A COMPUTER ALGORITHM FOR TESTING THE ISOMORPHIC PROPERTIES OF LINEAR GRAPHS

By

ROBERT GARY GOODMAN

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1963

Submitted to the faculty of the Graduate School of
the Oklahoma State University
in partial fulfillment of the requirements
for the degree of
MASTER OF SCIENCE
May, 1965

A COMPUTER ALGORITHM FOR TESTING

THE ISOMORPHIC PROPERTIES

OF LINEAR GRAPHS

Thesis Adviser: Richard L. Cummins

Paul A. McCollum

J. H. Boggs

Dean of the Graduate School

PREFACE

This thesis is a brief investigation of some aspects of isomorphism in linear graphs. The desire to initiate this study began while I was enrolled in a graduate course concerning linear graph theory and its applications to electrical networks. In particular, it was a question posed by the instructor that aroused this interest. The question was stated, "Is there a method or algorithm which will establish whether two graphs are isomorphic?" This thesis answers that question affirmatively and outlines a suitable method for establishing isomorphism.

The concept of isomorphism has long been clearly and concisely defined. There is, however, only a minimum of material on particular cases of isomorphism, and no monographs specifically on testing for isomorphism which could be used as a basis for this study. Irving M. Copi's INTRODUCTION TO LOGIC was used extensively for logical symbolism and theorems.

Indebtedness is acknowledged to Dr. R. L. Cummins for his incisive criticisms and helpful suggestions. Dr. Cummins first suggested the topic of testing for isomorphism. Special gratitude goes to the Computer Center of Oklahoma State University for providing the computer time necessary to complete the

program. The library staff of Oklahoma State University has been a great help by procuring materials for me.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

It was common in the 18th Century for a new area of mathematics to evolve from a need for techniques with which to solve the favorite puzzles of the time. It was in this way that Leonard Euler created graph theory in the year 1736.[1] Euler created (a better word might be formalized) graph theory as an aid to solving the famous Konigsberger bridge problem. An interesting description of the inception of graph theory is given in Linear Graphs and Electrical Networks by Seshu and Reed. For those not familiar with graph theory, the first three chapters of this book would be an excellent background for the material presented in this thesis.

After Euler's isolated contribution, investigation in the area of graph theory lay dormant for almost one hundred years. It was not until the middle of the 19th Century that a revival of interest in the study of graphs occurred. This revival was stimulated by an increasing application of graph theory to popular puzzle problems, the most celebrated of which is the Four Color Map Conjecture which DeMorgan posed around 1850. This problem, because of its continued interest to mathematicians, has been responsible for many contributions to graph theory.

In the early 1930's, Americans became interested in graph theory, as witnessed by the numerous papers written in this period. The most productive of these writers were Whitney and Tuttle. It was around this time that a German mathematician named Denes König wrote his pioneer book on graphs. Unfortunately, König's book has yet to be translated; in the last five or ten years, however, many books on the subject have been published in English.

Today, graph theory is a flourishing field. It is usually considered a branch of topology, although it overlaps large areas of set theory, combinational analysis, geometry, matrix theory, logic and many other fields. In graph theory, as in many other areas of mathematics, computers and new methods of programming have opened the door to more systematic and thorough investigations.

Definitions of Terms

In brief, a graph is a collection of points and lines which connect these points. Two names often used synonymously with a point of a graph are node and vertex. The lines which connect the vertices of a graph are called elements or edges. For identification purposes, the vertices and edges of a graph are labeled with numbers. Figure 1 shows a graph which has four vertices and five edges.

Each edge of a graph has two end-points. An edge may be attached to a vertex only at an end point.

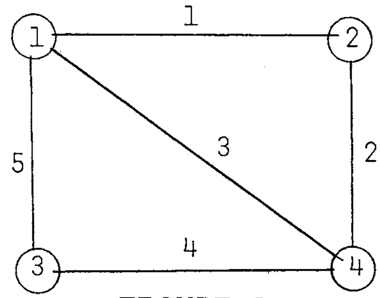


FIGURE 1

Graph with Four Vertices and Five Edges

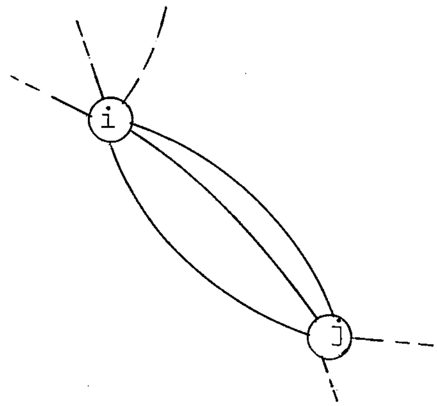
An edge of a graph is said to be incident at a vertex if an endpoint of the edge is attached to the vertex. Having only two endpoints, an edge can thus be incident at only two vertices of the graph. In Figure 1, edge 2 of the graph is incident at vertices 2 and 4.

The degree of a vertex is the number of the edges incident at that vertex. For the graph in Figure 1, the degree of vertex 3 is two while the degree of vertex 4 is three.

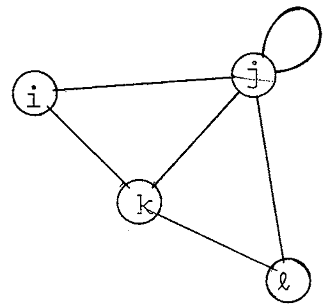
Two vertices are said to be connected when there exists an edge which is incident to both of them. Vertices 1 and 4 of Figure 1 are connected by edge number three. Vertices which are connected are spoken of as neighbors. For example: The neighbors of vertex 4 in Figure 1 are the vertices 1, 2, and 3.

Two or more edges which are incident at the same vertex pair are called parallel edges. An edge which is incident twice at the same vertex is called a self-loop. Examples of parallel edges and a self-loop are shown in Figure 2.

A homogeneous graph is a graph in which all the vertices are of the same degree.



Graph with Parallel Edges



Graph with a Self-Loop

FIGURE 2

Illustration of Parallel Edges and a Self-Loop

CHAPTER II

STATEMENT OF THE PROBLEM

A linear graph, or topological graph, will be isomorphic to another linear graph only if certain conditions exist. An algorithm for establishing isomorphism must contain a sufficient set of these conditions. It seems quite apropos that a concise and concrete definition of isomorphism be given at the beginning.

Definition of Isomorphism

Two graphs, G1 and G2, are isomorphic (or congruent)[2] if there is a one-to-one correspondence between the vertices of G1 and G2, and a one-to-one correspondence between the edges of G1 and G2 which preserves the incidence relationship.[1]

To prove that two graphs are isomorphic, we must define a unique one-to-one correspondence between vertices and edges and show that incidence relationships are preserved under this correspondence. If the one-to-one correspondence is found by using the fact that all incidence relationships must hold, that correspondence would be sufficient to show isomorphism. If no correspondence exists such that incidence relations are preserved the two graphs are not isomorphic. This Thesis gives a method of finding this correspondence if one exists.

It was felt that in the first investigation of isomorphism, only " reduced" graphs, that is graphs which have no parallel edges and no self-loops, should be considered.

Preliminary Theorem

Under the restriction that graphs have no parallel edges, Seshu and Reed's definition of isomorphism may be shortened to:

> Two graphs, G1 and G2, neither of which have parallel edges, are isomorphic if there is a one-to-one correspondence between the vertices of G1 and G2 which preserves the incidence relationship.

Why is this possible? Assume a one-to-one correspondence between vertices of G1 and G2 exists which preserves the incidence relationship, and that neither G1 nor G2 has parallel edges. Select any edge in G1, call it edge k. Since k is a single edge, it is the only edge between its end vertices. Let k's end vertices be i and j. Because of the one-to-one correspondence between vertices, there are two vertices of G2, i' and j', which correspond to i and j. All incidence relations hold and thus there must be an edge between i' and j'. Since there are no parallel edges in G2, there is one and only one edge connecting i' and j'. Call this edge k'. There is only one possible correspondence for edge k of G1 and that is edge k' of G2. By associating each edge of G1 with its two incident vertices, locating the corresponding vertices of G2 and thus the edge connecting them, a one-to-one correspondence can be obtained between the edges of G1 and the edges of G2. Thus, there is a one-to-one correspondence between vertices of G1 and G2, and a one-to-one correspondence between the edges of G1 and G2 which preserves the incidence relationship. Then, according to the original definition, graphs G1 and G2 are isomorphic. This conclusion is supported by the fact that the vertex incidence matrix completely defines a graph when the graph has single edges.[3] Since vertex-incidence matrices fully describe graphs with no parallel edges, they will be used henceforth as the only definition for graphs.

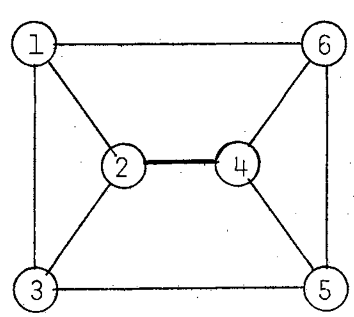
Vertex-Incidence Matrix as the Definition of a Graph

A vertex-incidence matrix is a square matrix of order Nv, where Nv is the number of vertices in the graph. Each element of the matrix is defined by the following rules:

$A_{i,j} = 1$; if vertex i is connected to vertex j.

$A_{i,j} = 0$; if vertex i is not connected to vertex j.

An example of a graph and its associated vertex-incidence matrix are shown below.



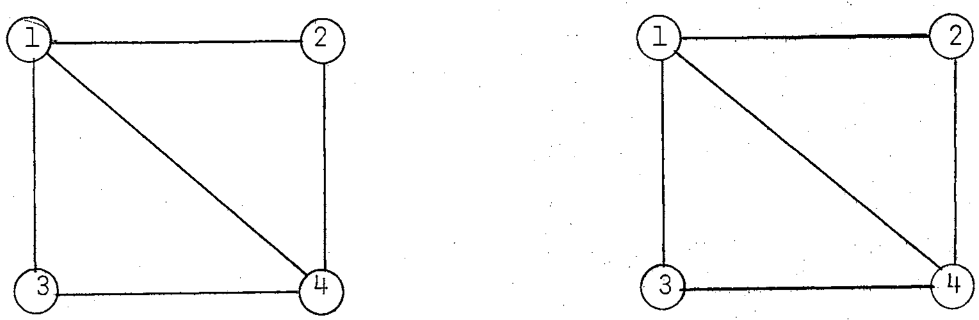
011001
101100
110010
010011
001101
100110

FIGURE 3

Example of a Graph and its Vertex-Incidence Matrix

With graphs which contain only single edges, the problem degenerates to proving the existence or non-existence of a one-to-one correspondence between vertices which preserves incidence relationships. Some examples will illustrate the concept of isomorphism and clarify later arguments. To simplify expressions, the symbol, θ, will be used in place of the phrase "corresponds to."

It should be obvious that two identical graphs are isomorphic. Figure 4 shows two identical graphs and one possible correspondence between their vertices.



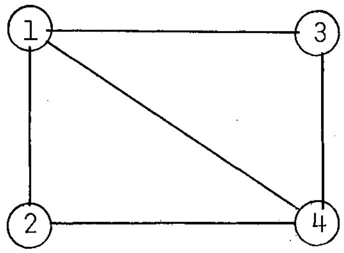
G1 G2

Vertex of G1		Vertex of G2
1	θ	1
2	θ	2
3	θ	3
4	θ	4

FIGURE 4

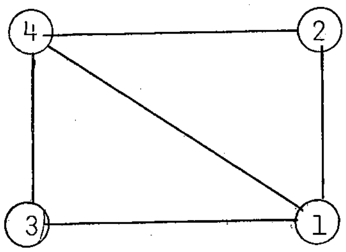
Two Isomorphic Graphs and Their Vertex Correspondence

This is, of course, the simplest correspondence. There are three other possible correspondences, two of which are shown below:



G2

1	θ	1
2	θ	3
3	θ	2
4	θ	4



G2

1	θ	4
2	θ	2
3	θ	3
4	θ	1

FIGURE 5

Other Vertex Correspondences of G1 and G2

As has been shown, two graphs which are isomorphic may have many possible correspondences, although there need be only one.

The isomorphic graphs above are special cases because their edges have the same dimensions. Isomorphism deals only with the topological properties of graphs. Thus, edges have no special length or direction and can be imagined as being made of elastic, which may be lengthened or shortened and moved as we please. Likewise, vertices may be moved as desired. This change in edges and vertices does not disturb the connectivity of the graph. Some examples of this property on isomorphic graphs are shown in Figure 6.

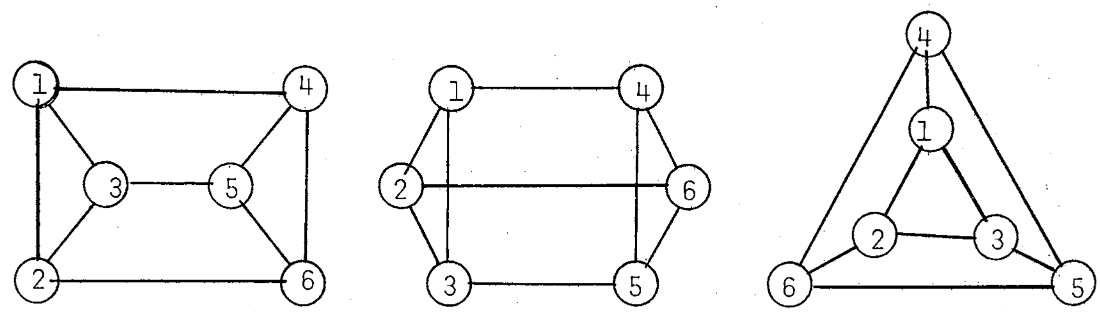


FIGURE 6

Example of Three Isomorphic Graphs

In the graphs above, the vertex correspondence between any two graphs is:

$$
\begin{array}{ccc}
1 & \theta & 1 \\
2 & \theta & 2 \\
3 & \theta & 3 \\
4 & \theta & 4 \\
5 & \theta & 5 \\
6 & \theta & 6
\end{array}
$$

This correspondence can more easily be represented in the matrix form;

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
$$

Row i of this correspondence matrix is associated with vertex i in G1. Column j of the matrix is associated with vertex j of G2. If vertex i of G1 corresponds to vertex j of G2, a 1 is placed in the i, j position. An example is given in Figure 7.

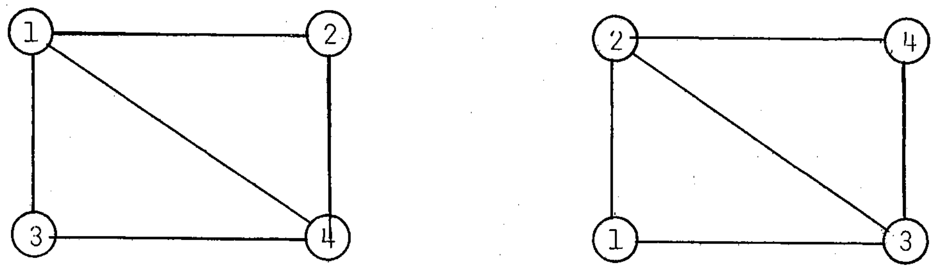
$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$	represents the correspondence	$\begin{matrix} 1 & \theta & 3 \\ 2 & \theta & 1 \\ 3 & \theta & 2 \\ 4 & \theta & 4 \end{matrix}$

FIGURE 7

Illustration of a Vertex-Correspondence Matrix

The two graphs in Figure 8 are isomorphic. They are shown with their respective vertex-incidence matrices, and a correspondence matrix which represents a possible isomorphic correspondence between the vertices of the two graphs.

If in writing the vertex-incidence matrix for G2, column 1 and row 1 were associated with vertex 2, column 2 and row 2 with vertex 4, column 3 and row 3 with vertex 1, and column 4 and row 4 with vertex 3, as the correspondence matrix suggests, the result would be a vertex-incidence matrix which is identical to the vertex-incidence matrix for G1. In matrix theory this changing of row and columns is called permutation. Permutation can also be accomplished by multiplication of the matrix of the same form as the correspondence matrix. Pre-multiplication of the vertex incidence matrix of G1 by CM permutes the rows of VIG1 in the manner desired. Since the order of indices is reversed when post-multiplying, the transpose of CM is used to obtain the desired permutation of columns.



G1 G2

Two Isomorphic Graphs

$$VIG1 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \qquad VIG2 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Vertex--Incidence Matrices

$$CM = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Correspondence Matrix

FIGURE 8

Some Matrices of Isomorphic Graphs

Thus

$$[VIG1] = [CM] \; [VIG2] \; [CM]^T$$

The matrix CM is the unknown in this equation. It would be possible, using a computer, to check this equation with possible CM matrices until an isomorphism is found. In order to find every isomorphism, every possible CM matrix would have to be tested. For a graph with 6 vertices, this would require 6! or 720 tests. By using set theory and the connection properties of graphs, a significant reduction of the number of necessary tests can be made.

CHAPTER III

PROOF OF THE ALGORITHM

We will begin by using set theory and propositional calculus to derive logical expressions which will be reduced by computer techniques. In the following discussion, graph 1 will be referred to as G1, and likewise, graph 2 will be referred to as G2.

The symbol $P_{1,1}$ will be equivalent to the statement: Vertex 1 of G1 corresponds to Vertex 1 of G2. In general, $P_{i,j}$ will be equivalent to the statement: Vertex i of G1 corresponds to Vertex j of G2. Note that the order of the subscripts on P is important. The statement: Vertex i of G1 does not correspond to vertex of G2, will by symbolized $\overline{P_{i,j}}$.

Similarly, the symbols C, R, and I are defined to represent the following statements:

C: There exists a one-to-one correspondence between vertices of G1 and vertices of G2.

R: All incidence relationships hold.

I: G1 and G2 are isomorphic.

Recall the definition of isomorphism: Two graphs G1 and G2 are isomorphic if there is a one-to-one correspondence between the vertices of G1 and G2 and a one-to-one correspondence between the edges of G1 and G2 which preserves the incidence relationships.[5] We have shown that for connected graphs with no parallel edges this definition can be rewritten as: Two graphs G1 and G2 are isomorphic if there is a one-to-one correspondence between the vertices of G1 and G2 which preserves the incidence relationships.

Implication of the Definition

Symbolically, this definition could be stated: I being true is equivalent to C being true and R being true. Written in logical shorthand:

$$I \leftrightarrow C \cdot R$$

The logical symbols from Copi[4] will be adopted in writing all logical expressions.

Logical Relations

Suppose we have two graphs, G1 and G2, both with N_V vertices, which we wish to test for isomorphism. Let the vertices of G1 be labeled $1,2,\ldots N_V$ and the vertices of G2 be labeled $1,2,\ldots N_V$. If there exists a one-to-one correspondence; i.e. C is true, it is necessary that vertex i of G1 correspond to one of the vertices of G2. Symbolically: C implies that $P_{i,1}$ is true or $P_{i,2}$ is true or.. or P_{i,N_V} is true.

Therefore, we write

$$C \rightarrow [P_{i,1} \vee P_{i,2} \vee P_{i,3} \cdots \vee P_{i,N_v}]$$

If this is done for each vertex of G1, we would have:

$$C \rightarrow [(P_{1,1} \vee P_{1,2} \vee \ldots P_{1,N_v})\ (P_{2,1} \vee P_{2,2} \cdots \vee P_{2,N_v}) \cdots$$
$$(P_{i,1} \vee P_{i,2} \vee P_{i,3} \cdots \vee P_{i,N_v}) \cdots (P_{N_v,1} \vee P_{N_v,2} \cdots$$
$$\vee P_{N_v,N_v})]$$

By defining $\prod_{i=1}^{N_v} f(P_{i,k})$ to be the logical product ("and" operation) of expressions involving $P_{i,k}$ where i ranges from 1 to N_v, we may more conveniently write the above implication as:

$$C \rightarrow [\prod_{i=1}^{N_v} (P_{i,1} \vee P_{i,2} \vee P_{i,3} \cdots P_{i,N_v})]$$

This represents all possible associations. As an example: if $N_v = 2$

$$C \rightarrow [(P_{1,1} \vee P_{1,2})\ (P_{2,1} \vee P_{2,2})]$$

Using the principle of distribution[4], this implication may be expressed

$$C \rightarrow [P_{1,1}\ P_{2,1} \vee P_{1,1}\ P_{2,2} \vee P_{1,2}\ P_{2,1} \vee P_{1,2}\ P_{2,2}]$$

Since this expression contains all possible associations, the terms $P_{1,1}P_{2,1}$ and $P_{1,2}P_{2,2}$ are included. These terms state that some vertex of one graph corresponds to two vertices of the other graph. This is a correspondence, but it is not a one-to-one correspondence. Further restrictions must be placed on the correspondence to make it one-to-one.

If vertex i of G1 corresponds to vertex j of G2, we can write, because of the necessary one-to-one correspondence:

$$C \rightarrow \{P_{i,j} \rightarrow [\ (\overline{P}_{i,1}\overline{P}_{i,2}\cdots\overline{P}_{i,j-1}\overline{P}_{i,j+1}\cdots\overline{P}_{i,N_v})$$
$$(\overline{P}_{i,j}\overline{P}_{2,j}\cdots\overline{P}_{i-1,j}\overline{P}_{i+1,j}\cdots\overline{P}_{i,N_v}\ )\ ]\ \}$$

By defining $\prod_{k=1}^{N_v} P_{i,k}$ to be the logical product of $P_{i,k}$ terms where k ranges from 1 to N_V, we may more conveniently write the above implications as:

$$C \rightarrow \{\ P_{i,j}\ \rightarrow [(\ \prod_{\substack{k=1\\k\neq j}}^{N_v} \overline{P}_{i,k})\ (\ \prod_{\substack{k=1\\k\neq j}}^{N_v} \overline{P}_{k,j}\ )\ \ ]\ \ \}$$

Note that for obvious reasons, the term $P_{i,j}$ is omitted from the logical product since $C \rightarrow [P_{i,j} \rightarrow \overline{P}_{i,j}]$ would not be a valid statement.

By absorption[4],

$$C\ \rightarrow [P_{i,j} \rightarrow (P_{i,j})\ (\ \prod_{\substack{k=1\\k\neq j}}^{N_v} \overline{P}_{i,k})\ (\ \prod_{\substack{k=1\\k\neq i}}^{N_v} \overline{P}_{k,j}\ )\ \ ]$$

If C is true and vertex i of G1 corresponds to vertex j, and only to j, of G2, it should be apparent that the statement

$$C\ \rightarrow [\ (P_{i,j})\ \ (\ \prod_{\substack{k=1\\k\neq j}}^{N_v} \overline{P}_{i,k}\ )\ (\ \prod_{\substack{k=1\\k\neq i}}^{N_v} \overline{P}_{k,j}) \rightarrow P_{i,j}\ \ ]$$

is true.

By Material Equivalence[4]

$$C \rightarrow [\ P_{i,j} \leftrightarrow (P_{i,j})\ \ (\ \prod_{\substack{k=1\\k\neq j}}^{N_v} \overline{P}_{i,k})\ \ (\ \prod_{\substack{k=1\\k\neq i}}^{N_v} \overline{P}_{k,j}\ )\ \ ]$$

This simply means that under a 1-1 correspondence, vertex i of G1 corresponds to one and only one vertex of G2 and vertex j of G2 corresponds to one and only one vertex of G1.

This statement is very important to the analysis and will be referred to several times. The statement eliminates self-contradictory terms of the form $P_{1,1} \ldots P_{i,j} \ldots P_{i,k}$. By replacement of $P_{i,j}$ with its equivalent expression, we would have

$$P_{1,1} \ldots P_{i,j} \ldots P_{i,k} \leftrightarrow P_{1,1} \ldots P_{i,j} \overline{P}_{i,1} \overline{P}_{i,2} \ldots \overline{P}_{i,k} \ldots P_{i,k}.$$

The logical product $P_{i,k} \, \overline{P}_{i,k}$ would be interpreted: vertex i of G1 corresponds to vertex k of G2 and vertex i of G1 does not correspond to vertex k of G2. This statement is obviously false, and therefore the expression

$$P_{1,1} \ \ldots \ P_{i,j} \ P_{i,1} P_{i,2} \ \ldots \ P_{i,k} \ \ldots \ P_{i,k}$$

is false. The equivalence will be used often in reducing complex statements as shown in the following example.

Assume $N_v = 2$, as before, then

$$C \rightarrow [P_{1,1} P_{2,1} \vee P_{1,1} P_{2,2} \vee P_{1,2} P_{2,1} \vee P_{1,2} \, P_{2,2}]$$

$$C \rightarrow [P_{1,1} \leftrightarrow P_{1,1} \, \overline{P}_{1,2} \, \overline{P}_{2,1}]$$

$$C \rightarrow [P_{1,2} \leftrightarrow P_{1,2} \, \overline{P}_{1,1} \, \overline{P}_{2,2}]$$

$$C \rightarrow [P_{2,1} \leftrightarrow P_{2,1} \, \overline{P}_{1,1} \, \overline{P}_{2,2}]$$

$$C \rightarrow [P_{2,2} \leftrightarrow P_{2,2} \, \overline{P}_{1,2} \, \overline{P}_{2,1}]$$

By replacing each $P_{i,j}$ in the first implication with its equivalent given in the succeeding statement we obtain

$$C \rightarrow [P_{1,1} \overline{P}_{1,2} \overline{P}_{2,1} P_{2,1} \overline{P}_{1,1} \overline{P}_{2,2} \vee P_{1,1} \overline{P}_{1,2} \overline{P}_{2,1} P_{2,2} \overline{P}_{1,2} \overline{P}_{2,1}$$

$$\vee P_{1,2} \overline{P}_{1,1} \overline{P}_{2,2} P_{2,1} \overline{P}_{1,1} \overline{P}_{2,2} \vee P_{1,2} \overline{P}_{1,1} \overline{P}_{2,2} P_{2,2} \overline{P}_{1,2} \overline{P}_{2,1}]$$

By elimination of self-contradictory terms, the above statement reduces to

$$C \rightarrow [P_{1,1}\overline{P}_{1,2}\overline{P}_{2,1}P_{2,2}\overline{P}_{1,2}\overline{P}_{2,1} \vee P_{1,2}\overline{P}_{1,1}\overline{P}_{2,2}P_{2,1}\overline{P}_{1,1}\overline{P}_{2,2}]$$

For graphs with more than 3 or 4 vertices this expression showing the existence of a correspondence becomes quite cumbersome. For this reason, it will be written

$$C \rightarrow [\prod_{i=1}^{N_v} (P_{i,1} \vee P_{i,2} \vee P_{i,3} \ldots P_{i,N_v})]$$

By defining the right side of the above implication as Q, it may be written

$$C \rightarrow Q$$

And from previous statements

$$C \rightarrow \{P_{i,j} \leftrightarrow [P_{i,j} (\prod_{\substack{k=1\\k\neq j}}^{N_v} \overline{P}_{i,k}) (\prod_{\substack{k=1\\k\neq i}}^{N_v} \overline{P}_{k,j})]\}$$

by defining $S_{i,j}$ as $\{P_{i,j} (\prod_{\substack{k=1\\k\neq j}}^{N_v} P_{i,k}) (\prod_{\substack{k=1\\k\neq j}}^{N_v} P_{k,j})\}$

the above may be written

$$C \rightarrow [P_{i,j} \leftrightarrow S_{i,j}]$$

Now, since

Q implies the existence of a correspondence

And

$P_{i,j} \leftrightarrow S_{i,j}$ for all i and j, states that any correspondence is one-to-one.

Therefore

$$[(P_{i,j} \leftrightarrow S_{i,j}) \cdot Q] \rightarrow C$$

By material equivalence[4],

$$C \leftrightarrow [(P_{i,j} \leftrightarrow S_{i,j}) \cdot Q] \quad .$$

We have established an equivalent expression which describes all the possible one-to-one correspondences.

We now pursue a logical expression equivalent to R. We observe that no vertex of G1 which is of degree d can correspond to a vertex of G2 which is of degree other than d. Thus we can separate the vertices into classes by their degrees. It follows that: Any vertex of G1 which is of degree d must correspond to some vertex of G2 which is contained in the set of vertices of G2 which are of degree d and must not correspond to any vertex of G2 which is of any degree other than d; Expressed Symbolically:

$$R \rightarrow (P_{h,i} \vee P_{h,j} \vee P_{h,k} .. P_{h,e})\ (\overline{P}_{h,w}\overline{P}_{h,x}\overline{P}_{h,y} ... \overline{P}_{h,z})$$

h is a vertex of G1 of degree d. Each i,j,k,...e is a vertex of G2 of degree d. Each w,x,y...z is a vertex of G2, not of degree d.

If this is done for each vertex of degree d, it could be written in the product form:

$$R \rightarrow \{[\prod_{h=n/n,}^{N_V} (P_{h,i} \vee P_{h,j} \vee P_{h,k} ... \vee P_{h,e})]\ [\prod_{h=n}^{N_V} (\overline{P}_{h,w}\overline{P}_{h,x}\overline{P}_{h,y} .. \overline{P}_{h,z})]$$

is a vertex of degree d).

d ranges through all possible values

If $S_{i,j}$ and R are true, it follows that each neighboring vertex of i of G1 must correspond to one of the neighbors of vertex j of G2, and not to any other vertex of G2. That is, the set of vertices which are the neighbors of i of G1 is equal to the set of vertices which are the neighbors of j of G2 and not equal to the set of vertices which are not neighbors of j of G2.

This implication assures a one-to-one correspondence within the degree classes of G1 and G2.

Written Symbolically:

$$R \cdot S_{i,j} \rightarrow \{ [(P_{a,m} v\ P_{a,n} v\ P_{a,q} \ldots)(P_{b,m} v\ P_{b,n} v\ P_{b,q} \ldots) \ldots.] \\ [(\overline{P}_{a,u}\overline{P}_{a,w}\overline{P}_{a,x} \ldots)(\overline{P}_{b,u}\overline{P}_{a,w}\overline{P}_{a,x} \ldots) \ . \ . \ .] \}$$

where a,b,c ... are neighbors of i.

m,n ... are neighbors of j.

and u,w,x ... are not neighbors of j.

Again, using the logical product notation, the above implication is written:

$$R\ S_{i,j} \rightarrow \{ [\prod_{k=1}^{N_v} (P_{k,m} v\ P_{k,n} v\ P_{k,q} \ldots)(\overline{P}_{k,u}\overline{P}_{k,v}\overline{P}_{k,w}\overline{P}_{k,x} \ldots)] \}$$

For simplification, define the right hand of the above implication as $W_{i,j}$ and rewrite as:

$$R\ S_{i,j} \rightarrow W_{i,j}$$

By exportation, the above equation is changed to:

$$R \rightarrow (S_{i,j} \rightarrow W_{i,j})$$

If all incidence relations hold, this expression should be valid for all combinations of the double subscript pair (i,j).

That is,

$$R \rightarrow [(S_{1,1} \rightarrow W_{1,1})(S_{1,2} \rightarrow W_{1,2})(S_{1,3} \rightarrow W_{1,3}) \ldots (S_{i,j} \rightarrow W_{i,j}) \\ \ldots (S_{n,n} \rightarrow W_{n,n})]$$

Written with $\prod$ notation

$$R \rightarrow \prod_{\substack{i=1 \\ j=1}}^{N_v} (S_{i,j} \rightarrow W_{i,j})$$

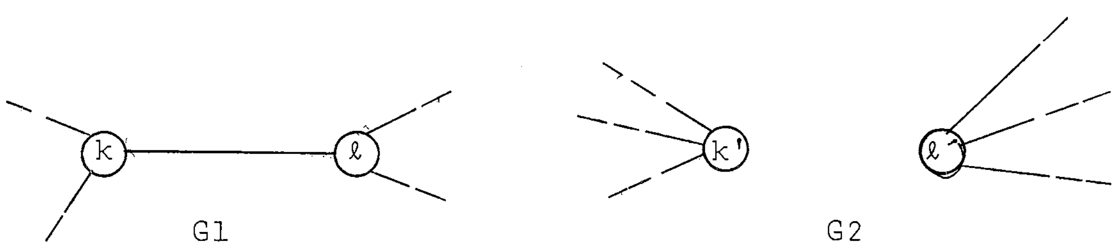
If vertex i of G1 corresponding to vertex j of G2, i.e. $(S_{i,j})$, implies that the subset of vertices of G1 which are neighbors of i is in a one-to-one correspondence with the subset of vertices of G2 which are neighbors of j, $(W_{i,j})$, for all combinations of the subscript pair (i,j), then all incidence relationships hold (R).

Written logically:

$$\left[\prod_{\substack{i=1\\ j=1}}^{N_v} (S_{i,j} \rightarrow W_{i,j}) \right] \rightarrow R$$

Proof by contradiction:

Assume that G1 and G2 are reduced graphs with the same number of vertices and the same number of edges. Suppose that $\prod_{i=1,j=1}^{N_v} (S_{i,j} \rightarrow W_{i,j})$ is true and that R is false. If all incidence do not hold $(\overline{R})$, there is at least one pair of connected vertices of G1 (call them k and ℓ)



G1 G2

which correspond to two vertices k' and ' of G2 which are not connected. If this is true, then there is at least one neighbor of k (meaning ℓ) which does not correspond to any neighbor of k'. Therefore, the neighbor sets of k and k' cannot be in a one-to-one correspondence. Symbolically stated:

$$(S_{k,k'} \rightarrow \overline{W}_{k,k'})$$

The hypothesis stated that $(S_{i,j} \rightarrow W_{i,j})$ for all cases of i and j and therefore it is necessarily true that

$$S_{k,k'} \rightarrow W_{k,k'}$$

A contradiction has been obtained. Therefore the assumption that R is false is false. Therefore, R must be true, which leads to:

$$\prod_{\substack{i=1\\j=1}}^{N_v} (S_{i,j} \rightarrow W_{i,j}) \rightarrow R$$

By material equivalence, the expressions

$$R \rightarrow \prod_{\substack{i=1\\j=1}}^{N_v} (S_{i,j} \rightarrow W_{i,j})$$

and

$$\prod_{\substack{i=1\\j=1}}^{N_v} (S_{i,j} \rightarrow W_{i,j}) \rightarrow R$$

become

$$R \leftrightarrow \prod_{\substack{i=1\\j=1}}^{N_v} (S_{i,j} \rightarrow W_{i,j})$$

An equivalent expression for $(S_{i,j} \rightarrow W_{i,j})$ is $(S_{i,j} \leftrightarrow S_{i,j}W_{i,j})$ as proved by the following truth table:

S	W	SW	$S \rightarrow W$	$S \leftrightarrow SW$	$(S \rightarrow W) \leftrightarrow (S \leftrightarrow SW)$
T	T	T	T	T	T
T	F	F	F	F	T
F	T	F	T	T	T
F	F	F	T	T	T

Thus, we may rewrite the equivalence for R as:

$$R \leftrightarrow \prod_{\substack{i=1\\j=1}}^{N_v} (S_{i,j} \leftrightarrow S_{i,j}W_{i,j})$$

Three of the equations thus far presented are of primary importance for completion of the proof. They are:

$$I \leftrightarrow C \cdot R$$

$$C \leftrightarrow [\, (P_{i,j} \leftrightarrow S_{i,j}) \cdot Q \,]$$

$$R \leftrightarrow [\, \prod_{\substack{i=1\\j=1}}^{N_v} (S_{i,j} \leftrightarrow S_{i,j} W_{i,j}) \,]$$

By the Rule of Replacement[4], these three equations reduce to:

$$I \leftrightarrow (P_{i,j} \leftrightarrow S_{i,j}) \cdot Q \cdot [\, \prod_{\substack{i=1\\j=1}}^{N_v} (S_{i,j} \leftrightarrow S_{i,j} W_{i,j}) \,]$$

Consequences of Logical Relations

In simplyfying the right hand side of the above expression, there are two possible outcomes:

1) If the expression is self-contradictory and all terms vanish, then the expression is false and denies the existence of any one-to-one correspondence between the vertices of G1 and the vertices of G2, such that all incidence relations hold. Conclusion: G1 and G2 are not isomorphic.
2) If the expression is not self-contradictory, it is a true statement and shows existence of a one-

to-one correspondence between the vertices of G1 and the vertices of G2 such that all incidence relations hold.

Conclusion: G1 and G2 are isomorphic.

CHAPTER IV

DESCRIPTION OF PROGRAM

A Fortran program which simplifies the logical expressions described in Chapter III is listed and discussed in Appendix A. This program was written for homogeneous graphs only, but may be converted for the general case by changing subroutine BMXX. An explanation of the necessary steps for this conversion is also included in Appendix A.

Some information about each pair of graphs to be tested for isomorphism must be read into the computer. The only information necessary would be some means of defining the neighbor sets of each vertex of both graphs. Vertex-incidence matrices will serve the purpose while keeping the input at a minimum.

There are two input cards to the program; one for each graph. On each input card there is a graph identification number (NG), the number of vertices in the graph (NV) and elements of the vertex-incidence matrix taken rowwise by columns.

Consideration is now given to how each logical expression will be stored in memory. Boolean Algebra is used throughout the program whenever numerical values represent logical expressions. In Boolean Algebra, any logical expression which is true is replaced by 1. Likewise, any false expression will be replaced by a 0. Of necessity, the Boolean 1's and 0's must have a "weight" or place value, just as the "weights" unit, tens, hundreds, etc. serve to clarify our decadic number system.

A special matrix will be used to handle all expressions of the $W_{i,j}$ form. Consider the matrix:

$$\begin{bmatrix} e_{1,1} & e_{1,2} & e_{1,3} & \cdots \\ e_{2,1} & e_{2,2} & e_{2,3} & \cdots \\ . & . & . & \cdots \\ . & . & . & \cdots \\ . & . & . & \cdots \\ e_{N_v,1} & e_{N_v,2} & e_{N_v,3} & \cdots \end{bmatrix}$$

Each element of this matrix has a truth value (1 or 0) and a "weight" ($P_{i,j}$). If the truth value is 1, the weight is entered into a secession of "or" terms. If the truth is 0, u the complemented weight is entered into a secession of "and" u terms, which is then put into logical product form with the secession of "or" terms. Expressions are created in this u manner only for rows. The logical product of all such row expressions is logically equivalent to the matrix.

Example: Suppose the first row of this matrix were

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

Its equivalent row expression would be $(P_{1,3} \vee P_{1,5} \vee P_{1,6})$ $(\overline{P}_{1,1}\ \overline{P}_{1,2}\ \overline{P}_{1,4})$.

Consider $\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$: its equivalent expression would be

$$[(P_{1,1} \vee P_{1,3})(\overline{P}_{1,2})]\ [(P_{2,1} \vee P_{2,2})(\overline{P}_{2,3})][(P_{3,2} \vee P_{3,3})(\overline{P}_{3,1})]$$

By using commutation [4] this can be rewritten as

$$(P_{1,1} \vee P_{1,3})\ (P_{2,1} \vee P_{2,2})\ (P_{3,2} \vee P_{3,3})\ \overline{P}_{1,1}\ \overline{P}_{2,3}\ \overline{P}_{3,1}$$

This form is exactly the same as for the $W_{i,j}$ with one exception. The $W_{i,j}$ expressions describe only the possible correspondences between the neighbors of i and the neighbors of j, they say nothing of the possible correspondences between the other vertices of the two graphs. Since the $W_{i,j}$ terms give no information as to the vertices which are not neighbors of i and j, it must be assumed that all correspondences between them are possible. This assumption is sometimes referred to as putting the expression in "cannonical" form. The $S_{i,j}W_{i,j}$ expressions can be written from the input vertex-incidence matrices. Suppose that the input vertex-incidence matrices for G1 and G2 are as shown in Figure 5. The neighbors of vertex 1 of G1 are 2, 3, and 6. The neighbors of vertex 1

$$\begin{bmatrix} 011001 \\ 101100 \\ 110010 \\ 010011 \\ 001101 \\ 100110 \end{bmatrix} \qquad \begin{bmatrix} 001011 \\ 001101 \\ 110010 \\ 010011 \\ 101100 \\ 110100 \end{bmatrix}$$

FIGURE 9

Illustration of Program Input

of G2 are 3, 5, and 6.

Thus:

$$P_{1,1}W_{1,1} \leftrightarrow P_{1,1}(P_{2,3} \vee P_{2,5} \vee P_{2,6})\,(P_{3,3} \vee P_{3,5} \vee P_{3,6})$$
$$(P_{6,3} \vee P_{6,5} \vee P_{6,6})\,(\overline{P}_{2,2}\ \overline{P}_{2,4}\ \overline{P}_{3,2}\ \overline{P}_{3,4}\ \overline{P}_{6,2}\ \overline{P}_{6,4})$$

Replacing $P_{1,1}$ with $S_{1,1}$ and rearranging terms the above equivalence is rewritten

$$S_{1,1}W_{1,1} \leftrightarrow [(P_{1,1})\ \overline{P}_{1,2}\overline{P}_{1,3}\overline{P}_{1,4}\overline{P}_{1,5}P_{1,6}]\ [(P_{2,3} \vee P_{2,5} \vee P_{2,6})$$
$$\overline{P}_{2,1}\overline{P}_{2,2}\overline{P}_{2,4}]\ [(P_{3,3} \vee P_{3,5} \vee P_{3,6})\ \overline{P}_{3,1}\overline{P}_{3,2}\overline{P}_{3,4}\ ]$$
$$[\overline{P}_{4,1}\overline{P}_{4,3}\overline{P}_{4,5}\overline{P}_{4,6}]\ \ [\overline{P}_{5,1}\overline{P}_{5,3}\overline{P}_{5,5}\overline{P}_{5,6}\ ]$$
$$[\ (P_{6,3} \vee P_{6,5} \vee P_{6,6})\ \ \overline{P}_{6,1}\overline{P}_{6,2}\overline{P}_{6,4}]$$

The matrix equivalent to $S_{1,1}W_{1,1}$ would then be

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & - & 0 & - & 0 & 0 \\ 0 & - & 0 & - & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

The blank locations are replaced with ones to put the matrix in "cannonical" form.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

This logical matrix can be used to represent Q also.

$$Q = \prod_{i=1}^{N_v} (P_{i,1} \vee P_{i,2} \vee P_{i,3} \cdots \vee P_{i,N_v})$$

$$Q = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \cdots\cdots & 1 \\ 1 & 1 & 1 & 1 & 1 & \cdots\cdots & 1 \\ 1 & 1 & 1 & 1 & 1 & \cdots\cdots & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & & \cdot \\ 1 & 1 & 1 & 1 & 1 & & 1 \end{bmatrix}$$

We now replace each $P_{i,j}$ term in the Q matrix with its equivalent $S_{i,j}W_{i,j}$ term to obtain what may be called a matrix of matrices. Using the vertex-incidence matrices shown in Figure 5, the matrix of matrices is formed as shown at the end of Appendix A.

The matrix in position (1,1) denotes all possible correspondences if $S_{1,1}$ is true. The possible correspondences with vertex 2 of G1 are 3, 5, and 6. The subroutine BMXX(I,J) finds the $P_{i,j}$ matrix according to its arguments I and J and stores the result in $A_{i,j,1}$. The BMXX subroutine also requires the vertex-incidence matrices of both graphs. To see how the two $P_{i,j}$ matrices combine, rewrite Q in the form

$$Q \leftrightarrow (P_{1,1} \vee P_{1,2} \vee P_{1,3} \vee P_{1,4} \vee P_{1,5} \vee P_{1,6}) \cdot$$
$$(P_{2,1} \vee P_{2,2} \vee P_{2,3} \vee P_{2,4} \vee P_{2,5} \vee P_{2,6}) \cdot$$
$$[\prod_{i=3}^{N_v} (P_{i,1} \vee P_{i,2} \cdots P_{i,N_v}) \quad]$$

Using distribution,

$$Q \leftrightarrow (P_{1,1}\, P_{2,1} \vee P_{1,1}\, P_{2,2} \vee P_{1,1}\, P_{2,3} \vee P_{1,1}\, P_{2,5}$$
$$\vee P_{1,1}\, P_{2,6} \vee P_{1,2}\, P_{2,1} \vee P_{1,2}\, P_{2,2} \vee \cdots\cdot P_{1,6}\, P_{2,6}) \cdot$$
$$[\prod_{i=3}^{N_v} (P_{i,1} \vee P_{i,2} \vee \cdots P_{i,N_v})]$$

In this expression $P_{1,1}$ is first combined with all $P_{2,i}$ terms, then $P_{1,2}$ is combined with all $P_{2,i}$ terms, then $P_{1,3}$ is used and so on until $P_{1,6}$ is combined with all $P_{2,i}$ terms.

What happens when the logical product of two $P_{i,j}$ terms is taken? Before examining this question further, it will be easier to start with a simplier example. Suppose the expression T is to be reduced to its lowest terms, T being defined as follows:

$$T = (A \vee B \vee C \vee D)\ \overline{B}\ \overline{E}\ \overline{F}$$

The reduction takes place as follows:

Distribution is used to obtain the form

$$T = A\,\overline{B}\,\overline{E}\,\overline{F} \vee B\,\overline{B}\,\overline{E}\,\overline{F} \vee C\,\overline{B}\,\overline{E}\,\overline{F} \vee D\,\overline{B}\,\overline{E}\,\overline{F}$$

The term $B\,\overline{B}\,\overline{E}\,\overline{F}$ is self-contradictory, while the others are not.

The self-contradictory term is eliminated to obtain

$$T = A\,\overline{B}\,\overline{E}\,\overline{F} \vee B\,C\,\overline{B}\,\overline{E}\,\overline{F} \vee D\,\overline{B}\,\overline{E}\,\overline{F}$$

Again, using distribution the expression simplifies to:

$$T = (A \vee C \vee D)\ \bar{B}\ \bar{E}\ \bar{F}$$

This example can be extended to larger expressions by using commutation.[4] The following reduction is an example of this extension. Suppose a reduction is to be made with

$$\begin{aligned} T = \ & [(P_{1,1})\ (\bar{P}_{1,2}\bar{P}_{1,3}\bar{P}_{1,4}\bar{P}_{1,5}\bar{P}_{1,6})(P_{2,3} \vee P_{2,5} \vee P_{2,6}) \\ & (\bar{P}_{2,1}\bar{P}_{2,2}\bar{P}_{2,4})(P_{3,3} \vee P_{3,5} \vee P_{3,6})(\bar{P}_{3,1}\bar{P}_{3,2}\bar{P}_{3,4}) \\ & (P_{4,2} \vee P_{4,4})(\bar{P}_{4,1}\bar{P}_{4,3}\bar{P}_{4,5}\bar{P}_{4,6})\ (P_{5,2} \vee P_{5,4}) \\ & (\bar{P}_{5,1}\bar{P}_{5,3}\bar{P}_{5,5}\bar{P}_{5,6})(P_{6,3} \vee P_{6,5} \vee P_{6,6})(\bar{P}_{6,1}\bar{P}_{6,2}\bar{P}_{6,4})] \\ & [(P_{1,1} \vee P_{1,2} \vee P_{1,5})(\bar{P}_{1,3}\bar{P}_{1,4}\bar{P}_{1,6})(P_{2,3}) \\ & (\bar{P}_{2,1}\bar{P}_{2,2}\bar{P}_{2,4}\bar{P}_{2,5}\bar{P}_{2,6})\ (P_{3,1} \vee P_{3,2} \vee P_{3,5})(\bar{P}_{3,3}\bar{P}_{3,4}\bar{P}_{3,6}) \\ & (P_{4,1} \vee P_{4,2} \vee P_{4,5})(\bar{P}_{4,3}\bar{P}_{4,4}\bar{P}_{4,6})\ (P_{5,4} \vee P_{5,6}) \\ & (\bar{P}_{5,1}\bar{P}_{5,2}\bar{P}_{5,3}\bar{P}_{5,5})(P_{6,4} \vee P_{6,6})\ (\bar{P}_{6,1}\bar{P}_{6,2}\bar{P}_{6,3}\bar{P}_{6,5})] \end{aligned}$$

Using the rule of commutation, the self-contradictory terms may be brought together regardless of their original position and simplified as shown in the example on page 31. The self-contradictory terms of the above expression are:

$$(P_{2,3} \vee P_{2,5} \vee P_{2,6})\ (\bar{P}_{2,5}\bar{P}_{2,6})$$

$$(P_{3,3} \vee P_{3,5} \vee P_{3,6})\ (\bar{P}_{3,3}\bar{P}_{3,6})$$

$$(P_{4,2} \vee P_{4,4})\ \ \bar{P}_{4,4}$$

$$(P_{5,1} \vee P_{5,3} \vee P_{5,5} \vee P_{5,6})\ \ \bar{P}_{5,6}$$

$$(P_{6,3} \vee P_{6,5} \vee P_{6,6})\ \ \bar{P}_{6,3}$$

$$(P_{1,1} \vee P_{1,2} \vee P_{1,5})\ \ \bar{P}_{1,2}\bar{P}_{1,5}$$

$$(\bar{P}_{3,1}\bar{P}_{3,2})\ (P_{3,1} \vee P_{3,2} \vee P_{3,5})$$

$(\overline{P}_{4,1}\overline{P}_{4,5})\ (P_{4,1} \vee P_{4,2} \vee P_{4,5})$

$(\overline{P}_{5,6})\ (P_{5,4} \vee P_{5,6})$ and

$(\overline{P}_{6,4})\ (P_{6,4} \vee P_{6,6})$

The above example is the same as simplifying the expression $P_{1,1}P_{2,3}$ in the example on page 56. This simplification eliminates terms which have a "one" in one $P_{i,j}$ matrix and a "zero" in the corresponding position of the other. Like digits in corresponding matrix positions do not cause any simplification to occur. The MPY subroutine makes use of these rules when it is called to take the logical product of two $P_{i,j}$ matrices. Using matrix notation $P_{1,1}P_{2,3}$ can be written:

$$\begin{vmatrix} 100000 \\ 001011 \\ 001011 \\ 010100 \\ 010100 \\ 001011 \end{vmatrix} - \begin{vmatrix} 110010 \\ 001000 \\ 110010 \\ 110010 \\ 000101 \\ 000101 \end{vmatrix} = \begin{vmatrix} 1000000 \\ 0010000 \\ 0000100 \\ 0100000 \\ 0001000 \\ 0000010 \end{vmatrix}$$

It follows from the combination $P_{1,1}P_{2,3}$ that if vertices 1 and 2 of G1 correspond to vertices 1 and 3 of G2, respectively, it must be true that vertices 3, 4, 5 and 6 of G1 must correspond to vertices 5, 2, 4 and 6 of G2 respectively. In essence, this is the basis for a reduction in the number of necessary tests. In this form of simplification, there are two irregular circumstances which might occur. The first is that in a series of $P_{i,j}$ correspondences a row or column of all zeros could occur.

Such as this:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

If it did, it would mean that there was no possible correspondence between some vertex of one graph (designated by the row or column of zeros) and any of the vertices of the other graph, and therefore the series is not a possible correspondence.

The second circumstance that might occur is represented by the following matrix.

$$\begin{array}{cc} & \begin{array}{ccccc} & \downarrow \text{ column } 2 & & & \end{array} \\ \begin{array}{c} \\ \\ \text{row } 3 \rightarrow \\ \\ \\ \end{array} & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

In this matrix there is a single P term, which says that with the combinations taken thus far, there is only one possible correspondence; i.e., vertex 3 of G1 must correspond to vertex 2 of G2.

If this $P_{i,j}$ term is replaced by its equivalent S expression, the terms $P_{4,2}$ and $P_{5,2}$ become self-contradictory, and are replaced by zeros. After each step of combining matrices, the program control is transferred to subroutine Zero, which checks for the occurence of these two circumstances and takes appropriate action if they do occur.

Knowing that the subroutines BMXX, MPY and ZERO are available, the problem now is one of combining the $P_{i,j}$ matrices in such a way that all possible correspondences are considered. A nest of DO loops was set up to accomplish this function. The skeleton for this nest is:

```
DO 10 I1 = 1,N
  .
  .
  .
IF (N.EQ.1) GO TO 101
DO 20 I2 = 1,N
  .
  .
  .
IF(N.EQ.2) GO TO 101
DO 30 I3 = 1,N
  .
  .
  .
DO 90 I9 = 1,N
  .
  .
  .
IF(N.EQ.9) GO TO 101
DO 100 I10 = 1,N
  .
  .
  .
GO TO 101
```

This nest of DO statements will test all correspondences. While the program was written for a maximum of 10 vertices per graph, there is no need to test all 10! cases when N<10. For this reason, IF statements are inserted to skip unnecessary tests.

From the rules set out for finding the logical product of two $P_{i,j}$ matrices, it can be seen that in order to combine $P_{i,j}$ with $P_{\ell,k}$ without obtaining a contradiction, there must be a "1" in the ℓ row and k column position of the matrix $P_{i,j}$.

Examine the case of combining $P_{1,1}$ with the matrices $P_{2,i}$. For the cases where i takes on the values 1, 2, or 4, the matrix $P_{1,1}P_{2,i}$ will have a row of zeros and is therefore self-contradictory. By checking for the occurrences of ones in row two of matrix $P_{1,1}$ (this is done by cycling subscript I2) and eliminating those cases where they do not appear, a significant reduction in the number of correspondences which must be interrogated is made. In this example, the products $P_{1,1}\,P_{2,1}$, $P_{1,1}P_{2,2}$ and $P_{1,1}\,P_{2,4}$ are self-contradictory, and when combined with any other $P_{i,j}$ matrices, remain self-contradictory. All the possible correspondences

$$P_{1,1}\;(P_{2,1} \vee P_{2,2} \vee P_{2,4})\;(\prod_{\substack{i=3\\j=1}}^{N_v} P_{i,j})$$

may thus be disregarded. This amounts to a reduction of $3 \cdot 4!$ or 72 tests.

The subscripts i and j are cycled such that matrix $P_{1,1}$ is combined with the first matrix of the form $P_{2,i}$ which fulfills the conditions described above. That matrix would be $P_{2,3}$. As shown in the example on page 32, the combination $P_{1,1}P_{2,3}$ leaves only one correspondence to test. At this level, the number of tests eliminated is 4! -1 or 23. After the above case is fully tested by cycling the subscripts I3 -I6 through the values 1 to 6 and eliminating unneccessary tests at each level of the DO loop nest, the cases $P_{1,1}P_{2,5}$ and $P_{1,1}P_{2,6}$ are examined in the same manner. At this point subscript I1 is changed to 2 and the process is repeated with $P_{1,1}$ replaced by $P_{1,2}$. Each correspondence which survives all levels of testing represents an isomorphism of the two graphs G1 and G2. The correspondence matrices are saved up in groups of ten and then printed out. An example of this printout is shown in Appendix B. An abundance of comment cards have been placed in the program with the hope that they will help to clarify the preceding description.

CHAPTER V

CONCLUSION AND DISCUSSION

The importance of this thesis is that there is now available a tool for classifying graphs. Consider the case where identical vertex incidence matrices written from one graph are used as input to the program. When this is done, the output is a list of all automorphisms of the graph. In some cases, it is easy to determine that two graphs are isomorphic; but it is not easy, in general, to enumerate all possible isomorphisms by inspection or even by the exhaustive process of trying all possible combinations. This is particularly true when the graph is non-planar. It is in this situation that the computer excels in speed and accuracy. Graphs cannot only be classified by their number of automorphisms, but also by their symmetries and the number of permutations necessary to change each automorphic vertex-correspondence matrix to the unit matrix. It is the intention of the author to continue his study of graphs by using the output of the isomorphic testing program for investigation of the symmetry and permutations of isomorphic graphs.

The choice of investigating homogeneous graphs was two-fold. First, it was the simpliest structure to study. Second, homogeneous graphs have special significance in map coloring studies. In particular, homogeneous graphs of order three seem to be the key to the solution of the Four-Color Map problem.

If the program inputs are the vertex-incidence matrices of two graphs which are seemingly non-similar, the program will test for the existence or non-existence of isomorphisms between the two graphs.

The speed of the program depends upon the computer used, the number of vertices in the graphs being tested and also upon the number of vertex-correspondence matrices which must be printed as output. As shown in the examples in appendix B, the case where graphs 11 and 12 were tested took 18.6 seconds to test and print 4 vertex-correspondence matrices. However, when testing graphs 9 and 10, which have 120 isomorphisms, the running time was 32.4 seconds. This increase is due to print-out time since graphs 11 and 12 have the greater number of vertices and thus require more testing time. The time required for typical examples has been within reason. It is interesting to note, however, that a complete graph, (i.e., a graph in which each vertex is connected to every other vertex) with 10 vertices would require approximately 10 hours of testing time and 50 hours printing time for a total of 60 hours or over 2 days. It is apparent that such cases as this should be avoided. This could be accomplished by not printing out the vertex-correspondence matrices and changing the program to terminate after one isomorphism is found.

The author not only intends future investigations of symmetry and permutations, but also plans to discover the answer to some interesting questions which have occurred to him. For instance, what is the number of vertex correspondences which must be chosen before isomorphism or non-isomorphism can be detected? Or, are all graphs which have the same number of vertices, elements and trees isomorphic? The author has found a fascinating life-time project.

LITERATURE CITED

1. Seshu, Sundaram and Myril B. Reed. LINEAR GRAPHS AND ELECTRICAL NETWORKS. Reading, Massachusetts: Addison-Wesley, 1961, pp. 1-19.

2. Whitney, Hassler. "Congruent Graphs and Connectivity of Graphs." AMERICAN JOURNAL OF MATHEMATICS, 54(1932), p. 157.

3. Ore, Oystein. THEORY OF GRAPHS. Providence, Rhode Island: American Mathematical Society Colloquim Publication, XXXVIII (1962), p. 19.

4. Copi, Irving M. INTRODUCTION TO LOGIC. New York: Macmillian, 1961, pp. 277-283.

APPENDIX A

PROGRAM LISTING

MAIN PROGRAM

```
      INTEGER A,SUM,C
C
C   THE ORDER OF SUBSCRIPTING DIMENSIONED VARIABLES IS BY ROW, COLUMN,
C   LAYER
C
      DIMENSION MCG1(10,10),MCG2(10,10),A(10,10,10),C(10,10,10)
      COMMON N,A,MCG1,MCG2,SUM
    1 FORMAT (2I4,70I1/(72I1))
    2 FORMAT (//63X,5HINPUT/54X,24HNODE CONNECTION MATRICES)
    3 FORMAT(40X,5HGRAPH,I4,34X,5HGRAPH,I4/)
    4 FORMAT(1H0,7HGRAPHS ,I4,5H AND ,I4,15H ARE ISOMORPHIC / 1H0,42HALL
     1 POSSIBLE VERTEX CORRESPONDENCES FOLLOW )
    5 FORMAT(1H ,10I1,9(2X,10I1))
    6 FORMAT(1H0,34HNON-EQUAL  ENUMERATION OF VERTICES / )
    7 FORMAT(1H0,38HNO POSSIBLE ISOMORPHIC CORRESPONDENCES /)
    8 FORMAT(1H0,34HNON-EQUAL  ENUMERATION OF ELEMENTS /)
    9 FORMAT(1H0,25HNUMBER OF VERTICES EQUALS,I8,6X,25HNUMBER OF ELEMENT
     1S EQUALS,I8 /)
   16 FORMAT(1H0)
   17 FORMAT (39X,10I1,33X,10I1)
   18 FORMAT(//13H NET TIME IS ,F6.2,8H MINUTES)
   19 FORMAT(//1H ,38HNUMBER OF POSSIBLE ISOMORPHISMS EQUALS,I8)
   21 FORMAT(//1H ,7HGRAPHS ,I4,5H AND ,I4,19H ARE NOT ISOMORPHIC)
   22 FORMAT(59X,16HISOMORPHISM TEST)
   23 FORMAT(1H ,10I1)
C
C   INITIALIZE
C
C   STORAGE BLOCK A IS WHERE THE LOGICAL MATRICES ARE COMBINED
C
C   IF G1 AND G2 ARE ISOMORPHIC, THE VERTEX-CORRESPONDENCE MATRICES
C   ARE SAVED , FOR LATER PRINTOUT, IN STORAGE BLOCK C
C
 1000 CALL CLOCK(TIMEON)
 1001 DO 1002 I=1,10
      DO 1002 J=1,10
      MCG1(I,J)=0
      MCG2(I,J)=0
      DO 1002 K=1,10
      A(I,J,K)=0
      C(I,J,K) = 0
 1002 CONTINUE
C
C   THE VARIABLE LOG KEEPS AN ACCOUNT OF THE NUMBER OF ISOMORPHISMS.
C
      LOG = 0
C
C
C   READ THE IDENTIFICATION NUMBER, NUMBER OF VERTICES, AND VERTEX-
C   INCIDENCE MATRIX FOR GRAPH 1.
```

```
C
      READ(5,1) NG1,NV1,((MCG1(I,J),J=1,NV1),I=1,NV1)
C
C
C   READ THE IDENTIFICATION NUMBER, NUMBER OF VERTICES AND VERTEX-
C   INCIDENCE MATRIX FOR GRAPH 2
C
      READ(5,1) NG2,NV2,((MCG2(I,J),J=1,NV2),I=1,NV2)
C
C   PRINT HEADINGS AND LABELS
C
      CALL PAGE
      WRITE(6,22)
      WRITE(6,2)
      WRITE(6,3) NG1,NG2
C
C   FIND THE MAXIMUM OF NV1 AND NV2. THIS IS DONE SO THAT ALL
C   INFORMATION READ IN IS PRINTED OUT EVEN THO THE NUMBER OF VERTICES
C   OF THE GRAPHS ARE UNEQUAL.
C
      N = MAX0(NV1,NV2)
C
C   PRINT OUT BOTH VERTEX-INCIDENCE MATRICES.
C
      DO 1003 I=1,N
 1003 WRITE(6,17) (MCG1(I,J),J=1,10),(MCG2(I,J),J=1,10)
C
C   CHECK TO SEE THAT THE NUMBER OF VERTICES IN EACH GRAPH ARE THE SAME
C
      IF (NV1.NE.NV2) GO TO 109
C
C
C   BY SUMMING THE NUMBER OF ONES IN EACH VERTEX-INCIDENCE MATRIX WE
C   OBTAIN NUMBERS WHICH ARE TWO TIMES THE NUMBER OF EDGES OF EACH
C   GRAPH. THESE NUMBERS ARE USED TO SEE IF THE TWO GRAPHS HAVE THE
C   SAME NUMBER OF EDGES.
C
      IELS1 = 0
      IELS2 = 0
      DO 1004 I = 1,N
      DO 1004 J=1,N
      IELS1 = IELS1 + MCG1(I,J)
      IELS2 = IELS2+MCG2(I,J)
 1004 CONTINUE
      IF (IELS1.NE.IELS2) GO TO 111
      IELS = IELS1 / 2
C
C   PRINT OUT THE NUMBER OF VERTICES AND THE NUMBER OF EDGES.
C
      WRITE (6,9) NV1,IELS
C
C   ENTER LEVEL 1 OF DO LOOP NEST -- M= 1
C
      DO 10 I1=1,N
      CALL BMXX(1,I1)
      IF(N.EQ.1) GO TO 101
```

```
C
C    EXIT LEVEL 1 - ENTER LEVEL 2
C
      DO 20 I2=1,N
      IF(A(2,I2,1).NE.1) GO TO 20
      CALL BMXX(2,I2)
      CALL MXP(2)
      CALL ZERO(2)
      IF(SUM.EQ.0) GO TO 20
      IF(N.EQ.2) GO TO 101
C
C    EXIT LEVEL 2 - ENTER LEVEL 3
C
      DO 30 I3=1,N
      IF(A(3,I3,2).NE.1) GO TO 30
      CALL BMXX(3,I3)
      CALL MXP(3)
      CALL ZERO(3)
      IF(SUM.EQ.0) GO TO 30
      IF(N.EQ.3) GO TO 101
C
C    EXIT LEVEL 3 - ENTER LEVEL 4,.................ETC.
C
      DO 40 I4=1,N
      IF(A(4,I4,3).NE.1) GO TO 40
      CALL BMXX(4,I4)
      CALL MXP(4)
      CALL ZERO(4)
      IF(SUM.EQ.0) GO TO 40
      IF(N.EQ.4) GO TO 101
      DO 50 I5=1,N
      IF(A(5,I5,4).NE.1) GO TO 50
      CALL BMXX(5,I5)
      CALL MXP(5)
      CALL ZERO(5)
      IF(SUM.EQ.0) GO TO 50
      IF(N.EQ.5) GO TO 101
      DO 60 I6=1,N
      IF(A(6,I6,5).NE.1) GO TO 60
      CALL BMXX(6,I6)
      CALL MXP(6)
      CALL ZERO(6)
      IF(SUM.EQ.0) GO TO 60
      IF(N.EQ.6) GO TO 101
      DO 70 I7=1,N
      IF(A(7,I7,6).NE.1) GO TO 70
      CALL BMXX(7,I7)
      CALL MXP(7)
      CALL ZERO(7)
      IF(SUM.EQ.0) GO TO 70
      IF(N.EQ.7) GO TO 101
      DO 80 I8=1,N
      IF(A(8,I8,7).NE.1) GO TO 80
      CALL BMXX(8,I8)
      CALL MXP(8)
      CALL ZERO(8)
```

```
      IF(SUM.EQ.0) GO TO 80
      IF(N.EQ.8) GO TO 101
      DO 90 I9=1,N
      IF(A(9,I9,8).NE.1) GO TO 90
      CALL BMXX(9,I9)
      CALL MXP(9)
      CALL ZERO(9)
      IF(SUM.EQ.0) GO TO 90
      IF(N.EQ.9) GO TO 101
      DO 100 I10=1,N
      IF(A(10,I10,9).NE.1) GO TO 100
      CALLBMXX(10,I10)
      CALL MXP(10)
      CALL ZERO(10)
      IF(SUM.EQ.0) GO TO 100
      GO TO 101
  100 CONTINUE
   90 CONTINUE
   80 CONTINUE
   70 CONTINUE
   60 CONTINUE
   50 CONTINUE
   40 CONTINUE
   30 CONTINUE
   20 CONTINUE
   10 CONTINUE
C
C    IF THE PROGRAM CYCLES THROUGH ALL DO LOOP INDICES AND LOG REMAINS
C    ZERO, THERE ARE NO POSSIBLE ISOMORPHISMS --GO TO 110 AND PRINT
C    APPROPRIATE MESSAGE
C
      IF (LOG.EQ.0) GO TO 110
      GO TO 106
  112 WRITE(6,19) LOG
  108 CALL CLOCK(TIMEOF)
      TIMERN = (TIMEOF - TIMEON) / 60.0
      WRITE(6,18) TIMERN
      GO TO 1000
C
C    THE PROGRAM ENTERS AT STATEMENT 101 EVERY TIME AN ISOMORPHISM IS
C    ENCOUNTERED.
C
  101 IF(LOG.EQ.0) WRITE(6,4) NG1,NG2
      LOG = LOG + 1
      IF(LOG.EQ.40) CALL PAGE
      LXX = LOG - 40
      LXX = MOD(LXX,50)
      IF(LXX.EQ.0) CALL PAGE
      IST = MOD(LOG,10)
      IF(IST.EQ.0) IST=10
C
C    SAVE THE VERTEX INCIDENCE MATRICES IN C(I,J,IST) AND PRINT A WHOLE
C    ROW WHEN LOG IS A MULTIPLE OF TEN
C
      DO 102 I=1,N
      DO 102 J=1,N
```

```
  102 C(I,J,IST) = A(I,J,N)
      ITEST = MOD(LOG,10)
      IF(ITEST.NE.0) GO TO 103
      WRITE(6,16)
      DO 104 I=1,N
  104 WRITE(6,5) ((C(I,J,K),J=1,10),K=1,10)
C
C    ZERO ALL OF STORAGE BLOCK C AFTER PRINTOUT.
C
      DO 105 I=1,10
      DO 105 J=1,10
      DO 105 K=1,10
  105 C(I,J,K) = 0
C
C    STATEMENTS 106 TO 109 PRINT PARTIAL ROWS OF CORRESPONDENCE MATRICES
C
  103 GO TO (10,20,30,40,50,60,70,80,90,100),N
  106 ITEST = MOD(LOG,10)
      IF(ITEST.EQ.0)  GO TO 112
      WRITE(6,16)
      DO 107 I=1,N
  107 WRITE (6,5) ((C(I,J,K),J=1,10),K=1,ITEST)
      WRITE(6,16)
      WRITE(6,19) LOG
      GO TO 108
C
C    CONTROL IS TRANSFERED TO 109 WHENEVER THERE ARE NON-EQUAL VERTEX
C    SETS
C
  109 WRITE (6,21) NG1,NG2
      WRITE (6,6)
      GO TO 108
C
C    CONTROL IS TRANSFERED TO 110 WHENEVER THERE ARE NO POSSIBLE
C    ISOMORPHISMS.
C
  110 WRITE (6,21) NG1,NG2
      WRITE (6,7)
      GO TO 108
C
C    CONTROL IS TRANSFERED TO 111 WHEN THE NUMBER OF EDGES OF G1 IS NOT
C    EQUAL TO THE NUMBER OF EDGES OF G2.
C
  111 WRITE (6,21) NG1,NG2
      WRITE (6,8)
      GO TO 108
      END
```

```
      SUBROUTINE BMXX(M,L)
C
C
C    THIS SUBROUTINE FINDS THE MATRIX P(M,L) FROM THE VERTEX-INCIDENCE
C    MATRICES (MCG1 AND MCG2) AND STORES THE RESULT IN A(I,J,M).
C
C    M IS THE LEVEL OF THE DO LOOP NEST AT WHICH THE BMXX SUBROUTINE
C    IS CALLED
C
C
      INTEGER A,SUM
      DIMENSION MCG1(10,10),MCG2(10,10),A(10,10,10)
      COMMON N,A,MCG1,MCG2,SUM
C
C
C    THE FOLLOWING DO LOOPS CYCLE I AND J WITH M REMAINING CONSTANT
C
C    L IS THE VALUE OF THE DO LOOP INDEX AT LEVEL M OF THE DO LOOP NEST
C
C
      DO 15 I=1,N
      DO 15 J=1,N
C
C
C    RULES FOR FINDING A(I,J,M)
C
C         1) IF MCG1(M,I) IS NOT EQUAL TO MCG2(L,J), A(I,J,M) = 0
C
C         2) IF MCG1(M,I) IS EQUAL TO MCG2(L,J), THEN A(I,J,M) = 1
C            UNLESS I EQUALS M AND J DOES NOT EQUAL L
C            OR UNLESS J EQUALS L AND I DOES NOT EQUAL M
C
C            THE LAST TWO RESTRAINTS INSURE THE APPEARANCE OF A 1 IN THE
C            (M,L) POSITION OF A(I,J,M) AND ZEROS ELSEWHERE IN ROW M
C            AND COLUMN L
C
C
      IF(MCG1(M,I).EQ.MCG2(L,J)) GO TO 5
      A(I,J,M)=0
      GO TO 15
    5 A(I,J,M)=1
      IF(I.EQ.M) A(I,J,M)=0
      IF(J.EQ.L) A(I,J,M)=0
   15 CONTINUE
      A(M,L,M)=1
      RETURN
      END
```

```
      SUBROUTINE MXP(M)
C
C     THIS SUBROUTINE COMBINES A(I,J,M) WITH A(I,J,M-1) TO PRODUCE THE
C     LOGICAL PRODUCT A(I,J,M)*A(I,J,M-1) WHICH IS STORED BACK INTO
C     THE MATRIX A(I,J,M)
C
      INTEGER A,SUM
      DIMENSION MCG1(10,10),MCG2(10,10),A(10,10,10)
      COMMON N,A,MCG1,MCG2,SUM
      K = M - 1
C
C
C     THE RULES FOR FINDING THE LOGICAL PRODUCT A(I,J,M)*A(I,J,M-1)
C     ARE SUMMARIZED IN THE FOLLOWING TABLE-
C
C                 A(I,J,M)      A(I,J,M-1)      A(I,J,M)*A(I,J,M-1)
C                    0              0
C                    0              1                    0
C                    1              0                    0
C                    1              1                    1
C
C     THESE RULES ARE SATISFIED BY THE ARITHMETIC PRODUCT OF A(I,J,M)
C     AND A(I,J,M-1)
C
C
      DO 5 II=1,10
      DO 5 JJ=1,10
      A(II,JJ,M) = (A(II,JJ,M))*(A(II,JJ,K))
    5 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ZERO(M)
C
C
C   THIS SUBROUTINE TESTS FOR TWO STATES OF THE MATRIX A(I,J,M) WHICH
C   CAN LEAD TO A REDUCTION IN THE TOTAL NUMBER OF CORRESPONDENCES
C   TESTED
C
          1) THE POSSIBILITY OF A ROW SUM OR COLUMN SUM BEING ZERO
C
C         2) THE POSSIBILITY OF ONE AND ONLY ONE 1 IN A ROW OR COLUMN.
C            STATED IN ANOTHER WAY - IF A ROW SUM OR COLUMN SUM IS 1.
C
C
      INTEGER A,SUM,SUMR,SUMC
      DIMENSION MCG1(10,10),MCG2(10,10),A(10,10,10)
      DIMENSION SUMR(10),SUMC(10),NEWSR(10),NEWSC(10)
      COMMON N,A,MCG1,MCG2,SUM
C
C   THE NEXT SIX STATEMENTS COMPUTE N ROW SUMS - SUMR(I)
C
  100 DO 1 I=1,N
      SUMR(I)=0
      DO 2 J=1,N
      SUMR(I)=SUMR(I)+A(I,J,M)
    2 CONTINUE
    1 CONTINUE
C
C   THE NEXT 11 STATEMENTS CHECK THE ROW SUMS FOR ZEROS OR ONES AND
C   TAKE THE APPROPRIATE ACTION.
      DO 15 I=1,N
C
C   IF ANY ROW SUM IS 0, THE VERTEX-CORRESPONDENCE BEING TESTED IS
C   NOT AN ISOMORPHISM  - SET SUM TO 0 AND RETURN TO THE MAIN PROGRAM
C
      IF(SUMR(I).EQ.0) GO TO 50
C
C   IF SUMR(I) IS NOT EQUAL TO 1 OR 0, THERE CAN BE NO REDUCTION OF
C   A(I,J,M) -- GO BACK AND CHECK SUMR(I+1)
C
      IF(SUMR(I).NE.1) GO TO 15
C
C   IF SUMR(I) EQUALS 1 CYCLE J TO FIND WHICH COLUMN THE 1 IS IN AND
C   ZERO ALL POSITIONS OF THAT COLUMN EXCEPT IN ROW I.
C
      DO 10 J=1,N
      IF(A(I,J,M).EQ.1) GO TO 11
   10 CONTINUE
   11 DO 12 L=1,N
      IF(L.EQ.I) GO TO 12
      A(L,J,M)=0
   12 CONTINUE
   15 CONTINUE
C
C   THE OPERATIONS FROM THIS POINT TO STATEMENT 30 PERFORM THE SAME
C   TESTS ON THE COLUMN SUMS /SUMC(I),I=1,N/ THAT WERE PERFORMED ON
C   THE ROW SUMS ABOVE.
```

```
C
      DO 16 J=1,N
      SUMC(J)=0
      DO 17 I=1,N
      SUMC(J)=SUMC(J)+A(I,J,M)
   17 CONTINUE
   16 CONTINUE
      DO 30 J=1,N
      IF(SUMC(J).EQ.0) GO TO 50
      IF(SUMC(J).NE.1) GO TO 30
      DO 25 I=1,N
      IF(A(I,J,M).EQ.1) GO TO 26
   25 CONTINUE
   26 DO 27 L=1,N
      IF(L.EQ.J) GO TO 27
      A(I,L,M)=0
   27 CONTINUE
   30 CONTINUE
C
C
C    CONSIDER THE CASE WHERE UPON ENTERING THE ZERO SUBROUTINE ,
C    A(I,J,M) WAS
C
C                                    01010
C                                    01001
C                                    00010
C                                    11101
C                                    11000
C
C    SINCE SUMR(3) IS 1 , COLUMN 4 IS ZEROED EXCEPT FOR ROW THREE.
C    THUS, AT THIS POINT IN THE SUBROUTINE A(I,J,M) IS
C
C                                    01000
C                                    01001
C                                    00010
C                                    11101
C                                    11000
C
C    SUMR(1) NOW EQUALS 1 BUT HAS ALREADY BEEN CHECKED AND FOUND NOT
C    EQUAL TO 1.  THUS, IF ANY ROW OR COLUMN SUM CHANGES, MORE REDUCTION
C    MAY BE POSSIBLE. THIS IS NOT ALWAYS THE CASE, BUT THE POSSIBILITY
C    DOES EXIST.
C
C
C    THE NEXT 15 STATEMENTS COMPUTE NEW ROW AND COLUMN SUMS AND COMPARE
C    THEM WITH THE PREVIOUS ROW AND COLUMN SUMS.  IF ANY ROW OR COLUMN
C    SUM HAS CHANGED, CONTROL IS TRANSFERED TO THE BEGINING OF THE
C    SUBROUTINE AND A(I,J,M) IS CHECKED AGAIN.  IF ALL ROW AND COLUMN
C    SUMS REMAIN THE SAME, ALL POSSIBLE REDUCTIONS HAVE BEEN MADE.
C    SUM IS SET TO 1 AND CONTROL IS RETURNED TO THE MAIN PROGRAM.
C
C
      DO 31 I=1,N
      NEWSR(I)=0
      DO 32 J=1,N
      NEWSR(I)=NEWSR(I)+A(I,J,M)
```

```
32 CONTINUE
   IF(NEWSR(I).NE.SUMR(I)) GO TO 100
31 CONTINUE
   DO 33 J=1,N
   NEWSC(J)=0
   DO 34 I=1,N
   NEWSC(J)=NEWSC(J)+A(I,J,M)
34 CONTINUE
   IF(NEWSC(J).NE.SUMC(J)) GO TO 100
33 CONTINUE
   SUM = 1
   RETURN
50 SUM = 0
   RETURN
   END
```

DESCRIPTION OF CHANGES
FOR THE GENERAL CASE

To change the program so that it will handle graphs with vertices of different degrees, only the BMXX subroutine must be altered. Assume that the vertex-incidence matrices were written using the lowest degree vertex as vertex 1 and the highest degree vertex as Nv. The vertices have now been separated into sets by degrees. This may not be considered the general case, but if the data were to be set down in any order, it seems natural that the first operation of the computer would be to sort the data and order it in some manner.

We know that two vertices of different degrees cannot correspond under isomorphism. We construct a matrix which allows vertices of the same degree to correspond, but not those of different degree.

A matrix that would perform this function for a graph with 3 vertices of degree 2 and 3 vertices of degree 3 would be

```
111000
111000
111000
000111
000111
000111
```

This matrix isolates the degree sets. If this matrix were combined in the "and" operation with each P(I,J) matrix, which is derived by equating the neighbor sets, we would generate logical matrices which would exhibit equality of degree sets and equality of neighbor sets.

This masking technique would not be at all difficult to accomplish. It would only be a matter of checking the row sum and column sum in the vertex-incidence matrix for each row-column position and setting the corresponding element of the masking matrix to 1 if they were equal and to 0 if they were not. If the masking matrix is found in this way, it seems at first glance that the order of the input data makes no difference.

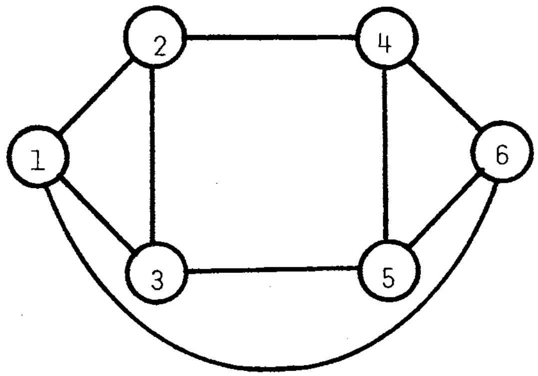
Q Matrix for the Example Discussed on Pages
28-29

1000000000	0100000000	0010000000	0001000000	0000100000	0000010000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
0101000000	1000100000	0001010000	1010000000	0100010000	0010100000
0101000000	1000100000	0001010000	1010000000	0100010000	0010100000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
1000000000	0100000000	0010000000	0001000000	0000100000	0000010000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
0101000000	1000100000	0001010000	1010000000	0100010000	0010100000
0101000000	1000100000	0001010000	1010000000	0100010000	0010100000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
1000000000	0100000000	0010000000	0001000000	0000100000	0000010000
0101000000	1000100000	0001010000	1010000000	0100010000	0010100000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
0101000000	1000100000	0001010000	1010000000	0100010000	0010100000
0101000000	1000100000	0001010000	1010000000	0100010000	0010100000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
0101000000	1000100000	0001010000	1010000000	0100010000	0010100000
1000000000	0100000000	0010000000	0001000000	0000100000	0000010000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
0101000000	1000100000	0001010000	1010000000	0100010000	0010100000
0101000000	1000100000	0001010000	1010000000	0100010000	0010100000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
1000000000	0100000000	0010000000	0001000000	0000100000	0000010000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
0101000000	1000100000	0001010000	1010000000	0100010000	0010100000
0101000000	1000100000	0001010000	1010000000	0100010000	0010100000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
0010110000	0011010000	1100100000	0100110000	1011000000	1101000000
1000000000	0100000000	0010000000	0001000000	0000100000	0000010000

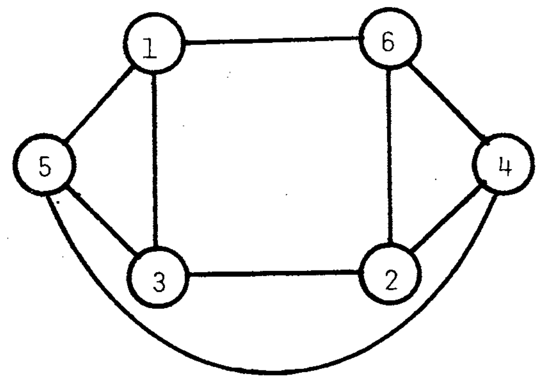
APPENDIX B

PROGRAM OUTPUT

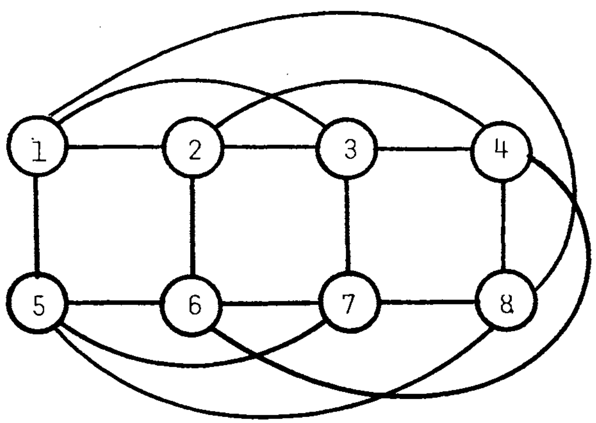
GRAPHS USED AS PROGRAM INPUT



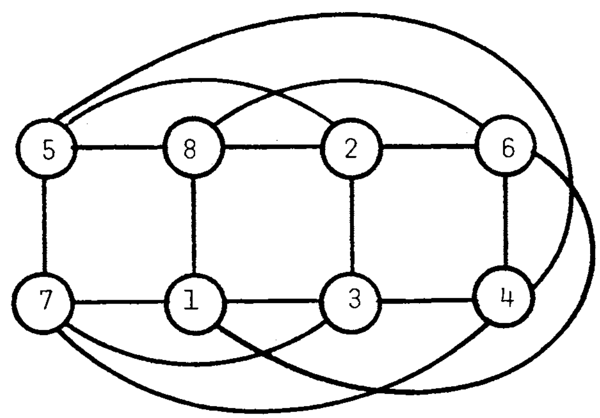
GRAPH 1



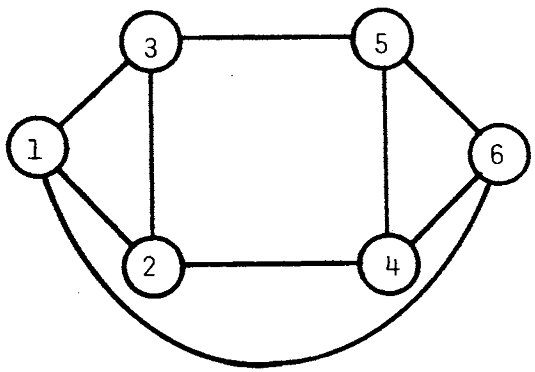
GRAPH 2



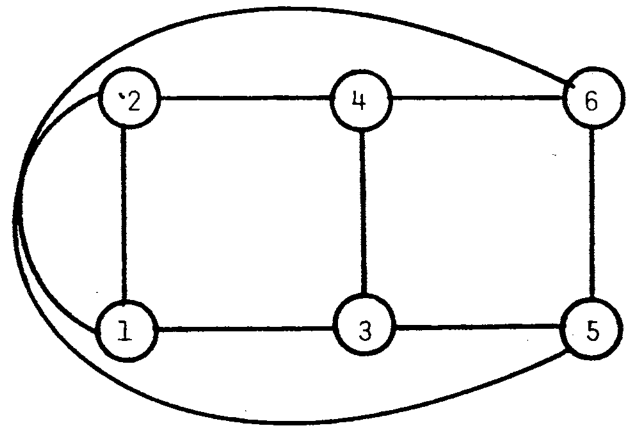
GRAPH 3



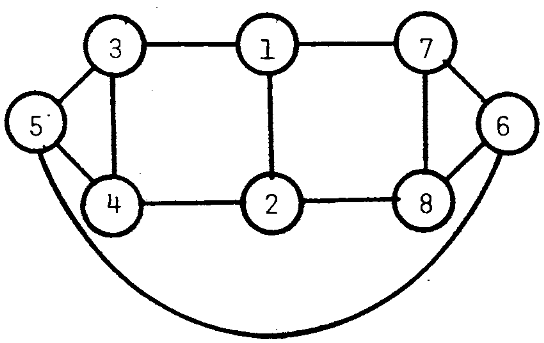
GRAPH 4



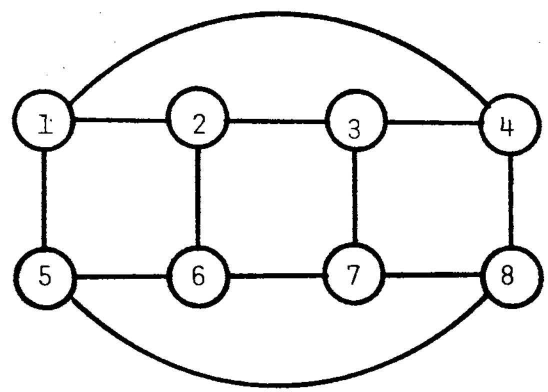
GRAPH 5



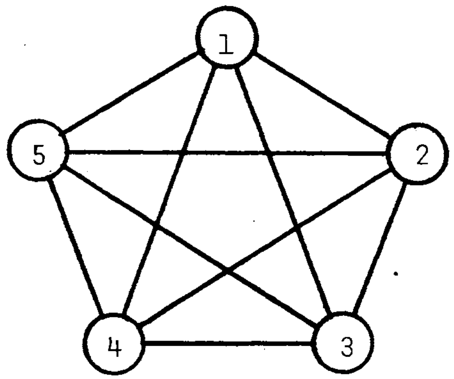
GRAPH 6



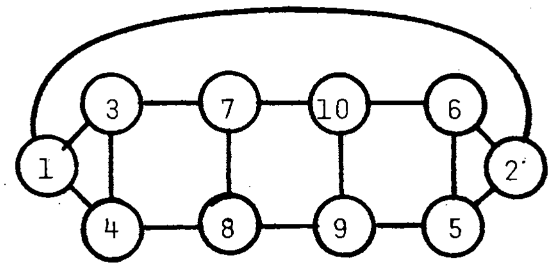
GRAPH 7



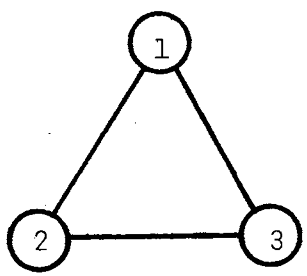
GRAPH 8



GRAPHS 9 and 10



GRAPHS 11 and 12



GRAPHS 13 and 14

Graphs 88, 89, 98, and 99 were fictional and were used only to show the various forms of output.

ISOMORPHISM TEST

INPUT
NODE CONNECTION MATRICES

```
GRAPH    1                                              GRAPH    2

0110010000                                              0010110000
1011000000                                              0011010000
1100100000                                              1100100000
0100110000                                              0100110000
0011010000                                              1011000000
1001100000                                              1101000000
```

NUMBER OF VERTICES EQUALS 6   NUMBER OF ELEMENTS EQUALS 9

GRAPHS 1 AND 2 ARE ISOMORPHIC

ALL POSSIBLE VERTEX CORRESPONDENCES FOLLOW

```
1000000000  1000000000  0100000000  0100000000  0010000000  0010000000  0001000000  0001000000  0000100000  0000100000
0010000000  0000100000  0001000000  0000010000  1000000000  0000100000  0100000000  0000010000  1000000000  0010000000
0000100000  0010000000  0000010000  0001000000  0000100000  1000000000  0000010000  0100000000  0010000000  1000000000
0100000000  0001000000  0000100000  1000000000  0000010000  0001000000  0010000000  1000000000  0000010000  0100000000
0001000000  0100000000  1000000000  0000100000  0001000000  0000010000  1000000000  0010000000  0100000000  0000010000
0000010000  0000010000  0010000000  0010000000  0100000000  0100000000  0000100000  0000100000  0001000000  0001000000

0000010000  0000010000
0100000000  0001000000
0001000000  0100000000
0010000000  0000100000
0000100000  0010000000
1000000000  1000000000
```

NUMBER OF POSSIBLE ISOMORPHISMS EQUALS 12

NFT TIME IS 0.15 MINUTES

ISOMORPHISM TEST

INPUT
NODE CONNECTION MATRICES

GRAPH 3	GRAPH 4
0110100100	0010011100
1011010000	0010110100
1101001000	1101001000
0110010100	0010111000
1000011100	0101001100
0101101000	1101000100
0010110100	1011100000
1001101000	1100110000

NUMBER OF VERTICES EQUALS 8 NUMBER OF ELEMENTS EQUALS 16

GRAPHS 3 AND 4 ARE ISOMORPHIC

ALL POSSIBLE VERTEX CORRESPONDENCES FOLLOW

1000000000	1000000000	0000100000	0000100000
0000001000	0000000100	0000001000	0000000100
0010000000	0000010000	0001000000	0100000000
0001000000	0100000000	0010000000	0000010000
0000000100	0000001000	0000000100	0000001000
0000100000	0000100000	1000000000	1000000000
0100000000	0001000000	0000010000	0010000000
0000010000	0010000000	0100000000	0001000000

NUMBER OF POSSIBLE ISOMORPHISMS EQUALS 4

NET TIME IS 0.19 MINUTES

ISOMORPHISM TEST

INPUT
NODE CONNECTION MATRICES

GRAPH 5	GRAPH 6
0110010000	0110010000
1011000000	1001100000
1100100000	1001100000
0100110000	0110010000
0011010000	0110010000
1001100000	1001100000

NUMBER OF VERTICES EQUALS 6 NUMBER OF ELEMENTS EQUALS 9

GRAPHS 5 AND 6 ARE NOT ISOMORPHIC

NO POSSIBLE ISOMORPHIC CORRESPONDENCES

NET TIME IS 0.05 MINUTES

ISOMORPHISM TEST

INPUT
NODE CONNECTION MATRICES

GRAPH 7	GRAPH 8
0110001000	0101100000
1001000100	1010010000
1001100000	0101001000
0110100000	1010000100
0011010000	1000010100
0000101100	0100101000
1000010100	0010010100
0100011000	0001101000

NUMBER OF VERTICES EQUALS 8 NUMBER OF ELEMENTS EQUALS 12

GRAPHS 7 AND 8 ARE NOT ISOMORPHIC

NO POSSIBLE ISOMORPHIC CORRESPONDENCES

NET TIME IS 0.23 MINUTES

```
                                        ISOMORPHISM TEST

                                             INPUT
                                    NODE CONNECTION MATRICES
                     GRAPH    9                                       GRAPH  10

                    0111100000                                       0111100000
                    1011100000                                       1011100000
                    1101100000                                       1101100000
                    1110100000                                       1110100000
                    1111000000                                       1111000000

NUMBER OF VERTICES EQUALS        5       NUMBER OF ELEMENTS EQUALS       10


GRAPHS     9 AND    10 ARE ISOMORPHIC

ALL POSSIBLE VERTEX CORRESPONDENCES FOLLOW

1000000000  1000000000  1000000000  1000000000  1000000000  1000000000  1000000000  1000000000  1000000000  1000000000
0100000000  0100000000  0100000000  0100000000  0100000000  0100000000  0010000000  0010000000  0010000000  0010000000
0010000000  0010000000  0001000000  0001000000  0000100000  0000100000  0100000000  0100000000  0001000000  0001000000
0001000000  0000100000  0010000000  0000100000  0010000000  0001000000  0001000000  0000100000  0100000000  0000100000
0000100000  0001000000  0000100000  0010000000  0001000000  0010000000  0000100000  0001000000  0000100000  0100000000

1000000000  1000000000  1000000000  1000000000  1000000000  1000000000  1000000000  1000000000  1000000000  1000000000
0010000000  0010000000  0001000000  0001000000  0001000000  0001000000  0001000000  0001000000  0000100000  0000100000
0000100000  0000100000  0100000000  0100000000  0010000000  0010000000  0000100000  0000100000  0100000000  0100000000
0100000000  0001000000  0010000000  0000100000  0100000000  0000100000  0100000000  0010000000  0010000000  0001000000
0001000000  0100000000  0000100000  0010000000  0000100000  0100000000  0010000000  0100000000  0001000000  0010000000

1000000000  1000000000  1000000000  1000000000  0100000000  0100000000  0100000000  0100000000  0100000000  0100000000
0000100000  0000100000  0000100000  0000100000  1000000000  1000000000  1000000000  1000000000  1000000000  1000000000
0010000000  0010000000  0001000000  0001000000  0010000000  0010000000  0001000000  0001000000  0000100000  0000100000
0100000000  0001000000  0100000000  0010000000  0001000000  0000100000  0010000000  0000100000  0010000000  0001000000
0001000000  0100000000  0010000000  0100000000  0000100000  0001000000  0000100000  0010000000  0001000000  0010000000
```

0100000000	0100000000	0100000000	0100000000	0100000000	0100000000	0100000000	0100000000	0100000000	0100000000
0010000000	0010000000	0010000000	0010000000	0010000000	0010000000	0001000000	0001000000	0001000000	0001000000
1000000000	1000000000	0001000000	0001000000	0000100000	0000100000	1000000000	1000000000	0010000000	0010000000
0001000000	0000100000	1000000000	0000100000	1000000000	0001000000	0010000000	0000100000	1000000000	0000100000
0000100000	0001000000	0000100000	1000000000	0001000000	1000000000	0000100000	0010000000	0000100000	1000000000

0100000000	0100000000	0100000000	0100000000	0100000000	0100000000	0100000000	0100000000	0010000000	0010000000
0001000000	0001000000	0000100000	0000100000	0000100000	0000100000	0000100000	0000100000	1000000000	1000000000
0000100000	0000100000	1000000000	1000000000	0010000000	0010000000	0001000000	0001000000	0100000000	0100000000
1000000000	0010000000	0010000000	0001000000	1000000000	0001000000	1000000000	0010000000	0001000000	0000100000
0010000000	1000000000	0001000000	0010000000	0001000000	1000000000	0010000000	1000000000	0000100000	0001000000

0010000000	0010000000	0010000000	0010000000	0010000000	0010000000	0010000000	0010000000	0010000000	0010000000
1000000000	1000000000	1000000000	1000000000	0100000000	0100000000	0100000000	0100000000	0100000000	0100000000
0001000000	0001000000	0000100000	0000100000	1000000000	1000000000	0001000000	0001000000	0000100000	0000100000
0100000000	0000100000	0100000000	0001000000	0001000000	0000100000	1000000000	0000100000	1000000000	0001000000
0000100000	0100000000	0001000000	0100000000	0000100000	0001000000	0000100000	1000000000	0001000000	1000000000

0010000000	0010000000	0010000000	0010000000	0010000000	0010000000	0010000000	0010000000	0010000000	0010000000
0001000000	0001000000	0001000000	0001000000	0001000000	0001000000	0000100000	0000100000	0000100000	0000100000
1000000000	1000000000	0100000000	0100000000	0000100000	0000100000	1000000000	1000000000	0100000000	0100000000
0100000000	0000100000	1000000000	0000100000	1000000000	0100000000	0100000000	0001000000	1000000000	0001000000
0000100000	0100000000	0000100000	1000000000	0100000000	1000000000	0001000000	0100000000	0001000000	1000000000

0010000000	0010000000	0001000000	0001000000	0001000000	0001000000	0001000000	0001000000	0001000000	0001000000
0000100000	0000100000	1000000000	1000000000	1000000000	1000000000	1000000000	1000000000	0100000000	0100000000
0001000000	0001000000	0100000000	0100000000	0010000000	0010000000	0000100000	0000100000	1000000000	1000000000
1000000000	0100000000	0010000000	0000100000	0100000000	0000100000	0100000000	0010000000	0010000000	0000100000
0100000000	1000000000	0000100000	0010000000	0000100000	0100000000	0010000000	0100000000	0000100000	0010000000

```
0001000000 0001000000 0001000000 0001000000 0001000000 0001000000 0001000000 0001000000 0001000000 0001000000
0100000000 0100000000 0100000000 0100000000 0010000000 0010000000 0010000000 0010000000 0010000000 0010000000
0010000000 0010000000 0000100000 0000100000 1000000000 1000000000 0100000000 0100000000 0000100000 0000100000
1000000000 0000100000 1000000000 0010000000 0100000000 0000100000 1000000000 0000100000 1000000000 0100000000
0000100000 1000000000 0010000000 1000000000 0000100000 0100000000 0000100000 1000000000 0100000000 1000000000

0001000000 0001000000 0001000000 0001000000 0001000000 0001000000 0000100000 0000100000 0000100000 0000100000
0000100000 0000100000 0000100000 0000100000 0000100000 0000100000 1000000000 1000000000 1000000000 1000000000
1000000000 1000000000 0100000000 0100000000 0010000000 0010000000 0100000000 0100000000 0010000000 0010000000
0100000000 0010000000 1000000000 0010000000 1000000000 0100000000 0010000000 0001000000 0100000000 0001000000
0010000000 0100000000 0010000000 1000000000 0100000000 1000000000 0001000000 0010000000 0001000000 0100000000

0000100000 0000100000 0000100000 0000100000 0000100000 0000100000 0000100000 0000100000 0000100000 0000100000
1000000000 1000000000 0100000000 0100000000 0100000000 0100000000 0100000000 0100000000 0010000000 0010000000
0001000000 0001000000 1000000000 1000000000 0010000000 0010000000 0001000000 0001000000 1000000000 1000000000
0100000000 0010000000 0010000000 0001000000 1000000000 0001000000 1000000000 0010000000 0100000000 0001000000
0010000000 0100000000 0001000000 0010000000 0001000000 1000000000 0010000000 1000000000 0001000000 0100000000

0000100000 0000100000 0000100000 0000100000 0000100000 0000100000 0000100000 0000100000 0000100000 0000100000
0010000000 0010000000 0010000000 0010000000 0001000000 0001000000 0001000000 0001000000 0001000000 0001000000
0100000000 0100000000 0001000000 0001000000 1000000000 1000000000 0100000000 0100000000 0010000000 0010000000
1000000000 0001000000 1000000000 0100000000 0100000000 0010000000 1000000000 0010000000 1000000000 0100000000
0001000000 1000000000 0100000000 1000000000 0010000000 0100000000 0010000000 1000000000 0100000000 1000000000

NUMBER OF POSSIBLE ISOMORPHISMS EQUALS     120

NET TIME IS   0.54 MINUTES
```

ISOMORPHISM TEST

INPUT
NODE CONNECTION MATRICES

GRAPH 11	GRAPH 12
0111000000	0111000000
1000110000	1000110000
1001001000	1001001000
1010000100	1010000100
0100010010	0100010010
0100100001	0100100001
0010000101	0010000101
0001001010	0001001010
0000100101	0000100101
0000011010	0000011010

NUMBER OF VERTICES EQUALS 10 NUMBER OF ELEMENTS EQUALS 15

GRAPHS 11 AND 12 ARE ISOMORPHIC

ALL POSSIBLE VERTEX CORRESPONDENCES FOLLOW

1000000000	1000000000	0100000000	0100000000
0100000000	0100000000	1000000000	1000000000
0010000000	0001000000	0000100000	0000010000
0001000000	0010000000	0000010000	0000100000
0000100000	0000010000	0010000000	0001000000
0000010000	0000100000	0001000000	0010000000
0000001000	0000000100	0000000010	0000000001
0000000100	0000001000	0000000001	0000000010
0000000010	0000000001	0000001000	0000000100
0000000001	0000000010	0000000100	0000001000

NUMBER OF POSSIBLE ISOMORPHISMS EQUALS 4

NET TIME IS 0.31 MINUTES

ISOMORPHISM TEST

INPUT
NODE CONNECTION MATRICES

GRAPH 13	GRAPH 14
0110000000	0110000000
1010000000	1010000000
1100000000	1100000000

NUMBER OF VERTICES EQUALS 3 NUMBER OF ELEMENTS EQUALS 3

GRAPHS 13 AND 14 ARE ISOMORPHIC

ALL POSSIBLE VERTEX CORRESPONDENCES FOLLOW

1000000000	1000000000	0100000000	0100000000	0010000000	0010000000
0100000000	0010000000	1000000000	0010000000	1000000000	0100000000
0010000000	0100000000	0010000000	1000000000	0100000000	1000000000

NUMBER OF POSSIBLE ISOMORPHISMS EQUALS 6

NET TIME IS 0.05 MINUTES

ISOMORPHISM TEST

INPUT
NODE CONNECTION MATRICES

GRAPH 88	GRAPH 89
0110010000	0110100100
1011000000	1011010000
1100100000	1101001000
0100110000	0110010100
0011010000	1000011100
1001100000	0101101000
0000000000	0010110100
0000000000	1001101000

GRAPHS 88 AND 89 ARE NOT ISOMORPHIC

NON-EQUAL ENUMERATION OF VERTICES

NET TIME IS 0.04 MINUTES

ISOMORPHISM TEST

INPUT
NODE CONNECTION MATRICES

GRAPH 98	GRAPH 99
0110010000	0110010000
1011000000	1011000000
1100100000	1100100000
0100110000	0100110000
0011010000	0011010000
1001100000	1011100000

GRAPHS 98 AND 99 ARE NOT ISOMORPHIC

NON-EQUAL ENUMERATION OF ELEMENTS

NET TIME IS 0.04 MINUTES

VITA

Robert Gary Goodman

Candidate for the degree of

Master of Science

Thesis: A COMPUTER ALGORITHM FOR TESTING THE ISOMORPHIC PROPERTIES OF LINEAR GRAPHS

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Tulsa, Oklahoma, December 9, 1940, the son of Clyde C. Goodman and Frankie B. Goodman.

Education: Attended grade school in Tulsa and Oklahoma City, Oklahoma; graduated from Northeast High School in Oklahoma City, Oklahoma in 1958; attended Central State College, Edmond, Oklahoma from September, 1958 to May, 1959; received the Bachelor of Science degree from Oklahoma State University, with a major in Electrical Engineering, in May, 1963; completed requirements for the Master of Science degree in May, 1965.

Professional Experience: Served as a Graduate Assistant in the Electrical Engineering Department, Oklahoma State University, 1963--1964; employed by Texas Instruments, Inc. since June, 1962; active service with Texas Instruments includes summer, 1962, summer, 1963, and a current five month interim period before returning to school; member of Eta Kappa Nu and Sigma Tau.