The Worst-Case Time Complexity for Finding All the Cliques

TECHNICAL REPORT
of UEC, UEC-TR-C5(2), 1988

The Worst-Case Time Complexity for Finding All the Cliques*

Etsuji Tomita Akira Tanaka Haruhisa Takahashi

Department of
Communications and Systems

THE UNIVERSITY
OF
ELECTRO-COMMUNICATIONS

Chofugaoka, Chofu, TOKYO 182 JAPAN

Abstract. We present an algorithm for finding all the cliques of an undirected graph, which is based upon Bron and Kerbosch's algorithm. Then we prove that its worst-case time complexity is $O(3^{n/3})=O(2^{n/1.89\ldots})$ for an n-vertex graph. This is the optimal result in a sense that there exist up to $3^{n/3}$ cliques in an n-vertex graph. Bron and Kerbosch's algorithm or its variant has been known to be very efficient by extensive experiments, while the theoretical analysis of its running time was long left to be unknown. The present paper gives a definite conclusion to this problem.

1. Introduction.

A maximal complete subgraph of an undirected graph G is called a *clique*. A clique of the complementary graph of G is a *maximal independent set*. Finding all the cliques or all the maximal independent sets of a given graph is a fundamental problem in the theory of graphs and has many diverse applications. Then a number of algorithms have been presented and evaluated experimentally or theoretically, see, e.g., [2], [4]-[9], [11]-[13], and [15]. Among them, Tsukiyama et al.[15] presented an algorithm for generating all the maximal independent sets in a graph G in $O(nm)$ time per maximal independent set, where n and m are the numbers of vertices and edges of G, respectively. Furthermore, Chiba and Nishizeki[5] improved it much to have a more efficient algorithm for listing all the cliques of G in $O(a(G)m)$ time per clique, where $a(G)$ is the arboricity of G with $a(G)\leqq O(m^{1/2})$ for a connected graph G. Few other theoretical time complexity analyses have been found for these problems except for a special work of Tarjan and Trojanowski[14], in which they presented an algorithm for finding *only one* maximum independent set in an n-vertex graph in $O(2^{n/3})$ time.

We present here an algorithm for finding all the cliques of an undirected graph which is essentially based upon Bron and Kerbosch[4]. Then we prove that its worst-case running time complexity is $O(3^{n/3})$ for a graph with n vertices. This is the theoretical limit with respect to n, since there exist up to $3^{n/3}=3^{k}$ cliques in a graph with $n=3k$ vertices as shown by Moon and Moser[10]. It has already been shown thus far that Bron and Kerbosch's algorithm is very efficient in reality by extensive experiments in [4], [8], [6], and Reingold et al.[13, p.390], cf.[3]. Our present paper is the first to give theoretical evidence for these results.

2. Preliminaries.

[1] Throughout this paper, we consider a simple undirected *graph* $G=(V,E)$ with a finite set V of *vertices* and a finite set E of unordered pairs (v,w) of distinct vertices, called *edges*. A pair of vertices v and w are said to be *adjacent* if $(v,w)\in E$. We call $\overline{G}=(V,\overline{E})$ with $\overline{E}=\{(v,w) \in V\times V \mid v\neq w$, and $(v,w)\notin E\}$ a *complementary* graph of G.

[2] For a vertex $v\in V$, let $\Gamma(v)$ be the set of all vertices which are adjacent to v in $G=(V,E)$, i.e.,

$$\Gamma(v)=\{w\in V \mid (v,w)\in E\} \quad (\not\ni v).$$

[3] For a subset $W\subseteq V$ of vertices, $G(W)=(W,E(W))$ with $E(W)=\{(v,w)\in W\times W \mid (v,w)\in E\}$ is called a *subgraph* of $G=(V,E)$ *induced* by W. For a set W of vertices, $|W|$ denotes the number of elements in W.

[4] Given a subset $Q\subseteq V$ of vertices, the induced subgraph $G(Q)$ is said to be *complete* if $(v,w)\in E$ for all $v,w\in Q$ with $v\neq w$. If this is the case, we may simply say that Q is a complete subgraph. In particular, if a complete subgraph is *maximal*, then it is called a *clique*. A subset $W\subseteq V$ of vertices is said to be *independent* if $(v,w)\notin E$ for all $v,w\in W$.

Here, $Q \subseteq V$ is a clique of G if and only if Q is a maximal independent set of the complementary graph $\overline{G}$.

3. The algorithm.

We present a backtrack searching algorithm CLIQUES for finding all the cliques of a given graph G=(V,E) ($V \neq \phi$) that is essentially based upon Bron and Kerbosch[4].

Here we introduce a global variable Q of a set of vertices which constitute a complete subgraph being found up to this time. Then we begin the algorithm CLIQUES by letting $Q:=\phi$, and extend it step by step by applying a recursive procedure EXTEND to V and its succeeding induced subgraphs searching for larger and larger complete subgraphs until they reach maximals.

Let $Q=\{p_1, p_2, \cdots, p_d\}$ be found to be a complete subgraph at some stage, and consider a *subg*raph G(SUBG) which is induced by a set of vertices

$$SUBG=V \cap \Gamma(p_1) \cap \Gamma(p_2) \cap \cdots \cap \Gamma(p_d)$$

where SUBG=V when $Q=\phi$ at the initial stage. Then apply the procedure EXTEND to SUBG searching for larger complete subgraphs. If $SUBG=\phi$ then Q is clearly a *maximal* complete subgraph, i.e., a clique. Otherwise, $Q \cup \{q\}$ is a larger complete subgraph for every $q \in SUBG$. Then consider smaller subgraphs $G(SUBG_q)$ which are induced by new sets of vertices

$$SUBG_q = SUBG \cap \Gamma(q)$$

for all $q \in SUBG$, and apply recursively the same procedure EXTEND to $SUBG_q$ to find larger complete subgraphs containing $Q \cup \{q\}$. See FIG.1 for an illustration.

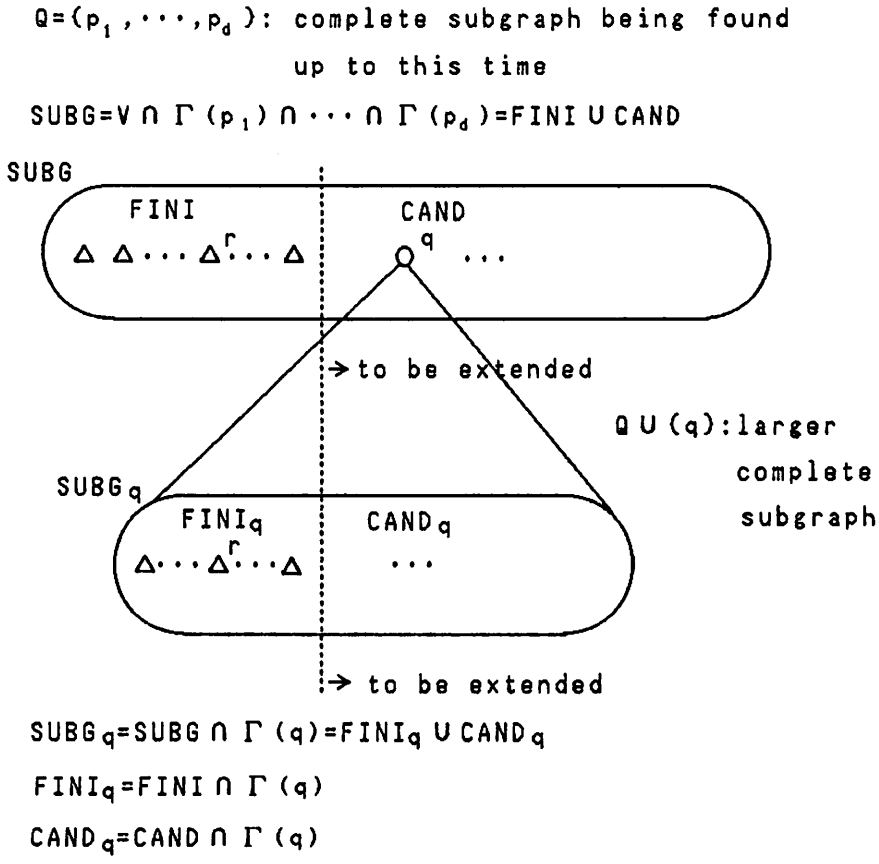


FIG.1. An illustration for the procedure EXTEND

Thus far we have shown only the basic framework of the algorithm for finding all the cliques (with possible duplications). This process can be represented by the following search forest, or the collection of search trees: The set of roots of the search forest is exactly the same as V of the graph $G=(V,E)$. For each $q \in SUBG$, all vertices in $SUBG_q = SUBG \cap \Gamma(q)$ are sons of q. Thus, a set of vertices along a path from a root to any vertex of the search forest constitutes a complete subgraph. For the terminology concerned with a tree and a forest, see, e.g., Aho et al.[1, pp.52-53]. We shall show an example of a search forest (with unnecessary subtrees deleted) later in FIG.2(b).

Now we proceed to describe two methods to prune unnecessary parts of the search forest.

First, for the previously described set $SUBG(\neq \phi)$, let

$$SUBG = FINI \cup CAND \quad (FINI \cap CAND = \phi).$$

Here suppose that we have already *finis*hed extending search subtrees from every vertex $q' \in FINI \subseteq SUBG$ to find all the cliques containing $Q \cup \{q'\}$, and that only the remaining vertex $q \in CAND \subseteq SUBG$ is a *cand*idate for further extension of the present complete subgraph Q to find new cliques. Consider the subgraph $G(SUBG_q)$ with $SUBG_q = SUBG \cap \Gamma(q)$, and let

$$SUBG_q = FINI_q \cup CAND_q \quad (FINI_q \cap CAND_q = \phi),$$

where

$$FINI_q = FINI \cap \Gamma(q), \quad \text{and} \quad CAND_q = CAND \cap \Gamma(q).$$

Then only the vertices in the subgraph $G(CAND_q)$ can be candidates for extending the complete subgraph $Q \cup \{q\}$ to find *new* larger cliques, since all the cliques containing $(Q \cup \{q\}) \cup \{r\}$ with $r \in FINI_q \subseteq FINI$ have already been finished to be found previously for any r by application of the procedure EXTEND to FINI as stated before. Thus further extension is to be considered only for vertices in $G(CAND_q)$ excluding ones in

$FINI_q = SUBG_q - CAND_q$.

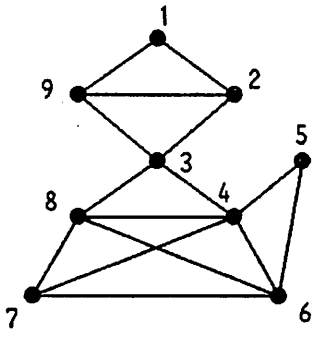
Secondly, given a certain vertex $u \in SUBG$, let $v \in SUBG \cap \Gamma(u)$. Then the set of all cliques properly containing $Q \cup \{u\}$ in the subgraph $G(Q \cup SUBG)$ is exactly the same as that properly containing $Q \cup \{v\}$ in $G(Q \cup SUBG)$, since $Q \cup \{u\} \cup \{v\}$ is a complete subgraph in $G(Q \cup SUBG)$. Therefore, if we extend a search subtree from u, then we need not extend any search subtree from $v \in SUBG \cap \Gamma(u)$. Taking the previous pruning method into consideration, too, the only search subtrees to be extended are from vertices in $(SUBG - SUBG \cap \Gamma(u)) - FINI = CAND - \Gamma(u)$ $(\exists u)$. Here, in order to minimize $|CAND - \Gamma(u)|$, we choose such a vertex $u \in SUBG$ to be the one which maximizes $|CAND \cap \Gamma(u)|$. In this way, the problem of finding all the cliques of G(CAND) can be decomposed into $k = |CAND - \Gamma(u)|$ such subproblems, see LEMMA (i) below.

With these two pruning methods, we have the following algorithm CLIQUES for finding all the cliques without duplications.

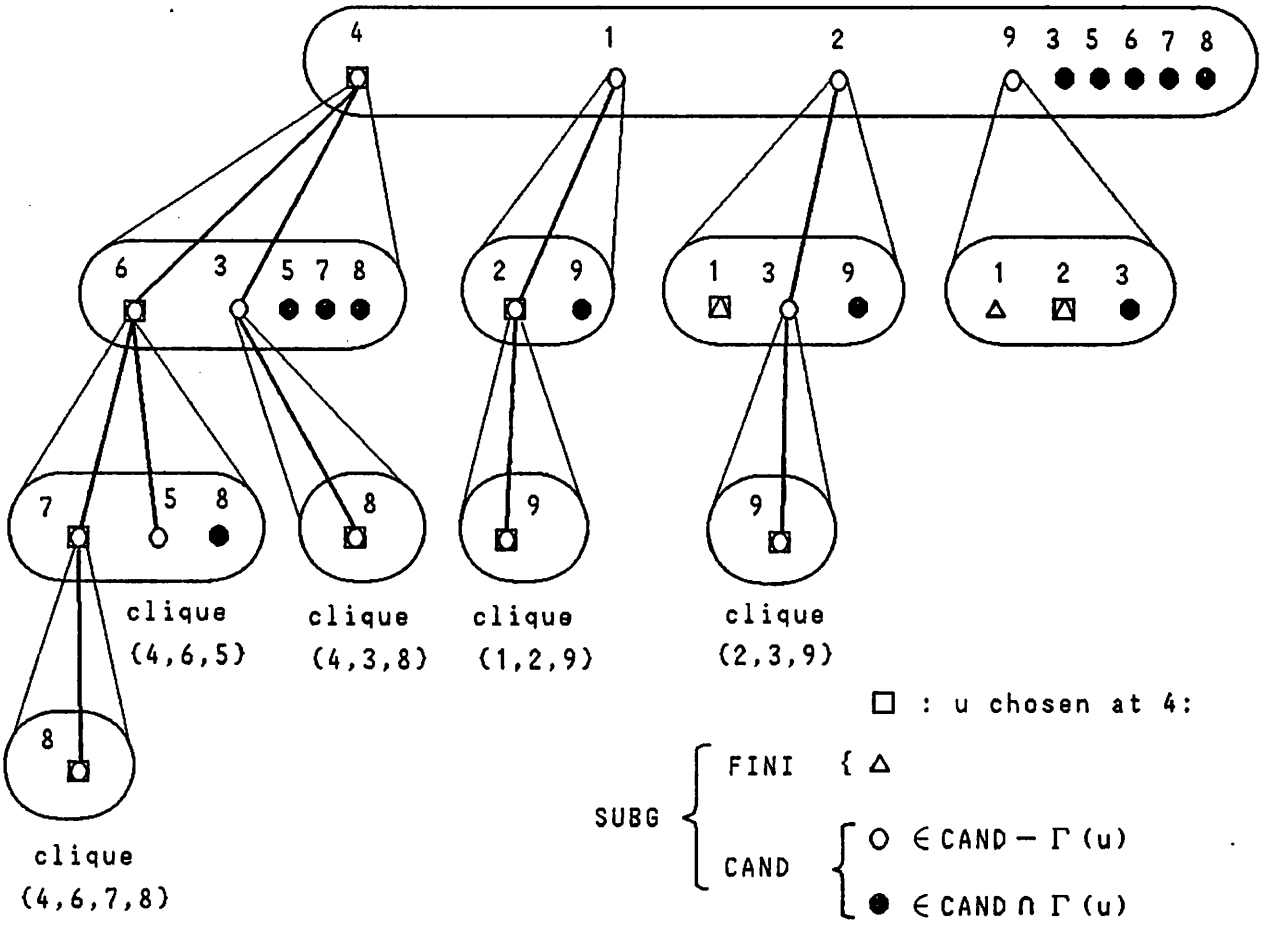
```
procedure CLIQUES(G)
          {Graph G=(V,E)}
begin
 0' : Q:=φ
     {global variable Q is to constitute a complete subgraph}
 1 : EXTEND(V,V)
     procedure EXTEND(SUBG, CAND)
     begin
 2 :   if SUBG=φ
 3 :         then print ("clique,")
                   {to represent that Q is a clique}
 4 :         else u:=vertex in SUBG, which maximizes | CAND∩ Γ (u)|
                   {let EXT_u=CAND-Γ (u)}
 5 :              while CAND-Γ (u)≠ φ
 6 :                 do q:=vertex in (CAND-Γ (u))
 7 :                    print (q, ",")
                        {to represent the next statement}
 7' :                   Q:=Q∪ {q}
 8 :                    SUBG_q:=SUBG∩ Γ (q);  CAND_q:=CAND∩ Γ (q)
 9 :                    EXTEND(SUBG_q, CAND_q)
10 :                    CAND:=CAND-{q}  {FINI:=FINI∪ {q}}
11 :                    print ("back,")
                        {to represent the next statement}
11' :                   Q:=Q-{q}
                    od
       fi
     end {of EXTEND}
end {of CLIQUES}
```

Note here that our sets Q, FINI, and CAND of vertices correspond to those of *compsub*, *not*, and *candidates* of Bron and Kerbosch[4]. See also Johnston[8] and Reingold et al.[13, pp.353-359, and pp.389-390] for reference. However, every time Q is found to be a clique at statement 2, we only print out a string of characters "clique," instead of Q itself in statement 3. This is because, otherwise, it is impossible to achieve the worst-case running time of $O(3^{n/3})$ for an n-vertex graph, since printing out of Q requires time proportional to the size of Q which is a global variable. Instead, in addition to statement 3, not only we print out q followed by a comma at statement 7 every time q is picked out as a new element of a complete subgraph, but also we print out a string of characters "back," at statement 11 after q is moved from CAND to FINI at statement 10. We can easily obtain a tree representation of all the cliques from the resultant sequence printed by statements 3, 7, and 11, as will be seen afterwards. Here, primed statements 0′, 7′, and 11′ are only for the sake of explanation, and should be deleted finally.

Example. Let us apply the above algorithm CLIQUES to a graph in FIG.2(a). Then the whole process is represented by a search forest in FIG.2(b), and we have the resultant printed sequence in FIG.3(a). In FIG.2(b), each set of vertices surrounded by a flat circle represents SUBG at that stage, in which vertex with △ mark is in FINI$\subseteq$SUBG at the beginning. Vertex u chosen in statement 4 is marked by ⊡ or ⊠ depending on whether it is in CAND or FINI, respectively. Other vertices in CAND$-\Gamma(u)$ are marked by ○, while vertices in CAND$\cap\Gamma(u)$ are marked by ●. Thus, all the cliques of G are {4,6,7,8}, {4,6,5}, {4,3,8}, {1,2,9}, and {2,3,9}.



(a) An input graph G

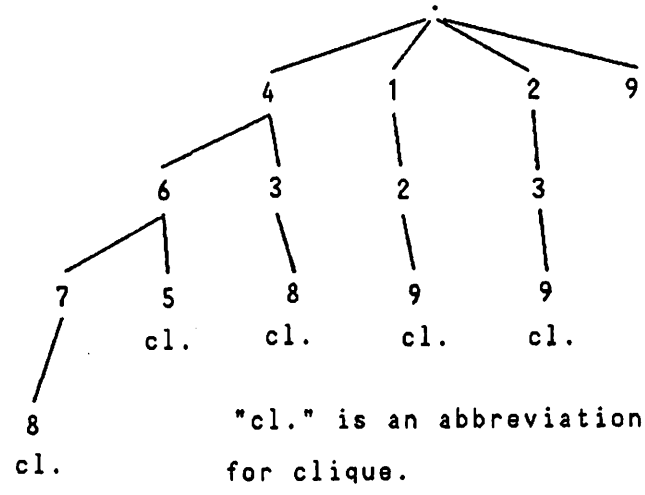


(b) A search forest for G

FIG.2. An example

4,6,7,8,clique,back,back,
5,clique,back,back,3,8,clique,
back,back,back,1,2,9,clique,
back,back,back,2,3,9,clique,
back,back,back,9,back,

(a) A resultant printed sequence



(b) A tree representation of cliques found.

FIG.3. A result of the example

Now given only the resultant printed sequence in FIG.3(a), we can obtain the same result as above by reconstructing from it a tree as in FIG.3(b) which represents a principal part of the previous search forest in FIG.2(b). Here, a dot "・" ($\notin V$) is introduced as a virtual root of the tree at the beginning. Then every time "q," is encountered in the sequence, we extend a downward edge whose end point is labeled by q. If "q," is followed by "clique,", the set of all the vertices along the path from the root to the vertex q excluding the root represents a clique. Every time "back," is encountered in the sequence, we go up the tree backward by one edge to find other alternatives. It is clear that this transformation can be done in time proportional to the length of the resultant sequence.

4. The worst-case time complexity.

We evaluate the worst-case running time of the previous algorithm CLIQUES(G) with the primed statements 0′, 7′, and 11′ having been deleted. So, this is equivalent to evaluating that of EXTEND(V, V). Now we begin by giving a few definitions.

[1] Let $T(n,m)$ be an upper bound on the worst-case running time of EXTEND(SUBG, CAND) when $|\,\mathrm{SUBG}\,| = n$ and $|\,\mathrm{CAND}\,| = m$ ($n \geqq m \geqq 0$).

[2] Let $T_k(n,m)$ be an upper bound on the worst-case running time of EXTEND(SUBG, CAND) when $|\,\mathrm{SUBG}\,| = n$, $|\,\mathrm{CAND}\,| = m$, and $|\,\mathrm{EXT}_u\,| = |\,\mathrm{CAND} - \Gamma(u)\,| = k$ at the first entrance to statement 5.

[3] Let us consider a nonrecursive procedure EXTEND_0(SUBG, CAND) which is obtained from EXTEND(SUBG, CAND) by deleting a recursive call 9: $\mathrm{EXTEND}(\mathrm{SUBG}_q, \mathrm{CAND}_q)$. The running time of EXTEND_0(SUBG, CAND) when $|\,\mathrm{SUBG}\,| = n$ and $|\,\mathrm{CAND}\,| = m$ can be made to be $O(n^2)$, then let this running time be less than or equal to the following quadratic equation

$P(n)=p_1n^2+p_2n+p_3$, where $p_1>0$, $p_2\geqq 0$, $p_3\geqq 0$. □

From the above definitions, we have that

(1) $$T(n,m)=\max_{0\leqq k\leqq m}\{T_k(n,m)\} \leqq \max_{1\leqq k\leqq m}\{T_k(n,m)\},$$

since $T_0(n,m)\leqq T_k(n,m)$ for any k, $1\leqq k\leqq m$.

The following lemma is a key for evaluating $T(n,m)$.

LEMMA. Consider EXTEND(SUBG, CAND) when $|\,\mathrm{SUBG}\,|=n$, $|\,\mathrm{CAND}\,|=m$, $|\,\mathrm{EXT_u}\,|=|\,\mathrm{CAND}-\Gamma(u)\,|=k\neq 0$, and $|\,\mathrm{CAND}\cap\Gamma(u)\,|=\ell$ at the first entrance to statement 5. In what follows, CAND stands only for this initial value, though it is decreased one by one at statement 10 in the while loop. Let $\mathrm{CAND}-\Gamma(u)=\{v_1, v_2, \cdots, v_k\}$ and the vertex at statement 6 be chosen in this order. Let

$\mathrm{SUBG}_i=\mathrm{SUBG}\cap\Gamma(v_i)$, and

$\mathrm{CAND}_i=(\mathrm{CAND}-\{v_1, v_2, \cdots, v_{i-1}\})\cap\Gamma(v_i)$.

Then

(i) $$T_k(|\,\mathrm{SUBG}\,|,\ |\,\mathrm{CAND}\,|)\leqq \sum_{i=1}^{k}(T(|\,\mathrm{SUBG}_i\,|,\ |\,\mathrm{CAND}_i\,|)+P(n),$$

(ii) a) $|\,\mathrm{CAND}_i\,|\leqq\ell$, and

b) $|\,\mathrm{SUBG}_i\,|\leqq n-k\leqq n-1$.

Proof. (i) It is obvious from procedure EXTEND(SUBG, CAND) and the definition of $P(n)$.

(ii) a) $|\,\mathrm{CAND}_i\,|\leqq|\,\mathrm{CAND}\cap\Gamma(v_i)\,|\leqq|\,\mathrm{CAND}\cap\Gamma(u)\,|=\ell$.

b) $$|\,\mathrm{SUBG}_i\,|=|\,\mathrm{SUBG}\cap\Gamma(v_i)\,|$$
$$=|\,\mathrm{FINI}\cap\Gamma(v_i)\,|+|\,\mathrm{CAND}\cap\Gamma(v_i)\,|,$$

with $|\,\mathrm{FINI}\cap\Gamma(v_i)\,|\leqq|\,\mathrm{FINI}\,|=n-m$ and $|\,\mathrm{CAND}\cap\Gamma(v_i)\,|\leqq\ell$. Here, $(n-m)+\ell=n-(m-\ell)=n-k\leqq n-1$, since $k\geqq 1$. Q.E.D.

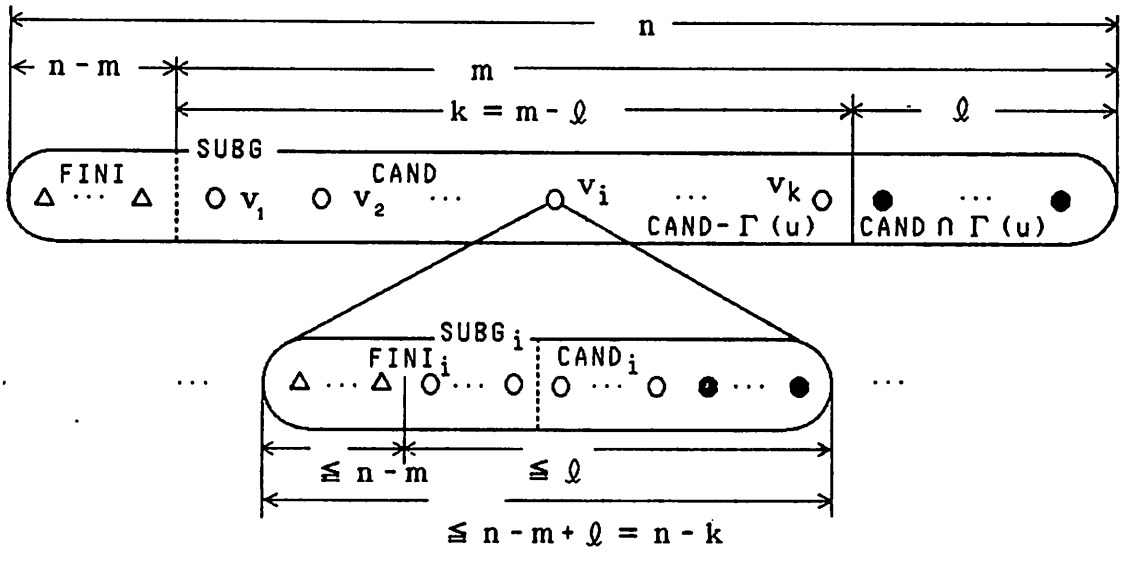


FIG.4. An illustration for LEMMA

THEOREM. The upper bound $T(n,m)$ on the worst-case running time of EXTEND(SUBG, CAND) with $|\,\mathrm{SUBG}\,| = n$ and $|\,\mathrm{CAND}\,| = m$ is expressed as follows for all $n \geqq m \geqq 0$:

$$T(n,m) \leqq C3^{n/3} - Q(n) \equiv R(n), \tag{2}$$

where

$$Q(n) = q_1 n^2 + q_2 n + q_3,$$

with

$$q_1 = p_1/2 > 0,\quad q_2 = (9p_1 + p_2)/2 > 0,\quad q_3 = 27p_1/2 + 9p_2/4 + p_3/2 > 0,$$

and

$$C = \mathrm{Max}\{C_1,\ C_2,\ C_3\}$$

with $C_1 = 3\,q_2/\ln 3$, $C_2 = p_3 + q_3$, and C_3 being the maximum value of $3(1-2 \cdot 3^{-2/3})^{-1} \cdot Q(n-3)/3^{n/3}$. (Note that $Q(n-3)/3^{n/3}$ is finite for $1 \leqq n < \infty$ and it approaches 0 as n tends to infinity. Hence, C_3 is a finite constant.)

Here, $R(n) \equiv C3^{n/3} - Q(n)$ is monotone increasing with $R(n) \geqq p_3$ for all integers $n \geqq 0$.

Proof. First, by considering a continuous function $R(x)$ with x being a real number and $C \geqq C_1$, we can easily prove that $R(n)$ is monotone increasing for all integers $n \geqq 0$. Furthermore, $R(0) = C - Q(0) \geqq (p_3 + q_3) - Q(0) = p_3$, since $C \geqq C_2 = p_3 + q_3$. So $R(n) \geqq p_3$ for all integers $n \geqq 0$.

Now we prove that Eq.(2) holds by induction on n.

Basis. $n(=m) = 0$. We have $T(0,0) \leqq P(0) = p_3$ by the definition of $P(n)$. So, Eq.(2) holds for $n = m = 0$, since $R(0) \geqq p_3$.

Induction step. We assume that Eq.(2) holds for all integers n, m, $0 \leqq m \leqq n \leqq N$, and prove that it also holds for $0 \leqq m \leqq n = N+1$. Consider EXTEND(SUBG, CAND) when $|\,\mathrm{SUBG}\,| = n = N+1$, $|\,\mathrm{CAND}\,| = m$ $(1 \leqq m \leqq n = N+1)$, $|\,\mathrm{EXT}_u\,| = |\,\mathrm{CAND} - \Gamma(u)\,| = k \neq 0$ with $\mathrm{CAND} - \Gamma(u) = \{v_1, v_2, \cdots, v_k\}$ at the first entrance to statement 5. Then just as in LEMMA (i), we have

$$T_k(n,m) = T_k(|\,\mathrm{SUBG}\,|\,,\ |\,\mathrm{CAND}\,|\,)$$

$$\leqq \sum_{i=1}^{k} (T(|\,\mathrm{SUBG}_i\,|\,,\ |\,\mathrm{CAND}_i\,|\,) + P(n).$$

where $|\,\mathrm{SUBG}_i\,| \leqq n-1=N$ by LEMMA (ii) b). Then the induction hypothesis applies to have

$$\sum_{i=1}^{k} (T(|\,\mathrm{SUBG}_i\,|\,,\ |\,\mathrm{CAND}_i\,|\,) \leqq \sum_{i=1}^{k} R(|\,\mathrm{SUBG}_i\,|\,).$$

Since $R(n)$ is monotone increasing and $|\,\mathrm{SUBG}_i\,| \leqq n-k$, we have

$$\sum_{i=1}^{k} R(|\,\mathrm{SUBG}_i\,|\,) \leqq kR(n-k).$$

Combining these inequalities gives

(3) $$T_k(n,m) \leqq kR(n-k)+P(n)$$

$$\equiv kC3^{(n-k)/3} - kQ(n-k) + P(n)$$

$$= k3^{-k/3} \cdot C3^{n/3} - \{kQ(n-k) - P(n)\}.$$

In case $k=3$, we have

$$T_3(n,m) \leqq C3^{n/3} - \{3\,Q(n-3\,) - P(n)\}.$$

Now consider the other cases where $k \neq 3$ (with $k \geqq 1$), and we shall show that

(4) $$k3^{-k/3} \cdot C3^{n/3} - \{kQ(n-k) - P(n)\}$$

$$\leqq C3^{n/3} - \{3\,Q(n-3\,) - P(n)\}$$

for all integers $n \geqq 1$, provided $C \geqq C_3$. Modifying Eq.(4) gives

(5) $$\frac{3\,Q(n-3\,) - kQ(n-k)}{(1 - k3^{-k/3}) \cdot 3^{n/3}} \leqq C, \text{ where } k \neq 3.$$

Here, $kQ(n-k) \geqq 0$ for $1 \leqq k \leqq m \leqq n$, and we can easily show that $k3^{-k/3} \leqq 2 \cdot 3^{-2/3} < 1$ for all positive integers $k \neq 3$. So, for the left hand side of Eq.(5), we have

$$\frac{3\,Q(n-3\,) - kQ(n-k)}{(1 - k3^{-k/3}) \cdot 3^{n/3}} \leqq \frac{3\,Q(n-3\,)}{(1 - 2 \cdot 3^{-2/3}) \cdot 3^{n/3}} \leqq C_3.$$

Now we are in the case where $C \geqq C_3$, then Eq.(4) holds for all integers

$n \geq 1$ and $1 \leq k \leq m \leq n$. (Equality holds when $k=3$.) Combining Eqs.(3) and (4) gives

$$T_k(n,m) \leq C3^{n/3} - \{3\,Q(n-3) - P(n)\}$$
$$= C3^{n/3} - Q(n) \quad \text{(from the definition of } Q(n)\text{)}.$$

Substituting this inequality into Eq.(1) gives

$$T(n,m) \leq C3^{n/3} - Q(n).$$

Thus, Eq.(2) also holds for $n=N+1 \geq m \geq 0$. Therefore, Eq.(2) has been induced to hold for *all* integers $n \geq m \geq 0$. Hence the result. Q.E.D.

In particular,

$$T(n,n) \leq C3^{n/3} - Q(n).$$

Therefore, we conclude that the worst-case running time of the algorithm CLIQUES(G) is $O(3^{n/3})$ for an n-vertex graph G=(V,E). Note here that the original Bron and Kerbosch's algorithm outputs the entire clique itself in $O(n)$ time every time it is found. Thus their algorithm takes $O(n3^{n/3})$ time as a whole.

Acknowledgment. The authors wish to express their gratitude to Mikio Shindo, presently with NEC Corporation, for his contribution in an early stage of this work. They also would like to acknowledge useful discussions with Prof. Takao Nisizeki of Tohoku University.

REFERENCES

[1] A.V.AHO, J.E.HOPCROFT, AND J.D.ULLMAN, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974.

[2] E.A.AKKOYUNLU, The enumeration of maximal cliques of large graphs, SIAM J.Comput.,2(1973),pp.1-6.

[3] E.BALAS AND C.S.YU, Finding a maximum clique in an arbitrary graph, SIAM J.Comput.,15(1986),pp.1054-1068.

[4] C.BRON AND J.KERBOSCH, Algorithm 457: Finding all cliques of an undirected graph, Comm.ACM,16(1973),pp.575-577.

[5] N.CHIBA AND T.NISHIZEKI, Arboricity and subgraph listing algorithms, SIAM J.Comput.,14(1985),pp.210-223.

[6] L.GERHARDS AND W.LINDENBERG, Clique detection for nondirected graphs: Two new algorithms, Computing, 21(1979),pp.295-322.

[7] D.S.JOHNSON, M.YANNAKAKIS AND C.H.PAPADIMITRIOU, On generating all maximal independent sets, Inform.Process.Lett.,27(1988),pp.119-123.

[8] H.C.JOHNSTON, Cliques of a graph— Variations on the Bron-Kerbosch algorithm, Int.J.Comput. and Inform.Sci., 5(1976),pp.209-238.

[9] E.L.LAWLER, J.K.LENSTRA AND A.H.G.RINNOOY KAN, Generating all maximal independent sets: NP-hardness and polynomial-time algorithms, SIAM J.Comput.,9(1980),pp.558-565.

[10] J.W.MOON AND L.MOSER, On cliques in graphs, Israel J.Math., 3(1965)pp.23-28.

[11] G.D.MULLIGAN AND D.G.CORNEIL, Corrections to Bierstone's algorithm for generating cliques, J.ACM,19(1972),pp.244-247.

[12] R.E.OSTEEN, Clique detection algorithms based on line addition and line removal, SIAM J.Appl.Math.,26(1974),pp.126-135.

[13] E.M.REINGOLD, J.NIEVERGELT, AND N.DEO, Combinatorial Algorithms: Theory and Practice, Prentice-Hall, Englewood Cliffs,NJ, 1977.

[14] R.E.TARJAN AND A.E.TROJANOWSKI, Finding a maximum independent set, SIAM J.Comput.,6(1977), pp.537-546.

[15] S.TSUKIYAMA, M.IDE, H.ARIYOSHI, AND I.SHIRAKAWA, A new algorithm for generating all the maximal independent sets, SIAM J.Comput.,6(1977), pp.505-517.

Appendix.

LEMMA A.1. (For the first part of *Proof* on page 15) Consider the following continuous function $R(x)$ with x being a real number:

$$R(x) = C3^{x/3} - Q(x),$$

where

$Q(x) = q_1x^2 + q_2x + q_3$, with q_1, q_2, and q_3 being just as in THEOREM. If $C \geqq C_1 = 3q_2/\ln 3$, then $R(x)$ is *monotone increasing* for $x \geqq 0$.

Proof. Differentiating the function $R(x)$ gives

$$dR(x)/dx = \ln 3 \cdot C3^{x/3-1} - 2q_1x - q_2.$$

In addition,

$$\begin{aligned} d^2R(x)/dx^2 &= (\ln 3)^2 \cdot C3^{x/3-2} - 2q_1 \\ &\geqq (\ln 3)^2 \cdot C_1 3^{x/3-2} - 2q_1 \\ &= (\ln 3/6) \cdot (9p_1 + p_2)3^{x/3} - p_1 > 0, \text{ for } x \geqq 0, \end{aligned}$$

since $p_1 > 0$, $p_2 \geqq 0$. Hence, $dR(x)/dx$ is monotone increasing for $x \geqq 0$. Thus, for $x \geqq 0$,

$$\begin{aligned} dR(x)/dx &\geqq [dR(x)/dx]_{x=0} \\ &= \ln 3 \cdot C/3 - q_2 \geqq \ln 3 \cdot C_1/3 - q_2 = 0. \end{aligned}$$

Therefore, $R(x)$ is monotone increasing for $x \geqq 0$. Q.E.D.

LEMMA A.2. (For the denominator of Eq.(5) on page 16) Consider the following continuous function $f(x)$ with x being a real number:

$$f(x) = x3^{-x/3}.$$

Then $f(x)$ is monotone increaseing for $x < 3/\ln 3 = 2.7307\cdots$, $f(x)$ is monotone decreasing for $x > 3/\ln 3$, and $f(x) \leqq f(3/\ln 3)$ for all x. In addition, $f(2) = 0.9614\cdots$, and $f(4) = 0.9244\cdots$.

Proof. $df(x)/dx = 3^{-x/3}(3/\ln 3 - x)$. Then $df(x)/dx > 0$ for $x < 3/\ln 3$, $df(x)/dx < 0$ for $x > 3/\ln 3$, and $[df(x)/dx]_{x=3/\ln 3} = 0$. The latters are trivial. Q.E.D.