

Session 9

Floyd and Warshall's Algorithm for All Pair Shortest Path Algorithms

This session is about finding shortest paths between every pair of vertices in a graph. Unlike Dijkstra's algorithm, which starts from one source, Floyd-Warshall works with a distance matrix and gradually improves every source-destination pair.

Objectives

✕ Do not copy. Read for understanding and the viva

- Understand all-pairs shortest path computation.
- Apply Floyd-Warshall algorithm using distance matrices.
- Implement the algorithm for different graphs.

Concept

✕ Do not copy. Read for understanding and the viva

Floyd-Warshall updates a distance matrix by allowing each vertex to become an intermediate vertex.

$$1 \quad D[k][i][j] = \min(D[k-1][i][j], D[k-1][i][k] + D[k-1][k][j])$$

Algorithm

■ Write in lab record

```
1 for k from 1 to n:
2     for i from 1 to n:
3         for j from 1 to n:
4             D[i][j] = min(D[i][j], D[i][k] + D[k][j])
```

Complexity

✕ Do not copy. Read for understanding and the viva

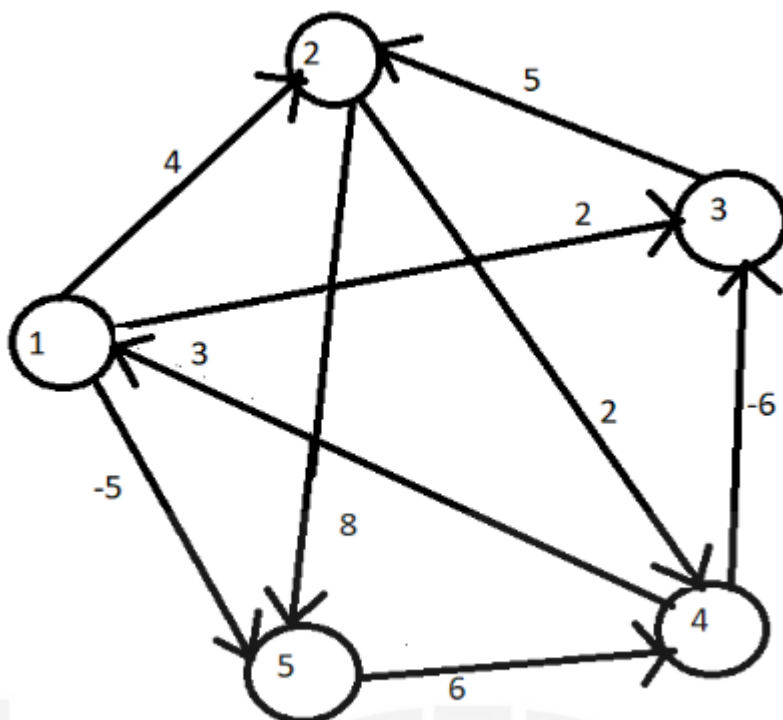
Metric	Value
Time complexity	$O(n^3)$
Space complexity	$O(n^2)$

Question 1

Problem Statement

Write in lab record

Apply Floyd-Warshall's algorithm for the given graph. Show the matrix D^5 and find the shortest path.



Answer

■ Write in lab record

Here is the step-by-step application of the Floyd-Warshall algorithm for the given graph, followed by the code implementations.

Graph Analysis

✕ Do not copy. Read for understanding and the viva

First, we identify the nodes and edges with their weights from the image.

- **Nodes:** 1, 2, 3, 4, 5
- **Edges:**
 - $1 \rightarrow 2$: 4
 - $1 \rightarrow 3$: 2
 - $1 \rightarrow 5$: -5
 - $2 \rightarrow 4$: 2
 - $2 \rightarrow 5$: 8
 - $3 \rightarrow 2$: 5
 - $4 \rightarrow 1$: 3
 - $4 \rightarrow 3$: 6
 - $5 \rightarrow 4$: 6

Initial Matrix (D^0)

■ Write in lab record

We represent this as an adjacency matrix where ∞ represents no direct edge.

$$D^0 = \begin{bmatrix} 0 & 4 & 2 & \infty & -5 \\ \infty & 0 & \infty & 2 & 8 \\ \infty & 5 & 0 & \infty & \infty \\ 3 & \infty & 6 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Floyd-Warshall Iterations

Write in lab record

The formula is: $D^{(k)}[i][j] = \min(D^{(k-1)}[i][j], D^{(k-1)}[i][k] + D^{(k-1)}[k][j])$

Iteration $k = 1$ (Intermediate Node 1)

Write in lab record

- **Path 4 $\rightarrow$ 2:** $\min(D^{(0)}[4][2], D^{(0)}[4][1] + D^{(0)}[1][2]) = \min(\infty, 3 + 4) = 7$
- **Path 4 $\rightarrow$ 3:** $\min(D^{(0)}[4][3], D^{(0)}[4][1] + D^{(0)}[1][3]) = \min(6, 3 + 2) = 5$
- **Path 4 $\rightarrow$ 5:** $\min(D^{(0)}[4][5], D^{(0)}[4][1] + D^{(0)}[1][5]) = \min(\infty, 3 + (-5)) = -2$

Iteration $k = 2$ (Intermediate Node 2)

Write in lab record

- **Path 1 $\rightarrow$ 4:** $\min(D^{(1)}[1][4], D^{(1)}[1][2] + D^{(1)}[2][4]) = \min(\infty, 4 + 2) = 6$
- **Path 3 $\rightarrow$ 4:** $\min(D^{(1)}[3][4], D^{(1)}[3][2] + D^{(1)}[2][4]) = \min(\infty, 5 + 2) = 7$
- **Path 3 $\rightarrow$ 5:** $\min(D^{(1)}[3][5], D^{(1)}[3][2] + D^{(1)}[2][5]) = \min(\infty, 5 + 8) = 13$

Iteration $k = 3$ (Intermediate Node 3)

Write in lab record

No significant changes lower the current minimums. For example, for path $1 \rightarrow 2$, the current value is 4. The alternate path via node 3 yields: $D^{(2)}[1][3] + D^{(2)}[3][2] = 2 + 5 = 7$. Since $4 < 7$, no update is made.

Iteration $k = 4$ (Intermediate Node 4)

■ Write in lab record

- **Path $2 \rightarrow 1$:** $\min(D^{(3)}[2][1], D^{(3)}[2][4] + D^{(3)}[4][1]) = \min(\infty, 2 + 3) = 5$
- **Path $2 \rightarrow 3$:** $\min(D^{(3)}[2][3], D^{(3)}[2][4] + D^{(3)}[4][3]) = \min(\infty, 2 + 5) = 7$
- **Path $2 \rightarrow 5$:** $\min(D^{(3)}[2][5], D^{(3)}[2][4] + D^{(3)}[4][5]) = \min(8, 2 + (-2)) = 0$
- **Path $3 \rightarrow 1$:** $\min(D^{(3)}[3][1], D^{(3)}[3][4] + D^{(3)}[4][1]) = \min(\infty, 7 + 3) = 10$
- **Path $3 \rightarrow 5$:** $\min(D^{(3)}[3][5], D^{(3)}[3][4] + D^{(3)}[4][5]) = \min(13, 7 + (-2)) = 5$
- **Path $5 \rightarrow 1$:** $\min(D^{(3)}[5][1], D^{(3)}[5][4] + D^{(3)}[4][1]) = \min(\infty, 6 + 3) = 9$
- **Path $5 \rightarrow 2$:** $\min(D^{(3)}[5][2], D^{(3)}[5][4] + D^{(3)}[4][2]) = \min(\infty, 6 + 7) = 13$
- **Path $5 \rightarrow 3$:** $\min(D^{(3)}[5][3], D^{(3)}[5][4] + D^{(3)}[4][3]) = \min(\infty, 6 + 5) = 11$

Iteration $k = 5$ (Intermediate Node 5)

■ Write in lab record

- **Path $1 \rightarrow 4$:** $\min(D^{(4)}[1][4], D^{(4)}[1][5] + D^{(4)}[5][4]) = \min(6, -5 + 6) = 1$

Final Distance Matrix $D^{(5)}$

■ Write in lab record

The final matrix contains the shortest path distances calculated between all pairs of nodes:

$$D^{(5)} = \begin{bmatrix} 0 & 4 & 2 & 1 & -5 \\ 5 & 0 & 7 & 2 & 0 \\ 10 & 5 & 0 & 7 & 5 \\ 3 & 7 & 5 & 0 & -2 \\ 9 & 13 & 11 & 6 & 0 \end{bmatrix}$$

Shortest Path Example:

■ Write in lab record

The shortest path from Node 1 to Node 4 is 1.

- Path: $1 - > 5 - > 4$
- Cost: $-5 + 6 = 1$

Implementation

■ Write in lab record

■

Lab record: write one language only. Pick yours once and every page opens on it; the other tabs are the same solution for comparison.

Python

floyd-warshall-algorithm.py

```
1 # Number of vertices in the graph
2 V = 5
3
4 # Define infinity as a large enough value.
5 INF = 99999
6
7
8 def floydWarshall(graph):
9     """
10     Implements the Floyd-Warshall algorithm to find all pairs shortest paths
11     """
12     # dist[][] will be the output matrix that will finally have the shortest
13     # distances between every pair of vertices
14     dist = list(map(lambda i: list(map(lambda j: j, i)), graph))
15
16     # Add all vertices one by one to the set of intermediate vertices.
17     # k is the intermediate vertex
18     for k in range(V):
19         # i is the source vertex
20         for i in range(V):
21             # j is the destination vertex
22             for j in range(V):
23                 # If vertex k is on the shortest path from i to j,
24                 # then update the value of dist[i][j]
25                 dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])
26
27     print("Shortest distance matrix (D5):")
28     for i in range(V):
29         for j in range(V):
30             if dist[i][j] == INF:
31                 print("%7s" % ("INF"), end=" ")
32             else:
33                 print("%7d" % (dist[i][j]), end=" ")
34     print()
35
36
37 # Driver program to test the above program
38 # Mapping: 1→0, 2→1, 3→2, 4→3, 5→4
39 graph = [
40     [0, 4, 2, INF, -5],
41     [INF, 0, INF, 2, 8],
42     [INF, 5, 0, INF, INF],
43     [3, INF, 6, 0, INF],
```

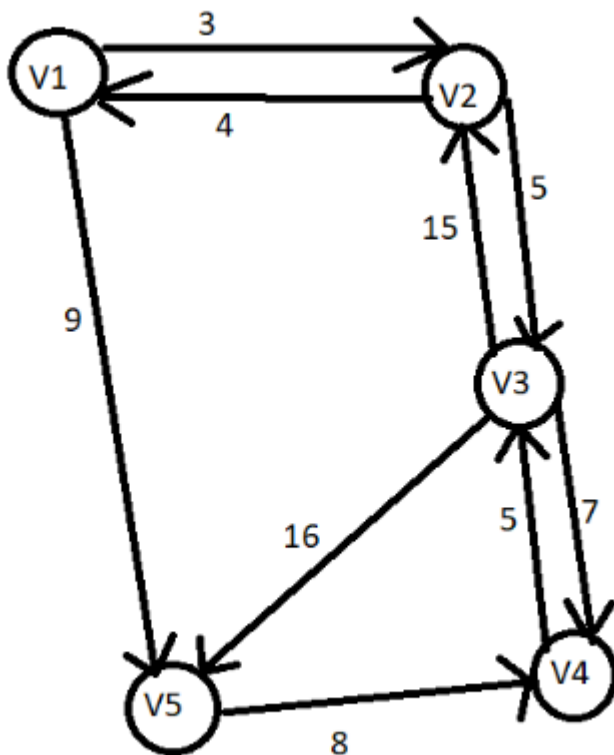
```
44     [INF, INF, INF, 6, 0],  
45 ]  
46  
47 floydWarshall(graph)
```

Question 2

Problem Statement

■ Write in lab record

Apply Floyd-Warshall's algorithm to compute the shortest path for the second graph. Compute the D^5 matrix.



Answer

■ Write in lab record

Graph Analysis

✗ Do not copy. Read for understanding and the viva

First, we identify the vertices and the weighted directed edges from the image:

- **Vertices:** $V = 1, 2, 3, 4, 5$
- **Edges:**
 - $1 \rightarrow 2$: weight 3
 - $2 \rightarrow 1$: weight 4
 - $1 \rightarrow 5$: weight 9
 - $2 \rightarrow 3$: weight 15
 - $2 \rightarrow 4$: weight 5
 - $4 \rightarrow 2$: weight 7
 - $4 \rightarrow 3$: weight 5
 - $5 \rightarrow 3$: weight 16
 - $5 \rightarrow 4$: weight 8

Initial Matrix (D^0)

■ Write in lab record

We represent this as an adjacency matrix where ∞ represents no direct edge.

$$D^{(0)} = \begin{bmatrix} 0 & 3 & \infty & \infty & 9 \\ 4 & 0 & 15 & 5 & \infty \\ \infty & \infty & 0 & \infty & \infty \\ \infty & 7 & 5 & 0 & \infty \\ \infty & \infty & 16 & 8 & 0 \end{bmatrix}$$

Floyd-Warshall Iterations

■ Write in lab record

Iteration $k = 1$ (Intermediate Node 1)

■ Write in lab record

We check if going through Node 1 offers a shorter path.

- **Path 2 → 5:**

$$D^{(0)}[2][1] + D^{(0)}[1][5] = 4 + 9 = 13 \quad (\text{Updates } \infty \rightarrow 13)$$

$$D^{(1)} = \begin{bmatrix} 0 & 3 & \infty & \infty & 9 \\ 4 & 0 & 15 & 5 & 13 \\ \infty & \infty & 0 & \infty & \infty \\ \infty & 7 & 5 & 0 & \infty \\ \infty & \infty & 16 & 8 & 0 \end{bmatrix}$$

Iteration $k = 2$ (Intermediate Node 2)

■ Write in lab record

We check paths through Node 2.

- **Path 1 → 3:**

$$D^{(1)}[1][2] + D^{(1)}[2][3] = 3 + 15 = 18$$

- **Path 1 → 4:**

$$D^{(1)}[1][2] + D^{(1)}[2][4] = 3 + 5 = 8$$

- **Path 4 → 1:**

$$D^{(1)}[4][2] + D^{(1)}[2][1] = 7 + 4 = 11$$

- **Path 4 → 5:**

$$D^{(1)}[4][2] + D^{(1)}[2][5] = 7 + 13 = 20$$

$$D^{(2)} = \begin{bmatrix} 0 & 3 & 18 & 8 & 9 \\ 4 & 0 & 15 & 5 & 13 \\ \infty & \infty & 0 & \infty & \infty \\ 11 & 7 & 5 & 0 & 20 \\ \infty & \infty & 16 & 8 & 0 \end{bmatrix}$$

Iteration $k = 3$ (Intermediate Node 3)

■ Write in lab record

Node 3 has no outgoing edges (Row 3 contains all ∞ values except for the diagonal position), meaning no shortest paths can be improved by passing through Node 3.

$$D^{(3)} = D^{(2)}$$

Iteration $k = 4$ (Intermediate Node 4)

■ Write in lab record

We check paths through Node 4.

- **Path $1 \rightarrow 3$:**

$$\min(18, D^{(3)}[1][4] + D^{(3)}[4][3]) = \min(18, 8 + 5) = 13$$

- **Path $2 \rightarrow 3$:**

$$\min(15, D^{(3)}[2][4] + D^{(3)}[4][3]) = \min(15, 5 + 5) = 10$$

- **Path $5 \rightarrow 1$:**

$$\min(\infty, D^{(3)}[5][4] + D^{(3)}[4][1]) = \min(\infty, 8 + 11) = 19$$

- **Path $5 \rightarrow 2$:**

$$\min(\infty, D^{(3)}[5][4] + D^{(3)}[4][2]) = \min(\infty, 8 + 7) = 15$$

- **Path $5 \rightarrow 3$:**

$$\min(16, D^{(3)}[5][4] + D^{(3)}[4][3]) = \min(16, 8 + 5) = 13$$

$$D^{(4)} = \begin{bmatrix} 0 & 3 & 13 & 8 & 9 \\ 4 & 0 & 10 & 5 & 13 \\ \infty & \infty & 0 & \infty & \infty \\ 11 & 7 & 5 & 0 & 20 \\ 19 & 15 & 13 & 8 & 0 \end{bmatrix}$$

Iteration $k = 5$ (Intermediate Node 5)

■ Write in lab record

We check paths through Node 5. None of the existing paths are improved by going through Node 5 (e.g., path $1 \rightarrow 2$ costs 3, whereas a path via node 5 would cost $9 + 15 = 24$).

$$D^{(5)} = D^{(4)}$$

Final Matrix $D^{(5)}$ and Shortest Paths

■ Write in lab record

The final matrix $D^{(5)}$ represents the absolute shortest distance between every pair of nodes:

$$D^{(5)} = \begin{bmatrix} 0 & 3 & 13 & 8 & 9 \\ 4 & 0 & 10 & 5 & 13 \\ \infty & \infty & 0 & \infty & \infty \\ 11 & 7 & 5 & 0 & 20 \\ 19 & 15 & 13 & 8 & 0 \end{bmatrix}$$

Shortest Paths:

■ Write in lab record

- **Shortest path from $V_1 \rightarrow V_3$:**
 - Total Distance: 13
 - Path Traversed: $1 \rightarrow 2 \rightarrow 4 \rightarrow 3$ (Cost computation: $3 + 5 + 5 = 13$)
- **Shortest path from $V_5 \rightarrow V_1$:**
 - Total Distance: 19
 - Path Traversed: $5 \rightarrow 4 \rightarrow 2 \rightarrow 1$ (Cost computation: $8 + 7 + 4 = 19$)

Implementation

■ Write in lab record

■

Lab record: write one language only. Pick yours once and every page opens on it; the other tabs are the same solution for comparison.

Python

floyd-warshall-algorithm.py

```
1 # Number of vertices in the graph
2 V = 5
3
4 # Define infinity as a large enough value.
5 INF = 99999
6
7
8 def floydWarshall(graph):
9     """
10     Implements the Floyd-Warshall algorithm to find all pairs shortest paths
11     """
12     # dist[][] will be the output matrix that will finally have the shortest
13     # distances between every pair of vertices
14     dist = list(map(lambda i: list(map(lambda j: j, i)), graph))
15
16     # Add all vertices one by one to the set of intermediate vertices.
17     # k is the intermediate vertex
18     for k in range(V):
19         # i is the source vertex
20         for i in range(V):
21             # j is the destination vertex
22             for j in range(V):
23                 # If vertex k is on the shortest path from i to j,
24                 # then update the value of dist[i][j]
25                 dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])
26
27     print("Shortest distance matrix (D5):")
28     for i in range(V):
29         for j in range(V):
30             if dist[i][j] == INF:
31                 print("%7s" % ("INF"), end=" ")
32             else:
33                 print("%7d" % (dist[i][j]), end=" ")
34     print()
35
36
37 # Driver program to test the above program
38 # Mapping: V1→0, V2→1, V3→2, V4→3, V5→4
39 graph = [
40     [0, 3, INF, INF, 9],
41     [4, 0, 15, 5, INF],
42     [INF, INF, 0, INF, INF],
43     [INF, 7, 5, 0, INF],
```

```
44     [INF, INF, 16, 8, 0],  
45 ]  
46  
47 floydWarshall(graph)
```

Question 3

Problem Statement

■ Write in lab record

Implement the all-pair shortest path algorithm using different graphs.

Answer

■ Write in lab record

This algorithm works by iteratively improving the estimate of the shortest path between two nodes by considering every other node as a potential intermediate point.

The Algorithm Logic

✗ Do not copy. Read for understanding and the viva

- Initialize the distance matrix `dist` with the direct edge weights (adjacency matrix).
- Loop through every node `k` (from 0 to V-1) to treat it as an intermediate node.
- For every pair of nodes `(i, j)`, check if the path `i → k → j` is shorter than the current known path `i → j`.
- Update `dist[i][j]` if a shorter path is found.

■

Lab record: write one language only. Pick yours once and every page opens on it; the other tabs are the same solution for comparison.

Python

floyd-warshall-algorithm.py

```
1 # Number of vertices in the graph
2 V = 5
3
4
5 def floydWarshall(graph):
6     """
7     Implements the Floyd-Warshall algorithm to find all pairs shortest paths
8     """
9     # dist[][] will be the output matrix that will finally have the shortest
10    # distances between every pair of vertices.
11    # We create a deep copy to avoid modifying the original graph structure.
12    dist = [row[:] for row in graph]
13
14    # Add all vertices one by one to the set of intermediate vertices.
15    # k is the intermediate vertex
16    for k in range(V):
17        # i is the source vertex
18        for i in range(V):
19            # j is the destination vertex
20            for j in range(V):
21                # If vertex k is on the shortest path from i to j,
22                # then update the value of dist[i][j]
23                # We check if dist[i][k] and dist[k][j] are not infinity
24                if dist[i][k] != float("inf") and dist[k][j] != float("inf"):
25                    if dist[i][k] + dist[k][j] < dist[i][j]:
26                        dist[i][j] = dist[i][k] + dist[k][j]
27
28    print("Shortest distance matrix (D5):")
29    for i in range(V):
30        for j in range(V):
31            if dist[i][j] == float("inf"):
32                print("%7s" % ("INF"), end=" ")
33            else:
34                print("%7d" % (dist[i][j]), end=" ")
35    print()
36
37
38 # Driver program to test the above program
39 # Mapping: V1→0, V2→1, V3→2, V4→3, V5→4
40 graph = [
41     [0, 3, float("inf"), float("inf"), 9],
42     [4, 0, 15, 5, float("inf")],
43     [float("inf"), float("inf"), 0, float("inf"), float("inf")],
```

```
44     [float("inf"), 7, 5, 0, float("inf")],  
45     [float("inf"), float("inf"), 16, 8, 0],  
46 ]  
47  
48 floydWarshall(graph)
```

[✎ Edit this page](#)

[< Previous](#)
Session 8

Session 10 [Next >](#)