SIAM J. COMPUT.
Vol. 4, No. 3, September 1975

AN ALGORITHM FOR DETERMINING WHETHER THE CONNECTIVITY OF A GRAPH IS AT LEAST k*

SHIMON EVEN†

Abstract. The algorithm presented in this paper is for testing whether the connectivity of a large graph of n vertices is at least k. First the case of undirected graphs is discussed, and then it is shown that a variation of this algorithm works for directed graphs. The number of steps the algorithm requires, in case $k < \sqrt{n}$, is bounded by $O(kn^3)$.

Key words. algorithm, connectivity, graph

1. Introduction. Let G be a finite undirected graph with n vertices and e edges. We assume that G has no self-loops and no parallel edges. A set of vertices, S, is called a *separating set* if there exists two vertices $a, b \notin S$ such that all paths between a and b pass through at least one vertex of S. The *connectivity*, c, of G is defined in the following way:

(i) if G is completely connected,[1] then $c = n - 1$,

(ii) if G is not completely connected, then c is the least number of vertices in a separating set.

Menger's theorem [1] states that if the connectivity of G is c, then for every two vertices a and b there exist c vertex-disjoint paths connecting a and b;[2] and conversely, if for every two vertices a and b there exist c vertex-disjoint connecting paths, then the connectivity of G is at least c. Dantzig and Fulkerson [2] introduced the relation between connectivity and network flow. Thus the Ford and Fulkerson [3] algorithm can be used to determine the connectivity of a graph. In fact, the max-flow min-cut theorem (and algorithm) immediately translates to the following: the maximum number of vertex-disjoint paths connecting vertices a and b is equal to the minimum cardinality separating set between a and b, in case there is no edge between a and b; otherwise the number of paths is one more than the minimum cardinality of a set separating a from b after the edge between them has been deleted.

Thus one can find the connectivity of a graph in the following way: for each pair of vertices, find the maximum number of vertex-disjoint paths. The minimum value over all pairs is the connectivity.

For each pair of vertices, we construct a flow network whose number of vertices is $2n$ and the number of edges is $2e + n$. The capacities are all one.[3] Each labeling and augmenting path realization costs $O(e)$ steps, and it corresponds to one path between the two vertices. Since the connectivity can be as high as $n - 1$, the whole procedure for finding the maximum number of vertex-disjoint paths connecting this pair of vertices is at most of cost $O(ne)$, or $O(n^3)$ if $O(e) = O(n^2)$.

* Received by the editors September 12, 1973, and in revised form August 7, 1974.

† Department of Computer Science, Technion-Israel Institute of Technology, Haifa, Israel. On leave of absence from Weizmann Institute of Science, Rehovot, Israel.

[1] Each pair of vertices is connected by an edge. In this case, G has no separating sets.

[2] Clearly, the vertices a and b are shared by all c paths, but no other vertex, and therefore no edge, is shared.

[3] For more details, see, for example, [4, p. 226]. There, some of the capacities are infinite and some are unit. Changing them all to one unit does not change anything.

Repeating this for all pairs will cost, then, at most $O(n^5)$. Recently, Even and Tarjan [5] have reduced these to $O(n^{2.5})$ and $O(n^{4.5})$, respectively.

Assume now that we are not interested in the connectivity itself, but rather would like to check whether the connectivity is at least k, where k is much smaller than n. It is natural to investigate the question of whether we can find an algorithm which requires less than $O(n^{4.5})$ steps.

Kleitman [6] has shown a method which takes at most $O(k^2n^3)$ steps. In § 2 I shall present a method, an improvement of Kleitman's technique, which takes at most $O(kn^3 + k^3n^2)$ steps. Directed graphs are discussed in § 3.

It is proper to comment here that the case of $k = 1$ is trivially solvable in $O(e)$ steps. The case $k = 2$ was solved in $O(e)$ steps by Hopcroft and Tarjan [7], who proceeded to solve the case $k = 3$ in $O(e)$ steps, too [8]. Their methods are different from the ones described above. They use the powerful technique of depth-first search (which was already known in the 19th century as a maze threading technique. See, for example, Lucas' [9] report of Trémaux's work). I do not believe that their methods will extend for higher k's.

2. The algorithm for undirected graphs. Let G be an undirected graph with n vertices and e edges. Let $L = \{v_1, v_2, \cdots, v_l\}$ be a set of vertices of G and u be a vertex of G not in L. Let k be a positive integer such that $k \leqq l$.

Let us add to G a new vertex a and connect it by an edge to each of the vertices in L. The new graph, $\tilde{G}$, will be called the *augmented graph*.

LEMMA 1. *If in G each vertex v_i $(1 \leqq i \leqq l)$ can be connected to u via k vertex-disjoint paths, then in $\tilde{G}$ there are k vertex-disjoint paths between a and u.*

Proof. Assume not. Then there is a separating set S, $|S| < k$, such that all paths from u to a pass through at least one vertex of S. Consider the set of vertices, U, such that, as for u, all the paths from them to a pass through at least one vertex of S. None of the vertices in L can be in U, since each vertex of L is connected by an edge to a. Thus there exists a vertex v in L which is not in U and not in S. Every path from u to v must pass through at least one vertex of S. Thus there cannot be k vertex-disjoint paths between u and v, a contradiction. Q.E.D.

Assume the set of G's vertices is $\{1, 2, \cdots, n\}$. Let j be the least vertex such that for some $i < j$ there are no k vertex-disjoint paths connecting i and j in G.

LEMMA 2. *Let j be as defined above and $\tilde{G}$ be the augmented graph where $L = \{1, 2, \cdots, j - 1\}$. There are no k vertex-disjoint paths connecting a and j in $\tilde{G}$.*

Proof. Consider a minimum separating set S, such that all paths between i and j pass through at least one vertex of S. It follows that $|S| < k$. Let U be the set of all vertices such that all paths from them to i must pass through at least one vertex of S. Clearly, $j \in U$. If a vertex $j' < j$ is in U, then there are no k vertex-disjoint paths from j' to i, and jth choice was erroneous. Thus j is the least vertex in U, or $L \cap U = \emptyset$. Namely, all paths from j to vertices in L must pass through, or end in, a vertex in S. It follows that in $\tilde{G}$ there are no k vertex-disjoint paths between a and j. Q.E.D.

We are now ready for the algorithm for determining whether the connectivity of G is at least k.

ALGORITHM 1.

1. For every i and j such that $1 \leqq i < j \leqq k$, check whether there are k

vertex-disjoint paths between them. If for some i and j the test fails, then G's connectivity is less than k.

2. For every j, $k + 1 \leqq j \leqq n$, form[4] $\tilde{G}$ and check whether there are k vertex-disjoint paths between a and j. If for some j the test fails, then G's connectivity is less than k.
3. The connectivity of G is at least k.

The proof of validity of this algorithm is as follows: if G's connectivity is at least k, then by Lemma 1, step 2 will detect no failure, and the algorithm will halt with the correct answer. If G's connectivity is less than k, then by Lemma 2, failure will occur, and again the algorithm will halt with the correct answer.

Step 1 of the algorithm requires at most $O(k^3 \cdot e)$ elementary steps, and step 2 requires at most $O(k \cdot n \cdot e)$. Thus the whole algorithm requires at most $O(k^3 \cdot e + k \cdot n \cdot e)$ steps. Since $O(e) \leqq n^2$, the number of steps is bounded by $O(k^3 \cdot n^2 + k \cdot n^3)$. For $k < \sqrt{n}$, $O(k \cdot n^3)$ is an upper bound.

The algorithm is an improvement of Kleitman's algorithm [6], which may require $O(k^2 \cdot n^3)$ steps. The saving is achieved by the addition of the vertex a and checking its connectivity to every vertex j (in $\tilde{G}$) instead of repeating this test for k different vertices.

3. The algorithm for directed graphs. Let G be a directed graph whose set of vertices is $\{1, 2, \cdots, n\}$ and with e edges. An *(i, j)-separating set*, S, is a set of vertices such that every directed path from i to j passes through at least one vertex in S. The *connectivity* of G is defined as follows:

(i) if the graph is completely connected (namely, $e = n(n - 1)$), then $c = n - 1$,

(ii) if the graph is not completely connected, then c is the least cardinality of a separating set.

Menger's theorem holds in this case, too, and the network flow technique applies. The straightforward technique of checking if there are k vertex-disjoint directed paths between every ordered pair of vertices takes at most $O(kn^4)$ steps.

Let $\vec{G}$ be an augmented graph constructed for j as follows: add a new vertex a to the graph and connect a by an edge to each of the vertices in $L = \{1, 2, \cdots, j - 1\}$. Similarly, $\overleftarrow{G}$ is constructed by adding a new vertex a and edges from each of the vertices in L to a. Assume now that $j > k$.

LEMMA 3. *If in G each $i \in L$ can be connected to j via k vertex-disjoint directed paths, then in $\vec{G}$ there are k vertex-disjoint directed paths from a to j.*

The proof of this Lemma and the following one is analogous to that of Lemma 1.

LEMMA 4. *If, in G, j can be connected to each $i \in L$ via k vertex-disjoint directed paths, then in $\overleftarrow{G}$, there are k vertex-disjoint directed paths from j to a.*

Let j be the least vertex such that for some $i < j$ either there are no k vertex-disjoint directed paths from i to j or there are no k vertex-disjoint directed paths from j to i.

LEMMA 5. *Assume j is as defined and that there are no k vertex-disjoint directed paths from i to j (from j to i). There are no k vertex-disjoint paths from a to j in $\vec{G}$ (from j to a in $\vec{G}$).*

The proof is analogous to that of Lemma 2.

[4] Clearly, $L = \{1, 2, \cdots, j - 1\}$.

ALGORITHM 2.

1. For every i and j such that $1 \leqq i < j \leqq k$, check whether there are k vertex-disjoint directed paths from i to j and also if there are k such paths from j to i. If one of these tests fails, then G's connectivity is less than k.
2. For every j, $k + 1 \leqq j \leqq n$, form $\vec{G}$ and check whether there are k vertex-disjoint directed paths from a to j; also form $\overleftarrow{G}$ and check whether there are k such paths from j to a. If for some j one of these tests fails, then G's connectivity is less than k.
3. The connectivity of G is at least k.

The proof of validity is similar to that of Algorithm 1. Again, step 1 takes at most $O(k^3n^2)$ and step 2, $O(kn^3)$. If $k < \sqrt{n}$, then the whole algorithm takes at most $O(kn^3)$ steps.

Acknowledgment. This work was done while the author was with the Department of Computer Science, Cornell University, during the summer of 1973. It is a revision of Report TR-73-184 of the Department of Computer Science, Cornell University, Ithaca, New York, 1973.

REFERENCES

[1] K. MENGER, *Zur allgemeinen Kurventheorie*, Fund. Math., 10 (1927), pp. 96–115.
[2] G. B. DANTZIG AND D. R. FULKERSON, *On the max-flow min-cut theorem of networks*, Linear Inequalities and Related Systems, Annals of Math. Study, no. 38, Princeton University Press, Princeton, N.J., 1956, pp. 215–221.
[3] L. R. FORD AND D. R. FULKERSON, *Flows in Networks*, Princeton University Press, Princeton, N.J., 1962.
[4] S. EVEN, *Algorithmic Combinatorics*, Macmillan, New York, 1973.
[5] S. EVEN AND R. E. TARJAN, *Network flow and testing graph connectivity*, this Journal, to appear.
[6] D. J. KLEITMAN, *Methods for investigating connectivity of large graphs*, IEEE Trans. Circuit Theory, CT-16 (1969), pp. 232–233.
[7] J. HOPCROFT AND R. TARJAN, *Algorithm* 447; *Efficient algorithms for graph manipulation*, Comm. ACM, 16 (1973), pp. 372–378.
[8] ———, *Dividing a graph into triconnected components*, this Journal, 2 (1973), pp. 135–158.
[9] E. LUCAS, *Récreations Mathématiques*, Paris, 1882.