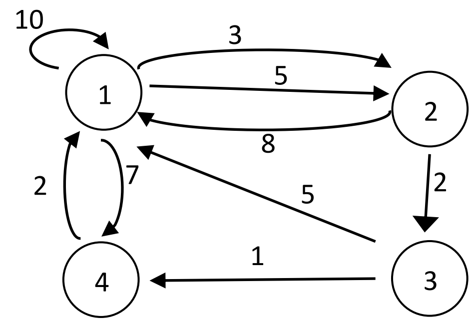
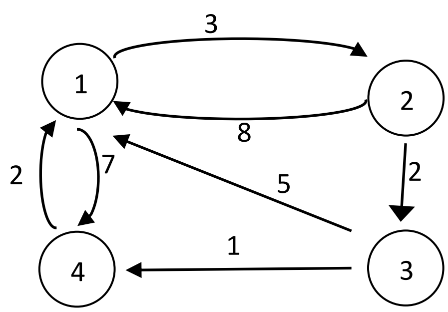
Shortest Path Finding: All-Pairs Shortest Path (Floyd-Warshall algorithm)



Step 1: Remove all Loops
Step 2: Remove all parallel edges, Keep the shortest one.



$$A^0 = \begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & \infty \\ 5 & \infty & 0 & 1 \\ 2 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

A^1 is calculated from A^0	Row 1 and Column 1 are fixed
A^2 is calculated from A^1	Row 2 and Column 2 are fixed
A^3 is calculated from A^2	Row 3 and Column 3 are fixed
A^4 is calculated from A^3	Row 4 and Column 4 are fixed

Paths through node 1: Calculating A^1 Row 1 and Column 1 are fixed

$$A^1 = \begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & & & \\ 5 & & & \\ 2 & & & \end{bmatrix} \end{array}$$

Path: 2 to 2 = $A^0[2][2] = 0$ Through 1: Path 2 to 1 + Path 1 to 2 = $A^0[2][1] + A^0[1][2]$
Path: 2 to 3 = $A^0[2][3] = 2$ Through 1: Path 2 to 1 + Path 1 to 3 = $A^0[2][1] + A^0[1][3]$
Path: 2 to 4 = $A^0[2][4] = \infty$ Through 1: Path 2 to 1 + Path 1 to 4 = $A^0[2][1] + A^0[1][4]$

$A^1[2][2] = \min(A^0[2][2], A^0[2][1] + A^0[1][2]) = \min(0, 8+3) = 0$
$A^1[2][3] = \min(A^0[2][3], A^0[2][1] + A^0[1][3]) = \min(2, 8+\infty) = 2$
$A^1[2][4] = \min(A^0[2][4], A^0[2][1] + A^0[1][4]) = \min(\infty, 8+7) = 15$

$$A^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & 15 \\ 5 & & & \\ 2 & & & \end{bmatrix} \end{array}$$

Path: 3 to 2 = $A^0[3][2] = \infty$ Through 1: Path 3 to 1 + Path 1 to 2 = $A^0[3][1] + A^0[1][2]$
Path: 3 to 3 = $A^0[3][3] = 0$ Through 1: Path 3 to 1 + Path 1 to 3 = $A^0[3][1] + A^0[1][3]$
Path: 3 to 4 = $A^0[3][4] = 1$ Through 1: Path 3 to 1 + Path 1 to 4 = $A^0[3][1] + A^0[1][4]$

$A^1[3][2] = \min(A^0[3][2], A^0[3][1] + A^0[1][2]) = \min(\infty, 5+3) = 8$
$A^1[3][3] = \min(A^0[3][3], A^0[3][1] + A^0[1][3]) = \min(0, 5+\infty) = 0$
$A^1[3][4] = \min(A^0[3][4], A^0[3][1] + A^0[1][4]) = \min(1, 5+7) = 1$

$$A^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & & & \end{bmatrix} \end{array}$$

Path: 4 to 2 = $A^0[4][2] = \infty$ Through 1: Path 4 to 1 + Path 1 to 2 = $A^0[4][1] + A^0[1][2]$
Path: 4 to 3 = $A^0[4][3] = \infty$ Through 1: Path 4 to 1 + Path 1 to 3 = $A^0[4][1] + A^0[1][3]$
Path: 4 to 4 = $A^0[4][4] = 0$ Through 1: Path 4 to 1 + Path 1 to 4 = $A^0[4][1] + A^0[1][4]$

$A^1[4][2] = \min(A^0[4][2], A^0[4][1] + A^0[1][2]) = \min(\infty, 2+3) = 5$
$A^1[4][3] = \min(A^0[4][3], A^0[4][1] + A^0[1][3]) = \min(\infty, 2+\infty) = \infty$
$A^1[4][4] = \min(A^0[4][4], A^0[4][1] + A^0[1][4]) = \min(0, 2+0) = 0$

$$A^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & \infty & 0 \end{bmatrix} \end{array}$$

Similarly the following can be calculated.

$$A^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & 5 & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix} \end{array}$$

$$A^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & 5 & 6 \\ 7 & 0 & 2 & 3 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix} \end{array}$$

$$A^4 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & 5 & 6 \\ 5 & 0 & 2 & 3 \\ 3 & 6 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix} \end{array}$$

```
// C++ Program for Floyd Warshall Algorithm
#include <bits/stdc++.h>
using namespace std;

#define V 4                    // Number of vertices in the graph
#define INF 99999

void printSolution(int dist[][V]);

void floydWarshall (int graph[][V])
{
   int dist[V][V], i, j, k;

   for (i = 0; i < V; i++)
      for (j = 0; j < V; j++)
         dist[i][j] = graph[i][j];

   for (k = 0; k < V; k++)                // Pick all vertices as source one by one
   {
      for (i = 0; i < V; i++)             // Pick all vertices as destination for the above picked source
      {
         for (j = 0; j < V; j++)          // If vertex k is on the shortest path from i to j, then update
```

```
            {
                if (dist[i][k] + dist[k][j] < dist[i][j])
                    dist[i][j] = dist[i][k] + dist[k][j];
            }
        }
    }

    printSolution(dist);
}

void printSolution(int dist[][V])
{
    cout<<"The following matrix shows the shortest distances"
            " between every pair of vertices \n";
    for (int i = 0; i < V; i++)
    {
        for (int j = 0; j < V; j++)
        {
            if (dist[i][j] == INF)
                cout<<"INF"<<"     ";
            else
                cout<<dist[i][j]<<"     ";
        }
        cout<<endl;
    }
}

int main()
{
    int graph[V][V] = { {0, 5, INF, 10},
        {INF, 0, 3, INF},
        {INF, INF, 0, 1},
        {INF, INF, INF, 0}
    };

    floydWarshall(graph);
    return 0;
}
```