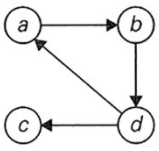
8.4 Warshall's and Floyd's Algorithms

DEFINITION The *transitive closure* of a directed graph with n vertices can be defined as the $n \times n$ boolean matrix $T = \{t_{ij}\}$, in which the element in the ith row and the jth column is 1 if there exists a nontrivial path (i.e., directed path of a positive length) from the ith vertex to the jth vertex; otherwise, t_{ij} is 0.



(a)

$A =$

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	0	1	0

(b)

$T =$

	a	b	c	d
a	1	1	1	1
b	1	1	1	1
c	0	0	0	0
d	1	1	1	1

(c)

FIGURE 8.11 (a) Digraph. (b) Its adjacency matrix. (c) Its transitive closure.

Since this method traverses the same digraph several times, we should hope that a better algorithm can be found. Indeed, such an algorithm exists. It is called *Warshall's algorithm* after Stephen Warshall, who discovered it [War62]. It is convenient to assume that the digraph's vertices and hence the rows and columns of the adjacency matrix are numbered from 1 to n. Warshall's algorithm constructs the transitive closure through a series of $n \times n$ boolean matrices:

$$R^{(0)}, \ldots, R^{(k-1)}, R^{(k)}, \ldots R^{(n)}. \qquad (8.9)$$

The central point of the algorithm is that we can compute all the elements of each matrix $R^{(k)}$ from its immediate predecessor $R^{(k-1)}$ in series (8.9). Let $r_{ij}^{(k)}$,the element in the ith row and jth column of matrix $R^{(k)}$, be equal to 1. This means that there exists a path from the ith vertex v_i to the jth vertex v_j with each intermediate vertex numbered not higher than k:

v_i, a list of intermediate vertices each numbered not higher than k, v_j. (8.10)

Two situations regarding this path are possible. In the first, the list of its intermediate vertices does not contain the *k*th vertex. Then this path from v_i to v_j has intermediate vertices numbered not higher than $k-1$, and therefore $r_{ij}^{(k-1)}$ is equal to 1 as well. The second possibility is that path (8.10) does contain the *k*th vertex v_k among the intermediate vertices. Without loss of generality, we may assume that v_k occurs only once in that list. (If it is not the case, we can create a new path from *vi* to v_j with this property by simply eliminating all the vertices between the First and last occurrences of v_k in it.) With this caveat, path (8.10) can be rewritten as follows:

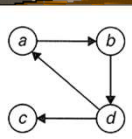
v_i, vertices numbered $\leq k-1$, v_k, vertices numbered $\leq k-1$, v_j.

What we have just proved is that if $r_{ij}^{(k)}$ =1, then either $r_{ij}^{(k-1)}$ =1 or both $r_{ik}^{(k-1)}=1$ and $r_{kj}^{(k-1)}=1$. It is easy to see that the converse of this assertion is also true. Thus, we have the following formula for generating the elements of matrix $R^{(k)}$ from the elements of matrix $R^{(k-1)}$:

$$r_{ij}^{(k)} = r_{ij}^{(k-1)} \quad \text{or} \quad \left(r_{ik}^{(k-1)} \quad \text{and} \quad r_{kj}^{(k-1)}\right). \qquad (8.11)$$

Formula (8.11) is at the heart of Warshall's algorithm. This formula implies the following rule for generating elements of matrix $R^{(k)}$ from elements of matrix $R^{(k-1)}$, which is particularly convenient for applying Warshall's algorithm by hand:

- If an element r_{ij} is 1 in $R^{(k-1)}$, it remains 1 in $R^{(k)}$.
- If an element r_{ij} is 0 in $R^{(k-1)}$, it has to be changed to 1 in $R^{(k)}$ if and only if the element in its row i and column k and the element in its column j and row k are both 1's in $R^{(k-1)}$. This rule is illustrated in Figure 8.12.



$R^{(0)}$ =

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	0	1	0

1's reflect the existence of paths with no intermediate vertices ($R^{(0)}$ is just the adjacency matrix); boxed row and column are used for getting $R^{(1)}$.

$R^{(1)}$ =

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	**1**	1	0

1's reflect the existence of paths with intermediate vertices numbered not higher than 1, i.e., just vertex a (note a new path from d to b); boxed row and column are used for getting $R^{(2)}$.

$R^{(2)}$ =

	a	b	c	d
a	0	1	0	**1**
b	0	0	0	1
c	0	0	0	0
d	1	1	1	**1**

1's reflect the existence of paths with intermediate vertices numbered not higher than 2, i.e., a and b (note two new paths); boxed row and column are used for getting $R^{(3)}$.

$R^{(3)}$ =

	a	b	c	d
a	0	1	0	1
b	0	0	0	1
c	0	0	0	0
d	1	1	1	1

1's reflect the existence of paths with intermediate vertices numbered not higher than 3, i.e., a, b, and c (no new paths); boxed row and column are used for getting $R^{(4)}$.

$R^{(4)}$ =

	a	b	c	d
a	**1**	1	**1**	1
b	**1**	**1**	**1**	1
c	0	0	0	0
d	1	1	1	1

1's reflect the existence of paths with intermediate vertices numbered not higher than 4, i.e., a, b, c, and d (note five new paths).

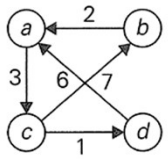
FIGURE 8.13 Application of Warshall's algorithm to the digraph shown. New 1's are in bold.

ALGORITHM *Warshall*($A[1..n, 1..n]$)
//Implements Warshall's algorithm for computing the transitive closure
//Input: The adjacency matrix A of a digraph with n vertices
//Output: The transitive closure of the digraph
$R^{(0)} \leftarrow A$
for $k \leftarrow 1$ to n do
  for $i \leftarrow 1$ to n do
    for $j \leftarrow 1$ to n do
      $R^{(k)}[i, j] \leftarrow R^{(k-1)}[i, j]$ or ($R^{(k-1)}[i, k]$ and $R^{(k-1)}[k, j]$)
return $R^{(n)}$

Floyd's Algorithm for the All-Pairs Shortest-Paths Problem

Given a weighted connected graph (undirected or directed), the *all-pairs shortest paths problem* asks to find the distances—i.e., the lengths of the shortest paths—from each vertex to all other vertices. This is one of several variations of the problem involving shortest paths in graphs. Because of its important applications to communications, transportation networks, and operations research, it has been thoroughly studied over the years. Among recent applications of the all-pairs shortest-path problem is precomputing distances for motion planning in computer games.

It is convenient to record the lengths of shortest paths in an $n \times n$ matrix D called the *distance matrix*: the element d_{ij} in the ith row and the jth column of this matrix indicates the length of the shortest path from the ith vertex to the jth vertex. For an example, see Figure 8.14.



(a)

$$W = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

(b)

$$D = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 7 & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{bmatrix} \end{array}$$

(c)

FIGURE 8.14 (a) Digraph. (b) Its weight matrix. (c) Its distance matrix.

We can generate the distance matrix with an algorithm that is very similar to Warshall's algorithm. It is called ***Floyd's algorithm*** after its co-inventor Robert W. Floyd.[1] It is applicable to both undirected and directed weighted graphs provided that **they do not contain a cycle of a negative length**. (The distance between any two vertices in such a cycle can be made arbitrarily small by repeating the cycle enough times.) The algorithm can be enhanced to find not only the lengths of the shortest paths for all vertex pairs but also the shortest paths themselves (Problem 10 in this section's exercises).

Floyd's algorithm computes the distance matrix of a weighted graph with n vertices through a series of $n \times n$ matrices:

$$D^{(0)}, \ldots, D^{(k-1)}, D^{(k)}, \ldots, D^{(n)}. \quad (8.12)$$

As in Warshall's algorithm, we can compute all the elements of each matrix $D^{(k)}$ from its immediate predecessor $D^{(k-1)}$ in series (8.12).

Let $d_{ij}^{(k)}$ be the element in the ith row and the jth column of matrix $D^{(k)}$. This means that $d_{ij}^{(k)}$ is equal to the length of the shortest path among all paths from the ith vertex v_i to the jth Vertex v_j with their intermediate vertices numbered not higher than k:

v_i, a list of intermediate vertices each numbered not higher than k, v_j. (8.13)

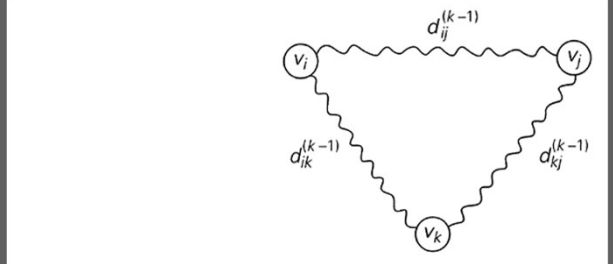


FIGURE 8.15 Underlying idea of Floyd's algorithm.

Since the length of the shortest path from v_i to v_k among the paths that use intermediate vertices numbered not higher than $k-1$ is equal to $d_{ik}^{(k-1)}$ and the length of the shortest path from v_k to v_j among the paths that use intermediate vertices numbered not higher than $k-1$ is equal to $d_{ij}^{(k-1)}$, the length of the shortest path among the paths that use the kth vertex is equal to $d_{ik}^{(k-1)} + d_{(kj}^{(k-1)}$. Taking into account the lengths of the shortest paths in both subsets leads to the following recurrence:

$$d_{ij}^{(k)} = \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\} \quad \text{for } k \geq 1, \quad d_{ij}^{(0)} = w_{ij}. \quad \textbf{(8.14)}$$

ALGORITHM *Floyd*(*W*[1..*n*, 1..*n*])

//Implements Floyd's algorithm for the all-pairs shortest-paths problem

//Input: The weight matrix *W* of a graph with no negative-length cycle

//Output: The distance matrix of the shortest paths' lengths

D **←*W*** **//is not necessary if *W* can be overwritten**

for *k*←1 to *n* do

  for *i* ←1 to *n* do

    for *j* ←1 to *n* do

      ***D*[*i, j*]←min{*D*[*i, j*], *D*[*i, k*]+ *D*[*k, j*]}**

return *D*

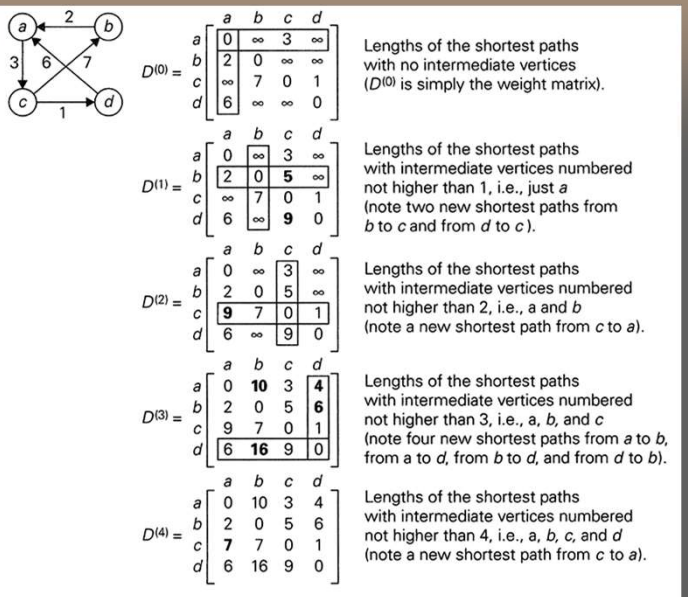


FIGURE 8.16 Application of Floyd's algorithm to the digraph shown. Updated elements are shown in bold.