ON THE TIME REQUIRED TO DETECT CYCLES AND CONNECTIVITY IN DIRECTED GRAPHS

by

R.C. Holt

and

E.M. Reingold

Technical Report

No. 70-63

June 1970

Department of Computer Science
Cornell University
Ithaca, New York 14850

ON THE TIME REQUIRED TO DETECT CYCLES AND CONNECTIVITY IN DIRECTED GRAPHS

by

R.C. Holt and E.M. Reingold

Abstract

It is shown that when a directed graph is represented as a binary connection matrix, the problem of finding the shortest path between two nodes of a directed graph, and the problem of determining whether the directed graph has a cycle require at least $O(n^2)$ operations. Thus the presently known best algorithms are optimal to within a multiplicative constant.

Introduction

This note shows that the best possible algorithms for certain problems involving directed graphs require at least $O(n^2)$ (on the order of n^2) operations, where n is the number of nodes in the graph. It is assumed that the graph is presented as an n by n binary connection matrix. It is also assumed that it requires at least one operation to inspect or modify any element of the matrix. For example, an operation which performs a parallel "or" between respective elements of rows of the matrix is not allowed. These results do not apply directly to graphs presented as other data structures, such as linked lists; corresponding proofs in these cases are complicated by the fact that the details of such representations vary widely.

In particular, we show that $O(n^2)$ operations are required to determine if a graph contains a cycle, or to determine if there is a path from node i to node j. Both of these results depend on the fact that if fewer than $O(n^2)$ operations are performed then not enough elements of the matrix can have been inspected to correctly determine the answer.

Results

Definition: An n by n binary matrix M is said to contain a *cycle* if and only if there is a sequence of elements of M such that:

$$M(k_1,k_2) = M(k_2,k_3) = \cdots = M(k_{m-1},k_m) = M(k_m,k_1) = 1.$$

Theorem: If an algorithm determines whether n by n binary matrices contain a cycle, then for some matrix the algorithm requires at least $n(n-1)/2$ operations.

Proof: Let A be an algorithm which determines whether n by n binary matrices contain a cycle, and let M be an n by n binary matrix which does not contain a cycle. Observe that there are $\binom{n}{2} = n(n-1)/2$ unordered pairs (i,j) where $1 \le i \le n$ and $1 \le j \le n$. Assume A inspects fewer than $n(n-1)/2$ elements of M; then there is an unordered pair (i,j) such that A inspects neither M(i,j) nor M(j,i). Construct M' to be identical to M except that M'(i,j) = M'(j,i) = 1; thus M' will contain a cycle. Clearly, A must come to the same conclusion for M' that it came to for M, contradicting the fact that M' contains a cycle. Thus A must use at least $n(n-1)/2$ operations.

Remark: A slightly higher limit, namely $n(n+1)/2$, can be obtained by noticing that any correct algorithm must also inspect all n diagonal elements, M(i,i), of M, before concluding that M contains no cycles.

Definition: Node i is said to be connected to node j in an n by n binary matrix M if and only if there is a sequence of elements of M such that:

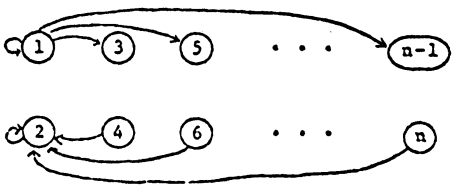
$$M(i,k_1) = M(k_1,k_2) = \cdots = M(k_{m-1},k_m) = M(k_m,j) = 1.$$

Theorem: If an algorithm determines whether node i is connected to node j in n by n binary matrices then for some matrix the algorithm requires at least $(n^2-1)/4$ operations.

Proof: Let A be an algorithm which determines whether node i is connected to node j. The theorem will be proved by constructing a matrix which is "difficult" for A to analyze. We will assume that n is even and we will prove the slightly stronger result that A requires $n^2/4$ operations. A similar construction will apply if n is odd. Assume i = 1 and j = 2. We define:

$$\left.\begin{array}{l} M(1,j) = 1 \text{ for } j \text{ odd}, \\ M(j,2) = 1 \text{ for } j \text{ even}, \\ M(i,j) = 0 \text{ otherwise}. \end{array}\right\} \qquad (1)$$

The graphical interpretation of (1) is that node 1 is connected to all odd numbered nodes, and that all even numbered nodes are connected to node 2.



Then, if any odd numbered node becomes connected to any even numbered node, node 1 becomes connected to node 2. There are n/2 odd numbered nodes and n/2 even numbered nodes. So, the number of elements M(i,j) such that i is odd and j is even is $n^2/4$. We will now show that A must inspect at least this many elements. Assume A always requires less than $n^2/4$ operations to determine if node 1 is connected to node 2. Clearly, in M as defined node 1 is not connected to node 2, and this is what A must conclude. However, since A used less

than $n^2/4$ operations there is an ordered pair (i,j) such that A did not inspect M(i,j) where i is odd and j is even. Construct M' identical to M except that M'(i,j) = 1, so that node 1 is connected to node 2. A will not inspect M'(i,j), and so A must conclude that node 1 is not connected to node 2 exactly as it concluded for M. Therefore A must use at least $n^2/4 \geq (n^2-1)/4$ operations to see if i is connected to j. When n is odd, a similar argument shows that A must use $(n^2-1)/4$ operations.

Conclusions

These results show that the known algorithms to compute various functions of binary matrices can be considered to be optimal to within a multiplicative constant. Marimont [3] gives an $O(n^2)$ algorithm to test for the existence of cycles. Warshall [5] gives an algorithm to calculate the transitive closure $M^* = \sum_{i=1}^{n} M^i$ of a matrix M. If arbitrarily large numbers can be represented in registers, then Warshall's algorithm is $O(n^2)$. Since M contains a cycle if and only if M* has a one on its diagonal, it follows that Warshall's algorithm is optimal. Dreyfus [2], Dijkstra [1], and Pohl [4] give $O(n^2)$ algorithms to find the shortest path between two nodes of a graph. Finding the shortest path between two nodes requires determining whether or not such a path exists and thus $O(n^2)$ shortest path algorithms are optimal.

Of course this does not mean that the best algorithms for computing these functions have been found. While we have shown

that the time required to compute these functions is $O(n^2)$ for the worst matrices, considerable improvement might be made, either statistically or for particular classes of matrices. Improvements in the multiplicative constant of an algorithm are another possibility; for example, we may be able to reduce the computation time from $7n^2$ to $2n^2$ in the worst case.

Until recently when an algorithm was presented, little or no effort was devoted to proving that the algorithm required all of the resources it used. Hopefully, these results show that such efforts can be worthwhile and proofs should be attempted.

Acknowledgement

The authors gratefully acknowledge the suggestion by Robert Wagner which simplified the proof of the second theorem.

References

1. Dijkstra, E. A note on two problems in connection with graphs. Numerisch Mathematik 1 (1959), 269-271.

2. Dreyfus, D. An appraisal of some shortest path algorithms. Operations Research 17 (1969), 395-412.

3. Marimont, R.B. A new method of checking the consistency of precedence matrices. JACM 6 (1959), 164-171.

4. Pohl, I. A theory of bi-directional search in path problems. Report No. RC 2713, IBM, Yorktown Heights, N.Y., (1969).

5. Warshall, S. A theorem on Boolean matrices. JACM 9 (1962), 11-12.