Algorithm to Find All Cliques in a Graph

A. Ashok Kumar
Department of Computer Science , St. Xavier's College (Autonomous), Palayamkottai - 627 002.
E-mail: ashokjuno@rediff.mail.com
S. Athisayanathan
Research Department of Mathematics, St. Xavier's College (Autonomous),Palayamkottai - 627 002.
E-mail: athisayanathan@yahoo.co.in
A. Antonysamy
Research Department of Mathematics, St. Xavier's College (Autonomous),Palayamkottai - 627 002.
E-mail: fr_antonysamy@hotmail.com

ABSTRACT

-Let $V = \{1, 2, 3, . . . , n\}$ be the vertex set of a graph G, $P(V)$ the powerset of V and $A \in P(V)$. Then A can be represented as an ordered n-tuple $(x_1x_2x_3 . . .x_n)$ where $x_i = 1$ if $i \in A$, otherwise $x_i = 0$ $(1 \le i \le n)$. This representation is called *binary count* (or *BC*) representation of a set A and denoted as $BC(A)$. Given a graph G of order n, it is shown that every integer m in $S = \{0, 1, 2, . . . , 2^n - 1\}$ corresponds to a subset A of V and vice versa. We introduce algorithms to find a subset A of the vertex set $V = \{1, 2, 3, . . . , n\}$ of a graph G that corresponds to an integer m in $S = \{0, 1, 2, . . . , 2^n - 1\}$, verify whether A is a subset of any other subset B of V and also verify whether the sub graph $< A >$ induced by the set A is a clique or not using *BC* representation. Also a general algorithm to find all the cliques in a graph G using *BC* representation is introduced. Moreover we have proved the correctness of the algorithms and analyzed their time complexities.

Key Words: adjacency matrix, binary count, clique, powerset, subset.

Date of Submission: March 05, 2010 Date of Acceptance: May 19, 2010

1. Introduction

By a *graph* $G = (V,E)$ we mean a finite undirected graph without loops or multiple edges. $|V|$ and $|E|$ denote the *order* and *size* of G respectively. We consider connected graphs with atleast two vertices. A *clique* of a graph G is a maximal complete subgraph of G.

Various algorithms for finding cliques of graphs were developed by Bierstone [1] and Mulligan [2] and presented in an analysis of clustering techniques by Augustson and Minker [3]. The Bierstone [1] algorithm has errors and it was corrected by Mulligan and Corneil [4]. In [5] Coen Bron and Joep Kerboscht presented two versions of backtracking algorithms, using a branch- and-bound technique to cut o. branches that cannot lead to a clique. The first version is a straightforward implementation of the basic algorithm. It is mainly presented to illustrate the method used. This version generates cliques in alphabetic (lexicographic) order. The second version is derived from the first and generates cliques in a rather unpredictable order in an attempt to minimize the number of branches to be traversed. This version tends to produce the larger cliques first and to generate sequentially cliques having a large common intersection. In this paper we introduce a new concept called *Binary Count* (*BC*) representation to represent a subset of a set. Using this *BC* representation we introduce algorithms to .nd all the cliques in a given undirected graph G which is different from their approach. The following definitions are used in the sequel.

Definition 1.1 A graph H is called a *subgraph* of a graph G, written $H \subseteq G$, if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. A subgraph H of a graph G is called a *spanning subgraph* of G if $V(H) = V(G)$. For any set S of vertices of G, the *induced subgraph* $G[S]$ (or $< S >$) is the maximal subgraph of G with vertex set S. Thus two vertices of S are adjacent in $< S >$ if and only if they are adjacent in G.

Definition 1.2 Let $G = (V,E)$ be a graph with $V = \{v_1, v_2, . . . , v_n\}$. The adjacency matrix $Adj(G) = [a_{ij}]$ of G is the $n \times n$ matrix defined by

$$a_{ij} = \begin{cases} 1 \text{ if } v_iv_j \in E \\ 0 \text{ otherwise} \end{cases}$$

Remark 1.3 If $Adj(G) = [a_{ij}]$ is the adjacency matrix of a complete graph G, then

$$a_{ij} = \begin{cases} 1 \text{ if } i \neq j \\ 0 \text{ otherwise} \end{cases}$$

Definition 1.4 Let f and g be two functions defined on the set of positive integers. The order of f is said to be lower than or equal to the order of g if there exists a positive real constant C and a nonnegative integer n_0 such that

$$: f(n) \le Cg(n) \text{ for all } n > n_0.$$

If the order of f is lower than or equal to the order of g, we write $f(n) = O(g(n))$ or say $f(n)$ is $O(g(n))$

(read as $f(n)$ is *big oh* of $g(n)$). This means that f does not grow faster than g; the function f may grow more slowly than g or at the same rate. The functions f and g are of the same order if $f(n) = O(g(n))$ and $g(n) = O(f(n))$. For other basic definitions and terminologies we refer to [6, 7].

2. Algorithms

In this section, first we introduce a representation to represent a subset A of the vertex set V of a graph G which is called the *binary count* (or *BC*) representation.

Definition 2.1 Let $V = \{1, 2, 3, \ldots, n\}$ be the vertex set of a graph G and $P(V)$ the powerset of V. Let $A \in P(V)$. Then A can be represented as an ordered n-tuple $(x_1x_2x_3 \ldots x_n)$ such that

$$x_i = \begin{cases} 1 \text{ if } i \in A \\ 0 \text{ otherwise} \end{cases}$$

This representation is called *binary count* representation of a set A and denoted as $BC(A)$. To indicate the i^{th} term in $BC(A)$ we use $BC(A(i))$. Note that this BC representation is different from a binary form of a decimal number and we do not use ',' to separate the coordinates in BC representation.

Definition 2.2 If $BC(A) = (x_1x_2x_3 \ldots x_n)$ is the BC representation of a set A, then $V = \{1, 2, 3, \ldots, n\}$ is the vertex set, the subset $A \in P(V)$ represented by $BC(A)$ is $A = \{i : x_i = 1, 1 \le i \le n\}$ and the integer corresponds to $BC(A)$ is $m = x_1 2^{n-1} + x_2 2^{n-2} + \ldots + x_{n-1} 2^1 + x_n 2^0$. In the following two theorems, given a graph G of order n we prove that every integer m in $S = \{0, 1, 2, \ldots, 2^n-1\}$ corresponds to a subset A of $V(G) = \{1, 2, 3, \ldots, n\}$ and also we prove that given a subset A of $V(G) = \{1, 2, 3, \ldots, n\}$ that corresponds to an integer m in $S = \{0, 1, 2, \ldots, 2^n - 1\}$.

Theorem 2.3 *Let* $V = \{1, 2, 3, \ldots, n\}$ *be the vertex set of a graph G. Then every integer m in* $S = \{0, 1, 2, \ldots, 2^n - 1\}$ *corresponds to a subset A of V.*

Proof. Let $V = \{1, 2, 3, \ldots, n\}$ be the vertex set of a graph G, $S = \{0, 1, 2, \ldots, 2^n -1\}$, X is the set of all binary forms of the elements of S and $P(V)$ is the powerset of V. Clearly $|S| = |X| = |P(V)| = 2^n$. Let $f : S \longrightarrow X$ be a map such that $f(m) = (x_1x_2x_3 \ldots x_n)$ and $g : X \longrightarrow P(V)$ be a map such that $g(x_1x_2x_3 \ldots x_n) = \{i : x_i = 1, 1 \le i \le n\}$. Now define a map $h : S \longrightarrow P(V)$, such that $h = g \circ f$. Since f and g are 1 - 1 and onto, h is also 1 - 1 and onto. Therefore every integer m in S corresponds to a subset A of V.

Example 2.4 Let $V = \{1, 2, 3, 4\}$ be the vertex set of a graph G, $S = \{0, 1, 2, \ldots, 15\}$, $X = \{(0000), (0001), (0010), (0011), (0100), (0101), (0110), (0111), (1000), (1001), (1010), (1011), (1100), (1101, (1110), (1111)\}$, the binary form of the elements of S and $P(V) = \{\{\phi\}, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}, \{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{2, 3, 4\}, \{1, 2, 3, 4\}\}$. Let $f : S \longrightarrow X$ be a map such that $f(m) = (x_1x_2x_3 \ldots x_n)$ where $(x_1x_2x_3 \ldots x_n)$ is the binary form of m and $g : X \longrightarrow P(V)$ be a map such that $g(x_1x_2x_3 \ldots x_n) = \{i : x_i = 1, 1 \le i \le n\}$. Now define a map $h : S \longrightarrow P(V)$, such that $h = g \circ f$. The following Fig 2.1 shows the 1-1 correspondence between the elements of S and $P(V)$.

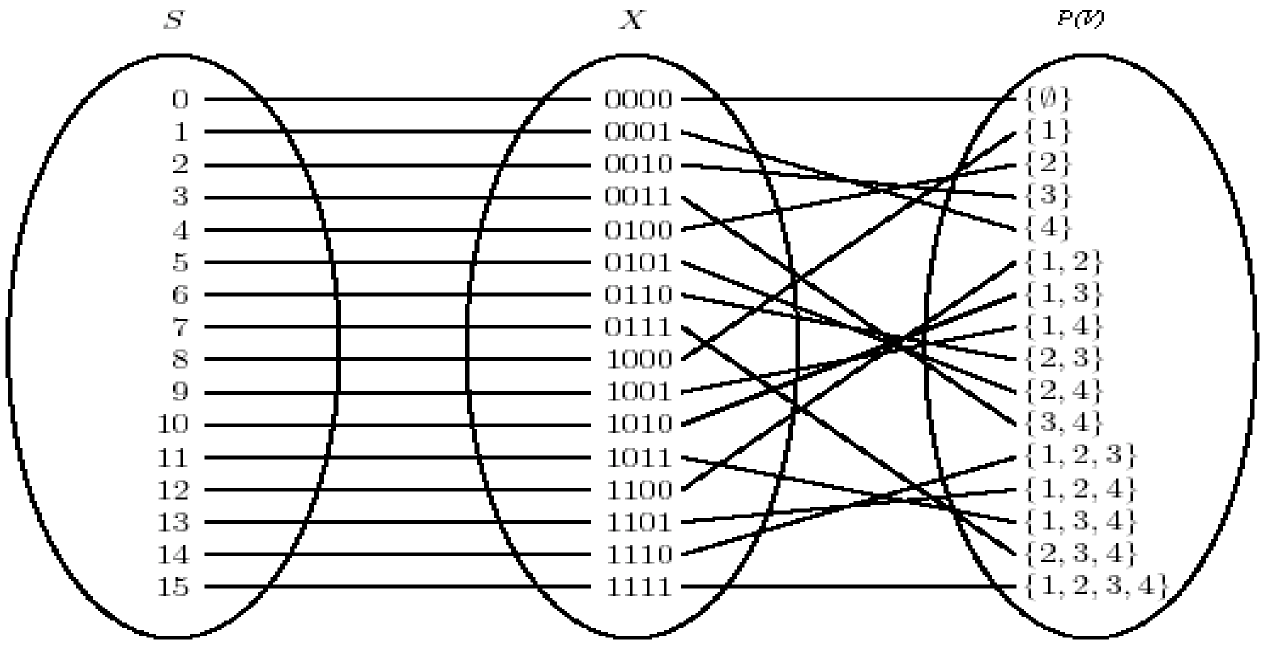

Fig. 2.1

Theorem 2.5 *Every sub set A of V corresponds to an integer m in* $S = \{0, 1, 2, \ldots, 2^n - 1\}$.

Proof. Let $V = \{1, 2, 3, \ldots, n\}$ be the vertex set of a graph G, $S = \{0, 1, 2, \ldots, 2^n - 1\}$, $P(V)$ the power set of V and $X = \{BC(A) : A \in P(V)\}$. Clearly $|S| = |X| = |P(V)| = 2^n$. Let $f : P(V) \longrightarrow X$ be a map such that $f(A) = BC(A)$ and $g : X \longrightarrow S$ be a map such that $g(BC(A)) = g((x_1x_2x_3 \ldots x_n)) = x_12^{n-1}+x_22^{n-2}+\ldots+ x_{n-1}2^1+x_n2^0$. Now define a map $h : P(V) \longrightarrow S$, such that $h = g \circ f$. Since f and g are 1-1 and onto, h is also 1 - 1 and onto. Therefore every integer m in S corresponds to a subset A of V.

Example 2.6 Let $V = \{1, 2, 3\}$ be the vertex set of a graph G, $P(V) = \{\{\phi\}, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$, $X = \{(000), (001), (010), (011), (100), (101), (110), (111)\}$ the BC forms of the elements of $P(V)$ and $S = \{0, 1, 2, \ldots, 7\}$. Let $f : P(V) \longrightarrow X$ be a map such that $f(A) = BC(A)$ and $g : X \longrightarrow S$ be a map such that $g(BC(A)) = g((x_1x_2x_3 \ldots x_n)) = x_12^{n-1} + x_22^{n-2} + \ldots + x_{n-1}2^1 + x_n2^0$. Now define a map $h : P(V) \longrightarrow S$, such that $h = g \circ f$. The following Fig. 2.2 shows the 1 - 1 correspondance between the elements of S and $P(V)$.

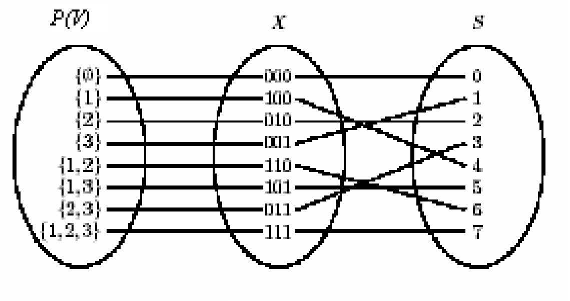

Fig.2.2

Let G be a connected graph with $V = \{1, 2, \ldots, n\}$ and $S = \{0, 1, 2, \ldots, 2^n - 1\}$. In the following, given an integer m in S we introduce an algorithm to find a subset A of V that corresponds to m.

Algorithm 2.7 *Let G be a graph,* $V = \{1, 2, \ldots, n\}$ *be a vertex set and* $S = \{0, 1, 2, \ldots, 2^n - 1\}$.

```
1. Let $m \in S$.
2. Convert m into binary form. Let the binary form of m be $(x_1x_2x_3 \ldots x_n)$.
3. $A = \{i : x_i = 1\}$
4. stop
```

Theorem 2.8 *Given an integer* $m \in S$*, the Algorithm 2.7 finds the subset A of the vertex set V of a graph G that corresponds to m.*

Proof. By Theorem 2.3, the Algorithm 2.7 finds A of V that corresponds to m.

Theorem 2.9 *The time complexity of the Algorithm 2.7 is* $O(n)$.

Proof. Step 2 takes $O(n)$ for converting the number to binary form. Step 3 takes $O(n)$ time to find the subset from the binary form. So the computing time for the Algorithm 2.7 is $O(n)$.

Example 2.10 Let $V = \{1, 2, 3, 4\}$ be the vertex set of a graph G, $S = \{0, 1, 2, \ldots, 15\}$.

1. Let $m = 14 \in S$.
2. Binary form of m is 1110.
3. Let $A = \{1, 2, 3\}$.
4. stop

Next, we introduce an algorithm to find a set A is the subset of a set B or not using BC representation.

Definition 2.11 Let $S = \{1, 2, 3, \ldots, n\}$, $A, B \in P(V)$, $BC(A) = (x_1x_2x_3 \ldots x_n)$ and $BC(B) = (y_1y_2y_3 \ldots y_n)$. Then, $A \subseteq B$ if $x_i = 1$ $(1 \le i \le n)$ in $BC(A)$ implies $y_i = 1$ $(1 \le i \le n)$ in $BC(B)$.

Algorithm 2.12 *Let* $BC(A)$ *and* $BC(B)$ *be the BC representation of two subsets A and B of V.*

```
1. subset = true
2. for i = 1 to n
3. if $BC(A(i)) = 1$ then
1. if $BC(B(i))$ _= 1 then subset = false; goto step 5
4. next i
5. return subset
6. stop
```

The following theorem follows immediately from the Definition 2.11.

Theorem 2.13 *Let A and B be two sets in BC representation. Then the Algorithm 2.12, finds whether A is a subset of B or not.*

Theorem 2.14 *The Algorithm 2.12 finds A is the subset of B or not in* $O(n)$ *time.*

Proof. The steps 2 to 4 are executed for n times for comparing the binary count representation of $BC(A(i))$ and $BC(B(i))$. So the time complexity of the Algorithm 2.12 is $O(n)$ time.

Example 2.15 Let $V = \{1, 2, 3, 4\}$ be the vertex set of a graph G. Let $A = \{1\}$, $B = \{2, 3\}$ and $C = \{1, 2, 3\} \in S$. Then $BC(A) = (1000)$, $BC(B) = (0110)$ and $BC(C) = (1110)$. Now $x_1 = 1$ in $BC(A)$ but $y_1 \neq 1$ in $BC(B)$ therefore $A \not\subset B$. Where as $x_2 = x_3 = 1$ in $BC(B)$ and $y_2 = y_3 = 1$ in $BC(C)$, $B \subseteq C$.

Now, we introduce an algorithm to find the subgraph $< A >$induced by the subset A of the vertex set V of a graph G is a clique or not.

Algorithm 2.16 *Let G be a graph of order n,* $V(G) = \{1, 2, 3, \ldots, n\}$ *and* $A \subseteq V$.

```
1. clique = false
2. Let A is in its BC form.
3. Let Adj(G) be the adjacency matrix of G.
4. Find the adjacency matrix Adj(< A >) as follows: Find the 0's in BC(A) and remove the corresponding rows and columns of Adj(G). The resulting matrix is Adj(< A >).
5. Verify Adj(< A >) is complete. If it is complete then clique = true.
6. return clique
7. stop
```

Theorem 2.17 *The Algorithm 2.16 returns whether the given subgraph < A > induced by the subset A of the vertex set V of a graph G is a clique or not.*

Proof. Step 1 initializes the Boolean variable *clique* = false. Step 2 reads the $BC(A)$. At step 4 we construct the $Adj(< A >)$ by removing the i^{th} row and i^{th} column from $Adj(G)$ of G for every $BC(A(i)) = 0$ $(1 \leq i \leq n)$. At step 5 the adjacency matrix of A is compared, whether the matrix is complete or not. If complete then *clique* = true otherwise *clique*= false.

Theorem 2.18 *The Algorithm 2.16 finds the subgraph < A > induced by the subset A is a clique or not in* $O(n^2)$ *time.*

Proof. Step 4 is executed for n times to construct the adjacency matrix $Adj(A)$ and the computation time is $O(n)$. In order to find the completeness the step 5 is executed for maximum of $O(n^2)$ times. So the computation time for the Algorithm 2.16 is $O(n^2)$.

Example 2.19 Let G be a graph given in Figure 2.3. Let $V = \{1, 2, 3, 4\}$ and $A = \{1, 2, 3\}$. Now, let us verify whether the sub graph $< A >$ induced by the subset A of the vertex set V of a graph is clique or not.

1. *clique* = false
2. Let $BC(A) = (1110)$.
3. Let $Adj(G)$ be the given adjacency matrix of G.

$$Adj(G) = \begin{pmatrix} 0111 \\ 1010 \\ 1101 \\ 1010 \end{pmatrix}$$

4. The adjacency matrix $Adj(A)$ of $< A >$ is obtained by the Algorithm 2.16 is as follows: In $BC(A)$, 4*th* element is 0, therefore remove 4^{th} row and 4^{th} column of $Adj(G)$. So that

$$Adj(A) = \begin{pmatrix} 011 \\ 101 \\ 110 \end{pmatrix}$$

5. All the entries of $Adj(A)$, except the diagonal are 1, A is complete. So, *clique* = true.

Therefore the algorithm returns the sub graph $< A >$ induced by the subset A of the vertex set V of a graph G is clique.

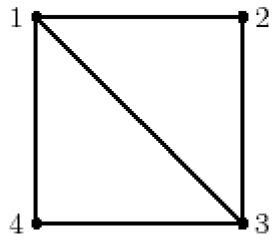

Fig. 2.3 : G

3. The General Algorithm

Finally, we introduce a general algorithm to fid all the cliques in a connected graph G of order n.

Algorithm 3.1 *Let* $V = \{1, 2, 3, \dots, n\}$ *be the vertex set of a graph G,* $S = \{0, 1, 2, \dots, 2^n - 1\}$*, let* $C = \{c_1, c_2, \dots, c_p\}$ *denotes the set of all cliques, initially C is empty.*

```
1. Initialize $C = \{\phi\}$.
2. for $m = 2^n - 1$ to 1
   Find the subset A by calling the Algorithm 2.7.
   Check A is a subset of any other set in the clique list C using the Algorithm 2.12.
   If subset = true, then goto step 3
   Call Algorithm 2.16 to find the sub graph < A > induced by A is a clique or not.
   If clique = true, then store A in the clique list C.
3. Next m
4. Stop
```

The following theorem follows immediately from Theorems 2.8, 2.13 and 2.17.

Theorem 3.2 *The Algorithm 3.1 lists the cliques in a graph G.*

Theorem 3.3 *All cliques in a graph G can be found in* $O(2^n n^2)$ *using Algorithm 3.1.*

Proof. The step 2 is executed for $2^n - 1$ times to find the subset of vertex set V. This step calls the Algorithms 2.7, 2.12 and 2.16. The time complexity of the Algorithms 2.7, 2.12 and 2.16 are $O(n)$, $O(n)$ and $O(n^2)$ respectively. So, the time complexity of the step 2 is $O(2^n n^2)$. Thus the complexity of Algorithm 3.1 is $O(2^n n^2)$.

Example 3.4 Using the Algorithm 2.16, let us find all the cliques in the graph G given in Figure 2.3.

Let $V = \{1, 2, 3, 4\}$, $S = \{0, 1, 2, 3, \dots, 15\}$.

Step 1: $C = \{\phi\}$

Step 2: Let $m = 15$

- **Step 2.1:** by the Algorithm 2.7, m corresponds to the sub set $A = \{1, 2, 3, 4\}$.
- **Step 2.2:** by the Algorithm 2.12, checks that the set A is a sub set of any other set in the clique list C and returns *subset* = false.
- **Step 2.3:** Since *subset* = false goto next step 2.4.
- **Step 2.4:** the Algorithm 2.16 checks A is a clique or not and returns *clique* = false.
- **Step 2.5:** since *clique* = false, goto step 3

Step 3: goto step 2.

Step 2: Let $m = 14$,

- **Step 2.1:** by the Algorithm 2.7, m corresponds to the sub set $A = \{1, 2, 3\}$.
- **Step 2.2:** by the Algorithm 2.12, checks that the set A is a sub set of any other set in the clique list C and returns *subset* = false.
- **Step 2.3:** Since *subset* = false goto step 2.4.
- **Step 2.4:** the Algorithm 2.16 checks A is a clique or not and returns *clique* = true.
- **Step 2.5:** since *clique* = true, store A into clique list $C = \{(1110)\}$, goto step 3

Step 3: goto step 2.

Similarly we find all the subsets A of the vertex set V of G that corresponds to an integer m in $S = \{13, 12, . . . , 1\}$ and verify whether A is a subset of any other subset of C and also verify whether the sub graph $< A >$ induced by A is a clique or not. Hence this algorithm lists the following cliques $C = \{(1110), (1011)\}$ and by Definition 2.1 the vertex sets of the cliques are $\{\{1, 2, 3\}, \{1, 3, 4\}\}$.

4. Conclusion

In this paper we have introduced a new representation called BC representation to represent a subset of a set and using this BC representation, we have developed an algorithm to find a subset A of $V = \{1, 2, . . ., n\}$ that corresponds an integer $m \in S = \{0, 1, 2, . . . , 2^n - 1\}$, an algorithm to find a set A is the subset of a set B or not, an algorithm to find the subgraph $< A >$ induced by the subset A of the vertex set V of a graph G is a clique or not and a general algorithm to find all the cliques in a graph G. Further using this BC representation we can develop algorithms to find clique graph of a graph G and algorithms for central structures.

References

[1] E. Bierstone, Cliques and generalized cliques in a finite linear graph, *Unpublished report.*

[2] G.D. Mulligan, *Algorithms for finding cliques of a graph*, M.Sc. thesis, Dep. Of Computer Sci., U. of Toronto, Toronto, Ontario, Can.

[3] J .G. Augustson and J . Minker, An analysis of some graph theoretical cluster techniques. *Journal of ACM*. **17**, No.4 (1970), 571-588.

[4] G.D. Mulligan and D.G. Corneil, Corrections to Bierstone's Algorithm for Generating Cliques, *Journal of the Association for Computing Machinery*. **19**, No. 2 (1972), 244-247.

[5] Coen Bron and Joep Kerboscht, Finding All Cliques of an Undirected Graph [H], *Communications of the ACM*. **16**, No.9 (1973),575 - 580.

[6] G. Chartrand and Ortrud R. Oellermann, *Applied and Algorithmic Graph Theory*, McGraw-Hill International editions (1993).

[7] K.R. Parthasarathy, *Basic Graph Theory*, Tata McGraw-Hill Publishing Company Limited, New Delhi (1994)

Authors Biography

Author 1:
Name: A.Ashok Kumar
Qualification: M.Sc., M.Phil., (Ph.D.)
Experience: 14 Years Teaching in Computer Science (13 years at St.Xavier's College(Autonomous), Palayamkottai-627002)
Research Experience: 4 Years
Books Published : 3
Present working Address: Assistant Professor, Department Of Computer Science, V.S.S. Govt. Arts College, Pulankurichi, Sivagangai Dist., Tamil Nadu.
E-mail: ashokjuno@rediff.mail.com

Author 2:
Name: Dr. S. Athisayanathan
Qualification: M.Sc., M.Phil., Ph.D.
Experience: 27 Years Teaching in Mathematics
Research Experience: 8 Years
Present Working Address: Professor, Research Department of Mathematics, St.Xavier's College(Autonomous), Palayamkottai-627002, TamilNadu.
E-mail: athisayanathan@yahoo.co.in

Author 3:
Name: Rev. Dr. A. Antonysamy S.J.
Qualification: M.Sc., M.Phil., Ph.D.
Experience: 35 Years Teaching in Mathematics (worked in St.Joseph's College, Trichy and St. Xavier's College, Palayamkottai at various levels like Vice-principal and Principal)
Research Experience: 20 Years
Present Working Address: Principal, St.Xavier's College, Kathmandu, Nepal.
E-mail: fr_antonysamy@hotmail.com