQBBBF AAAABABBAQBBAI AAAABABBAQBDAI AAAABABBAQABBB AAAABABBAQBGBF AAAABABBAQBGBD AAAABABBAQBBBF
AAAABABBQBBBF AAAABABBEBADAA AAAABABBEBAGAB AAAABABBHFAAFB AAAABABBEBGABA AAAABABBHBAABB AAAABABBHQAACB AAAABABBHQBAGH
AAAABABBRGAAAA AAAABABBRGBACA AAAABABBRFADCB AAAABABBRGAAGB AAAABABBRQADCB AAAABABBRQADGB AAAABABBRQBAGH AAAABABBRQBDGH
```

AAAABABFAFABAB	AAAABABFBFABCB	AAAABABFBFAFAB	AAAABABFBFAFCB	AAAABABFBGAFGA	AAAABABFBQABBB	AAAABABFBQABCB	AAAABABFBQABGB
AAAABABFBQAFBB	AAAABABFBQAFCB	AAAABABBAFBGCH	AAAABABGAFBGFB	AAAABABBAQBABH	AAAABABGAQBDCH	AAAABABGAQBGBB	AAAABABGAQEDBB
AAAABABBAQEDCB	AAAABABBAQGDBB	AAAABABGBQBAGF	AAAABABGBQBAGH	AAAABABGBQBDBF	AAAABABGBQBDBH	AAAABABGBQBDGF	AAAABABGBQBDGH
[illegible]	AAAABABGBQBDGB	[illegible]	AAAABABGHQBACH	AAAABABGRFBDCH	AAAABABGRQBAGF	AAAABABGRQBDCH	AAAABABGRQBDGF
[illegible]	AAAABABGRQEDGB	[illegible]	AAAABABQAQBBBB	AAAABABQAQBDCF	AAAABABQBQBBBH	AAAABABQBQBBCF	AAAABABQRQBBGH
[illegible]	AAAABABQRQBDGH	AAAABABQBQBDGF	AAAABABQBQBDGH	AAAABABQBQBFBH	AAAABABQBQBFCF	AAAABABQBQBFGB	AAAABABQRQBDCH
[illegible]	AAAABABQRQBFCF	AAAABABQRQBFCH	AAAABADAAAAACB	AAAABADAAAABAB	AAAABADAAAABCB	AAAABADAAAADAB	AAAABADAAAADCB
AAAABADAAAEACB	AAAABADAAAEDAB	AAAABADAAAEDCB	AAAABADAAFAAIB	AAAABADAAFABFB	AAAABADAAFADFB	AAAABADAAFEAIB	AAAABADAAFEBAB
AAAABADAAFEBCB	AAAABADAAFEDFB	AAAABADAADAACB	AAAABADAAQEACB	AAAABADAEAABAB	AAAABADAEAADAB	AAAABADAEAEDAB	AAAABADAEFABAB
AAAABADAEFADAB	AAAABADAEFEDAB	AAAABADAHQAACH	AAAABADAHAEACB	AAAABADAHFAAIB	AAAABADAHFEAIB	AAAABADAHQAACB	AAAABADAHQEACB
AAAABADARAAAAB	AAAABADARAEAAB	AAAABADARQADCB	AAAABADARQEDCB	AAAABADBAAAAAB	AAAABADBAAAACB	AAAABADBAAABCB	AAAABADBAAADCB
AAAABADBABAAFB	AAAABADBABABCB	AAAABADBABADCB	AAAABADBABBDFB	AAAABADBABBGCB	AAAABADBAFAAFB	AAAABADBAFABCB	AAAABADBAFADCB
AAAABADBAFBBFB	AAAABADBAFBBIB	AAAABADBAQBBCB	AAAABADBEBABAB	AAAABADBEBADAB	AAAABADBEFABAB	AAAABADBEFADAB	AAAABADBHBAACB
[illegible]	AAAABADBHFAACB	AAAABADBRAAAAB	AAAABADBRAAACB	AAAABADBRAADCB	AAAABADBRBAAAB	AAAABADBRBAACB	AAAABADBRBADAB
AAAABADBRFAAAB	AAAABADBRFAACB	AAAABADBRFADAB	AAAABADGAAEAAB	AAAABADGAAEACB	AAAABADGAAEDCB	AAAABADGABBDFB	AAAABADGABBGCB
AAAABADGABEAFB	AAAABADGABEDCB	AAAABADGAFEAFB	AAAABADGAFEDCB	AAAABADGEBEDAB	AAAABADGEFEDAB	AAAABADGHBBAFB	AAAABADGHBEACB
AAAABADGHFEACB	AAAABADGRAEAAB	AAAABADGRAEACB	AAAABADGRAEDCB	AAAABADGRBEAAB	AAAABADGRBEACB	AAAABADGRBEDAB	AAAABADGRFEAAB
AAAABADGRFEACB	AAAABADGQFEDAB	AAAABADQAAEDAB	AAAABADQABBBCB	AAAABADQABEDAB	AAAABADQAFBDFB	AAAABADQAFBDIB	AAAABADQAFEDAB
[illegible]	AAAABADQEAEDAB	AAAABADQRAEDCB	AAAABADQRBEDAB	AAAABADQRFEDAB	AAAABAEABAAABA	AAAABAEABAAAGD	AAAABAEABAADBA
AAAABAEABAADGA	AAAABAEABAAFBD	AAAABAEABAEAGD	AAAABAEABAEFBD	AAAABAEABBADBA	AAAABAEABBEFBD	AAAABAEABGAAGA	AAAABAEABGADBA
AAAABAEABGADGA	AAAABAEABGAFGD	AAAABAEABGEFBD	AAAABAEABQAAGD	AAAABAEABQADBD	AAAABAEABQADGD	AAAABAEABQAFBD	AAAABAEABQEAGD
AAAABAEABQEBBD	AAAABAEABQEBGD	AAAABAEABQEDBD	AAAABAEABQEDGD	AAAABAEBBAAABA	AAAABAEBBAAABD	AAAABAEBBAADBA	AAAABAEBBAADBD
AAAABAEBBAADGA	AAAABAEBBAADBD	AAAABAEBBBAABA	AAAABAEBBBAABD	AAAABAEBBABBBD	AAAABAEBBGAABA	AAAABAEBBGAABD	AAAABAEBBGABBD
AAAABAEBBGABGD	AAAABAEFBAAFBA	AAAABAEFBBABBA	AAAABAEFBGABBA	AAAABAEFBGABGA	AAAABAFAAABDCF	AAAABAFAAAGACB	AAAABAFAAAGDCB
AAAABAFAABGBCB	AAAABAFAAFBBCF	AAAABAFAAQBBCB	AAAABAFAAQGACB	AAAABAFAHAGACB	AAAABAFAHQGACB	AAAABAFARQBDCF	AAAABAFARQGDCB
AAAABAFBAABGCF	AAAABAFBAAFAAF	AAAABAFBABBGCB	AAAABAFBAFBBIB	AAAABAFBAFBDIF	AAAABAFBAFBGCF	AAAABAFBAQBBCB	AAAABAFBAQBDCF
AAAABAFBRQBDCF	AAAABAFGAABGCB	AAAABAFGAAGACB	AAAABAFGAAGDCB	AAAABAFGABGDCB	AAAABAFGAFBDIB	AAAABAFGAFBGCB	AAAABAFGAQBDCB
AAAABAFGHBGACB	AAAABAFGHFBAIB	AAAABAFGHFGACB	AAAABAFGHQBACB	AAAABAFGRABDCF	AAAABAFGRAGACB	AAAABAFGRAGDCB	AAAABAFGRBBDCB
AAAABAFGRBGACB	AAAABAFGRFGACB	AAAABAFGRQBDCB	AAAABAFQAABBCB	AAAABAFQABBDCB	AAAABAFQAFBBCB	AAAABAFQRABDCF	AAAABAFQRAGDCB
AAAABAFQRBBDCB	AAAABAFQRQBDCB	AAAABAHAAABBCF	AAAABAHAAABBCH	AAAABAHAAABDCF	AAAABAHAAAGACB	AAAABAHAAAGBCB	AAAABAHAAAGDCB
AAAABAHAAFBDCH	AAAABAHAAQGACB	AAAABAHABABFCF	AAAABAHABAGFCB	AAAABAHABBGFCB	AAAABAHABFBFCF	AAAABAHABFGFCB	AAAABAHABQBBCF
AAAABAHABQBDCF	AAAABAHABQBDCH	AAAABAHABQBFCH	AAAABAHABQGBCB	AAAABAHABQGDCB	AAAABAHBAAFAAF	AAAABAHBAAFDAH	AAAABAHBAAFGAF
AAAABAHBAAFGAH	AAAABAHBAFBAIB	AAAABAHBAFBBIF	AAAABAHBAFBBIH	AAAABAHBAFBDIF	AAAABAHBAFBGIB	AAAABAHBAQBACH	AAAABAHBAQBBCF
AAAABAHBAQBBCH	AAAABAHBAQBDCF	AAAABAHBAQBGCB	AAAABAHBBAFAAF	AAAABAHBBAFAAH	AAAABAHBBQBBCF	AAAABAHBBQBDCF	AAAABAHBBQBDCH
AAAABAHBGAFGAH	AAAABAHGAAGACB	AAAABAHGAAGDCB	AAAABAHGABBGCB	AAAABAHGABGDCB	AAAABAHGAFBACH	AAAABAHGAFBDIB	AAAABAHGAQBDCB
AAAABAHGBABACH	AAAABAHGBABDCF	AAAABAHGBABDCH	AAAABAHGBAGACB	AAAABAHGBAGDCB	AAAABAHGBBBDCB	AAAABAHGBBGACB	AAAABAHGBFBACH
AAAABAHGBFGACB	AAAABAHGBQBDCB	AAAABAHQABBBCB	AAAABAHQABBDCB	AAAABAHQBABBCF	AAAABAHQBABDCF	AAAABAHQBABDCH	AAAABAHQBABFCH
AAAABAHQBAGBCB	AAAABAHQBAGDCB	AAAABAHQBBBBCB	AAAABAHQBBBDCB	AAAABAHQBFBBCH	AAAABAHQBFBFCB	AAAABAHQBQBBCB	AAAABAHQBQBDCB
AAAABANABAAACA	AAAABANABAAAGA	AAAABANABAADCA	AAAABANABAEFCD	AAAABANABAEFGD	AAAABANABGAAGA	AAAABANABGAFGD	AAAABANABQADCD
AAAABANABQAFCD	AAAABANABQAFGD	AAAABANABQEBCD	AAAABANABQEDCD	AAAABANBBAAACA	AAAABANBBAAACD	AAAABANBBAAAGA	AAAABANBBAAAGD
AAAABANBBAADCA	AAAABANBBAADCD	AAAABANBBGABGD	AAAABANBBQABGD	AAAABANFBAAFCA	AAAABANFBGABGA	AAAABANQBAEBCD	AAAABANQBAEDCD
AAAABBBAAAAAAB	AAAABBBAAAAABD	AAAABBBAAABAAD	AAAABBBAAABAAF	AAAABBBAAABABF	AAAABBBAAABBAD	AAAABBBAAABBAF	AAAABBBAAABBAH
AAAABBBAAABBBD	AAAABBBAABAAAA	AAAABBBAABAAAD	AAAABBBAABAAFA	AAAABBBAABAAFD	AAAABBBAABABAD	AAAABBBAABABFA	AAAABBBAABBAFA
AAAABBBAABBAFD	AAAABBBAABBBGD	AAAABBBAABEBBD	AAAABBBAABBBFB	AAAABBBAABCABD	AAAABBBAABCAFB	AAAABBBAABCBAA	AAAABBBAABCBBB
[illegible]	AAAABBBAABEBBB	AAAABBBAABGBBB	AAAABBBABABAGF	AAAABBBABABAGH	AAAABBBABABBBF	AAAABBBABABBGF	AAAABBBABABFAF
[illegible]	AAAABBBABACAAH	AAAABBBABACAAH	AAAABBBABBBFAD	AAAABBBABBEBAD	AAAABBBABBEFAB	AAAABBBABBGBBB	AAAABBBABBGFAB
[illegible]	AAAABBBABEBAGA	AAAABBBABEBFGA	AAAABBBABECFGA	AAAABBBABGBBBD	AAAABBBABGBBGD	AAAABBBABGBFBD	AAAABBBABGBFGA
[illegible]	AAAABBBABGCFGA	AAAABBBAEABBAD	AAAABBBAEABBAF	AAAABBBAEABBAH	AAAABBBAEABBGD	AAAABBBAEACBAA	AAAABBBAEACBGA
[illegible]	AAAABBBAEABBAB	AAAABBBAEBABAA	AAAABBBAEBBBAA	AAAABBBAEBCBAB	AAAABBBAEBGBAB	AAAABBBAHBAAFA	AAAABBBAHBAAFD
[illegible]	AAAABBBAHBBAFD	AAAABBBAHBBFBD	AAAABBBAHBCABD	AAAABBBAHBCAFB	AAAABBBAHBCFBB	AAAABBBAHBEFBB	AAAABBBAHBGFBB
[illegible]	AAAABBBBABAAFA	AAAABBBBABABAD	AAAABBBBABABBD	AAAABBBBABABFA	AAAABBBBABABFB	AAAABBBBABAGAB	AAAABBBBABBBAD
[illegible]	AAAABBBBAEAABA	AAAABBBBAEBABA	AAAABBBBAEBBBA	AAAABBBBAGAABA	AAAABBBBAGABBA	AAAABBBBAGBABA	AAAABBBBBBBBAD
[illegible]	AAAABBBBBECABA	AAAABBBBBGAAGD	AAAABBBBBGABGD	AAAABBBBBGABGD	AAAABBBBBGBABD	AAAABBBBBGBAGA	AAAABBBBBGBBBA
[illegible]	AAAABBBBBGCABA	AAAABBBBBGCBBA	AAAABBBBBGCBGA	AAAABBBBEBBGAA	AAAABBBBEBBGAB	AAAABBBBEBCBAA	AAAABBBBEBCBAB
[illegible]	AAAABBDAAAABAB	AAAABBDAAAABBAB	AAAABBDAEABBAB	AAAABBDBAABGAB	AAAABBDBAACBAB	AAAABBDBABAAAB	AAAABBDBABABAB
[illegible]	AAAABBDBEABGAB	AAAABBDBEACBAB	AAAABBEABAAAGA	AAAABBEABAAAGD	AAAABBEABAAFDA	AAAABBEABAAFGD	AAAABBEABBAADA
[illegible]	AAAABBEABEAAGA	[illegible]	AAAABBEABBAFGD	AAAABBEBBAAABD	AAAABBEBBAAADA	AAAABBEBBAAAGA	AAAABBEBBBAABD
[illegible]	AAAABBEBBBABBD	[illegible]	AAAABBEBBEAAGA	AAAABBEBBGAABD	AAAABBEBBGAAGA	AAAABBEBBGABBD	AAAABBEBBGABGD
[illegible]	AAAABBEFBBAFBA	[illegible]	AAAABBEFBBAFBA	AAAABBEHBGABGD	AAAABBFAABBAAB	AAAABBFBAABAAF	AAAABBFBAABBAF
[illegible]	AAAABBFBAAFBAF	[illegible]	AAAABBFBEABBAF	AAAABBFBEABGAB	AAAABBFBEAFBAF	AAAABBHAABBAAB	AAAABBHAABBBAB
[illegible]	[illegible]	[illegible]	AAAABBHABBCBAB	AAAABBHBABCBAB	AAAABBHBBABAAF	AAAABBHBBABAAH	AAAABBHBBACAAB
[illegible]	[illegible]	[illegible]	AAAABBNABAAAGA	AAAABBNABAAAGD	AAAABBNABACBGH	AAAABBNABGAAGA	AAAABBNABGAAGD
[illegible]	[illegible]	[illegible]	AAAABBNBBACAAA	AAAABBNBBACBGH	AAAABBNFBGABGA	AAAABDEABAAFBA	AAAABDEABBAABA
[illegible]	[illegible]	[illegible]	AAAABEFBAABBAB	AAAABEFBEABBAF	AAAABEFBEABGAB	AAAABEFBRABDAF	AAAABEFGAABDAB
[illegible]	[illegible]	[illegible]	AAAABEHBAABGAB	AAAABEHBBABDAF	AAAABEHBBABDAH	AAAABEHBBACDAB	AAAABEHBGABGAB
[illegible]	[illegible]	[illegible]	AAAABEHQAABBAB	AAAABEHQBABDAB	AAAABEHQBACBAB	AAAABENBBACBAD	AAAABENBBACDAA
[illegible]	[illegible]	[illegible]	AAAABGHABABBAB	AAAABGHABABDAB	AAAABGHABACBAB	AAAABGHBBABDAF	AAAABGHBBABDAH
[illegible]	[illegible]	[illegible]	AAAABGNABACBGD	AAAABGNBBAADAA	AAAABGNBBAADGA	AAAABGNBBACBAD	AAAABGNBBACDAA
[illegible]	[illegible]	[illegible]	AAAABKNABACHCH	AAAABKNABACBGD	AAAABKNHBEABGA	AAAACADAAABACB	AAAACADAAABDCB
[illegible]	[illegible]	[illegible]	AAAACADBAABDCB	AAAACADBABBAFB	AAAACADBABBDCB	AAAACADBAFBAFB	AAAACADBAFBDCB
[illegible]	[illegible]	[illegible]	AAAACADGABEACB	AAAACADGABEAFB	AAAACADGABEDAB	AAAACADGAFBAIB	AAAACADGAFBDFB
[illegible]	[illegible]	[illegible]	AAAACAFAABBDAB	AAAACAFAABBDCB	AAAACAFAABGACB	AAAACAFAAFBDAB	AAAACAFAAFBDCB
[illegible]	[illegible]	[illegible]	AAAACAFBAABDCB	AAAACAFBABBDCB	AAAACAFBAFBDAF	AAAACAFBAFBDCB	AAAACAFGAABACB
[illegible]	[illegible]	[illegible]	AAAACAFGAFBDCB	AAAACAKAABEABB	AAAADANABACBGH	AAAADANBBACBGH	AAAADANFBACBGH
[illegible]	[illegible]	[illegible]	AAAADBEBBAACDA	AAAADBEBBAACGA	AAAADBEBBBACDA	AAAADBEBBEACGA	AAAADBEBBGACGA
[illegible]	[illegible]	[illegible]	AAAADENFBACBAA	AAAADGNBBAABAA	AAAADGNBBAABGA	AAAADGNBBACBAA	AAAADGNHBAABGA
[illegible]	[illegible]	[illegible]	AAAAFDGABEAFBA	AAAAFDGHBBACBA	AAAAFDGHBBAFBA	AAAAFDGHBEAFBA	AAABAABAAAEBAB
[illegible]	[illegible]	[illegible]	AAABAABAABABHG	AAABAABAABBBFD	AAABAABAABBBHD	AAABAABAABDBAD	AAABAABAABDBCB
[illegible]	[illegible]	[illegible]	AAABAABAABEBAD	AAABAABAABEBAG	AAABAABAABEBCD	AAABAABAABEBCG	AAABAABAABGBFG
[illegible]	[illegible]	[illegible]	AAABAABAACBBHD	AAABAABAACDBBD	AAABAABAAGBBBD	AAABAABAAGDBBD	AAABAABABAAAAA
[illegible]	[illegible]	[illegible]	AAABAABABADAAA	AAABAABABADBAA	AAABAABABADBAD	AAABAABABADBCA	AAABAABABADBCD

```
AAABAABABADFAB AAABAABABADFCA AAABAABABADFCD AAABAABABAEBAD AAABAABABAEBCD AAABAABABAEFAB AAABAABABAEFCD AAABAABABAGFAB
AAABAABABBAFAG AAABAABABBBBAD AAABAABABBBBCD AAABAABABBDBAG AAABAABABBDBCG AAABAABABBDFAA AAABAABABBDFAD AAABAABABBDFCB
AAABAABABBEBAG AAABAABABBEBCG AAABAABABBEFAD AAABAABABBEFCB AAABAABABBGBAG AAABAABABBGBCG AAABAABABBGFCB AAABAABABCABAA
AAABAABABCABAD AAABAABABCAFAB AAABAABABCBBAF AAABAABABCBBBF AAABAABABCBBBD AAABAABABCBFAA AAABAABABCBFAD AAABAABABCBFBF
AAABAABABCBFBD AAABAABABCDBAB AAABAABABCDBCB AAABAABABCEBAB AAABAABABCEBCB AAABAABABCEFCB AAABAABABCGBAB AAABAABABCGBBB
AAABAABABCGBCB AAABAABABCGFBB AAABAABABCGFCB AAABAABABFAFAB AAABAABABFBBAF AAABAABABFDBAB AAABAABABFEBAB AAABAABABFEBCB
AAABAABABFEFCB AAABAABABFGBAB AAABAABABFGBCB AAABAABABGBBBD AAABAABABGBEBQ AAABAABABGDBBQ AAABAABABGDEBQ AAABAABABQBBBF
AAABAABABQBFBF AAABAABABQEBCB AAABAABABQGBBB AAABAABABQGFBB AAABAABAFABFBF AAABAABAFAGEBB AAABAABAFCBFBF AAABAABAFCBFBD
AAABAABAFCGFBB AAABAABAFQDFBB AAABAABAFQDFBD AAABAABAFQEFBB AAABAABAFQEFBD AAABAABAHAEFBD AAABAABAHCDFBD AAABAABAHGBFBQ
AAABAABAHGDFBD AAABAABAHQAFBD AAABAABAHQBFBF AAABAABAHQDFBD AAABAABAHQEFBB AAABAABAHQGFBB AAABAABBBAABCD AAABAABBBADBAD
AAABAABBBADBCA AAABAABBBADBCB AAABAABBBBABAG AAABAABBBBABCD AAABAABBBBBBAQ AAABAABBBBBBBQ AAABAABBBBBBCD AAABAABBBBDBAD
AAABAABBBBDBBG AAABAABBBBDBBD AAABAABBBBDBCB AAABAABBBCABAD AAABAABBBCBBAQ AAABAABBBCBBBF AAABAABBBCBBBQ AAABAABBBCBBCD
AAABAABBBCBBCF AAABAABBBFBBAF AAABAABBHQAACB AAABAABEFBDAHQ AAABAABGFBBAFQ AAABAADAAAAACB AAABAADAACBBHB AAABAADAAFAAIB
AAABAADAAFABAB AAABAADAAFABCB AAABAADAAFABFB AAABAADAAFBBIB AAABAADAAFDBAB AAABAADAAFDBCB AAABAADAAFDBFB AAABAADAAFEAIB
AAABAADAAQAACB AAABAADAAQBBCB AAABAADAAQEACB AAABAADAHAAACB AAABAADAHAEACB AAABAADAHQAACB AAABAADAHQEACB AAABAADBAAAAAB
AAABAADBAAAACB AAABAADBAAABAB AAABAADBAADBAB AAABAADBAADGAB AAABAADBABABCB AAABAADBABABFB AAABAADBABABHB AAABAADBABAGAB
AAABAADBABBBFB AAABAADBABBGFB AAABAADBABDBCB AAABAADBABDBFB AAABAADBABDBHB AAABAADBABDGCB AAABAADBABDGFB AAABAADBABDGHB
AAABAADBACABAB AAABAADBAFABAB AAABAADBAFBBFB AAABAADBAFBGFB AAABAADBAFDAIB AAABAADBAFDBAB AAABAADBAFDGAB AAABAADBAQDACB
AAABAADBHADACB AAABAADEAADAAB AAABAADEAAEACB AAABAADEACBGHB AAABAADEAFBGIB AAABAADEAFDGAB AAABAADEAFDGCB AAABAADEAFDGFB
AAABAADEAFEAIB AAABAADEAQBGCB AAABAADEAQEACB AAABAADGAAEAAB AAABAADGAAEACB AAABAAEABAAFBD AAABAAEABAEBBD AAABAAEABBEFBD
AAABAAEABCABBD AAABAAEABQABBD AAABAAEAFAAABD AAABAAEAFAEABD AAABAAEAFBEFBD AAABAAEAFQAABD AAABAAEAFQEABD AAABAAEBBAAABA
AAABAAEBBAAABD AAABAAEBBAABBD AAABAAEBBBABBD AAABAAEBBCABBA AAABAAEBBGABBD AAABAAEBFGAABA AAABAAFAAABBAB AAABAAFAAABBCB
AAABAAFAABBBHB AAABAAFAABGBAB AAABAAFAACBBAB AAABAAFAACBBCB AAABAAFAAFBBHB AAABAAFAHABACB AAABAAFAHQGACB AAABAAFBACBBCF
AAABAAFBAFBBCF AAABAAFBAFBGCF AAABAAFEAABGAB AAABAAFEAABGCB AAABAAFEAAGACB AAABAAFEABBGHB AAABAAFEAFBGHB AAABAAFEAQGACB
AAABAAFGAABGCB AAABAAFGAAGAAB AAABAAFGAAGACB AAABAAFGABBGCB AAABAAFGAFBGCB AAABAAFGHQBACB AAABAAGAAAAABD AAABAAGAAAEABD
AAABAAGAABEBBD AAABAAGAABEBBG AAABAAGAAGABBA AAABAAGAAGABBD AAABAAGAAGABBG AAABAAGAAQAABD AAABAAGAAQEABD AAABAAGABADBBD
AAABAAGABADFBD AAABAAGABBEBBG AAABAAGABBEFBG AAABAAGABGDBBG AAABAAGABGDFBG AAABAAGAHAAABD AAABAAGAHADABA AAABAAGAHAEABD
AAABAAGAHBEFBD AAABAAGAHBEFBG AAABAAGAHGAFBA AAABAAGAHGAFBD AAABAAGAHGAFBG AAABAAGAHGDFBA AAABAAGAHGDFBD AAABAAGAHGDFBG
AAABAAGAHQAABD AAABAAGAHQEABD AAABAAGBAAAABA AAABAAGBAAAABD AAABAAGBAAABBA AAABAAGBABABBD AAABAAGBABABBG AAABAAGBABAGBA
AAABAAGBAGABBA AAABAAGBBAABBD AAABAAGBBBABBD AAABAAGBBGAABD AAABAAGBBGABBD AAABAAGBBGDABD AAABAAGBBQAABD AAABAAGBBQDABD
AAABAAGBHADABA AAABAAGBHADABD AAABAAGBHGAABA AAABAAGBHGAABD AAABAAGBHGAABG AAABAAKAABABBG AAABAAKAABABFB AAABAAKAABEBBB
AAABAAKAAFAAFB AAABAAKAAFEAFB AAABAAKBABABBB AAABAAKBABDBBB AAABAAKBHBAABG AAABAAKEAFEAFB AAABAAKEAGEABG AAABAAKEHGDABG
AAABAAMAABBAFQ AAABAAMAABBAHQ AAABAAMAABBBFB AAABAAMAABBBHB AAABAAMAABGBBB AAABAAMAHBBAFQ AAABAAMAHBBAHQ AAABAAMAHBDAFQ
AAABAAMAHBDAHG AAABAAMAHBGAHG AAABAAMBABBBBQ AAABAAMBABBBFB AAABAAMBABBGBQ AAABAAMBABBGFB AAABAAMBABDBBG AAABAAMBABDBBQ
AAABAAMBABDBFB AAABAAMBABDBHB AAABAAMBABFAHQ AAABAAMBAFBBHB AAABAAMBAFBGHB AAABAAMBHBBAFQ AAABAAMBHBDAFQ AAABAAMBHBDAHG
AAABAAMBHBDAHQ AAABAAMBHGBABQ AAABAAMBHGDABG AAABAAMBHGDABQ AAABAAMEABBAHQ AAABAAMEABBGFB AAABAAMEABBGHB AAABAAMEAGBABQ
AAABAAMEHGDABG AAABAAMGABBGBB AAABAAMGHBBABQ AAABABBAAABBAD AAABABBAAABBAF AAABABBAAABBBF AAABABBAAABCAA AAABABBAAABCBF
AAABABBAAADBAA AAABABBAAADCAB AAABABBAAAEAAB AAABABBAAAEBBB AAABABBAAAGBAB AAABABBAAAGBBB AAABABBAAAGCBB AAABABBAABDBHD
AAABABBAABDBHQ AAABABBAABEBAG AAABABBAABGBFG AAABABBAABGBHG AAABABBAACBBAD AAABABBAACBBAF AAABABBAACBBAQ AAABABBAACBBBF
AAABABBAACBBHA AAABABBAACDBAA AAABABBAACDBHB AAABABBAACEBBB AAABABBAACGBAB AAABABBAACGBBB AAABABBAAEBBBQ AAABABBABABBAF
AAABABBABABBBF AAABABBABABCAD AAABABBABABCBF AAABABBABABFAD AAABABBABABFBF AAABABBABADBAD AAABABBABADCAA AAABABBABADCAB
AAABABBABADFAA AAABABBABADFAB AAABABBABAEAAD AAABABBABAEBAB AAABABBABAGBAB AAABABBABAGBBB AAABABBABAGCBB AAABABBABAGFBB
AAABABBABBBBAQ AAABABBABBBCAQ AAABABBABBBFAD AAABABBABBDBAQ AAABABBABBDCAQ AAABABBABBDFAD AAABABBABBDFAG AAABABBABBEBAD
AAABABBABBEBAG AAABABBABBEFAB AAABABBABBGBAG AAABABBABBGCAG AAABABBABBGFAB AAABABBABCBBAD AAABABBABCBBBQ AAABABBABCBFAA
AAABABBABCBFAB AAABABBABCBFBQ AAABABBABCDBAA AAABABBABCDBAB AAABABBABCDBBQ AAABABBABCDFBQ AAABABBABEBBBQ AAABABBABEBFBQ
AAABABBAFAEFBB AAABABBAFAEFBD AAABABBAFCBFBQ AAABABBAFCDFBQ AAABABBAFEDFBQ AAABABBAHABFBF AAABABBAHAEFBB AAABABBAHAGFBB
AAABABBAHEBFBQ AAABABBBAADBAD AAABABBBAADGAD AAABABBBABDBFD AAABABBBABDBHD AAABABBBABDGFD AAABABBBABDGHD AAABABBBACDBAD
AAABABBBACDBHQ AAABABBBACDGAD AAABABBBACDGHQ AAABABBBBABBBF AAABABBBBABCAD AAABABBBBABCAF AAABABBBBABCBF AAABABBBBADGAD
AAABABBBBADCAB AAABABBBBBBBAD AAABABBBBBBBAQ AAABABBBBBBBBB AAABABBBBBBBBD AAABABBBBBBBBQ AAABABBBBBBCAD AAABABBBBBBCAQ
AAABABBBBBDBAD AAABABBBBBDBAG AAABABBBBBDBAQ AAABABBBBBDCAD AAABABBBBBDCAG AAABABBBBBDCAQ AAABABBBBCBBAD AAABABBBBCBBAF
AAABABBBBCBBAQ AAABABBBBCBBBB AAABABBBBCBBBD AAABABBBBCBBBQ AAABABBBBCDBAB AAABABBBBCDBBB AAABABBBBCDBBD AAABABBBBCDBBQ
AAABABBBBEBBBQ AAABABBBFABCBB AAABABBBFABCBD AAABABBBFADCBB AAABABBBFEDCBG AAABABBEAADGAA AAABABBEAAEAAB AAABABBEABDGHD
AAABABBEABDGHQ AAABABBEACBGHA AAABABBEACDGAA AAABABBEACDGHB AAABABBEBABAAF AAABABBEBADAAA AAABABBEBAEAAB AAABABBEBAGAAB
AAABABBEFAGABB AAABABBEFEBABQ AAABABBEHAEABD AAABABBEHBEAHD AAABABBRAADGAB AAABABBRABDGAD AAABABBRABDGHG AAABABBRABDGHQ
AAABABBRBADAAA AAABABBRBADAAB AAABABDAAABBAB AAABABDAACBBAB AAABABDBAADBAB AAABABDBAADGAB AAABABDBABBCFB AAABABDBABDBFB
AAABABDBABDBHB AAABABDBABDGFB AAABABDBABDGHB AAABABDBACDBAB AAABABDBACDBAB AAABABDEAABBAB AAABABDEABBGFB AAABABDEABBGHB
AAABABDEABDGHB AAABABDEACBGAB AAABABDEHBEAHB AAABABDRABDGAB AAABABFAAABAAB AAABABFAABBBAB AAABABFAABBCAB AAABABFAABBCHB
AAABABFBAAFAAF AAABABFBAAFBAF AAABABFBACBBAF AAABABFBACFBAF AAABABFEAABAAB AAABABFEABBGAB AAABABFEHBGAHB AAABABGAAAEBBD
AAABABGAHBEFBD AAABABGEAAEABD AAABABKAABEBBB AAABABKAABEBBG AAABABKBABDBFB AAABABKBABDGFB AAABABMAABBBFB AAABABMAABBBHB
AAABABMAABDBHB AAABABMEABBGFB AAABABMEABBGHB AAABABMEABDGHB AAABABMEHBBAFQ AAABABMEHBBAHQ AAABABMEHBGAHQ AAABABMEHBGAHG
AAABADEABAAABD AAABADEABAACBD AAABADEAFAAABA AAABADEAFAAABD AAABADEAFAAFBA AAABADEAFCAFBA AAABADEAFEA[illegible]BA AAABADEAGEACBA
AAABADEHFAACBA AAABADEHFEAFBA AAABADERFAACBA AAABADERFEAABA AAABADERGEACBA AAABADKAHEAABG AAABADKEHEDABG AAABADKRABACBB
AAABADKRABACBG AAABADMAABBCBB AAABADMAHEDABD AAABADMEHEDABG AAABADMRHEDABG AAABADMRHEDABD AAABAFGAAGDCBG AAABAFGAHQAABD
AAABAFGEAGEABD AAABAFGEAGEABG AAABAFGEBBEABG AAABAFGEBGDABD AAABAFGEBGDABD AAABAFGEBGEABG AAABAFGEBQDABD AAABAFGEHGDABD
AAABAFGEHGEABD AAABAFGEHGEABG AAABAFGEHQDABD AAABAFGHABACBA AAABAFGRABACBD AAABAFGRABACBG AAABAFGRAGAABA AAABAFGRAGAABD
AAABAFGRAGAABG AAABAFGRAGACBA AAABAFGRBBAABG AAABAFGRBBACBD AAABAFGRBBDABG AAABAFGRBBDABD AAABAFGRBGAABG AAABAFGRBGDABG
AAABAFGRBGDABQ AAABAFGRHGAABA AAABAFGRHGAABD AAABAFGRHGAABG AAABAFKAABACBG AAABAFKAABEAFB AAABAFKAHFAAFB AAABAFKEAFEABG
AAABAFKEABEAFB AAABAFKEHBEABG AAABAFKEHBEAFB AAABAFKEHBDABG AAABAFKRABAABG AAABAFKRABACBB AAABAFKRHBAABG AAABAFMEAFGABG
AAABAFMEHBBABQ AAABAFMEHBGABG AAABAFMEHFBAFB AAABAFMEHGDABG AAABAFMRHGDABG AAABAFMRHGDABD AAABAHKAABACBB AAABAHKAHFAAFB
AAABAHKAHFEAFB AAABAHKBABAABG AAABAHKBABACBB AAABAHKBHBAABG AAABAHMAABBCFB AAABAHMAABBCHB AAABAHMAABBCBB AAABAHMAHBBAFQ
AAABAHMAHBBAHQ AAABAHMBABBCBQ AAABAHMBABBCFB AAABAHMBABDCFB AAABAHMRABDCHB AAABAHMRABFABD AAABAHMBHBBAFQ AAABAHMBHGDAHG
AAABAHMBHBDAHQ AAABAHMBHFDAFB AAABAHMBHFDAHB AAABAHMBHGBABQ AAABAHMBHBDABG AAABAHMBHGDABD AAABAHMGABGAFB AAABAHMGHBBABD
AAABAHMGHBGAFB AAABAHMGHFBAFB AAABBADAAAABCB AAABBADARAEDCB AAABBADBAAABCB AAABBADBABABCB AAABBADBACAAHB AAABBADBAFAAIB
AAABBADBAFABCB AAABBADBABAACB AAABBADBHAAACB AAABBADBHCAACB AAABBADBHFAAFB AAABBADBHFAAHB AAABBADBRAAACB AAABBADBRBAACB
AAABBADBRFAACB AAABBADEHAEACB AAABBADEHCEACB AAABBADEHFEAHB AAABBADEHFEAIB AAABBADEHQEACB AAABBADERAEACB AAABBADERQEACB
AAABBADGHAEACB AAABBADGHBEAFB AAABBADGHBEAHB AAABBADGHCEACB AAABBADGHFEAFB AAABBADGHFEAHB AAABBADGRAEACB AAABBADGRBEACB
AAABBADGRFEACB AAABBAEABAADBD AAABBAEABAAFBD AAABBAEABCAABD AAABBAEABGADBD AAABBAEABGAFBD AAABBAEABQABBD AAABBAEBBAAABA
AAABBAEBGAAABD AAABBAEBBAABBD AAABBAEBBBAABA AAABBAEBGBAABD AAABBAEBBBAABA AAABBAEBBGAABA AAABBAEBGAABBD AAABBAEBBGABBA
AAABBAFEHBGAHB AAABBAFGHBGAHB AAABBBDAAABBAB AAABBBDBAAAAAB AAABBBDAABABAB AAABBBDBACAAAB AAABBBDBEABGAB AAABBBDBHGBAHB
AAABBBDGHCBAHB AAABBBDEAABGAB AAABBBEABBAADA AAABBBEBBAAABD AAABBBEBBAAAIA AAABBBEBBCAABA AAABBBEBBEAADA AAABBBEHGEAFBA
AAABBBFBAAFAAF AAABBBFBACFAAF AAABBBFBEABGAB AAABBEFBEABDAF AAABBEFBEABGAB AAABBEFBRABDAB AAABBEHGBABDAB AAABCADAAFEAAB
```

```
AAABCADAAFEAHB AAABCAFAACBDAB AAABCAFAACBDCB AAABCAGBBADABD AAABCAGBBCDABD AAABCAGEAAEABD AAABCAGEAAJABD AAABCAGEAQJABD
AAABCAGEBAJABD AAABCAGEBCJABD AAABCAKEAFEAFB AAABCAKEAFJAFB AAABCAMBABBAFQ AAABCAMBAGBABQ AAABCAMEABBAFQ AAABCBGBAADABD
AAABCBGBBADABD AAABCBGBBCDABD AAABCBGBBCDABQ AAABCBGEBAEABD AAABCBGEBCDABQ AAABCBGEBCEABD AAABCDGEABEABD AAABCDGEABJABD
AAABCDBEBBEABD AAABCDBEBBEABG AAABCDKEABEABB AAABCDKEABEABG AAABCDKEABJABB AAABCDKEABJABG AAABCDMEABBABQ AAABCDMEABGABB
AAABCDMEABGABB AAABCFGEABEABD AAABCFGEABEABG AAABCFGEABJABD AAABCFGEABJABG AAABCFGEBBEABD AAABCFGEBGEABD AAABCFKEABEA[illegible]
AAABCFKEABJABB AAABCFMEABDABQ AAABCFMEABGABB AAABCHKBABDABB AAABCHMBABDABG AAABCHMBABDABQ AAABCHMGABGABB AAABDBEABBABDA
AAABDBEABBACDA AAABDBEABEABDA AAABDBEBBAACDA AAABDBEBBBABDA AAABDBEBBBACDA AAABDBEBBCACBA AAABDBEBBEACDA AAABDBEHBBABDA
AAABDBEHBBACDA AAABDBEHBCACBA AAABDBEHBEABDA AAABDBERBAACBA AAABDBERBEACBA AAABEAHEACBDAB AAABEAHEACBDCB AAABEBHEABGAAB
AAABFBGBBEACBA AAABFBGHBCACBA AAABFBGRBAACBA AAABFBGRBEACBA AAABGAHBAFBDCH AAABGAHBAQBGCB AAABGAHEACBDCB AAABGAHEAQBDCB
AAABGAHGAFBGCH AAABGBHEABBGAB AAABGBHEACBAAB AAACABEABGMBLD AAACABGABGDALG AAACADEABBAFBD AAACADEAFBAFBD AAACADEAFBAIBA
AAACADEAFEAFBA AAACADKAABAABB AAACAFGAABACBG AAACAFKAABAABB AAACAFKAABAABG AAACAFKAABDABB AAACBBEABGAAGA AAACBBEABGAAGD
AAACDBEBBBACBA AAACDBEBBEACBA AAACDBEBBGACBA AAACFBGBBBACBA AAACFBGBBGACBA AAACFBGFBBAFBA AAACFBGFBGAFBA AAADAAFAAFBBAB
AAADAAFEAFBGAB AAADBEFBEABGAB AAADBEFBEABGAF AAADBEFBRABDAB AAADBENBBACBAH AAADBENBGACBAH AAADCAFAAFBDAB AAAEAAEABBAFBD
AAAEAAEABGAFBD AAAEAAKAABEAFB AAAEAAKAHBEAFB AAAEABKAABEAFB AAAEABKAHBEAFB AAAEABKEABEAFB AAAEAHKAABAABB AAAEAHKAABAABG
AAAEAHKAAHEABB AAAEAHKAABEABG AAAEAHKAHBEAFB AAAEAHMAABDABB AAAEBBEABGAAGA AAAEBBEABGAAGD AAAEBBEABGABGD AAAECAKEABEAFB
AAAECBGEBBJABD AAAECBGRBBDABD AAAECBGRBBDABQ AAAECBKEABMABG AAAEDBERBBACBA AAAEDBERBGACBA AAAFBGNBBACBAH AAAFBGNBGACBAH
AAAGAHKAABAABG AAAGCBGEBBDABD AAAGCBGEBBEABD AAAGCBGEBBJABG AAAGCBGEBBDABG AAAGCBGRBBDABG AAAGFBGHBBACBA AAAIAAKAABEABB
AAAIAAKAHBEABB AAAIAHKAHBEABB AAAICAKEABEABB AABAABBAAAABAD AABAABBAAABBAD AABAABBAAABCAB AABAABBAAABCBD AABAABBAAAEBAB
AABAABBAAAGCBB AABAABBAABBBFD AABAABBAABBCAD AABAABBAABECAE AABAABBAABECAG AABAABBAAEBCBA AABAABBABABCGB AABAABBABAECAB
AABAABBABAGCAB AABAABBABAGFGB AABAABBABAGHGB AABAABBABBGCQB AABAABBABBGFQG AABAABBABEBCGA AABAABBABEBHGA AABAABBACABCAB
AABAABBACAEHAB AABAABBACAGCBB AABAABBACAGHAB AABAABBACBBCQA AABAABBACBGCQE AABAABBACBGCQG AABAABBACEBCBA AABAABBAFAGFBB
AABAABBAHEBHBA AABAABBBAABBAD AABAABBBAABBBD AABAABBBABBBAD AABAABBBABBBBD AABAABBBABBBFA AABAABBBADBBBB AABAABBBADBCBA
AABAABBBAEBCBA AABAABBBAGBBBA AABAABBBAGBCBA AABAABBBBDBAGA AABAABBBBDBCAA AABAABBBBEBAGA AABAABBBBGBAGA AABAABBBCDBAAB
AABAABBBCDBABA AABAABBBCDBABB AABAABBBCDBBAB AABAABBBCEBABA AABAABBBCEBBBA AABAABBBCGBABA AABAABBBFEBABA AABAABBBFGBABA
AABAABBBHEBABA AABAABDBAAABAB AABAABDBAABBAB AABAABDBABABFB AABAABDBABACAB AABAABDBABBBAB AABAABDBADABAB AABAABDBCBBAAB
AABAABDBHBAAFB AABAABDFABABAB AABAABEFBGAFBD AABAABFBAABBAB AABAABFBABBBAB AABAABGAAEACBD AABAABGBAAAABD AABAABGBAAABBD
AABAABGBADAABD AABAABGBADABBD AABAABGBAEAABD AABAABGBAEABBD AABAABGBHDAABA AABAABGBHEAABE AABAABGFABABBD AABAABGFAGABBG
AABAABIBBBAADB AABAABIBBBABQG AABAABIFBBABQG AABAABIFBBAHQB AABAABKBAEAABG AABAABKBAEABBG AABAABKFABABBB AABAABKFABABBG
AABABABAAAAAAA AABABABBABAAFE AABABABBABABFD AABABABBABABFE AABABABBABBBFA AABABABBADAAAE AABABABBADAACE AABABABBAGAABE
AABABABBABABBE AABABABBADAABR AABABABBADAACR AABABABBAQABBR AABABABBAQABCR AABABABBAQBBBH AABABABBBBBBQD AABABABBBGBBBD
AABABABCBBBBBD AABABABBCDBABH AABABABBEBABQE AABABABBEFABQR AABABABBHAAABR AABABABBHAAACR AABABABBHFAACR AABABABBRAAAGR
AABABABFABABCD AABABABFABABCE AABABABFABABFA AABABABFAFABCR AABABABFAFABFB AABABABFAGABBA AABABABFAQABBB AABABABFAQABCB
AABABABFEBABAD AABABABFEBABAE AABABABFEFABAR AABABABFEFABQB AABABABQAFEBFB AABABABQAFGBFB AABABABQAQEBBB AABABABQCABHBH
AABABADBAFBBIB AABABADFABABCB AABABADFADABCB AABABADGAFEAIB AABABAEBBBAABE AABABAEBBBADBA AABABAEBBBADBD AABABAEBBDAAGD
AABABAEBBDADGE AABABAEBBGAABE AABABAEBBGAAGA AABABAEBBGAAGD AABABAEBBGADBA AABABAEBBGADBD AABABAEBBGADGA AABABAEBBGADGD
AABABAEBBDAABD AABABAEBBQABBD AABABAEBBQABGD AABABAEFBBAHBE AABABAEFBDAHGD AABABAEFBGAHBE AABABAEFBGAHGA AABABAEFBGAHGD
AABABAEFBDABBD AABABAEFBQABGD AABABAEFBQAHBD AABABAFBAFBBIB AABABAHBAFBBIH AABABAHQADGBCR AABABAHQAFBBIB AABABAHQAFGBCR
AABABAHQBBGFAB AABABAHQBFGFAB AABABAHQBFGFAR AABABAHQBFGFCR AABABAHQBFGHCR AABABAIBBBAABE AABABAIBBBAAQB AABABAIBBBABQE
AABABAIBBBADBB AABABAIBBDAABR AABABAIBBDAAGB AABABAIBBDADBR AABABAIBBDADGE AABABAIBBDADGR AABABANBBDAAAE AABABANBBDAACE
AABABANBBDAAGE AABABANBBDAAGR AABABANBBDAAQA AABABANBBDAAQD AABABANBBDAAQR AABABANBBFAACR AABABANBBGAAGA AABABANBBGAAGD
AABABANBBGADGD AABABANBBQAACD AABABANBBQAACR AABABANBBQAAGD AABABANQBFEHCR AABABANQBQEHCR AABABARQBBEFAB AABABARQBFEFAB
AABABARQBFEFCR AABABARQBFEHCR AABABBDBAAAAAB AABABBDBADAAAB AABABBDBCABAAB AABABBEBBAAAGD AABABBEBBDAADE AABABBEBBDAAGE
AABABBEBBEAADE AABABBEBBEAAGE AABABBEFBAAHGD AABABBEFBDAHDE AABABBEFBDAHGE AABABBEFBEAHDE AABABBEFBEAHGE AABABBFAADBBAB
AABABBFAEDGBAB AABABBIBBBAADB AABABBIBBBAADE AABABBIBBBAAQB AABABBIBBDAABR AABABBIBBDAADB AABABBIBBDAADE AABABBIBBDAAGB
AABABBIBBDAAGE AABABBIFBDAFDB AABABBIFBDAFDR AABABBIFBDAFGB AABABBIFBDAFGR AABABBIFBDAFQE AABABBIFBDAFQR AABABBIFBDAHBR
AABABBIFBDAHDB AABABBIFBDAHDE AABABBIFBDAHGB AABABBIFBDAHGE AABABBRBBDAAAB AABABBRFBDAFAB AABABBRFBDAHAB AABABGHBADBBAB
AABABGHBADBDAB AABABGHBADCBAB AABABGHBBDBBAB AABABGHBBDBDAB AABABGHBCDCBAB AABABGHBGDBBAB AABABGNBBDADAD AABABGNBBDADAE
AABABGNBBDADAR AABABGNBBDADGD AABABGNBBDADGE AABABGRBBDADAB AABACADAADEDCB AABACADAAFEDFB AABACAFAAABDCB AABACAFAADBDCB
AABACAFAADBDCB AABACAFBAABDCB AABACAFBABBDCB AABACAFGAABDCB AABACAFGABBDCB AABADBEBBBACDE AABADBEBBEACDE AABADBIBBBACDE
AABADAHBBEABGE AABAEAHAAABDCB AABAEAHBAABDCB AABAEAHGAABDCB AABAEAHGABBDCB AABAFBGBBBACDE AABAFBGBBEACDE AABBAAEBBAAABD
AABBAAEBBAABBD AABBAAEBBBABBD AABBAAEBBBABBE AABBAAEBBGAABE AABBAAEBBGABBE AABBABDBABDQAB AABBBAEBBAAABD AABBBAEBBAABBD
AABBBAEBBBAABD AABBBAEBBGAABD AABBBAIBBBAABB AABBCAGBAGDABD AABBCAGBBBDABD ABBACAFAAABDCB ABBACAFAABBDCB
```

DRAWINGS WITH 25 CROSSINGS:

```
AAAAAAAAAAAABB AAAAAAAAAAABAB AAAAAAAAAABABF AAAAAAAAAABBAF AAAAAAAAAABBBF AAAAAAAAABAAFG AAAAAAAAABABAG AAAAAAAAABABFG
[illegible]    [illegible]    AAAAAAAAABGBAG AAAAAAAAAFBBAF AAAAAAAABABBGF AAAAAAAABABFBF AAAAAAAABABFGB AAAAAAAABBGBAG
[illegible]    [illegible]    AAAAAABAAABAAA AAAAAABAAAEAAB AAAAAABAAAEBAB AAAAAABAAAEBBD AAAAAABAAAEBCB AAAAAABAABAAAA
[illegible]    [illegible]    AAAAAABAABEBCD AAAAAABAABEBCG AAAAAABAABEBFD AAAAAABAABEBFG AAAAAABAABGBFG AAAAAABAAFEBAB
[illegible]    [illegible]    AAAAAABAAFGBFB AAAAAABAAGEBBD AAAAAABAAGEBBG AAAAAABAAQABBD AAAAAABAAQABCB AAAAAABAAQBBBF
[illegible]    [illegible]    AAAAAABAAQEBBB AAAAAABAAQGBBB AAAAAABAAQGBCB AAAAAABABABFGF AAAAAABABAEFAD AAAAAABABAGFGB
[illegible]    [illegible]    AAAAAABABFEFAF AAAAAABABFEFAB AAAAAABABFGFAB AAAAAABABQBBGF AAAAAABABQEBCD AAAAAABABQEFGB
[illegible]    [illegible]    AAAAAABAFAGFBF AAAAAABAFAGFBD AAAAAABAFGEFBG AAAAAABAFQEFBB AAAAAABAFQEFBD AAAAAABAHAEFBD
[illegible]    [illegible]    AAAAAABAHQAFBD AAAAAABAHQBFBF AAAAAABAHQEFBB AAAAAABAHQGFBB AAAAAABBAAABAD AAAAAABBAAAGAB
[illegible]    [illegible]    AAAAAABBABGBCD AAAAAABBABABCD AAAAAABBABABCG AAAAAABBABABFG AAAAAABBABAGAA AAAAAABBABAGBD
[illegible]    [illegible]    AAAAAABBABBGFD AAAAAABBABBBFD AAAAAABBABBGCD AAAAAABBABBGFA AAAAAABBAFABCB AAAAAABBAFABFB
[illegible]    [illegible]    AAAAAABBAGBBBD AAAAAABBAGBGBA AAAAAABBAQABBB AAAAAABBAQBBBD AAAAAABBAQBBBF AAAAAABBAQBBCF
[illegible]    [illegible]    AAAAAABBQFBBCF AAAAAABBQBBBF  AAAAAABBQBBCD  AAAAAABBQBBCF  AAAAAABFBAAFAD AAAAAABFBAAFCD
[illegible]    [illegible]    AAAAAABFBBAFAD AAAAAABFBBAFCD AAAAAABFBGAFBG AAAAAABFBGAFGA AAAAAABFBQAFBB AAAAAABFBQAFBD
[illegible]    [illegible]    [illegible]    AAAAAADAAFAAAB AAAAAADAAFABAB AAAAAADAAFABCB AAAAAADAAFABFB AAAAAADAAFBBIB
[illegible]    [illegible]    AAAAAADBAAABAB AAAAAADBAABGCB AAAAAADBABABCB AAAAAADBABABFB AAAAAADBABBBFB AAAAAADBABBGCB
[illegible]    [illegible]    AAAAAADBAFAAFB AAAAAADBAFABAB AAAAAADBAFBBFB AAAAAADBAFBGCB AAAAAADBHBAACB AAAAAADBHBAAFB
[illegible]    [illegible]    AAAAAADGABBGCB AAAAAADGABBGCB AAAAAADGAFBGCB AAAAAADGAFEAAB AAAAAADGAFEACB AAAAAADGAFEAFB
[illegible]    [illegible]    [illegible]    [illegible]    AAAAAAEABAAABA AAAAAAEABAEBGD AAAAAAEABAEFBD AAAAAAEABGAAGA
[illegible]    [illegible]    AAAAAAEABGAFGA AAAAAAEABGAFGD AAAAAAEABGEBBD AAAAAAEABQABGD AAAAAAEABQAFBD AAAAAAEABQEBBD
[illegible]    [illegible]    AAAAAAEAFGAFBA AAAAAAEBBAAABA AAAAAAEBBAAABD AAAAAAEBBAABGD AAAAAAEBBGABGD AAAAAAEBFBAABA
[illegible]    [illegible]    AAAAAAEFBAABBD AAAAAAEFBAABGA AAAAAAEFBAAFBA AAAAAAEFBBABBD AAAAAAEFBBAFBA AAAAAAEFBGABBD
[illegible]    [illegible]    AAAAAAKAABAABB AAAAAAKAABAFBD AAAAAAKAABEBBB AAAAAAKBABABBB AAAAABBAAAAAAB AAAAABBAAAACBD
[illegible]    [illegible]    AAAAABBAAABBAF AAAAABBAAABBAD AAAAABBAAABBAF AAAAABBAAABBBF AAAAABBAAABCAB AAAAABBAAABCBD
AAAAABBAAAEABD AAAAABBAAAEBBB AAAAABBAAAGAAB AAAAABBAAAGBAB AAAAABBAAAGBBB AAAAABBAABACAD AAAAABBAABBBFD AAAAABBAABBCAD
```

5

```
AAAAABBAABBCFD AAAAABBAABEAAD AAAAABBAABEBAG AAAAABBAABEBFG AAAAABBAABECAB AAAAABBAABECFG AAAAABBAABBBFG AAAAABBAABGCAG
AAAAABBAABGCFG AAAAABBAAEBCBA AAAAABBAAGBBBD AAAAABBAAGBCBD AAAAABBAAGEBBG AAAAABBABABAGF AAAAABBABABBGF AAAAABBABABCAD
AAAAABBABABCBF AAAAABBABABCGB AAAAABBABABCGD AAAAABBABABFAF AAAAABBABAEAAD AAAAABBABAEBAD AAAAABBABAEFAB AAAAABBABAEFGB
AAAAABBABAEFGD AAAAABBABAGAGB AAAAABBABAGBGB AAAAABBABAGCBB AAAAABBABAGFAB AAAAABBABBECAD AAAAABBABBECAG AAAAABBABBEFAD
AAAAABBABBEFAG AAAAABBABBGCAG AAAAABBABBGFAG AAAAABBABEBCGA AAAAABBABBBCGD AAAAABBABGBFGD AAAAABBABGECGD AAAAABBABGEFGD
AAAAABBAFABABF AAAAABBAFABCBB AAAAABBAFABCBD AAAAABBAFAEFBB AAAAABBAFAGABB AAAAABBAFBBCFD AAAAABBAFBECFD AAAAABBAFBECFG
AAAAABBAFBGCFB AAAAABBAFGBFBD AAAAABBAHABFBF AAAAABBAHAEABD AAAAABBAHAEFBB AAAAABBAHAGFBB AAAAABBAHGBFBD AAAAABBAHGEFBG
AAAAABBBAABCAF AAAAABBBAABCBD AAAAABBBAABCBF AAAAABBBAABGBF AAAAABBBABBCAD AAAAABBBABBGAD AAAAABBBABBGBD AAAAABBBABBGFA
AAAAABBBABBGFB AAAAABBBAGBGBA AAAAABBBBBBCAD AAAAABBBBEBCGA AAAAABBBBGBAGD AAAAABBBBGBBBD AAAAABBBBGBCBD AAAAABBBBGBCGA
AAAAABBBFEBCBA AAAAABBBFGBABD AAAAABDAAABAAB AAAAABDAAABBAB AAAAABDAABACAB AAAAABDAABBBFB AAAAABDAABBCAB AAAAABDAABBCFB
AAAAABDAABEAAB AAAAABDBAAABAB AAAAABDBAABGAB AAAAABDBABABFB AAAAABDBABACAB AAAAABDBABACFB AAAAABDBABBCAB AAAAABDBABBGAB
AAAAABDBHBAAFB AAAAABDBHBBAFB AAAAABEABAAAGA AAAAABEABAAAGD AAAAABEABAACGD AAAAABEABAAFGD AAAAABEABAEBGD AAAAABEABAECBD
AAAAABEABAEFBD AAAAABEABEAAGD AAAAABEABEACGD AAAAABEABEAFGD AAAAABEABGAFGD AAAAABEABGEABD AAAAABEABGEBGD AAAAABEABGECGD
AAAAABEABGEFBD AAAAABEAFAAABA AAAAABEAFAAABD AAAAABEAFAACBD AAAAABEAFAAFBD AAAAABEAFBEFBD AAAAABEAQAACBA AAAAABEBQAAAGA
AAAAABEBBGAAGD AAAAABEBBGABGD AAAAABEBBGACGD AAAAABEBFAAABA AAAAABEFBAABGD AAAAABEFBAAFBD AAAAABEFBAAFGA AAAAABEFBBAFBD
AAAAABEFBEABGD AAAAABEFBEAFBD AAAAABEFBGABBD AAAAABEFBGABGD AAAAABEFBGAFBD AAAAABEFBGAFGA AAAAABEFQAABBA AAAAABEFQGABBA
AAAAABEHBAABGD AAAAABEHBAACGA AAAAABEHBAAFGA AAAAABEHBEABGD AAAAABEHBEACBD AAAAABEHBEAFBD AAAAABEHBGABGD AAAAABEHBGACGD
AAAAABEHQAABBA AAAAABEHQEACBA AAAAABFAABBBAB AAAAABFAABGAAB AAAAABFBAABAAF AAAAABFBAABBAF AAAAABFBAABGAB AAAAABFBAAFBAF
AAAAABFBABBCAB AAAAABFBABBGAB AAAAABGAAAAABA AAAAABGAAAAABD AAAAABGAAAABBD AAAAABGAAAACBA AAAAABGAABEBBD AAAAABGABEAABD
AAAAABGAAEABBD AAAAABGAAGABBD AAAAABGAAGEBBG AAAAABGABAECBD AAAAABGAHAAABA AAAAABGAHAAABD AAAAABGAHAAFBD AAAAABGAHBEFBD
AAAAABGAHEAABD AAAAABGAHEAFBD AAAAABGAHGEFBG AAAAABGBAAAABD AAAAABGBAAABBD AAAAABGBAAAGBA AAAAABGBABABBG AAAAABGBAEAABD
AAAAABGBAEABBD AAAAABGBAEAGBG AAAAABGBAGABBD AAAAABGBAGAGBA AAAAABGBHAAABA AAAAABGFAAABBA AAAAABGFABABBG AAAAABGFAEABBG
AAAAABGFBGABBD AAAAABGHAAABBA AAAAABGHABABBD AAAAABGHABACBD AAAAABGHAEABBG AAAAABGHAEACBA AAAAABGHBEACBD AAAAABKAABEBBB
AAAAABKAABEBBG AAAAABKAAEAABG AAAAABKAAEABBG AAAAABKAAGEABG AAAAABKAHEAABG AAAAABKAHGEABG AAAAABKBABAAFB AAAAABKBABABFB
AAAAABKBABACFB AAAAABKBAEAABG AAAAABKBAEABBG AAAAABKBHBAAFB AAAAABKHABABBB AAAAABKHABABBB AAAAABKHABACBB AAAAABKHABACBG
AAAAADEABAAABA AAAAADEABAAABD AAAAADEABAACBD AAAAADEABAAFBD AAAAADEAFAACBA AAAAADEAFAAFBA AAAAADEAFBAIBD AAAAADEAFEAABA
AAAAADEHBAACBA AAAAADEHBAAFBA AAAAADEHFBAIBA AAAAADEHFEACBA AAAAADEHFEAFBA AAAAADGAABACBD AAAAADGAABEABD AAAAADGAAEAABG
AAAAADGABAEABD AAAAADGABBEFBD AAAAADGABBEFBG AAAAADGAHBEFBG AAAAADGAHEAABA AAAAADKAABACBB AAAAADKAABACBG AAAAADKAABEABB
AAAAADKAABEABG AAAAAFKAABAABG AAAAAFKAABAAFB AAAAAFKAABACBB AAAAAFKAABEABB AAAAAFKAHBEABG AAAAAFKAHBEAFB AAAABABA[illegible]AACA
AAAABABAAAADCA AAAABABAABADFA AAAABABAAGADBA AAAABABAEAADAA AAAABABAEBADAA AAAABABARAAAAA AAAABABBABADFA AAAABABBABAGFB
AAAABABBAFAGCB AAAABABBAGADBA AAAABABBAGBGBA AAAABABBAQBGBF AAAABABBAQBGBH AAAABABBEBAGAA AAAABABBEFAGAB AAAABABBHGAABA
AAAABABBRBADCA AAAABABFAFABCB AAAABABFBQAFGB AAAABABFEFABAB AAAABABGAQBDBH AAAABABGAQBGCH AAAABABGQBABH AAAABABGRQBABH
AAAABABGRQBDGH AAAABABQAQBBCF AAAABABQAQBBCH AAAABABQAQBDBF AAAABABQAQBDCH AAAABABQBQBBBF AAAABABQBQBBGF AAAABABQBQBFBF
AAAABABQRQBFGH AAAABABQRQBDGH AAAABADAAAAAAB AAAABADAAAEAAB AAAABADAAAEBAB AAAABADAAAEBCB AAAABADAAFABIB AAAABADAAFADIB
AAAABADAAFEBFB AAAABADAAFEDIB AAAABADAAQABCB AAAABADAAQADCB AAAABADAAQEDCB AAAABADAEAEBAB AAAABADAEFEBAB AAAABADBAAAGAB
AAAABADBABABFB AAAABADBABADFB AAAABADBABAGAB AAAABADBABBBFB AAAABADBAFAAIB AAAABADBAFABFB AAAABADBAFADFB AAAABADBAFAGAB
AAAABADBAFBGFB AAAABADBAFBGIB AAAABADBAQAACB AAAABADBAQBGCB AAAABADBEAAGAB AAAABADBHAAACB AAAABADBHBAAFB AAAABADBHFAAFB
AAAABADBRBADCB AAAABADBRFADCB AAAABADBRQAACB AAAABADFABABAB AAAABADFAFABAB AAAABADGABEDFB AAAABADGAFBGFB AAAABADGAFBGIB
AAAABADGAFEAIB AAAABADGAFEDFB AAAABADGAQBGCB AAAABADGAQEACB AAAABADGHAEACB AAAABADGHBEAFB AAAABADGHFEAFB AAAABADGRBEDCB
AAAABADGRFEDCB AAAABADGRQEACB AAAABADQAAEBAB AAAABADQAAEDCB AAAABADQABBDFB AAAABADQABEBAB AAAABADQABEDCB AAAABADQAFBBFB
AAAABADQAFBBIB AAAABADQAFEBAB AAAABADQAFEDCB AAAABADQAQBBCB AAAABADQEAEBAB AAAABADQEBEDAB AAAABADQEFEDAB AAAABADQRBEDCB
AAAABADQRFEDCB AAAABAEABAAAGA AAAABAEABAEFGD AAAABAEABGEFGD AAAABAEABQAFGD AAAABAEABQEFBD AAAABAEBBAAAGA AAAABAEBBAAAGD
AAAABAEBRBADBA AAAABAEBRBADBD AAAABAEBBGAAGA AAAABAEBBGAAGD AAAABAEBBBADBA AAAABAEBBGADBD AAAABAEBBGADGA AAAABAEBBGADGD
AAAABAEBBQAABD AAAABAEBBQABBD AAAABAEBBQABGD AAAABAEFBAAFBD AAAABAEFBBABBD AAAABAEFBBAFBA AAAABAEFBGABBD AAAABAEFBGABGD
AAAABAEFBGAFBA AAAABAFAAABBCF AAAABAFAAAGBCB AAAABAFAAFBDIF AAAABAFAAQBDCF AAAABAFAAQGDCB AAAABAFBAAFBAF AAAABAFBAFBBIF
AAAABAFBAFBGIB AAAABAFBAQBBCF AAAABAFBAQBGCB AAAABAFBEAFGAF AAAABAFGABBGCB AAAABAFGAQBACB AAAABAFGHABACB AAAABAFGRBGDCB
AAAABAFGRFBDCF AAAABAFGRFGDCB AAAABAFGRQGACB AAAABAFQAABDCF AAAABAFQAAGDCB AAAABAFQABBBCB AAAABAFQABGDCB AAAABAFQAFBDCF
AAAABAFQAFBDIB AAAABAFQAQBDCB AAAABAFQRBGDCB AAAABAFQRFBDCF AAAABAFQRFGDCB AAAABAHAAABACH AAAABAHAAABDCH AAAABAHAAFBAIH
AAAABAHAAFBBIF AAAABAHAAFBBIH AAAABAHAAFBDIF AAAABAHAAQBACH AAAABAHAAQBBCF AAAABAHAAQBBCH AAAABAHAAQBDCF AAAABAHAAQGBCB
AAAABAHAAQGDCB AAAABAHABQBFCF AAAABAHABQGFCB AAAABAHBAAFAAH AAAABAHBAFBDIH AAAABAHBAFBGIF AAAABAHBAFBGIH AAAABAHBAQBDCH
AAAABAHBAQBBCF AAAABAHBAQBGCH AAAABAHBGAFBAF AAAABAHGAABACH AAAABAHGAABDCH AAAABAHGAABGCH AAAABAHGAFBDCH AAAABAHGAFBGCH
AAAABAHGAFBGIB AAAABAHGAQBGCB AAAABAHGAQGACB AAAABAHGBBGDCB AAAABAHGBFBDCF AAAABAHGBFBDCH AAAABAHGBFGDCB AAAABAHGBQBACH
AAAABAHGBQGACB AAAABAHQAABBCF AAAABAHQAABBCH AAAABAHQAABDCF AAAABAHQAAGBCB AAAABAHQAAGDCB AAAABAHQABGBCB AAAABAHQABGDCB
AAAABAHQAFBBCF AAAABAHQAFBBCH AAAABAHQAFBBIB AAAABAHQAFBDCF AAAABAHQAFBDIB AAAABAHQADBBCB AAAABAHQADBDCB AAAABAHQBABFCF
AAAABAHQBAGFCB AAAABAHQBBBFCB AAAABAHQBBGBCB AAAABAHQBBGDCB AAAABAHQBFBBCF AAAABAHQBFBDCF AAAABAHQBFBDCH AAAABAHQBFBFCH
AAAABAHQBFGBCB AAAABAHQBFGDCB AAAABAHQBQBBCH AAAABAHQBQBFCB AAAABANABGEFGD AAAABANABQEFCD AAAABANABQEFGD AAAABANBBGAAGA
AAAABANBBGAAGD AAAABANBBGADGA AAAABANBBGADGD AAAABANBBQAACD AAAABANBBQAAGD AAAABANBBQABCD AAAABANBBQADGD AAAABANFBAAFCD
AAAABANFBAAFGD AAAABANFBGABGD AAAABANFBQABGD AAAABANQBAEFCD AAAABANQBAEFGD AAAABANQBGEBGD AAAABANQBGEDGD AAAABANQBQEBGD
AAAABANQBQEDGD AAAABBBAAABAAH AAAABBBAAABABH AAAABBBAAABBBF AAAABBBAAABBBH AAAABBBAAACABD AAAABBBAAACBAH AAAABBBAAACBBD
AAAABBBAABABFD AAAABBBAABBBFA AAAABBBAABCAAA AAAABBBAABCAAD AAAABBBAABCBAD AAAABBBAABCBBD AAAABBBAABCBFB AAAABBBAABEAAD
AAAABBBAABEAFD AAAABBBAABEBBD AAAABBBAABEBFB AAAABBBAABGBFB AAAABBBAAGABBD AAAABBBABABFGF AAAABBBABACABH AAAABBBABBEFAD
AAAABBBABECAGA AAAABBBABGBFGD AAAABB[illegible]ABBGF AAAABBBAEABBGH AAAABBBAEACBAD AAAABBBAEACBAH AAAABBBAEACBGD AAAABBBAEAEBAB
AAAABBBAEAEBGB AAAABBBAEAGBGB AAAABBBAEBABAD AAAABBBAEBBBAD AAAABBBAEBCBAA AAAABBBAEBEBAB AAAABBBAHBCFBD AAAABBBAHBEAFD
AAAABBBAHBEFBD AAAABBBAHGAFBD AAAABBBBABAAFD AAAABBBBABAGBB AAAABBBBABBAFD AAAABBBBABBBBD AAAABBBBABBBFB AAAABBBBABBGAD
AAAABBBBABCAAD AAAABBBBABCABD AAAABBBBAEAGBA AAAABBBBAEBGBA AAAABBBBAEBCBA AAAABBBBAGAABD AAAABBBBAGBABD AAAABBBBAGBBBA
AAAABBBBBBBBBB AAAABBBBBBBBBD AAAABBBBBECAGA AAAABBBBBGBAGD AAAABBBBBBCAGA AAAABBBBEBABAD AAAABBBBEBAGAA AAAABBBBEBABAB
AAAABBBBEBBBAD AAAABBBBHEAABA AAAABBBBHEBABA AAAABBBBHGAABA AAAABBBBHGBABA AAAABBBFAEABBA AAAABBBFBECFGA AAAABBBFBGABBD
AAAABBBFBGABGD AAAABBBFBGAFBD AAAABBBHAEABBA AAAABBDAAACBAB AAAABBDAEACBAB AAAABBDBAAAAAB AAAABBDBAACAAB AAAABBDBABAAFB
AAAABBDBABBAFB AAAABBDBABBGAB AAAABBDBABCAAB AAAABBDBABCBAB AAAABBDBEBABAB AAAABBDBEBBBAB AAAABBEABAAADA AAAABBEABEAADA
AAAABBEBBAAAGD AAAABBEBBGAAGD AAAABBEFBAAFBD AAAABBEFBBABBD AAAABBEFBBAFDA AAAABBEFBGABBD AAAABBEFBEABBD AAAABBEHBAAFDA
AAAABBEHBBAFDA AAAABBEHBEAFDA AAAABBFAAABAAB AAAABBFAABBBAB AAAABBFAABBBAB AAAABBFAABGAAB AAAABBFAEABBAB AAAABBFBAABBAF
AAAABBFBAAFAAF AAAABBFBABBGAB AAAABBFBEABGAF AAAABBFBEBBBAB AAAABBHAAADAAB AAAABBHAABCBAB AAAABBHAABGAAB AAAABBHAABGBAB
AAAABBHABABFAB AAAABBHABACAAB AAAABBHABACFAB AAAABBHABBCAAB AAAABBHABBGAAB AAAABBHABBBBAB AAAABBHBAACAAB AAAABBHBAACCAB
AAAABBHBAAFAAF AAAABBHBABCAAB AAAABBHBBACAAH AAAABBHBBAFAAF AAAABBHBBAFAAH AAAABBHBBBCAAB AAAABBHBGACGAB AAAABBHBGAFGAH
AAAABBHGBBBBAB AAAABBHGBBCBAB AAAABBHFBACFAB AAAABBHFBACFAH AAAABBNABAAAAA AAAABBNABAAAAD AAAABBNABACAAA AAAABBNABACAGA
AAAABBNAABAEBAD AAAABBNAAEAAGA AAAABBNABBAFBD AAAABBNBBAAAAD AAAABBNBBAAAGD AAAABBNBBACAAH AAAABBNBBACAGA AAAABBNBEAABA
AAAABBNBBBAAGA AAAABBNBBGABBD AAAABBNBBACBBA AAAABBNFBACBGH AAAABBNHBAAFAA AAAABDEABAAABA AAAABDEABAAABD AAAABDEABAAFBD
AAAABDEABBAFBD AAAABDEABBEABD AAAABDEABEAAAA AAAABDEHBAAFBA AAAABDEHBBAFBA AAAABDEHBEAFBA AAAABEFBAABBAF AAAABEFBAABBAB
AAAABEFBEABGAF AAAABEFBEAFBAF AAAABEFGEABGAB AAAABEFDAABDAB AAAABEFDEABBAB AAAABEHBAABDAH AAAABEHBAABBAF AAAABEHBAABBAH
AAAABEHBAACDAB AAAABEHBBACDAH AAAABEHBBABGAF AAAABEHBBACBAB AAAABEHBAABGAB AAAABEHDAABDAB AAAABEHDAABDAB AAAABEHDBABDAH
AAAABEHBBACBAH AAAABEHBBACDAB AAAABENBBAADAA AAAABENBBACDAD AAAABENBBACDAH AAAABENFBACBAD AAAABENBBACDAA AAAABENBBACDAD
```

```
AAAABGHAAABBAB  AAAABGHAAABDAB  AAAABGHABABBAH  AAAABGHABACBAH  AAAABGHBAABDAH  AAAABGHBAAFDAF  AAAABGHBBAFDAF  AAAABGHBBAFDAH
AAAABGNABAADAA  AAAABGNABAADAD  AAAABGNABACBGH  AAAABGNABACDAA  AAAABGNABACDAD  AAAABGNABACDGD  AAAABGNABAEDGD  AAAABGNBBAADAD
AAAABGNBBAADBD  AAAABGNBBACBGD  AAAABGNBBACBGH  AAAABGNBBACDAD  AAAABGNBBACDAH  AAAABGNBBACDGA  AAAABKNABAADCA  AAAABKNABAADCD
AAAABKNABACBGH  AAAABKNABACDCA  AAAABKNABACDCD  AAAABKNABACDGD  AAAABKNABAEDGD  AAAACADAABEDAB  AAAACADAABEDCB  AAAACADAABEDFB
AAAACADAAFBDIB  AAAACADAAFEACB  AAAACADAAFEAFB  AAAACADAAQBDCB  AAAACADBABBDFB  AAAACADBAFBAIB  AAAACADBAFBDFB  AAAACADBAQBACB
AAAACADGAAADCB  AAAACADGAAEDAB  AAAACADGABBDCB  AAAACADGABEDCB  AAAACADGABEDFB  AAAACADGAFBDIB  AAAACADGAFBDCB  AAAACADGAFEAAB
AAAACADGAFEACB  AAAACADGAFEAFB  AAAACADGAFEDAB  AAAACADGAQBDCB  AAAACADQAABDCB  AAAACADQABBDCB  AAAACADQAFBDCB  AAAACAFAAABACB
AAAACAFAAABDAB  AAAACAFAAABDCB  AAAACAFAABGDCB  AAAACAFAAQBACB  AAAACAFBAABDCF  AAAACAFBAFBAIB  AAAACAFBAFBDCF  AAAACAFBAQBACB
AAAACAFBABBDCB  AAAACAFGAFBAIB  AAAACAFBADBACB  AAAACAFQAABDAB  AAAACAFQABBDAB  AAAACAFQAFBDAB  AAAACAKAABEABG  AAAACAKAABEAFB
AAAADAEBBBABBA  AAAADAEBBGABBA  AAAADAEBBGABGA  AAAADAEFBBABBA  AAAADANABGCBGA  AAAADBEABAACDA  AAAADBEABEACDA  AAAADBNABGCBGA
AAAADENBBAABAA  AAAADENBBACBAH  AAAADENFBAABAA  AAAADENFBACBAH  AAAADGNABAABAA  AAAADGNABACBAA  AAAADGNABACBGA  AAAADGNBBACBAH
AAAADGNBBACBBA  AAAADGNBBEABGA  AAAADGNHAAABAA  AAAADGNHBEABGA  AAAADKNABACBGA  AAAADKNHBEABGA  AAAAFDGABAACBA  AAAAFDGABEACBA
AAAAFDGHBAACBA  AAAAFDGHBEACBA  AAAAGEHGAABDAB  AAABAABAAAAACA  AAABAABAAAABAA  AAABAABAAAABAD  AAABAABAAAABCA  AAABAABAAAABCD
AAABAABAAADBAD  AAABAABAAADBCA  AAABAABAAADBCD  AAABAABAABDBHD  AAABAABAABEBHD  AAABAABAABEBHB  AAABAABAACABAD  AAABAABAACABCD
AAABAABAACDBCD  AAABAABAACDBCD  AAABAABABADFAA  AAABAABABADFAD  AAABAABABAEFAD  AAABAABABBBFAD  AAABAABABBDBAQ  AAABAABABBDBCQ
AAABAABABBDFAB  AAABAABABBDFCD  AAABAABABBEFAG  AAABAABABBGFAB  AAABAABABCAFAA  AAABAABABCAFAD  AAABAABABCBBAQ  AAABAABABCBFAF
AAABAABABCDBAA  AAABAABABCDBAD  AAABAABABCDBBD  AAABAABABCDBCA  AAABAABABCDBCD  AAABAABABCDFAB  AAABAABABCDFBQ  AAABAABABCEBAD
AAABAABABCEBCD  AAABAABABCEFAB  AAABAABAFCEFCD  AAABAABABCGFAB  AAABAABABFBFAF  AAABAABABFDFAB  AAABAABABFEFAB  AAABAABABFGFAB
AAABAABAFCEFCD  AAABAABAFCDFBQ  AAABAABAFGGFBD  AAABAABAFDBFBF  AAABAABAFQGFBB  AAABAABAHQEFBD  AAABAABBBADBCD  AAABAABBBBABCG
AAABAABBBBDBCD  AAABAABBBBDBAB  AAABAABBBBDBAD  AAABAABBBBDBCD  AAABAABBBCABCD  AAABAABBBCBBCQ  AAABAABBBCDBBQ  AAABAABBBFBBCF
AAABAABBBBFBAQ  AAABAABBBFDBBD  AAABAADAAAAACA  AAABAADAAAAAAB  AAABAADAAAABAB  AAABAADAAAABCB  AAABAADAAADBAB  AAABAADAAADBCB
AAABAADAACABAB  AAABAADAACABCB  AAABAADAAFABHB  AAABAADAAFDBHB  AAABAADAAFEBAB  AAABAADAAFEBCB  AAABAADAAFEBFB  AAABAADBAAABCB
AAABAADBAAAGAB  AAABAADBAADBCB  AAABAADBAADGCB  AAABAADBABAGCB  AAABAADBABAGFB  AAABAADBABAGHB  AAABAADBACABCB  AAABAADBACABHB
AAABAADBACABAB  AAABAADBAFAAIB  AAABAADBAFABCB  AAABAADBAFABFB  AAABAADBAFABHB  AAABAADBAFAGAB  AAABAADBAFBBIB  AAABAADBAFBGIB
AAABAADBAFDBCB  AAABAADBAFDBFB  AAABAADBAFDBHB  AAABAADBAFDGCB  AAABAADBAFDGFB  AAABAADBAFDGHB  AAABAADBAQAACB  AAABAADBAQBBCB
AAABAADBAQBBCB  AAABAADBHAAACB  AAABAADBHQDACB  AAABAADEAADGAB  AAABAADEAADGCB  AAABAADEAAEAAB  AAABAADEAFDGHB  AAABAADEHAEACB
AAABAADEHQEACB  AAABAADGAFBGIB  AAABAADGAFEAIB  AAABAADGHAEACB  AAABAAEABAEFBD  AAABAAEABCAFBD  AAABAAEABCEBBD  AAABAAEABQAFBD
AAABAAEABQEBBD  AAABAAEAFAAABA  AAABAAEAFAAFBA  AAABAAEAFAAFBD  AAABAAEAFCAFBA  AAABAAEAFCAFBD  AAABAAEBBCABBD  AAABAAEBBGAABD
AAABAAEBBQAABD  AAABAAEBFAAABA  AAABAAEBFAAABD  AAABAAFAAAGAAB  AAABAAFAABGBHB  AAABAAFAACBBHB  AAABAAFAAFBBAF  AAABAAFAAFBBCF
AAABAAFAAFBBIB  AAABAAFAAQBBCB  AAABAAFBACFBHF  AAABAAFBAFBBIB  AAABAAFBAFBGIB  AAABAAFBAQBBCB  AAABAAFBAQBGCB  AAABAAFEAAGAAB
AAABAAFEAFBGAF  AAABAAFEAFBGCF  AAABAAFEAFBGIB  AAABAAFEAQBGCB  AAABAAFEHAGACB  AAABAAFEHQGACB  AAABAAFGAQGACB  AAABAAFGHAGACB
AAABAAGAAAAABA  AAABAAGAAAABBA  AAABAAGAAAABBD  AAABAAGABGDBBD  AAABAAGABGDBBQ  AAABAAGABGDFBD  AAABAAGABGDFBQ  AAABAAGAHAAABA
AAABAAGAHAAFBA  AAABAAGAHAAFBD  AAABAAGAHADFBA  AAABAAGAHADFBD  AAABAAGAHGDFBQ  AAABAAGBAAABBD  AAABAAGBAAAGBA  AAABAAGBABAGBD
AAABAAGBABAGBB  AAABAAGBAGAABD  AAABAAGBAGABBD  AAABAAGBAGABBG  AAABAAGBAGAGBA  AAABAAGBAQAABD  AAABAAGBBADBBD  AAABAAGBBBABBG
AAABAAGBBGDBBD  AAABAAGBBGABBG  AAABAAGBBGDBBD  AAABAAGBHAAABA  AAABAAGBHAAABD  AAABAAGBHGDABD  AAABAAGBHQDABD  AAABAAKAABEBBG
AAABAAKAABEBFB  AAABAAKBABABBG  AAABAAKBABABFB  AAABAAKBABAGBB  AAABAAKBABDBBG  AAABAAKBABDBFB  AAABAAKBAFAAFB  AAABAAKBAGAABG
AAABAAKBHGDABG  AAABAAKEHGEABG  AAABAAMAABBBBQ  AAABAAMAABDBBQ  AAABAAMAABDBHB  AAABAAMAABGBBG  AAABAAMAABGBFB  AAABAAMAABGBHB
AAABAAMAAFBBFB  AAABAAMAAFDBHB  AAABAAMAHBDAHQ  AAABAAMBABBBHQ  AAABAAMBABBGHQ  AAABAAMBABFAFQ  AAABAAMBABFBBQ  AAABAAMBAFBBFB
AAABAAMBAFBGFB  AAABAAMBAGFABQ  AAABAAMEABBGBQ  AAABAAMEAFBGFB  AAABAAMEAFBGHB  AAABAAMEHBBAFQ  AAABAAMEHBBAHQ  AAABAAMEHBDAFQ
AAABAAMEHGBABQ  AAABAAMEHGDABQ  AAABAAMBABBAFQ  AAABAAMGABBGFB  AAABAAMGAGBABQ  AAABAAMGHBBAHQ  AAABABBAAABCAD  AAABABBAAABCAF
AAABABBAAADBAD  AAABABBAAADCAA  AAABABBAAAEBAB  AAABABBAAAEBBD  AAABABBAAAGCAB  AAABABBAABEBAD  AAABABBAABEBHG  AAABABBAACBBHD
AAABABBAACBBHF  AAABABBAACBBHQ  AAABABBAACDBAD  AAABABBAACDBAQ  AAABABBAACDBHA  AAABABBAACEBAB  AAABABBAACEBBD  AAABABBABABCAF
AAABABBABAAFAF  AAABABBABADCAD  AAABABBABADFAD  AAABABBABAEBAD  AAABABBABAECAB  AAABABBABAEFAB  AAABABBABAGCAB  AAABABBABAGFAB
AAABABBABBBEAD  AAABABBABBDFAD  AAABABBABBEFAD  AAABABBABBEFAG  AAABABBABBGFAG  AAABABBABCBBAF  AAABABBABCBBAQ  AAABABBABCBBBF
AAABABBABCBFAD  AAABABBABCBFBF  AAABABBABCDBAD  AAABABBABCDFAA  AAABABBABCDFAB  AAABABBABCEBAB  AAABABBABCGBAB  AAABABBABCGBBB
AAABABBABCGFBB  AAABABBAFABCBF  AAABABBAFABFBF  AAABABBAFAGCBB  AAABABBAFAGFBB  AAABABBAFCBFBF  AAABABBAFEBFBQ  AAABABBAHAEFBD
AAABABBBAADCAD  AAABABBBACDBHD  AAABABBBACDGHD  AAABABBBBADCAD  AAABABBBBCBBHF  AAABABBBBCDBAD  AAABABBBBCDBAQ  AAABABBBFABCBF
AAABABBEAADGAD  AAABABBEACBGHD  AAABABBEACBGHF  AAABABBEACBGHQ  AAABABBEACDGAD  AAABABBEACDGAQ  AAABABBEACDGHA  AAABABBEBAEAAD
AAABABBRAADCAB  AAABABBRABDGHD  AAABABBBBADAAD  AAABABBRBADCAA  AAABABBRFADCBB  AAABABDAAABCAB  AAABABDAAADBAB  AAABABDAACBBHB
AAABABDAACDBAB  AAABABDBAADCAB  AAABABDBABBBFB  AAABABDBABBGFB  AAABABDBACDBHB  AAABABDBACDGHB  AAABABDEAADGAB  AAABABDEACBGHB
AAABABDEACDBAB  AAABABDRABDGHB  AAABABFAAABBAB  AAABABFAABBBHB  AAABABFAABGBAB  AAABABFAABGCAB  AAABABFAABGCHB  AAABABFAACBBAB
AAABABFBAAFCAF  AAABABFBAAFGAF  AAABABFEABBGAB  AAABABFEABBGHB  AAABABGEHAEABD  AAABABKAABEBFB  AAABABKBABDBFG  AAABABKBABDGFG
AAABABMAABDBFD  AAABABMAABDBHG  AAABABMAABBGFB  AAABABMAABGBHB  AAABABMEABDGFQ  AAABABMEABDGHG  AAABADEABAAFBD  AAABADEAFAAFBD
AAABADEAFCAIBA  AAABADEAFCAFBD  AAABADEAGAACBA  AAABADEHFAAFBA  AAABADEHFCAFBA  AAABADEHFEAIBA  AAABADEHQEACBA  AAABADERFAAABA
AAABADERFAAABA  AAABADKHABACBB  AAABADKHABACBG  AAABADKRHEDABG  AAABADMAABBCBQ  AAABADMAABGCBB  AAABADMAABGCBG  AAABADMAHEBABQ
AAABADKEHEDABD  AAABAFGAAGDCBD  AAABAFGAAAABBD  AAABAFGAHGEABD  AAABAFGEBGEABD  AAABAFGEBQEABD  AAABAFGHABACBD  AAABAFGHABACBG
[illegible]     AAABAFGHBBACBG  AAABAFGHBGACBG  AAABAFGHAGACBG  AAABAFGRAGDCBB  AAABAFGRAGDCBD  AAABAFGRBBACBG  AAABAFGRBGAABD
[illegible]     AAABAFGRBBDABD  AAABAFGBHGDABD  AAABAFKAAFAAFB  AAABAFKAHFEAFB  AAABAFKHABACBB  AAABAFKRABACBB  AAABAFKRABDCFB
[illegible]     AAABAFHAHBBAFD  AAABAFHEHGDABQ  AAABAHKAAFAAFB  AAABAHKBABACBB  AAABAHKBABDCFB  AAABAHKBHFDAFB  AAABAHKBHGDABG
[illegible]     AAABAHMAABDCHB  AAABAHMAABGCBG  AAABAHMAABBCHB  AAABAHMAHBDAHQ  AAABAHMBABBCHQ  AAABAHMBABFAHQ  AAABAHMBABFCBQ
[illegible]     AAABBADAAAEBCB  AAABBADAAACABB  AAABBADAADABCB  AAABBADABQEDCB  AAABBADBAAAGCB  AAABBADBABABFB  AAABBADBABABHB
[illegible]     [illegible]     [illegible]     AAABBADEHAEAHB  AAABBADBAFAGCB  AAABBADBHCAAHB  AAABBADBHFAAIB  AAABBADBHQAACB
[illegible]     [illegible]     AAABBADBRDAACB  AAABBADERCEACB  AAABBADGHCEAHB  AAABBADGHFEAIB  AAABBADGHQEACB  AAABBADGRAEDCB
[illegible]     AAABBADGRDEACB  AAABBAEAFCABBD  AAABBAEABQADBD  AAABBAEABQAFBD  AAABBAEBBAADBA  AAABBAEBBAADBD  AAABBAEBBBABBD
[illegible]     [illegible]     AAABBAERBCABBA  AAABBAERBGABBD  AAABBAERBQAABD  AAABBAFAHCGAHB  AAABBAFGHCGAHB  AAABBBDAAAABAB
[illegible]     [illegible]     [illegible]     AAABBBDBABGCAB  AAABBBDBACAAHB  AAABBBDBACABAB  AAABBBDBHQAAFB  AAABBBDBHBAAHB
[illegible]     [illegible]     [illegible]     [illegible]     AAABBBEABBAADA  AAABBBEABBAFDA  AAABBBEABGAADA  AAABBBEBBCAABD
[illegible]     [illegible]     [illegible]     AAABBBDEABAAAB  AAABBEFBEABGAF  AAABBEHBBABDAF  AAABBEHBBABDAH  AAABCADAAFEAIB
[illegible]     [illegible]     AAABCAFAAFBDIB  AAABCAFAADBDCB  AAABCAFBAFBDIB  AAABCAFBADBDCB  AAABCAFGAAGDCB  AAABCAFGACGDCB
[illegible]     [illegible]     AAABCAKAAFEAFB  AAABCAMAABBAFQ  AAABCAMEABBAFQ  AAABCAMGAABDAB  AAABCBDBAADAAB  AAABCBGEAAEABD
[illegible]     [illegible]     AAABCBGEBCJABD  AAABCDGEBBJABD  AAABCDGEBBJABG  AAABCDMEACBABQ  AAABCFGEAGEABD  AAABCFGEAGEABG
[illegible]     [illegible]     AAABCFGEBGEABG  AAABCFGEBJAABD  AAABCFGEBGDABQ  AAABCFGEBGDABQ  AAABCFGEBGEABG  AAABCFGEBGJABD
[illegible]     [illegible]     AAABCFKEABEAFB  AAABCFKEABJABG  AAABCFKEABJAFB  AAABCFMEABBABQ  AAABCFMEABGABG  AAABCFMEABGAFB
[illegible]     [illegible]     AAABDBERBAABBA  AAABDBEABAACDA  AAABDBEABEACDA  AAABDBEBBCABDA  AAABDBEBBCACDA  AAABDBEHBAACDA
[illegible]     [illegible]     [illegible]     AAABEAHAAFBDCF  AAABGAHEAQBICB  AAABFBGABBAFDA  AAABFBGHBBAFDA  AAABFBGHBCAFBA
[illegible]     [illegible]     AAABGAHEAFBGIH  AAABGAHFADBGCH  AAABGAHEAFBGIH  AAABGAHEAFBDCH  AAABGAHEADBGCH  AAABGAHGAFBDCH
[illegible]     [illegible]     AAACABGABGMCLD  AAACABGABGMCLD  AAACABGABGMCLB  AAACABGABGMCLD  AAACABGABGMFLG  AAACABGABGMFLD
AAACAFKAABDCBB  [illegible]     AAACAKKAABDCBB  AAACAKKAABDCBB  AAACAFGAABACBA  AAACAFGAABACBD  AAACAFKAABACBB  AAACAFKAABACBG
AAACDBEBBEACDA  [illegible]     AAACBBEBBGAFGD  AAACBBEBBBAABD  AAACBBEBBBAADA  AAACBBEBBEAADA  AAACBBEBBGAABD  AAACDBEBBBACDA
AAAEAAEAFBAFBD  AAAEAAKAABABBB  AAAEAAKAABABBG  AAAEAAKAABDBBB  AAAEAAKAHBMABG  AAAEAAMAABDABQ  AAAEABKEHBEAFB  AAAEAHKAABACBB
```

AAAEAHKAABACBG	AAAEAHKAABDCBB	AAAEAHKAABEAFB	AAAEAHKAHBMABG	AAAEBGEHBBAFBA	AAAEBBEHBGAFBA	AAAECAKEABMABG	AAAECAMEABDABQ
AAAFCAFQAABDAB	AAAFCBFBACBAAB	AAAFEAHGAABQAB	AAAFEAHGACBQAB	AAAFEAHQAABDAB	AAAGAAKAABDBBB	AAAGABMAABQAHB	AAAGABMAHBQAHG
AAAGABMEABQAHG	AAAGAHKAABACBG	AAAGAHMAHBGAHG	AAAGCAMEABGAHG	AAAGCBGEBBJABD	AAAGCBGRBBDABD	AAAGCBGRBBDAHA	AAAGCAGEBBQABD
AAAGCDKEABEABG	AAAGCDKEABJABG	AAAGCDMEABGABG	AAAGFBGHBBAFBA	AAAGFBGRBBACBA	AAAIAAKAABEAFB	AAAIAAKAHBEABB	AAAIAAKAHBEAFB
AAAIAHKAABEABB	AAAIAHKAHBEABG	AAAIAHKAHBEAFB	AAAICAKEABEABG	AAAICAKEABEAFB	AABAABBAAABCAA	AABAABBAAAGCAB	AABAABBABAGCGB
AABAABBACABCAA	AABAABBACABHAA	AABAABBACAECAB	AABAABBACAGCAB	AABAABBBAABCBD	AABAABBBADBBAD	AABAABBBADBBBD	AABAABBBADBCAA
AABAABBBADBCAB	AABAABBBADBCBB	AABAABBBBDBADA	AABAABBBBDBCAB	AABAABBBBDBCBB	AABAABBBBERCGA	AABAABBBBGBCGA	AABAABBBCDBAAA
AABAABBBCDBABD	AABAABBBCDBBBB	AABAABBBCEBCBA	AABAABBBCGBABD	AABAABBBCGBBBA	AABAABBBCGBCBA	AABAABBBFEBCBA	AABAABBBFGBABD
AABAABDBAAAAAB	AABAABDBABBCAB	AABAABDBADAAAB	AABAABDBADBBAB	AABAABDBCABAAB	AABAABDBCABBAB	AABAABDBCBBBAB	AABAABDBHBBAFB
AABAABDFAAABAB	AABAABDFABABFB	AABAABDFADABAB	AABAABDHADABAB	AABAABEBFDAABD	AABAABEBFDACBD	AABAABFAAABBAB	AABAABFAADBBAB
AABAABFBABBCAB	AABAABFBADBBAB	AABAABGBAAACBD	AABAABGBAEACBQ	AABAABGBAGABBD	AABAABGBHAAABD	AABAABGBHDAABD	AABAABGBHEAABD
AABAABGFAAABBD	AABAABGFADABBD	AABAABIBBBAADB	AABAABIBFDAABB	AABAABIFBBAFQB	AABAABIFBDAFGB	AABAABKBAGABBG	AABAABKBHEAABG
AABAABKFABABFB	AABABABBABBBFD	AABABABBAGBBBD	AABABABBCQBBBH	AABABABBHBAAFE	AABABABBHDAACE	AABABABBHQAABR	AABABABBHQAACR
AABABABBRFAAQR	AABABABFABABFD	AABABABFABABFE	AABABABFAGABBE	AABABABFAQABBR	AABABABFAQABCR	AABABABFEBABQD	AABABABFEBABQE
AABABABFEFABQR	AABABABQABBBFA	AABABABQABBBFD	AABABABQABEBFD	AABABABQAGBBBD	AABABABQAGEBBD	AABABABQAQBBBH	AABABABDAQEBBR
AABABABQABEBCR	AABABADFABABFB	AABABADFAFABFB	AABABADFEBABQB	AABABADFEFABQB	AABABADGHFEAIB	AABABAEBBBADBE	AABABAEBBDAAGE
AABABAEBBGAAGE	AABABAEBBGADBE	AABABAEBBGADGE	AABABAEBBQAAGD	AABABAEBBQADBD	AABABAEBBQADGD	AABABAEFBDAHGE	AABABAEFBGAHGE
AABABAEFBQAHGD	AABABAFQAFBBIB	AABABAHBAFBDIH	AABABAHQBQGFCR	AABABAHQBQGHCR	AABABAIBBBADBE	AABABAIBBDAAGE	AABABAIBBDAAGR
AABABAIBBQAAQR	AABABAIBBDADQE	AABABAIBBFAAQB	AABABAIBBGAABE	AABABANBBDAAQE	AABABANBBFAAQR	AABABANBBGAAGE	AABABANBBGADGE
AABABANBBQAAGR	AABABANQBQEHGR	AABABARQBQEFCR	AABABARQBQEHCR	AABABBDBCDBAAB	AABABBDBCDBBAB	AABABBFAAAGAAB	AABABBIBBBAAQE
AABABBIBBDAADR	AABABBIBBDAAGR	AABABBIBBDAAQE	AABABBIBBDAAQR	AABABBIBBEAADE	AABABBIBBEAAGE	AABABBIBBGAAGE	AABABBIFBDAHDR
AABABBIFBDAHGR	AABABBIFBDAHQE	AABABBIFBDAHQR	AABABBIFBEAHDE	AABABBIFBEAHGE	AABABBIFBGAHGE	AABABBRFBDAFQR	AABABGHBADCBAR
AABABGHBADCDAB	AABABGHBCDBBAB	AABABGHBCDBDAB	AABABGHBCDCBAR	AABABGHBCDCDAB	AABABGNBBDADGR	AABABGNBBDADQD	AABABGNBBDADQE
AABABGNBBDADQR	AABABGRBCDCBAB	AABABMNHBGAHGE	AABABMRHBDAHQR	AABACADAAAEDCB	AABACADAAQEACB	AABACADGAAEAAB	AABACADGAAEACB
AABACADGAAEDCB	AABACADGADEAAB	AABACADGADEDCB	AABACADGAFEDFB	AABACAFAAAGDCB	AABACAFAAFBDIB	AABACAFAAQBDCB	AABACAFBADBDCB
AABACAFGAAGDCB	AABACAFGADBDCB	AABACAFGADGDCB	AABACAFQAABDCB	AABACAFQABBDCB	AABACAGAAQEABD	AABACBGAADEABD	AABADBHAAACBAR
AABADBHBADCBAR	AABADBIBBBACQE	AABADBIBBEACDE	AABAEAHAAAGDCB	AABAEAHAAQBDCB	AABAEAHBADBDCB	AABAEAHGAAGDCB	AABAEAHGADBDCB
AABAEAHGADGDCB	AABAEAHQAABDCB	AABAEAHQABBDCB	AABBAAEBBGAABD	AABBAAEBBQAABD	AABBAAEBFAAABD	AABBAAIBBGAABE	AABBAAIBBGAABG
AABBAAIBBGABBE	AABBAAKAAFABFB	AABBABDBAADBAB	AABBABDBABDBHB	AABBADERQEACBE	AABBADIHBEACBB	AABBAFGHAGACBE	AABBAFGHAGACBG
AABBAFGHCGACBE	AABBAFGHCGACBG	AABBAFGHCGAHBG	AABBBAEBBAADBD	AABBBAEBBBAABE	AABBBAEBBBABBD	AABBBAEBBBABBE	AABBBAEBBGAABE
AABBBAEBBGABBD	AABBBAEBBGABBE	AABBBAEBBQAABD	AABBBAHBBAFAAH	AABBBAIBBBAABE	AABBBAIBBBABBB	AABBBAIBBBABBE	AABBCAFAAAGDCB
AABBCAFGAAGDCB	AABBCAGBAADABD	AABBCAGBBADABD	AABBCAGBBGDABG	AABBCDKECBEABG	AABBCFGEAGEABG	AABCABGAAJMBDD	AABCAGIBCDABDB
AABCBBEBBDAAGD	AABCBBEBBGAAGD	AABCBBEFBDAHGD	AABCBBEFBGAHGD	AABCBGIBCDABDB	AABDBEHBAACDAB	AABDBEHBBACDAB	AABDBEHBBACDAH
AABFBBHBBABAAB	AABFBBHBBABAAH	AABFBBHBBACAAB	AABFBBHBBACAAH	AABGADIHCDACBE	AAGCBBIBBDAAGE	AAGCBBIFBDAHGE	ABBACAFBABBDCB

DRAWINGS WITH 27 CROSSINGS:

AAAAAAAAAABAAF	AAAAAAAAABGBFG	AAAAAAAAABABAGF	AAAAAAAAABABFAF	AAAAAAAABBAFAG	AAAAAAAAFBGAFB	AAAAAAAAFBGFBB	AAAAAABAAAAACD
AAAAAABAAAABAA	AAAAAABAAAABAD	AAAAAABAAAABCA	AAAAAABAAAABCD	AAAAAABAAQAACD	AAAAAABAAQEBBD	AAAAAABAAQEBCB	AAAAAABABQBFGF
AAAAAABABQGFGB	AAAAAABAFQBFBF	AAAAAABAFQGFBB	AAAAAABAHQEFBD	AAAAAABBAAABCD	AAAAAABBAAAGAA	AAAAAABBAAAGBD	AAAAAABBAAAGCB
AAAAAABBABAGAD	AAAAAABBABAGAG	AAAAAABBABAGCA	AAAAAABBABAGFA	AAAAAABBABBGFD	AAAAAABBAFAGAB	AAAAAABBABAGBA	AAAAAABBAGBGBD
AAAAAABBAQABBD	AAAAAABBAQBBCD	AAAAAABBAQBGBD	AAAAAABBAQBGBF	AAAAAABBAQBGCF	AAAAAABBBQBBGF	AAAAAABFBBAFAG	AAAAAABFBBAFCG
AAAAAABFBFAFAB	AAAAAABFBFAFCB	AAAAAABFBGAFGD	AAAAAABFBQAFCB	AAAAAADAAAAACB	AAAAAADAAAABAB	AAAAAADAAAABCB	AAAAAADAAAEACB
AAAAAADAAFAAIB	AAAAAADAAFEAIB	AAAAAADAAFEBAB	AAAAAADAAFEBCB	AAAAAADAAFEBFB	AAAAAADAAQAACB	AAAAAADAAQEACB	AAAAAADAHAAACB
AAAAAADAHAEACB	AAAAAADAHFAAIB	AAAAAADAHFEAIB	AAAAAADAHQAACB	AAAAAADAHQEACB	AAAAAADBAAAAAB	AAAAAADBAAAACB	AAAAAADBAAABCB
AAAAAADBABAGAB	AAAAAADBABBGFB	AAAAAADBAFABCB	AAAAAADBAFABFB	AAAAAADBAFBBIB	AAAAAADBAFBGFB	AAAAAADBAQBBCB	AAAAAADBHFAACB
AAAAAADBHFAAFB	AAAAAADFABABAB	AAAAAADGAAEAAB	AAAAAADGAAEACB	AAAAAADGABBGFB	AAAAAADGAFBGFB	AAAAAADGHFEACB	AAAAAADGHFEAFB
AAAAAAEABAAAGD	AAAAAAEABAAFGA	AAAAAAEABAAFGD	AAAAAAEABAEAGD	AAAAAAEABGAAGD	AAAAAAEABGEAGD	AAAAAAEABGEBGD	AAAAAAEABGEFBD
AAAAAAEABGEFGD	AAAAAAEABQAAGD	AAAAAAEABQEAGD	AAAAAAEABQEBGD	AAAAAAEABQEFBD	AAAAAAEAFAAABD	AAAAAAEAFAAFBA	AAAAAAEAFAAFBD
AAAAAAEAFAEABD	AAAAAAEAFGEFBD	AAAAAAEAFQAABD	AAAAAAEAFQEABD	AAAAAAEAGAABGA	AAAAAAEBBGAABD	AAAAAAEBBGAAGA	AAAAAAEBBGAAGD
AAAAAAEBBGABBD	AAAAAAEBBQAABD	AAAAAAEBBQABBD	AAAAAAEBFGAABA	AAAAAAEFBAABGD	AAAAAAEFBAAFBD	AAAAAAEFBBAFBD	AAAAAAEFBGABGD
AAAAAAEFBGAFBD	AAAAAAKAABAAFG	AAAAAAKAABEAFG	AAAAAAKAABEBBG	AAAAAAKAABEBFB	AAAAAAKAAFAAFB	AAAAAAKAAFEAFB	AAAAAAKAAGEABG
AAAAAAKAHBEAFG	AAAAAAKAHFEAFB	AAAAAAKBABABBG	AAAAABBAAAABAD	AAAAABBAAABCAA	AAAAABBAAABCAF	AAAAABBAAAEAAB	AAAAABBAAAEBAB
AAAAABBAAAEBBD	AAAAABBAAAECBB	AAAAABBAAABCBB	AAAAABBAABEBAD	AAAAABBAABEBFD	AAAAABBAABECAG	AAAAABBAABECFD	AAAAABBAAGEBBD
AAAAABBABABCAF	AAAAABBABABFGF	AAAAABBABAECAB	AAAAABBABAECGB	AAAAABBABAECGD	AAAAABBABAEFAD	AAAAABBABAGCAB	AAAAABBABAGFGB
AAAAABBAFABFBF	AAAAABBAFAGFBB	AAAAABBAFBGCFG	AAAAABBAHQEFBD	AAAAABBAHGEFBD	AAAAABBBAABCAD	AAAAABBBABBBFD	AAAAABBBABBCFD
AAAAABBAGBBBD	AAAAABBAGBCBD	AAAAABBBBGBBBD	AAAAABDAAAABAB	AAAAABDAABEBAB	AAAAABDAABEBFB	AAAAABDAABECFB	AAAAABDBAAAAAB
AAAAABDBAABCAB	AAAAABDBABAGAB	AAAAABDBABBBFB	AAAAABDBABBCFB	AAAAABDFABABAB	AAAAABDHABABAB	AAAAABDHABABFB	AAAAABDHABACFB
AAAAABEABAEAGD	AAAAABEABGECBD	AAAAABEABGEFGD	AAAAABEAFAEABD	AAAAABEBBAAAGD	AAAAABEBBAACGD	AAAAABEBBEAAGD	AAAAABEBBEACGD
AAAAABEBBGAAGD	AAAAABEBBGACGD	AAAAABEBFAAABD	AAAAABEBFAACBD	AAAAABEFBGABGD	AAAAABEFBGAFBD	AAAAABEHBGACGD	AAAAABGHBGAFGD
AAAAABEHBAACBA	AAAAABFAAABAAB	AAAAABFAAABBAB	AAAAABFAABBCAB	AAAAABFAABGBAB	AAAAABFBAABCAB	AAAAABFBAABGAF	AAAAABFBAAFAAF
AAAAABGAAAACBD	AAAAABGAAAEABD	AAAAABGAAEACBD	AAAAABGAAGEBBD	AAAAABGAHAEABD	AAAAABGAHGEFBD	AAAAABGBAAACBD	AAAAABGBABAGBD
AAAAABGBAEACBD	AAAAABGBAGABBD	AAAAABGBABAGBB	AAAAABGBHAAABD	AAAAABGBHEAABD	AAAAABGFABABBD	AAAAABGHAAACBA	AAAAABGHAEACBG
AAAAABKAABEBFB	AAAAABKAABECFB	AAAAABKAAEACBB	AAAAABKBABABFG	AAAAABKBABACFG	AAAAABKBABAGBB	AAAAABKBABAGBB	AAAAABKBAEACBG
AAAAABKBHBAAFG	AAAAABKBHEAABG	AAAAABKFABABBB	AAAAABKFABABBG	AAAAABKHABABFB	AAAAABKHABACFB	AAAAADEABAEABD	AAAAADEABBEFBD
AAAAADEAFAAABA	AAAAADEAFAAABD	AAAAADEAFAACBD	AAAAADEAFAAFBD	AAAAADEAFBEFBD	AAAAADEAFEAIBA	AAAAADEAFEACBA	AAAAADEHFAACBA
AAAAADEHFAAFBA	AAAAADGAAEAABA	AAAAADGAAEACBA	AAAAADGAAEACBB	AAAAADGABAEFBD	AAAAADBAHBEFBD	AAAAADKAHFEAAB	AAAAAEFBAABBAB
AAAAAFKAABACBG	AAAAAFKAABACFB	AAAAAFKAABEABG	AAAAAFKAABEAFB	AAAAAFKAHFAAFB	AAAAAFKAHGAABG	AAAABABAGAAAAA	AAAABABBABAGFA
AAAABABBAFAGFB	AAAABABBAGAGBA	AAAABABBAQAGBB	AAAABABBAQAGCB	AAAABABFAFABFB	AAAABABFAGABBA	AAAABABFAQABBB	AAAABABFAQAACB
AAAABABGADBGBF	AAAABABGADBGBH	AAAABABQADBBBF	AAAABABQADBBGH	AAAABABQAQBQBH	AAAABABQBQBFGF	AAAABABQFQBFGF	AAAABABQFQBFGH
AAAABADAAFEBIB	AAAABADAAQEBCB	AAAABADBAAAGCB	AAAABADBABAGCB	AAAABADBABBGFB	AAAABADBAFABIB	AAAABADBAFBDIB	AAAABADBAFAGCB
AAAABADBAQABCB	AAAABADBAQADCB	AAAABADEEBAGAB	AAAABADHEFAGAB	AAAABADHFAAIB	AAAABADBHQAACB	AAAABADBHQADCB	AAAABADFAAAACB
AAAABADFABABCB	AAAABADFAFABCB	AAAABADFEBAAAB	AAAABADFEFABAB	AAAABADGABBGFB	AAAABADBAFEDIB	AAAABADGAQEDCB	AAAABADGHFEAIB
AAAABADGHDEACB	AAAABADGRBEDCB	AAAABADQAAEBCB	AAAABADQABBBFB	AAAABADHABEBCB	AAAABADQABEDFB	AAAABADHAFEBCB	AAAABADQAFEDFB
AAAABADEHEBAAB	AAAABADEFEBAB	[illegible]	AAAABAEABAEFGD	AAAABAEBBQAAGD	AAAABAEBBQADBD	AAAABAEBBQADBD	AAAABAEFBAAFGD
AAAABAEFBGAFBD	AAAABAEFBGAFBD	[illegible]	AAAABAEFBQABBD	AAAABAFAAFBBIF	AAAABAFAAAGBCF	AAAABAFAAAGBCB	AAAABAFBAFBDIE
[illegible]	AAAABAFGAFBGIB	[illegible]	AAAABAFBAQBDCB	AAAABAFGHDBACB	AAAABAFBBQBDCF	AAAABAFBBQBDCB	AAAABAFQAABBCF
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AAAABAHAAFBDIH
[illegible]	AAAABAHGAFBAIB	[illegible]	AAAABAHBAQABCB	AAAABAHBBQBDCF	AAAABAHBBQBDCH	AAAABAHGBQBDCB	AAAABAHQAAGDCH
AAAABAHAAFBDCH	AAAABAHQBBGFCB	AAAABAHQBFBFCF	AAAABAHQBFGFCB	AAAABAHQBQBBCF	AAAABAHQBQBDCF	AAAABAHQBQBDCH	AAACBAHQBGBFCH

```
AAAABAHQBQGBCB AAAABAHQBQGDCB AAAABANBBQADCD AAAABANFBQABCD AAAABANQBQEBCD AAAABANQBQEDCD AAAABBBAAAAAAA AAAABBBAAAAAAD
AAAABBBAAACAAD AAAABBBAAACBBH AAAABBBAABBBFD AAAABBBAABCAFA AAAABBBAABCAFD AAAABBBAABCBFA AAAABBBAABEBAD AAAABBBAAGBBBD
AAAABBBAEACBGH AAAABBBAEAEBGD AAAABBBAEBCBAD AAAABBBAHBCAFA AAAABBBAHBCAFD AAAABBBAHGBFBD AAAABBBBABABFD AAAABBBBABAGAD
AAAABBBBABAGBD AAAABBBBABAGFA AAAABBBBABAGFB AAAABBBBABBGBD AAAABBBBABBGFB AAAABBBBABCBBD AAAABBBBAECABA AAAABBBBAGABBD
AAAABBBBAGAGBA AAAABBBBAGBGBA AAAABBBBAGCABA AAAABBBBAGCBBA AAAABBBBBGBBBD AAAABBBBBGBBGD AAAABBBBEBBGAD AAAABBBBEBCBAD
AAAABBBBEBCGAA AAAABBBBEBCGAB AAAABBBBHGAABD AAAABBBBHGBABD AAAABBBFAGABBA AAAABBBFBGAFGD AAAABBDAAAAAAB AAAABBDAAACAAB
AAAABBDBAAAGAB AAAABBDBAACGAB AAAABBDBABABFB AAAABBDBABAGAB AAAABBDBEAAGAB AAAABBDBEACGAB AAAABBDBEBBGAB AAAABBDBEBCBAB
AAAABBDBHBAAFB AAAABBDBHBBAFB AAAABBDHABABAB AAAABBEFBAAFGD AAAABBEFBBAFBD AAAABBEFBGAFBD AAAABBEHBAAFGD AAAABBEHBGAFGD
AAAABBFAABGBAB AAAABBFAEBBBAB AAAABBFBAAFGAF AAAABBFBEAFGAF AAAABBFBEBBGAB AAAABBHAABCAAB AAAABBHABABAAF AAAABBHABABAAH
AAAABBHABABFAH AAAABBHABACFAH AAAABBHABAGAAB AAAABBHABBBFAB AAAABBHABBCFAB AAAABBHAGBBBAB AAAABBHAGBCBAB AAAABBHBAACGAH
AAAABBHBAAFAAH AAAABBHBABCGAB AAAABBHBGAFGAF AAAABBHBGBBGAB AAAABBNABACAAH AAAABBNABAEAAD AAAABBNABGCAGA AAAABBNBBGAAGD
AAAABBNFBGABGD AAAABBNFBGCBGA AAAABBNHBAAFGD AAAABDEABAEABD AAAABEFBAABGAF AAAABEFBAAFBAF AAAABEFBAAFDAF AAAABEFGAABGAB
AAAABEFQAABBAB AAAABEFQEABDAF AAAABEFQRABDAF AAAABEHBAACGAB AAAABEHBAAFDAF AAAABEHBBAFDAF AAAABEHBBAFDAH AAAABEHBGAFGAH
AAAABEHGBABDAF AAAABEHGBABDAH AAAABEHQAACBAB AAAABEHQBABBAF AAAABEHQBABDAF AAAABEHQBABDAH AAAABENBBAADAD AAAABENQBAFDAH
AAAABGHAAACBAB AAAABGHABABBAF AAAABGHABABDAF AAAABGHABABDAH AAAABGHABAGBAB AAAABGHABAGDAB AAAABGHBAAFDAH AAAABGNABACDAH
AAAABGNABAEBAD AAAABGNABAEDAD AAAABGNBBACDGD AAAABKNABACDCH AAAABKNABAEBCD AAAABKNABAEDCD AAAACADAAFEAAB AAAACADAAFEDAB
AAAACADAAFEDCB AAAACADAAFEDFB AAAACADBAFBDIB AAAACADBAQBDCB AAAACADGAAEAAB AAAACADGAAEACB AAAACADGAAEDCB AAAACADGABBDFB
AAAACADGABEDAB AAAACADGAFBDFB AAAACADGAFEDCB AAAACADGAFEDFB AAAACADQABBDFB AAAACADQABEDAB AAAACADQAFBDFB AAAACAFAAABAAB
AAAACAFAAFBDAF AAAACAFAAFBDCF AAAACAFAAFBDIB AAAACAFAAQBDCB AAAACAFBAFBDIB AAAACAFBAQBDCB AAAACAFGAAGACB AAAACAFGAAGDCB
AAAACAFGAFBDIB AAAACAFGAQBDCB AAAACAFQAABDCB AAAACAFQABBDCB AAAACAFQAFBDCB AAAACBDBAABAAB AAAACBFBAABAAB AAAADANBBGCBGA
AAAADANFBGCBGA AAAADBNBBGCBGA AAAADBNFBGCBGA AAAADGNABACBAH AAAAGEHGAABGAB AAABAABAAAAAAA AAABAABAAAEBAD AAABAABAAAEBCD
AAABAABAACABHD AAABAABAACDBAQ AAABAABAACDBCQ AAABAABAACDBHD AAABAABABBDFAQ AAABAABABCBFAQ AAABAABABCDFAA AAABAABABCDFAD
AAABAABABCEFAD AAABAABBBBDBCG AAABAABBBBDBCQ AAABAABBBCDBAQ AAABAABBBCDBCD AAABAADAAAEBAB AAABAADAAAEBCB AAABAADAACABHB
AAABAADAACEBAB AAABAADAACEBCB AAABAADAAFABIB AAABAADAAFDBIB AAABAADAAFEBHB AAABAADAAQABCB AAABAADAAQDBCB AAABAADBAAAGCB
AAABAADBACAGCB AAABAADBACAGHB AAABAADBAFAGCB AAABAADBAFAGFB AAABAADBAFAGHB AAABAADBHQAACB AAABAADEAFDGIB AAABAADEAQDGCB
AAABAADGHQEACB AAABAAEABCEFBD AAABAAEABQEFBD AAABAAEAFAEFBD AAABAAEAFCEFBD AAABAAEAFQAFBD AAABAAEBBGABBD AAABAAEBBQABBD
AAABAAEBFQAABD AAABAAFAAABBAF AAABAAFAAABBCF AAABAAFAAAGBAB AAABAAFAAAGBCB AAABAAFAACBBAF AAABAAFAACBBCF AAABAAFAACGBAB
AAABAAFAACGBCB AAABAAFBAFBBIF AAABAAFBAFBGIF AAABAAFBAQBBCF AAABAAFBAQBGCF AAABAAFEAABGAF AAABAAFEAABGCF AAABAAFGAFBGIB
AAABAAFGAQBGCB AAABAAFGHQGACB AAABAAGAAAEBBD AAABAAGAAGABBD AAABAAGAAQABBD AAABAAGAHAEFBD AAABAAGAHGAFBD AAABAAGAHGDFBD
AAABAAGAHQAFBD AAABAAGAHQDFBD AAABAAGBAAAGBD AAABAAGBAGAGBD AAABAAGBAGAGBG AAABAAGBBBDBBG AAABAAGBBBDBBQ AAABAAGBBGABBQ
AAABAAGBBGDBBG AAABAAGBBGDBBQ AAABAAGBBQABBD AAABAAGBHGAABD AAABAAGBHQAABD AAABAAKAAFABFB AAABAAKAAFDBFB AAABAAKAAGABBG
AAABAAKAAGDBBG AAABAAKBABAGBG AAABAAKBABAGFB AAABAAKBHGAABG AAABAAMAAFDBFB AAABAAMAAGDBBG AAABAAMBABBBFQ AAABAAMBABBGFQ
AAABAAMBABDBFG AAABAAMBABDBFQ AAABAAMBABDBHG AAABAAMBABDBHQ AAABAAMBABFBHQ AAABAAMBABFGBQ AAABAAMBAFDBFB AAABAAMBAFDBHB
AAABAAMBAFDGFB AAABAAMBAFDGHB AAABAAMBAGBBBQ AAABAAMBAGBGBQ AAABAAMEHBDAHQ AAABAAMGABBGBQ AAABAAMGAFBGFB AAABAAMGHBBAFQ
AAABAAMGHGBABQ AAABABBAAADCAD AAABABBAAAEAAD AAABABBAAAECAB AAABABBAABEBHD AAABABBAACDBHD AAABABBAACDBHQ AAABABBAACEBHB
AAABABBABAECAD AAABABBABAEFAD AAABABBABCBFAF AAABABBABCBFAQ AAABABBABCDBAQ AAABABBABCDFAD AAABABBABCEBAD AAABABBABCEFAB
AAABABBABCGFAB AAABABBEAAEAAD AAABABBEACDGHD AAABABBEACDGHQ AAABABBRAADGAD AAABABBRBADCAB AAABABDAAADCAB AAABABDAACDBHB
AAABABDEAAEAAB AAABABDEACDGHB AAABABDRAADGAB AAABABEAFAEFBD AAABABEAFCEFBD AAABABFAAABCAB AAABABFAAAGAAB AAABABFAABGBHB
AAABABFAACBBHB AAABABFEAAGAAB AAABABGAAAEBBD AAABABGAHAEFBD AAABABMAABBBFQ AAABABMAABBBHQ AAABABMAABDBHQ AAABABMAABGBFG
AAABABMAABGBHG AAABABMEABBGFQ AAABABMEABBGHQ AAABABMEABDGHQ AAABADEABBEFBD AAABADEABCAFBD AAABADEAFAAIBD AAABADEHFAAIBA
AAABADEHQAACBA AAABADERBAAABD AAABADERBAACBD AAABADERFAAABD AAABADKAAEACBG AAABADKAAEDCBG AAABADKRAEAABG AAABADKRHEAABG
AAABADMAAEDCBG AAABADMEHEBABQ AAABADMRAEDCBG AAABADMRAEDCBQ AAABAFGAAGDCBD AAABAFGAAQEABD AAABAFGEAGEABD AAABAFGEAQEABD
AAABAFGEHGEABD AAABAFGEHQEABD AAABAFGHAGACBG AAABAFGHBGACBD AAABAFGRAGAABD AAABAFGRBGACBD AAABAFGRBGDCBG AAABAFGRBGDCBQ
AAABAFGRHGAABD AAABAFKAAGDCBG AAABAFKEAGEABG AAABAFKEHBEAFG AAABAFKEHFEAFB AAABAFKEHGEABG AAABAFKHABACBG AAABAFKRAGAABG
AAABAFKRHGAABG AAABAFMAABBCBQ AAABAFMAABGCBG AAABAFMAAGDCBG AAABAFMEHBBAFQ AAABAFMEHBGAFG AAABAFMEHGBABQ AAABAFMRAGDCBG
AAABAFMRAGDCBQ AAABAHKAAGDCBG AAABAHKBAFAAFB AAABAHKBAGAABG AAABAHKBHFAAFB AAABAHKBHGAABG AAABAHMAAGDCBG AAABAHMBABBCFQ
AAABAHMBABDCFG AAABAHMBABDCFQ AAABAHMBABDCHG AAABAHMBABDCHQ AAABAHMBABFAFQ AAABAHMBABFCHQ AAABAHMBAGBCBQ AAABAHMBAGDCBG
AAABAHMBAGDCBQ AAABAHMBAGFABQ AAABAHMGHBBAFQ AAABAHMGHGBABQ AAABBADAACEBCB AAABBADAAQEBCB AAABBADARCEDCB AAABBADBABAGFB
AAABBADBABAGHB AAABBADBACABHB AAABBADBACAGCB AAABBADBAFABIB AAABBADBAFAGFB AAABBADBAFAGHB AAABBADBAQABCB AAABBADBRBADCB
AAABBADBRFADCB AAABBADEHCEAHB AAABBADGRBEDCB AAABBADGRFEDCB AAABBAEABCADBD AAABBAEABCAFBD AAABBAEBBBADBA AAABBAEBBBADBD
AAABBAEBBCABBD AAABBAEBBGADBA AAABBAEBBGADBD AAABBAEBBQABBD AAABBAFEHCGAHB AAABBBDAACABAB AAABBBDAEAABAB AAABBBDBABABFB
AAABBBDBABABHB AAABBBDBACAGAB AAABBBDBEAABAB AAABBBDBEBABAB AAABBBDBEBBBAB AAABBBDBEBBGAB AAABBBDBECBBAB AAABBBDBECBBAB
AAABBBDBHCAAHB AAABBBDEAAEAAB AAABBBDEACEAAB AAABBBDEHBEAHB AAABBBDHABABAB AAABBBEABCAADA AAABBBEHBAAFBD AAABBBEHBCAFBA
AAABBBFAAAGAAB AAABBBFAABGBAB AAABBBFAACBBAB AAABBBFAACGAAB AAABBBFBACFGAF AAABBBFBEAFGAF AAABBBFBEBBGAB AAABBBFBECBGAB
AAABBBFEACBGAB AAABBDEABAAFBD AAABBDEABAEABD AAABBDEABCAABD AAABBEHBGAFGAH AAABCAFAAAGDCB AAABCAFAACBDCF AAABCAFAACGDCB
AAABCDGEACEABD AAABCDGEACJABD AAABCDGEBAEABD AAABCDGEBCEABD AAABCDMEACGABB AAABCFGEBBJABG AAABCFGEBGEABD AAABCFGEBGJABG
AAABCFGEBQEABD AAABCFKAAGDABG AAABCHKAAGDABG AAABCHMAACGABB AAABCHMBABDAFQ AAABCHMBABDAHG AAABCHMBABDAHG AAABDAEBBGABBA
AAABDBEABCABDA AAABDBEABCACDA AAABDBEHBCABDA AAABDBEHBCACDA AAABDBERBAACDA AAABDBERBEACDA AAABEAHAACBDCF AAABEAHEAAGDCB
AAABEAHEACGDCB AAABEAHGAAGDCB AAABEAHGACGDCB AAABEBHEACGAAB AAABGAHBAFBDIH AAABGAHBAQBDCH AAABGAHGAFBGIH AAABGAHGAFBDCH
AAABGAHGAQBGCH AAACABEABEMAGD AAACABEABEMCLQ AAACABEABEMFLQ AAACABEABGMCGD AAACABEABGMFLQ AAACABGABGDCLB AAACABGABGDFLG
AAACADEAFEAIBA AAACADMAABDCBB AAACAFMAABDABQ AAACAFMAABDCBB AAACBBEABGAALA AAACBBEABGAALD AAACBBEBBGAAHD AAACBBEFBBAFDA
AAACBBNABGAAGA AAACBBNABGAAGD AAADBENBBACBAD AAADBENBBACDAA AAADBENBBACDAH AAAEAAMAABDBBB AAAEAHKAABMABB AAAEAHMAABDABQ
AAAEAHMAABDCBB AAAEBBEABGAALA AAAEBBEABGAALD AAAEBBEABGABLD AAAEBBEABGAFGD AAAEBBEHBGABGD AAAEBBEHBGAFGA AAAEBBNABGAAGA
AAAEBBNABGAAGD AAAEBBNABGABGD AAAECAMAABDABQ AAAEDBEABGACLA AAAFBBNBBACAAQ AAAFBBNBBACAAH AAAFBGNBBACBAD AAAFBGNBBACDAA
AAAFBGNBBACDAH AAAFCBFBAABAAB AAAFDGNBBACBAA AAAFDGNBBACBAH AAAGABMEHBDAHG AAAGAHMAABGAHB AAABCAMAABDAHG AAAGCAMAABGAHB
AAAGCDGEBBDABQ AAAGCDGEBBEABD AAAGCDGRBBDABB AAAIAHKAABEAFB AABAABBAAAACAD AABAABBAAABCAD AABAABBAAAECAB AABAABBBAABCAD
AABAABBBABBBFD AABAABBBADBCBD AABAABBBAGBBBD AABAABBBAGBCBD AABAABBBBDBCQA AABAABBBCDBAAD AABAABBBCDBBAD AABAABBBCDBBBD
AABAABDBAAACAB AABAABDBAABCAB AABAABDBABBBFB AABAABDBABBCFB AABAABDBADACAB AABAABDBCBBAQB AABAABDBCBBCAB AABAABDBCDBAAB
AABAABDBCDBBAB AABAABDHAAABAB AABAABFAAAGAAB AABAABFAAAGBAB AABAABFBAABCAB AABAABGFAGABBD AABAABIFBBAFDB AABAABKFABABFB
AABAABKFAGABBG AABAAEFBAABBAB AABAAGIBQDABDB AABABABBCBBBFD AABABADFAFABIB AABABADFAQABCB AABABADQAFEBIB AABABAHQAFBBIH
AABABAHQAQGBCR AABABAIBBBAAQE AABABAIBBBADQE AABABAIBBDAAQE AABABAIBBFAAQR AABABAIBBGAAGE AABABAIBBGADKE AABABAIBBGADGE
AABABBRBBDAAQR AABABBRFBDAHQR AABABGRBBDADQR AABABGRBCDCBAR AABACADAAFEDIB AABACADAAQEDCB AABACADGAQEACB AABACADQAAEDCB
AABACADQADEDCB AABACADQAFEDFB AABACAFAAQGDCB AABACAFBAFBDIB AABACAFBAQBDCB AABACAFGAQBDCB AABACAFQAAGDCB AABACAFQADBDCB
AABACAFQADGDCB AABACBDACABAAB AABACBGAAAEABD AABADBHAAACBAH AABADMRACDCBQR AABAEAHAADGDCB AABAEAHBAQBDCB AABAEAHGAQBDCB
AABAEAHQAAGDCB AABAEAHQADBDCB AABAEAHQADGDCB AABBAAEBBGABBD AABBAAEBBQABBD AABBAAEBFQAABD AABBAAIBBGABBG AABBAAIBCGAABG
AABBAAIBFGAABG AABBAAKAAFDBFB AABBAAKAAFEBFB AABBABDBAADCAB AABBABDBABBBFB AABBABDBCADAAB AABBABFAAAGBAB AABBADEKFAAABD
AABBADIHBEAHBG AABBADIHCEACBE AABBADIHCEACBG AABBADIHQEACRE AABBADIRBEAABG AABBADIRBEACBG AABBADIRFEAABB AABBADIRFEACBG
AABBADIRQEACBE AABBADKRAEAABG AABBADKRHEAABG AABBAEFBAABBAB AABBAFGHCGAHBE AABBAFIHCGACBE AABBAFIHCGACBG AABBAFIRGGAABG
AABBAFIRBGACBG AABBAFIRFGAABE AABBAFIRFGAABG AABBAFIRQGACBE AABBAFKEAGEABG AABBAFKRAGAABG AABBAFKRHGAABG AABBAHIBBGAABG
AABBAHIBBGACBG AABBAHIBFGAABG AABBAHIBQGACBE AABBAHKBAGAABG AABBAHKBHGAABG AABBBAEBGBADBD AABBBAEBBGADBD AABBBAEBBQABBD
AABBBAIBBBADBB AABBBAIBBGAABE AABBBAIBBGABBE AABBBAIBCBAABE AABBBAIBCBABBE AABBCAGBAGDABD AABBCAGBAQDABD AABBCAGBBGDABD
```

```
AABBCABBBQDABD AABBCAGBCGDABD AABBCAGBCGDABG AABBCDGEAADABD AABBCDGEBADABD AABBCDGEBAEABD AABBCFGEAGJABG AABBCFGECGEABG
AABBCFKAAGDABG AABBCFKEAGDABG AABBCFKECBEABB AABBCHKAAGDABG AABCAGIBCDABQB AABCAGIBQDABDB AABCBGIBCDABDR AABDBEHBAACDAH
AABDBEHBBACDAR AABDBEHBCACDAB AABDBENBBACDAA AABDBENBBACDAH AABFBBHBAAFAAH AABFBBHGBACAAR AABFBBHBBAFAAH AABFBBHBCABAAB
AABFBBHBCACAAB AABFBBNBBACAAA AABFBBNBBACAAH AABFBGNBBACDAA AABFBGNBBACDAH AABFDGNBBACBAA AABGADIHCDACBB AADAHBKBCBACBB
AAGCBBIBBEAAGE AAGCBBIBBGAAGE AAGCBBIFBEAHGE AAGCBBIFBGAHGE ABAAGELQDABDAD ABBECHMAABDABB
```

DRAWINGS WITH 29 CROSSINGS:

```
AAAAAAAAAAAAAB AAAAAAAABABFGF AAAAAAAABBGFAG AAAAAAAABFBFAF AAAAAABAAAAAAD AAAAAABAAAEBAD AAAAAABAAAEBCD AAAAAABAAQABCD
AAAAAABBAAAGAD AAAAAABBAAAGCA AAAAAABBABAGCD AAAAAABBABAGCG AAAAAABBABAGFD AAAAAABBABAGFG AAAAAABBAFAGCB AAAAAABBAFAGFB
AAAAAABBAGAGBD AAAAAABBAGAGBG AAAAAABBAQAGBB AAAAAABBAQBGCD AAAAAABFBQAFCD AAAAAABFBQAFGB AAAAAABFBQAFGD AAAAAADAAAAAAB
AAAAAADAAAEAAB AAAAAADAAAEBAB AAAAAADAAAEDCB AAAAAADAAFABIB AAAAAADAAQABCB AAAAAADBAAAGAB AAAAAADBABAGCB AAAAAADBABAGFB
AAAAAADBAFAAIB AAAAAADBAFAGAB AAAAAADBAFBGIB AAAAAADBAQAACB AAAAAADBAQBGCB AAAAAADBHAAACB AAAAAADFAAABAB AAAAAADFABABCB
AAAAAADFABABFB AAAAAADFAFABAB AAAAAADGAFBGIB AAAAAADGAFEAIB AAAAAADGAQBGCB AAAAAADGAQEACB AAAAAADGHAEACB AAAAAAEABAAAGA
AAAAAAEABAEFGD AAAAAAEABGAFGD AAAAAAEABQAFGD AAAAAAEAFAAABA AAAAAAEAFAEFBD AAAAAAEAFQAFBD AAAAAAEBBAAAGA AAAAAAEBBAAAGD
AAAAAAEBBGABGD AAAAAAEBBQABGD AAAAAAEBFAAABA AAAAAAEBFAAABD AAAAAAEFBAAFGA AAAAAAEFBGABBD AAAAAAEFBGAFGA AAAAAAEFBQABBD
AAAAAAEFGAABGA AAAAAAKAABABFG AAAAAAKAAFABFB AAAAAAKAAGABBG AAAAAAKBABAGBB AAAAABBAAAAAAA AAAAABBAAAAAAD AAAAABBAAABCAD
AAAAABBAAABCAF AAAAABBAAAECBD AAAAABBAAAGCAB AAAAABBAABECAD AAAAABBABABCGF AAAAABBABAECAD AAAAABBABAGCGB AAAAABBAFABCBF
AAAAABBAFAGCBB AAAAABBBABBGFD AAAAABBBAGBGBD AAAAABBBBGBCGD AAAAABBBFGBCBD AAAAABDAAAAAAB AAAAABDAAABCAB AAAAABDAABECAB
AAAAABDBAAACAB AAAAABDBAAAGAB AAAAABDBABAGFB AAAAABDBABBGFB AAAAABDFAAABAB AAAAABDFABABFB AAAAABDHABACAB AAAAABEABAECGD
AAAAABEABAEFGD AAAAABEABGEFGD AAAAABEAFAEFBD AAAAABEFBAAFGD AAAAABEFBEAFGD AAAAABEFBGAFGD AAAAABEHBAACGD AAAAABEHBAAFGD
AAAAABEHBEACGD AAAAABEHBEAFGD AAAAABEHBGAFGD AAAAABFAABGCAB AAAAABFBAABCAF AAAAABFBAAFCAF AAAAABFBAAFGAF AAAAABGAAAEBBD
AAAAABGAAGEBBD AAAAABGAHAEFBD AAAAABGAHGEFBD AAAAABGBAAAGBD AAAAABGBAEAGBD AAAAABGBAGAGBD AAAAABGFAAABBD AAAAABGFAEABBD
AAAAABGFAGABBD AAAAABGHAAABBD AAAAABGHAEABBD AAAAABKAABEBFG AAAAABKAABECFG AAAAABKAAGEBBG AAAAABKBABAGFB AAAAABKBAEAGBG
AAAAABKFABABFB AAAAABKFAEABBG AAAAABKHABABFB AAAAABKHABACFG AAAAABKHAEABBG AAAAADEABAEFBD AAAAADEAFAAIBA AAAAADEAFAEABD
AAAAADEAFBEIBD AAAAADEAQAACBA AAAAADEHFEAIBA AAAAADEHQEACBA AAAAADGAAAAABA AAAAADGAAAAABD AAAAADGAHAEABD AAAAADKAAEAABG
AAAAADKHABACBB AAAAADKHABACBG AAAAAEFBAABBAF AAAAAFKAAFAAFB AAAAAFKAAGAABG AAAAAFKAHFEAFB AAAAAFKAHGEABG AAAAAFKHABACBB
AAAABADBABAGFB AAAABADBAFAGFB AAAABADFABABFB AAAABADFAFABFB AAAABADQABEBFB AAAABADQAFEBFB AAAABADQAFEDIB AAAABADQAQEDCB
AAAABAEFBGAFGD AAAABAEFBQAFBD AAAABAFQAFBDIF AAAABAFQAQBDCF AAAABAFQAQGDCB AAAABAHGAFBDIH AAAABAHGAFBGIH AAAABAHGAQBDCH
AAAABAHBAQBGCH AAAABAHQAFBBIF AAAABAHQAFBBIH AAAABAHQAFBDIF AAAABAHQAQBBCF AAAABAHQAQBBCH AAAABAHQAQBDCF AAAABAHQAQGBCB
AAAABAHQAQGDCB AAAABAHQBQBFCF AAAABAHQBQGFCB AAAABANFBGAFGD AAAABANFBQAFCD AAAABANFBQAFGD AAAABANQBGEFGD AAAABANQBQEFCD
AAAABANQBQEFGD AAAABBBAAACAAH AAAABBBAAACABH AAAABBBAABCBFD AAAABBBAABEBFD AAAABBBAAGCBBD AAAABBBAAGEBBD AAAABBBAEAEBAD
AAAABBBAEBEBAD AAAABBBAHGCFBD AAAABBBAHGEFBD AAAABBBBABBBFD AAAABBBBABCAFD AAAABBBBAECGBA AAAABBBBAG[illegible]BBD AAAABBBBAGCABD
AAAABBBBEBAGAD AAAABBBBHECABA AAAABBBBHGCABA AAAABBBFAECBBA AAAABBDAAAEAAB AAAABBDAAAEBAB AAAABBDAEAEBAB AAAABBDBABBBFB
AAAABBDBABCAFB AAAABBDBABCGAB AAAABBDBEBAGAB AAAABBDHABABFB AAAABBDHEBABAB AAAABBEFBGAFGD AAAABBFAAABAAF AAAABBFAAABBAF
AAAABBFAAAGAAB AAAABBFAAAGBAB AAAABBFAEABBAF AAAABBFAEAGBAB AAAABBFAEBGBAB AAAABBHAAABAAF AAAABBHAAACAAB AAAABBHAAAGAAB
AAAABBHABABFAF AAAABBHABACAAH AAAABBHABAGFAB AAAABBHABBGFAB AAAABBHAGBGBAB AAAABBHBAACAAH AAAABBHBGBCGAB AAAABBHFBBCFAB
AAAABBNABBCAGH AAAABBNABAEFAD AAAABBNABECAGA AAAABBNBBACAGH AAAABBNBBECAGA AAAABBNBBGCAGA AAAABBNHBAAFAD AAAABDEABAEFBD
AAAABDEABBEFBD AAAABEFBEAFGAF AAAABEFGEABGAF AAAABEFQAABDAF AAAABEFQEABBAF AAAABEHBAACDAH AAAABEHBAAFDAH AAAABEHBAAFGAF
AAAABEHBAAFGAH AAAABEHBGAFGAF AAAABEHGAABDAH AAAABEHGAABGAF AAAABEHGAABGAH AAAABEHQAABBAF AAAABEHQAABBAH AAAABEMQAABDAF
AAAABEHQAACDAB AAAABEHQBACDAH AAAABGHAAABBAF AAAABGHAAABBAH AAAABGHAAABDAF AAAABGHAAAGBAB AAAABGHAAAGDAB AAAABGNABACDGH
AAAABGNBBACDGH AAAABKNABACDGH AAAACADAAAEACB AAAACADAAAEDAB AAAACADAAAEDCB AAAACADAAFEAIB AAAACADAAQEACB AAAACADGAAEDAB
AAAACADGABEDCB AAAACADGABEDFB AAAACADGAFBDIB AAAACADGAFEAIB AAAACADGAFEQAB AAAACADGAQBDCB AAAACADGAQEACB AAAACADQAAEDAB
AAAACADQABEDCB AAAACADQABEDFB AAAACADQAFBDIB AAAACADQAFEDAB AAAACADQAQBDCB AAAACAFAAABDAF AAAACAFAAABDCF AAAACAFAAAGDCB
AAAACAFBAFBDIF AAAACAFBAQBDCF AAAACAFGAQGACB AAAACAFQABGDCB AAAACAFQAFBDAF AAAACAKAAFEAFB AAAACBDAAABAAB AAAACBFBAABAAF
AAAACFKAAK[illegible] AAAADANQBGCBGA AAAADGNABACBGH AAAADGNBBACBGH AAAADKNABACBGH AAAAGEHQAABDAB AAABAABAACDBHQ AAABAABAACEBHD
AAABAABABCDBAD AAABAABABCDBCD AAABAABBBCDBCD AAABAADAACEBHB AAABAADAAFEBIB AAABAADAAQEBCB AAABAADBAFABIB AAABAADBAFDBIB
AAABAADBAFDGIB AAABAADBAQABCB AAABAADBAQDBCB AAABAADBAQDGCB AAABAAEAFQEFBD AAABAAFAAFBBIF AAABAAFAAQBBCF AAABAAFAAQGBCB
AAABAAFEAFBGIF AAABAAFEAQBGCF AAABAAGAAQEBBD AAABAAGAHQEFBD AAABAAGBAGABBD AAABAAGBAQABBD AAABAAGBBGDBBD AAABAAGBBQDBBD
AAABAAKAAFEBFB AAABAAKBABABFG AAABAAKBABDBFG AAABAAKBAFABFB AAABAAKBAFDBFB AAABAAKBAEDGFB AAABAAKBAGABBG AAABAAMAABBBFQ
AAABAAMAABBBHQ AAABAAMAABDBFQ AAABAAMAABDBHG AAABAAMAABGBFG AAABAAMAABGBHG AAABAAMAAFDBHB AAABAAMAAFGBFB AAABAAMAAGBBBQ
AAABAAMAAGDBBQ AAABAAMBABFBFQ AAABAAMBABFGHQ AAABAAMBAGFBBQ AAABAAMEABBGFQ AAABAAMEABBGHQ AAABAAMEAGBGBQ AAABAAMGABBGHQ
AAABABBAAAEBAD AAABABBAACEBAD AAABABBABCDFAQ AAABABBABCEFAD AAABABBRAADCAD AAABABBRBADCAD AAABABDAAAEBAB AAABABDAACEBAB
AAABABDBAADCAB AAABABFAAABBAF AAABABFAAAGBAB AAABABFAACBBAF AAABABFAACGBAB AAABABFEAABGAF AAABADEABAEFBD AAABADEAFAEFBD
AAABADEAFCEFBD AAABADKRAEACBG AAABADKRAEDCBG AAABADMAAEDCBQ AAABAEFBAAFBAF AAABAFGRAGACBD AAABAFGRAG[illegible]CBD AAABAFGRBGDCBD
AAABAFKRABDCFG AAABAFKRAGACBG AAABAFKRAGDCBG AAABAFMAABDCFQ AAABAFMAAGDCBQ AAABAHKBABDCFG AAABAHKBAGACBG AAABAHKBAGDCBG
AAABAHMAABBCFQ AAABAHMAABBCHQ AAABAHMAABDCFQ AAABAHMAABDCHG AAABAHMAABGCHG AAABAHMAAGBCBQ AAABAHMAAGDCBQ AAABAHMBABFCFQ
AAABAHMBAGFCBQ AAABBADAACABHB AAABBADBACAGHB AAABBADBAFAGIB AAABBADBAQAGCB AAABBADBRCADCB AAABBADBRQADCB AAABBADGRCEDCB
AAABBADGRQEDCB AAABBAEBBCADBA AAABBAEBBCADBD AAABBAEBBQADBD AAABBBDAAAEBAB AAABBBDAACBBHB AAABBBDAECBBAB AAABBBDBABAGFB
AAABBBDBABAGHB AAABBBDBACABHB AAABBBDBEBAGAB AAABBBDBECABAB AAABBBDEACBDHB AAABBBDEACEAHB AAABBBDHAAABAB AAABBBDHABABFB
AAABBBEABCAFDA AAABBBFAAAGBAB AAABBBFAHCGAHB AAABBDEABCEABD AAABBEFBAAFGAF AAABBEFBEAFGAF AAABBEHBBAFDAF AAABBEHBBAFDAH
AAABBEHBGAFGAF AAABCAFAAFBDIF AAABCAFAAQBDCF AAABCAFAAQGDCB AAABCAFGAQGDCB AAABCBDEAAJAAB AAABCBDRAADAAB AAABCBFEAAGAAB
AAABCDGEAAEABD AAABCDGEAAJABD AAABCDGEBAJABD AAABCDGEBCJABD AAABCFGEAGEABD AAABCFGEAGJABD AAABCFGEAQEABD AAABCFGEAQJABD
AAABCFGEBGJABD AAABCFGEBQJABD AAABCFKEAFEAFB AAABCFKEAFJAFB AAABCFKEAGEABG AAABCFKEAGJABG AAABCFMEABBAFQ AAABCFMEAFGAFB
AAABCHKAAFEAFB AAABCHKAAGEABG AAABCHKBAFDAFB AAABCHMAABDAHG AAABCHMAABGAHG AAABCHMAAGDABQ AAABDAEBBCABBA AAABDBERBCACDA
AAABEAHAAQBDCF AAABEAHEAQGDCB AAABEBHEAAGAAB AAABFBGABCAFDA AAABFBGHBCAFDA AAABGAHAAFBDIH AAABGAHAAQBDCH AAABGAHEAFBDIH
AAABGAHEAQBDCH AAABGAHGAFBDIH AAABGAHGAQBDCH AAACABEABEMCGD AAACABEABEMFGD AAACABEABGMFGD AAACADKAAEACBG AAACADKAAEDCBG
AAACADMAABDCBQ AAACADMAABDCBG AAACAFKAAGDCBG AAACBBDAABABFB AAACBBDBABAAFB AAACBBEABEAALA AAACBBEABGAFLD AAACBBEBBEAALA
AAACBBEBBGAALA AAACBBEFBEAFLA AAACBBNABEAAGA AAACBBNABGAFGD AAACDBEBBEACLA AAACDBEBBGACLA AAADAAFAAFBBIF AAADBENBBACDAD
AAADCAFAAFBDIF AAAEBBEABGAFLA AAAFBBNBBACAAD AAAFBGNBBACDAD AAAGAAMAABDBBQ AAAGA[illegible]HBDAHQ AAAGAHMAABDCBQ AAAGAHMAABDCBG
AAAGAHMAHBDAHQ AAAGAHMAHBDAHG AAAGCAHEABDAHQ AAAGCAMEABDAHG AAAGCDGEBBJABD AAAGCDGRBBDABD AAAGCDGRBBDABQ AAAIAHKAHFEAFB
AAAICAKEAFEAFB AABAABBBADBCAD AABAABBBCDBADA AABAABBBCDBCBD AABAABBBCGBBBD AABAABBBCGBCBD AABAABBBFGBCBD AABAABDBADBCAB
AABAABDBCABCAB AABAABFAAABCAB AABAABFAADBCAB AABAABFBADBCAB AABACADGAFEDIB AABACADGAQEDCB AABACAFGAQGDCB AABACAFQAFBDIB
AABACAFQAQBDCB AABACBDAADEAAB AABACBDBCABAAB AABACDGAADEABD AABADGHBADCBAR AABADGRBCAABAR AABADGRHCAABAR AABAEAHGAQGDCB
AABAEAHQAQGDCB AABBAAIBCGAABE AABBAAIBCGABBE AABBAAIBCGABBG AABBAAKBAFABFB AABBABDBCADBAB AABBABDECADAAB AABBABDECAEAAB
AABBABFAAAGCAB AABBADIHCEAHBG AABBADIHQEACBG AABBADIRCEAABG AABBADIRCEACBE AABBADIRCEACBG AABBADIRQEACBG AABBADKHAEACBG
AABBADKHCEACBG AABBADKICEAHBG AABBADKRAEACBG AABBADKRCEAABG AABBAFGHAGACBD AABBAFGHCGACBD AABBAFGHCGAHBQ AABBAFIHCGAHBG
AABBAFIHQGACBG AABBAFIRCGAABG AABBAFIRCGACBE AABBAFIRCGACBG AABBAFIRQGACBG AABBAFKECFDAFB AABBAFKECFEAFB AABBAFKECGEABG
AABBAFKHQGACBG AABBAFKHCGACBG AABBAFKHCGAHBG AABBAFKRAGACBG AABBAFKRCGAABG AABBAHIBCGAABG AABBAHIBCGACBE AABBAHIBCGACBG
AABBAHIBCGACBG AABBAHKBCGAABG AABBAHKBCGAABG AABBBAEBBGADBE AABBBAEBBGADBE AABBBAEBBQADBD AABBBAHBCAFAAH AABBBAIBBBADBE
AABBBHIBBBAABE AABBCAFGAQGDCB AABBCAGBCADABD AABBCBFEAAGAAB AABBCDGEAAEABD AABBCDKECBJABG AABBCFGEAGEABD AABBCFKAAGEABG
```

```
AABBCFKACGDABG AABBCFKACGEABG AABBCFKEAFEAFB AABBCFKEAGEABG AABBCFMRAGDABG AABBCHKACGDABG AABBCHMRAGDABG AABGAEGAAJMCDQ
AABCAGIBCDABTB AABCBBEBBGAALD AABCBBEFBGAHLD AABCBBNBBGAAGD AABCBBNFBGAHGD AABCBGIBCDABTB AABDBEHRAACDAR AABDBEHBCACDAH
AABDBENBBACDAD AABDBENBBACDAR AABDBERBBACDAB AABFBBHBCABAAH AABFBBHBCACAAH AABFBBNBBACAAD AABFBBNBBACAAR AABFBBRBBACAAB
AABFBGNBBACDAD AABFBGNBBACDAR AABFCBFBAABAAB AABFDGNBBACBAR AABFEBHBAABAAB AABGADIHQDACBB AABGBBHERBGAAB AACADMNABDCBQA
AACBBAEBBAADBD AACDBENBBACDAA AACFBGNBBACDAA AACFDGNBBACBAA AAECBBIFBDAFBR AAECBBIFBDAFGR AAECBBNFBDAFGR AAFCBBIBCBAABB
AAGABARQBQEHCR AAGCBBIBBJAAGE AAGCBBIBBOAAGE AAGDBENBBACDAA AAGFBAHBBAFAAH AAGFBBHBBACAAH AAGFBGNBBACDAA ABAAAGJADABBAF
ABAAGELQDABDAH
```

DRAWINGS WITH 31 CROSSINGS:

```
AAAAAABAAAAAAA AAAAAABAAQEBCD AAAAAABBAAAGCD AAAAAABBAQABCD AAAAAABBAQAGBD AAAAAABBAQAGCB AAAAAADAAFEBIB AAAAAADAAQEBCB
AAAAAADBAAAGCB AAAAAADBAFABIB AAAAAADBAFAGCB AAAAAADBAFAGFB AAAAAADBAQABCB AAAAAADBHFAAIB AAAAAADBHQAACB AAAAAADFAAABCB
AAAAAADFAFABCB AAAAAADFAFABFB AAAAAADGHFEAIB AAAAAADGHQEACB AAAAAAEABGEFGD AAAAAAEABQEFGD AAAAAAEAFQEFBD AAAAAAEBBGAAGD
AAAAAAEBBQAAGD AAAAAAEBFQAABD AAAAAAEFBAAFGD AAAAAAEFBGABGD AAAAAAEFBGAFBD AAAAAAEFBGAFGD AAAAAAEFBQABGD AAAAAAEFBQAFBD
AAAAAAKAABEBFG AAAAAAKAAFEBFB AAAAAAKAAGEBBG AAAAAAKBABABFG AAAAAAKBABAGBG AAAAAAKBABAGFB AAAAAAKBAFABFB AAAAABBAAAACAD
AAAAABBAAAEAAD AAAAABBAAAEBAD AAAAABBAAAECAB AAAAABDAAAACAB AAAAABDAAAEAAB AAAAABDAAAEBAB AAAAABDHAAABAB AAAAABEFBGAFGD
AAAAABFAAABAAF AAAAABFAAABBAF AAAAABFAAABCAB AAAAABFAAAGAAB AAAAABFAAAGBAB AAAAABGAAAECBD AAAAABGBAGAGBD AAAAABGFAGABBD
AAAAABGHAAACBD AAAAABGHAEACBD AAAAABKBABAGFG AAAAABKBAGAGBG AAAAABKFABABFG AAAAABKHAEACBG AAAAADEAFAAIBD AAAAADEAFAEFBD
AAAAADEHFAAIBA AAAAADEHQAACBA AAAAADGAAAACBD AAAAADGAAAEABD AAAAADGAHAEFBD AAAAADKAAEACBG AAAAAEFBAABGAF AAAAAEFBAAFBAF
AAAAAFKAABECFB AAAAAFKAAFEAFB AAAAAFKAAGACBG AAAAAFKAAGEABG AAAAAFKHABACBG AAAAAFKHABACFB AAAABADBAFAGIB AAAABADBAQAGCB
AAAABADFAFABIB AAAABADFAQABCB AAAABADQAFEBIB AAAABADQAQEBCB AAAABAEFBQAFGD AAAABAFQAFBBIF AAAABAFQAQBBCF AAAABAFQAQGBCB
AAAABAHQAFBDIH AAAABAHQAQBDCH AAAABBBBABAGFD AAAABBBBABBGFD AAAABBBBABCBFD AAAABBBBAGAGBD AAAABBBBAGBGBD AAAABBBBAGCBBD
AAAABBBBAGCGBA AAAABBBBEBCGAD AAAABBBBHGCABD AAAABBBFAGABBD AAAABBDBABAGFB AAAABBDBABBGFB AAAABBDBABCBFB AAAABBDBEBCGAB
AAAABBDBHBCAFB AAAABBDFABABFB AAAABBHAAABAAH AAAABBNFBGAFGD AAAABEFBAAFGAF AAAABEFGAABGAF AAAABEFQAABBAF AAAABEHQAABDAH
AAAABEHQAACBAH AAAABGHAAABDAH AAAABGHAAACBAH AAAACADAAAEAAB AAAACADAAFEDIB AAAACADAAQEDCB AAAACADGAAEDCB AAAACADGAFEDIB
AAAACADGAFEDCB AAAACADGAFEDFB AAAACADGAQEDCB AAAACADQAAEDCB AAAACADQAFEDCB AAAACADQAFEDFB AAAACAFAAFBDIF AAAACAFAAQBDCF
AAAACAFAAQGDCB AAAACAFGAQGDCB AAAACAFQAABDCF AAAACAFQAAGDCB AAAACAFQAFBDCF AAAACAFQAFBDIB AAAACAFQAQBDCB AAAACBFAAABAAB
AAAACFKAABEABG AAAACFKAABEAFB AAAAGEHGAABDAH AAAAGEHGAABGAH AAABAABABCDFAQ AAABAADBAFAGIB AAABAADBAQAGCB AAABAAGBAGAGBD
AAABAAGBAQAGBD AAABAAKBABAGFG AAABAAKBAFAGFB AAABAAKBAGAGBG AAABAAMAABDBHQ AAABAAMBABFGFQ AAABAAMBAGFGBQ AAABAAMGABBGFQ
AAABAAMGAGBGBQ AAABABBAAAECAD AAABABBAACEBHD AAABABDAAAECAB AAABABDAACEBHB AAABABFAAABCAF AAABABFAAAGCAB AAABABFAACBBHF
AAABADEABCEFBD AAABADEAFAEIBD AAABADKHAEACBG AAABADMAAEBCBQ AAABAEFBAAFGAF AAABAFGHAGACBQ AAABAFKHAGACBG AAABAFMAABBCFQ
AAABAFMAAGBCBQ AAABAHMAABDCHQ AAABBADAACEBHB AAABBBDAACABHB AAABBBDAACEBAB AAABBBDAEAEBAB AAABBBDAECABAB AAABBBDBACAGHB
AAABBBDBECAGAB AAABBBDEHCEAHB AAABBBDHABABHB AAABBBDHACABAB AAABBBDHEAABAB AAABBBDHEBABAB AAABBBEHBCAFBD AAABBBFAABGBHB
AAABBBFAACBBAF AAABBBFAACGBAB AAABBBFAEAGBAB AAABBBFBECFGAF AAABBBFEACBGAF AAABBDEABAEFBD AAABBDEABBEFBD AAABBDEABCAFBD
AAABBDEHBAAFBD AAABCBFAAAGAAB AAABCFKAAFEAFB AAABCFKAAGEABG AAABCFMAABBAFQ AAABCHMAABDAHQ AAABEAHGAQGDCB AAABGAHGAFBDIH
AAABGAHGAQBDCH AAACADMAAEDCBG AAACAFMAABDCBQ AAACAFMAAGDCBG AAACBBDBABABFB AAACBBDBHBAAFB AAACBBEBBGAALD AAACBBEFBGAFLA
AAACBBNBBGAAGD AAADAAFAAFBBKF AAADCAFAAFBDKF AAAEAAKAABMBBG AAAEAAMAABDBBQ AAAEAAMAABDBBG AAAEAHKAABMCBG AAAEAHMAABDCBQ
AAAEAHMAABDCBG AAAEBBEABGAFLD AAAEBBEABGEBLD AAAEBBEHBGAFGD AAAEBBNABGAFGD AAAEBBNHBGABGD AAAEBBNHBGAFGA AAAECHMAABDABQ
AAAGAAMEHBDAHQ AAAGAHMAABGCHG AAAGAHMAABDAHG AAAGCAMAABDAHQ AAAGCAMAABDAHG AAAGCHMAABGAHG AAAIAHKAAFEAFB AAAICAKAAFEAFB
AABAABBBCDBAQD AABAABBBCDBCAD AABAABDBCBBCQB AABAABDBCDBCAB AABAABFAAAGCAB AABACADQAFEDIB AABACADQAQEDCB AABACAFQAQGDCB
AABACBDAAAEAAB AABACBDBCDBAAB AABACBFAAAGAAB AABACDGAAAEABD AABACFKAAFEAFB AABACFKAAGEABG AABAEAHQAQGDCB AABBAAKBAFDBFB
AABBABDBCADCAB AABBABDRCADAAB AABBADIHCEAHBE AABBADIRCEAABE AABBADKRCEACBG AABBAFIHCGAHBE AABBAFIRCGAABE AABBAFKRCGACBG
AABBAHIBCGAABE AABBAHKBCGACBG AABBBAIBBGADBE AABBBAIBCBADBE AABBBAIBCGAABE AABBBAIBCGABBE AABBBDIHBEAHBE AABBBFIHBGAHBE
AABBBHIBBGAABE AABBBHIBCBAABE AABBBHIBCFAAFB AABBCAGBCGDABD AABBCAGBCQDABD AABBCAKBCGDABG AABBCBDBCADAAB AABBCBDECADAAB
AABBCBDECAEAAB AABBCDGEAAJABD AABBCDGEBAJABD AABBCDGECADABD AABBCDGECAEABD AABBCDGRAADABD AABBCDGRBADABD AABBCFGEAGJABD
AABBCFGECGEABD AABBCFKEAFJAFB AABBCFKEAGJABG AABBCFKECBJABG AABBCFKECBJAFB AABBCFKECFDAFB AABBCFKECFEAFB AABBCFKECGDABG
AABBCFKECGEABG AABBCFKRAGDABG AABBCHKBAGDABG AABBHDKHCEACBG AABBHFKHCGACBG AABBHFKHCGAHBG AABBHFKRCBACBG AABCBBEFBGAFLD
AABCBGIBCDABTR AABDBAHBCFNAIH AABDBEHBCACDAR AABDBERBBACDAR AABDBERBCACDAB AABFBBHBCACAAR AABFBBHBCAFAAH AABFBBRBBACAAR
AABFBBRBCACAAB AACDBENBBACDAD AACFBGNBBACDAD AADCHBKBCBACDB AADGHBKRCBACBB AAECBBIFBDAFDR AAECBBIFBDAFQR AAECBENFBDAFQR
AAGBBAIBBGAABE AAGBBAIBBGABBE AAGCBBIBBDAAQR AAGCBBIFBDAHQR AAGFBBHBBAFAAH ABANCHQADCDABQ
```

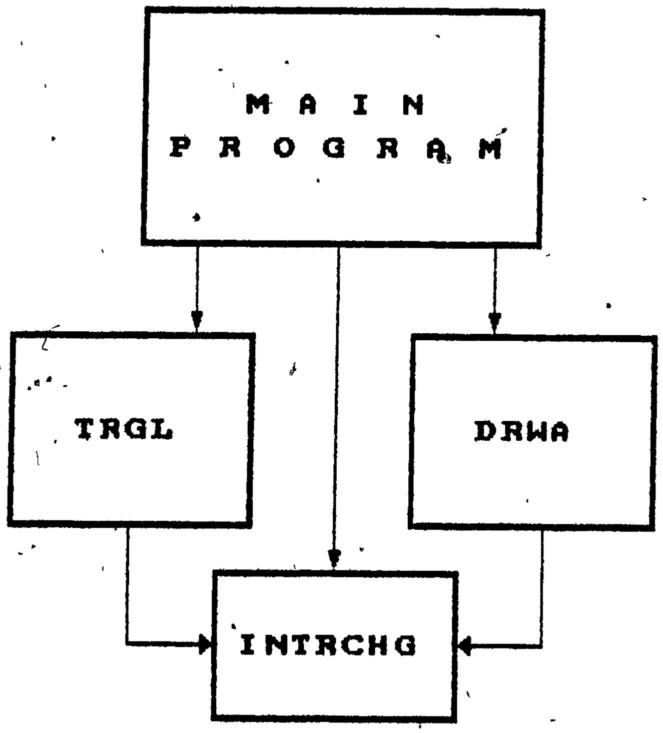
DRAWINGS WITH 35 CROSSINGS:

```
AAAAAAAAAAAAAA AAAAAABBAQAGCD AAAAAADBAFABIB AAAAAADBAQAGCB AAAAAADFAFABIB AAAAAADFAQABCB AAAAAAEFBGAFGD AAAAAAEFBQAFGD
AAAAAAKBABAGFG AAAAAAKBAFAGFB AAAAAAKBAGAGBG AAAAABBAAAECAD AAAAABDAAAECAB AAAAABDHAAACAB AAAAABFAAABCAF AAAAABFAAAGCAB
AAAAADEAFAEIBD AAAAADGAAAECBD AAAAADKHAEACBG AAAAAEFBAAFGAF AAAAAFKHABACFG AAAAAFKHAGACBG AAAABBBBABCGFD AAAABBBBAGDGBD
AAAABBFAGCBBD  AAAABBDBABCGFB AAAABBDFABCGFB AAAABBHAAACAAH AAAABEHQAACDAH AAAACADGAFEDIB AAAACADGAQEDCB AAAACADQAFEDIB
AAAACADQAQEDCB AAAACAFQAFBDIF AAAACAFQAQBDCF AAAACAFQAQGDCB AAAACBDAAAEAAB AAAACBFAAABAAF AAAACBFAAAGAAB AAAACFKAABEAFG
AAAACFKAAFEAFB AAAACFKAAGEABG AAAAGEHQAABDAH AAAAGEHQAABDAH AAABBBDAACEBHB AAABBBDAECEBAB AAABBBDHACABHB AAABBBDHECABAB
AAABBBFAACBBHF AAABBBFAECGBAB AAABBBFEACBGHF AAABBDEABCEFBD AAABBDEHBCAFBD AAACADMAAEDCBQ AAACADMAAEDCBG AAACAFMAAGDCFQ
AAACAFMAAGDCBQ AAACBBDBABAGFB AAACBBDFABABFB AAACBBEFBGAFLD AAADAAFAAIBBKF AAADAAMEACBGHQ AAADCAFAAIBDKF AAAEAAKAAGMBBG
AAAEAAMAABDBFQ AAAEAAMAAGDBBG AAAEBBEABGEFLD AAAEBBNHBGAFGD AAAECHMAAGDABQ AAAGAAMAABDBHQ AAAGAHMAABDCHQ AAAGAHMAABDCHG
AAAGCHMAABDAHQ AAAGCHMAABDAHG AAAIAAKAAFEBFB AAAICHKAAFEAFB AABAABBBCDBCQD AABABGRBCDCDQR AABBABDRCADCAB AABBBAIBCGADBE
AABBBDIHCEAHBE AABBBFIHCGAHBE AABBBHIBCGAABE AABBCBDECAJAAB AABBCBDRCADAAB AABBCDGECAJABD AABBCDGRCADABD AABBCFGECGJABD
AABBCFKECFJAFB AABBCFKECGJABG AABBCFKRCGDABG AABBCHKBCGDABG AABBDAIBCGABBE AABBHAKBCGABBG AABBHDKHCEAPBG AABBHDKRCEACBG
AABBHFKHCGAPBG AABBHFKRCGACBG AABDBAHBCFNDIH AABDBERBCACDAR AABFBBRBCACAAR AACBBBFAEAGBAB AADGHBKRCBACDB AADGHDKRCBACBB
AAECBBIFBDAFTR AAECBBRFBDAFQR AAGADGNBBDCBGX AAGBBAIBBGADBE AAGBBHIBBGAABE AAGCBBIBBDAATR AAGCBBIFBDAHTR AAGCBBRBBDAAQR
AAGCBBRFBDAHQR ACADGNPBDCBGAB BBAAKELFDHFBAH
```

APPENDIX B

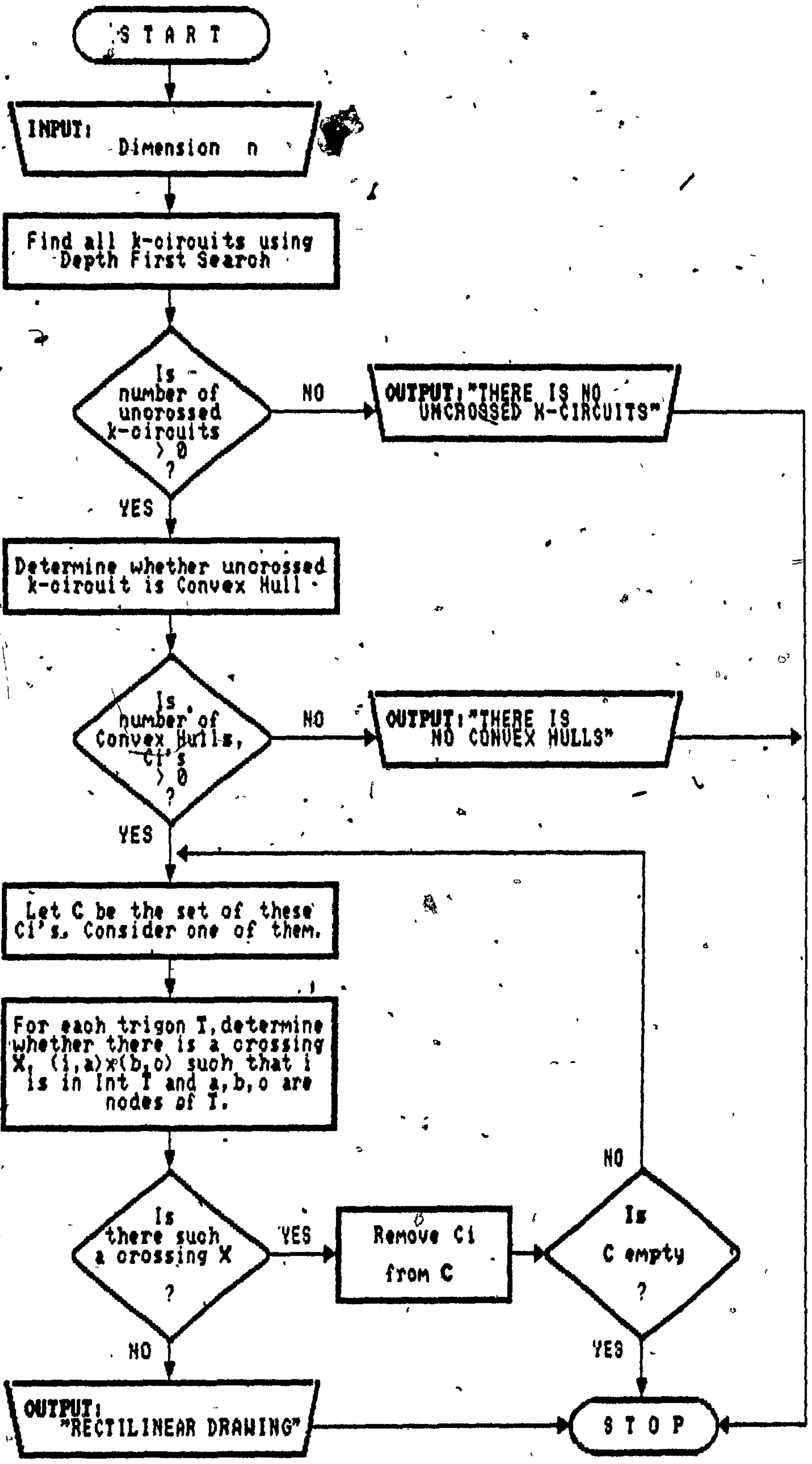
PROGRAM TO DETERMINE WHETHER K_n WITH A GIVEN SET OF CROSSINGS CAN BE MAPPED INTO A RECTILINEAR DRAWING D_n

I. GENERAL DIAGRAM



MAIN	:	Determines whether K_n could be mapped into a rectilinear drawing D_n having a given set of crossings.
TRGL	:	Determines the number of times a given arc crosses the arcs of a trigon.
DRWA	:	Determines whether (v,i)x(j,k) while v is in Int T=(i,j,k,i).
INTRCHG	:	Interchanges the values between two variables p and q such that p takes the value of q and q takes the value of p.

II. GENERAL FLOWCHART


START
INPUT: Dimension n
Find all k-circuits using Depth First Search
Is number of uncrossed k-circuits > 0 ?
NO
OUTPUT: "THERE IS NO UNCROSSED K-CIRCUITS"
YES
Determine whether uncrossed k-circuit is Convex Hull
Is number of Convex Hulls, Ci's > 0 ?
NO
OUTPUT: "THERE IS NO CONVEX HULLS"
YES
Let C be the set of these Ci's. Consider one of them.
For each trigon T, determine whether there is a crossing X, (i,a)×(b,c) such that i is in Int T and a,b,c are nodes of T.
Is there such a crossing X ?
YES
Remove Ci from C
Is C empty ?
NO
YES
NO
OUTPUT: "RECTILINEAR DRAWING"
STOP

III. COMPUTER PROGRAM (FORTRAN 77)

```
*******************************************************
*                                                     *
* G I V E N   A   S E T   O F   C R O S S I N G S   O F  *
*                                                     *
* T H E   C O M P L E T E   G R A P H   Kn ,  A L O N G  *
*                                                     *
* W I T H   T H E   D I M E N S I O N   N   A N D   T H E *
*                                                     *
* N U M B E R   O F   C R O S S I N G S ,   T H I S      *
*                                                     *
* P R O G R A M   W I L L   D E T E R M I N E            *
*                                                     *
* W H E T H E R   Kn   C O U L D   B E   M A P P E D     *
*                                                     *
* I N T O   A   R E C T I L I N E A R   D R A W I N G  Dn *
*                                                     *
*******************************************************
*                                                     *
*                                                     *
*                                                     *
*                                                     *
*                                                     *
      IMPLICIT INTEGER*2 (A-Z)
      COMMON Y1,Y2,Y3,Y4,NX,X1,X2,X3,X4,XX,IS
*
* X1(.),X2(.),X3(.),X4(.) =  the set of crossings of Kn  *
*                                                     *
      DIMENSION X1(100),X2(100),X3(100),X4(100)
*                                                     *
*                                                     *
* TR(.,.) = all the trigons of Kn                     *
* NT(i,j) = the j-th node located in the i-th trigon  *
* IN(i)   = number of nodes located in the i-th trigon *
*                                                     *
      DIMENSION TR(120,7),NT(120,3),IN(120)
*                                                     *
*                                                     *
*  M(.,.) and LB(.,.) are used in the depth first search *
*  process.                                           *
*                                                     *
*  M(i,j) = v when edge (i,v) is the j-th uncrossed edge *
*                                                     *
*  LB(i,j) = 1 if node j has been visited from node i *
*          = 0 otherwise                              *
*                                                     *
*  (A1(.),A2(.)) = uncrossed edges                    *
*                                                     *
*                                                                      *
      DIMENSION M(10,9),LB(10,10),A1(50),A2(50)
*                                                                      *
*                                                                      *
*  CR(.,.)   = all uncrossed k-circuits                                *
*                                                                      *
*  CRCT(.,.) = an uncrossed k-circuit                                  *
*                                                                      *
*  LG(i) - 1 = number of arcs of the i-th uncrossed                    *
*              k-circuit.                                              *
*                                                                      *
*  TC(i) used when sorting CR(.,.)                                     *
*                                                                      *
*                                                                      *
      DIMENSION CR(50,11),CRCT(11),LG(50),TC(50)
*                                                                      *
*                                                                      *
*                                                                      *
*    ND(.)   = nodes different from the nodes of the                   *
*              uncrossed k-circuit                                     *
*                                                                      *
*    DD(.)   = number of nodes different from the nodes                *
*              of the uncrossed k-circuit                              *
*                                                                      *
*    IX(.) used in calculating the number of crossings                 *
*          involving the nodes of an uncrossed k-circuit               *
*                                                                      *
*                                                                      *
      DIMENSION ND(50,7),DD(50),IX(50)
*                                                                      *
*                                                                      *
*                                                                      *
*                                                                      *
*                                                                      *
```

```
*************************************************************
***************                           *******************
***************        S T E P   1        *******************
***************                           *******************
***************        I N P U T          *******************
***************                           *******************
*************************************************************
*    Input:   1. dimension, N                               *
*             2. number of crossings, NX                    *
*             3. crossings, X1(.),X2(.),X3(.),X4(.)          *
*                                                           *
*    Sort the crossings in ascending order and such that    *
*    if (a,b) x (c,d) then a < b , c < d and a < c .        *
*************************************************************
       WRITE(*,'(1X,''  INPUT  N ,  NUMBER OF CROSSINGS'')')
       READ(*,'(2I2)') N,NX
       WRITE(*,'(1X,''  INPUT  CROSSINGS'')')
       DO 1000 I=1, NX
          READ(*,'(4I2)') X1(I),X2(I),X3(I),X4(I)
          IF (X1(I).GT.X2(I)) CALL INTRCHG(X1(I),X2(I))
          IF (X3(I).GT.X4(I)) CALL INTRCHG(X3(I),X4(I))
          IF (X1(I).GT.X3(I)) THEN
             CALL INTRCHG(X1(I),X3(I))
             CALL INTRCHG(X2(I),X4(I))
          ENDIF
1000   CONTINUE
       DO 1020 I=1, NX-1
          A = X1(I)
          B = X2(I)
          C = X3(I)
          D = X4(I)
          DO 1010 J=I+1, NX
             IF (A.LT.X1(J)) GOTO 1010
             IF (A.GT.X1(J)) GOTO 1005
             IF (B.LT.X2(J)) GOTO 1010
             IF (B.GT.X2(J)) GOTO 1006
             IF (C.LT.X3(J)) GOTO 1010
             IF (C.GT.X3(J)) GOTO 1007
             IF (D.LT.X4(J)) GOTO 1010
             IF (D.GT.X4(J)) GOTO 1008
1005         A     =  X1(J)
             X1(J) =  X1(I)
             X1(I) =  A
1006         B     =  X2(J)
             X2(J) =  X2(I)
             X2(I) =  B
1007         C     =  X3(J)
             X3(J) =  X3(I)
             X3(I) =  C
1008         D     =  X4(J)
             X4(J) =  X4(I)
             X4(I) =  D
1010      CONTINUE
1020   CONTINUE
```

```
*****************************************************************
********                                                 ********
********                   S T E P   2                   ********
********                                                 ********
********   F I N D   A L L   U N C R O S S E D           ********
********                                                 ********
********   K - C I R C U I T S   U S I N G               ********
********                                                 ********
********   D E P T H   F I R S T   S E A R C H           ********
********                                                 ********
*****************************************************************
*                                                               *
*                                                               *
*         Obtain all uncrossed edges ( A1 , A2 )                *
*                                                               *
*****************************************************************
*
*
      NA = 0
      NC = 0
      DO 2020 I=1, N-1
         DO 2010 J=I+1, N
            DO 2000 K=1, NX
               IF ((X1(K).EQ.I .AND.  X2(K).EQ.J)
     *                          .OR.
     *             (X3(K).EQ.I .AND.  X4(K).EQ.J))
     *         GOTO 2010
2000        CONTINUE
            NA = NA+1
            A1(NA) = I
            A2(NA) = J
2010     CONTINUE
2020  CONTINUE
*
*
*****************************************************************
*                                                               *
*                  Forming array M(.,.)                         *
*                                                               *
*         M(i,.) = v  when edge (i,v) is uncrossed              *
*                                                               *
*****************************************************************
*
      DO 2040 I=1, N
         DO 2030 J=1, N-1
            M(I,J) = 0
2030     CONTINUE
2040  CONTINUE
      DO 2050 I=1, NA
         IX(I) = 0
2050  CONTINUE
```

```
         DO 2060 I=1, NA
            M1 = A1(I)
            M2 = A2(I)
            IX(M1) = IX(M1)+1
            IX(M2) = IX(M2)+1
            M(M1,IX(M1)) = M2
            M(M2,IX(M2)) = M1
 2060    CONTINUE
         DO 2080 I=1, N
            DO 2070 J=1, N-1
               IF ( M(I,J).EQ.0) THEN
                  IX(I) = J-1
                  GOTO 2080
               ENDIF
 2070       CONTINUE
 2080    CONTINUE
*
*
*
***********************************************************
*                                                         *
*               Initialize LB(.,.) to zero                *
*                                                         *
***********************************************************
*
*
         DO 2100 I=1, N
            DO 2090 J=1, IX(I)
               LB(I,J) = 0
 2090       CONTINUE
 2100    CONTINUE
*
*
*
***********************************************************
*                                                         *
*          D E P T H   F I R S T   S E A R C H            *
*                                                         *
*          NC      = number of uncrossed k-circuits       *
*          CRCT(.) = an uncrossed k-circuit               *
*                                                         *
***********************************************************
*
*
         NC = 0
         DO 2110 I=1, N
            IF (M(I,1).GT.0) THEN
               RS = I
               GOTO 2115
            ENDIF
 2110    CONTINUE
 2115    CRCT(1) = RS
         A = RS
         I = 1
```

```
*
*
*
*****************************************************************
*                                                               *
*                                                               *
*                          (  1   when node j has been          *
*          LB(i,j) =       (      visited from node i           *
*                          (                                    *
*                          (  0   otherwise                     *
*                                                               *
*                                                               *
*****************************************************************
*
*
2117  DO 2120 J=1, IX(A)
         IF (LB(A,J).EQ.1) GOTO 2120
         P = M(A,J)
         I = I+1
         CRCT(I) = P
         B = J
         LB(A,J) =1
         GOTO 2135
2120  CONTINUE
      IF (I.EQ.1) GOTO 2155
      DO 2130 J=1, IX(A)
         LB(A,J) = 0
2130  CONTINUE
      I = I-1
      A = CRCT(I)
      GOTO 2117
2135  IF (CRCT(I-2).EQ.CRCT(I)) GOTO 2152
      DO 2150 K=1, I-2
         IF (CRCT(K).NE.P) GOTO 2150
         NC = NC+1
         LG(NC) = I-K+1
         DO 2140 L=K, I
            CR(NC,L-K+1) = CRCT(L)
2140     CONTINUE
         GOTO 2152
2150  CONTINUE
      GOTO 2153
2152  I = I-1
2153  A = CRCT(I)
      GOTO 2117
2155  IF (NC.EQ.O) THEN
         WRITE(*,'(1X,''..................................''/
     *              ''THERE IS  NO UNCROSSED k-CIRCUIT''/
     *              ''..................................'')')
         GOTO 9999
      ENDIF
```

```
************************************************************
*                                                          *
*    Arranging circuit CR(i,.) such that CR(i,2)<CR(i,k),  *
*                                                          *
*          where k is the length of CR(i,.)                *
*                                                          *
************************************************************
      DO 2200 I=1, NC
         DO 2160 J=1, LG(I)
            TC(J) = CR(I,J)
            TC(J+LG(I)-1) = CR(I,J)
2160     CONTINUE
         MN = N+1
         DO 2170 J=1, LG(I)-1
            IF (TC(J).GT.MN) GOTO 2170
            MN = TC(J)
            FR = J
2170     CONTINUE
         IF (TC(FR+1).GT.TC(FR+LG(I)-2)) GOTO 2182
         DO 2180 R=1,LG(I)
            CR(I,R) = TC(FR+R-1)
2180     CONTINUE
         GOTO 2200
2182     DO 2190 R=1, LG(I)
            CR(I,R) = TC(FR+LG(I)-R)
2190     CONTINUE
2200  CONTINUE
************************************************************
*                                                          *
*               Eliminate duplicate k-circuits             *
*                                                          *
************************************************************
      DO 2230 I=1, NC-1
         L = LG(I)
         DO 2220 K=I+1, NC
            IF (CR(K,1).EQ.0 .OR. LG(K).NE.L)
     *      GOTO 2220
            DO 2210 J=1, L
               IF (CR(I,J).NE. CR(K,J)) GOTO 2220
2210        CONTINUE
            CR(K,1) = 0
2220     CONTINUE
2230  CONTINUE
      MC = 0
      DO 2250 I=1, NC
         IF (CR(I,1).EQ.0) GOTO 2250
         MC = MC+1
         DO 2240 J=1, LG(I)
            CR(MC,J) = CR(I,J)
2240     CONTINUE
         LG(MC) = LG(I)
2250  CONTINUE
*
*
```

```
*
***************************************************************
*****                                                 *********
*****                S T E P   3                      *********
*****                                                 *********
*****   D E T E R M I N E   W H E T H E R   A N       *********
*****                                                 *********
*****   U N C R O S S E D   K-CIRCUIT IS A  CH.       *********
*****                                                 *********
***************************************************************
*
*
      OB = 0
      DO 3130 II=1, MC
         L = LG(II)
*
*
*
***************************************************************
*                                                             *
*    NN = number of nodes different from the nodes of         *
*         the uncrossed circuit under consideration.          *
*                                                             *
*         These nodes are stored in array ND(.)               *
*                                                             *
***************************************************************
*
*
         NN = 0
         DO 3010 J=1, N
            DO 3000 I=1, L-1
               IF (CR(II,I).EQ.J) GOTO 3010
3000        CONTINUE
            NN = NN+1
            ND(II,NN) = J
3010     CONTINUE
         DD(II) = NN
*
*
*
```

```
*************************************************************
*                                                           *
*    XN = number of crossings involving the nodes of the    *
*         uncrossed k-circuit.                              *
*                                                           *
*************************************************************
*
*
         XN = NX
         DO 3020 I=1, NX
            IX(I) = 0
3020     CONTINUE
         DO 3040 J=1, NN
            NU = ND(II,J)
            DO 3030 I=1, NX
               IF (IX(I).EQ.1) GOTO 3030
               IF (NU.EQ.X1(I) .OR. NU.EQ.X2(I) .OR.
     *             NU.EQ.X3(I) .OR. NU.EQ.X4(I))
     *         IX(I) = 1
               IF (IX(I).EQ.1) XN = XN-1
3030        CONTINUE
3040     CONTINUE
*
*
*
*************************************************************
*                                                           *
*        If XN < k!/(4!(k-4)!) then C is not a CH           *
*              where k is length of C                       *
*                                                           *
*************************************************************
*
*
         IF (XN.EQ.(L-1)*(L-2)*(L-3)*(L-4)/24)GOTO 3045
         IF (I1.LT.MC .OR. OB.GT.0) GOTO 3130
         WRITE(*,'(1X,''.........................''/
     *        1X,''THERE IS NO CONVEX HULL''/
     *        1X,''.........................'')')
3045     IF (NN.EQ.0 ) GOTO 3105
         IF (L.EQ.4) GOTO 3105
         DO 3100 I=1, NN
            NU = ND(II,I)
            DO 3090 J=1, NX
               IF (X1(J).EQ.NU .OR. X2(J).EQ.NU)
     *         GOTO 3065
               IF (X3(J).EQ.NU .OR. X4(J).EQ.NU)
     *         GOTO 3047
               GOTO 3090
3047           DO 3050 K=1, L-1
                  IF (CR(II,K).EQ.X1(J)) GOTO 3055
3050           CONTINUE
               GOTO 3065
```

```
3055              DO 3060 K=1, L-1
                     IF (CR(II,K).EQ.X2(J)) GOTO 3100
3060              CONTINUE
                  GOTO 3090
3065              DO 3070 K=1, L-1
                     IF (CR(II,K).EQ.X3(J)) GOTO 3075
3070              CONTINUE
                  GOTO 3090
3075              DO 3080 K=1, L-1
                     IF (CR(II,K).EQ.X4(J)) GOTO 3100
3080              CONTINUE
3090           CONTINUE
               GOTO 3130
3100        CONTINUE
3105        OB = OB+1
            DO 3110 K=1, L
               CR(OB,K) = CR(II,K)
3110        CONTINUE
            LG(OB) = L
            DD(OB) = DD(II)
            DO 3120 K=1, DD(OB)
               ND(OB,K) = ND(II,K)
3120        CONTINUE
3130     CONTINUE
         IF (OB.EQ.0) THEN
            WRITE(*,'(1X,''.......................''/
     *                1X,''THERE IS NO CONVEX HULL''/
     *                1X,''.......................'')')
            GOTO 9999
         ENDIF
*
*
*
```

```
*****************************************************************
*******                                                *********
*******                  S T E P 4                     *********
*******                                                *********
*******   F O R   E A C H   T R I G O N   T ,          *********
*******                                                *********
*******   D E T E R M I N E   W H E T H E R            *********
*******                                                *********
*******   T H E R E   I S   A   C R O S S I N G        *********
*******                                                *********
*******   ( V , A ) x ( B , C )   S U C H   T H A T    *********
*******                                                *********
*******   V   I S   I N   I N T   T ,   A N D   A ,    *********
*******                                                *********
*******   B , C   A R E   N O D E S   O F   T .        *********
*******                                                *********
*****************************************************************
*
*
      DO 4100 II=1, OB
         L = LG(II)
         NN = DD(II)
         WRITE(*,'(1X,///'' CONVEX HULL = '',20I2)')
     *                 (CR(II,K),K=1, L)
         WRITE(*,'(//)')
*
*
*
```

```
*
************************************************************
*                                                          *
*    Determine whether the drawing under consideration     *
*                                                          *
*    has a subdrawing equivalent to drawing A.             *
*                                                          *
************************************************************
*
*
                  DO 4060 Q=1, IN(IJK)
                     NU = NT(IJK,Q)
                     Y1 = I
                     Y2 = NU
                     Y3 = J
                     Y4 = K
                     CALL DRWA
                     IF (IS.EQ.1) GOTO 4100
                     Y1 = J
                     Y2 = NU
                     Y3 = I
                     Y4 = K
                     CALL DRWA
                     IF (IS.EQ.1) GOTO 4100
                     Y1 = K
                     Y2 = NU
                     Y3 = I
                     Y4 = J
                     CALL DRWA
                     IF (IS.EQ.1) GOTO 4100
4060              CONTINUE
4070           CONTINUE
4080        CONTINUE
4090     CONTINUE
          WRITE(*,'(1X,''.........................'')')
          WRITE(*,'(1X,''. R E C T I L I N E A R .'')')
          WRITE(*,'(1X,''.........................''////)')
4100   CONTINUE
9999   END
*
*
*
```

```
*
*
*
**************************************************************
*                                                            *
*        W = 1  when  T is the C H.                          *
*                                                            *
*        Hence W = 1 implies that node v is in Int T.        *
*                                                            *
**************************************************************
*
*
                IF (W.EQ.1) THEN
                   XX = 1
                   GOTO 4035
                ENDIF
                IF (V.EQ.I .OR.
     *                V.EQ.J .OR.
     *                   V.EQ.K )   GOTO 4040
                XX = 0
                Y1 = I
                Y2 = J
                Y3 = V
                Y4 = EX
                CALL TRGL
                Y1 = I
                Y2 = K
                Y3 = V
                Y4 = EX
                CALL TRGL
                Y1 = J
                Y2 = K
                Y3 = V
                Y4 = EX
                CALL TRGL
 4035           IF (XX.EQ.1 .OR.  XX.EQ.3) THEN
                   IN(IJK) = IN(IJK)+1
                   NT(IJK,IN(IJK)) = V
                ENDIF
 4040        CONTINUE
             IF (IN(IJK).EQ.0)GOTO 4070
             WRITE(*,'(1X,3I2,5X,17I2)') I,J,K,
     *                    (NT(IJK,R), R=1, IN(IJK))
             TR(IJK,1) = I
             TR(IJK,2) = J
             TR(IJK,3) = K
*
*
```

```
*****************************************************************
*                                                               *
*    Determine whether the drawing under consideration          *
*                                                               *
*    has a subdrawing equivalent to drawing A.                  *
*                                                               *
*****************************************************************
*
*
                 DO 4060 Q=1, IN(IJK)
                    NU = NT(IJK,Q)
                    Y1 = I
                    Y2 = NU
                    Y3 = J
                    Y4 = K
                    CALL DRWA
                    IF (IS.EQ.1) GOTO 4100
                    Y1 = J
                    Y2 = NU
                    Y3 = I
                    Y4 = K
                    CALL DRWA
                    IF (IS.EQ.1) GOTO 4100
                    Y1 = K
                    Y2 = NU
                    Y3 = I
                    Y4 = J
                    CALL DRWA
                    IF (IS.EQ.1) GOTO 4100
4060             CONTINUE
4070          CONTINUE
4080       CONTINUE
4090    CONTINUE
         WRITE(*,'(1X,''........................'')')
         WRITE(*,'(1X,''. R E C T I L I N E A R .'')')
         WRITE(*,'(1X,''........................''////)')
4100  CONTINUE
9999  END
*
*
*
```

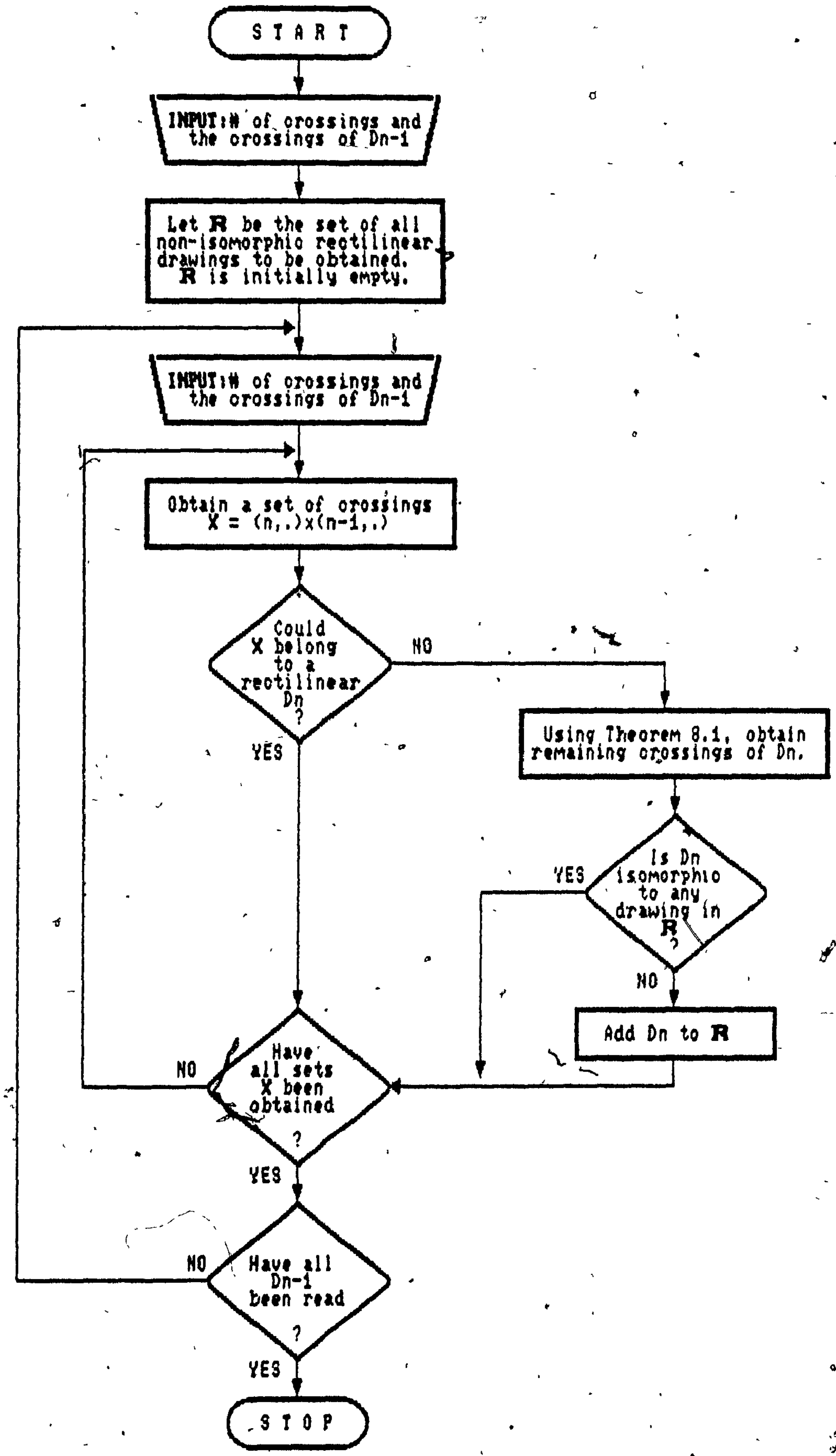
```
*****************   S U B R O U T I N E   *******************
*                                                             *
*                                                             *
*  Determine the number of times (EX,V)  crosses the arcs    *
*                                                             *
*  of the triangle T = (i,j,k,i).                             *
*                                                             *
*                                                             *
***************************************************************
*                                                             *
      SUBROUTINE TRGL
      IMPLICIT INTEGER*2 (A-Z)
      COMMON Y1,Y2,Y3,Y4,NX,X1,X2,X3,X4,XX,IS
      DIMENSION X1(100),X2(100),X3(100),X4(100)
      IF (Y3.GT.Y4) CALL INTRCHG(Y3,Y4)
      IF (Y1.GT.Y3) THEN
         CALL INTRCHG(Y1,Y3)
         CALL INTRCHG(Y2,Y4)
      ENDIF
      DO 10 S=1, NX
         IF (Y1.EQ.X1(S) .AND. Y2.EQ.X2(S) .AND.
     *       Y3.EQ.X3(S) .AND. Y4.EQ.X4(S)) XX = XX+1
10    CONTINUE
      RETURN
      END
*
*
```

```
****************   S U B R O U T I N E   ******************
*                                                         *
*                                                         *
*     Determine whether (v,i)x(j,k) while v is in Int T.  *
*                                                         *
*                                                         *
***********************************************************
*
      SUBROUTINE DRWA
      IMPLICIT INTEGER*2 (A-Z)
      COMMON Y1,Y2,Y3,Y4,NX,X1,X2,X3,X4,XX,IS
      DIMENSION X1(100),X2(100),X3(100),X4(100)
      IF (Y1.GT.Y2) CALL INTRCHG(Y1,Y2)
      IF (Y1.GT.Y3) THEN
         CALL INTRCHG(Y1,Y3)
         CALL INTRCHG(Y2,Y4)
      ENDIF
*
*  If (v,i)x(j,k) then drawing is non-rectilinear.
*
      IS = 0
      DO 10 S=1, NX
         IF (Y1.EQ.X1(S) .AND. Y2.EQ.X2(S) .AND.
     *       Y3.EQ.X3(S) .AND. Y4.EQ.X4(S))  THEN
                WRITE(*,'(1X,''('',I2,'','',I2,
     *           '')x('',I2,'','',I2,'')'')')Y1,Y2,Y3,Y4
                WRITE(*,'(1X,''........................'')')
                WRITE(*,'(1X,''. N O N - RECTILINEAR .'')')
                WRITE(*,'(1X,''........................''
     *                                    ////)')
                IS = 1
                RETURN
         ENDIF
10    CONTINUE
      RETURN
      END
*
*
*
****************   S U B R O U T I N E   ******************
*                                                         *
*            Interchanges the values of A and B           *
*                                                         *
***********************************************************
      SUBROUTINE INTRCHG(A,B)
      INTEGER*2 A,B,T
      T = A
      A = B
      B = T
      RETURN
      END
*
*
*
```

APPENDIX C.1

PROGRAM TO GENERATE THE NON-ISOMORPHIC RECTILINEAR DRAWINGS D_n USING THE NON-EQUIVALENT RECTILINEAR DRAWINGS D_{n-1}

I. GENERAL FLOWCHART


START
INPUT:# of crossings and the crossings of Dn-1
Let R be the set of all non-isomorphic rectilinear drawings to be obtained. R is initially empty.
INPUT:# of crossings and the crossings of Dn-1
Obtain a set of crossings X = (n,.)x(n-1,.)
Could X belong to a rectilinear Dn ?
NO
YES
Using Theorem 8.1, obtain remaining crossings of Dn.
Is Dn isomorphic to any drawing in R ?
YES
NO
Add Dn to R
Have all sets X been obtained ?
NO
YES
Have all Dn-1 been read ?
NO
YES
STOP

III- COMPUTER PROGRAM (FORTRAN 77)

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                       *
*   Let D_{n-1} be a rectilinear drawing of the complete *
*                                                       *
*   graph K_{n-1}.  We label the nodes of D_{n-1} such that *
*                                                       *
*   the node n-1 be on its convex hull and such that the*
*                                                       *
*   arcs (n-1,i) and (n-1,i+1) be adjacent to each other*
*                                                       *
*   with respect to node n-1, as shown in Fig.8.1.3 of  *
*                                                       *
*   Chapter 8.                                          *
*.......................................................*
*                                                       *
*   G I V E N  T H E  C R O S S I N G S  O F  E A C H   *
*                                                       *
*   O F  T H E  N O N - E Q U I V A L E N T             *
*                                                       *
*   R E C T I L I N E A R  D_{n-1} ,  A L O N G  W I T H *
*                                                       *
*   T H E I R  A P P R O P R I A T E  N O D E S '       *
*                                                       *
*   L A B E L S  A S  E X P L A I N E D  I N            *
*                                                       *
*   C H A P T E R  8 ,  T H I S  P R O G R A M  W I L L *
*                                                       *
*   G E N E R A T E  A L L  N O N-I S O M O R P H I C   *
*                                                       *
*   D R A W I N G S  D_n                                *
*                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
      IMPLICIT INTEGER*2 (A-Z)
* Let Dn be the drawing on hand, and                          *
* let Dn* be a drawing belonging to the set of                *
* non-isomorphic drawings already generated and against       *
* which we compare Dn for isomorphism.                        *
*                                                             *
* X1(.),X2(.),X3(.),X4(.) = crossings of Dn                   *
* SX1(.),SX2(.),SX3(.),SX4(.) = crossings of Dn*              *
* Z1(.),Z2(.),Z3(.),Z4(.) = crossings of Dn-1                 *
      DIMENSION X1(210),X2(210),X3(210),X4(210)
      DIMENSION SX1(210),SX2(210),SX3(210),SX4(210)
      DIMENSION Z1(210),Z2(210),Z3(210),Z4(210)
* NODES(.),RSPND(.),RSPAR(.) = nodes, nodes                   *
*                              responsibilities and           *
*                              arcs responsibilities          *
*                              respectively, for Dn           *
* SNODES(.),SRSPND(.),SRSPAR(.) = nodes, nodes                *
*                                 responsibilities and        *
*                                 arcs responsibilities       *
*                                 respectively, for Dn*       *
      DIMENSION NODES(10),SNODES(10)
      DIMENSION RSPND(10),SRSPND(10)
      DIMENSION RSPAR(45),SRSPAR(45)
*                                                             *
*                                                             *
* INDX(.) and IARC(.) are used in the process of              *
* generating the arcs (n,i) of drawing Dn.                    *
*                                                             *
* INDX(.) will contain the value 1 if arc (n,i) crosses       *
*         arc (n-1,j), and the value 0 otherwise.             *
* IARC(.) will contain the largest values that the            *
*         elements of INDX(.) could attain, namely 1.         *
*                                                             *
      DIMENSION INDX(28),IARC(28),SINDX(28)
*                                                             *
*                                                             *
*                                                             *
* MN(.) and MX(.) are used in the process of determining      *
* whether the sub-drawing consisting of the nodes r,s,t,      *
* n, n-1 and their corresponding arcs, is rectilinear.        *
* MN(.) will contain nodes r,s and t.                         *
* MX(.) will contain the maximum value for each of the        *
*       nodes r, s and t.                                     *
*                                                             *
      DIMENSION MN(3),MX(3)
*                                                             *
*                                                             *
*                                                             *
* PERM(.) = the different labels of the nodes of Dn-1         *
*                                                             *
      DIMENSION PERM(10)
*                                                             *
*                                                             *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                       *
*  Input the dimension, n,                              *
*       the number of crossings, and the crossings of   *
*       D_{n-1}.                                        *
*                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        OPEN (2,FILE='C:NONISO',STATUS='NEW')
        NDRAW = 0
        WRITE(*,'(1X,''  INPUT N'')')
        READ (*,501) N
        IF (N.EQ.0) GOTO 9999
  100   WRITE(*,'(1X,///
       *  '' # OF REGIONS,# OF CROSSINGS'')')
        READ (*,501) PC,NXO
        IF (PC.EQ.0 .OR. NXO.EQ.0) GOTO 9999
        NN1 = N*(N-1)/2
        N3N2 = (N-3)*(N-2)/2
        N1 = N-1
        WRITE(*,'(1X,''INPUT CROSSINGS'')')
        DO 1000 I=1,NXO
           READ (*,501) X1(I),X2(I),X3(I),X4(I)
           IF (PC.GT.1) THEN
              Z1(I) = X1(I)
              Z2(I) = X2(I)
              Z3(I) = X3(I)
              Z4(I) = X4(I)
           ENDIF
 1000   CONTINUE
 1002   DO 1004 I=1 , N3N2
           IARC(I) = 1
           INDX(I) = 0
 1004   CONTINUE
 1005   NX =NXO
        K = 0
        DO 1020 I1=1 , N-3
           DO 1010 I2=I1+1 , N-2
              K = K+1
              IF (INDX(K) .NE. 0) THEN
                 NX = NX+1
                 X1(NX) = I1
                 X2(NX) = N
                 X3(NX) = I2
                 X4(NX) = N1
              ENDIF
 1010      CONTINUE
 1020   CONTINUE
*
*
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                         *
* The set of crossings (n,.) x (n-1,.) is checked to      *
* determine whether it could belong to a rectilinear      *
* drawing.                                                *
*                                                         *
* If this set cannot belong to a rectilinear drawing,     *
* then it is ignored. Otherwise, the remaining crossings*
* of the drawing are obtained using Theorem 8.1 of        *
* Chapter 8.                                              *
*                                                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                         *
*                                                         *
      DO 1030 I=1 , 3
         MX(I)=N+I-5
         MN(I)=I
1030  CONTINUE
1035  R = MN(1)
      S = MN(2)
      T = MN(3)
      A = 0
      B = 0
      C = 0
      D = 0
      DO 1040 I=1,NX
         IF (R.EQ.X1(I) .AND. T.EQ.X2(I) .AND.
     *         S.EQ.X3(I) .AND. N1.EQ.X4(I))  C=1
         IF (R.EQ.X1(I) .AND. N.EQ.X2(I) .AND.
     *         S.EQ.X3(I) .AND. N1.EQ.X4(I))  A=1
         IF (R.EQ.X1(I) .AND. N.EQ.X2(I) .AND.
     *         T.EQ.X3(I) .AND. N1.EQ.X4(I))  B=1
         IF (S.EQ.X1(I) .AND. N.EQ.X2(I) .AND.
     *         T.EQ.X3(I) .AND. N1.EQ.X4(I))  D=1
1040  CONTINUE
*                                                         *
*                                                         *
*                                                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
* Checking whether the crossings belong to a rectilinear*
* drawing.                                                      *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
*                                                               *
      IF ((A.EQ.0 .AND. B.EQ.0 .AND. C.EQ.1 .AND. D.EQ.1)
     *                  .OR.
     *     (A.EQ.0 .AND. B.EQ.1 .AND. C.EQ.0 .AND. D.EQ.0)
     *                  .OR.
     *     (A.EQ.0 .AND. B.EQ.1 .AND. C.EQ.1 .AND. D.EQ.0)
     *                  .OR.
     *     (A.EQ.0 .AND. B.EQ.1 .AND. C.EQ.1 .AND. D.EQ.1)
     *                  .OR.
     *     (A.EQ.1 .AND. B.EQ.0 .AND. C.EQ.0 .AND. D.EQ.0)
     *                  .OR.
     *     (A.EQ.1 .AND. B.EQ.0 .AND. C.EQ.0 .AND. D.EQ.1)
     *                  .OR.
     *     (A.EQ.1 .AND. B.EQ.0 .AND. C.EQ.1 .AND. D.EQ.1)
     *                  .OR.
     *   (A.EQ.1 .AND. B.EQ.1 .AND. C.EQ.0 .AND. D.EQ.0))
     *   GOTO 2112
*                                                               *
*                                                               *
*                                                               *
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
*        Obtaining the remaining of the crossings.                              *
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
         A = 0
         B = 0
         C = 0
         D = 0
         DO 1050 P=1,NX
            IF (X1(P).EQ.R .AND.  X2(P).EQ.T .AND.
     *          X3(P).EQ.S .AND.  X4(P).EQ.N1)      A = 1
            IF (X1(P).EQ.S .AND.  X2(P).EQ.N .AND.
     *          X3(P).EQ.T .AND.  X4(P).EQ.N1)      B = 1
            IF (X1(P).EQ.R .AND.  X2(P).EQ.N .AND.
     *          X3(P).EQ.S .AND.  X4(P).EQ.N1)      C = 1
            IF (X1(P).EQ.R .AND.  X2(P).EQ.N .AND.
     *          X3(P).EQ.T .AND.  X4(P).EQ.N1)      D = 1
1050     CONTINUE
         IF ((A.EQ.0 .AND.  B.EQ.0) .OR.
     *      (A.EQ.1 .AND.  B.EQ.1)) THEN
            NX =NX+1
            X1(NX) = R
            X2(NX) = T
            X3(NX) = S
            X4(NX) = N
            GOTO 1055
          ENDIF
         IF ((C.EQ.1 .AND.  D.EQ.0) .OR.
     *      (C.EQ.0 .AND.  D.EQ.1)) THEN
            NX =NX+1
            X1(NX) = R
            X2(NX) = N
            X3(NX) = S
            X4(NX) = T
         ENDIF
1055     IF (MN(3).EQ.MX(3)) THEN
            LST = 2
            GOTO 1067
         ENDIF
1065     MN(3) = MN(3) + 1
         GOTO 1035
1067     IF (MN(LST).EQ.MX(LST)) THEN
            IF  (LST.GT.1) THEN
                 LST = LST-1
                 GOTO 1067
            ELSE
                 GOTO 1095
            ENDIF
         ENDIF
1085     MN(LST) = MN(LST) + 1
         DO 1090 I=LST+1 , 3
            MN(I) = MN(I-1) + 1
1090     CONTINUE
         GOTO 1035
*
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                       *
*       Arranging each crossing (a,b) x (c,d) such      *
*       that a < b , c < d , and  a < c ;  and          *
*       sorting the crossings in ascending order.       *
*                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
1095      CALL ARRNGX(NX,X1,X2,X3,X4)
          CALL SORTX(NX,X1,X2,X3,X4)
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                       *
*      Obtaining nodes and arcs responsibilities        *
*                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
          CALL NODRSP(N,NX,X1,X2,X3,X4,RSPND,NODES)
          CALL ARCRSP(N,NN1,NX,X1,X2,X3,X4,RSPAR)
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                       *
*     File NONISO on drive c will contain the           *
*     non-isomorphic drawings.  Each of these is        *
*     compared against the drawing on hand.  If         *
*     it is isomorphic to any of the drawings           *
*     already stored in NONISO then it is ignored,      *
*     otherwise it is stored in NONISO and its          *
*     crossings are displayed.                          *
*                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      IF (NDRAW.GT.0) THEN
         OPEN (2,FILE='C:NONISO')
         ISO = 0
         DO 2070 I=1, NDRAW
            READ(2,501) SNX
            READ (2,501)
     *            (SX1(J),SX2(J),SX3(J),SX4(J),J=1,SNX)
            READ (2,501) (SRSPND(J), J=1, N)
            READ (2,501) (SNODES(J), J=1, N)
            READ (2,501) (SRSPAR(J), J=1, NN1)
            READ (2,501) (SINDX(J), J=1, N3N2)
            IF (SNX.EQ.NX) THEN
               DO 2050 J=1, N
                  IF (SRSPND(J).NE.RSPND(J)) GOTO 2070
2050           CONTINUE
               DO 2060 J=1, NN1
                  IF (SRSPAR(J).NE.RSPAR(J)) GOTO 2070
2060           CONTINUE
               CALL ISOMOR
     *            (N,NX,RSPND,NODES,SNODES,X1,X2,X3,X4,
     *                          SX1,SX2,SX3,SX4,ISO)
               IF (ISO.EQ.1) GOTO 2112
            ENDIF
2070     CONTINUE
      ENDIF
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                              O U T P U T                           *
** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
      WRITE(2,501) NX
      WRITE(2,501)
     *        (X1(J),X2(J),X3(J),X4(J), J=1,NX)
      WRITE(2,501) (RSPND(J), J=1, N)
      WRITE(2,501) (NODES(J), J=1, N)
      WRITE(2,501) (RSPAR(J), J=1, NN1)
      WRITE(2,501) (INDX(J), J=1, N3N2)
      CLOSE(2)
      NDRAW = NDRAW + 1
      WRITE(*,601) NDRAW, (INDX(I),I=1, N3N2)
      WRITE(*,602)
         DO 2110 I=1, NX
            WRITE(*,603) I, X1(I),X2(I),X3(I),X4(I)
2110     CONTINUE
2112     IF (INDX(N3N2).GE.IARC(N3N2)) THEN
            LAST = N3N2
            GOTO 2125
         ENDIF
         INDX(N3N2) = INDX(N3N2) + 1
         GOTO 1005
2117     IF (INDX(LAST).LT.IARC(LAST)) GOTO 2135
2125     IF (LAST.EQ.1) GOTO 2145
         LAST = LAST - 1
         GOTO 2117
2135     INDX(LAST) = INDX(LAST) + 1
         DO 2137 I=LAST+1 , N3N2
            INDX(I) = 0
2137     CONTINUE
         GOTO 1005
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                    *
*  Changing the region in which the n-th node is placed.             *
*                                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
2145     IF (PC.NE.1) THEN
            PC = PC -1
            WRITE(*,'(1X,///''  INPUT NEW LABELS'')')
            READ(*,501) (PERM(I), I=1, N-1)
            DO 2170 I=1 , NXO
               DO 2160 J=1,N-1
                  IF (Z1(I).EQ.J) X1(I) = PERM(J)
                  IF (Z2(I).EQ.J) X2(I) = PERM(J)
                  IF (Z3(I).EQ.J) X3(I) = PERM(J)
                  IF (Z4(I).EQ.J) X4(I) = PERM(J)
2160           CONTINUE
2170        CONTINUE
```

```
                DO 2180 I=1 , NXO
                   IF (X1(I).GT.X2(I))
     *                   CALL INTRCHG(X1(I),X2(I))
                   IF (X3(I).GT.X4(I))
     *                   CALL INTRCHG(X3(I),X4(I))
                   IF (X1(I).GT.X3(I)) THEN
                      CALL INTRCHG(X1(I),X3(I))
                      CALL INTRCHG(X2(I),X4(I))
                   ENDIF
 2180           CONTINUE
                GOTO 1002
             ENDIF
             GOTO 100
 501    FORMAT(28I2)
 601    FORMAT('0',' D R A W I N G # ',I4,/
     *          ===================='/1X,28(I2))
 602    FORMAT(1X,///'   C R O S S I N G S '/
     *          -----------------'/)
 603    FORMAT(' ',I2,'   (',I2,',',I2,') x (',
     *                                 I2,',',I2,')')
 9999   END
```

```
*********************************************************
****************  S U B R O U T I N E  ****************
*********************************************************
*                                                       *
* Nodes responsibilities are calculated here.  These are*
* sorted in descending order and their corresponding   *
* nodes arre rearranged accordingly.                    *
*                                                       *
*********************************************************
      SUBROUTINE NODRSP(N,NX,X1,X2,X3,X4,RSPND,NODES)
      IMPLICIT INTEGER*2 (A-Z)
      DIMENSION X1(NX),X2(NX),X3(NX),X4(NX)
      DIMENSION RSPND(N),NODES(N)
      DO 10 J=1, N
         RSPND(J) = 0
10    CONTINUE
      DO 30 I=1, NX
         DO 20 J=1, N
            IF (X1(I).EQ.J .OR.
     *              X2(I).EQ.J .OR.
     *              X3(I).EQ.J .OR.
     *              X4(I).EQ.J)    RSPND(J) = RSPND(J)+1
20       CONTINUE
30    CONTINUE
      DO 40 I=1, N
         NODES(I) = I
40    CONTINUE
      DO 60 I=1, N-1
         DO 50 J=I+1, N
            IF (RSPND(I).LT.RSPND(J)) THEN
               CALL INTRCHG(RSPND(I),RSPND(J))
               CALL INTRCHG(NODES(I),NODES(J))
            ENDIF
50       CONTINUE
60    CONTINUE
      RETURN
      END
```

```
****************************************************************
****************  S U B R O U T I N E  *************************
****************************************************************
*                                                              *
*  Arcs responsibilities are calculated then arranged in*
*  descending order.                                           *
*                                                              *
****************************************************************
      SUBROUTINE ARCRSP(N,M,NX,X1,X2,X3,X4,RSPAR)
      IMPLICIT INTEGER*2 (A-Z)
      DIMENSION X1(NX),X2(NX),X3(NX),X4(NX)
      DIMENSION RSPAR(M)
      DO 10 J=1, M
         RSPAR(J) = 0
10    CONTINUE
      DO 40 I=1, NX
         JK = 0
         DO 30 J=1, N-1
            DO 20 K=J+1, N
               JK = JK+1
               IF ((X1(I).EQ.J .AND. X2(I).EQ.K) .OR.
     *             (X3(I).EQ.J .AND. X4(I).EQ.K))
     *               RSPAR(JK) = RSPAR(JK)+1
20          CONTINUE
30       CONTINUE
40    CONTINUE
      DO 60 I=1, M -1
         DO 50 J=I+1, M
            IF (RSPAR(I).LT.RSPAR(J))
     *               CALL INTRCHG(RSPAR(I),RSPAR(J))
50       CONTINUE
60    CONTINUE
      RETURN
      END
```

```
***************************************************************
****************  S U B R O U T I N E  ************************
***************************************************************
*                                                             *
*     A crossing (a,b)x(c,d) is arranged such that            *
*            a < b ,  c < d  and  a < c                       *
***************************************************************
      SUBROUTINE ARRNGX(NX,X1,X2,X3,X4)
      IMPLICIT INTEGER*2 (A-Z)
      DIMENSION X1(NX),X2(NX),X3(NX),X4(NX)
      DO 10 I=1, NX
         IF (X1(I).GT.X2(I)) CALL INTRCHG(X1(I),X2(I))
         IF (X3(I).GT.X4(I)) CALL INTRCHG(X3(I),X4(I))
         IF (X1(I).GT.X3(I)) THEN
            CALL INTRCHG(X1(I),X3(I))
            CALL INTRCHG(X2(I),X4(I))
         ENDIF
10    CONTINUE
      RETURN
      END
```

```
*********************************************************
*****************  S U B R O U T I N E  *****************
*********************************************************
*                                                       *
* The crossings of a draving are sorted in ascending    *
* order.                                                *
*                                                       *
*********************************************************
      SUBROUTINE SORTX(NX,X1,X2,X3,X4)
      IMPLICIT INTEGER*2 (A-Z)
      DIMENSION X1(NX),X2(NX),X3(NX),X4(NX)
      DO 30 I=1, NX-1
         DO 20 J=I+1, NX
            IF (X1(I).LT.X1(J)) GOTO 20
            IF (X1(I).GT.X1(J)) GOTO 10
            IF (X2(I).LT.X2(J)) GOTO 20
            IF (X2(I).GT.X2(J)) GOTO 12
            IF (X3(I).LT.X3(J)) GOTO 20
            IF (X3(I).GT.X3(J)) GOTO 14
            IF (X4(I).LT.X4(J)) GOTO 20
            IF (X4(I).GT.X4(J)) GOTO 16
10          CALL INTRCHG(X1(I),X1(J))
12          CALL INTRCHG(X2(I),X2(J))
14          CALL INTRCHG(X3(I),X3(J))
16          CALL INTRCHG(X4(I),X4(J))
20       CONTINUE
30    CONTINUE
      RETURN
      END
*
*
```

```
*****************************************************
*****************  S U B R O U T I N E  *****************
*****************************************************
*                                                   *
*     Two drawings are compared for isomorphism.  Node   *
*     responsibilities are used to reduce the number    *
*     of comparisons.                               *
*                                                   *
*     Variable ISO takes the value 1 whenever the two   *
*     drawings are isomorphic.  Otherwise it keeps its  *
*     original value of zero.                       *
*                                                   *
*****************************************************
      SUBROUTINE ISOMOR(N,NX,RSPND,NODES,SNODES,
     *                          X1,X2,X3,X4,
     *                          SX1,SX2,SX3,SX4,ISO)
      IMPLICIT INTEGER*2 (A-Z)
      DIMENSION RSPND(N),NODES(N),SNODES(N)
      DIMENSION X1(NX),X2(NX),X3(NX),X4(NX)
      DIMENSION SX1(NX),SX2(NX),SX3(NX),SX4(NX)
*                                                   *
*PR(.),PRM(.),MP(.,.),MINI(.),MAXI(.) are usedtogenerate*
*                                     new nodes' labels*
*                                                   *
      DIMENSION PR(6),PRM(6),MP(6,6),MINI(9),MAXI(9)
*                                                   *
*  Y1(.),Y2(.),Y3(.),Y4(.) = crossings after relabelling*
*                            the nodes              *
*                                                   *
      DIMENSION Y1(15),Y2(15),Y3(15),Y4(15)
      DO 100 L=1, N
         PR(L) = 0
100   CONTINUE
      J = 1
      PR(1) = 1
      DO 110 L=2, N
         IF (RSPND(L-1).GT.RSPND(L)) THEN
            J = J+1
            PR(J) = 1
            GOTO 110
         ENDIF
         PR(J) = PR(J)+1
110   CONTINUE
      S = 0
      R = 0
      DO 130 I=1, N
         DO 120 J=1, N
            MP(I,J) = 0
120      CONTINUE
130   CONTINUE
```

```
      DO 220 J=1, N
         S = S+PR(J-1)
         IF (PR(J).NE.0) THEN
            DO 210 K=1, PR(J)
               R = R+1
               DO 200 L=1, PR(J)
                  MP(NODES(L+S),K) = SNODES(R)
200            CONTINUE
210         CONTINUE
         ENDIF
220   CONTINUE
      DO 320 I=1, N
         PI = 0
         DO 300 J=1, N
            IF (MP(I,J).EQ.0) GOTO 310
            PI = PI+1
300      CONTINUE
310      MINI(I) = 1
         MAXI(I) = PI
320   CONTINUE
      IS = 1
330   DO 410 RW=IS, N
         PRMI = MP(RW,MINI(RW))
         PRM(RW) = PRMI
         DO 400 J=1, RW-1
            IF (PRM(J).EQ.PRMI) GOTO 700
400      CONTINUE
410   CONTINUE
      DO 510 K=1, NX
         DO 500 J=1, N
            IF (X1(K).EQ.J) Y1(K) =PRM(J)
            IF (X2(K).EQ.J) Y2(K) =PRM(J)
            IF (X3(K).EQ.J) Y3(K) =PRM(J)
            IF (X4(K).EQ.J) Y4(K) =PRM(J)
500      CONTINUE
510   CONTINUE
      CALL ARRNGX(NX,Y1,Y2,Y3,Y4)
      CALL SORTX(NX,Y1,Y2,Y3,Y4)
      DO 600 I=1, NX
         IF (SX1(I).NE.Y1(I) .OR. SX2(I).NE.Y2(I) .OR.
     *       SX3(I).NE.Y3(I) .OR. SX4(I).NE.Y4(I))
     *                                  GOTO 700
600   CONTINUE
      ISO = 1
      GOTO 900
```

```
700    IS = RW
       IF (MINI(RW).GE.MAXI(RW)) THEN
          LST = RW
          GOTO 730
       ENDIF
710    MINI(RW) = MINI(RW)+1
       GOTO 330
720    IF (MINI(LST).LT.MAXI(LST)) GOTO 740
730    IF (LST.EQ.1) GOTO 900
       LST = LST-1
       IS = IS-1
       GOTO 720
740    MINI(LST) = MINI(LST)+1
       DO 750 L=LST+1, N
          MINI(L) =1
750    CONTINUE
       GOTO 330
900    RETURN
       END
*
*
```

```
*****************************************************************
**************** S U B R O U T I N E ****************
*****************************************************************
*                                                               *
*        Interchange the values of two variables                *
*                                                               *
*****************************************************************
      SUBROUTINE INTRCHG(A,B)
      INTEGER*2 A,B,T
      T = A
      A = B
      B = T
      RETURN
      END
```

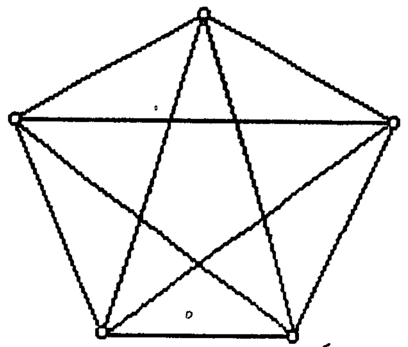
APPENDIX C.2

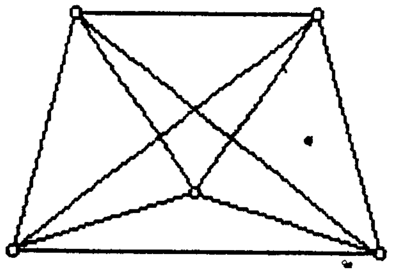
ALL THE NON-ISOMORPHIC

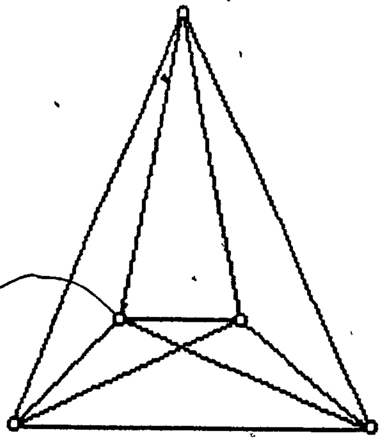
RECTILINEAR DRAWINGS D_n OF K_n

FOR $n = 5, 6, 7$.

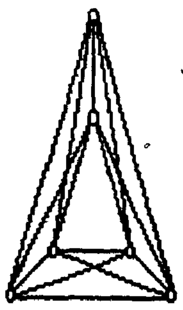
n = 5

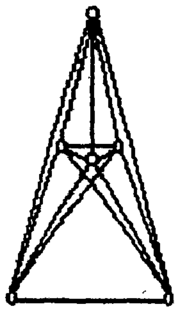

(1)

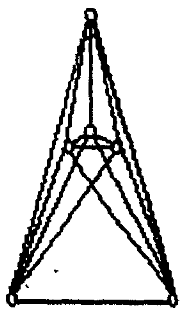

(2)

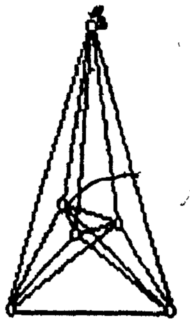

(3)

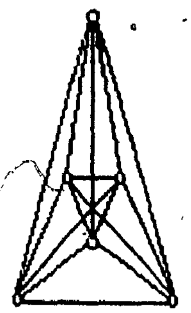
n = 6

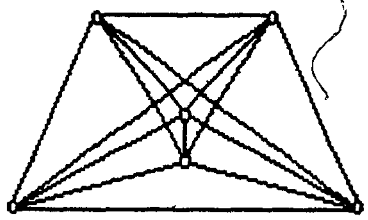

(1)

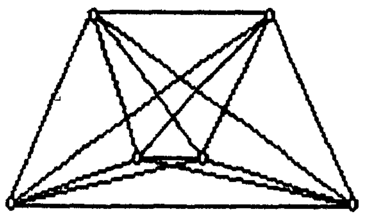

(2)

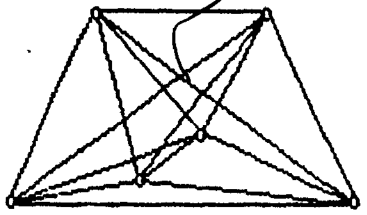

(3)

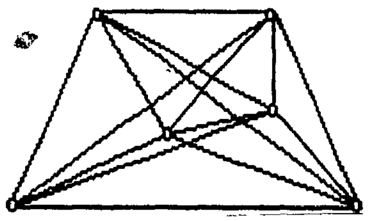

(4)

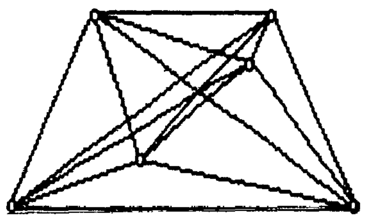

(5)

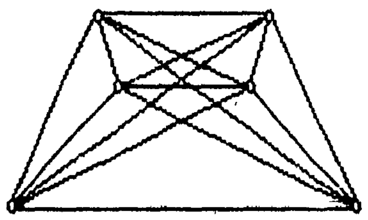

(6)

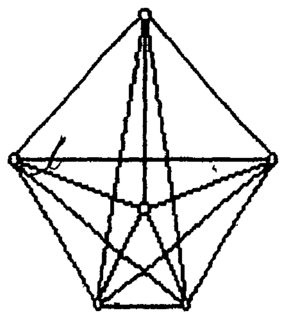

(7)

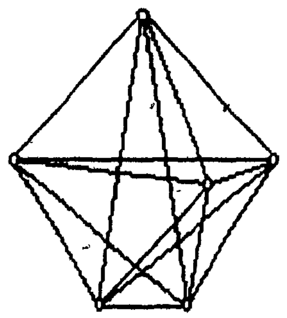

(8)

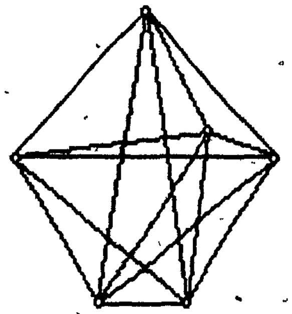

(9)

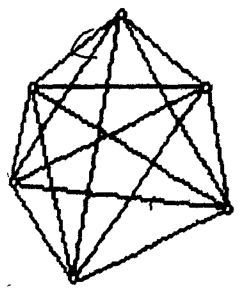

(10)

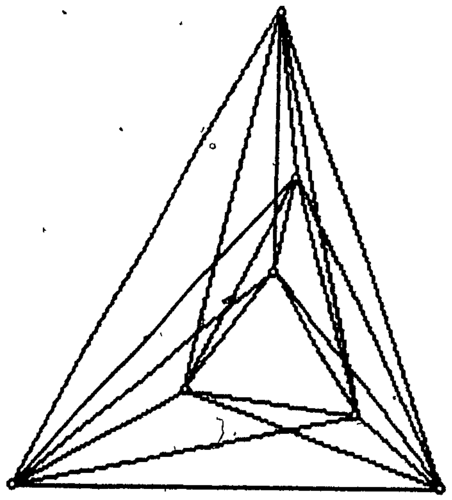

(11)

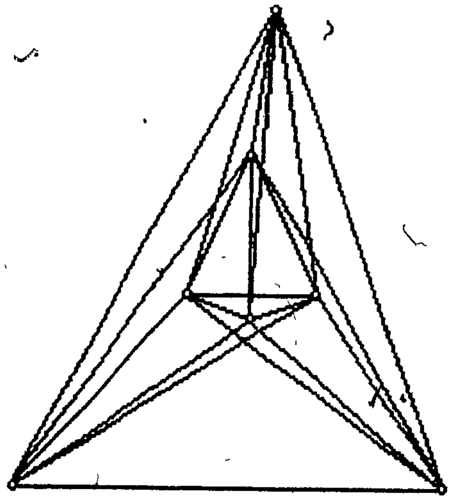

(12)

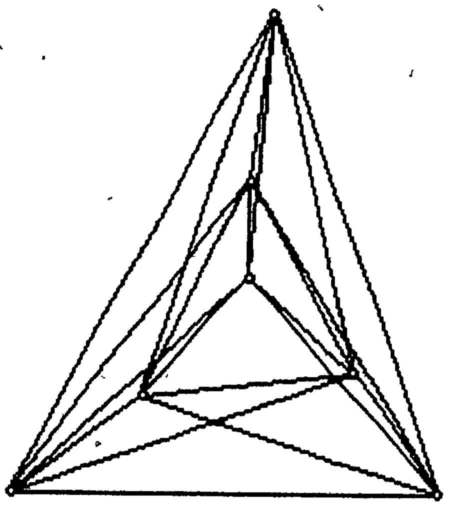

(13)

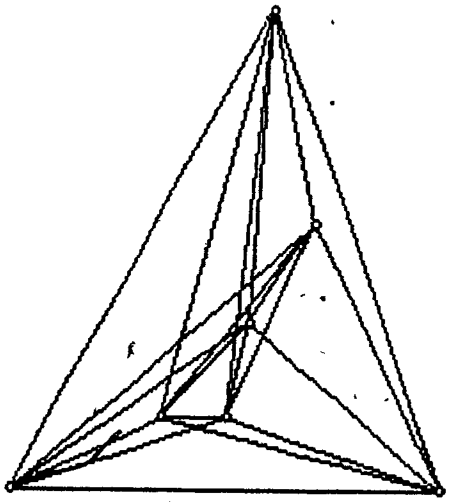

(14)

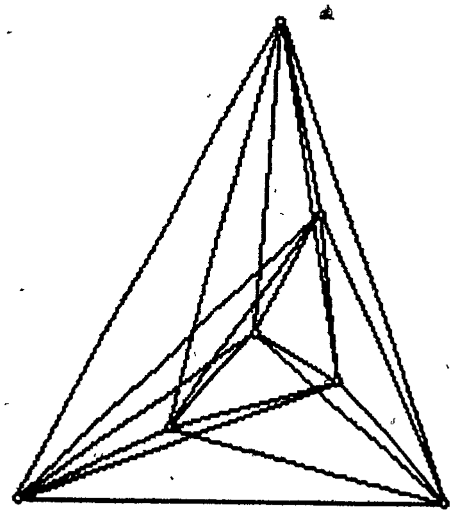

(15)

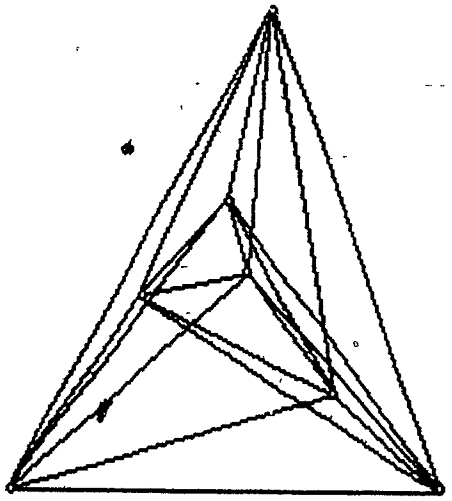

(1)

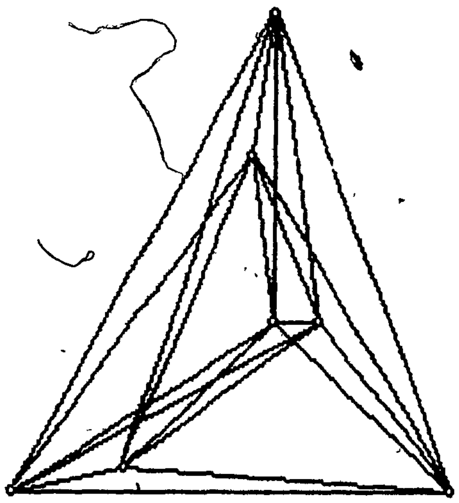

(2)

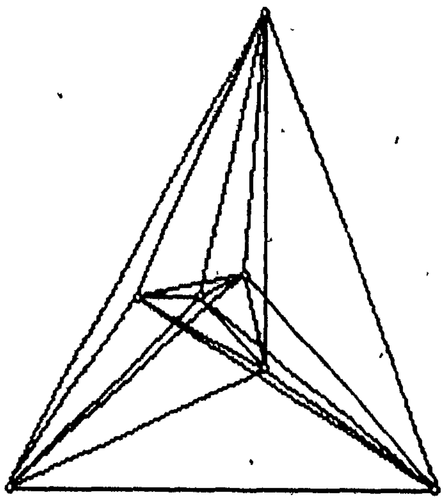

(3)

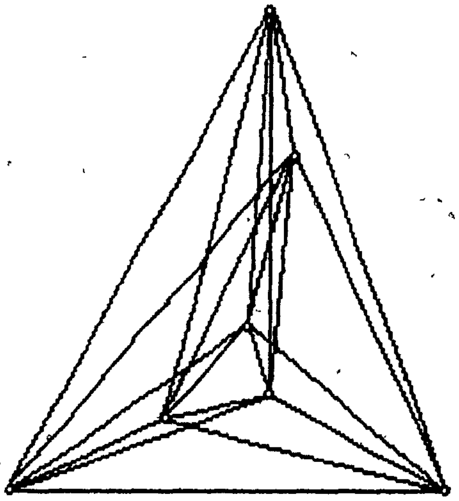

(4)

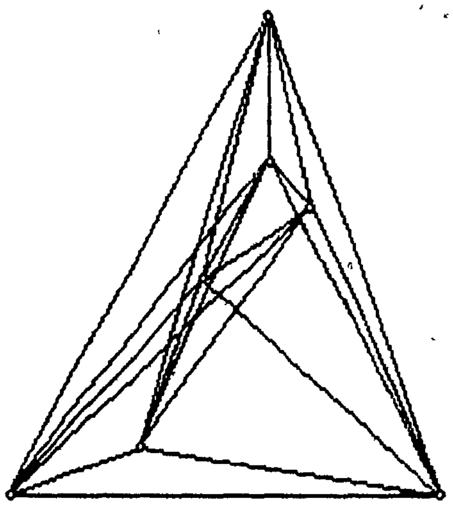

(5)

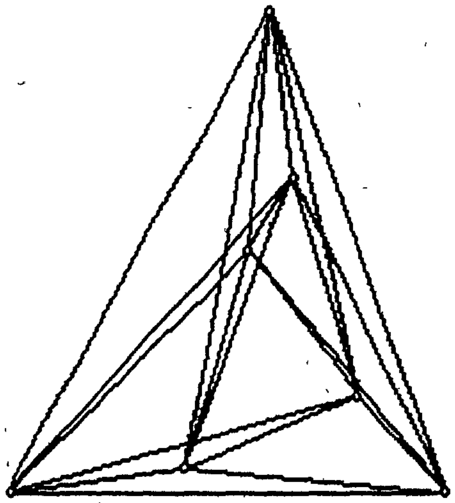

(6)

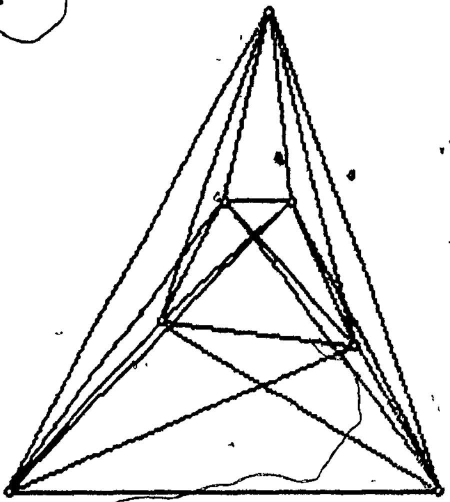

(7)

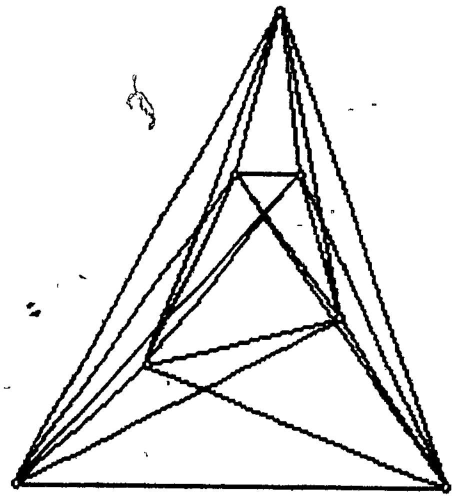

(8)

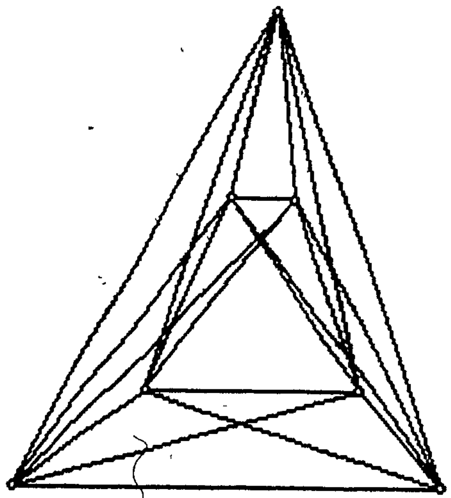

(9)

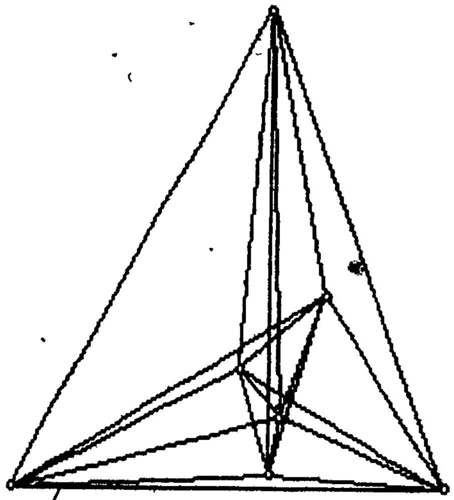

(10)

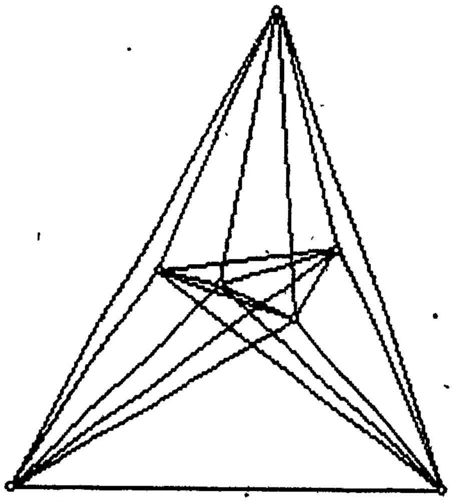

(11)

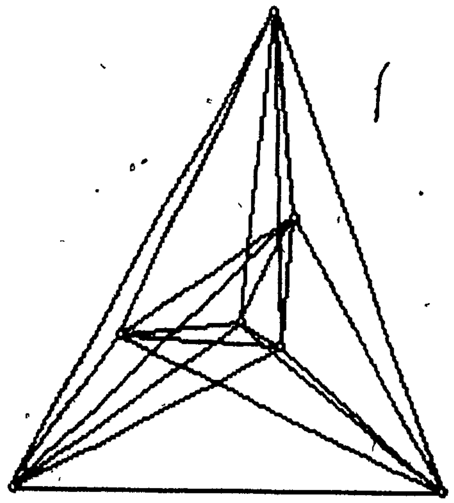

(12)

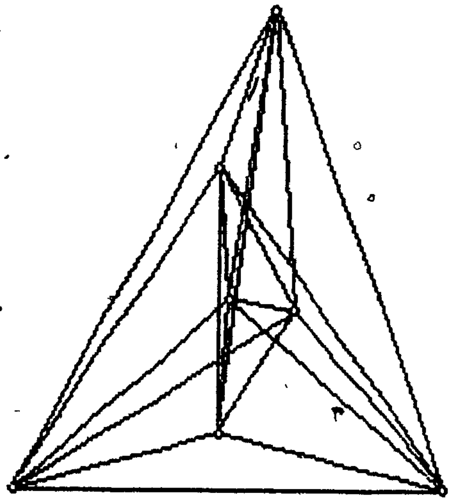

(13)

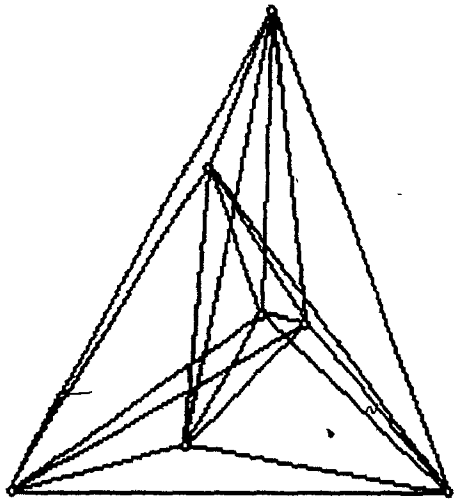

(14)

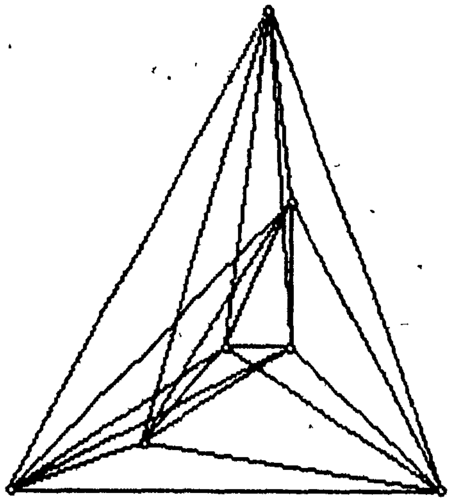

(15)

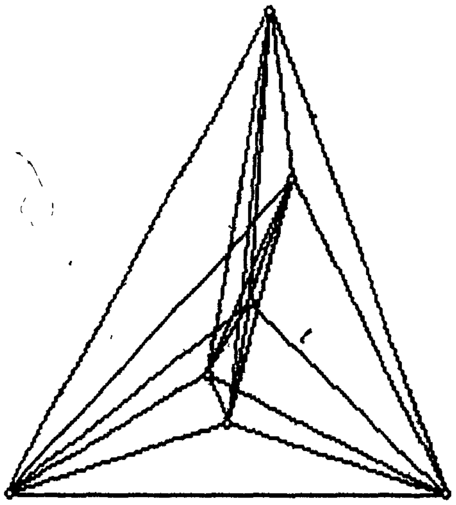

(16)

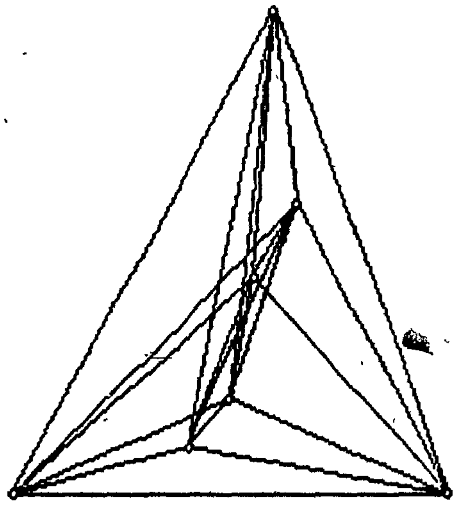

(17)

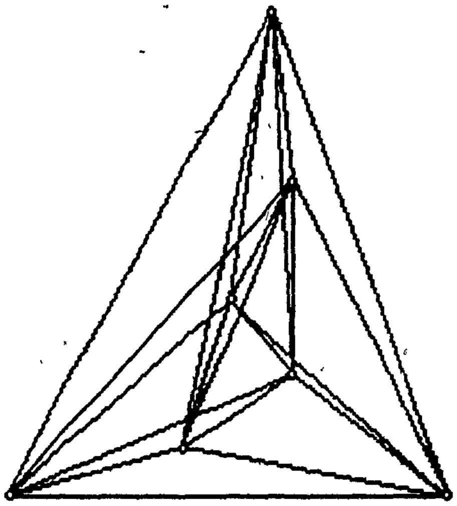

(18)

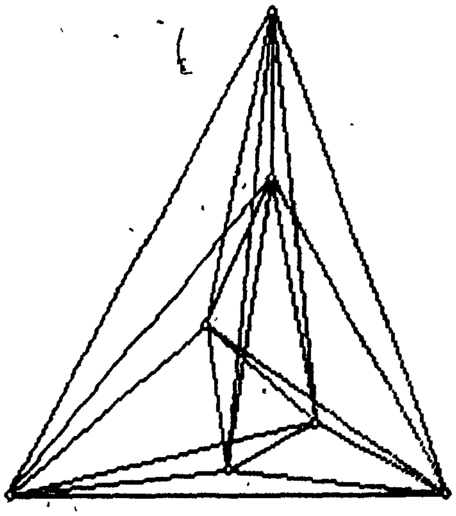

(19)

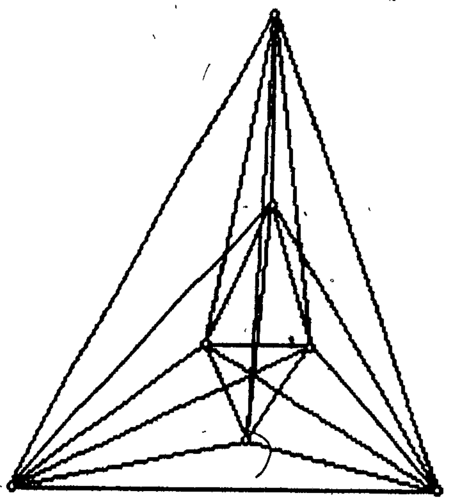

(20)

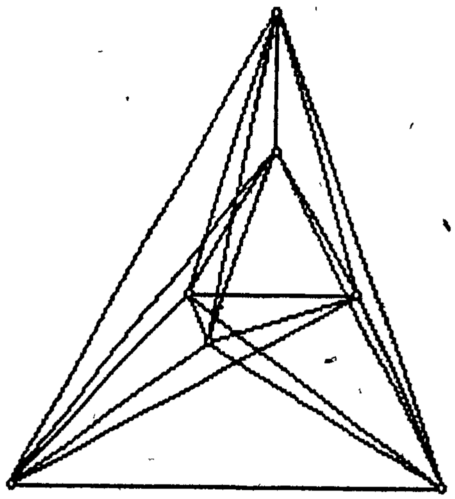

(21)

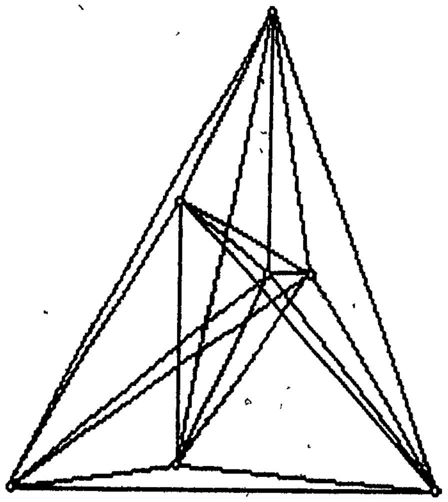

(22)

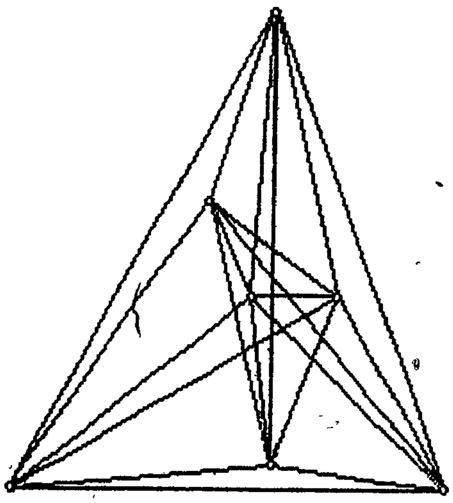


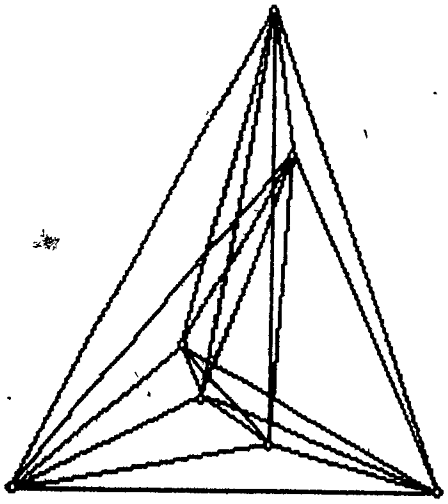


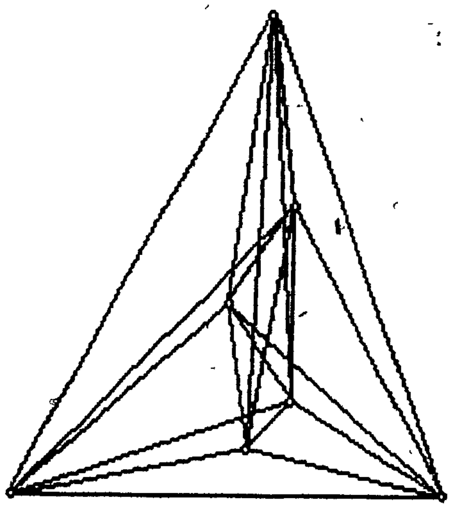

(25)

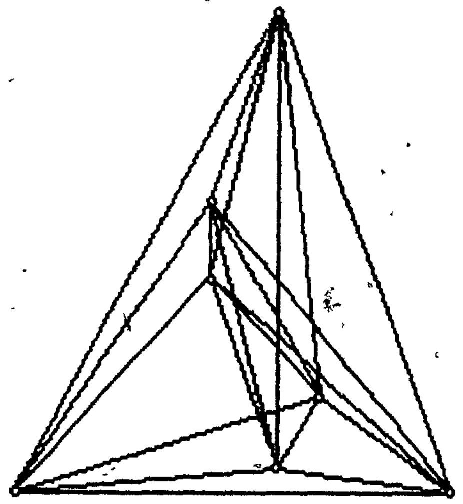

(26)

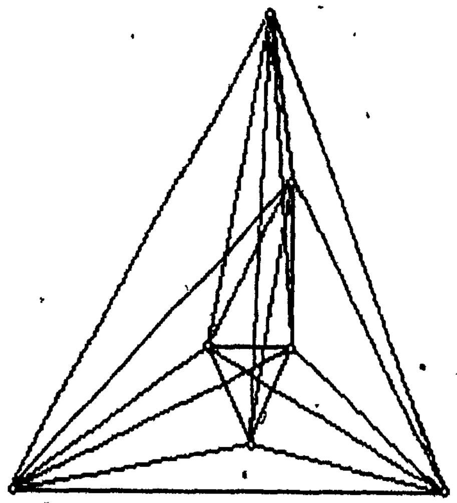

(27)

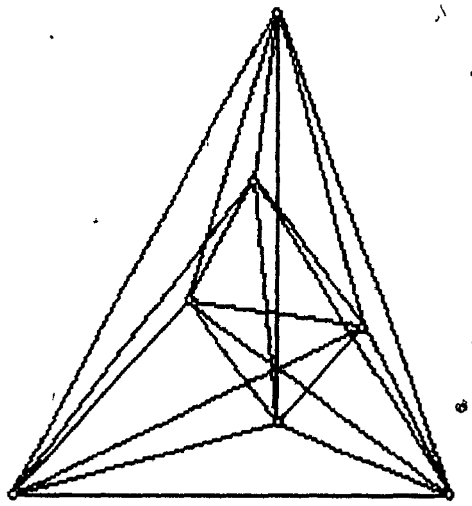

(28)

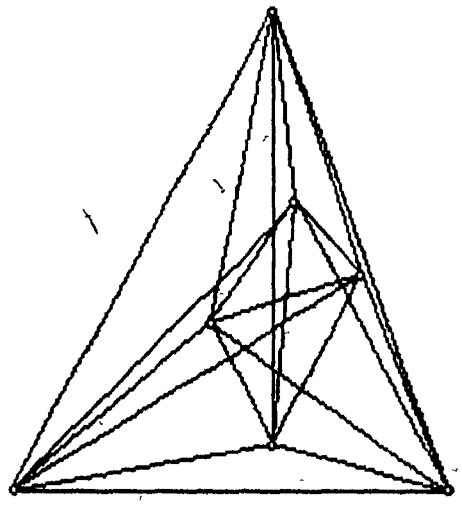

(29)

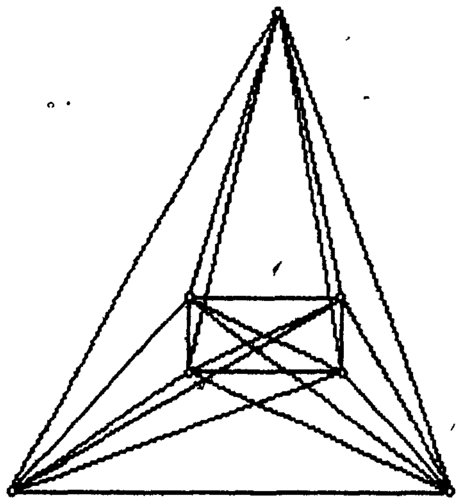

(30)

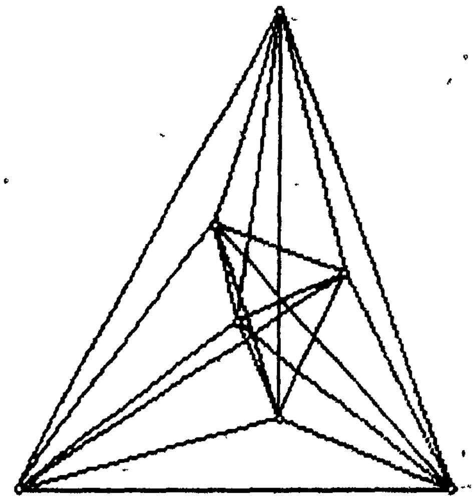

(31)

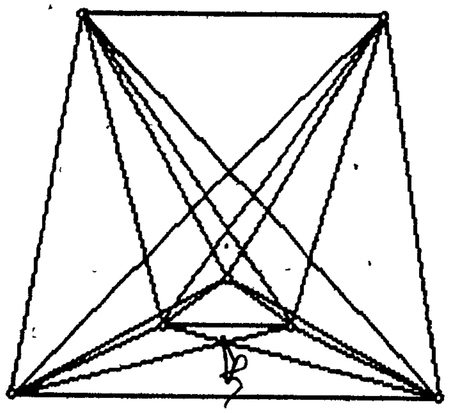

(32)

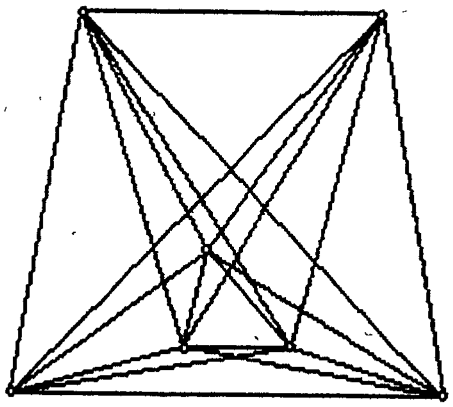

(33)

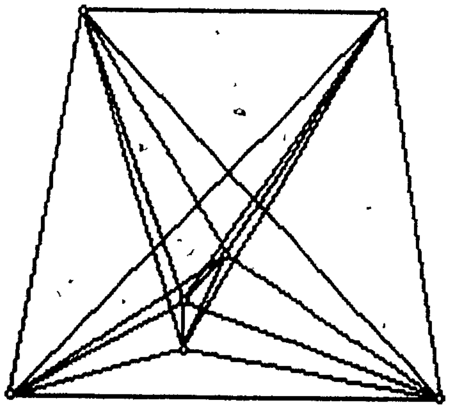

(34)

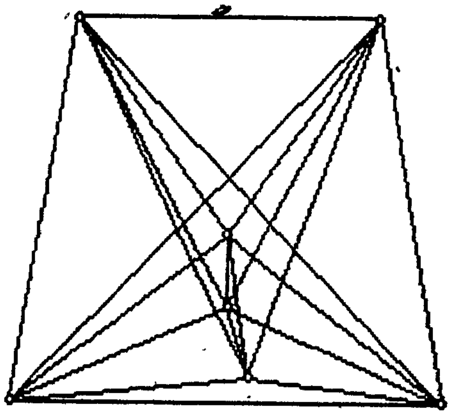

(35)

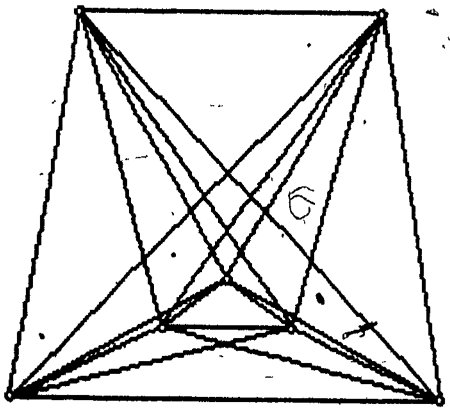

(36)

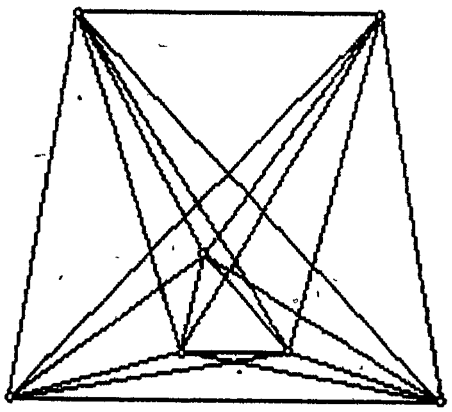

(37)

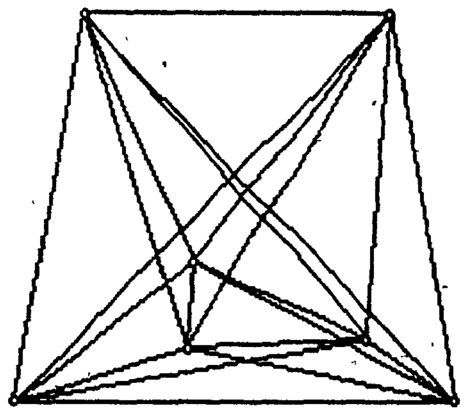

(38)

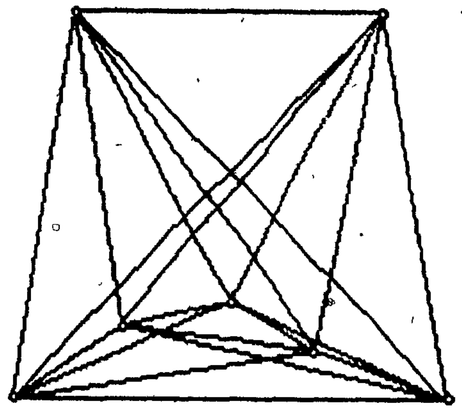

(39)

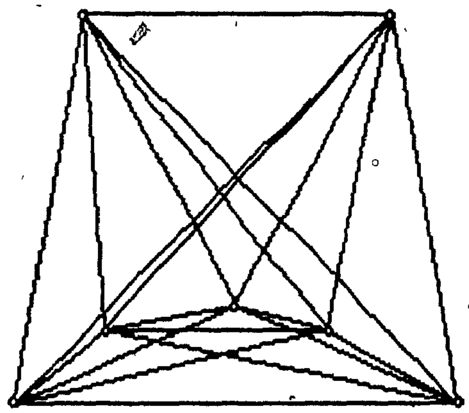

(40)

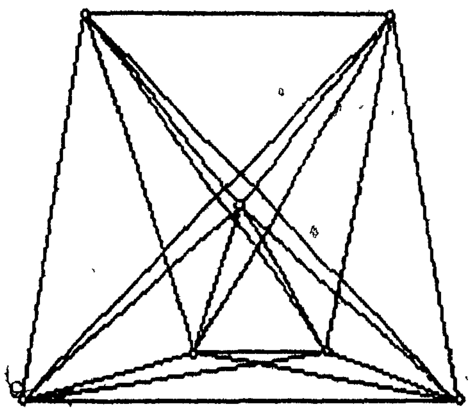

(41)

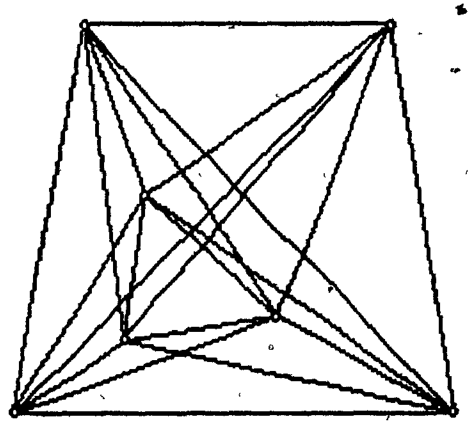

(42)

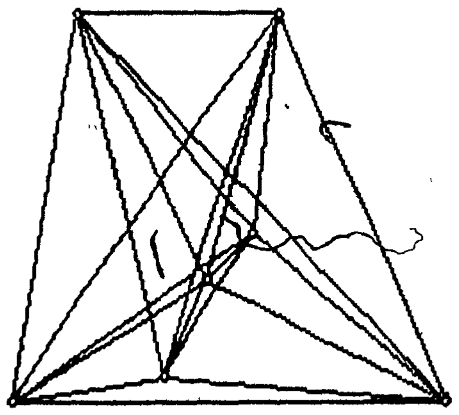

(43)

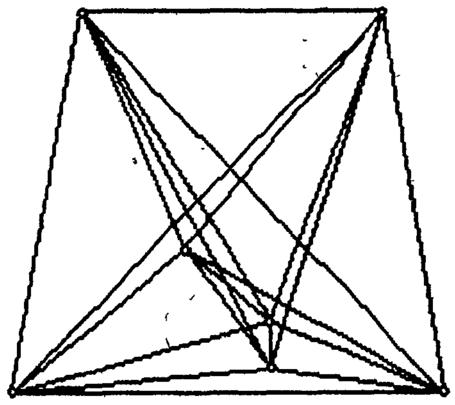

(44)

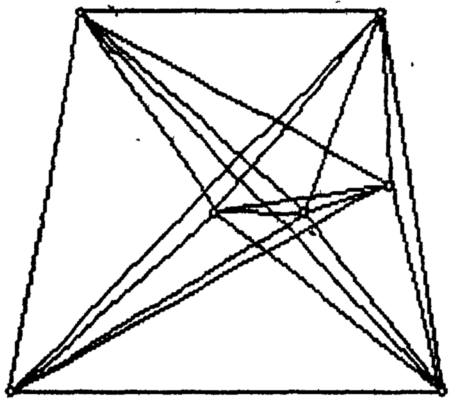

(45)

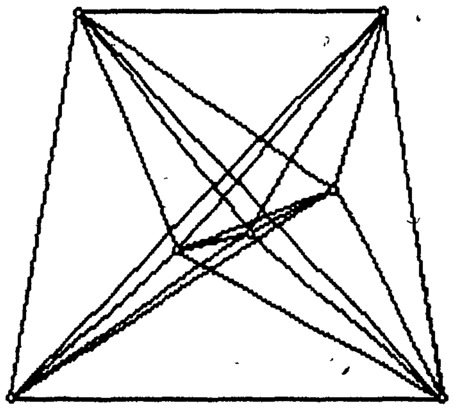

(46)

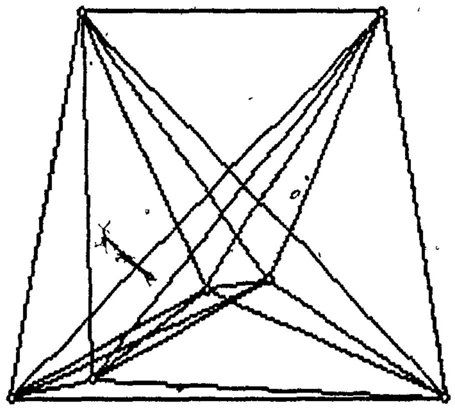

(47)

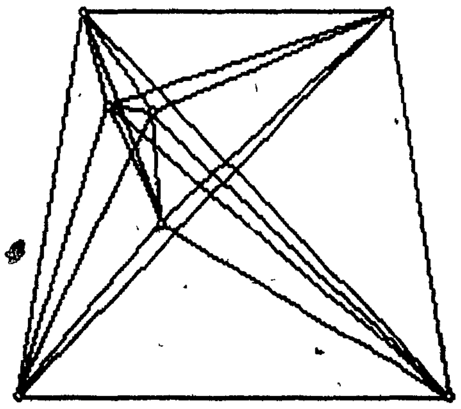

(48)

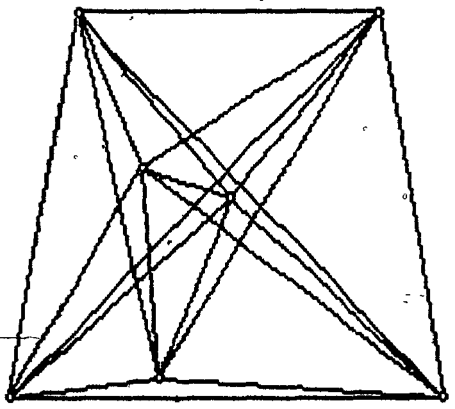

(49)

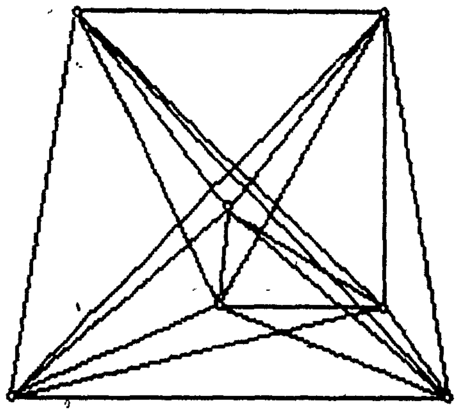

(50)

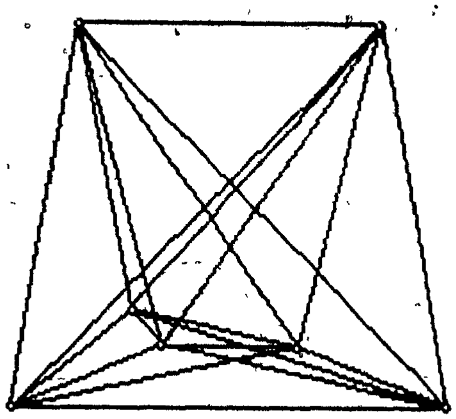

(51)

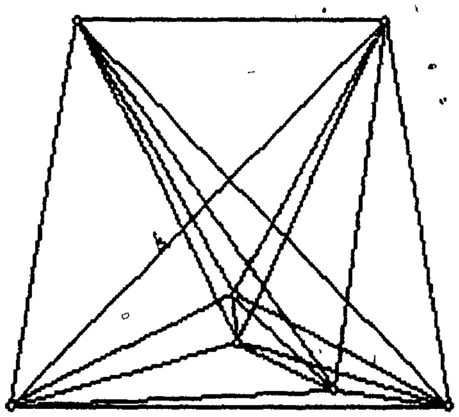

(52)

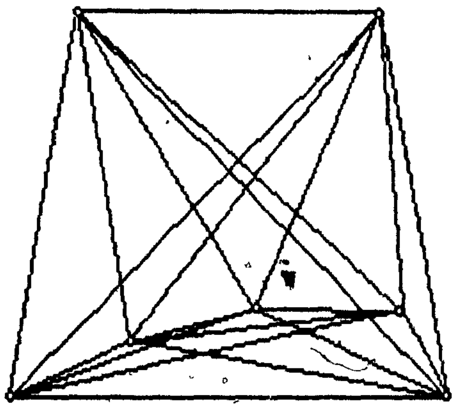

(53)

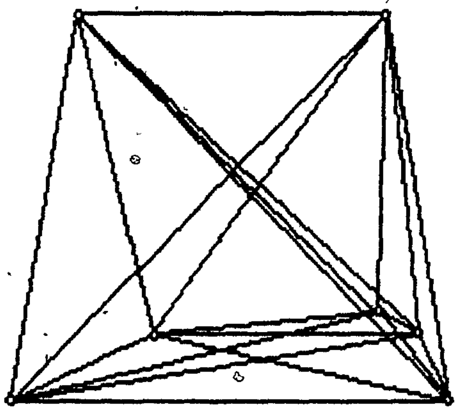

(54)

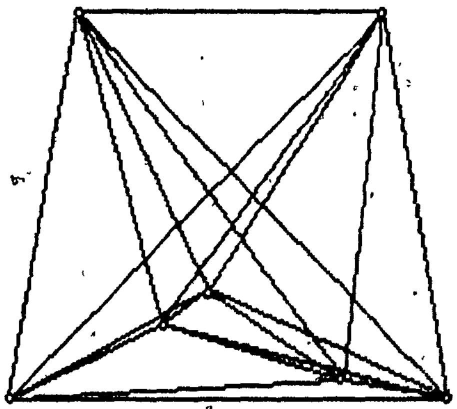

(55)

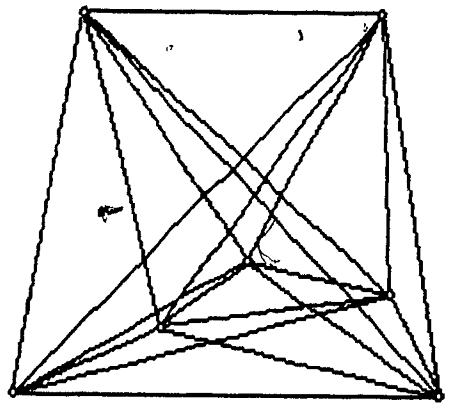

(56)

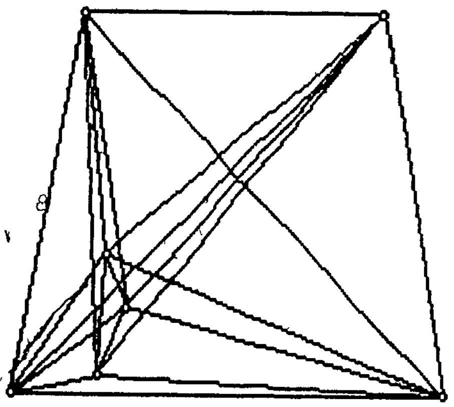

(57)

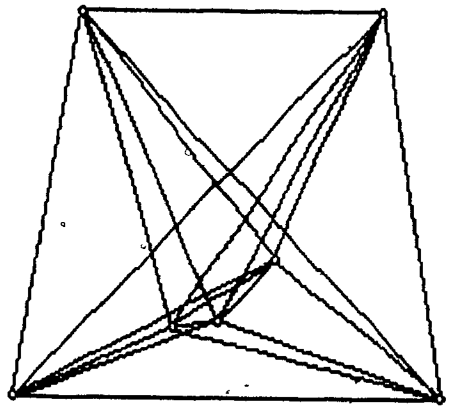

(58)

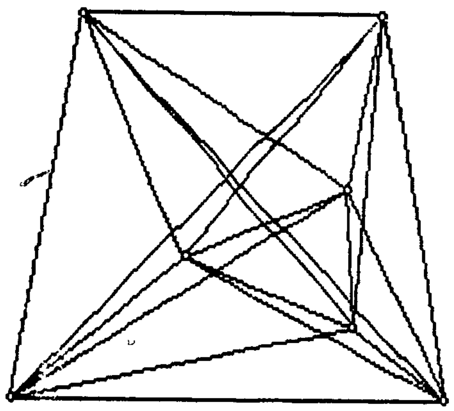

(59)

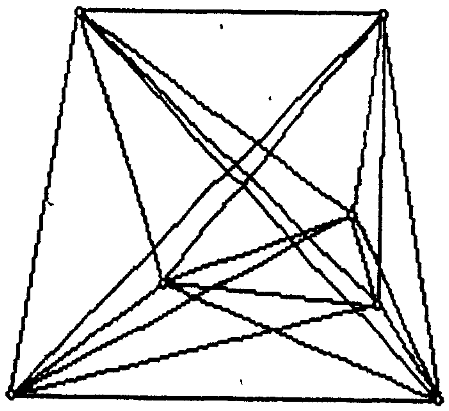

(60)

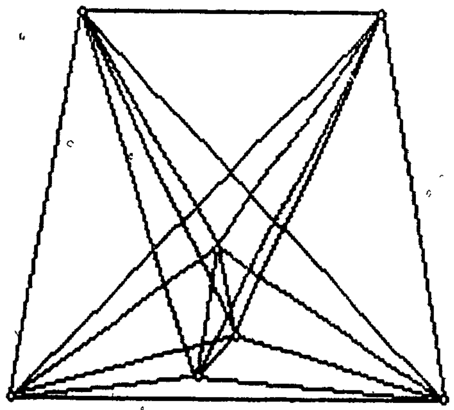

(61)

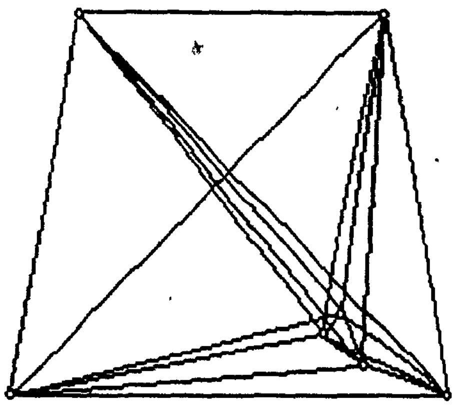

(62)

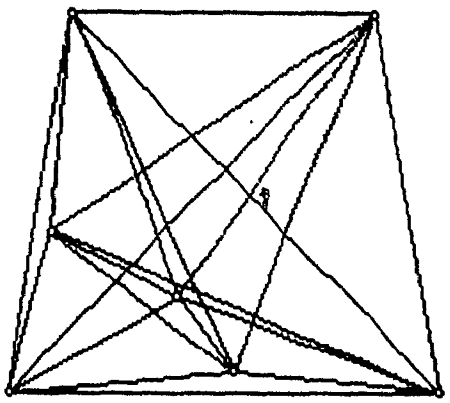

(63)

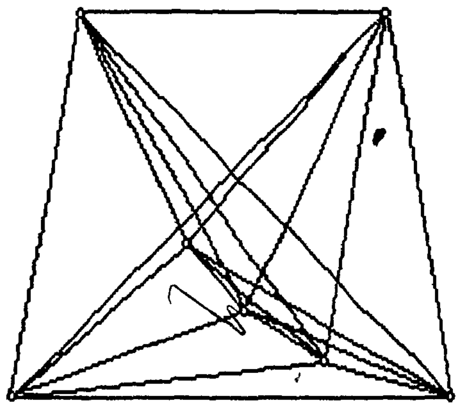

(64)

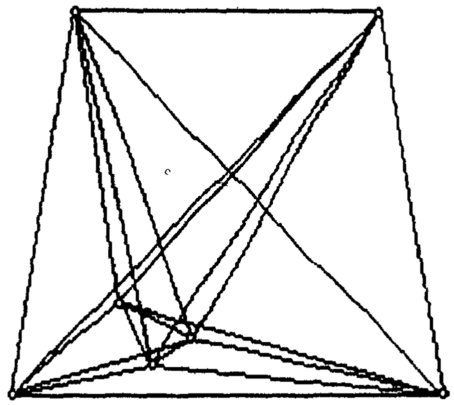

(65)

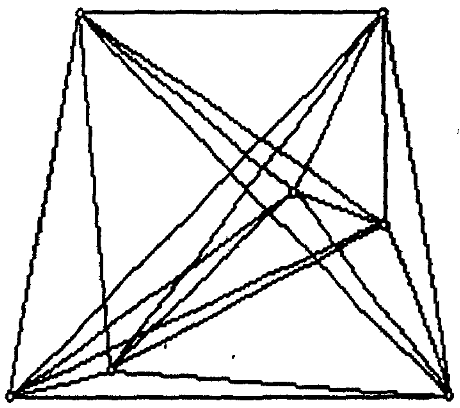

(66)

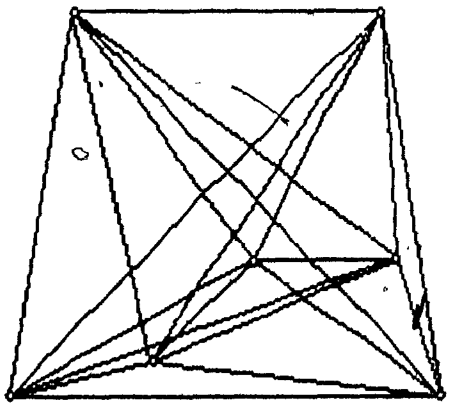

(67)

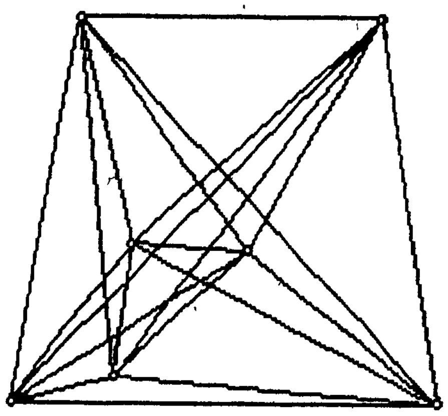

(68)

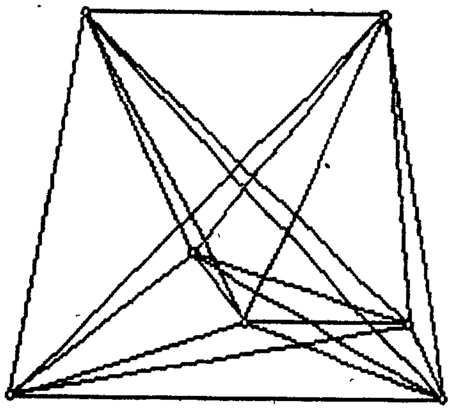

(69)

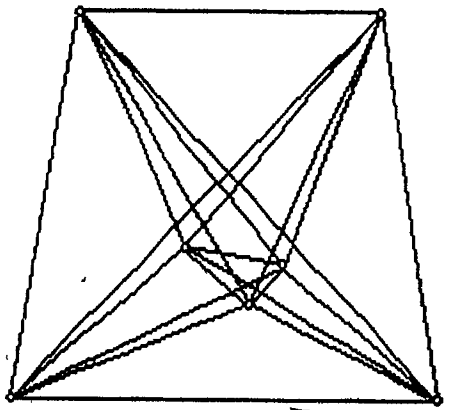

(70)

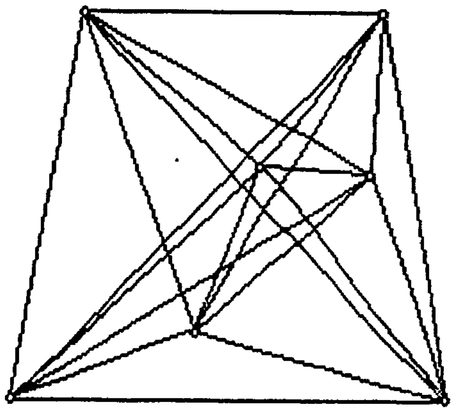

(71)

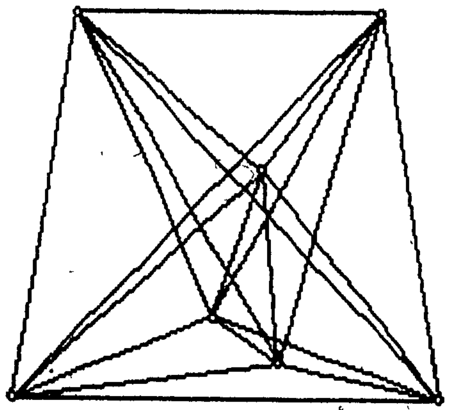

(72)

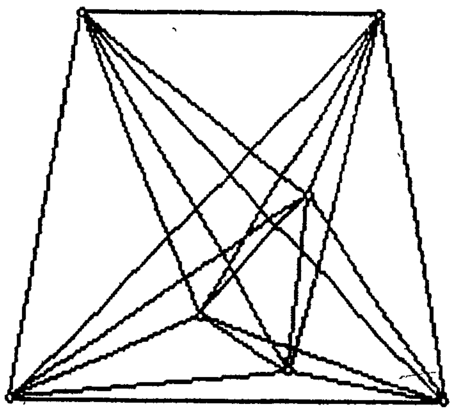

(73)

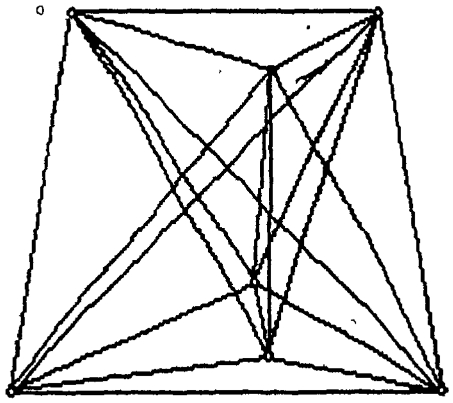

(74)

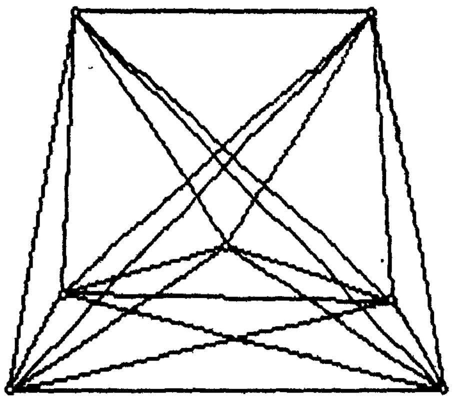

(75)

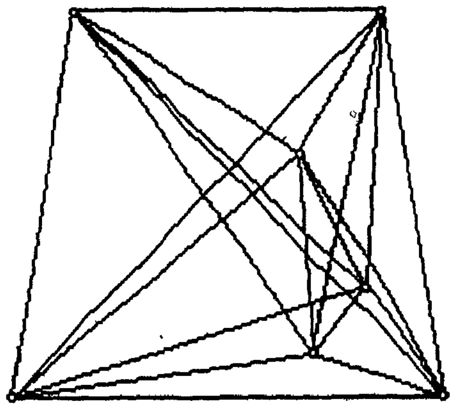

(76)

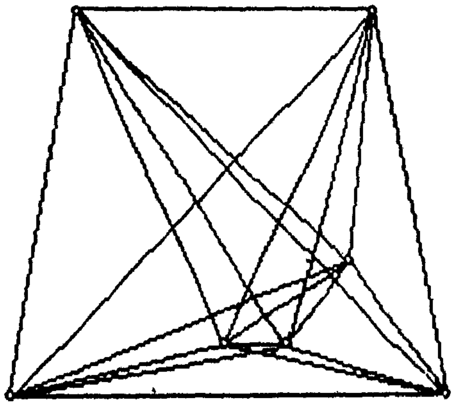

(77)

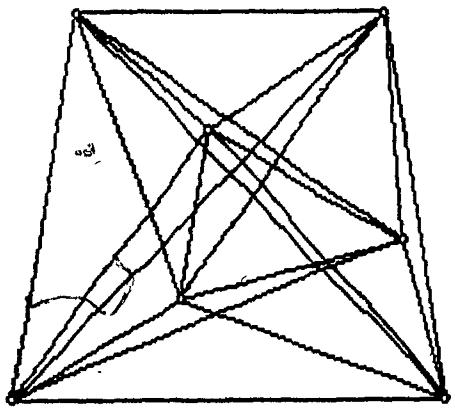

(78)

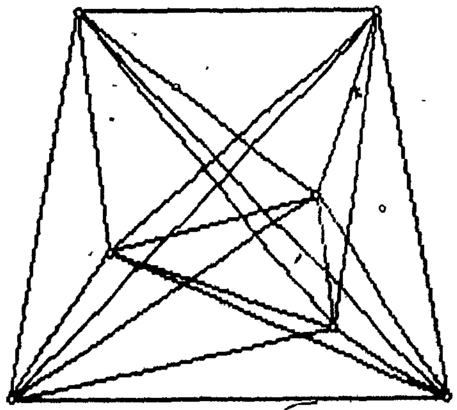

(79)

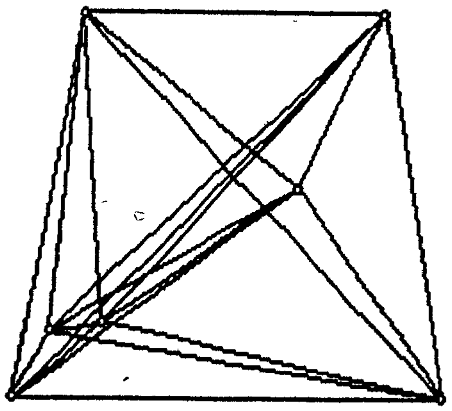

(80)

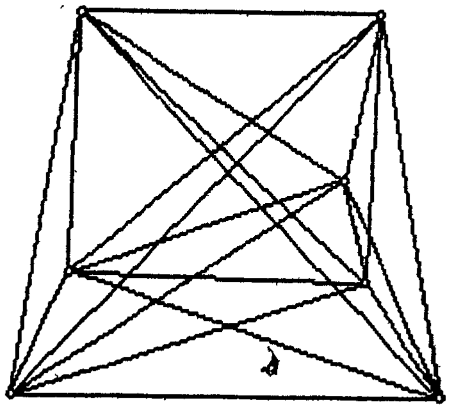

(81)

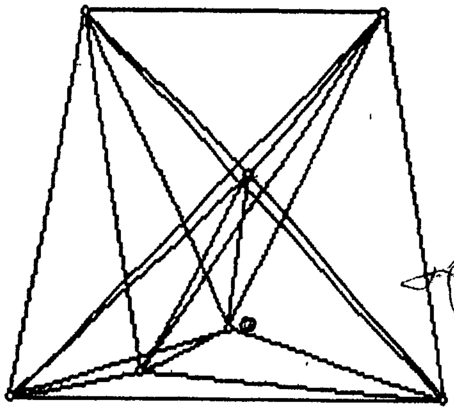

(82)

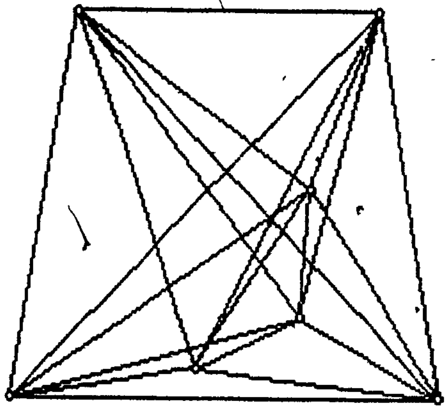

(83)

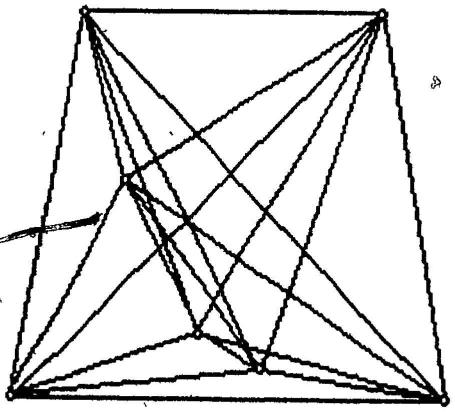

(84)

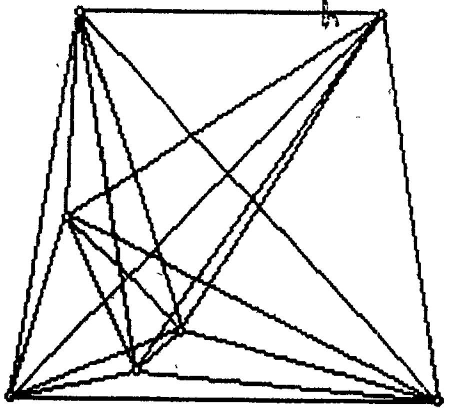

(85)

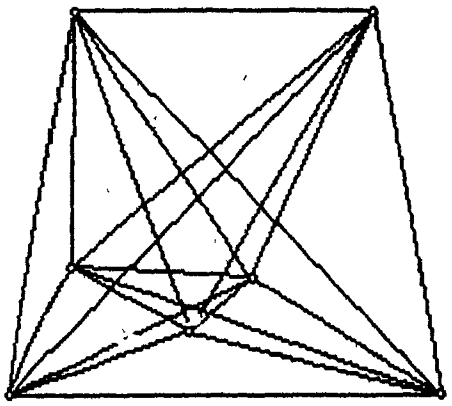

(86)

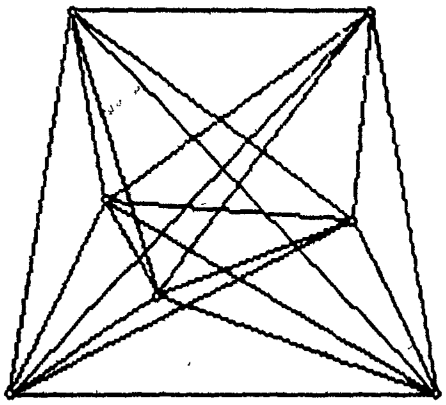

(87)

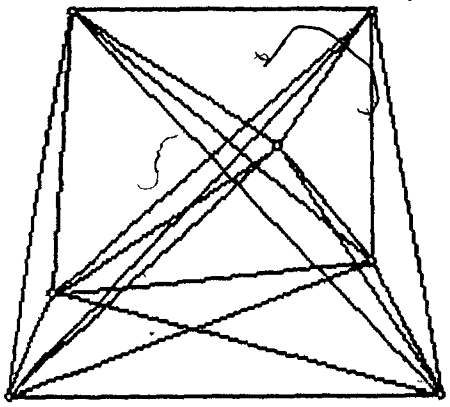

(88)

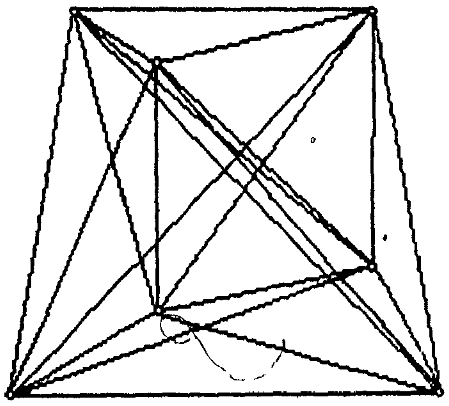

(89)

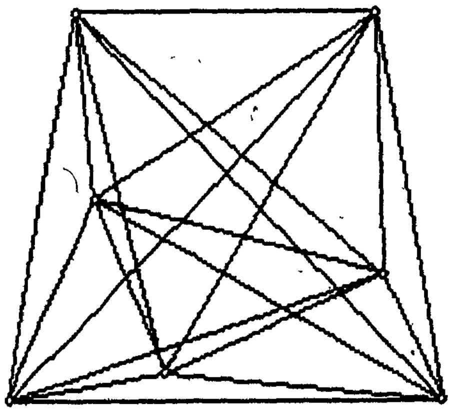

(90)

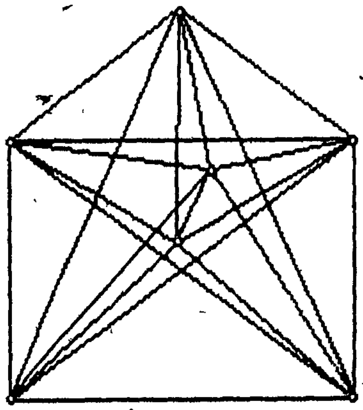

(91)

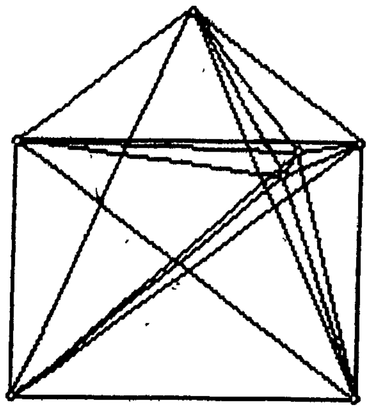

(92)

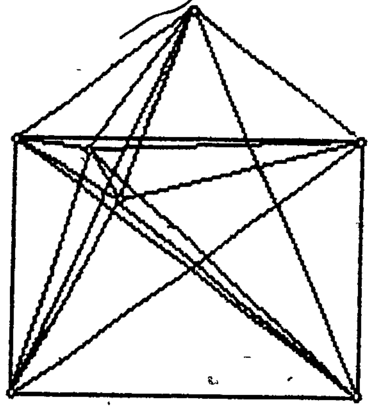

(93)

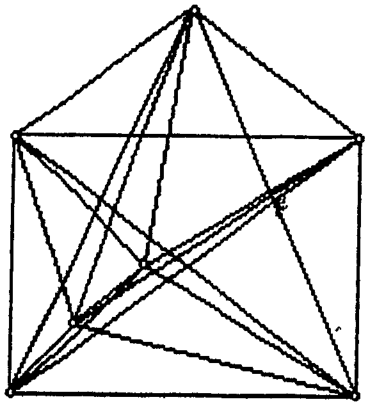

(94)

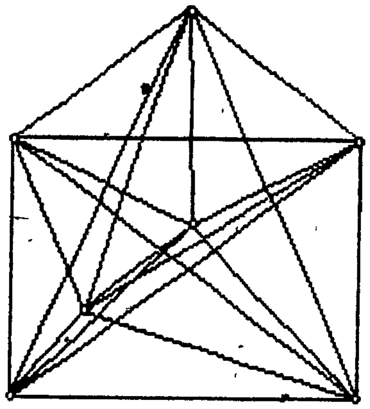

(95)

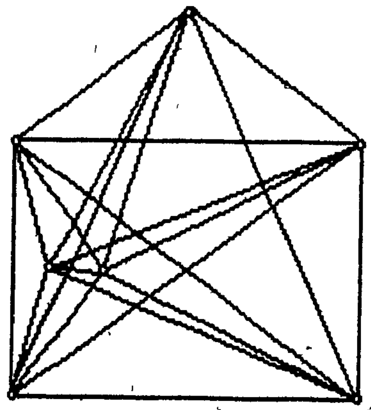

(96)

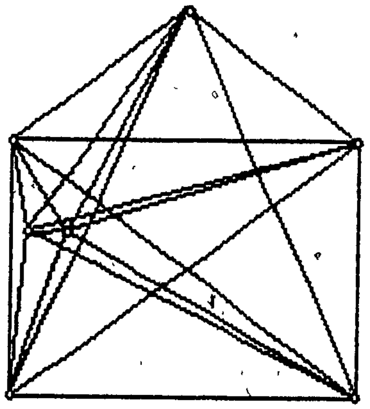

(97)

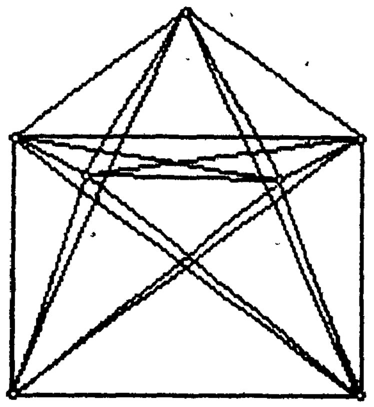

(98)

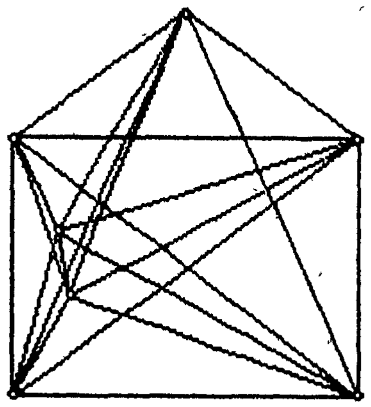

(99)

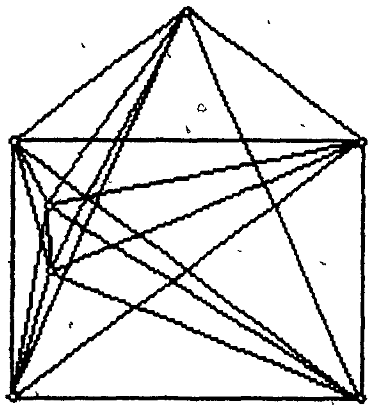

(100)

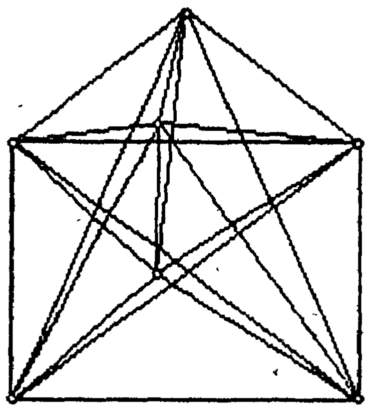

(101)

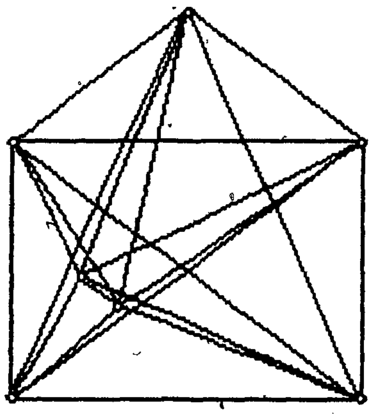

(102)

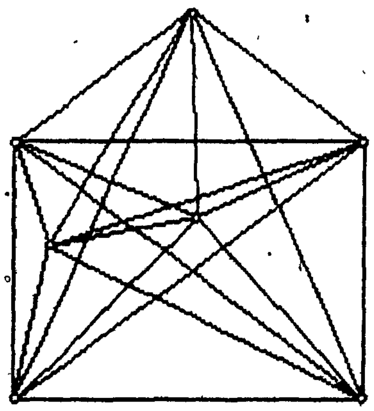

(103)

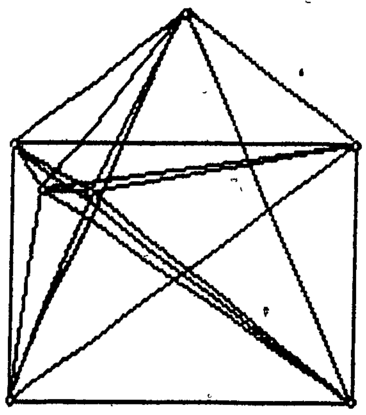

(104)

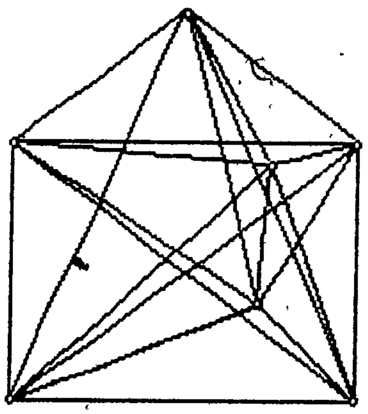

(105)

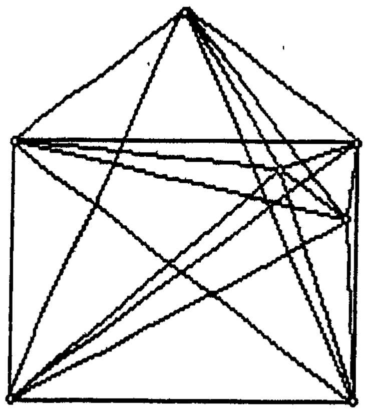

(106)

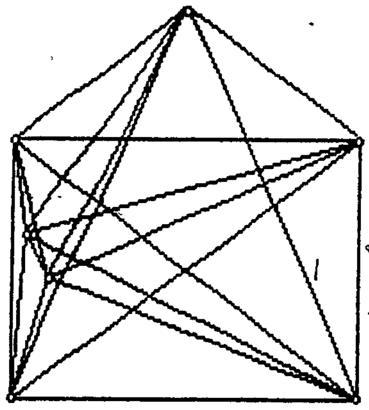

(107)

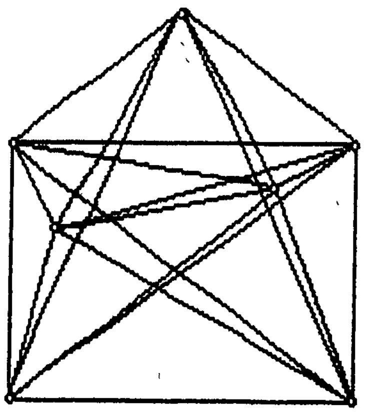

(108)

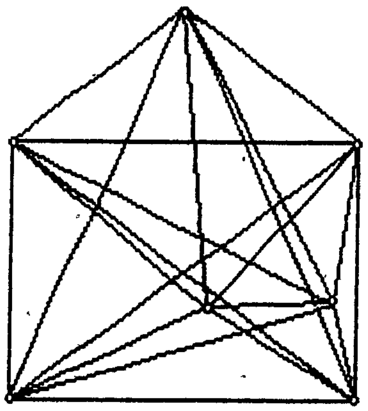

(109)

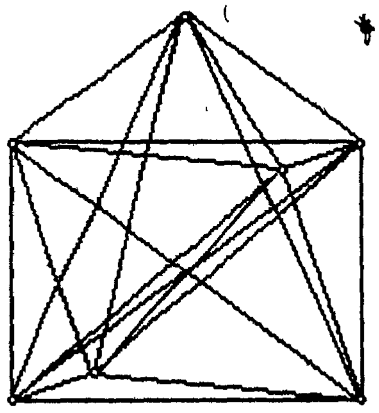

(110)

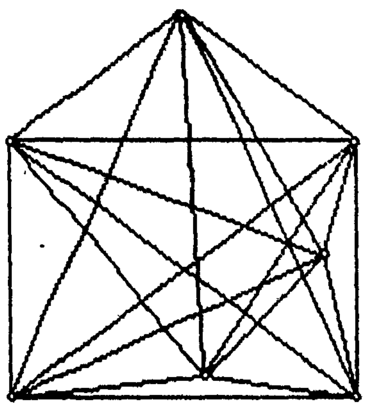

(111)

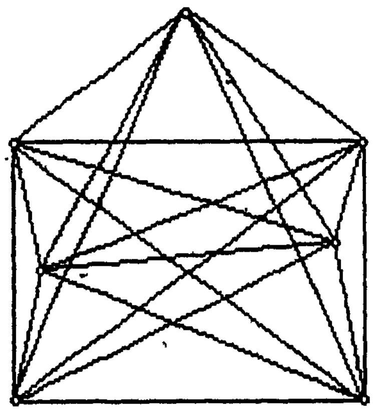

(112)

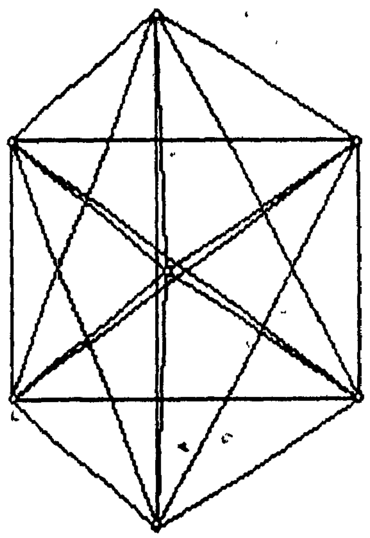

(113)

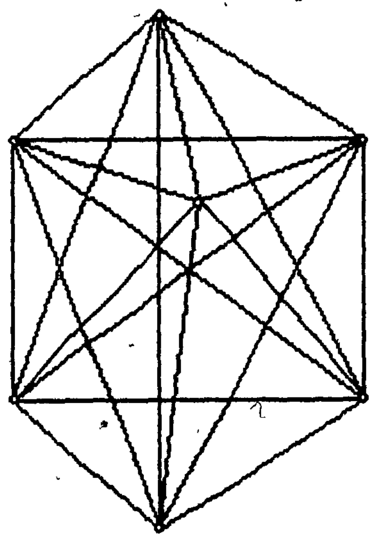

(114)

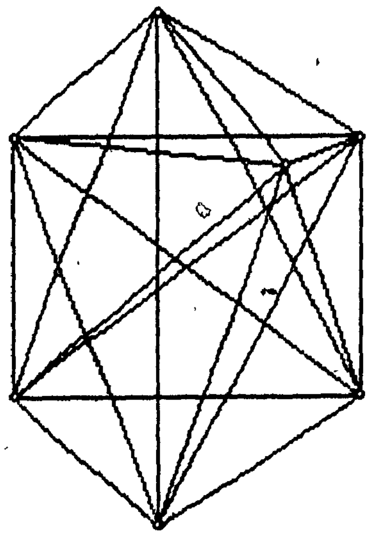

(115)

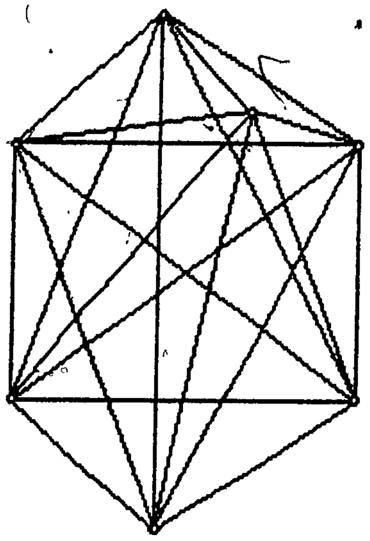

(116)

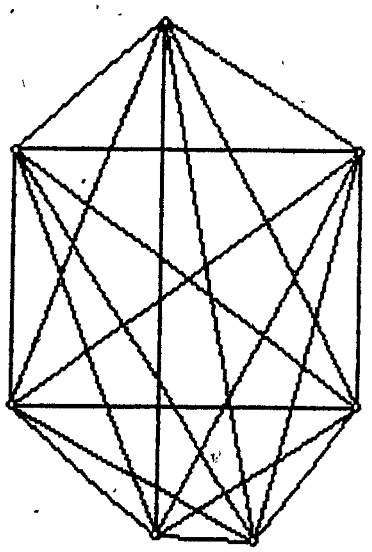

(117)


(118)


(119)


(120)


(121)


(122)

References

[1] K. Wagner, "Bemerkungen zum Vierfarbenproblem", *Jahresb der Deutsche Math. - Verein*, 14 (1936), 26–32

[2] I. Fáry, "On straight line representation of planar graphs", *Acta Univ. Szeged Sect. Sci. Math.*, 11 (1948) 229–233.

[3] R. K. Guy, "A combinatorial problem, Nabla", (*Bull. Malayan Math. Soc.*) 7 (1960), 68 - 72.

[4] F. Harary and A. Hill, "On the number of crossings in a complete graph", *Proc. Edinburgh Math. Soc.* (2), 13 (1962–3) 333–338.

[5] J. Blazek and M. Koman, "A minimal problem concerning complete plane graphs", in M. Fiedler (ed.), *Theory of Graphs and its applications* (Proc. Sympos. Smolenice, 1963) 113–117. MR 30 #4249.

[6] W. T. Tutte, "How to draw a graph", *J. London Math Soc.* (3) 13 (1963) 743–767; MR 28 #2610

[7] T. L. Saaty, "The minimum number of intersections in complete graphs, *Proc. Nat. Acad. Sci. U.S.A.*, 52 (1964) 688–690.

[8] R. K. Guy, *The crossing number of the complete graph*, Calgary Research Paper, No. 8, January, 1967.

[9] T. L. Saaty, "Two theorems on the minimum number of intersections of complete graphs", *J. Combinational Theory*, 2 (1967) 571–584.

[10] F. Harary, *Graph Theory*, Addsion-Wesley, Reading, Mass., 1969. Chapter 11.

[11] L. Woo, "An algorithm for straight-line representation of simple planar graphs", *J Franklin Inst.*, 287 (1969) 197–208.

[12] H. F. Jensen, "An upper bound for the rectilinear crossing number of the complete graph", *J. Combinatorial Theory* (B), 10 (1971), 212 - 216.

[13] R.K. Guy, "Latest results on crossing numbers", in *Recent Trends in Graph Theory*, Springer, N.Y., 1971, 143 - 156.

[14] R.K. Guy, "Crossing number of graphs", in *Graph Theory and Applications*, Springer Verlag, Berlin/New York, 1972, 113 - 124.

[15] P. Erdős and R. K. Guy, "Crossing number problems", *American Math Monthly*, 80 (1973), 52 - 58.

[16] R. B. Eggleton, *Crossing numbers of graphs*, Ph.D. Thesis, University of Calgary, 1973.

[17] A. Aho, J. Hopcroft, J. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass., 1974.

[18] R.K. Guy, Unpublished Correspondence between A. Uytterhœven of Baal, Belgium and J. Backelin of the University of Stockholm, 1977.

[19] Monroe Newborn and W. O. J. Moser, "Optimal crossing-free Hamiltonian circuit drawings of K_n", *J. Combin. Theory Ser.* B 29 (1980), 13 - 26.

[20] M. Ajtai, V. Chvátal, M. Newborn and E. Szemerédi, "Crossing-free subgraphs", *Annals of Discrete Mathematics*, Vol. 12, (1982). pp. 9-12.