DETERMINING THE STABILITY NUMBER OF A GRAPH

by

V. Chvátal

STAN-CS-76-583
DECEMBER 1976

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

Determining the Stability Number of a Graph

V. Chvátal

Computer Science Department
Stanford University
Stanford, California 94305

Abstract.

We formalize certain rules for deriving upper bounds on the stability number of a graph. The resulting system is powerful enough to (i) encompass the algorithms of Tarjan's type and (ii) provide very short proofs on graphs for which the stability number equals the clique-covering number. However, our main result shows that for almost all graphs with a (sufficiently large) linear number of edges, proofs within our system must have at least exponential length.

This research was supported in part by National Science Foundation grant MCS 72-03752 A03 and by the Office of Naval Research contract N00014-76-C-0330 at Stanford University; and by National Research Council grant A9211 at University of Montréal. Reproduction in whole or in part is permitted for any purpose of the United States Government.

1. Introduction.

By a graph, we shall mean what is sometimes called a Michigan graph: one that is finite, undirected, without loops and multiple edges. A set S of vertices in a graph G is called independent or stable if no two vertices in S are adjacent; the largest cardinality a(G) of a stable set in G is called the stability number of G . Now, let G be a graph and let t be a positive integer such that

$$\alpha(G) \leq t \; ; \tag{1.1}$$

how laborious is it to verify a proof of (1.1)? Of course, this question has a direct bearing on the conjecture that P $\neq$ NP ; in particular, the celebrated theorem of Cook [2] suggests that it is extremely time-consuming to verify proofs of (1.1). We shall refrain from elaborating on this interesting point; instead, we direct the reader to [2], [14] and [1]. As for evaluating $\alpha(G)$, the best available algorithm is due to Tarjan and Trojanowski [10]: its running time on a graph of order n is $O(2^{n/3})$.

The framework of the present paper is quite modest: restricting the intuitive notion of a proof rather drastically, we shall study the resulting system of "recursive proofs". This system remains powerful enough to

(i) encompass a certain class of algorithms that includes the Tarjan-Trojanowski algorithm,

(ii) provide very short proofs of (1.1) for every graph G whose set of vertices can be covered by $\alpha(G)$ cliques.

Nevertheless, we shall show that there are valid inequalities (1.1) whose proofs must be excessively long. More explicitly, for every sufficiently large d there is a positive ε with the following property: for an overwhelming majority of all graphs G with n vertices and dn edges there are valid inequalities (1.1) whose recursive proofs must have length at least $(1+\varepsilon)^n$. (The assumption that the number of edges of G grows *linearly* with n is crucial: in fact, the conclusion fails as soon as d is allowed to grow beyond every bound. For details, see Proposition 4.1.)

At this moment, it may be worth pointing out two shortcomings that practitioners sometimes find in results on computational complexity: the worst case criterion and the asymptotic point of view. The first of these objections does not apply to our result at all but the second one certainly does: the numerical values of ε are very small. (One could improve on them by taking a little more care in the computations but even then they probably would not be very impressive.)

In Section 2, we point out those properties of random graphs which appear in the proof of the main result: looking at small subgraphs of G , and then extrapolating in a straightforward way, one would expect a(G) to be much larger than it actually is. In that sense, $\alpha(G)$ is very much a "global parameter". And it is precisely this global character which makes the proofs of (1.1) so long. In Section 3, we describe a certain class of crude algorithms for evaluating a(G) and then touch briefly upon the more sophisticated algorithm of Tarjan and Trojanowski. That section provides the motivation for the definition of a recursive proof presented in Section 4. The exponential that appears in our main result originates from an upper bound on the tail of the hypergeometric distribution; it finds its way into the theorem via a lemma on binary trees which we set aside in Section 5.

In the context of another NP-complete problem (namely, that of satisfiability of Boolean expressions), there are many results similar in spirit to ours; most of them can be found in [3]. In particular, the proof system investigated recently by Galil [ll] is very much like ours; however, the similarity does not extend beyond the superficial level.

2. Random Graphs.

In this section, we shall deal with graphs whose vertices are labeled as $v_1, v_2, \ldots \ldots$ Two such graphs may be distinct even if they are isomorphic; hence their total number is $2^{n(n-1)/2}$. If P is a property which a graph may or may not have then we shall denote by $t(P,n)$ the number of those graphs with n vertices which'do have the property, Finally, we shall say that almost all graphs have the property P if the ratio $t(P,n)/2^{n(n-1)/2}$ tends to one as n tends to infinity. A typical statement of this kind appears in the following lemma. The lemma itself seems to be a part of the graph-theoretical folklore. It appears at least implicitly in a 1947 paper by Erdős [5]; further refinements can be found in works of Matula [17], Grimmett and McDiarmid [12], Erdős and Bollobás [8] and perhaps others.

Lemma 2.1. Almost all graphs G of order n have the property that $a(G) < 2\log n/\log 2$.

Proof. Denote $2 \log n/\log 2$, rounded up to the nearest integer, by $k(n)$. Clearly, the number of those graphs of order n for which $\alpha \geq k$, divided by the number of all graphs of order n , does not exceed

$$\binom{n}{k} 2^{-k(k-1)/2} \tag{2.1}$$

By elementary estimations, (2.1) is at most

$$\left(\frac{en}{k} 2^{-(k-1)/2}\right)^k . \tag{2.2}$$

For all sufficiently large n , we have

$$\frac{en}{k} 2^{-(k-1),/2} = e2^{1/2}/k < .99$$

and so (2.1) tends to zero as n tends to infinity. □

In the theory of random graphs developed by Erdös and Rényi [8], [9], [10], one investigates graphs with n vertices and m edges. Clearly, the number of such graphs is

$$\binom{\binom{n}{2}}{m} \qquad (2.3)$$

We shall denote by $t(P,n,m)$ the number of those graphs with n vertices and m edges which have some property P . If m is a function of n such that each m(n) is a nonnegative integer not exceeding $n(n-1)/2$ and if the ratio of $t(P,n,m)$ to (2.3) tends to one as n tends to infinity then we shall say that almost all graphs with n vertices and m edges have the property P . The following lemma has been used by Erdös in [6] and elsewhere. (Throughout the paper, log denotes the natural logarithm.)

Lemma 2.2. If $m(n) \geq 16n$ for all sufficiently large n then almost all graphs G with n vertices and m edges have the property that

$$a(G) \quad \overset{n^{2}}{<} \frac{n}{m} \log \frac{m}{n} \quad . \qquad (2.4)$$

Proof. Denote the right-hand side of (2.4), rounded up to the nearest -integer, by k(n) ; note that $k(n) \to \infty$ as $n \to \infty$. Clearly, the number of those graphs with n vertices and m edges for which $\alpha > k$, divided by the number of all graphs with n vertices and m edges, does not exceed

$$\frac{\binom{n}{k}\binom{\binom{n}{2}-\binom{k}{2}}{m}}{\binom{\binom{n}{2}}{m}} \qquad (2.5)$$

By elementary estimations, (2.5) does not exceed

$$\left(\frac{en}{k}\right)^k\left(1-\frac{k(k-1)}{n(n-1)}\right)^m < \left(\frac{en}{k}\exp\left(-\frac{m(k-1)}{n(n-1)}\right)\right)^k .$$

In addition, we have

$$\frac{en}{k}\exp\left(-\frac{m(k-1)}{n(n-1)}\right) < \frac{em}{n\log(m/n)}\exp\left(-\log\frac{m}{n}+\frac{m}{n^2}\right) .$$

Since the last quantity becomes smaller than .99 for all sufficiently large n, we conclude that (2.5) tends to zero as n tends to infinity. □

Next, let us digress a little. When m, n, s are nonnegative integers such that $m \leq n$ and when t is a positive real number, we shall set $p = m/n$, denote by Σ^* the summation over all integers $j > s(p+t)$ and define

$$B(m,n,s,t) = \Sigma^*\binom{s}{j}\left(\frac{m}{n}\right)^j\left(\frac{n-m}{n}\right)^{s-j} ,$$

$$H(m,n,s,t) = \Sigma^*\frac{\binom{m}{j}\binom{n-m}{s-j}}{\binom{n}{s}} .$$

Thus B is the familiar "tail of the binomial distribution" and H is the "tail of the hypergeometric distribution", The well-known interpretation of these quantities goes as follows. Imagine a barrel containing n apples, exactly m of which are rotten; take a random sample of s apples. Technically, the sampling can be done in at least two ways. We might pick and examine the apples one by one, each time throwing the apple back into the barrel before reaching in again: this is called sampling with replacement. Or we might just grab the s apples at the same time: that is called sampling without replacement. Whichever method we use, we should expect about ps rotten apples in the sample. The quantities B and H give the probability that at least $(p+t)s$ rotten apples will appear in the sample with and without replacement, respectively.

An elegant argument (apparently due to S. N. Bernstein) shows that

$$B(m,n, s,t) \leq \left(\left(\frac{p}{p+t}\right)^{p+t}\left(\frac{1-p}{1-p-t}\right)^{1-p-t}\right)^{s} .$$

A similar bound for H seems to be far more difficult to establish. A special case of a theorem of Hoeffding ([13], Theorem 4) states that

$$H(m,n, s,t) \leq \left(\left(\frac{p}{p+t}\right)^{p+t}\left(\frac{1-p}{1-p-t}\right)^{1-p-t}\right)^{s} . \qquad (2.6)$$

It is a routine matter to convert (2.6) into weaker but more tractable bounds; we are about to do that for $t = p$,

Lemma 2.3. $H(m,n,s,m/n) \leq e^{-ms/4n}$,

Proof. If $p > 1/2$ then the left-hand side vanishes. If $p \leq 1/2$ then (2.6) implies

$$\frac{1}{s}\log H(m,n,s,p) \leq 2p\log\frac{1}{2} + (1-2p)\log\left(1+\frac{p}{1-2p}\right)$$

$$\leq 2p\log\frac{1}{2} + p < -p/4$$

which is the desired conclusion. □

Upper bounds on H are useful in proving statements about random graphs, such as the following one.

Lemma 2.4. Almost all graphs G with n vertices and m edges have the following property: every subgraph of G induced by s vertices such that

$$s > \frac{4n^2}{m} \log \frac{m}{n} \qquad (2.7)$$

has fewer than $2ms^2/n^2$ edges.

Proof. Clearly, the number of those graphs which do not have the property, divided by the number of all graphs with n vertices and m edges, does not exceed

$$\sum_s \binom{n}{s} H\left(\binom{s}{2}, \binom{n}{2}, m, \binom{s}{2}/\binom{n}{2} \right) . \qquad (2.8)$$

By Lemma 2.3, this quantity does not exceed

$$\sum_s \binom{n}{s} \exp\left(- \frac{ms(s-1)}{4n(n-1)} \right) < \sum_s \left(\frac{en}{s} \exp\left(- \frac{m(s-1)}{4n^2} \right) \right)^s .$$

By (2.7), we have

$$\frac{en}{s} \exp\left(- \frac{m(s-1)}{4n^2} \right) \leq \frac{em}{4n \log(m/n)} \exp\left(\frac{m}{4n^2} - \log \frac{m}{n} \right) < .99 .$$

Hence (2.8), being bounded from above by

$$\sum_s (.99)^s < 100(.99)^{4n^2 \log(m/n)/m}$$

tends to zero as n tends to infinity. □

3. Algorithms.

In this section, we shall first describe a class of crude algorithms for finding a largest stable set in a graph and point out that by the use of appropriate data structures, the running time of these algorithms can be cut down considerably. Then we shall briefly outline a class of more sophisticated algorithms which we shall call Tarjan algorithms,

Let us suppose that, given a graph $G = (V,E)$ and a subset S of V, we wish to find a largest stable subset A of S. We may begin by choosing a vertex $v \in S$; the desired set A either does not contain v or it does contain v . In the first case, A is a largest stable subset of the set $S_1 = s-(v)$; in the second case, $A-\{v\}$ is the largest stable subset of the set S_2 obtained from S by deleting v with all of its neighbors in S . We shall denote S_1 by $S-v$ and S_2 by $S*v$; with this notation, we have

$$\alpha(S) = \max(a(S-v) , 1+\alpha(S*v)) .$$

Thus we have reduced the original problem into two similar, but smaller, subproblems: one for $S-v$ and the other for $S*v$.

Now, an algorithm for finding a largest stable set in G suggests itself: begin with $S = V$, do what we have just done and then simply iterate away. One may visualize a binary tree with nodes labeled by subsets of V . The root is labeled by V itself; if a node is labeled by a nonempty set S then its left son is labeled by $S-v$ and its right son is labeled by $S*v$ for some $v \in S$. If G has n vertices altogether and if each vertex has fewer than d neighbors then the tree will have at least $2^{n/d}$ nodes. Of course, that does not mean that the algorithm will create at least $2^{n/d}$ subproblems: different nodes of the tree may have the same label. (To take an extreme example, note that all the leaves of the tree will be labeled by $\emptyset$.)

We shall describe a possible implementation of the algorithm. For definiteness, let us assume that we have a fixed "choice function" f which assigns to each nonempty subset S of V a vertex $f(S) \in S$. Such a function gives rise to an algorithm which we shall call the *f-driven algorithm,*

In its first phase, the algorithm creates a list of certain subsets of v, which will be called *subproblems.* It will be convenient to keep the list ordered, with larger subproblems preceding the smaller ones; within each group of subproblems of the same size, the order may be lexicographic. At each moment, we shall have a partial list of subproblems, with a pointer at one of them. At the very beginning, V will be the only subproblem on the list; the first phase will terminate as soon as the pointer gets to $\emptyset$. When the pointer is at a nonempty set S, we define $S_1 = S-f(S)$ and $S_2 = S*f(S)$. Then we add S_1 and S_2 on the list (unless they are already present), shift the pointer to the successor of S and iterate.

In the second phase, we pass through the list in a reverse order (from $\emptyset$ to V) and evaluate a(G) for each subproblem S. To begin with, we have $\alpha(\emptyset) = 0$; for each nonempty subproblem S, we have $a(S) = \max(\alpha(S_1), 1+\alpha(S_2))$.

In the third phase, we shall find a largest stable set A in G. To begin with, let us set $A = \emptyset$ and $S = V$. With each iteration, the set S will shrink; when it will become empty, A will be the desired largest stable set in G. Each iteration is simple. If $\alpha(S) = \alpha(S_1)$ then we replace S by S_1; otherwise $a(S) = 1+\alpha(S_2)$ in which case we add f(S) to A and replace S by S_2.

It is crucial to use the appropriate data structures when implementing the first phase. Trivially, the number of subproblems on the list never exceeds 2^n . If we implement the list as a balanced tree (see [15] or [1]) then each of the look-ups and insertions can be handled within a number of set-comparisons proportional to n . If each $f(S)$ can be evaluated within a steps and if the total number of subproblems is b then the running time of the algorithm is $O(abn^2)$. For at least a few choices of f that come to mind, a is polynomial in n . In that case, b threatens to be the decisive factor in the upper bound.

Needless to say, the number of subproblems depends on the choice function f ; for most functions f , that number seems difficult to estimate. To simplify the situation, we shall restrict ourselves to very special choice functions: when the vertices of G are ordered as $v_1, v_2, \ldots, v_n$, the function f chooses that vertex of S which has the smallest subscript. The resulting f-driven algorithm will be called an order-driven algorithm.

The following proposition and its corollaries (Propositions 3.2 -3.5) are due to Szemerédi. In its statement, N(k) denotes the number of stable subsets of $\{v_1, v_2, \ldots, v_k\}$. Here and later on, we shall find it convenient to denote by S*T the subset of S resulting when all the vertices in T and all their neighbors are deleted.

Proposition 3.1. The order-driven algorithm applied to a graph with vertices $v_1, v_2, \ldots, v_n$ creates at most

$$1 + \sum_{k=0}^{n-1} \min(N(k), 2^{n-k-1})$$

subproblems.

Proof. For each subproblem S , let k be the largest subscript such that $\{v_1, v_2, \ldots, v_k\} \cap S = \emptyset$. It is not difficult to see that

$S = \{v_{k+1}, v_{k+2},$ ● ♏🖂📫❖▮☸🖂👌

for some stable subset B of $\{v_1, v_2, \ldots, v_k\}$. Hence for each fixed k , there are at most $N(k)$ subproblems S . In addition, if $k < n$ then there are only 2^{n-k-1} subsets S of $\{v_{k+1}, \ldots, v_n\}$ such that $v_{k+1} \in S$. □

Proposition 3.2. The order-driven algorithm applied to a graph G of order n such that $a(G) < n/2$ creates at most

$$n^2 \binom{n}{\alpha(G)}$$

subproblems."

Proof. Trivially, we have

$$N(k) \leq \sum_{i=0}^{\alpha(G)} \binom{k}{i} \leq n \binom{n}{\alpha(G)}$$

for each k ; the rest follows from Proposition 3.1. □

Proposition 3.3. For almost all graphs G of order n , the order-driven algorithm creates at most

$$n^{2(1 + \log n/\log 2)}$$

subproblems.

(The proof follows immediately from Proposition 3.2 and Lemma 2.1.)

Proposition 3.4. If $m(n)/n \to \infty$ then almost all graphs G with n vertices and m edges have the following property: for every constant $c > 1$, the order-driven algorithm on G creates $o(c^n)$ subproblems.

Proof. By Lemma 2.2, we have $\alpha(G) = o(n)$ for almost all graphs with n vertices and m edges; the rest follows from Proposition 3.2. □

Proposition 3.5. For every graph with n vertices, the order-driven algorithm creates at most $3 \cdot 2^{(n-1)/2}-1$ subproblems.

Proof. We have

$$\sum_{k=0}^{n-1} \min(N(k), 2^{n-k-1}) \leq \sum_{k=0}^{n-1} \min(2^k, 2^{n-k-1}) < 3 \cdot 2^{(n-1)/2}-2 \ ;$$

the rest follows from Proposition 3.1. □

Note that the bound of Proposition 3.5 is sharp: it is attained by the graph with vertices $v_1, v_2, \ldots, v_{2m+1}$ and edges $v_1 v_{2m+1}, v_2 v_{2m}, \ldots, v_m v_{m+2}$. Nevertheless, if we can choose the ordering of the vertices then the bound can be improved.

Proposition 3.6. Every graph with n vertices can be ordered in such a way that the order-driven algorithm creates $O(n 2^{3n/7})$ subproblems.

Proof. We shall first describe the ordering and then we shall show that it has the desired property. Suppose that we have already constructed the initial segment $v_1, v_2, \ldots, v_{4t}$ for some $t \geq 0$. If the graph $H = G-\{v_1, v_2, \ldots, v_{4t}\}$ contains a path $w_1 w_2 w_3 w_4$ then we set $v_{4t+i} = w_i$ for $1 \leq i \leq 4$ and iterate. Otherwise each component of H is a star or a triangle. In that case, we denote $4t$ by m and enumerate the vertices of H as $v_{m+1}, v_{m+2}, \ldots, v_n$ in such a way that

(i) the vertices of each component of order j are enumerated as $v_{i+1}, v_{i+2}, \ldots, v_{i+j}$ for some i ,

(ii) if that component is a star then v_{i+1} is its center.

It is not difficult to verify that $N(k) < 2^{(3k+1)/4}$ for each $k = 1,2,\dots,m$. If $m \geq 4n/7$ then

$$\sum_{k=0}^{n-1} \min(N(k), 2^{n-k-1}) = O(n2^{3n/7}) .$$

If $m < 4n/7$ then we resort to another argument: note that each subproblem has the form $\{v_{k+1}, v_{k+2}, \dots, v_n\}*B$ such that $1 \leq k \leq n$ and B is a stable subset of $\{v_1, v_2, \dots, v_m\}$. Since $N(m) \leq 2^{(3m+1)/4}$, the total number of subproblems is $O(n2^{3n/7})$. □

It is not unlikely that the bound of Proposition 3.6 can be improved. Let us call a number c admissible if every graph with n vertices can be ordered in such a way that the order-driven algorithm creates $O(c^n)$ subproblems; let c_0 denote the infimum of all admissible c . By Proposition 3.6, we have $c_0 \leq 2^{3/7}$; on the other hand, the main result of this paper implies that $c_0 > 1$. What is the exact value of c_0 ? Similar questions apply to the wider class of f-driven algorithms and to the even wider class of Tarjan algorithms which we are about to outline.

As pointed out at the beginning of this section, every f-driven algorithm applied to a graph gives rise to a binary tree whose nodes are labeled by subproblems: if a node x is labeled by a nonempty subproblem S then the left son of x is labeled by S-v and the right son of x is labeled by $S*v$ for some $v \in S$. Elimination of duplications on the list of subproblems amounts to pruning the tree: we simply omit nodes whose presence would result in duplicated labels. The idea of Tarjan [19] leads to pruning of a different kind. In an f-driven algorithm, each subproblem S is generated in the form $(V-A)*B$ such that B is a stable set; eventually, such a subproblem yields a stable set of size $\alpha(S)+|B|$. If another subproblem S_1 is generated in the form $(V-A_1)*B_1$ such that

$S_1 \subseteq S$ and $|B_1| \leq |B|$ then S_1 can be discarded: in a sense, S_1 is <u>dominated</u> by S. In terms of the binary tree, we might index each node x by the number r of right-hand turns on the path from the root to x; a branch rooted at a node x_1 (labeled by S_1 and indexed by r_1) may be pruned off whenever there is another node x (labeled by S and indexed by r) such that $S_1 \subseteq S$ and $r_1 \leq r$.

Now we have arrived at two kinds of pruning: these might be called "duplication pruning" and "dominance pruning", the former being (in a sense) a special case of the latter. An f-driven algorithm with the option of using both duplication pruning and dominance pruning to eliminate subproblems will be called a <u>Tarjan algorithm</u>. Of course, <u>systematic use</u> of dominance pruning may shorten the list of subproblems quite considerably. In terms of running time, however, the means could defeat the purpose: in general, it may take a very long time to decide whether the subproblem that has been just created is dominated by at least one of the subproblems already on the list. Thus it may be wise to pass up the option of (possible) dominance pruning in most cases, resorting to it only in those simple situations where the dominating subproblem is almost staring at us. Such a strategy led Tarjan [19] to an algorithm whose worst-case running time for a graph with n vertices is $O(1.286^n)$. Later on, Tarjan and Trojanowski [20] designed an improved version of that algorithm with running time $o(2^{n/3})$. It may be worth pointing out that these upper bounds come out of rather rudimentary applications of dominance pruning only: the argument does not take duplication pruning into account at all. Thus, it is not inconceivable that (with the subproblems kept in a balanced tree, so that duplication pruning is easy to implement) the worst-case running time of the Tarjan-Trojanowski algorithm is even better than

$O(2^{n/3})$. Nevertheless, the main result of this paper implies the existence of a constant c > 1 and arbitrarily large graphs G with n vertices such that every Tarjan algorithm applied to G must create at least c^n different subproblems;' (In fact, almost all graphs with n vertices and dn edges have this property as long as d is sufficiently large.)

One more comment: from the practical point of view, the Tarjan - Trojanowski algorithm might be preferable even to (hypothetical) f-driven algorithms creating c^n subproblems for c fairly close to 1 . The point is that the space requirements of such algorithms would be roughly nc^n whereas the space required by the Tarjan - Trojanowski algorithm is only polynomial in n .

4.1 Recursive Proofs.

For the moment, let us deal with an arbitrary but fixed graph $G = (V,E)$. By a statement, we shall mean an ordered pair (S,t) such that S is a subset of V and t is a nonnegative integer. (Such a statement is to be interpreted as the inequality $a(S) \leq t$ which, of course, may be true or false.) By a recursive proof of a statement (S,t) over G , we shall mean a sequence of statements

$$(S_i,t_i) \ , \qquad i = 0,1,\ldots,m \tag{4.1}$$

such that $(S_0,t_0) = \emptyset$, $(S_m,t_m) = (S,t)$ and such that each statement (S_k, t_k) with $k \geq 1$ can be derived from the previous statements (S_i, t_i) , $0 \leq i < k$, by at least one of the following two rules.

1. The dichotomy rule: from $(S\text{-}v,t_i)$ and $(S*v,t_j)$ we can derive $(S\, , \max(t_i,1+t_j))$.
2. The monotone rule: from (S,t) we can derive (S',t') whenever $S' c\ S$ and $t' > t$.

Clearly, if (4.1) is a recursive proof of (S,t) then $\alpha(S_i) \leq t_1$ for every i ; in particular, $a(s) \leq t$. Conversely, if $a(S) < t$ then there is a recursive proof of (S,t) . In order to see that, consider the family F of subproblems created by some f-driven algorithm that has just found a largest stable subset of S . Enumerate all the ordered pairs $(S^*,\alpha(S^*))$ with $S^* \epsilon F$ as (4.1) in such a way that $|S_i| \leq |S_{i+1}|$ for every i . Clearly, the resulting sequence constitutes a recursive proof of $(S,\alpha(S)\ )$; if $t > \alpha(S)$ then one additional application of the monotone rule completes a recursive proof of (S,t) .

It will be convenient to define the length of (4.1) as m . Now, Propositions 3.1 - 3.6 yield direct corollaries in terms of recursive proofs. We shall state explicitly only one of them, Proposition 4.1. If $c > 1$ and if $m(n)/n \to \infty$ then, for almost all graphs $G = (V,E)$ with n vertices and m edges, there are recursive proofs of $(V,\alpha(G))$ of length $o(c^n)$.

In addition, every Tarjan algorithm applied to $G = (V,E)$ yields a recursive proof of $(V,\alpha(G))$. Hence for every graph $G = (V,E)$ of order n , there is a recursive proof of $(V,\alpha(G))$ of length $O(2^{n/3})$.

Now, we shall show that for a certain class of graphs $G = (V,E)$, there exist very short recursive proofs of $(V,\alpha(G))$. This class consists of all those graphs G for which $a(G)$ equals $Q(G)$, the smallest number of cliques whose union is V . (Trivially, we have $\alpha(G) < Q(G)$ for every graph G .) It may be instructive to split the argument into three easy propositions.

Proposition 4.2. If $G = (V,E)$ is a complete graph of order n then there is a recursive proof of $(V,1)$ whose length is n .

Proof. Enumerating the vertices of G as $v_1,v_2,\ldots,v_n$, define $S_i = \{v_1,v_2,\ldots,v_i\}$. Trivially, the sequence $(\emptyset,0),(S_1,1),\ldots,(S_n,1)$ constitutes a recursive proof. □

Proposition 4.3. Let $G_1 = (V_1,E_1)$ and $G_2 = (V_2,E_2)$ be graphs such that $V_1 \cap V_2 = \emptyset$; let $G_1 \cup G_2$ denote the graph $(V_1 \cup V_2 , E_1 \cup E_2)$. If there are recursive proofs of $(V_j,\alpha(G_j))$ of length m_j for each $j = 1,2$ then there is a recursive proof of $(V_1 \cup V_2 , \alpha(G_1)+\alpha(G_2))$ whose length does not exceed m_1+m_2 .

Proof. If (S_i,t_i) with $i = 0,1,\ldots,m$ is a recursive proof of $(V_2,\alpha(G_2))$ then a recursive proof of $(V_1,\alpha(G_1))$, followed by the sequence

$$(V_1 \cup S_i, \alpha(G_1)+t_i) \quad , \quad i = 1,2,\ldots,m_2$$

constitutes a recursive proof of $(V_1 \cup V_2, \alpha(G_1)+\alpha(G_2))$. □

Proposition 4.4. Let F be a subgraph of G and let (4.1) be a recursive proof over F . Then there is a recursive proof of (S_m, t_m) over G whose length does not exceed 2m ,

Proof. We shall create the desired proof over G from (4.1) by inserting a new statement immediately before each (S_k, t_k) that has been obtained from the previous statements by the dichotomy rule. For every such (S_k, t_k) , there are subscripts i, j and a vertex $v \in S_k$ such that $i < k$, $j < k$, $t_k = \max(t_i, 1+t_j)$ and $S_i = S_k - v$, $S._j = S_k * v$ in F . The statement to be inserted immediately before (S_k, t_k) is (Sk, t_k-1) such that $S'_k = S_k * v$ in G , Clearly, (Sk, t_k-1) follows from (S_j, t_j) by the monotone rule whereas (S_k, t_k) follows from (S_i, t_i) and (S'_k, t_k-1) by the dichotomy rule. Cl

Proposition 4.5. For every graph $G = (V,E)$ of order n there is a recursive proof of (V, Q(G)) whose length does not exceed 2n .

Proof. Consider the subgraph F of G consisting of $\theta(G)$ cliques whose union equals V . By Proposition 4.2 and by repeated applications of Proposition 4.3, there is a recursive proof of (V, 0(G)) over F whose length equals n . The rest follows from Proposition 4.4. □

We shall close this section with another easy observation which will be handy later. The proof can be left to the reader.

Proposition 4.6. If (4.1) is a recursive proof over $G = (V,E)$ and if $W \subset V$ then

$$(S_i \cap W, t_i) \quad , \qquad i = 0,1,\ldots, m$$

is a recursive proof over the subgraph of G induced by W .

5. A Lemma on Binary Trees.

Let a, b, r, s be nonnegative integers, A binary tree whose nodes are colored red and blue will be called (a,b,r,s) -constrained if, along each path from the root to a leaf,

(i) exactly a nodes are followed by their left sons and exactly b nodes are followed by their right sons,

(ii) at most r nodes are red,

(iii) at least s red nodes are followed by their right sons.

If, for some choice of integers a, b, r and s, there is at least one (a,b,r,s) -constrained tree then we denote by $f(a,b,r,s)$ the largest possible number of-leaves in such a tree; otherwise we set $f(a,b,r,s) = 0$, Trivially, we have

$$f(a,b,r,s) \le \binom{a+b}{b}$$

and

$$f(a,b,r,s) = 0 \quad \text{whenever} \quad s>b \quad \text{or} \quad s > r .$$

The purpose of this section is to derive the following upper bound on $f(a,b,r,s)$.

-Lemma 5.1. If $s \ge 2br/(a+b-1)$, $s \ge r-a+1$ and $s \ge 1$ then

$$f(a,b,r,s) \le \binom{a+b}{b} \frac{ab}{a+b} e^{-br/4(a+b)} .$$

First of all, we shall establish a simple recursive bound.

Fact 5.2. $$f(a,b,r,s) \le \max \begin{cases} f(a-1,b,r,s) + f(a,b-1,r,s) \\ f(a-1,b,r-1,s) + f(a,b-1,r-1,s-1) \end{cases} .$$

Proof. Let T be an (a,b,r,s) -constrained tree. If its root is blue then the left sub-tree is either empty or $(a-1,b,r,s)$ -constrained and the right sub-tree is either empty or $(a,b-1,r,s)$ -constrained. If the root is red then the left subtree of T is either empty or $(a-1,b,r-1,s)$ -constrained and the right sub-tree of T is either empty or $(a,b-1,r-1,s-1)$ -constrained. Hence the desired conclusion. □

Next, for every choice of nonnegative integers a , b , r , s such that

$$s < b \quad , \quad s < r \quad , \quad s \geq r-a+1$$

we define

$$F(a,b,r,s) = \sum_{i=0}^{b-s} \binom{r+i}{s+i} \binom{a+b-r-1-i}{b-s-i} \quad .$$

It is easy to verify that

$$F(a,b,r,0) = \binom{a+b}{b} \ ,$$

$$F(a,b,r,b) = \binom{r}{b} \ ,$$

$$F(a,b,r,r) = \binom{a+b-r}{a} \ ,$$

$$F(a,b,r,r-a+1) = \binom{a+b}{a} - \binom{r}{a} \ ,$$

$$F(a-1,b,r,s) + F(a,b-1,r,s) = F(a,b,r,s) \ ,$$

$$F(a-1,b,r-1,s) + F(a,b-1,r-1,s-1) = F(a,b,r,s)$$

whenever the left-hand side terms are defined.

Fact 5.3. We have $f(a,b,r,s) \leq F(a,b,r,s)$ whenever the right-hand side is defined.

This inequality can be proved by induction on a+b in a straightforward way; we omit the tedious details. It is not unlikely that there is a direct combinatorial proof of Fact 5.3. Furthermore, it is not difficult to show that $f(a,b,r,s) = F(a,b,r,s)$ whenever the right-hand side is defined; however, that is irrelevant for our purpose.

Proof of Lemma 5.1. Wemayassume $s \le b$ and $s \le r$ for otherwise the left-hand side vanishes. Then, by Fact 5.3,

$$f(a,b,r,s) \le F(a,b,r,s) \ .$$

Since $r \ge s \ge 2br/(a+b-1)$, we have $2b/(a+b-1) \le 1$ and

$$s+i \ge 2b \frac{r+i}{a+b-1}$$

for every nonnegative i , Hence, with the notation of Section 2,

$$\binom{r+i}{s+i}\binom{a+b-r-l-i}{b-s-i} \le H(r+i,a+b-1,b,(r+i)/(a+b-1))\cdot\binom{a+b-1}{b} \ .$$

By Lemma 2.3, we h&e

$$f(a,b,r,s) \le \ .\sum_{i=0}^{b-s} \binom{a+b}{b} e\frac{a}{a+b}^{\;-br/4(a+b)}$$

which implies the desired result.

6. The Main Result.

A graph G of order n will be called (d, ε) -sparse if

(i) every vertex of G has degree less than d ,

(ii) every subgraph of G induced by m vertices such that $m \geq \varepsilon n$ has fewer than dm^2/n edges.

Theorem 6.1. Let n , t be positive integers and let d, ε be positive reals such that

$$n < 10td \ ,$$

$$n \geq 500t^{2/3} d \ ,$$

$$n \geq 100t^{3/4} d^{3/4} \ ,$$

$$n \geq 2000t \ ,$$

$$\varepsilon < n^2/1810t^2 d^2 \ ;$$

let G = (V,E) be a (d,ε) -sparse graph of order n , Then every recursive proof of (V,t) has length at least

$$\frac{90d}{n} \exp \frac{n}{20d^2} \tag{6.1}$$

Proof, We shall set

$$a = \lfloor n^3/45000t^2d^2 \rfloor \ , \quad b = \lfloor n^2/900td^2 \rfloor$$

. and show that every recursive proof of (V,t) has length at least

$$\frac{a+b}{ab} \exp \frac{b^2}{a+b} \ . \tag{6.2}$$

The reader may easily verify that $a \geq b \geq 200$ and so

$$a > \frac{200}{201} \cdot \frac{n^3}{45\ 000t^2d^2} \ , \quad b > \frac{200}{201} \cdot \frac{n^2}{900td^2} \ .$$

Then it follows that (6.2) is at least (6.1).

Let us outline our strategy. With each recursive proof of (V,t) , we shall associate a binary tree T whose nodes will be labeled by statements from the proof, The assignment of labels to nodes will not be one-to-one (to take an extreme example, all the leaves of T will be labeled by $\emptyset$) and so the number of nodes of T may be much greater than the length of the proof. We shall find a node z with a certain convenient property and then we shall construct a new binary tree T^* . Even though T^* will not be a subtree of T in a strict sense, its nodes will come from T ; in particular, z will be the root of T* . Finally, we shall show that within the set N of leaves of T* , no label is repeated too often. More precisely, for each subset S of V we shall define

$$N(S) = \{x \in N: \ x \text{ is labeled by } (S,t') \text{ for some } t'\}$$

and prove that

$$|N(S)| \leq |N| \cdot \frac{ab}{a+b} \exp\left(- \frac{b^2}{a+b} \right) \quad . \tag{6.3}$$

Since N will be nonempty, (6.3) will imply the desired result: indeed, the number of those sets S for which $N(S) \neq \emptyset$ must be at least (6.2).

Before going into the details, the reader may welcome a preview of the idea behind the proof of (6.3), however vague such a preview may have to be. Let (W,t^*) be the statement that labels z . In the absence of the monotone rule, the tree T^* is constructed in such a way that every subproblem S labeling a leaf of T* is obtained from W by simply deleting a vertices and by deleting b vertices with their neighbors. If we had our way, the subgraph H induced by W-S would consist of a isolated vertices and b disjoint stars: in that case, we could reconstruct the two sets of vertices, proving that $|N(S)| = 1$. Actually, we shall be content even if things are not all that clear-cut, as long as we can *approximately* reconstruct the two sets. That will be the case as long as H is reasonably large. (If H is large then *most* of the b vertices must have large degrees. At the same time, the second defining property of a (d,ε) -sparse graph implies that the average degree in H is rather small. Hence the vertices of large degrees are quite conspicuous.) In order to *guarantee* that H will be large, we have to choose z appropriately. In general, the rules for constructing T* are designed to neutralize the desultory effects of the monotone rule. Now that the poor reader is properly confused, we can proceed to the details.

Constructing T , we shall find it convenient to call certain statements in the proof *eligible*: a statement will be called eligible if it is $(\emptyset,0)$ or if it follows from some two earlier statements by the dichotomy rule. Only the eligible statements, with a possible exception of (V,t) , will be used to label the nodes of T . The construction of T is recursive; the root of T is labeled by (V,t) . Suppose that we have constructed a node x labeled by a statement (S_k,t_k) and having no sons at this moment. If $(S_k,t_k) = (\emptyset,0)$ then x will be a leaf of T . Otherwise there are eligible statements (S_i,t_i) , (S_j,t_j) and a vertex $v \in S_k$ such that

$$i,j < k \ , \quad S_i \supseteq S_k - v \ , \quad S_j \supseteq S_k * v \ , \quad t_k \geq \max(t_i \ , 1+t_j) \ .$$

In that case, we shall create both sons of x , label the left one by (S_i,t_i) and label the right one by (S_j,t_j) . For further reference, we shall set $S(x) = S_k$, $t(x) = t_k$ and" $v(x) = v$. It will be useful to note that

$$t_i \leq t_k \quad \text{and} \quad t_j \leq t_k-1 \ . \tag{6.4}$$

Next, we shall find the special node z . A node y will be called a descendant of a node x if there is a path $x_0,x_1,\ldots,x_k$ such that $x = x_0$, $y = x_k$ and each x_{i+1} is a son of x_i . If, in addition, exactly b nodes x_i are followed by their right sons x_{i+1} then Y will be called a b-descendant of x . Repeated applications of (6.4) show that

$$\text{if } y \text{ is a b-descendant of } x \text{ then } t(y) \leq t(x)-b \ . \tag{6.5}$$

We claim that

$$\left.\begin{array}{l}\text{there is a node } z \text{ such that } S(z) > n/2 \text{ and such that} \\ |S(y)| < |S(z)| - bn/2t \text{ for every b-descendant } y \text{ of } z \ ,\end{array}\right\} \tag{6.6}$$

A node with this property can be found by constructing a certain sequence $y_0,y_1,\ldots$ of nodes of T such that y_0 is the root of T , If the most recently constructed y_i has a b-descendant y such that $|S(y)| \geq |S(y_i)| - bn/2t$ then set $y_{i+1} = y$; otherwise stop. By (6.5) and by the construction of the sequence, we have

$$|S(y_i)| \geq n(1-bi/2t) \ , \quad t(y_i) \leq t-bi$$

for every i . Since $t(y_i) \geq 0$, we must have $i \leq t/b$ and so $|S(y_i)| \geq n/2$ for every i . In particular, the very last y_i in the sequence has the properties required of z . We shall denote S(z) by W .

With each descendant x of z , we shall associate two subsets $A(x)$, $B(x)$ of' V : considering the path $x_0, x_1, \ldots, x_k$ from $x_0 = z$ to $x_k = x$ we shall define

$$A(x) = \{v(x_i) : 0 \le i < k \text{ and } x_{i+1} \text{ is the left son of } x_i\} ,$$

$$B(x) = \{v(x_i) : 0 < i < k \text{ and } x_{i+1} \text{ is the right son of } x_i\} .$$

Clearly, we have

$$|S(x) \cap W| \ge |W| - |(A(x) - B(x)) \cap W| - \sum_{v \in B(x)} (1 + d(v)) \qquad (6.7)$$

for every descendant x of z , In particular,

$$\text{if } |(A(x) - B(x)) \cap W| \le a \text{ and } |B(x)| \le b \text{ then } S(x) \ne \emptyset : \qquad (6.8)$$

just observe that

$$a + b(1+d) \le 2a + bd < 3bd < n/2 .$$

Before proceeding to construct T^* , we shall associate a node x^* with each descendant x of z such that

$$|(A(x) - B(x)) \cap W| \le a \text{ and } |B(x)| \le b .$$

Consider the path $x_0, x_1, \ldots, x_k$ such that $x = x_0$, x_k is a leaf of T and each x_{i+1} is the left son of x_i . Note that $B(x_i) = B(x)$ for every i . There must be at least one i such that $0 < i < k$ and

$$v(x_i) \in W , \quad v(x_i) \notin A(x) \cup B(x) \qquad (6.9)$$

for otherwise

$$(A(x_k) - B(x_k)) \cap W = (A(x) - B(x)) \cap W \quad \text{and} \quad B(x_k) = B(x)$$

but $S(x_k) = \emptyset$ contradicting (6.8). We shall denote the first x_i satisfying (6.9) by x^* ; note that

$$(A(x^*) - B(x^*)) \cap W = (A(x) - B(x)) \cap W .$$

At last, we are ready to construct T^* . Each of its nodes x will come from T and satisfy

$$v(x) \in W \ , \ v(x) \notin A(x) \cup B(x) \ ,$$

$$|(A(x) - B(x)) \cap W| \leq a \ , \quad |B(x)| \leq b \ , \ B(x) \subseteq W \ ,$$

The construction of T^* is recursive; therootof T^* is z. Suppose that we have already constructed some node x of T^* ; let x_L denote the left son of x in T and let x_R denote the right son of x in T . If $|(A(x)-B(x)) \cap W| = a$ then x will have no left son in T^* ; otherwise we shall make x_L^* the left son of x in T^* . If $|B(x)| = b$ then x will have no right son in T^* ; otherwise we shall make x_R^* the right son of x in T^* . It will be useful to make note of the following property of T^* :

along each path from the root to a leaf,
exactly a nodes are followed by their left sons,
exactly b nodes are followed by their right sons, (6.10)
and these $a+b$ nodes x give rise to distinct vertices $v(x)$.

Finally, we shall prove the inequality (6.3). Without loss of generality, we may assume that $N(S) \neq \emptyset$ and so $S = S(y)$ for some $y \in N$. Denote by H the subgraph of G induced by $W-S(y)$ and denote by m the order of H . Since y is a b-descendant of z in T , (6.6) implies that

$$m > bn/2t \ .$$

On the other hand, (6.7) implies that

$$m \leq a+b(1+d) \leq 2a+bd < 3bd \ .$$

Enumerate the vertices of H as $u_1, u_2, \ldots, u_m$ in such a way

$$d_H(u_1) \geq d_H(u_2) \geq \cdots \geq d_H(u_m) \ .$$

Since $bn/2t > n^3/1809t^2d^2$ and since G is (d,ε)-sparse, the graph H has fewer than dm^2/n edges. That is,

$$\sum_{i=1}^{m} d_H(u_i) < 2dm^2/n \quad .$$

It is now easy to see that, for every positive integer r , we have

$$d_H(u_i) < 2dm^2/nr \quad \text{whenever } i > r \ . \tag{6.11}$$

We shall use (6.11) with

$$r = \lfloor am/4bd \rfloor \quad .$$

Let us note at once that $r \geq 200$ and so

$$r > \frac{200}{201} \cdot \frac{am}{4bd} \quad .$$

It will be also useful to note that

$$\frac{2rbd}{a?-} <; \qquad ,$$

$$\frac{2dm^2b}{nr} \leq \frac{201}{200} \cdot \frac{8b^2d^2}{an} \cdot m \leq \left(\frac{201}{200}\right)^2 \mid \frac{4m}{\text{[illegible]}} \quad ,$$

$$r \leq a \quad , \tag{6.12}$$

$$\frac{2br}{a+b-1} \geq \frac{br}{a} \geq 1 \quad . \tag{6.13}$$

And while we are at it, let us also verify that

$$a < \frac{201}{200} \cdot \frac{bn}{50t} < \frac{201}{200} \cdot \frac{m\,m}{25} \quad ,$$

$$b < \frac{1}{1000} \cdot \frac{bn}{2t} < \frac{m}{1000} \quad .$$

So much for high-school algebra. Now, we shall set $R = \{u_1, u_2, \ldots, u_r\}$ and prove that

$$|B(x) \cap R| \geq \frac{2rb}{a+b-1} \tag{6.14}$$

for every $x \in N(S)$. To begin with, (6.7), (6.11) and $B(x) \subseteq W$ imply

$$m \leq a + b + d|B(x) \cap R| + 2dm^2b/nr \quad .$$

If (6.14) failed then we would have

$$m < a+b+2rbd/a + 2dm^2b/nr \ .$$

However, the right-hand side of this inequality is at most

$$m\left(\frac{201}{200}\cdot\frac{1}{25} + \frac{1}{1000} + \frac{1}{2} + \left(\frac{201}{200}\right)^2 \cdot \frac{4}{9}\right) < m \ .$$

Hence (6.14) must hold.

The rest is easy, Consider the subtree of T* consisting of all the paths from the root z to leaves in N(S) ; color each of its nodes x red if $v(x) \in R$ and blue otherwise. By (6.10) and (6.14), this tree is $(a,b,r,2br/(a+b-1))$ -constrained. Because of (6.12) and (6.13), Lemma 5.1 applies and shows that

$$|N(S)| \leq \binom{a+b}{b} \cdot \frac{ab}{a+b} \exp\left(-\frac{b^2}{a+b}\right)$$

By virtue of (6.10), this is the desired inequality (6.3).

Theorem 6.2. Let m be a function of n such that $m(n) = o((n/\log n)^2)$ but $m(n) \geq 10^{10}n$ for all sufficiently large n , Then, for almost all graphs G = (V,E) with n vertices and m edges, every recursive proof of $(V,\alpha(G))$ has length at least

$$\frac{360m}{n^2} \exp \frac{n^2}{m\left(350 \log \frac{m}{n}\right)^3} \ .$$

Proof. By Lemma 2.2 and by Lemma 2.4, almost all graphs with n vertices and: m edges have the following two properties:

(P1) $\alpha(G) < \frac{n^2}{m} \log \frac{m}{n}$,

(P2) every subgraph induced by s vertices such that

$$s > \frac{4n^2}{m} \log \frac{m}{n}$$

has at most $2ms^2/n^2$ edges.

We shall show that all sufficiently large graphs with these two properties satisfy the conclusion of Theorem 6.2. Let us define

$$k(n) = \left\lfloor \frac{n^2}{m}\left(50 \log \frac{m}{n}\right)^3 \right\rfloor ,$$

$$d(n) = 4km/n^2 ,$$

$$\varepsilon(n) = \frac{4n^2}{km} \log \frac{m}{n}$$

Firstly, we shall show that every graph with n vertices and $m \geq 10^{10}n$ edges satisfying (P2) contains an induced (d,ε) -sparse subgraph of order k . To begin with, $2k(n) < n$. Next, an easy averaging argument shows that G contains an induced subgraph H_0 with $2k$ vertices and at most $4mk^2/n^2$ edges. Beginning with H_0 , we shall construct a sequence $H_0, H_1, \ldots$ of induced subgraphs of H_0 as follows: if the last constructed H_i has a vertex v of degree at least d then set $H_{i+1} = H_i - v$, otherwise stop. Clearly, if an H_i gets constructed then H_0 had at least di edges and so $i \leq k$. In particular, the very last H_j in the sequence has at least k vertices; in H_j , we shall choose an induced subgraph H of order k . Let W denote the set of vertices of H . By (P2), every subgraph of H with $s \geq \varepsilon k \geq 4n^2 \log(m/n)/m$ vertices has at most $2ms^2/n^2 < ds^2/k$ edges. Hence H is (d,ε) -sparse.

Next, by (Pl), we have

$$a(G) < \frac{n^2}{m} \log \frac{m}{n} .$$

On the other hand, we have

$$a(G) \geq a(H) \geq k/(d+1) .$$

For all sufficiently large n , Theorem 6.1 asserts that every proof of $(W, \alpha(G))$ has length at least

$$\frac{90d}{k} \exp \frac{k}{20d^2} > \frac{360m}{n^2} \exp \frac{n^2}{m\left(350 \log \frac{m}{n}\right)^3}$$

By Proposition 4.6, this is also a lower bound on the length of every recursive proof of $(V, \alpha(G))$. □

Let us state a special case of Theorem 6.2 in a compact form.

Theorem 6.3. For every sufficiently large integer d there is a constant $c > 1$ with the following property: for almost all graphs $G = (V,E)$ with n vertices and dn edges, every recursive proof of $(V, \alpha(G))$ has length at least c^n .

In closing, two remarks may be in order. Firstly, it would be interesting to construct an infinite class $\mathcal{C}$ of graphs for which there is a constant $c > 1$ with the following property: for every graph $G = (V,E)$ in $\mathcal{C}$ and with n vertices, every recursive proof of $(V, \alpha(G))$ has length at least c^n. Secondly, it is somewhat frustrating that Theorem 6.2 does not apply to graphs with cn^2 edges. Perhaps the following is true.

Conjecture 6.4. There is a positive constant c with the following property: for almost all graphs $G = (V,E)$ with n vertices, every recursive proof of $(V, \alpha(G))$ has length at least $n^{c \log n}$.

7. Concluding Remarks.

There are many "natural" proof systems which extend our system of recursive proofs; we shall mention just a few. It would be interesting to strengthen our results by proving their analogues for the extended proof systems.

To begin with, one might adjoin the following inference rule:

from (S_1,t_1) and (S_2,t_2) we can deduce $(S_1 \cup S_2, t_1+t_2)$.

Proposition 4.3 shows that, to some extent, this rule is implicit in the system of recursive proofs. Nevertheless, its addition just might make the system considerably more powerful. Along this line, further extensions lead to the system of <u>cutting plane proofs</u> which we are about to describe briefly. Let us consider a graph G = (V,E) with vertices $v_1,v_2,..$● ☺✝■ none of which is isolated. A cutting plane proof of (V,t) is a sequence of inequalities

$$\sum_{j=1}^{n} a_{ij}x_j \le b_i \qquad (i = 1,2,\ldots,m)$$

such that

(i) all the numbers a_{ij} and b_i are nonnegative integers,

(ii) for every $k = 1,2,\ldots,m$, either the k-th inequality reads $x_r+x_s < 1$ for some edge v_rv_s or else there are nonnegative multipliers $y_1,y_2,\ldots,y_{k-1}$ such that

$$\sum_{i=1}^{k-1} y_i a_{ij} \ge a_{kj} \quad , \quad \left\lfloor \sum_{i=1}^{k-1} y_i b_i \right\rfloor \le b_k \ ,$$

(iii) the last inequality reads $\sum_{j=1}^{n} x_j \le t$.

It is not difficult to see that $\alpha(G) < t$ whenever there is a cutting plane proof of (V,t) . The converse is easy as well: in fact, every recursive proof of (V,t) can be converted into a cutting plane proof of (V,t) , (The details are left to the reader.)

An inference rule which strengthens the monotone rule and is <u>not</u> subsumed in the notion of a cutting plane proof goes as follows. Let us write $S_1 < S_2$ if there is a one-to-one mapping $f: S_1 \to S_2$ such that $f(u)$ and $f(v)$ are adjacent only if u and v are. Clearly,

from (S,t) we can derive (S',t') whenever $S' < S$ and $t' \geq t$.

Again, it would be interesting to find out whether the addition of this inference rule makes the system of recursive proofs considerably stronger.

Colin McDiarmid [18] investigated a proof system, similar to our system of recursive proofs, for deriving lower bounds on the chromatic number of graphs. We shall describe his system very briefly. Let G be a graph whose vertices are labeled by nonempty and pairwise disjoint subsets of $\{1,2,\ldots,n\}$; let u and v be two vertices of G . We shall denote by G' the graph obtained from G by adding the edge uv ; we shall denote by G'' the graph obtained from G by identifying u with v (in which case the label of the new vertex is the union of the labels of u and v). As usual, $\omega(G)$ denotes the order of the largest clique in G . By a recursive proof of $[G_m,t_m]$, we shall mean a sequence

$$[G_i,t_i] \quad , \quad i = 1,2,\ldots,m$$

such that, for each k , either $t_k \leq \omega(G_k)$ or else there are subscripts $i, j < k$ such that $G_i = G_k'$, $G_j = G_k''$ and $t_k = \min(t_i,t_j)$. Clearly, if there is a recursive proof of $[G_m,t_m]$ then $\chi(G_m) \geq t_m$. McDiarmid has proved that, for almost all graphs with n vertices, every recursive proof of $[G,\chi(G)]$ has length at least

$$\exp(.157\, n\, (\log n)^{1/2}) \quad .$$

His result implies that the average running time of the Corneil-Graham algorithm for finding the chromatic number of a graph [4] grows faster than every exponential. On the other hand, Lawler [16] has designed an algorithm for finding the chromatic number of a graph of order n within $O(2.45^n)$ steps. (Of course, these facts are of an asymptotic nature and imply nothing about the relative merits of the two algorithms applied to graphs with, say, 200 vertices.)

Finally, I wish to thank several friends for their help with my work on this paper. To Colin McDiarmid and Endre Szemerédi I am indebted for many stimulating conversations. Persi Diaconis told me about Hoeffding's paper [13]. David Avis, Don Knuth, Ivo Rosenberg, and Bob Tarjan read various versions of the manuscript and made many helpful suggestions to improve the presentation.

References

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

[2] S. A. Cook, "The complexity of theorem-proving procedures," Proceedings of Third Annual ACM Symposium on Theory of Computing, 1971, 151-158.

[3] S. A. Cook and R. A. Reckhow, "On the lengths of proofs in the propositional calculus," Proceedings of Sixth Annual ACM Symposium on Theory of Computing, 1974, 135-148.

[4] D. G. Corneil and B. Graham, "An algorithm for determining the chromatic number of a graph," SIAM J. Comput. 2 (1973), 311-318.

[5] P. Erdös, "Some remarks on the theory of graphs," Bull. Amer. Math. Soc. 53 (1947), 292-294.

[6] P. Erdös, "On circuits and subgraphs of chromatic graphs," Mathematika 9 (1962), 170-175.

[7] P. Erdös and B. Bollobás, "Cliques in random graphs," to appear in Math. Proc. Camb. Phil. Soc.

[8] P. Erdös and A. Renyi, "On random graphs, I," Publ. Math. Debrec. 6 (1959), 290-297.

[9] P. Erdös and A. Renyi, "On the evolution of random graphs," Magyai Tud. Akad. Mat. Kut. Int. Közl. 5 (1960), 17-61.

[10] P. Erdös and A. Renyi, "On the strength of connectedness of a random graph," Acta Math. Acad. Sci. Hungar. 12 (1961), 261-267.

[11] Z. Galil, "On enumeration procedures for theorem proving and for integer programming," IBM report RC 5719 (# 24648), 1975.

[12] G. R. Grimmett and C. J. H. McDiarmid, "On colouring random graphs," Math. Proc. Camb. Phil. Soc. 77 (1975), 313-324.

[13] W. Hoeffding, "Probability inequalities for sums of bounded random variables," Amer. Stat. Assoc. J. 58 (1963), 13-29.

[14] R. M. Karp, "Reducibility among combinatorial problems," Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York, 1972, 85-104.

[15] D. E. Knuth, The Art of Computer Programming, Vol. 3: Sorting and Searching, Addison-Wesley, Reading, Mass., 1973.

[16] E. L. Lawler, "A note on the complexity of the chromatic number problem," Information Processing Letters 5 (1976), 66-67.

[17] D. W. Matula, "On the complete subgraphs of a random graph," Proc. 2nd Conf. Comb. Th. and Appl. (1970), Chapel Hill, 356-369.

[18] C. J. H. McDiarmid, "Determining the chromatic number of a graph," submitted to SIAM J. Computing.

[19] R. E. Tarjan, "Finding a maximum clique," Technical Report 72-123, Computer Science Dept., Cornell University (1972).

[20] R. E. Tarjan and A. E. Trojanowski, "Finding a maximum independent set," to appear in SIAM J. Computing.