An Algorithm to Compute Transitive Closure Matrix for Acyclic Directed Graph

Ming-Chi Lee and Chen-Huang Dong

Department of Information Technology
National Ping Tung Institue of Commerce
Pingtung, Taiwan 900
Republic of China.
E-mail: lmc@npic.edu.tw

Abstract

Warshall proposed an algorithm to compute the transitive closure matrix for any directed graph in 1962. Over the past four decades, it has been widely applied to various applications. However, no matter what kind of directed graph it is, this algorithm requires $O(n^3)$ time. In fact, there are many applications of the transitive closure matrix just for acyclic directed graph, and so finding it efficiently is important. In this paper, a fast algorithm is proposed to compute the transitive closure for the acyclic directed graph. This algorithm takes $O(k \times n)$ time for an acyclic directed graph with n vertices and k edges.

Keywords: Directed Graph, Transitive Closure Matrix

1 Introduction

Given a directed graph $G = (V, E)$, the transitive closure $T = (V, F)$ of G is a directed graph such that there is an edge (v, w) in T if and only if there is a directed path from v to w in G. Over the past four decades, there were two approaches to solve the problem. The first one was proposed by Warshall in 1962 [1]. Warshall proposed a Theorem on Boolean Matrices, which presented an algorithm to compute the transitive closure matrix for any directed graph. The original use of boolean matrices was to represent program topology [4, 5]. In consequence, it led to interest in algorithms for transforming the boolean matrix to the transitive closure matrix. The other is to transform the *all-pairs shortest-paths* problem directly into the transitive closure problem. Many researches showed the solution to the the all-pairs shortest-paths problem can be transformed directly into the solution for transitive closure problem [2]. In fact, the modified algorithm of the *all-pairs shortest-paths* problem to a transitive closure algorithm is the same as Warshall's algorithm [3].

However no matter what kind of directed graph it is, Warshall's algorithm requires $O(n^3)$ time. Even an acyclic graph or a tree structure takes $O(n^3)$ time. In fact, there are many applications of the transitive closure just for acyclic directed graph, and so finding it efficiently is important. The purpose of this paper is to develop a fast algorithm to compute the transitive closure matrix for the acyclic directed graph. The paper is laid out as follows. In Section 2, we introduce briefly the Warshall's algorithm and some basic definitions related to transitive closure matrix. In Section 3, we propose this fast algorithm and prove it to be correct. In Section 4, we analyze the complexity of this algorithm, and finally in Section 5, we show the conclusion.

2 Background

Let $G = (V, E)$ be a directed graph with n vertices and k edges, where $n \geq 1$ and $k \geq 1$. G may be represented by an adjacency matrix, which is a 2-dimensional $n \times n$ array, say M, with the property that $M[i, j] = 1$ iff the edge $v_i \rightarrow v_j$ is in E. $M[i, j] = 0$ if there is no such edge in E, and where $1 \leq i, j \leq n$. This adjacency matrix M may be defined as follows:

$$M = [v_{ij}]_{n \times n} \begin{cases} v_{ij} = 1 & \text{if } v_i \rightarrow v_j \in E, \text{ and } 1 \leq i, j \leq n \\ v_{ij} = 0 & \text{otherwise} \end{cases}$$

The transitive closure matrix M^+ with respect to M can be defined as follows:

$$M^+ = \sum_{k=1}^{n-1} M^k = M^1 + M^2 \cdots + M^{n-1} \quad (1)$$

In the following, we introduce briefly Warshall's algorithm to compute the transitive closure matrix [1]. Using adjacency matrix Warshall's algorithm will requires at least $O(n^3)$ as $n^3 - n$ entries of the matrix have to be examined. Note that when the input matrix is a sparse matrix (i.e., most entries are zero), this algorithm is very inefficient.

Warshall's Theorem : Given a square ($d \times d$) matrix G each of whose elements m_{ij}^+ is 0 or 1, define $G^+ = 1$ by $m_{ij} = 1$ if and only if either $m_{ij} = 1$ or there exist integers $k_1, \cdots, k_n$ such that $m_{ik_1} = m_{k_1k_2} = \cdots = m_{k_{n-1}k_n} = m_{k_nj} = 1$; $m_{ij} = 0$, otherwise. Define G^+ by the following construction:

1. Set $G^+ = G$
2. Set $i = 1$
3. $(\forall\ j\ \ni: m_{ji}^+ = 1)(\forall k)$ set $m_{jk}^+ = m_{jk}^+ \wedge m_{ik}^+$
4. Increment i by 1
5. If $i \leq d$, goto step 2; otherwise, stop.

Proof: See [1].

3 Transitive closure algorithm for acyclic directed graph

The problem: Given an acyclic directed graph $G = (V, E)$ with n vertices and k edges, where $V = \{v_1, v_2, \cdots, v_n\}$ and $E = \{e_1, e_2, \cdots, e_k\}$, compute the transitive closure $\mathcal{F}(E, k)$ of G.

Before solving the problem, we introduce some basic definitions of directed graph and its adjacency matrix.

Definition 1: For an acyclic directed graph $G = (V, E)$, suppose that v_i and v_j are two vertices in V. Then v_i is a *predecessor* of vertex v_j iff there is a directed path from vertex v_i to vertex v_j. The path is denoted as $v_i \stackrel{k}{\rightarrow} v_j$, where k represents the distance between v_i and v_j and $k \geq 1$. v_i is an *immediate predecessor* of v_j iff $v_i \rightarrow v_j$ is an edge in G.

Lemma 1: An acyclic directed graph always contains a vertex with no predecessor (i.e., indegree 0).

Proof. see [3]. □

Lemma 2: Given an acyclic directed graph $G = (V, E)$ with its transitive closure matrix M, suppose that v_j is a vertex in V. Then, all the predecessors of v_j corresponds to the nonzero entries in the jth column of M.

Proof. Suppose that the predecessors of v_j is a set of vertices called Ω, where $0 \leq |\Omega| < n$. By definition 1, for all the predecessor $v_i \in \Omega$, there must exist a path $v_i \stackrel{k}{\rightarrow} v_j$ between v_i and v_j. By equation (1), the definition of transitive closure matrix , $v_i \stackrel{k}{\rightarrow} v_j$ corresponds to $M[i, j] = 1$, which can be obtained by M^k. Hence all the predecessors of v_j are respect to the nonzero entries in the jth column of M. □

Instead of proceeding on the vertices like Warshall's algorithm, our algorithm proceeds on the edges of E, which has been sorted in the topological order. The problem involves k edges and is solved by induction. The first attempt is to reduce the problem by removing one edge. Let $G' = G - e_k$ represent that the k_{th} edge e_k is removed from G. In an inductive approach, if we have computed the transitive closure of G' already, the transitive closure of G can be achieved by means of adding e_k to G'. Let $\mathcal{F}(E, k)$ denote the transitive closure matrix of G and $\mathcal{F}(E - e_k, k - 1)$ be the transitive closure matrix of G'. Now we try to find a matrix function called $\oint(e_k)$, which can transform $\mathcal{F}(E - e_k, k - 1)$ to $\mathcal{F}(E, k)$ by adding e_k to G'. This matrix function $\oint(e_k)$ is defined as follows.

Definition 2: Let $e_k = v_i \rightarrow v_j$ and $\oint(v_i \rightarrow v_j)$ is a matrix function, which consists of two operations : (1) Set $M_{n\times n}[i, j] = 1$, where $i \leq n$, and $j \leq n$. (2) Add the ith column of M to jth column of M.

After adding $v_i \rightarrow v_j$ to G', all the predecessors of v_i can reach v_j transitively. By Lemma 2, the predecessors of vertex v_i are kept in the ith column of M. Therefore, we can obtain the transitive closure matrix $\mathcal{F}(E, k)$ by adding the ith column of M to jth column of M. To achieve the transitive closure matrix $\mathcal{F}(E, k)$, an induction hypothesis is proposed.

Induction hypothesis: we have computed the transitive closure matrix $\mathcal{F}(E - e_k, k - 1)$ of G', and $\mathcal{F}(E, k) = \mathcal{F}(E - e_k, k - 1) + \oint(e_k)$

Hence we can obtain the transitive closure of G if and only if we can prove that $\mathcal{F}(E, k) = \mathcal{F}(E - e_k, k - 1) + \oint(e_k)$ holds.

Theorem 1: For an acyclic directed graph $G = (V, E)$ with $|E| = k$ and $|V| = n$. Let $G' = G - e_k$, where e_k be the kth edge in E, which has been sorted in the topological order. Suppose that $\mathcal{F}(E-e_k, k-1)$ is the transitive closure matrix of G'. Then $\mathcal{F}(E,k)= \mathcal{F}(E - e_k, k-1) + \oint(e_k)$ holds.

Proof. Let e_k denotes the kth edge $v_i \rightarrow v_j$ in E. After adding $v_i \rightarrow v_j$ to G', there are four cases between $v_i \rightarrow v_j$ and G':

1. v_i belongs to G', but v_j is a new vertex and not in G'(see Figure 1): Because v_i is the immediate predecessor of v_j, all the predecessors of v_i can reach v_j by transitivity. By Lemma 2, we see that the predecessors of vertex v_i correspond to the nonzero entries in the jth column of M. Hence the predecessors of vertex v_j can be obtained from the ith column of M. By definition 2, we can transform $\mathcal{F}(E - e_k, k-1)$ to $\mathcal{F}(E,k)$ by means of performing the matrix function $\oint(v_i \rightarrow v_j)$. Thus, $\mathcal{F}(E,k)= \mathcal{F}(E-e_k, k-1) + \oint(e_k)$ holds.

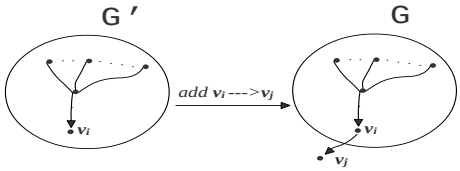

Figure 1: Add $v_i \rightarrow v_j$ to G', where v_j is a new vertex

2. both v_i and v_j belongs to G' already (see Figure 2): After adding $v_i \rightarrow v_j$ to G', v_j inherits all the predecessors from v_i. This case has the same result as case 1. Thus, $\mathcal{F}(E,k)= \mathcal{F}(E-e_k, k-1) + \oint(e_k)$ holds.

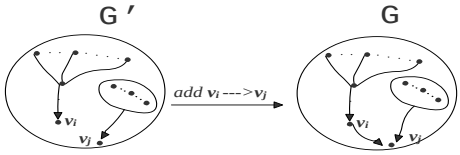

Figure 2: Add $v_i \rightarrow v_j$ to G', where v_i and v_j belongs to G'

3. both v_i and v_j do not belong G' (see Figure 3): Because v_i and v_j are two newly added vertices, it means that they do not have any predecessors, so all the entries of ith column and jth column of $M_{n\times n}$ must be zero originally. Therefore, whether or not add ith column of $M_{n\times n}$ to jth column of $M_{n\times n}$ do not affect the result. Thus, $\mathcal{F}(E,k)= \mathcal{F}(E - e_k, k-1) + \oint(e_k)$ holds.

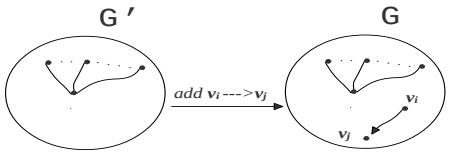

Figure 3: Add $v_i \rightarrow v_j$ to G', where v_i and v_j do not belong to G'

4. v_j belongs to G', but v_i is a newly added vertex (see Figure 4): This case will never occur because E has been sorted in the topological order. By the property of topological sorting, if the edges in E have been sorted in the topological order, any edge in E containing vertex with indegree 0 must appear before than those which does not contain vertex with indegree 0.

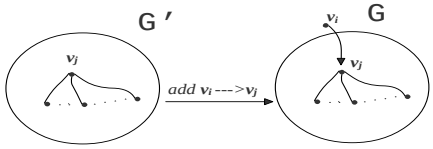

Figure 4: Add $v_i \rightarrow v_j$ to G', where v_i is a new vertex

□

Before describing the algorithm, we define three functions, which will be used to design the transitive closure algorithm.

Definition 3:

1. $adjacent(v_i)$: is a set of edges which are adjacent to vertex v_i.
2. $\mathcal{F}(E,k)$ is a recursive function defined as follows, which compute the transitive closure on the edges of E:

$$\mathcal{F}(E,k) = \begin{cases} \mathcal{F}(E - (v_i \rightarrow v_j), k-1) + \oint(v_i \rightarrow v_j) \\ \quad \text{if } v_i \text{ with indegree 0 and } v_i \rightarrow v_j \\ \quad \in adjacent(v_i); \\ \phi \text{ if } k = 0 \text{ or } E = \phi; \end{cases}$$

3. For each edge $v_i \rightarrow v_j \in E$, $\oint(v_i \rightarrow v_j)$ consists of two matrix operations defined as follows:

$$\oint(v_i \rightarrow v_j) = \{M \mid M[i,j] = 1 \text{ and } M[j\text{th column}] = M[i\text{th column}] + M[j\text{th column}]\}$$

Let $E = \{e_1, e_2, \cdots, e_k\}$ be a set of edges sorted by topological sorting. By induction, the transitive closure is easily obtained by the expression: $\sum_{i=1}^{k} \oint(e_i) = \oint(e_1) + \oint(e_2) + \cdots + \oint(e_k)$.

Theorem 2: Let $G = (V, E)$ be an acyclic graph and E has been sorted by topological sorting, where $|E| = k$ and $E = \{e_1, e_2, \cdots, e_k\}$. Then, the transitive closure matrix $\mathcal{F}(E,k) = \sum_{i=1}^{k} \oint(e_i) = \oint(e_1) + \oint(e_2) + \cdots + \oint(e_k)$, where $\mathcal{F}(E,k) \neq \phi$.

Proof. This proof is by induction on k.

Induction Base : Let $k = 1$ and $E = \{e_1\}$, where $e_1 = v_1 \rightarrow v_2$. By the definition 2, we can obtain:

$$\mathcal{F}(E,1) = \mathcal{F}(E - e_1, 0) + \oint(e_1) = \oint(e_1) = \begin{matrix} v_1 \\ v_2 \end{matrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

This is the same result as Warshall's algorithm has. Thus, $\mathcal{F}(E,1) = \mathcal{F}(E - e_1, 0) + \oint(e_1)$. Likewise, consider the base case $k = 2$: let $E = \{e_1, e_2\}$, where $e_1 = v_x \rightarrow v_y$ and $e_2 = v_z \rightarrow v_w$. According to topological order, we have to consider the following cases:

1. $v_y = v_z$: $\mathcal{F}(E,2)$ has the same result as Warshall's algorithm:

$$\mathcal{F}(E,2) = \mathcal{F}(E - e_2, 1) + \oint(e_2) = \mathcal{F}(E - e_2 - e_1, 0) + \oint(e_2) + \oint(e_1) = \begin{matrix} v_x \\ v_y = v_z \\ v_w \end{matrix} \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

2. $v_x = v_z$: we get $\mathcal{F}(E,2)$, which is the same result as Warshall's algorithm:

$$\mathcal{F}(E,2) = \mathcal{F}(E - e_2, 1) + \oint(e_2) = \mathcal{F}(E - e_2 - e_1, 0) + \oint(e_2) + \oint(e_1) = \begin{matrix} v_x = v_z \\ v_y \\ v_w \end{matrix} \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

3. $v_y = v_w$: $\mathcal{F}(E,2)$ has the same result as Warshall's algorithm:

$$\mathcal{F}(E,2) = \mathcal{F}(E - e_2 - e_1, 0) + \oint(e_2) + \oint(e_1) = \begin{matrix} v_x \\ v_y = v_w \\ v_z \end{matrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

4. $v_x \neq v_y$ and $v_z \neq v_w$. $\mathcal{F}(E,2)$ has the same result as Warshall's algorithm:

$$\mathcal{F}(E,2) = \mathcal{F}(E - e_2 - e_1, 0) + \oint(e_2) + \oint(e_1) = \begin{matrix} v_x \\ v_y \\ v_z \\ v_w \end{matrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

According to the above four cases, we can obtain that $\mathcal{F}(E,2) = \oint(e_1) + \oint(e_2)$ holds.

$$\begin{aligned} \mathcal{F}(E,2) &= \mathcal{F}(E - e_2, 1) + \oint(e_2) \\ &= \mathcal{F}(E - e_2 - e_1, 0) + \oint(e_2) + \oint(e_1) \\ &= \oint(e_2) + \oint(e_1) \end{aligned}$$

Induction hypothesis: For all j, $1 \leq j \leq k$, the transitive closure matrix $\mathcal{F}(E,j) = \sum_{i=1}^{j} \oint(e_i)$ hold.
Inductive Step: The transitive closure matrix $\mathcal{F}(E,k) = \sum_{i=1}^{k} \oint(e_i)$ is true by the induction hypothesis. Let $E' = E + e_{k+1}$. Now consider $\mathcal{F}(E', k+1)$:

$$\begin{aligned} \mathcal{F}(E', k+1) &= \mathcal{F}(E' - e_{k+1}, k) + \oint(e_{k+1}) \\ &= \mathcal{F}(E,k) + \oint(e_{k+1}) \\ &= \mathcal{F}(E - e_k, k-1) + \oint(e_{k+1}) + \oint(e_k) \\ &\vdots \\ &= \mathcal{F}(e_1, 1) + \sum_{i=1}^{k} \oint(e_i) \\ &= \sum_{i=1}^{k+1} \oint(e_i) \end{aligned}$$

□

Here we introduce the algorithm briefly. For an acyclic directed graph, we can find a vertex with indegree 0 by Lemma 1. Once we find it, remove it and adjust the indegree of its adjacent vertices for the resulting graph. For all its adjacent edges, execute matrix function $\oint(adjacent\ edge)$ to compute the transitive closure matrix step by step. Repeat the process until there are no more vertex with indegree 0. In Figure 5, we propose the fast transitive closure algorithm for the acyclic directed graph.

Algorithm 1: Acyclic Transitive Closure Matrix
Input: an acyclic directed graph $G = (V, E)$
Output: At the end, matrix $M_{n\times n}$ represents the transitive closure with respect to the directed graph G.
begin
Let Φ be the set of vertices with indegree 0
initialize the matrix $M_{n\times n}$ with zero (i.e., $M[i, j] = 0$ for $i, j = 1, \cdots, n$).
While Φ is not empty
  choose a vertex v from Φ (i.e., with indegree 0).
  $\Phi = \Phi - \{v\}$;
  for all edges $v \to w \in E$ leading out of v do
    remove $v \to w$ from E;
    if w with indegree 0, then $\Phi = \Phi \cup \{w\}$.
      M[v, w]=1;
      M[wth column] = M[wth column] + M[vth column];
    endif
end

Figure 5: The transitive closure algorithm for the acyclic directed graph

Next, we illustrate this algorithm with the graph of Figure 6-a. First, we have to sort the E in the topological order. Initially, we see that the first vertex to be picked is v_1 as it is the only one vertex with no predecessors. Thus, we remove v_1 and get $adjacent(v_1) = \{v_1 \to v_2, v_1 \to v_3\}$. Next, v_1 and $adjacent(v_1)$ are deleted. In the resulting graph Figure 6-b, v_2 and v_3 have no predecessor. Thus, we can pick v_2 and v_3 in sequence. Repeat the process until no more vertex left. In this example, $E = \{v_1 \to v_2, v_1 \to v_3, v_2 \to v_4, v_3 \to v_5, v_4 \to v_5\}$ is a set of the edges in topological order, where $|E|$=5. Thus, the transitive closure of G can be obtained by

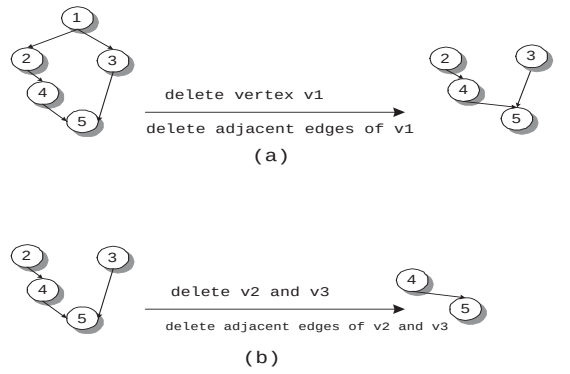

Figure 6: Illustrate the algorithm with an acyclic directed graph.

the expression: $M^+ = \sum_{i=1}^{5} \oint(e_i)$. In the following, we illustrate it with the all edges in E. Initially, $M_{n\times n}$ is set to zero. $v_1 \to v_2$ is first edge picked from E. Remove it from E, and add it to M. Then, $M[1, 2] = 1$ is set, and the 1st column of M is added into the 2nd column of M. The corresponding matrix is shown below:

$$M = \begin{matrix} v_1 \\ v_2 \end{matrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \overset{v_1\to v_2}{\longrightarrow} M = \begin{matrix} v_1 \\ v_2 \end{matrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Next, $v_1 \to v_3$ is added to M. $M[1, 3] = 1$ is set, and the 1st column of M is added to the 3nd column of M. The corresponding matrix is shown as follows.

$$M = \begin{matrix} v_1 \\ v_2 \end{matrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \overset{v_1\to v_3}{\longrightarrow} M = \begin{matrix} v_1 \\ v_2 \\ v_3 \end{matrix} \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Thirdly, $v_2 \to v_4$ and $v_3 \to v_5$ are picked. For the edge $v_2 \to v_4$, $M[2, 4] = 1$ is set, and the 2nd column of M is added to the 4th column of M. For the edge $v_3 \to v_5$, $M[3, 5] = 1$ is set, and the 3nd column of M is added to the 5th column of M.

$$M = \begin{matrix} v_1 \\ v_2 \\ v_3 \end{matrix} \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \overset{v_2\to v_4}{\longrightarrow} M = \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$M = \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \overset{v_3\to v_5}{\longrightarrow}$$

$$M = \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix} \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Finally, $v_4 \to v_5$ is picked. $M[4, 5] = 1$ is set, and the 4th column of M is added to the 5th column of M. Figure 7 shows the resulting transitive closure matrix.

4 Complexity Analysis

The time complexity of this algorithm consists of two parts. The first part is the topological sorting for the edges in E. For an acyclic directed graph $G = (V, E)$ with $|V| = n$ and $|E| = k$, the time complexity of topological sorting is $O(k + n)$, which is linear in the size of the input [2]. The second part is to analyze the time complexity of matrix function $\oint(e_i)$, for all the e_i in E. Since the matrix function $\oint(e_i)$

$$M = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{v_4 \rightarrow v_5}$$

$$M = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 7: The transitive closure matrix of the graph in Figure 5.

takes $O(n)$ time and there are k edges in E, this part requires $O(k \times n)$ time. Hence the overall running time of this algorithm is $O(k \times n)+O(k+n)$. For an acyclic directed graph, the size k is between $n-1$ to $\frac{n\times(n-1)}{2}$. When the acyclic directed graph is a tree structure, the number of edges k is $n-1$. The time complexity is $n \times (n-1) + 2 \times n - 1 = n^2 + n - 1$. Under this condition, the time complexity is $O(n^2)$. On the other hand, when the number of edges is $\frac{n\times(n-1)}{2}$, the time complexity of this algorithm is $\frac{n^2\times(n-1)}{2} + \frac{n\times(n-1)}{2} + n = \frac{n^3+1}{2}$, which requires half as much as the time complexity of Warshall's algorithm approximately. In this worst case, the time complexity takes $O(n^3)$. Although the worst case is almost the same as Warshall's algorithm, the average time complexity of our algorithm is between $O(n^2)$ and $O(n^3)$ and no matter what kind of acyclic directed graph it is, our algorithm is apparently faster than Warshall's algorithm.

5 Conclusion

Warshall proposed a transitive closure algorithm for any directed graph in 1962. However, no matter what kind of directed graph it is, this algorithm requires $O(n^3)$ time. In fact, there are many applications are just for acyclic directed graph. In this paper, we present a fast algorithm based on topological sorting to compute the transitive closure for the acyclic directed graph. This algorithm takes $O(k \times n)$ time, and the average time complexity is between $O(n^2)$ and $O(n^3)$. The advantage of this algorithm is that no matter what kind of acyclic directed graph it is, this algorithm is faster than Warshall's algorithm.

References

[1] Stephen Warshall., "A Theorem on Boolean Matrices," *Journal of the ACM,* Vol. 9, January 1962, pp.11-12.

[2] Ellis Horowize, Sartaj Sahni, Fundamentals of Data Structures in SPARKS, *Computer Sciences Press, Inc.*, pp. 292-300, 1983.

[3] Manber, Udi, *Introduction to Algorithms A Creative Approach,* Addison-Wesley, New York, 1988, pp.214-216.

[4] Prosser, Reese T., "Application of Boolean Matrices to the Analysis of Flow Diagrams," Proc. Eastern Joint Comput. Conf. No. 16(1959), pp. 133.

[5] Marimont, Rosalind B. " A New Method of Checking the Consistency of Precendence Matrices," Journal of ACM Vol. 6, (1959), pp. 164.