BIT 6 (1966), 347–349

SCIENTIFIC NOTES

AN ALGORITHM FOR COMPUTING ALL PATHS IN A GRAPH

LARS-ERIK THORELLI

Introduction.

Let G be a directed graph with n nodes numbered $1,2,\ldots,n$. The $n \times n$ incidence matrix A of G is defined by

$$a_{ij} = \begin{cases} 1 \text{ if } G \text{ contains an arc going from} \\ \quad \text{node } i \text{ to node } j\,, \\ 0 \text{ otherwise}\,. \end{cases}$$

The $n \times n$ reachability matrix B of G is defined by

$$b_{ij} = \begin{cases} 1 \text{ if } i=j \text{ or there exists a directed} \\ \quad \text{path in } G \text{ from node } i \text{ to node } j\,, \\ 0 \text{ otherwise}\,. \end{cases}$$

It is well known that B and A are connected by the equation $B = (A+I)^{n-1}$, where I denotes the unit matrix, and the summation involved in the computation of the matrix elements shall be carried out in the Boolean sense, that is, by using the rules $0+0=0$, $0+1=1+0=1+1=1$.

Various algorithms for computing the reachability matrix have been used, the probably most economical of which is due to Warshall [1]. We give below an algorithm for computing B which is especially suitable if the number of arcs in G is relatively small compared to the number of nodes.

The algorithm.

The algorithm uses the following method of representing the matrices involved. The input matrix, A, is represented by an enumeration of all pairs (i,j) with $i \neq j$ and $a_{ij}=1$:

$$A \leftrightarrow B_1,B_2,\ldots,B_N\,.$$

We denote the first number in the pair B_k by $hd(B_k)$ and the second number by $tl(B_k)$, so that $B_k = (hd(B_k),tl(B_k))$.

During the execution of the algorithm new components $B_{N+1},B_{N+2},\ldots$ are added to the list, so that at the end all pairs corresponding to 1's in the reachability matrix B are included, apart from those of the main diagonal.

The algorithm consist of the following steps.

(α) $i := 1$
(β) $j := 1$
(γ) If $i=j$, go to step (ε).
(δ) If $tl(B_i)=hd(B_j)$ and $hd(B_i)\neq tl(B_j)$, then it is investigated if the pair $(hd(B_i),tl(B_j))$ is included in the list. If this is not the case, N is increased by 1 and B_N is taken as $(hd(B_i),tl(B_j))$.
(ε) $j := j+1$. Go to step (γ) if $j\leqq N$.
(ζ) $i := i+1$. Go to step (β) if $i\leqq N$.
(η) Stop .

Proof of the validity of the algorithm.

Two assertions must be proved about the final state of the list $B_1,\ldots,B_N$.

(1) $b_{hd(B_k),tl(B_k)} = 1$ for $k = 1,\ldots,N$.

This is easily shown by an induction argument.—Let $hd(B_k)=u$, $tl(B_k)=v$. If (u,v) was included in $B_1,\ldots,B_N$ at the start, then $a_{uv}=1$ and consequently $b_{uv}=1$.—The addition of a new pair (u,v) presupposes that there has been found pairs (u,w) and (w,v) in the existing list. This means (by the inductive assumption) that there exists a directed path from u to w and a directed path from w to v in the graph G. Consequently there exists a directed path from u to v, i.e., $b_{uv}=1$.

(2) For all u,v with $u\neq v$ and $1\leqq u,v\leqq n$, $b_{uv}=1$ implies the existence of a k with $1\leqq k\leqq N$ such that $B_k=(u,v)$.

We assume the contrary, i.e., there exists a directed path from a node u to a node v where $(u,v)\neq B_k$ for $k=1,\ldots,N$. Let $u=u_0-u_1-\ldots-u_r=v$ denote such a path of minimal length r. $(u_r,\ldots,u_{r-1}$ denote the intermediate nodes.) It is easily seen that $r\geqq 2$. The definition of r implies that the pairs $B_p=(u_0,u_{r-1})$ and $B_q=(u_{r-1},u_r)$ both are included in the list $B_1,\ldots,B_N$. Note that B_q was included in the list before the computation started.—If B_p was also present at the start, then the algorithm would have added the pair (u_0,u_r) at $i=p, j=q$.—If B_p was added during the computation, say at $i=i_1<p$, then the algorithm would have added (u_0,u_r) when i had been increased to p. In each case a contradiction is established.