

## Floyd-Warshall in C++

```
#include <bits/stdc++.h>
using namespace std;

class Solution {
public:
    void shortest_distance(vector<vector<int>>&matrix) {
        int n = matrix.size();
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (matrix[i][j] == -1) {
                    matrix[i][j] = 1e9;
                }
                if (i == j) matrix[i][j] = 0;
            }
        }

        for (int k = 0; k < n; k++) {
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    matrix[i][j] = min(matrix[i][j],
                                         matrix[i][k] + matrix[k][j]);
                }
            }
        }

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (matrix[i][j] == 1e9) {
                    matrix[i][j] = -1;
                }
            }
        }
    }
};

int main() {
    int V = 4;
    vector<vector<int>> matrix(V, vector<int>(V, -1));
    matrix[0][1] = 2;
    matrix[1][0] = 1;
    matrix[1][2] = 3;
    matrix[3][0] = 3;
    matrix[3][1] = 5;
    matrix[3][2] = 4;

    Solution obj;
    obj.shortest_distance(matrix);

    for (auto row : matrix) {
        for (auto cell : row) {
            cout << cell << " ";
        }
        cout << endl;
    }

    return 0;
}
```

### Objective

You are given a directed weighted graph in the form of an **adjacency matrix**. You are using the **Floyd-Warshall algorithm** to compute **shortest distances between every pair of vertices**.

### ★ Input Matrix (after setup)

The initial matrix setup (after setting the given edges):

```

    0  1  2  3
0 | -1  2 -1 -1
1 |  1 -1  3 -1
2 | -1 -1 -1 -1
3 |  3  5  4 -1
```

Converted to:

```

    0  1  2  3
0 |  0  2 1e9 1e9
1 |  1  0  3 1e9
2 | 1e9 1e9  0 1e9
3 |  3  5  4  0
```

### 🧠 Floyd-Warshall Algorithm Dry Run

We'll now go through each intermediate node  $k$  and update the matrix.

#### 🔄 For $k = 0$

Try to go  $i \rightarrow 0 \rightarrow j$

No new updates help here, as 0 is only connected to 1.

#### 🔄 For $k = 1$

Try  $i \rightarrow 1 \rightarrow j$ :

- $0 \rightarrow 1 \rightarrow 2 = 2 + 3 = 5 \rightarrow$  Update  $\text{matrix}[0][2]$  from  $1e9 \rightarrow 5$
- $3 \rightarrow 1 \rightarrow 2 = 5 + 3 = 8 \rightarrow$  Update  $\text{matrix}[3][2]$  from  $4 \rightarrow 4$  (already smaller, no change)

🔄 For k = 2

Only relevant updates:

- $3 \rightarrow 2 \rightarrow 0 = 4 + 1e9 \rightarrow$  no update
- Nothing meaningful added as 2 is a disconnected node

🔄 For k = 3

- $0 \rightarrow 3 \rightarrow 0 \rightarrow$  Not reachable
- But let's try:
  - $0 \rightarrow 3 \rightarrow 2$ :  $\text{matrix}[0][3] + \text{matrix}[3][2] = 1e9 + 4 = 1e9 \rightarrow$  No update
  - Same for others, no improvement.

✓ Final Matrix (replace 1e9 with -1)

```
0 2 5 -1
1 0 3 -1
-1 -1 0 -1
3 5 4 0
```

📄 Output

```
0 2 5 -1
1 0 3 -1
-1 -1 0 -1
3 5 4 0
```

**Output:-**

```
0 2 5 -1
1 0 3 -1
-1 -1 0 -1
3 5 4 0
```