On Tutte’s Algorithm for Recognizing Binary Graphic Matroids

A Master’s Paper in Computer Science
by Zhaoxia Qu

©2002 Zhaoxia Qu

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science

April 2002
Abstract

This paper gives a formal description of Tutte’s algorithm for recognizing binary graphic matroids, provides a proof of correctness and a running time analysis. The paper also provides a Java implementation of the algorithm. The program takes as input a binary matrix representing a binary matroid. If the matroid is graphic, it outputs a step-by-step construction and a drawing of a graph whose bond matroid is the matroid. Otherwise, it gives an explanation for why the matroid is not graphic.
# Table of Contents

Abstract

Acknowledgement

List of Figures

1 Introduction

2 Definitions and Theorems
   2.1 Matroid
   2.2 Chain Group
   2.3 The Matroid of A Chain Group
   2.4 Graph
   2.5 Graphic Matroid
   2.6 Rank of A Chain Group
   2.7 Representative Matrix
   2.8 Minors
   2.9 Connectivity
   2.10 Bridges

3 Recognizing Binary Graphic Matroids
   3.1 Row Grouping Algorithm
   3.2 Graphic Test Algorithm
   3.3 Break Down Algorithm
   3.4 Partition Algorithm
   3.5 Classification Algorithm
   3.6 Incidence Matrix Construction Algorithm
   3.7 Switch Algorithm

4 Correctness Proof of the Algorithm
   4.1 The Main Algorithm
Acknowledgement

I would like to thank Dr. Thang N. Bui for his input and guidance. He spent a tremendous amount of time and insight in guiding me for the project and the paper. I also would like to thank Dr. Sandra Kingan and Dr. Robert Kingan, for all the help they have given me. I am grateful to Dr. Pavel Naumov, Dr. Linda Null for reviewing this paper. I am also grateful to Mr. Nathan Fiedler, for the graph drawing part in my program is developed based on his GraphMaker package. Finally, I would like to thank my husband Gutian Xiao and my child Sean Xiao for their support.
# List of Figures

1. The Main Algorithm .................................................. 18
2. Row Grouping Algorithm (RGA) ................................. 20
3. Graphic Test Algorithm (GTA) ................................. 22
4. Break Down Algorithm (BDA) ................................. 23
5. Partition Algorithm (PA) .......................................... 24
6. Classification Algorithm (CA) ................................. 26
7. Incidence Matrix Construction Algorithm (IMCA) ....... 28
8. Switch Algorithm (SA) .............................................. 29
9. The Petersen graph constructed by the program ........... 56
10. The Petersen graph after rearrangement of the vertices of Figure 9 .................................................. 57
11. The graph constructed by the program ....................... 59
12. The graph after rearrangement of the vertices of Figure 11 .................................................. 60
1 Introduction

The concept of matroids originates from Whitney’s paper in matroid theory [7]. When working in the field of graph theory, Whitney noticed several similarities between the ideas of independence and dimension in the study of vector spaces. He used the concept of matroid to formalize these similarities in his paper. Unfortunately, his work was ignored until a breakthrough occurred in 1958 when Tutte characterized those matroids which arise from graphs [3] [4]. Later, in 1965, many mathematicians recognized the importance of matroids in transversal theory. Since then, interest in matroid theory has exploded, and the study of matroid theory has become an important branch of combinatorial mathematics.

Matroid theory draws heavily on both graph theory and linear algebra. One important aspect of matroid theory is to characterize the graphicness of matroids. In 1960, Tutte published a paper describing an algorithm for determining whether a given binary matroid is graphic [2]. We are interested in implementing his algorithm, so that we can have a visual tool for recognizing binary graphic matroids. However, his description of the algorithm is informal and not easy to follow. So we also provide a formal description of the algorithm so that it is straightforward to follow and implement. As many of Tutte’s terminologies are different from the current terminology, his terminology will also be explained in this paper.

The rest of this paper is organized as follows: Section 2 introduces the definitions and some theorems that are necessary to establish the correctness proof of the algorithm. Section 3 describes the algorithm. A correctness proof of the algorithm is provided in Section 4. A running time analysis of the algorithm is given in Section 5. The Java program is described in Section 6. A conclusion is drawn in the last section.
2 Definitions and Theorems

This section introduces some of Tutte’s terminology and theorems. Most of them can be found in Tutte’s papers and book [2] [3] [4] [5] [6]. The proofs of the theorems are mostly from the same place where the theorem is cited. Some theorems may be stated a little differently and the proofs of some theorems may also be slightly different from the original proofs. But they all convey the same idea. The reader should note that some terminology is different from the current terminology.

2.1 Matroid

A matroid $M$ is an ordered pair $(E, Q)$, where $E$ is a finite set, $Q$ is a class of subsets of $E$ such that the following two axioms hold ([6], p1):

**Axiom 1:** No member of $Q$ is a proper subset of another.

**Axiom 2:** Let $a$ and $b$ be distinct members of $E$. Let $X$ and $Y$ be members of $Q$ such that $a \in X \cap Y$ and $b \in X - Y$. Then there exists $Z \in Q$ such that $Z \subseteq (X \cup Y) - \{a\}$ and $b \in Z$.

The members of $E$ are called the cells of $M$. The members of $Q$ are called the circuits of $M$.

**Theorem 2.1.1** ([6], 1.11)

Let $L$ be a class of non-null subsets of $E$. Suppose $L$ satisfies matroid Axiom 2. If $a \in X$ and $X \in L$ then there is a minimal member $Y$ of $L$ such that $a \in Y$ and $Y \subseteq X$.

A member of $L$ is called minimal if does not contain another member of $L$ as a proper subset.

**Proof:** If $a \in X$ and $X \in L$, then there exists $Y \in L$ such that $a \in Y$ and $Y \subseteq X$. Choose such a $Y$ so that $|Y|$, the number of elements in $Y$, is as small as possible. If $Y$ is minimal, then the theorem is proved. And indeed $Y$ is a minimal member of $L$. The reason for this is as follows: Suppose $Y$
is not minimal, then there exists \( Z \in L \) such that \( Z \subseteq Y - \{a\} \), by the definition of \( Y \). Choose an element \( b \in Z \). Since \( L \) satisfies matroid Axiom 2, there exists \( Z' \in L \) such that \( a \in Z' \) and \( Z' \subseteq (Y \cup Z) - \{b\} \). This implies that \( Z' \) is a proper subset of \( Y \) that contains \( a \), which contradicts the choice of \( Y \).

**Theorem 2.1.2** ([6], 1.12)

Let \( L \) be a class of non-null subsets of \( E \). Let \( Q \) be the class of minimal members of \( L \). Suppose \( L \) satisfies matroid Axiom 2. Then \( M = (E, Q) \) is a matroid.

**Proof:** Since \( Q \) is the class of minimal members of \( L \), \( Q \) must satisfy matroid Axiom 1.

Let \( a, b \) be distinct members of \( E \), and \( X, Y \) be members of \( Q \) such that \( a \in X \cap Y \) and \( b \in X - Y \). Since \( L \) satisfies matroid Axiom 2, there exists \( Z \) in \( L \) such that \( b \in Z \) and \( Z \subseteq (X \cup Y) - \{a\} \). By Theorem 2.1.1, there exists a minimal member \( Z' \) of \( L \), which implies \( Z' \in Q \), such that \( b \in Z' \) and \( Z' \subseteq Z \). Hence we conclude that \( Q \) also satisfies matroid Axiom 2. Since \( Q \) satisfies both matroid Axioms 1 and 2, \( M = (E, Q) \) is a matroid.

### 2.2 Chain Group

Let \( E \) be a finite set. Let \( R \) be a commutative ring with a unit element and no divisors of zero. A **chain** on \( E \) over \( R \) is a mapping \( f \) of \( E \) into \( R \). Let \( a \) be an element of \( E \). Then \( f(a) \) is referred to as the **coefficient** of \( a \) in \( f \). The **domain** or **support** of \( f \), written as \( \|f\| \), is the class of all members of \( E \) having nonzero coefficients in \( f \). The chain on \( E \) in which every coefficient is zero is called the **zero chain** on \( E \) and is denoted in formulas by the symbol \( 0 \) ([6], p2).

A **chain group** \( N \) on \( E \) over \( R \) is a class of chains on \( E \) over \( R \) that is closed under the operations of addition and multiplication by an element of \( R \). A chain \( f \in N \) is an **elementary chain** of \( N \) if it is nonzero and if there is no nonzero chain \( g \) of \( N \) such that \( \|g\| \) is a proper subset of \( \|f\| \) ([6], p2).
2.3 The Matroid of A Chain Group

Theorem 2.3.1 ([6], 1.22)

Let $N$ be a chain group on $E$ over $R$. Let $Q$ be the class of supports of elementary chains of $N$. Then $M(N) = (E, Q)$ is a matroid.

Proof: Clearly $Q$ satisfies matroid Axiom 1.

Let $L$ be the class of supports of nonzero chains of $N$. Then $Q$ is a subset of $L$. Let $a$ and $b$ be distinct elements of $E$, and $X$ and $Y$ be members of $L$ such that $a \in (X \cap Y)$ and $b \in X - Y$. Then there exist chains $f$ and $g$ of $N$ such that $\|f\| = X$ and $\|g\| = Y$. Let $h = g(a)f - f(a)g$. Then $h(a) = g(a)f(a) - f(a)g(a) = 0$, and hence $a \not\in \|h\|$. As $b \in X - Y$, $f(b) \neq 0$, $g(b) = 0$, hence $h(b) = g(a)f(b) - f(a)g(b) \neq 0$ and $b \in \|h\|$. Thus $h$ is a nonzero chain of $N$ satisfying $b \in \|h\|$ and $\|h\| \subseteq (X \cup Y) - \{a\}$. Thus we conclude that $L$ satisfies matroid Axiom 2.

Since $Q$ is the class of supports of elementary chains of $N$, $Q$ is the class of minimal members of $L$, by the definition of elementary chain. Hence $M(N) = (E, Q)$ is a matroid, by Theorem 2.1.2.

We refer to $M(N)$ as the matroid of the chain group $N$.

A binary chain group is a chain group over the field of residues mod 2. A binary matroid is the matroid of a binary chain group ([6], p3).

An integral chain group is a chain group over the ring of integers. Let $N$ be an integral chain group. We define a primitive chain of $N$ as an elementary chain of $N$ whose coefficients are restricted to the values 0, 1, and $-1$. A regular chain group is an integral chain group in which every elementary chain is an integral multiple of a primitive chain. A regular matroid is the matroid of a regular chain group ([6], p3).

2.4 Graph

A graph $G$ is an ordered pair $(V(G), E(G))$ where $V(G)$ is a nonempty finite set of elements (called vertices), and $E(G)$ is a finite family of unordered pairs of elements of $V(G)$ (called edges). The two ends of a given edge may
be distinct, in which case the edge is called a link; or, they may be coincident, in which case the edge is called a loop.

Let $G$ be a graph. A $(u,v)$-chain in $G$ is a sequence

$$u = v_0, e_1, v_1, e_2, \ldots, v_{n-1}, e_n, v_n = v$$

where $n \geq 0$, each $v_i$ is a vertex in $V(G)$, and each $e_i$ is the edge $\{v_{i-1}, v_i\}$ in $E(G)$. A graph $G$ is called connected if there is a $(u,v)$-chain for all $u, v \in V(G), u \neq v$.

Let $G$ be a graph. A subgraph $H$ of $G$ is a graph in which $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. $H$ is said to be a subgraph of $G$ generated or induced by $V(H)$ if $E(H) = \{(u,v) | u \in V(H), v \in V(H), (u,v) \in E(G)\}$. A connected component or component of $G$ is a connected, generated subgraph $H$ of $G$ which is maximal in the sense that no larger connected generated subgraph $K$ of $G$ contains all the vertices of $H$.

### 2.5 Graphic Matroid

Let $G$ be a graph. A bond of $G$ is a minimal set of edges of $E(G)$ whose removal increases the number of connected components in $G$. Let the vertices of $V(G)$ be partitioned into two disjoint sets $V_1$ and $V_2$. Let $G_1$ and $G_2$ be the subgraphs of $G$ induced by $V_1$, $V_2$, respectively. Let $H_1$ be a component of $G_1$ and $H_2$ be a component of $G_2$. If there is an edge $e = (u,v)$ such that $u \in V(H_1)$ and $v \in V(H_2)$. Then the set $S$ of all edges in $G$ that have one end in $V(H_1)$ and the other end in $V(H_2)$, including $e$, is a bond of $G$. $H_1$ and $H_2$ are referred to as the two end-graphs of $S$ in $G$.

We orient $G$ by distinguishing one end of each edge as positive and the other as negative. For each $e \in E(G)$ and each $v \in V(G)$, we define an integer $\eta(e,v)$ as follows. If $e$ and $v$ are not incident, or if $e$ is a loop, then $\eta(e,v) = 0$. Otherwise $\eta(e,v) = 1$ or $-1$ according to whether $v$ is the positive or the negative end of $e$.

Chains on $V(G)$ or $E(G)$ over a ring $R$ are called 0-chains and 1-chains, respectively, of $G$ over $R$ ([6], p6).
To each 0-chain $f$ of $G$ over $R$, there corresponds a 1-chain of $G$ over $R$, which is denoted by $\delta f$, and is called the coboundary of $f$. It is defined as follows:

$$(\delta f)(e) = \sum_{v \in V(G)} \eta(e, v)f(v),$$

for each $e \in E(G)$. Thus if $u$ is the positive and $v$ the negative end of $e$ we have

$$(\delta f)(e) = f(u) - f(v).$$

We note that $\delta 0 = 0$. Moreover, coboundaries satisfy the laws

$$\delta(f + g) = \delta(f) + \delta(g),$$

$$\delta(\lambda f) = \lambda \delta(f).$$

Hence the coboundaries of the 0-chains of $G$ over $R$ are the elements of a chain group on $E(G)$ over $R$. We denote this chain group $\Delta_R(G)$ and call it the coboundary group of the oriented graph $G$, over $R$ ([6], p8).

**Theorem 2.5.1** ([6], 1.33)

Let $G$ be a graph. Let $S$ be a subset of $E(G)$. There exists an elementary chain $g$ of $\Delta_R(G)$ such that $S = \|g\|$ if and only if $S$ is a bond of $G$.

**Proof:** ($\Leftarrow$) Assume that $S$ is a bond of $G$. We show that there exists an elementary chain $g$ of $\Delta_R(G)$ such that $S = \|g\|$.

Let $H_1$ and $H_2$ be the two end-graphs of $S$. We define a 0-chain $h$ of $G$ over $R$ as follows. $h(x) = 1$ if $x \in V(H_1)$, and $h(x) = 0$ otherwise. Let $g = \delta h$. Clearly, $\|g\| = S$.

Claim: $g$ is an elementary chain in $\Delta_R(G)$.

Proof of the claim: Suppose, to the contrary, that $g$ is not an elementary chain in $\Delta_R(G)$. Then there exists a nonzero chain $f = \delta h_1$ of $\Delta_R(G)$ such that $\|f\| \subset \|g\| = S$, where $h_1$ is a 0-chain of $G$. The edges of both $H_1$ and $H_2$ must have 0 coefficients in $f$, which implies that the coefficients of the vertices of $H_1$ in $h_1$ are the same and so are the coefficients of the vertices of $H_2$ in $h_1$. Let $e$ be an edge in $S - \|f\|$. Since $f(e) = 0$, the two ends of $e$, one
in $H_1$, the other in $H_2$, must have the same coefficients in $h_1$. But this implies that all the vertices of both $H_1$ and $H_2$ have the same coefficients in $h_1$, which in turn implies that all other edges of $S$ must also have 0 coefficients in $f$ as $e$. Thus $f$ must be a zero chain, a contradiction.

$(\Rightarrow)$ Assume that there exists an elementary chain $g$ of $\Delta_R(G)$ such that $S = \|g\|$. We show that $S$ is a bond of $G$.

There exists a 0-chain $h$ of $G$ such that $g = \delta h$, by definition. Since $g$ is nonzero, there exists an edge $e = (u, v)$ such that $g(e) = h(u) - h(v) \neq 0$. Let $H$ be a graph derived from $G$ by deleting the edges of $S$. Let $H_1$ be the component of $H$ containing $u$, and $H_2$ be the component of $H$ containing $v$. Then for every vertex $x$ in $H_1$, $h(x) = h(u)$; and for every vertex $y$ in $H_2$, $h(y) = h(v)$. Let $h_1$ be a 0-chain of $G$ such that $h_1(x) = 1$ if $x \in H_1$, and $h_1(x) = 0$ otherwise. Then each edge of $\|\delta h_1\|$ has one end in $H_1$. In addition, $\|\delta h_1\| \subseteq S = \|g\|$. Since $g$ is elementary, $\|\delta h_1\| = S$. Hence each edge of $S$ has one end in $H_1$. Let $h_2$ be a 0-chain of $G$ such that $h_2(x) = 1$ if $x \in H_2$, and $h_2(x) = 0$ otherwise. Similarly, we can deduce that $\|\delta h_2\| = S$ and each edge of $S$ has one end in $H_2$. Since each edge of $S$ has one end in $H_1$ and the other end in $H_2$, $S$ is a bond of $G$. 

Let $I$ be the ring of integers. In the above proof, it is easy to see that the coefficients of $\delta h_1$ and $\delta h_2$ are restricted to the values 0, 1 and $-1$. Hence these two 0-chains of $G$ are primitive chains of $\Delta_I(G)$. In addition,

$$\delta h_1 = -\delta h_2,$$

$$g = (g(e))(\delta h_1),$$

where $g(e) \in I$. Hence we can deduce the following theorem:

**Theorem 2.5.2** ([6], 1.34)

\(\Delta_I(G)\) is a regular chain group.

From Theorem 2.5.1, we can deduce that the circuits of the matroid of $\Delta_R(G)$ are defined by the bonds of $G$, and that this matroid is neither dependent of the ring $R$, nor dependent of the orientation assigned to $G$. We call this matroid the bond matroid of $G$, and denote it by $B(G)$. 
A matroid is called \emph{graphic} if it can be represented as the bond matroid of a graph. Note this definition is different from the current terminology. In the current terminology, a matroid is called \emph{cographic} if it can be represented as the bond matroid of a graph.

Since $\Delta_I(G)$ is a regular chain group, Theorem 2.5.3 follows clearly:

\begin{theorem}[6, 1.34]
Every graphic matroid is regular.
\end{theorem}

### 2.6 Rank of A Chain Group

Let $N$ be a chain group on $E$ over $R$. Chains $f_1, \ldots, f_k$ in $N$ are \emph{linearly dependent} if $\sum_{i=1}^{k} r_i f_i = 0$ where the coefficients $r_i$ are elements of $R$ and not all zero. If no such condition holds then the chains are \emph{linearly independent}. The \textit{rank} of $N$, denoted by $r(N)$, is the maximum number of linearly independent chains of $N$ [2].

### 2.7 Representative Matrix

A \emph{representative matrix} $R$ of a chain group $N$ on $E$, and hence its matroid $M(N)$, is a matrix of $|E|$ columns and $r(N)$ rows, where the columns correspond to the elements of $E$, the rows correspond to the members of a set of $r(N)$ linearly independent chains of $N$ and each element $R_{ij}$ is the coefficient of the $j$th element of $E$ in chain $f_i$ ([6], p57). In the case of a binary chain group, $R_{ij}$ only has value 0 or 1.

By standard results of linear algebra, the property of $R$ being a representative matrix of $N$ is invariant under the following “elementary operations”:

1. Permuting the rows
2. Adding to one row a multiple of another by an element of $R$
3. Multiplying a row by $-1$
Through elementary operations on $R$ and possibly a permutation of the columns, a matrix called a standard representative matrix of $N$, in which the first $r(N)$ columns constitute a unit matrix, can be obtained.

**Theorem 2.7.1** ([2], Theorem 1)

In a standard representative matrix $R$ of a binary chain group $N$, each row represents an elementary chain.

**Proof:** In a representative matrix of a binary chain group $N$, the chains of $N$ correspond to a linear combination of the rows of the matrix, and the total number of chains of $N$ is $2^{r(N)}$. Suppose in the matrix $R$ there exists one row that represents a non-elementary chain $f$ of $N$. Then there exist nonzero chains $g$ and $h$ of $N$ such that $f = g + h$, one of which has zero coefficients for all the elements of $E$ corresponding to the first $r(N)$ columns of matrix $R$. But this is impossible, since $g$ and $h$ must correspond to linear combinations of the rows of $R$. \[\square\]

### 2.8 Minors

Let $G$ be a graph. Let $S$ be a subset of $E(G)$. We define $G : S$ as a subgraph of $G$ where $V(G : S) = V(G)$ and $E(G : S) = S$. We define $G \cdot S$, called the reduction of $G$ to $S$, as a subgraph of $G$ where $V(G \cdot S) = \{u, v \mid (u, v) \in S\}$ and $E(G \cdot S) = S$. Clearly, $G : S$ is obtained from $G$ by deleting all the edges of $E - S$, and $G \cdot S$ is obtained from $G : S$ by deleting all the isolated vertices. We define another graph $G \ ctr S$, called the contraction of $G$ to $S$ with $V(G \ ctr S) = \{A \mid A$ is a component of $G : (E(G) - S)\}$, and $E(G \ ctr S) = \{(A,B) \mid (u,v) \in S, u \in V(A), v \in V(B)\}$. By deleting the isolated vertices of $G \ ctr S$, we obtain the reduced contraction $G \times S = (G \ ctr S) \cdot S$ of $G$ to $S$. If $G$ is oriented, we will assume that $G : S$, $G \cdot S$, $G \ ctr S$, and $G \times S$ to be correspondingly oriented ([6], p13).

Let $N$ be a chain group on a set $E$ over $R$. Let $S$ be any subset of $E$. A restriction of a chain $f$ of $N$ to $S$ is a chain $g$ on $S$ such that $g(x) = f(x)$ for every $x \in S$. The restrictions to $S$ of the chains of $N$ constitute a chain
2 DEFINITIONS AND THEOREMS

group on $S$. We denote this chain group by $N \cdot S$, and call it the reduction of $N$ to $S$. The restrictions to $S$ of those chains $f$ of $N$ for which $\|f\| \subseteq S$ also constitute a chain group. We denote this chain group by $N \times S$, and call it the contraction of $N$ to $S$. Let $g$ be the restriction to $S$ of a chain $f$ of $N$. If $f(x) = 0$ for every $x \in E - S$, we say that $f$ is the zero-extension of $g$ ([6], p14).

Consider the oriented graph $G \cdot S$. If $f$ is the coboundary of a 0-chain $h$ in $G$, it is clear that the restriction of $f$ to $S$ is the coboundary, in $G \cdot S$, of the restriction of $h$ to $V(G \cdot S)$. It follows that:

**Theorem 2.8.1** ([6], 2.211)

The reduction to $S$ of the coboundary group of $G$ over $R$, $(\Delta_R(G)) \cdot S$, is the coboundary group of the reduction to $S$ of $G$ over $R$, $\Delta_R(G \cdot S)$.

Now consider the oriented graph $G \times S$. Suppose that $g = \delta h$ is a chain of $\Delta_R(G \times S)$, and $f$ is the zero-extension of $g$ in $\Delta_R(G)$. We define a 0-chain $h_1$ of $G$ as follows: $h_1(x) = h(X)$ for every $x \in X$ if $X \in V(G \times S)$, and $h_1(x) = 0$ otherwise. Clearly, $f = \delta h_1$.

Now suppose that $f = \delta h_1$ is a chain in $\Delta_R(G)$ where $\|f\| \subseteq S$ and $g$ is the restriction of $f$ to $S$. Then $h_1$ has the same coefficient for the vertices in the same component of $G : (E - S)$. We could define a 0-chain $h$ of $G \times S$ as follows: $h(X) = h_1(x)$ for $x \in X$. Clearly $g = \delta h$. Hence the following theorem could be deduced:

**Theorem 2.8.2** ([6], 2.213)

The contraction to $S$ of the coboundary group of $G$ over $R$, $(\Delta_R(G)) \times S$ is the coboundary group of the reduced contraction to $S$ of $G$ over $R$, $\Delta_R(G \times S)$.

Let $M = (E, Q)$ be a matroid. Let $S$ be any subset of $E$. Let $L$ be the class of all non-null intersections of $S$ with members of $Q$. Let $L'$ be the set of all minimal members of $L$. Clearly, $L$ satisfies matroid Axiom 2 as does $Q$. Hence $(S, L')$ is a matroid, by Theorem 2.1.2. We denote this matroid by $M \cdot S$, and call it the reduction of $M$ to $S$. Let $Q'$ be the class of all members
of $Q$ that are subsets of $S$. It is easy to see that $(S, Q')$ is also a matroid. We denote it by $M \times S$, and call it the contraction of $M$ to $S$ ([6], p17).

Let $N$ be a chain group on $S$, then by a comparison of the definitions, we can deduce the following theorem:

**Theorem 2.8.3** ([6], 2.31)

1. $M(N \cdot S) = M(N) \cdot S$
2. $M(N \times S) = M(N) \times S$

Let $G$ be a graph. Let $S$ be any subset of $E(G)$. By Theorem 2.8.3, Theorem 2.8.1, and Theorem 2.8.2, we can easily deduce the following two theorems:

**Theorem 2.8.4** ([6], 2.321)

The reduction to $S$ of the bond matroid of $G$, $B(G) \cdot S$, is the bond matroid of the reduction to $S$ of $G$, $B(G \cdot S)$.

**Theorem 2.8.5** ([6], 2.322)

The contraction to $S$ of the bond matroid of $G$, $B(G) \times S$, is the bond matroid of the reduced contraction to $S$ of $G$, $B(G \times S)$.

By the above two theorems, we can deduce that:

**Theorem 2.8.6** ([5], 4.10)

A minor of a graphic matroid is graphic.

Let $M = (E, Q)$ be a matroid. Let $S$ be a subset of $E$ and $T$ be a subset of $S$. Then we have the following theorem ([6], p18):

**Theorem 2.8.7** ([6], p18)

1. $(M \times S) \times T = M \times T$,
2. $(M \cdot S) \cdot T = M \cdot T$,
3. $(M \cdot S) \times T = (M \times (E - (S - T))) \cdot T$, 

4. \((M \times S) \cdot T = (M \cdot (E - (S - T))) \times T.\)

**Proof:** Theorem 2.8.7, 1: Can be easily deduced from the definitions.

For Theorem 2.8.7, 2: Let \(X\) be a circuit of \((M \cdot S) \cdot T\). Then \(X\) is the intersection with \(T\) of a circuit of \(M \cdot S\), and therefore of a circuit of \(M\). Hence there exists a circuit \(Y\) of \(M \cdot T\) such that \(Y \subseteq X\).

Conversely, suppose \(Y\) is a circuit of \(M \cdot T\). Then there is a circuit \(Z\) of \(M\) such that \(Z \cap T = Y\). But \(Z\) contains a circuit \(Z_1\) of \(M \cdot S\) which meets \(Y\), by theorem 1. Hence there is a circuit \(X\) of \((M \cdot S) \cdot T\) such that \(X \subseteq Z_1 \cap Y \subseteq Z \cap T = Y\).

Applying matroid Axiom 1 to \((M \cdot S) \times T\) and \(M \times T\), we deduce that these matroids are identical.

For Theorem 2.8.7, 3: Let \(X\) be a circuit of \((M \cdot S) \times T\). Then \(X\) is a circuit of \(M \cdot S\) contained in \(T\). It follows that there is a circuit \(X_1\) of \(M\) such that \(X_1 \cap T = X\), and \(X_1 \cap (S - T)\) is null. But then \(X_1\) is a circuit of \(M \times (E - (S - T))\). Hence there is a circuit \(Y\) of \((M \times (E - (S - T))) \cdot T\) such that \(Y \subseteq X_1 \cap T = X\).

Conversely, suppose \(Y\) is a circuit of \((M \times (E - (S - T))) \cdot T\). Then there is a circuit \(Y_1\) of \(M \times (E - (S - T))\) such that \(Y = Y_1 \cap T\). But then \(Y_1\) is a circuit of \(M\) such that \(Y_1 \cap (S - T)\) is null. Hence there is a circuit \(X\) of \(M \cdot S\) such that \(X \subseteq Y_1 \cap S = Y_1 \cap T = Y\). But then \(X \subseteq Y\), and so \(X\) is a circuit of \((M \cdot S) \times T\).

Applying matroid Axiom 1 to \((M \cdot S) \times T\) and \((M \times (E - (S - T))) \cdot T\), we deduce that these matroids are identical.

For Theorem 2.8.7, 4: It is obtained by writing \(E - (S - T)\) for \(S\) in Theorem 2.8.7, 3.

A chain group of the form \((N \cdot S) \times T\) is called a *minor* of \(N\) ([6], p16). Similarly, a matroid of the form \((M \cdot S) \times T\) is called a *minor* of \(M\) ([6], p19). Minors of \(M\) include \(M\) itself and all the reductions and contractions of \(M\), since \(M = (M \cdot E) \times E\), \(M \cdot S = (M \cdot S) \times S\), and \(M \times S = (M \cdot E) \times S\).
2.9 Connectivity

Let $M = (E, Q)$ be a matroid. A separator of $M$ is a subset $S$ of $E$ such that no circuit of $M$ meets both $S$ and $E - S$. A separator is elementary if it is non-null and contains no other non-null separator. Clearly, the elementary separators of $M$ are disjoint and their union is $E$ ([6], p27).

A matroid $M = (E, Q)$ is connected if it has no separator other than $E$ and its null subset ([6], p28).

**Theorem 2.9.1** ([6], 3.12)

A subset $S$ of $E$ is a separator of a matroid $M = (E, Q)$ if and only if $M \cdot S = M \times S$.

**Proof:** Suppose $S$ is a separator of $M$. Then a circuit of $M$ has a nonnull intersection with $S$ if and only if it is itself a subset of $S$. This implies $M \cdot S = M \times S$.

Conversely, suppose $M \cdot S = M \times S$. Let $Y$ be any circuit of $M$. If it meets $S$, it contains a circuit $Y_1$ of $M \cdot S$, that is, of $M \times S$. But then $Y_1$ is a circuit of $M$. Hence, $Y = Y_1 \subseteq S$ by matroid Axiom 1. We deduce that $S$ is a separator of $M$.

Let $M = (E, Q)$ be a matroid. If $S$ is an elementary separator of $M$, then we call the matroid $M \cdot S$, that is, $M \times S$, a component of $M$ ([6], p28).

**Theorem 2.9.2** ([6], 3.13)

Let $S$ be a separator of a matroid $M = (E, Q)$. Then, for each $T \subseteq E$, $S \cap T$ is a separator of both $M \cdot T$ and $M \times T$.

**Proof:** Let $Y$ be a circuit of $M \cdot T$ or $M \times T$. Then there is a circuit $Z$ of $M$ such that $Y = Z \cap T$. But either $Z \subseteq S$ or $Z \subseteq E - S$. Hence either $Y \subseteq S \cap T$, or $Y \subseteq T - (S \cap T)$.

Let $G$ be a graph. Let $S$ be a subset of $E(G)$. The vertices of attachment of $S$, denoted by $W(S)$, is the common vertices of $G \cdot S$ and $G \cdot [E(G) - S]$. If $|W(S)| = 1$, then the single vertex of attachment of $S$ is called a cut-vertex of $G$. The graph $G$ is called separable if it either has a cut-vertex or is
disconnected. The maximal non-null nonseparable subgraphs of a graph $G$ are called *separates* of $G$.

**Theorem 2.9.3** ([6], 3.24)

*If $G \cdot E(G)$ is nonseparable, then $B(G)$ is a connected matroid.*

### 2.10 Bridges

Let $Y$ be a circuit of a matroid $M = (E, Q)$. The *bridges* of $Y$ in $M$ are defined as the elementary separators of the matroid $M \cdot (E - Y)$. To each such bridge $B$ there corresponds a matroid $M \times (B \cup Y)$. We refer to this matroid as a *$Y$-component* of $M$.

**Theorem 2.10.1** ([6], 4.13)

*Let $Y$ be a circuit of a matroid $M = (E, Q)$. Let $B$ be a bridge of $Y$ in $M$. Then the $Y$-component $M \times (B \cup Y)$ is connected unless

$$[M \cdot (E - Y)] \times B = M \times B.$$*

*If this condition holds, however, then $B$ and $Y$ are the elementary separators of $M \times (B \cup Y)$. 

**Proof:** Let $Z$ be a separator of $M \times (B \cup Y)$. Then either $Y \subseteq Z$ or $Y \cap Z = \phi$. Moreover $Z \cap B$ is a separator of the matroid $[M \times (B \cup Y)] \cdot B$, by Theorem 2.9.2; and this matroid is $[M \cdot (E - Y)] \times B$, by Theorem 2.8.7 4. But $B$ is an elementary separator of $M \cdot (E - Y)$. Hence $B \subseteq Z$ or $B \cap Z = \phi$, by hypothesis. Accordingly, $B$ and $Y$ are the only possible nontrivial separators of $M \times (B \cup Y)$.

The necessary and sufficient condition for $B$ and $Y$ to be separators of $M \times (B \cup Y)$ is

$$[M \times (B \cup Y)] \cdot B = [M \times (B \cup Y)] \times B,$$

by Theorem 2.9.1. Since

$$[M \times (B \cup Y)] \cdot B = [M \cdot (E - Y)] \times B,$$
by Theorem 2.8.7 4, and

\[ [M \times (B \cup Y)] \times B = M \times B, \]

by Theorem 2.8.7 1, this condition is equivalent to

\[ [M \cdot (E - Y)] \times B = M \times B. \]

**Theorem 2.10.2** ([2], Theorem 2)

*If the matroid \( M = (E, Q) \) is connected then each \( Y \)-component of \( M \) is connected.*

**Proof:** Let \( B \) be a bridge of \( Y \) in \( M \). Suppose the \( Y \)-component of \( M \), \( M \times (B \cup Y) \) is disconnected. Then

\[ M \times B = [M \cdot (E - Y)] \times B, \]

by Theorem 2.10.1. Since

\[ [M \cdot (E - Y)] \times B = [M \cdot (E - Y)] \cdot B, \]

by Theorem 2.9.1, and

\[ [M \cdot (E - Y)] \cdot B = M \cdot B, \]

by Theorem 2.8.7 2, we have

\[ M \times B = M \cdot B \]

Hence \( B \) is a separator of \( M \), by Theorem 2.9.1, which implies that \( M \) is disconnected, which contradicts the fact that \( M \) is connected. \( \blacksquare \)

**Theorem 2.10.3** ([6], 4.51)

*Let \( M \) be the bond matroid of a graph \( G \). Let \( Y \) be a circuit of \( M \) having at most one bridge in \( M \). Then there is a vertex \( a \) of \( G \) such that \( Y \) is the set of all edges of \( G \) joining \( a \) to other vertices.*
Proof: Let $H$ be the graph obtained from $G$ by deleting the edges of $Y$. Then $H = G : (E(G) - Y)$, and we can write the bond matroid of $H$, $B(H)$ as:

$$B(H) = B(G : (E(G) - Y)) = B(G \cdot (E(G) - Y)) = B(G) \cdot (E(G) - Y),$$

by Theorem 2.8.4. Clearly, for each component $C$ of $H$, $E(C)$ is a separator of $B(H)$. So it must be a superset of an elementary separator $B$ of $B(H)$ if it is non-null. If $Y$ has at most one bridge in $B(G)$, that is, $B(H)$ has just one elementary separator, then at most one component of $H$ contains edges, other components are edgeless. Thus for the two end-graphs of $Y$, at least one of them must be edgeless, i.e., it just contains one single vertex. Let’s refer to this single vertex as $a$. Since $a$ is one end-graph of $Y$ in $G$, $Y$ is the set of all edges of $G$ joining $a$ to other vertices.

For a bridge $B$ of a circuit $Y$ in a binary matroid $M$, if the circuits of the matroid $M \times (B \cup Y) \cdot Y$ are disjoint subsets $S_1, S_2, \ldots, S_k$ of $Y$ whose union is $Y$, we say that $B$ partitions $Y$, and that $\{S_1, S_2, \ldots, S_k\}$ is the partition of $Y$ determined by $B$. Then the standard representative matrix of $M \times (B \cup Y) \cdot Y$ has just one nonzero element in each column, and its rows corresponds to the circuits $S_i$. Add to the matrix an extra row which has a 1 in each column, we obtain the incidence matrix of a graph $G_Y$ corresponding to this matroid $M \times (B \cup Y) \cdot Y$.

**Theorem 2.10.4** If the matroid $M \times (B \cup Y)$ is graphic, and let $G_{BY}$ be a graph whose bond matroid represents $M \times (B \cup Y)$. Then $G_Y = G_{BY} \cdot Y$.

Proof: Since $Y$ has only one bridge in the matroid $M \times (B \cup Y)$, by Theorem 2.10.3, there must be a vertex $a$ in graph $G_{BY}$ such that $Y$ is the set of all edges joining $a$ to other vertices in $G_{BY}$. Hence all $G_{BY} \cdot Y$ contains are the vertex $a$, all the edges incident on $a$, and the other ends of these edges. Each set of edges incident on $a$ and another vertex is a bond in $G_{BY} \cdot Y$. Since

$$B(G_{BY} \cdot Y) = M \times (B \cup Y) \cdot Y = B(G_Y)$$

and $G_Y$ also contains a vertex with all the edges in $G_Y$ incident on it. It is easy to see that $G_{BY} \cdot Y = G_Y$, by a comparison of them.
Theorem 2.10.5 ([2], Theorem 3)

If $M$ is regular then each bridge of $Y$ in $M$ partitions $Y$.

Let $B_1$ and $B_2$ be bridges of $Y$ in $M$ which determine partitions $P_1 = \{P_{11}, P_{12}, \ldots, P_{1m}\}$ and $P_2 = \{P_{21}, P_{22}, \ldots, P_{2n}\}$ of $Y$, respectively. We say $B_1$ and $B_2$ do not overlap if there exists $P_{1i}$ and $P_{2j}$ such that $P_{1i} \cup P_{2j} = Y$. Otherwise, we say they overlap. $Y$ is an even circuit of $M$ if the following two conditions hold:

1. Each bridge of $Y$ in $M$ partitions $Y$.
2. The bridges of $Y$ in $M$ can be arranged into two disjoint classes so that no two members of the same class overlap.

Theorem 2.10.6 ([2], Theorem 5)

In a graphic matroid every circuit is even.

Theorem 2.10.7 ([2], Theorem 8)

Let $Y$ be an even circuit of a connected binary matroid $M$ such that every $Y$-component of $M$ is graphic. Then $M$ is graphic.

3 Recognizing Binary Graphic Matroids

This section describes the algorithm for recognizing binary graphic matroids in detail. The main algorithm is described first, then the sub-algorithms that are used in the algorithm are described in the subsections of this section.

Given as input a binary matrix $R$, representing a binary matroid $M$, the main algorithm breaks down the matrix $R$ into a list $L_R$ of matrices, each representing a component of $M$. Then for each matrix $R_i$ in $L_R$, it calls the Graphic Test Algorithm (GTA) with $R_i$ as input. If the GTA returns null for a matrix $R_i$ in $L_R$, indicating that the component of $M$ represented by $R_i$ is not graphic, it returns a null matrix indicating that $M$ is not graphic. Otherwise, for each matrix $R_i$ in $L_R$, the GTA returns a matrix $I_i$, which is the incidence matrix of a graph whose bond matroid is the component of $M$.
3 RECOGNIZING BINARY GRAPHIC MATROIDS

Figure 1: The Main Algorithm

Input: a binary matrix \( R \) representing a binary matroid \( M \)
Output: NULL or an incidence matrix of a graph whose bond matroid is \( M \)

1. Check if \( R \) is a zero matrix
   2. if \( R \) is a zero matrix
       return \( R \)
   3. Check if \( R \) is in standard form
   4. if \( R \) is not in standard form
       Convert \( R \) to standard form through elementary row operations
   5. Run the Row Grouping Algorithm (RGA) using \( R \) as input.
   6. Let \( L_\rho \) be the list of row groups that RGA returns.
   7. Initialize an empty matrix \( I \)
   8. for each row group \( \rho_i \) in \( L_\rho \)
       9. Initialize an empty matrix \( R_i \):
          for each row index \( j \in \rho_i \)
              10. Append the row \( r_j \) of \( R \) to \( R_i \)
       endfor
   11. Run the Graphic Test Algorithm (GTA) using \( R_i \) as input
   12. Let \( I_i \) be the output of GTA
   13. if \( I_i \) is NULL
       return NULL
   14. else
       for each row \( r_j \) of matrix \( I_i \)
           15. Append \( r_j \) to matrix \( I \)
       endfor
   16. endelse
   17. return \( I \)

represented by \( R_i \). The main algorithm then returns a matrix \( I \), whose rows are the rows of each \( I_i \) altogether. The returned matrix \( I \) is the incidence matrix of a graph whose bond matroid is \( M \).

The main algorithm is described in detail in Figure 1. We follow the standard convention that after a return statement is executed in an algorithm, the algorithm terminates immediately. We assume that the row index and column index of a matrix start at 1. We define a matrix in standard form as a matrix that is in or can be in the form \([U|X]\) within a permutation of columns, where \( U \) is a unit matrix.
3.1 Row Grouping Algorithm

Let $R$ be a binary matrix. A row group $\rho$ of $R$ is a set of row indices of $R$ such that if $i$ and $j$ are in $\rho$ then there exists a column index $k$ such that $R[i][k] = R[j][k] = 1$. For each row group $\rho$, the corresponding column group is the set $\{ j \mid R[i][j] = 1, i \in \rho \}$.

The Row Grouping Algorithm takes as input a binary matrix $R$. It uses breath-first search to find an ordered list of row groups, and a corresponding ordered list of column groups. The detailed algorithm is described in Figure 2.

3.2 Graphic Test Algorithm

Given as input a binary matrix $R$ in standard form, representing a binary connected matroid $M$, The Graphic Test Algorithm (GTA) returns NULL if it determines that $M$ is not graphic. Otherwise, it returns a matrix, which is the incidence matrix of a graph whose bond matroid is $M$.

The Graphic Test Algorithm is basically a divide and conquer algorithm. In the base case when $R$ has no more than two 1s in each column, it constructs and returns a matrix $I$, whose rows are all the rows of $R$ in addition to a row $r$, which is the mod 2 sum of all the rows of $R$.

When the base condition is not satisfied, GTA breaks the problem down into smaller subproblems by running the Break Down Algorithm (BDA) using as input the matrix $R$ and a column index $j$, where the column $c_j$ of $R$ has at least three 1s. It returns NULL if BDA returns NULL. Otherwise, BDA returns an ordered list of matrices, $L_R$, each has less rows than $R$, in company with a corresponding list of column groups, $L_\kappa$, and a set $Y$ of column indices.

In the conquer step, the algorithm calls itself recursively with each matrix $R_i$ in $L_R$ as input. It returns NULL if the recursive algorithm returns NULL for a matrix $R_i$ in $L_R$. Otherwise, the recursive algorithm returns a matrix $I_i$ for each input matrix $R_i$ in $L_R$. Let $L_I$ denote the list of $I_i$.

In the combine step, the algorithm calls the Partition Algorithm (PA) with $L_R$ and $Y$ as input. It returns NULL if PA returns NULL. Otherwise,
Figure 2: Row Grouping Algorithm (RGA)

Input: a binary matrix $R$

Output: an ordered list of row groups and a corresponding ordered list of column groups

where each row/column group is a set of row/column indices of $R$

Initialize an empty ordered list of row groups, $L_\rho$

Initialize an empty ordered list of column groups, $L_\kappa$

Mark each row and column of $R$ unvisited

for each row $r_i$ in matrix $R$

if $r_i$ is not visited

Mark $r_i$ visited

if $r_i$ is not a zero row

Initialize an empty queue $Q$

$\rho \leftarrow \emptyset$ // $\rho$: a row group, which is a set of row indices of $R$

$\kappa \leftarrow \emptyset$ // $\kappa$: a column group, which is a set of column indices of $R$

Enqueue $(i, Q)$

while $Q$ is not empty

$\hspace{1em} j \leftarrow \text{Dequeue}(Q)$

$\rho \leftarrow \rho \cup \{j\}$

for each column $c_k$ of $R$

if $c_k$ is not visited and $R[j][k] = 1$

Mark $c_k$ visited

$\kappa \leftarrow \kappa \cup \{k\}$

for each row $r_l$ of $R$

if $r_l$ is not visited and $R[l][k] = 1$

Mark $r_l$ visited

Enqueue $(l, Q)$

endfor

endfor

endwhile

Append($\rho$, $L_\rho$)

Append($\kappa$, $L_\kappa$)

endif

endfor

return $L_\rho$ and $L_\kappa$
PA returns an ordered list $L_P$ of partitions of $Y$, where each $P_i$ in $L_P$ is obtained from the matrix $R_i$ in $L_R$. Then the algorithm runs the Classification Algorithm (CA) using $L_P$, $Y$, $L_\kappa$, and $L_I$ as input. It returns NULL if CA returns NULL. Otherwise, the output $C$ of CA includes two lists for each list $L_P$, $L_\kappa$, and $L_I$, and two corresponding lists of partition element pair lists. The algorithm then calls the Incidence Matrix Construction Algorithm (IMCA) with $C$ and $Y$ as input, and returns the matrix constructed and returned by IMCA. The detailed algorithm is described in Figure 3.

### 3.3 Break Down Algorithm

The Break Down Algorithm is used in the Graphic Test Algorithm (GTA). The input to it in GTA is a binary matrix $R$, and a column index $j$ of $R$, which satisfies $R[a][j] = 1$ for at least three distinct rows $r_a$ of $R$. The algorithm tries to break down the matrix $R$ into a list of matrices $L_R$. Each matrix $R_i$ in $L_R$ has fewer rows than the matrix $R$, and has one common row, which is a row $r_a$ of $R$ that satisfies $R[a][j] = 1$. The rest of the rows of $R_i$ are those rows of $R$ whose indices are in the same row group returned by the Row Grouping Algorithm (RGA) when using as input a matrix $R'$, which is the same as matrix $R$ except that $r_a$ is replaced by a zero row and for each $b \in Y = \{b | R[a][b] = 1\}$, the column $c_b$ is replaced by a zero column.

The algorithm will try at most three distinct rows of $R$, where each row $r_a$ satisfies $R[a][j] = 1$. When trying a row $r_a$, if the list of row groups that the RGA returns contains only one entry, then the breaking down will not be successful since $L_R$ will contain only one matrix, which is essentially the same matrix as $R$. If this is the case for all three rows that are tried, then the algorithm returns NULL. Otherwise, it returns the list of matrices in company with the list of column groups returned by the RGA, and the set $Y = \{b | R[a][b] = 1\}$, for the successfully tried row $r_a$ of $R$. The algorithm is shown in detail in Figure 4.
Figure 3: Graphic Test Algorithm (GTA)

1. **Input**: a binary matrix \( R \) in standard form, representing a binary connected matroid \( M \)
2. **Output**: NULL or an incidence matrix of a graph whose bond matroid is \( M \)

\[
j \leftarrow 1 \quad // \text{column index of } R
\]

5. **while** \( j \) is a column index of \( R \) and column \( c_j \) has no more than two 1s
6. \quad Increment \( j \) by one

8. **//Base case:**
9. \quad if \( j \) is no longer a column index of \( R \)
10. \quad Construct a row \( r \), which is the mod 2 sum of all the rows of \( R \)
11. \quad Construct a matrix \( I \), whose rows are the rows of \( R \) in addition to row \( r \)
12. \quad return \( I \).

17. **//Divide**: divide the problem into smaller subproblems
16. **//We enter this part if \( j \) is the index of a column of \( R \) that has more than two 1s
17. \quad Run the Break Down Algorithm (BDA) using \( R \) and \( j \) as input
18. \quad Let \( O \) be the output of BDA

20. **//Conquer**: solve the subproblems recursively
21. **if** \( O \) is NULL \quad return NULL
22. **else**
23. \quad Let \( Y \) be the set of column indices in \( O \)
24. \quad Let \( L_\kappa \) be the list of column groups in \( O \)
25. \quad Let \( L_R \) be the list of matrices in \( O \)
26. \quad Initialize an empty ordered list of matrices, \( L_I \)
27. **for** each matrix \( R_i \) in \( L_R \)
28. \quad Run the Graphic Test Algorithm (GTA) recursively using \( R_i \) as input
29. \quad Let \( I_i \) be the output of GTA
30. **if** \( I_i \) is NULL \quad return NULL
31. **else** \quad Append \( I_i, L_I \)
32. **endfor**
33. **endelse**

35. **//Combine**: combine the solutions of the subproblems into a solution for the problem
36. \quad Run the Partition Algorithm (PA) using \( L_R \) and \( Y \) as input
37. \quad Let \( L_P \) be the output of PA
38. **if** \( L_P \) is NULL \quad return NULL
39. **else**
40. \quad Run the Classification Algorithm (CA) using \( L_P, Y, L_\kappa \) and \( L_I \) as input
41. \quad Let \( C \) be the output of CA
42. **if** \( C \) is NULL \quad return NULL
43. **else**
44. \quad Run the Incidence Matrix Construction Algorithm (IMCA)
45. \quad using \( C \) and \( Y \) as input
46. \quad Let \( I \) be the incidence matrix returned by IMCA
47. \quad return \( I \)
48. **endelse**
49. **endelse**
**Figure 4**: Break Down Algorithm (BDA)

1. **Input**: a binary matrix $R$, and a column index $j$ of $R$, which satisfies $R[a][j] = 1$ for at least three distinct rows $r_a$ of $R$
2. **Output**: NULL or an ordered list of matrices, a corresponding ordered list of column groups and a set of column indices

3. $c \leftarrow 1$ // a counter
4. $t \leftarrow false$ // true if the break down is successful
5. $a \leftarrow 0$ // row index of $R$
6. Initialize an empty ordered list of row groups, $L_\rho$
7. Initialize an empty ordered list of column groups, $L_\kappa$
8. $Y \leftarrow \emptyset$ // $Y$: stores the set $\{b | R[a][b] = 1\}$
9. while $t$ is false and $c \leq 3$
10. 11. Increment $c$ by one
12. 13. Increment $a$ by one
14. 15. while $R[a][j] = 0$
16. 17. Increment $a$ by one
18. 19. $Y \leftarrow \emptyset$
20. 21. for each column $c_b$ of $R$
22. 23. if $R[a][b] = 1$
24. 25. $Y \leftarrow Y \cup \{b\}$
26. endfor
27. Construct a matrix $R'$ by first letting $R' = R$
28. Then in $R'$, replace $r_a$ by a zero row,
29. and for each $b \in Y$, replace column $c_b$ by a zero column
30. Run the Row Grouping Algorithm (RGA) using $R'$ as input
31. $L_\rho \leftarrow$ the list of row groups that the RGA returns
32. $L_\kappa \leftarrow$ the list of column groups that the RGA returns
33. if $L_\rho$ contains more than one entry
34. 35. set $t$ to true
36. endwhile
37. if $c > 3$
38. Return NULL
39. else // $t$ is true, i.e., break down was successful
40. Initialize an empty ordered list of matrices $L_R$
41. for each row group $\rho_i$ in $L_\rho$
42. 43. Initialize an empty matrix $R_i$
44. for each row index $k \in \rho_i$
45. 46. append the row $r_k$ of $R$ to matrix $R_i$
47. Append the row $r_a$ of $R$ to matrix $R_i$
48. Append($R_i$, $L_R$)
49. endfor
50. endelse
51. Return $L_R$, $L_\kappa$, $Y$
3.4 Partition Algorithm

Given as input a list $L_R$ of matrices, and a set $Y$ of column indices, the Partition Algorithm returns a list $L_P$ of partitions of $Y$ where each partition $P_i$ in $L_P$ is obtained from the matrix $R_i$ in $L_R$. Or, it returns NULL if no partition of $Y$ could be obtained from a matrix $R_i$ in $L_R$. The detailed algorithm is shown in Figure 5.
3.5 Classification Algorithm

Given as input a set $Y$ of column indices, an ordered list $L_P$ of partitions of $Y$, a corresponding ordered list $L_I$ of incidence matrices, another corresponding ordered list $L_\kappa$ of column groups, the Classification Algorithm tries to divide the list $L_P$ into two lists, $L_{P_1}$ and $L_{P_2}$ according to the following criterion: for any two partitions $P_i$ and $P_j$ in the same list, there must be a partition element pair $\mathcal{P} = (I, J)$ such that $I \cup J = Y$, $I \in P_i$ and $J \in P_j$. If this arrangement can not be made, the algorithm returns NULL. Otherwise, it adds two lists, $L_{L_1}$ and $L_{L_2}$. In each list $L_{L_k}$, each entry $L_i$ is a list of partition element pairs, and has $i$ entries. Each entry $\mathcal{P}_j$ in $L_i$ is a pair $(I, J)$ where $I \cup J = Y$, $I \in P_{i+1}$, $J \in P_j$, and $P_{i+1}$ is the $(i + 1)$th entry, $P_j$ is the $j$th entry of $L_{P_k}$. The algorithm also divides the list, $L_I$ into two lists, $L_{I_1}$ and $L_{I_2}$, and the list $L_\kappa$ into two lists, $L_{\kappa_1}$ and $L_{\kappa_2}$. For each $I_i$ in $L_I$ and $\kappa_i$ in $L_\kappa$, if $P_i$ in $L_P$ is the $j$th entry of $L_{P_k}$, then $I_i$ is the $j$th entry in $L_{I_k}$ and $\kappa_i$ is the $j$th entry of $L_{\kappa_k}$. The algorithm is described in detail in Figure 6.

3.6 Incidence Matrix Construction Algorithm

The Incidence Matrix Construction Algorithm is used in the Graphic Test Algorithm, after the recursive Graphic Test Algorithm returns an incidence matrix $I_i$ for each $R_i$ used as its input. It constructs an incidence matrix $X$ of a graph whose bond matroid is the matroid $M$ represented by the input matrix $R$ of the Graphic Test Algorithm. The detailed algorithm is shown in Figure 7.

The incidence matrix $X$ is constructed through unioning all the matrices $I_i$. The list of all the incidence matrices $I_i$ was previously divided into two lists, $L_{I_1}$ and $L_{I_2}$, in the Classification Algorithm (CA). The algorithm first constructs an incidence matrix $X_i$ from each list $L_{I_i}$. Then it constructs $X$ by unioning the two incidence matrices $X_1$ and $X_2$.

For each list $L_{I_i}$, if it has just one entry $I_1$, then the matrix $X_i$ is $I_1$. Otherwise, $X_i$ is constructed by unioning the first two matrices in $L_{I_i}$ to form a new incidence matrix, then unioning the newly formed incidence matrix
Figure 6: Classification Algorithm (CA)

Input: a set $Y$ of column indices, an ordered list $L_I$ of incidence matrices,
a corresponding ordered list $L_\kappa$ of column groups,
a corresponding ordered list $L_P$ of partitions of $Y$,

Output: NULL or two ordered lists for each input list
and two ordered lists of partition element pair lists

Initialize two empty ordered lists of partitions of $Y$, $L_P_1$ and $L_P_2$
Initialize two empty ordered lists of incidence matrices, $L_I_1$ and $L_I_2$
Initialize two empty ordered lists of column groups, $L_\kappa_1$ and $L_\kappa_2$
Initialize two empty ordered lists of partition element pair lists,
$L_L_1$ and $L_L_2$

for each $P_i$ in $L_P$
    if $i = 1$
        Append($I_i$, $L_I_1$), Append($\kappa_i$, $L_\kappa_1$), Append($P_i$, $L_P_1$)
    else if $P_i$ is the last entry in $L_P$ and $L_P_2$ is empty
        Append($I_i$, $L_I_2$), Append($\kappa_i$, $L_\kappa_2$), Append($P_i$, $L_P_2$)
    else
        $p \leftarrow$ false  //true if $P_i$ has been appended
        $j \leftarrow 1$
        $k \leftarrow 1$
        Initialize an empty ordered list of partition element pairs $L_P$
        while $j \leq$ number of entries in $L_P$ and $p$ is false
            if there exists a partition element pair $\mathcal{P} = (I, J)$
                such that $I \cup J = Y$, $I \in P_i$ and $J \in P_j$
                Append($\mathcal{P}$, $L_P$)
                if $P_j$ is the last entry in $L_P$
                    Append($I_i$, $L_I_k$), Append($\kappa_i$, $L_\kappa_k$)
                    Append($P_i$, $L_P_k$), Append($L_P$, $L_L_k$)
                    set $p$ to true
                else
                    Increment $j$ by one
            else  //no such pair
                if $k = 1$
                    if $L_P_2$ is empty
                        Append($I_i$, $L_I_1$), Append($\kappa_i$, $L_\kappa_1$), Append($P_i$, $L_P_1$)
                        set $p$ to true
                    else  //L_P_2 is not empty
                        $k \leftarrow 2$
                        $j \leftarrow 1$
                        Empty $L_P$
                else  //k = 2
                    return NULL
        endelse
    endwhile
endfor
return $L_P_1$ and $L_P_2$, $L_I_1$ and $L_I_2$,
$L_\kappa_1$ and $L_\kappa_2$, $L_L_1$ and $L_L_2$
with the third matrix, if any, and so on.

Each incidence matrix in $L_1$ and $L_2$ must have a common row $r$ where $Y$ is the set of indices of those columns that have a 1 in row $r$. For each list $L_i$, each step of the union of two incidence matrices $I_a$ and $I_b$ (each having a common $r$) takes place as follows:

Construct an incidence matrix $I$, whose rows are those rows of $I_b$ except the common row $r$, in addition to those rows of $I_a$ except the common row $r$. Then transform $I$ into a matrix having the common row $r$.

The transformation of $I$ takes place as follows: The algorithm uses the information obtained in the previously run Classification Algorithm to obtain a target row $t$. And then it runs the switch algorithm to transform row $t$ into the common row $r$.

### 3.7 Switch Algorithm

The Switch Algorithm is used in the Incidence Matrix Construction Algorithm. The input to it is a set $Y$ of column indices, an incidence matrix $I$, and a row index $t$. The algorithm transforms the row $r_i$ of $I$ into a row $r$ where $Y$ is the set of indices of those columns that have a 1 in row $r$. The detailed algorithm is shown in Figure 8.

### 4 Correctness Proof of the Algorithm

The main algorithm and the sub-algorithms used in the main algorithm are described in detail in Figures 1 through 8. We shall prove the correctness of these algorithms according to the descriptions in these figures line by line. Since Row Grouping Algorithm is quite straightforward, the proof of it will be skipped.

#### 4.1 The Main Algorithm

The Main Algorithm is described in Figure 1.
Figure 7: Incidence Matrix Construction Algorithm (IMCA)

```
Input: a set \( Y \) of column indices, two ordered lists of partitions of \( Y \), \( L_{P_1} \) and \( L_{P_2} \),
  two ordered lists of incidence matrices, \( L_{I_1} \) and \( L_{I_2} \),
  two ordered lists of column groups, \( L_{\kappa_1} \) and \( L_{\kappa_2} \),
  two ordered lists of partition element pair lists, \( L_{L_1} \) and \( L_{L_2} \)
Output: an incidence matrix of a grpah

Initialize two empty matrices, \( X_1 \) and \( X_2 \)

for \( i = 1 \) to 2
  for each matrix \( I_j \) in \( L_{I_i} \)
    Find the row \( r_a \) in \( I_j \) where the set \( \{ b | I_j[a][b] = 1 \} = Y \)
    if \( r_a \) is not the last row
      Swap \( r_a \) with the last row
    if \( j = 1 \)
      \( X_i \leftarrow I_j \)
    else
      Let \( m_i \) and \( m_j \) be the number of rows in \( X_i \) and \( I_j \), respectively
      Construct a matrix \( X \) of \( m_i + m_j - 2 \) rows,
      in which the first \( m_i - 1 \) rows are the first \( m_i - 1 \) rows of \( X_i \)
      and the last \( m_j - 1 \) rows are the first \( m_j - 1 \) rows of \( I_j \)
      \((I, J) \leftarrow \) the partition element pair \( P_1 \) in \( L_j \) of \( L_{L_i} \)
      for each partition element pair \( P_k = (S, T) \) in \( L_j \), where \( L_j \) in \( L_{L_i} \)
        if \( S = I \)
          for each \( b \in \kappa_k \), where \( \kappa_k \in L_{\kappa_i} \)
            Mark each row \( r_a \) of \( X \) where \( X[a][b] = 1 \)
      endfor
    endelse
  endfor
  Let \( b \) be a column index in \( Y - I \)
  Search the first \( m_i - 1 \) rows of \( X \)
  for the row \( r_a \), where \( X[a][b] = 1 \)
  Initialize an empty queue \( Q \)
  \( t \leftarrow a \)
  while \( r_a \) is not marked
    for each row \( r_c \) where \( c < m_i \) and \( X[a][d] = X[c][d] = 1 \) for some \( d \)
      if \( r_c \) is marked
        \( t \leftarrow c \), break the while loop
      else
        Enqueue\((c, Q)\)
      endelse
  endwhile
  Run the Switch Algorithm (SA) using \( X, Y, t \) as input.
  \( X_i \leftarrow \) the matrix returned by SA
endfor
endfor

Let \( m_1 \) and \( m_2 \) be the number of rows in \( X_1 \) and \( X_2 \), respectively
Construct a matrix \( X \) of \( m_1 + m_2 - 2 \) rows,
  in which the first \( m_1 - 1 \) rows are the first \( m_1 - 1 \) rows of \( X_1 \)
  and the last \( m_2 - 1 \) rows are the first \( m_2 - 1 \) rows of \( X_2 \)
return \( X \)```
Figure 8: Switch Algorithm (SA)

Input: a matrix $X$, a set $Y$ of column indices, and a row index $t$ of $X$

Output: modified matrix $X$

$I \leftarrow \{ b \mid X[t][b] = 1 \}$

while $Y - I \neq \emptyset$

Let $b$ be a column index such that $b \in Y - I$

Search the column $c_b$ of $X$ for two rows $r_a$, $r_c$

such that $X[a][b] = X[c][b] = 1$ and $a > b$

Construct a matrix $X' = X$.

In $X'$, replace row $r_t$ and row $r_a$ with zero rows, and for each $d$

such that $X[a][d] = 1$ or $X[t][d] = 1$, replace column $c_d$ with a zero column

$\rho \leftarrow \emptyset$

if the row $r_c$ of $X'$ is a zero row

$\rho \leftarrow \rho \cup \{ c \}$

else

Run the Row Grouping Algorithm (RGA) using $X'$ as input

Let $L_\rho$ be the list of row groups returned by RGA

$\rho \leftarrow$ the row group that contains $c$

endelse

for each $e \in \rho$

for each column $c_d$ in $X$

if $X[e][d] = 1$

if $X[a][d] = 1$

$X[a][d] \leftarrow 0$

$X[t][d] \leftarrow 1$

else if $X[t][d] = 1$

$X[a][d] \leftarrow 1$

$X[t][d] \leftarrow 0$

endif

endif

endfor

$I \leftarrow \{ b \mid X[t][b] = 1 \}$

endwhile

if $r_t$ is not the last row of $X$

Swap $r_t$ with the last row in $X$

return $X$
Lines 4 through 6: If the input matrix $R$ is a zero matrix, then $R$ represents an empty matroid $M$. $R$ is also an incidence matrix of a graph $G$ that has no link. Clearly the bond matroid of $G$ is $M$.

Lines 7 through 9: If $R$ is not in standard form, then elementary row operations are used to transform $R$ into a standard form. These operations do not alter the property of the matrix being a representative matrix of the same matroid $M$ (see Section 2.7). Within a permutation of columns, the matrix $R$ in standard form is a standard representative matrix. In a standard representative matrix, each row represents an elementary chain (Theorem 2.7.1). In other words, the set $Y = \{ b \mid R[a][b] = 1 \}$ for each row $r_a$ in $R$ represents a circuit of $M$.

Lines 10 through 11: With $R$ as its input, the Row Grouping Algorithm (RGA) returns a list $L_\rho$ of row groups and a corresponding list $L_\kappa$ of column groups. Since a row group $\rho$ of $R$ is a set of row indices of $R$ such that if $i$ and $j$ are in $\rho$ then there exists a column index $k$ such that $R[i][k] = R[j][k] = 1$; and for each row group $\rho$, the corresponding column group is the set $\{ j \mid R[i][j] = 1, i \in \rho \}$, it is easy to verify that each column group $\kappa$ in $L_\kappa$ represents an elementary separator $S$ of $M$.

Lines 12 through 27: Lines 13 through 26 are a for loop. Inside the for loop, in lines 14 through 17, for each row group $\rho_i$ in $L_\rho$, a matrix $R_i$, whose rows are those rows of $R$ whose indices are in $\rho_i$, is constructed. As the corresponding column group $\kappa_i$ represents an elementary separator $S_i$ of $M$, $R_i$ is a binary matrix in standard form representing the matroid $M \cdot S_i = M \times S_i$, which is a connected component of $M$. Hence in line 18, each input $R_i$ to the Graphic Test Algorithm (GTA) is a valid input. For each matrix $R_i$, GTA outputs $I_i$. We shall prove in Section 4.2 that GTA returns NULL if it determines that the matroid represented by its input matrix is not graphic, or returns an incidence matrix of a graph whose bond-matroid is the matroid represented by its input matrix. In line 20, if $I_i$ is NULL, then the matroid represented by $R_i$, $M \times S_i$, is not graphic. Since $M \times S_i$ is a minor of $M$ and a minor of a graphic matroid is graphic (Theorem 2.8.6), $M$ cannot be graphic, hence the algorithm returns NULL in line 21, indicating that $M$ is
not graphic. If each $I_i$ is not NULL, then in lines 22 through 25, a matrix $I$ whose rows are all the rows of each $I_i$ altogether is constructed. Since each $I_i$ represents a graph whose bond matroid is a component of $M$, $M \times S_i$, it is clear that $I$ represents a graph $G$, where the components of $G$ are those graphs represented by each $I_i$, and the bond matroid of $G$ is $M$. Hence it is correct to return $I$ in line 27.

4.2 The Graphic Test Algorithm

The Graphic Test Algorithm (GTA) is described in Figure 3. It is basically a divide and conquer algorithm. We shall prove the correctness of GTA by induction.

Lines 8 through 13 describe the base case, in which each column of $R$ has no more than two 1s. A row $r$ which is the mod 2 sum of all the rows of $R$ is constructed in line 10, and a matrix $I$, whose rows are those rows of $R$ in addition to row $r$, is constructed in line 11. Clearly $I$ is an incidence matrix of a graph whose bond matroid is $M$. Hence $M$ is graphic, and it is correct to output $I$ in line 12.

Lines 15 through 18 describe the divide step. When the base condition is not satisfied, i.e., there exists a column of $R$ that has at least three 1s, the algorithm tries to break down the problem into smaller subproblems by running the Break Down Algorithm (BDA) using as input the matrix $R$ and a column index $j$, where the column $c_j$ of $R$ has at least three 1s (line 17).

Lines 20 through 33 describe the conquer step. We shall prove in Section 4.3 that if BDA returns NULL, then the matroid represented by its input matrix is not graphic. Hence if the output of BDA is NULL, it is correct for the algorithm to return NULL in line 21. We shall also prove in Section 4.3 that if the matrix $R$ is successfully broken down into a list $L_R$ of matrices, each matrix $R_i$ in $L_R$ represents a $Y$-component of $M$, $M \times (B_i \cup Y)$, where $Y$ is a circuit of $M$ represented by the set of column indices (also indicated by $Y$) returned by BDA, and $B_i$ is a bridge of $Y$ in $M$, represented by the column group $\kappa_i$ in the list $L_\kappa$ returned by BDA.
At line 28, for each matrix $R_i$ in $L_R$, GTA is run recursively with $R_i$ as its input, and outputs $I_i$. Assume that GTA runs correctly for each of its input $R_i$ (Induction Hypothesis). If any output $I_i$ is NULL, then the matroid $M \times (Bi \cup Y)$, represented by the matrix $R_i$ is not graphic. Since $M \times (Bi \cup Y)$ is a minor of $M$, and a minor of a graphic matroid is graphic (Theorem 2.8.6), the matroid $M$ represented by matrix $R$ cannot be graphic. Hence it is correct for the algorithm to return NULL in line 30. If each $I_i$ is a matrix, then $I_i$ is the incidence matrix of a graph whose bond matroid is the $Y$-component $M \times (Bi \cup Y)$ (Induction Hypothesis), and the algorithm continues to the combine step.

Lines 35 through 49 describe the combine step. In the combine step, first, with $L_R$ and $Y$ as its input, the Partition Algorithm (PA) is run to determine whether each bridge $B_i$ of $Y$, represented by the column group $\kappa_i$ in $L_\kappa$, partitions $Y$. We shall prove in Section 4.4 that if PA returns NULL, then there is a bridge $B_i$ that does not partition $Y$; otherwise, in the list $L_P$ returned by PA, each entry $P_i$ is the partition of $Y$ determined by the bridge $B_i$. Since if $M$ is regular, then each bridge of $Y$ in $M$ partitions $Y$ (Theorem 2.10.5). So if PA returns NULL, the matroid $M$ cannot be regular. And since every graphic matroid is regular, $M$ cannot be graphic if it is not regular. Hence it is correct for the algorithm to return NULL in line 38.

If each bridge $B_i$ of $Y$ in $M$ partitions $Y$, the list $L_P$ of partitions of $Y$ is obtained and returned by PA. And the algorithm continues to line 40, where the classification algorithm is run to determine whether $Y$ is an even circuit. We shall prove in Section 4.5 that if CA returns NULL, then $Y$ is not even; otherwise, $Y$ is even, and its input list $L_P$ of partitions of $Y$ is divided into two lists $\mathcal{L}_{P_1}$ and $\mathcal{L}_{P_2}$, according to the following criterion: for any two partitions $P_i$ and $P_j$ in the same list $\mathcal{L}_{P_k}$, there exists a partition element pair $\mathcal{P} = (I, J)$ such that $I \cup J = Y$, $I \in P_i$ and $J \in P_j$; and $\mathcal{P}$ will be added as the $j$th entry of $(i - 1)$th partition element pair list in the list $\mathcal{L}_{L_k}$ returned by CA. The other two input lists of CA, $L_I$ and $L_\kappa$, each of their $i$th entry corresponds to the $i$th entry in $L_P$, are divided into two lists accordingly.
Since in a graphic matroid every circuit is even (Theorem 2.10.6), if CA returns NULL, then \( Y \) is not even, and then the matroid \( M \) cannot be graphic. Hence it is correct for the algorithm to return NULL in line 42.

If the output of CA is not NULL, then the algorithm continues to line 43. Up to this point in the algorithm, \( Y \) is even and each \( Y \)-component of \( M \) is graphic. By Theorem 2.10.7, we can conclude that \( M \) is graphic. The Incidence Matrix Construction Algorithm (IMCA) is run in line 44. We shall prove in Section 4.6 that the incidence matrix returned by IMCA is the incidence matrix of a graph whose bond matroid is \( M \).

### 4.3 The Break Down Algorithm

The Break Down Algorithm is described in Figure 4.

Lines 12 through 30 are a while loop. The algorithm can enter the while loop at most three times, by the loop counter condition. Each time it enters the while loop, a distinct row \( r_a \) where \( R[a][j] = 1 \) is selected, and there constructs a matrix \( R' \), which is the same as matrix \( R \) except that the row \( r_a \) is replaced by a zero row and for each \( b \in Y = \{ b | R[a][b] = 1 \} \), the column \( c_b \) is replaced by a zero column. It is easy to see that if \( R \) represents the matroid \( M = (E, Q) \), then \( R' \) represents the matroid \( M \cdot (E - Y) \), where \( Y \) is a circuit of \( M \), represented by the set \( \{ b | R[a][b] = 1 \} \). As reasoned in Section 4.1, after the Row Grouping Algorithm (RGA) is run using \( R' \) as its input, in the list of column groups \( L_\kappa \) returned by RGA, each entry \( \kappa_i \) represents an elementary separator \( S_i \) of \( R' \), that is, a bridge \( B_i \) of \( Y \). And for each row group \( \rho_i \) in the list \( L_\rho \) returned by RGA, the matrix \( R'_i \), whose rows are those rows of \( R \) whose indices are in \( \rho_i \), represents the matroid \( [M \cdot (E - Y)] \times B_i = [M \times (B_i \cup Y)] \cdot B_i \) (by Theorem 2.8.7 3). Thus each matrix \( R_i \) in the list \( L_R \) returned by the algorithm, which is constructed in lines 35 through 41, and whose rows are those rows of \( R'_i \) in addition to the row \( r_a \) of \( R \), represents the matroid \( M \times (B_i \cup Y) \).

If there is only one entry in the list of column groups returned by RGA, then there is only one bridge of \( Y \) in \( M \) for that choice of \( Y \). If this is
the case each time the algorithm enters the while loop, then the algorithm will terminate after it has entered the while loop for three times, and return NULL. In this case, there are three distinct choices of $Y$, and for each choice of $Y$, $j \in Y$ and there is only one bridge of $Y$ in $M$. By Theorem 2.10.3, if $M$ is graphic, then each distinct $Y$ corresponds to the set of edges incident on a distinct vertex $v$, then the column $c_j$ corresponds to an edge that has three ends, which is impossible. Hence $M$ is not graphic if the algorithm returns NULL.

4.4 The Partition Algorithm

The Partition Algorithm is described in Figure 5. It is used in the Graphic Test Algorithm (GTA). The input to it in GTA is the list $L_R$ of matrices broken down in the previously run Break Down Algorithm (BDA), and the set $Y$ of column indices also returned by BDA. We know from the proof in Section 4.3 that each matrix $R_i$ in $L_R$ represents the matroid $M \times (B_i \cup Y)$, where $M$ is the matroid represented by the input matrix $R$ of GTA, $Y$ is a circuit of $M$ represented by the set $Y$, and $B_i$ is a bridge of $Y$ in $M$ represented by the column group $\kappa_i$ in the list $L_\kappa$ returned by BDA.

For each matrix $R_i$ in $R$, the algorithm first constructs a matrix $R'_i$, which is the same as $R_i$ except that each column whose index is not in $Y$ is replaced by a zero column (lines 7 through 11). Clearly, $R'_i$ represents the matroid $[M \times (B_i \cup Y)] \cdot Y$.

The algorithm then transforms each matrix $R'_i$ into a standard form through elementary row operations (line 12). We already know that elementary row operations do not alter the property of the matrix being a representative matrix of the same matroid. So each $R'_i$ still represents the matroid $[M \times (B_i \cup Y)] \cdot Y$. Now since each $R'_i$ is in standard form, each row of it represents an elementary chain. In other words, for each row $r_a$ in $R'_i$, the set $\{b | R'_i[a][b]\}$ represents a circuit of the matroid $[M \times (B_i \cup Y)] \cdot Y$.

For each matrix $R'_i$, the number of 1s in each column is counted (lines 13 through 14). If in matrix $R'_i$, each column has at most one 1, then the circuits
of the matroid \([M \times (B_i \cup Y)] \cdot Y\), each represented by the set \(\{b \mid R'_i[a][b]\}\) for a row \(r_a\) in \(R'_i\), are disjoint subsets of \(Y\). In this case, we say that \(B_i\) partitions \(Y\) and each partition element in the partition \(P_i\) determined by \(B_i\) is a circuit of the matroid \([M \times (B_i \cup Y)] \cdot Y\) (See section 2.10). It is quite clear that the set \(P_i\) in the list \(L_P\) returned by the algorithm, which is constructed in lines 19 through 26, is the partition determined by \(B_i\).

If there exists a column \(c_j\) in \(R'_i\) that has more than one 1, i.e., there exist two distinct rows \(r_a\) and \(r_b\) of \(R'_i\) such that \(R'_i[a][j] = R'_i[b][j]\), then the circuit \(A\) of \([M \times (B_i \cup Y)] \cdot Y\), represented by the set \(\{c \mid R'_i[a][c]\}\), and the circuit \(B\) of \([M \times (B_i \cup Y)] \cdot Y\), represented by the set \(\{c \mid R'_i[b][c]\}\), are not disjoint. In this case the algorithm returns NULL (line 16). Hence if the algorithm returns NULL, it indicates that there exists a bridge \(B_i\) that does not partition \(Y\).

### 4.5 The Classification Algorithm

The Classification Algorithm (CA) is described in Figure 6. It is used in the Graphic Test Algorithm (GTA). The input to it in GTA is the set \(Y\) of column indices, the list \(L_P\) of partitions of \(Y\), the list \(L_I\) of incidence matrices, and the list \(L_\kappa\) of column groups. The algorithm tries to put each entry in \(L_P\) into two disjoint lists \(L_{P_1}\) and \(L_{P_2}\), and each entry in each of the other corresponding lists into two disjoint lists accordingly. For any two partitions \(P_i\) and \(P_j\) to be put in the same list, the following condition (lines 23 through 24) must be satisfied: \(I \cup J = Y\), \(I \in P_i\) and \(J \in P_j\). This condition is also the condition for the two bridges \(B_i\) and \(B_j\) to not overlap, where \(B_i\) determines the partition \(P_i\) and \(B_j\) determines the partition \(P_j\) (See Section 2.10).

If all the partitions in the list \(L_P\) can be put into two lists, then all the column groups in the list \(L_\kappa\) can be put into two lists accordingly. Since each column group \(\kappa_i\) represents the bridge \(B_i\), which determines the partition \(P_i\), all the bridges can be arranged into two disjoint classes such that no two bridges in the same class overlap. Then in this case \(Y\) is an even circuit (See
Section 2.10).

If there exists a partition $P_i$ that can be put in neither of the two lists, then the algorithm returns NULL in line 42. In this case, the bridge $B_i$ that determines $P_i$ also cannot be put into either of the two classes accordingly, hence $Y$ is not even.

### 4.6 Incidence Matrix Construction

The Incidence Matrix Construction Algorithm (IMCA) is described in Figure 7. It is used in the Graphic Test Algorithm (GTA). The input to it in GTA is the output of the previously run Classification Algorithm, the lists $L_{P_1}, L_{P_2}, L_{I_1}, L_{I_2}, L_{\kappa_1}, L_{\kappa_2}, L_{L_1}$, and $L_{L_2}$, and the set $Y$ of column indices, which represents a circuit $Y$ of the matroid $M$ represented by GTA’s input matrix $R$. Up to the point where the algorithm is run in GTA, $M$ has been determined graphic (See Section 4.2), and the task of IMCA is to construct and return an incidence matrix $X$ of a graph whose bond matroid is $M$. In addition, up to this point, $Y$ has been determined to be an even circuit, and the list $L_P$ has been divided into two lists, $L_{P_1}$ and $L_{P_2}$ according to the following criterion: for any two partitions $P_i$ and $P_j$ in the same list, there exists a partition element pair $(I, J)$ such that $I \cup J = Y$, $I \in P_i$ and $J \in P_j$. Each of the other two lists $L_I$ and $L_\kappa$, is divided into two lists, $L_{I_1}$ and $L_{I_2}$, $L_{\kappa_1}$ and $L_{\kappa_2}$ accordingly. Hence any two column groups in the same list represent two bridges that do not overlap.

To prove that the matrix $X$ constructed in the algorithm represents a graph whose bond matroid is $M$, let’s look at the construction process in Figure 7 line by line.

Lines 8 through 44 are a for loop, the algorithm enters the for loop twice, each time working on a list $L_{I_i}$ to construct a matrix $X_i$.

Lines 9 through 43 is another for loop inside the above for loop. The algorithm enters the for loop each time for a matrix $I_j$ in the list $L_{I_i}$.

Inside the second for loop, lines 10 through 12 look for a common row/vertex $r_a$ whose incident edges constitute the set $Y$, and if the row $r_a$ is not the last
row, then swap it to the last row. In the proof below, we show that the common row $r_a$ can be found in each incidence matrix $I_j$ in a $\mathcal{L}_{I_i}$:

**Proof:** Each incidence matrix $I_j$ in a $\mathcal{L}_{I_i}$ represents a graph whose bond matroid is a $Y$-component of $M$. Since in any $Y$-component, $Y$ has only one bridge, by Theorem 2.10.3, in any graph whose bond matroid is the $Y$-component, there exists a vertex $v$ where the set of edges incident on $v$ is $Y$. Hence each incidence matrix $I_j$ in a list $\mathcal{L}_{I_i}$ must have a common row $r$, which represents the vertex $v$ whose incident edge set is $Y$.

If $I_j$ is the first entry in the list $\mathcal{L}_{I_i}$, then it becomes $X_i$ in lines 13 through 14. Otherwise, in lines 15 through 42, a new matrix $X_i$ is constructed from the matrix $X_i$ that has been constructed so far and the matrix $I_j$.

In lines 16 through 19, there constructed a new matrix $X$ with $m_i + m_j - 2$ rows, whose first $m_i - 1$ rows are those rows of $X_i$ except the last row, and last $m_j - 1$ rows are those rows of $I_j$ except the last row. We already know that $I_j$ is an incidence matrix, and the last row in $I_j$ is the common row $r$. We will show later in this section that $X_i$ is also an incidence matrix, and the last row in $X_i$ is also the common row $r$. Hence the newly constructed matrix $X$ is the incidence matrix of a graph, in which the first $m_i - 1$ rows represent the vertices in an end graph $H_1$ of $Y$, and the last $m_j - 1$ rows represent the vertices in the other end graph $H_2$ of $Y$.

Let’s denote $G$ to be the graph represented by $X$, $G_1$ to be the graph represented by $X_1$, and $G_2$ to be graph represented by $I_j$. The bond matroid of $G_1$ is a $Y$-component of $M$, $M \times (B_j \cup Y)$, where $B_j$ is a bridge of $Y$ in $M$ represented by the $j$th column group in the list $\mathcal{L}_{\kappa_i}$. We can see clearly that $G \times (B_j \cup Y) \cong G_2$.

In line 40 the Switch Algorithm is run to transform $X$ into an incidence matrix of a graph $G'$ whose bond matroid $B(G')$ is the same as the bond matroid $B(G)$ of $G$, and the last row in the matrix is the common row $r$. We will prove this later in this section. In line 41, the matrix transformed from $X$ in the Switch Algorithm becomes $X_i$. We can prove by induction that the $X_i$ at the end of the for loop that ends at line 43 has the common row $r$ as
its last row, and is the incidence matrix of a graph whose bond matroid is
\[ M \times (B_1 \cup B_2 \ldots \cup B_j \cup Y), \]
where \( B_1, B_2, \ldots B_j \) are the bridges of \( Y \) in \( M \) represented by the 1, 2, \ldots, \( j \)th column groups in the list \( \mathcal{L}_{\kappa_i} \):

\textit{proof:}

\textit{Base case:} (where \( j = 1 \))

\( X_i \) is \( I_1 \) that is assigned to it at line 14, and the last row is the common row by lines 10 through 12. And \( X_i \) is the incidence matrix of a graph whose bond matroid is the \( Y \)-component of \( M, M \times (B_1 \cup Y) \).

\textit{Inductive step:}

In the beginning of the for loop that ends at line 43, \( X_i \) is the incidence matrix of the graph \( G_1 \) whose bond matroid is
\[ M \times (B_1 \cup B_2 \ldots \cup B_{j-1} \cup Y), \]
by the Induction Hypothesis. Clearly the graph \( G_1 \cong G \times (B_1 \cup B_2 \ldots \cup B_{j-1} \cup Y) \), and the graph \( G_2 \cong G \times (B_j \cup Y) \). So the bond matroid \( B(G) \) of graph \( G \) is
\[ B(G_1) \cup B(G_2) = [M \times (B_1 \cup B_2 \ldots \cup B_{j-1} \cup Y)] \cup [M \times (B_j \cup Y)] \]
\[ = M \times (B_1 \cup B_2 \ldots \cup B_j \cup Y) \]
We will see later in this section, that the Switch Algorithm transforms \( X \) into another incidence matrix \( X' \), which has the common row as its last row and represents a graph \( G' \) whose bond matroid \( B(G') \) is the same as the bond matroid of \( G \). As \( X' \) is assigned to \( X_i \) at the end of the for loop in line 41, the \( X_i \) at the end of the for loop that ends at line 43 is the incidence matrix of a graph whose bond matroid is
\[ M \times (B_1 \cup B_2 \ldots \cup B_j \cup Y). \]

As reasoned above, at the end of the for loop that starts at line 8 and ends at line 44, \( X_1 \) represents a graph whose bond matroid is \( M \times (A \cup Y), \)
where $A$ is the union of all the bridges represented by the column groups in $L_{\lambda_1}$; $X_2$ represents a graph whose bond matroid is $M \times (B \cup Y)$, where $B$ is the union of all the bridges represented by the column groups in $L_{\lambda_2}$. And the last row in both $X_1$ and $X_2$ is the common row $r$. Hence the matrix $X$ constructed in lines 45 through 48 represents the graph whose bond matroid is 

$$[M \times (A \cup Y)] \cup [M \times (B \cup Y)] = M \times (A \cup B \cup Y) = M.$$ 

Now we prove that the Switch Algorithm can transform the graph $G$, represented by the matrix $X$ in the for loop that starts at line 9 and ends at line 43, into a graph $G'$ whose bond matroid $B(G')$ is the same as the bond matroid $B(G)$ of $G$, and one end graph of $Y$ is a single vertex. We prove this by the following theorem:

**Theorem 4.6.1** ([5], 8.4)

Let $G$ be a nonseparable graph. Let $Y$ be a bond of $G$. Let $B_1, \ldots, B_n$ be the bridges of $Y$ in $B(G)$. If $B_1, \ldots, B_n$ are pairwise nonoverlapping, and neither end-graph of $Y$ is a single vertex, then $G$ can be transformed into a graph $G'$ such that the bond matroid of $G$, $B(G) = B(G')$ and one end-graph of $Y$ is a single vertex.

**Proof:**

Let $H_1, H_2$ be the two end-graphs of $Y$ in $G$. It is clear that each $G \cdot B_i$ is a separate of $H_1$ or $H_2$. Let $H_2$ denote the end-graph that has fewer edges. Let $v$ be a vertex of $G \cdot B_i$, we denote by $C(B_i, v)$ the component of $H_1 : [E(H_1) - B_i]$ or $H_2 : [E(H_2) - B_i]$ containing vertex $v$. We denote by $Y(B_i, v)$ the set of edges incident on the vertices of $C(B_i, v)$, which is the same set of edges incident on $v$ in $[G \times (B_i \cup Y)] \cup Y$. For a bridge $B_i, Y(B_i, v)$ is an element of the partition $P_i$ determined by $B_i$ (see section 2.10). Since $B_1, \ldots, B_n$ are pairwise nonoverlapping, for any two bridges $B_i, B_j$, there exist a vertex $v_i$ in $G \cdot B_i$ and a vertex $v_j$ in $G \cdot B_j$ such that $Y(B_i, v_i) \cup Y(B_j, v_j) = Y$.

Let $G \cdot B_i$ be a separate of $H_1$. Let $G \cdot B_2$ be a separate of $H_2$. Then there exist a vertex $v_i$ in $G \cdot B_i$ and a vertex $v_2$ in $G \cdot B_2$ such that $Y(B_i, v_i) \cup Y(B_2, v_2) = Y$. Find all the pairs $(B_i, v_i)$ in $H_1$ that satisfy the relation
Y(B_i, v_i) \cup Y(B_2, v_2) = Y$. Let \((B_1, v_1)\) be the pair for which \(C(B_1, v_1)\) has the least possible number of edges.

Let \(A_1, \ldots, A_k, \) where \(A_1 = B_1, \) be the sets of edges of the separates of \(H_1\) having \(v_1\) as a vertex. For each \(A_i, \) let \(E_i\) be the subgraph of \(H_1, \) which is the union of \(G \cdot A_i\) and those subgraphs \(C(A_i, v)\) of \(H_1\) such that \(v\) is a vertex of \(G \cdot A_i\) other than \(v_1.\) Then the graphs \(E_i\) have only one common vertex \(v_1.\) And since \(H_1\) is connected, it is the union of the graphs \(E_i.\)

For each \(A_i, \) we have the relation: \(Y(A_i, p_i) \cup Y(B_2, q_i) = Y.\) Since \(G\) is nonseparable, for each such relation, there must be an edge \(a_i\) such that \(a_i \in Y(B_2, q_i)\) but \(a_i \not\in Y(A_i, p_i)\) and an edge \(b_i\) such that \(b_i \in Y(A_i, p_i)\) but \(b_i \not\in Y(B_2, q_i).\)

For \(A_1 = B_1, \) we have \(p_1 = v_1, q_1 = v_2,\) and an edge \(a_1\) with one end being \(v_2\) and the other end being a vertex of \(E_1\) other than \(v_1.\) For each \(A_i\) other than \(A_1,\) suppose \(p_i \neq v_1,\) then since \(a_1\) cannot be an element of \(Y(A_i, p_i),\) \(a_1\) must be an element of \(Y(B_2, q_i),\) hence \(q_i\) must be \(v_2.\) But then \(C(A_i, p_i)\) has fewer edges than \(C(B_1, v_1),\) which contradicts the definition of \(B_1, v_1.\) Hence in each relation \(Y(A_i, p_i) \cup Y(B_2, q_i) = Y, p_i = v_1.\)

Considering the edge \(a_i,\) we see that \(q_i\) is uniquely determined for each \(A_i.\) Let \(Z_i\) denote the set of all members of \(Y\) having one end a vertex of \(E_i\) other than \(v_1.\) Then \(Z_i\) is non-null since it includes \(a_i.\) By the relation \(Y(A_i, p_i) \cup Y(B_2, q_i) = Y, Z_i\) must be a subset of \(Y(B_2, q_i),\) which means each member of \(Z_i\) has its other end a vertex of \(C(B_2, q_i).\)

For each vertex \(v,\) of \(G \cdot B_2,\) we denote by \(R(v)\) the subgraph of \(G\) which is the union of \(C(B_2, v),\) the graph \(E_i\) for which \(q_i = v,\) and the members of the corresponding \(Z_i.\) For a given vertex \(v\) the graph \(R(v)\) may have only one vertex. If not the set \(E(R(v))\) is non-null and its vertices of attachment in \(G\) are \(v\) and \(v_1.\) Since \(G\) is nonseparable, there are at least two such vertices of attachment, and \(v\) and \(v_1\) are the only possibilities. Moreover if \(x\) and \(y\) are distinct vertices of \(G \cdot B_2\) then \(R(x)\) and \(R(y)\) have at most one common vertex \(v_1\) and no common edge. We may therefore independently switch \(R(x)\) through the vertices \(v_1, x\) for each non-null \(E(R(x)).\) In the switch process, each edge that is incident on \(v_1\) is changed to be incident on \(x,\) and each edge
that is incident on $x$ is changed to be incident on $v_1$; the rest of the graph remains unchanged. Since the bonds of the graph are still the same after the switch process, the bond matroid of the graph remains the same. After each $R(x)$, for which $E(R(x))$ is non-null, has been switched, $G$ is transformed into another graph $G'$ that has the same bond matroid as $B(G)$, but for the end-graphs $H'_1$ and $H'_2$, $E(H'_1) = E(H_2) - B_2$, and $E(H'_2) = E(H_1) + B_2$. If $|E(H'_1)| \neq 0$, by a transformation similar to the transformation of $G$ to $G'$, or a series of these transformations, we eventually will transform $G$ into a new graph which has the same bond matroid but one end-graph of $Y$ in the new graph is a single vertex, that is, has zero edges.

Inside the for loop that starts at line 9 and ends at line 42, the graph $G$ represented by $X$ is nonseparable, since $M$ is a connected matroid. It is clear that the bridges of $Y$ in $B(G)$ are $B_1, B_2, \ldots, B_j$, which are represented by the column groups $\kappa_1, \kappa_2, \ldots, \kappa_j$ in the list $L_\kappa$. Since these column groups are in the same list, no two of these bridges overlap. And since $X$ does not include the common row $r$, the graph $G$ does not have the common vertex.

As the theorem states, the graph $G$ must be able to be transformed into a graph $G'$ whose bond matroid is the same as the bond matroid of $G$, and one end graph of $Y$ in $G'$ is a single vertex. The end-graph $H_2$ of $Y$ in $G$ is $G \cdot B_j$, which is nonseparable. From the above proof, we can see that after one round of transformation, $G$ will be turned into graph $G'$ in which the end-graph $H'_1$ of $Y$ has $|E(H'_2)| - |B_j| = 0$ number of edges, which means that the end-graph $H'_1$ of $Y$ in $G'$ is a single vertex. As reasoned in the proof, in the same round of transformation, each subgraph that is switched is attached to the rest of the graph through a common vertex $v_1$ in $H_1$ and another vertex $v$ in $H_2$. And each switch occurs independently without interfering with others. After one round of switch, $v_1$ becomes the single vertex in the end-graph $H'_1$ of $Y$ in $G'$. Hence once $v_1$ is identified, it will be the target for the common row, and the switch will be straightforward: for each edge $e \in Y$ that is not incident on $v_1$, identify its two ends, $v_2$ and $v_3$. Suppose $v_2$ is in $H_2$, then obtain the subgraph that contains $v_3$ and is attached to the rest of the graph through $v_1$ and $v_2$, and make a switch of the subgraph. It
5 RUNNING TIME ANALYSIS OF THE ALGORITHM

is quite clear that the Switch Algorithm described in Figure 8 accomplishes the switches in terms of the incidence matrix. After the transformation in the Switch Algorithm, the formed common row \( r \) is assured to be the last row in lines 34 through 35 (Figure 8).

The process of identifying the target row/vertex \( r_t \) for the common row/vertex \( r \) is described in lines 20 through 39. Since each partition element in the partition \( P_j \) is the set of edges incident on a vertex in the graph \( [G \times (B_j \cup Y)] \cdot Y \), the partition element \( I \) in line 20 uniquely identifies a row/vertex \( r_c \) in the last \( m_j - 1 \) rows, where \( I \subseteq \{ b \mid X[c][b] = 1 \} \) and \( I = Y(B_j, r_c) \).

In lines 21 through 25, the bridges \( B_k \) that satisfy the relationship \( Y(B_k, v_k) \cup Y(B_j, r_c) = Y \) are searched and the rows/vertices of \( G \cdot B_k \) are marked.

Lines 26 through 39 search for the target row/vertex \( r_t \). In lines 26 through 28, a row/vertex \( r_a \) in \( H_1 \) whose incident edges include an edge \( b \) in \( Y - I \) is identified. \( r_a \) must be a vertex in a subgraph \( C(B_k, v_k) \), where \( Y(B_k, v_k) \cup Y(B_j, r_c) = Y \), since the column/edge \( c_b \in Y(B_k, v_k) \). If row/vertex \( r_a \) is marked, then it must be target vertex \( r_t \), since among all the subgraphs \( C(B_k, v_k) \) that satisfy \( Y(B_k, v_k) \cup Y(B_j, r_c) = Y \), the one that has \( v_k = r_a \) must have the least number of edges. Otherwise, since any two separates of \( H_1 \) have at most one vertex in common, breath-first search is used to find the marked row/vertex \( r_t \) that has the shortest path to row/vertex \( r_a \), and the subgraph \( C(B_k, r_t) \) is a subgraph of all those subgraphs \( C(B_k, v_k) \) and hence has the least number of edges. And the row/vertex \( r_t \) will be the target row/vertex.

5 Running Time Analysis of the Algorithm

In this section the running time for the various sub-algorithms is analyzed first, then the running time analysis of the Graphic Test Algorithm is performed, and finally we analyze the running time for the Main Algorithm.

Given as input a binary matrix \( R \) with \( m \) rows and \( n \) columns, it is easy to see that the Row Grouping Algorithm takes \( O(mn) \) time and the
transformation to standard form takes $O(m^2n)$ time.

5.1 The Break Down Algorithm

Given as input a binary matrix $R$ with $m$ rows and $n$ columns, the running time for the Break Down Algorithm is analyzed as follows.

Lines 6 through 11 take $O(1)$ time. Lines 12 through 30 are a while loop, the algorithm iterates this loop at most three times. Inside the while loop, lines 13 through 14 take $O(1)$ time, lines 15 through 16 take $O(m)$ time, lines 18 through 21 take $O(n)$ time, lines 22 through 24 take $O(mn)$ time, line 25 takes $O(mn)$ time, and lines 26 through 29 take $O(1)$ time. Hence the whole while loop takes $O(mn)$ time.

Lines 31 through 32 take $O(1)$ time. Lines 33 through 42 take $O(mn)$ time. Hence the running time for the Break Down Algorithm is $O(mn)$.

5.2 The Partition Algorithm

The Partition Algorithm is called in the Graphic Test Algorithm (GTA) one time. Given that the input to GTA is a binary matrix $R$ with $m$ rows and $n$ columns, the running time of the Partition Algorithm is analyzed as follows.

Lines 6 through 28 are a for loop. The algorithm iterates the for loop $s$ times, where $s$ is the number of matrices in the input list $L_R$. Inside the for loop, lines 7 through 11 take $O(m_i n)$ time, line 12 takes $O(m_i^2 n)$ time, lines 13 through 17 take $O(m_i n)$ time, and lines 19 through 26 take $O(m_i n)$ time, where $m_i$ is the number of rows in $R_i'$. So the whole for loop takes time:

$$
\sum_{i=1}^{s} O(m_i^2 n) = O((\sum_{i=1}^{s} m_i^2) n) = O((\sum_{i=1}^{s} m_i) n)
$$

$$
= O((m + s - 1)^2 n) = O(m^2 n),
$$

since $s = O(m)$. Thus the running time for the algorithm is $O(m^2 n)$.  

5.3 The Classification Algorithm

The Classification Algorithm is called in the Graphic Test Algorithm (GTA) one time. Given that the input to GTA is a binary matrix $R$ with $m$ rows and $n$ columns, the running time of the Classification Algorithm is analyzed as follows.

Lines 7 through 11 take $O(1)$ time. Lines 12 through 46 are a for loop. The algorithm iterates the for loop $s$ times, where $s$ is the number of partitions in the input list $L_P$. Inside the for loop, lines 13 through 21 take $O(1)$ time, and lines 22 through 44 are a while loop. The iteration of the while loop is at most $s$ times. Inside the while loop, lines 23 through 31 take $O(n^2)$ time, and lines 33 through 43 take $O(1)$ time. Hence the while loop takes $O(sn^2) = O(mn^2)$ time. Also, the for loop takes $O(smn^2) = O(m^2n^2)$ time. So the running time of the algorithm is $O(m^2n^2)$

5.4 The Switch Algorithm

The Switch Algorithm (SA) is called in the Incidence Matrix Construction Algorithm (IMCA). Given that the input matrix of SA is an $m$ by $n$ matrix, the running time of SA is analyzed as follows.

Line 4 takes $O(n)$ time. Lines 5 through 33 are a while loop. The algorithm iterates this while loop $O(n)$ times. Inside the while loop, lines 6 through 8 take $O(m)$ time, lines 9 through 11 take $O(mn)$ time, lines 13 through 19 take $O(mn)$ time, lines 20 through 31 take $O(mn)$ time, and line 32 takes $O(n)$ time. Hence the while loop takes $O(mn^2)$ time. Line 34 through 35 take $O(n)$ time. So the running time of SA is $O(mn^2)$.

5.5 The Incidence Matrix Construction Algorithm

The Incidence Matrix Construction Algorithm (IMCA) is called in the Graphic Test Algorithm (GTA) one time. Given that the input to GTA is a binary matrix $R$ with $m$ rows and $n$ columns, the running time of IMCA is analyzed as follows.
5 RUNNING TIME ANALYSIS OF THE ALGORITHM

Lines 8 through 44 are a for loop. The algorithm iterates this for loop two times. Lines 9 through 43 are a for loop immediately inside the other for loop. The algorithm iterates this for loop \(s_i\) time, where \(i = 1\) or \(2\), and \(s_1 + s_2 = s\). Inside this for loop, lines 10 through 14 take \(O(mn)\) time, lines 16 through 19 take \(O(mn)\) time, lines 20 through 25 take \(O(mn)\) time, lines 26 through 30 take \(O(m + n)\) time, and lines 31 through 39 take \(O(mn)\) time. Line 40 takes \(O(mn^2)\) time. So the for loop (Lines 8 through 44) takes \(O(sm^2n) = O(m^2n^2)\) time.

Lines 45 through 48 take \(O(mn)\) time. Thus the running time of IMCA is \(O(m^2n^2)\).

5.6 The Graphic Test Algorithm

The Graphic Test Algorithm (GTA) is called in the Main Algorithm. Given as input a binary matrix with \(m\) rows and \(n\) columns, the running time \(T_{\text{GTA}}(m, n)\) of GTA is analyzed as follows.

Lines 4 through 6 take \(O(mn)\) time. Lines 9 through 12 take \(O(mn)\) time. Line 17 takes \(O(mn)\) time. Lines 22 through 33 take \(\sum_{i=1}^{s} T_{\text{GTA}}(m_i, n)\) time, where \(s\) is the number of matrices in \(L_R\), and \(m_i\) is the number rows in matrix \(R_i\), since GTA is called recursively with input \(R_i, i = 1, 2, \ldots, s\).

As reasoned in Section 5.2, line 36 takes \(O(m^2n)\) time. As reasoned in Section 5.3, line 40 takes \(O(m^2n^2)\) time. Lines 44 through 46 also take \(O(m^2n^2)\) time, as reasoned in Section 5.5. Thus the total running time of the algorithm is

\[
T_{\text{GTA}}(m, n) = \sum_{i=1}^{s} T_{\text{GTA}}(m_i, n) + O(m^2n^2),
\]

We can prove by induction that \(T_{\text{GTA}}(m, n) = O(m^2n^2)\).

5.7 The Main Algorithm

Given as input a binary matrix with \(m\) rows and \(n\) columns, the running time \(T(m, n)\) of the Main Algorithm is analyzed as follows.
Lines 4 through 6 take $O(mn)$ time. Lines 7 through 9 take $O(m^2n)$ time. Line 10 takes $O(mn)$ time.

Lines 13 through 26 are a for loop. The algorithm iterates this for loop $s$ times, where $s$ is the number of entries in $L_{\rho}$. Inside the for loop, lines 14 through 17 take $O(m_i n)$ time, where $m_i$ is the number of rows in matrix $R_i$, line 18 takes $T_{GT A}(m_i, n) = O(m_i^2 n^2)$ time, and lines 19 through 25 take $O(m_i n)$ time. Hence the whole for loop takes

$$\sum_{i=1}^{s} [T_{GT A}(m_i, n) + O(m_i n)] = \sum_{i=1}^{s} [O(m_i^2 n^2) + O(m_i n)]$$

$$= \sum_{i=1}^{s} O(m_i^2 n^2) = O(m^2 n^2)$$

time. Thus the running time for the Main Algorithm is $O(m^2 n^2)$.

6 The Java Program

The main program called InputApplet, is an applet. When the program is run, a main window shows up. This window contains two input lines, which take the input of the number of rows and the number of columns of the input binary matrix, and one text area, which takes the input of the binary matrix. It also contains three buttons: the “Compute” button, the “Show reason” button, and the “Show graph” button. The “Show reason” button and the “Show graph” button are disabled before the first click of the “Compute” button. The user can click the “Compute” button to determine whether the matroid represented by the input binary matrix is graphic.

If the program determines that the input is graphic, both the “Show reason” and the “Show graph” buttons will be enabled, and at the bottom of the window, there will be a line of text indicating that the matroid is graphic. Otherwise, the “Show graph” button will be disabled, and at the bottom of the window there will be a line of text indicating that the matroid is not graphic.

When the “Show reason” button is pressed, another window which contains the reasoning of the graphicness for the matroid will pop up. When
the input matroid is graphic, the “Show graph” button is enabled. Clicking on the “Show graph” button will bring up a third window which contains a graph whose bond matroid is the matroid represented by the input matrix. This window contains a Layout menu. The user can select a layout from three choices: Polygon layout, Ring layout, and Random layout. The user can also change the layout of the graph by simply dragging a vertex to another place using the mouse.

If the user presses “Enter” on the keyboard while the cursor is blinking in the second input line, a binary matrix in standard form will appear in the text area. This matrix has the number of rows and number of columns as indicated in the two input lines, and has block of 1s in each column that follows unit matrix part, which was randomly generated. We know that matroids represented by the matrices generated this way are graphic. This is used for testing the correctness of the program.

If the applet is run as an application, then the “File” menu could be used to open a file containing a binary matrix. The input file must be in the following format: The first line contains the number of rows and number of columns of the matrix separated by a space. The rest of the file consists of the actual binary matrix. To input a binary matrix from a file, click on the “File” menu, select “Open”, this will bring up a dialog box to browse the directory and select the desired file. After the user opens the file successfully, the number of rows, number of columns, and the binary matrix will appear in the indicated place. The rest of the steps is the same.

6.1 Test of The Program

As stated earlier in in this section, the main program contains a “File” menu. If the program is run as an application, one can open a file that contains a binary matrix, and test if that matroid is graphic. We have stored many files that contain known graphic and non-graphic binary matroids in the directory sampleInputs. The program works correctly for all the files in that directory. In addition, one can press “Enter” on the keyboard after entering the number
of rows and number of columns, while the cursor is blinking in the second input line. A binary matrix in standard form with the indicated number of rows and number of columns will be generated and appears in the text area. It has block of 1s in each column that follows the unit matrix part, which was randomly generated, which guarantees that the matroid is graphic. Clicking on the “Compute” button will tell whether the matroid generated is indeed graphic. We have tested this more than 50 times, with different number of rows and number of columns. The program works correctly in every test we have run so far. Below are some sample runs of the program.

6.1.1 Example 1

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Given as input the above matrix, the program determines that the matroid represented by the input matrix is not graphic. The step-by-step reasoning, which is contained in the window that shows up when the “Show reason” button is pressed, is shown below:

Determine whether the matroid $M$ represented by the following matrix is graphic:

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Group the rows into row groups.

For two rows $a$ and $b$ in the same row group, there must be a column $c$ such that $c$ has a 1 in both row $a$ and row $b$.

The set of columns with 1s in the rows of the same row group is an elementary separator of the matroid represented by the matrix.

The elementary separators of the matroid represented by the matrix are: $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

The row groups are:

$\{1, 2, 3, 4\}$

For each elementary separator $S_i$,

adjoin the rows in the corresponding row group together.

The formed matrix is a standard representative matrix of the matroid $M \times S_i$, which is a connected component of $M$.

These components of $M$ are:

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Determine whether these components are graphic.

The matroid $M$ is graphic if and only if all these components are graphic.
Determine whether the component 1 is graphic:

\[
\begin{matrix}
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
\end{matrix}
\]

The matrix has at least one column with at least three 1s in it.
Select the first column that has at least three 1s.
Select the first row that has a 1 in the selected column.

\(Y\) (the set of columns with a 1 in this row) = \{1, 4, 5, 6\}
Replace row 1 and all the columns with a 1 in this row with 0s,
the resulted matrix \(R'\) is:

\[
\begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
\end{matrix}
\]

Get the bridges of \(Y\), which are the elementary separators
of the matroid represented by matrix \(R'\)
Group the rows into row groups.
For two rows \(a\) and \(b\) in the same row group,
there must be a column \(c\) such that \(c\) has a 1 in both row \(a\) and row \(b\).
The set of columns with 1s in the rows of the same row group
is an elementary separator of the matroid represented by the matrix.
The elementary separators of the matroid represented by the matrix are:
\{2, 3, 7, 8, 9\}
The row groups are:
\{2, 3, 4\}

Since \(Y\) has only one bridge, repeat the procedure for row 2.
\(Y\) (the set of columns with a 1 in this row) = \{1, 2, 5, 7\}
Replace row 2 and all the columns with a 1 in this row with 0s,
the resulted matrix \(R'\) is:

\[
\begin{matrix}
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
\end{matrix}
\]

Get the bridges of \(Y\), which are the elementary separators
of the matroid represented by matrix \(R'\)
Group the rows into row groups.
For two rows \(a\) and \(b\) in the same row group,
there must be a column \(c\) such that \(c\) has a 1 in both row \(a\) and row \(b\).
The set of columns with 1s in the rows of the same row group
is an elementary separator of the matroid represented by the matrix.
The elementary separators of the matroid represented by the matrix are:
\{3, 4, 6, 8, 9\}
The groups of rows are:
\{1, 3, 4\}

Since \(Y\) has only one bridge, repeat the procedure for row 3.
\(Y\) (the set of columns with a 1 in this row) = \{2, 3, 5, 8\}
Replace row 3 and all the columns with a 1 in this row with 0s,
the resulted matrix \(R'\) is:

\[
\begin{matrix}
1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\end{matrix}
\]

Get the bridges of \(Y\), which are the elementary separators
of the matroid represented by matrix \(R'\)
Group the rows into row groups.
For two rows \(a\) and \(b\) in the same row group, there must be a column \(c\) such that \(c\) has a 1 in both row \(a\) and row \(b\).

The set of columns with 1s in the rows of the same row group is an elementary separator of the matroid represented by the matrix.

The elementary separators of the matroid represented by the matrix are:
\[\{1, 4, 6, 7, 9\}\]

The row groups are:
\[\{1, 2, 4\}\]

Since there is only one bridge for all 3 choices of \(Y\),
We conclude that this matroid is not graphic.
Since every minor of a graphic matroid is graphic, the matroid \(M\) is not graphic.

6.1.2 Example 2

\[
\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
\end{array}
\]

Given as input the above matrix, the program determines that the matroid represented by the input matrix is not graphic. The step-by-step reasoning, which is contained in the window that shows up when the “Show reason” button is pressed, is shown below:

Determine whether the matroid \(M\) represented by the following matrix is graphic:
\[
\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
\end{array}
\]

Group the rows into row groups.
For two rows \(a\) and \(b\) in the same row group, there must be a column \(c\) such that \(c\) has a 1 in both row \(a\) and row \(b\).

The set of columns with 1s in the rows of the same row group is an elementary separator of the matroid represented by the matrix.

The elementary separators of the matroid represented by the matrix are:
\[\{1, 2, 3, 4, 5, 6, 7, 8\}\]

The groups of rows are:
\[\{1, 2, 3, 4\}\]

For each elementary separator \(S_i\),

adjoin the rows in the corresponding row group together.

The formed matrix is a standard representative matrix of the matroid \(M \times S_i\), which is a connected component of \(M\).

These components of \(M\) are:
\[
\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
\end{array}
\]

Determine whether these components are graphic.
The matroid \(M\) is graphic if and only if all these components are graphic.

Determine whether the component 1 is graphic:
\[
\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
\end{array}
\]
The matrix has at least one column with at least three 1s in it. 
Select the first column that has at least three 1s. 
Select the first row that has a 1 in the selected column. 
selected column = 5, selected row = 1. 
Y (the set of columns with a 1 in this row) = {1, 5, 6, 7} 
Replace row 1 and all the columns with a 1 in this row with 0s, 
the resulted matrix R' is: 
0 0 0 0 0 0 0 0 
0 0 0 1 0 0 0 0 
0 0 1 0 0 0 0 1 
0 0 0 0 1 0 0 1 
Get the bridges of Y, which are the elementary separators 
of the matroid represented by matrix R' 
Group the rows into row groups. 
For two rows a and b in the same row group, 
there must be a column c such that c has a 1 in both row a and row b. 
The set of columns with 1s in the rows of the same row group 
is an elementary separator of the matroid represented by the matrix. 
The elementary separators of the matroid represented by the matrix are: 
{2} {3, 4, 8} 
The groups of rows are: 
{2} {3, 4} 
From the following matrix: 
1 0 0 0 1 1 1 0 
0 1 0 0 1 1 0 0 
0 0 1 0 1 0 1 1 
0 0 0 1 0 1 1 1 
Adjoin all the rows in the same row group together, and add row 1 as the last row. 
The formed matrix represents a Y component of the matroid represented by the matrix. 
The following matrices are formed: 
0 1 0 0 1 1 0 0 
1 0 0 0 1 1 1 0 
0 0 1 0 1 0 1 1 
0 0 0 1 0 0 1 1 
1 0 0 0 1 1 1 0 
For each Y component, delete the columns with a 0 in the last row. 
Reduce the formed matrix to standard form(within a permutation of columns). 
Check each column. If there is a column with more than one 1 in it, 
then the corresponding bridge does not partition Y. 
Then since in a graphic matroid, every bridge partitions Y, 
the matroid is not graphic. 
Otherwise, the corresponding bridge determines a partition. 
And each element of the partition 
is the set of columns with a 1 in the same row in the matrix. 
The reduction to Y of Y component 1 is: 
0 1 1 0 
1 1 1 1 
Reduce it to standard form: 
0 1 1 0 
1 0 0 1 
Bridge 1 determines partition {{5, 6} {1, 7}}
The reduction to $Y$ of $Y$ component 2 is:
\[
\begin{bmatrix}
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1
\end{bmatrix}
\]
Reduce it to standard form:
\[
\begin{bmatrix}
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1
\end{bmatrix}
\]
Bridge 2 has more than one 1 in column 4, hence it does not partition $Y$ and the matroid is not graphic.
The component 1 of the matroid $M$ is not graphic.
Since every minor of a graphic matroid is graphic, the matroid $M$ is not graphic.

### 6.1.3 Example 3

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 1
\end{bmatrix}
\]

Given as input the above matrix, the program determines that the matroid represented by the input matrix is not graphic. The step-by-step reasoning, which is contained in the window that shows up when the “Show reason” button is pressed, is shown below:

Determine whether the matroid $M$ represented by the following matrix is graphic:
\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}
\]

Group the rows into row groups.
For two rows $a$ and $b$ in the same row group, there must be a column $c$ such that $c$ has a 1 in both row $a$ and row $b$.
The set of columns with 1s in the rows of the same row group is an elementary separator of the matroid represented by the matrix.
The elementary separators of the matroid represented by the matrix are:
\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}
The groups of rows are:
\{1, 2, 3, 4, 5, 6\}
For each elementary separator $S_i$, adjoin the rows in the corresponding row group together.
The formed matrix is a standard representative matrix of the matroid $M \times S_i$, which is a connected component of $M$.
These components of $M$ are:
\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}
\]

Determine whether these components are graphic.
The matroid $M$ is graphic if and only if all these components are graphic.

Determine whether the component 1 is graphic:

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

The matrix has at least one column with at least three 1s in it.

Select the first column that has at least three 1s.

Select the first row that has a 1 in the selected column.

selected column = 1, selected row = 1.

$Y$ (the set of columns with a 1 in this row) = \{1, 2, 3, 4, 5\}

Replace row 1 and all the columns with a 1 in this row with 0s,

the resulted matrix $R'$ is:

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

Get the bridges of $Y$, which are the elementary separators of the matroid represented by matrix $R'$.

Group the rows into row groups.

For two rows $a$ and $b$ in the same row group, there must be a column $c$ such that $c$ has a 1 in both row $a$ and row $b$.

The set of columns with 1s in the rows of the same row group is an elementary separator of the matroid represented by the matrix.

The elementary separators of the matroid represented by the matrix are:

\{6\} \{7\} \{9\} \{10\}

The groups of rows are:

\{2\} \{3\} \{4\} \{5\}

From the following matrix:

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

Adjoin all the rows in the same row group together, and add row 1 as the last row.

The formed matrix represents a $Y$ component of the matroid represented by the matrix.

The following matrices are formed:

\[
\begin{bmatrix}
0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
For each \( Y \) component, delete the columns with a 0 in the last row.

Reduce the formed matrix to standard form (within a permutation of columns).

Check each column. If there is a column with more than one 1 in it, then the corresponding bridge does not partition \( Y \).

Then since in a graphic matroid, every bridge partitions \( Y \), the matroid is not graphic.

Otherwise, the corresponding bridge determines a partition.

And each element of the partition is the set of columns with a 1 in the same row in the matrix.

The reduction to \( Y \) of \( Y \) component 1 is:

\[
\begin{bmatrix}
0 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

Reduce it to standard form:

\[
\begin{bmatrix}
0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

Bridge 1 determines partition \{2, 3, 4\} \{1, 5\}

The reduction to \( Y \) of \( Y \) component 2 is:

\[
\begin{bmatrix}
0 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

Reduce it to standard form:

\[
\begin{bmatrix}
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

Bridge 2 determines partition \{2, 3\} \{1, 4, 5\}

The reduction to \( Y \) of \( Y \) component 3 is:

\[
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

Reduce it to standard form:

\[
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1
\end{bmatrix}
\]

Bridge 3 determines partition \{1, 2\} \{3, 4, 5\}

The reduction to \( Y \) of \( Y \) component 4 is:

\[
\begin{bmatrix}
0 & 0 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

Reduce it to standard form:

\[
\begin{bmatrix}
0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

Bridge 4 determines partition \{3, 4\} \{1, 2, 5\}

The reduction to \( Y \) of \( Y \) component 5 is:

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

Reduce it to standard form:

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

Bridge 5 determines partition \{1, 2, 3\} \{4, 5\}

For two partitions \( P_1 \) and \( P_2 \), determined by bridges \( B_1 \) and \( B_2 \), respectively,

If the union of one element in \( P_1 \) and one element in \( P_2 \) is \( Y \), then \( B_1 \) and \( B_2 \) do not overlap. Otherwise, \( B_1 \) and \( B_2 \) overlap.
Try to group the bridges of $Y$ and hence the corresponding $Y$ components into at most two disjoint classes, such that any two bridges in the same class do not overlap. If this arrangement could be made, then $Y$ is an even circuit. Otherwise, $Y$ is not even; and since in a graphic matroid every circuit is even, the matroid is not graphic.

Bridge $Y$ component 1 is added into class 1.
The union of the partition element 2 in partition 2 with the partition element 1 in partition 1 is $Y$.

Bridge $Y$ component 2 is added into class 1.
There is no element in partition 3 whose union with an element in partition 1 is $Y$.
Hence bridge 3 overlaps with bridge 1, and cannot be added into class 1.

Bridge $Y$ component 3 is added into class 2.
The union of the partition element 2 in partition 4 with the partition element 1 in partition 1 is $Y$.
There is no element in partition 4 whose union with an element in partition 2 is $Y$.
Hence bridge 4 overlaps with bridge 2, and cannot be added into class 1.

The union of the partition element 2 in partition 4 with the partition element 2 in partition 3 is $Y$.

Bridge $Y$ component 4 is added into class 2.
There is no element in partition 5 whose union with an element in partition 1 is $Y$.
Hence bridge 5 overlaps with bridge 1, and cannot be added into class 1.

The union of the partition element 1 in partition 5 with the partition element 2 in partition 3 is $Y$.
There is no element in partition 5 whose union with an element in partition 4 is $Y$.
Hence bridge 5 overlaps with bridge 4, and cannot be added into class 2.

Since bridge 5 cannot be added into either of these two classes, $Y$ is not even and the matroid is not graphic.
The component 1 of the matroid $M$ is not graphic.
Since every minor of a graphic matroid is graphic, the matroid $M$ is not graphic.

### 6.1.4 Example 4

Given as input the following binary matrix, which represents the bond matroid of a Petersen graph, the program constructs a Petersen graph shown in Figure 9, which can be changed into the graph shown in Figure 10 by rearranging the vertices of the graph in the program.

```
100000000110000
010000000111000
001000000111100
000100000011110
000100000011110
000010000011111
000001000010111
000000100010101
00000001000101
```

### 6.1.5 Example 5

Given as input the following binary matrix, the program constructs a graph shown in Figure 11. By rearranging the vertices of the graph in the program,
Figure 9: The Petersen graph constructed by the program
Figure 10: The Petersen graph after rearrangement of the vertices of Figure 9
we can obtain a graph shown in Figure 12, so that it can be easily verified that each set of columns (edges) with a 1 in the same row of the input matrix is a bond of the graph.

\begin{verbatim}
100000000011100
010000000111100
001000000100100
000100000011000
000010000010010
000001000001010
000000100011100
000000100111001
000000010011001
000000001001001
\end{verbatim}

7 Conclusion

In this paper, we provided a formal description of Tutte’s algorithm for recognizing binary graphic matroids, so that the algorithm can be readily implemented. The correctness of the algorithm was shown in Section 4, and the running time was analyzed in Section 5. A Java implementation of the algorithm is provided. Some sample runs of the Java program were included in this paper. The program works correctly in all the tests we’ve run so far. As the Java program provides an explanation for why a given binary matroid represented by a binary matrix is graphic or non-graphic, it can also serve as an educational tool.

The Java program only tests whether a binary matroid is graphic. It would be nicer if we had a program to test whether a more general matroid were graphic, not just binary. Seymour published a paper in 1981 entitled “Recognizing Graphic Matroids”, which describes an algorithm for testing whether a matroid represented by means of an independent oracle is graphic, in which Tutte’s algorithm is used as a basis [1]. It would be nice to extend our program to implement this algorithm and have a way of determining whether a general matroid is graphic and if so, present a drawing of the graph that corresponds to the matroid.
Figure 11: The graph constructed by the program
Figure 12: The graph after rearrangement of the vertices of Figure 11
References


