FLOYD'S ALGORITHM (All-Pairs Shortest-Paths Problem)

Floyd'sl algorithm is an algorithm for finding shortest paths for all pairs in a weighted connected graph (undirected or directed) with (+/-) edge weights.

A **distance matrix** is a matrix (two-dimensional array) containing the distances, taken pairwise, between the vertices of graph.

The lengths of shortest paths in an n × 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.

We can generate the distance matrix with an algorithm that is very similar to Warshall's algorithm is called Floyd's algorithm.

Floyd's algorithm computes the distance matrix of a weighted graph with *n* vertices through a series of *n* × *n* matrices:

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

The element $d_{ij}^{(k)}$in the *i*th row and the *j*th column of matrix $D^{(k)}$ $(i, j = 1, 2, \ldots, n, k = 0,1,$

$\ldots, n)$ is equal to the length of the shortest path among all paths from the *i*th vertex to the *j*th vertex with each intermediate vertex, if any, numbered not higher than*k.*

Steps to compute $D^{(0)}, \ldots, D^{(k-1)}, D^{(k)}, \ldots, D^{(n)}$

- The series starts with $D^{(0)}$, which does not allow any intermediate vertices in its paths; hence, $D^{(0)}$is simply the weight matrix of thegraph.
- As in Warshall's algorithm, we can compute all the elements of each matrix $D^{(k)}$ from its immediate predecessor $D^{(k-1)}$.
- The last matrix in the series, $D^{(n)}$, contains the lengths of the shortest paths among all paths that can use all n vertices as intermediate and hence is nothing other than the distancematrix.

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 thank.

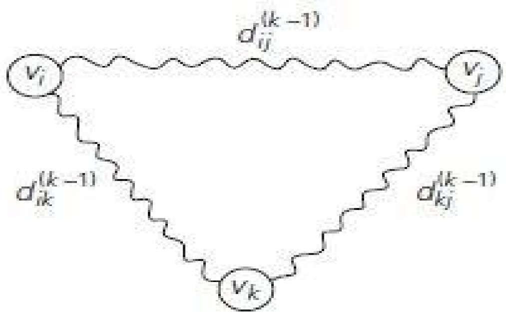


FIGURE 3.4 Underlying idea of Floyd's algorithm.

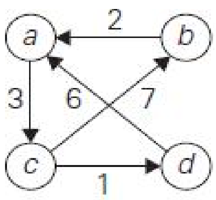
The length of the shortest path can be computed by the following recurrence:

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

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

Floyd's Algorithm's time efficiency is only Θ(n^3). Space efficiency is Θ(n^2). i.e matrix size.



$$D^{(0)} = \begin{array}{cc} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

Lengths of the shortest paths with no intermediate vertices ($D^{(0)}$ is simply the weight matrix).

$$D^{(1)} = \begin{array}{cc} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \mathbf{5} & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \mathbf{9} & 0 \end{bmatrix} \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 1, i.e., just ***a*** (note two new shortest paths from *b* to *c* and from *d* to *c*).

$$D^{(2)} = \begin{array}{cc} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ \mathbf{9} & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{bmatrix} \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 2, i.e., **a**and ***b*** (note a new shortest path from *c* to *a*).

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

Lengths of the shortest paths

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

with intermediate vertices numbered not higher than 3, i.e., **a**, ***b***, and ***c*** (note four new shortest paths from *a* to *b*, from a to *d*, from *b* to *d*, and from *d* to *b*).

Lengths of the shortest paths with intermediate vertices numbered not higher than 4, i.e., **a**, ***b***, ***c***, and ***d*** (note a new shortest path from *c* to *a*).

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