Question No: 1

Multi Choice Type Question

Floyd- Warshall algorithm was proposed by ____________

- ○ Stephen Floyd and Robert Warshall
- [✓ Robert Floyd and Stephen Warshall] ✓
- ○ Bernad Floyd and Robert Warshall
- ○ Robert Floyd and Bernad Warshall

Status	**Correct**	Mark obtained	**1/1**	Hints used	**0**	Level	**Medium**	Question type	**MCQ Single Correct**	Subject	**Algorithms**	Topic	**Graph**	Sub Topic

Question No: 2

Multi Choice Type Question

What is the main objective of a shortest-path algorithm?

- ○ To find the path with the fewest edges in a graph
- [✓ To find the most efficient path between two points in a graph] ✓

Question No: 3

Multi Choice Type Question

Which of the following is a disadvantage of using Floyd-Warshall algorithm compared to Dijkstra's algorithm for finding the shortest path in a graph?

- Higher time complexity ✓
- Only applicable to directed graphs
- Cannot handle negative edge weights
- Only applicable to unweighted graphs

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Easy** | Question type **MCQ Single Correct** | Subject **Algorithms** | Topic **Graph** | Sub Topic **Grap**

Question No: 4

Multi Choice Type Question

In the Floyd Warshall algorithm, how many iterations are required to find the shortest path between all pairs of vertices in a graph with N vertices?

- N+1
- N^2
- N ✓

Question No: 5

Multi Choice Type Question

What is the time complexity of the Floyd-Warshall algorithm for finding the all pairs shortest path in a graph with n nodes?

- O(n log n)
- O(n^2)
- O(n)
- [✓ O(n^3)]

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Easy** | Question type **MCQ Single Correct** | Subject **Algorith**

Question No: 6

Multi Choice Type Question

In the Floyd-Warshall algorithm, what does it mean if the path matrix contains a value ∞ for a pair of nodes?

- The nodes have a direct edge with weight ∞
- The path is of infinite length
- None of the mentioned options
- [✓ There is no path between the nodes]

Question No: 7

Multi Choice Type Question

Which of the following is a critical difference between Dijkstra's algorithm and the Floyd-Warshall algorithm?

- ○ The Floyd-Warshall algorithm uses a priority queue.
- ○ Dijkstra's algorithm can handle graphs with negative edge weights.
- ○ The Floyd-Warshall algorithm is a single-source algorithm.
- [✓ Dijkstra's algorithm always finds the shortest path to all nodes.]

Status	Correct	Mark obtained	1/1	Hints used	0	Level	Medium	Question type	MCQ Single Correct

Question No: 8

Multi Choice Type Question

Floyd Warshall Algorithm can be used for finding _____________

- [✓ Transitive closure]

Question No: 9

Multi Choice Type Question

Which of the following algorithms solves the single-source shortest paths?

- [✓ Dijkstra's Algorithm]
- Floyd-Warshall Algorithm
- Krushkal's Algorithm
- Prims Algorithm

Status	**Correct**	Mark obtained	**1/1**	Hints used	**0**	Level	**Medium**	Question type	**MCQ Single**

Question No: 10

Multi Choice Type Question

Which of the following algorithms can find the shortest path between all pairs of nodes in a graph?

- [✓ Floyd-Warshall algorithm]

Question No: 11

Multi Choice Type Question

A search algorithm is an example of which type of shortest path algorithm?

- Dynamic programming algorithm
- Greedy algorithm
- Dijkstra's algorithm
- [✓ Heuristic algorithm]

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Medium** | Question typ

Question No: 12

Multi Choice Type Question

In the context of shortest path algorithms, what does "edge weight" represent?

- [✓ The cost or distance associated with traversing an edge]

Question No: 13

Multi Choice Type Question

Which data structure is often used to implement the Floyd-Warshall algorithm?

- [✓ Array]
- Linked List
- Hash Table
- None of the mentioned options

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Easy** | Questio

Question No: 14

Multi Choice Type Question

What happens when the value of k is 0 in the Floyd Warshall Algorithm?

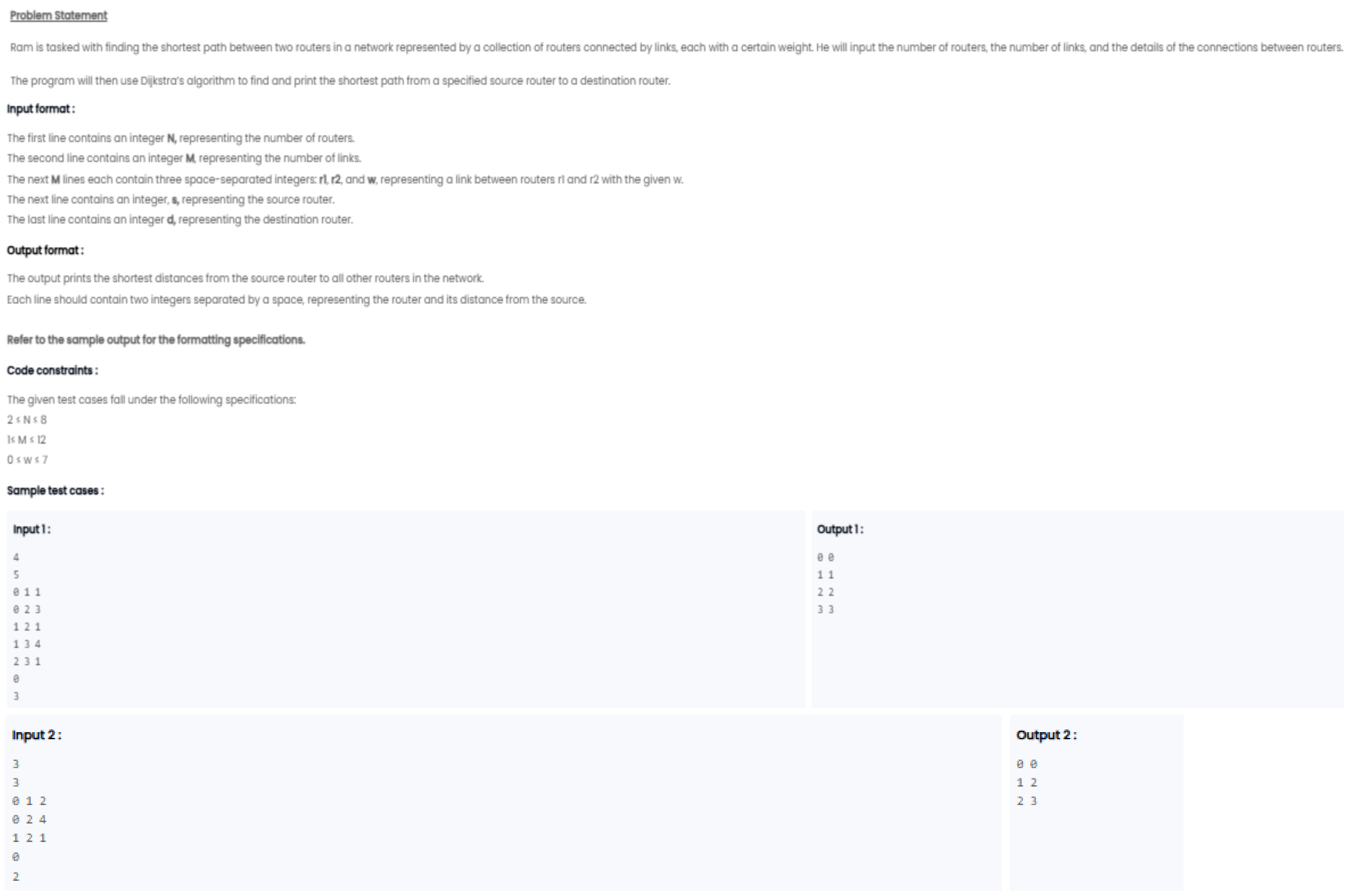
- [✓ 0 intermediate vertex]

Question No: 15

Multi Choice Type Question

What is the time complexity of the Floyd Warshall Algorithm, where V is the number of vertices in the graph?

- [✓ $O(V^3)$]



Problem Statement

Ram is tasked with finding the shortest path between two routers in a network represented by a collection of routers connected by links, each with a certain weight. He will input the number of routers, the number of links, and the details of the connections between routers.

The program will then use Dijkstra's algorithm to find and print the shortest path from a specified source router to a destination router.

Input format :

The first line contains an integer **N,** representing the number of routers.
The second line contains an integer **M,** representing the number of links.
The next **M** lines each contain three space-separated integers: **r1**, **r2**, and **w**, representing a link between routers r1 and r2 with the given w.
The next line contains an integer, **s,** representing the source router.
The last line contains an integer **d,** representing the destination router.

Output format :

The output prints the shortest distances from the source router to all other routers in the network.
Each line should contain two integers separated by a space, representing the router and its distance from the source.

Refer to the sample output for the formatting specifications.

Code constraints :

The given test cases fall under the following specifications:
2 ≤ N ≤ 8
1≤ M ≤ 12
0 ≤ w ≤ 7

Sample test cases :

Input 1 :

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

Output 1 :

```
0 0
1 1
2 2
3 3
```

Input 2 :

```
3
3
0 1 2
0 2 4
1 2 1
0
2
```

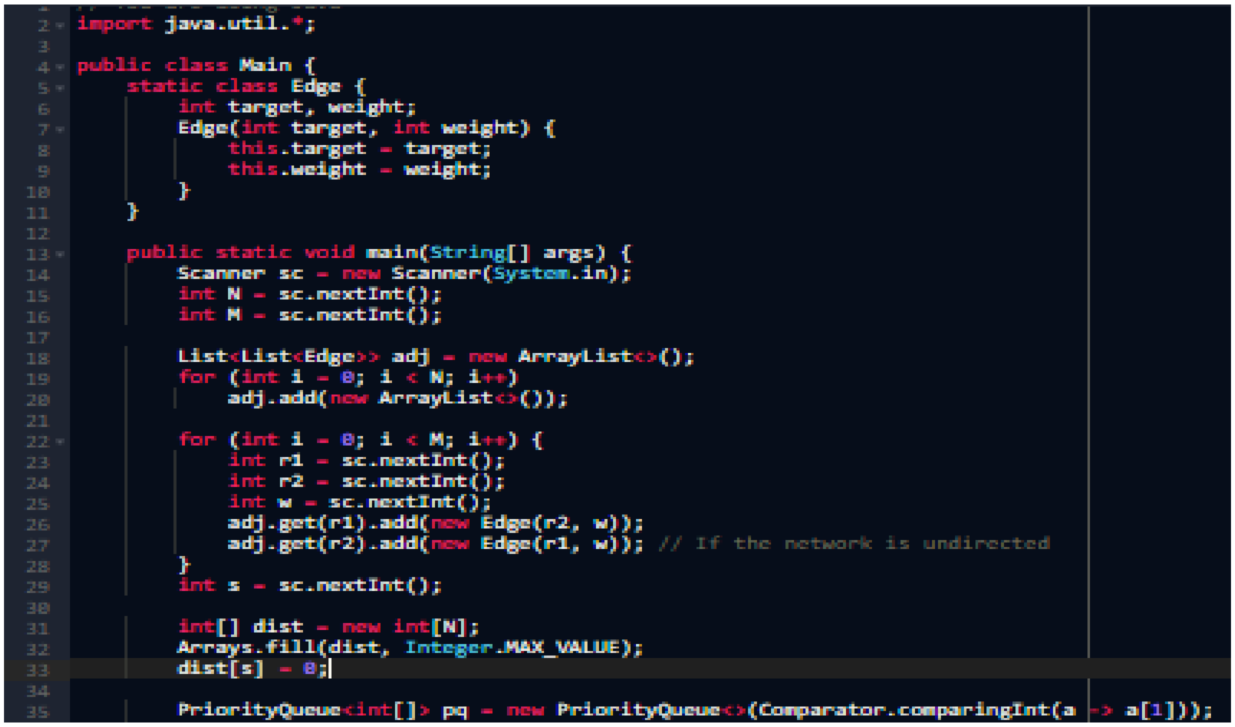
Output 2 :

```
0 0
1 2
2 3
```

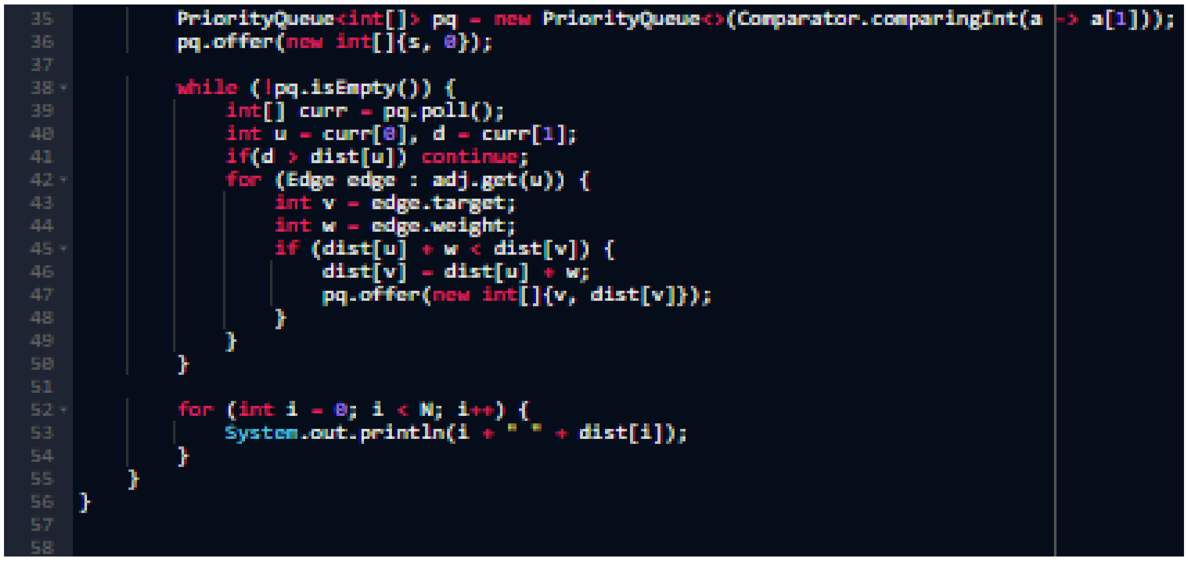
```
// You are using Java

import java.util.*;



public class Main {

    static class Edge {

        int target, weight;

        Edge(int target, int weight) {

            this.target = target;

            this.weight = weight;

        }

    }
```

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int N = sc.nextInt();
    int M = sc.nextInt();

    List<List<Edge>> adj = new ArrayList<>();
    for (int i = 0; i < N; i++)
        adj.add(new ArrayList<>());

    for (int i = 0; i < M; i++) {
        int r1 = sc.nextInt();
        int r2 = sc.nextInt();
        int w = sc.nextInt();
        adj.get(r1).add(new Edge(r2, w));
        adj.get(r2).add(new Edge(r1, w)); // If the network is undirected
    }
    int s = sc.nextInt();

    int[] dist = new int[N];
    Arrays.fill(dist, Integer.MAX_VALUE);
    dist[s] = 0;

    PriorityQueue<int[]> pq = new PriorityQueue<>(Comparator.comparingInt(a -> a[1]));
    pq.offer(new int[]{s, 0});
```

```
        while (!pq.isEmpty()) {
            int[] curr = pq.poll();
            int u = curr[0], d = curr[1];
            if(d > dist[u]) continue;
            for (Edge edge : adj.get(u)) {
                int v = edge.target;
                int w = edge.weight;
                if (dist[u] + w < dist[v]) {
                    dist[v] = dist[u] + w;
                    pq.offer(new int[]{v, dist[v]});
                }
            }
        }

        for (int i = 0; i < N; i++) {
            System.out.println(i + " " + dist[i]);
        }
    }
}
```



```java
import java.util.*;

public class Main {
    static class Edge {
        int target, weight;
        Edge(int target, int weight) {
            this.target = target;
            this.weight = weight;
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int M = sc.nextInt();

        List<List<Edge>> adj = new ArrayList<>();
        for (int i = 0; i < N; i++)
            adj.add(new ArrayList<>());

        for (int i = 0; i < M; i++) {
            int r1 = sc.nextInt();
            int r2 = sc.nextInt();
            int w = sc.nextInt();
            adj.get(r1).add(new Edge(r2, w));
            adj.get(r2).add(new Edge(r1, w)); // If the network is undirected
        }
        int s = sc.nextInt();

        int[] dist = new int[N];
        Arrays.fill(dist, Integer.MAX_VALUE);
        dist[s] = 0;

        PriorityQueue<int[]> pq = new PriorityQueue<>(Comparator.comparingInt(a -> a[1]));
```



```java
        PriorityQueue<int[]> pq = new PriorityQueue<>(Comparator.comparingInt(a -> a[1]));
        pq.offer(new int[]{s, 0});

        while (!pq.isEmpty()) {
            int[] curr = pq.poll();
            int u = curr[0], d = curr[1];
            if(d > dist[u]) continue;
            for (Edge edge : adj.get(u)) {
                int v = edge.target;
                int w = edge.weight;
                if (dist[u] + w < dist[v]) {
                    dist[v] = dist[u] + w;
                    pq.offer(new int[]{v, dist[v]});
                }
            }
        }

        for (int i = 0; i < N; i++) {
            System.out.println(i + " " + dist[i]);
        }
    }
}
```

Problem Statement

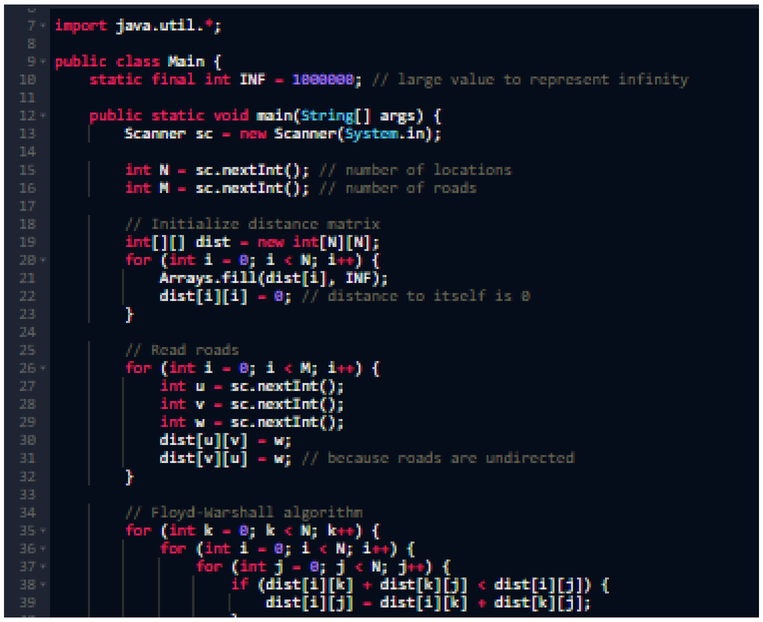
In a city, there are several locations connected by roads. Each road has a certain distance. You are tasked with finding the shortest distances between all pairs of locations in the city using the Floyd-Warshall algorithm.

Write a program that calculates the shortest path between all pairs of locations in the city.

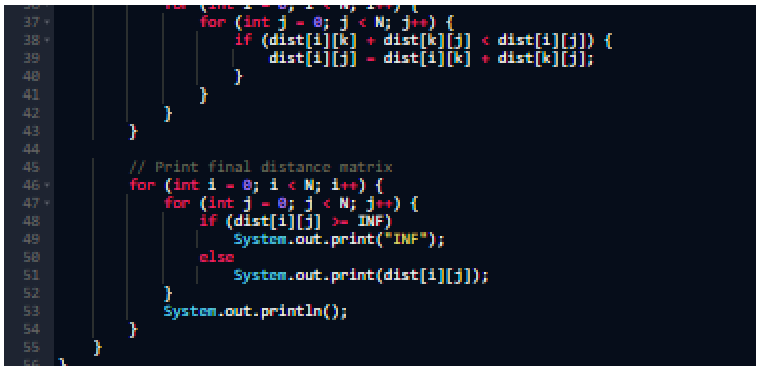
Example
Input:
4
4
0 1 3
1 2 1
2 3 1
0 2 5

Output:
0 3 4 5
3 0 1 2
4 1 0 1
5 2 1 0

Explanation:
The city has 4 locations (0 to 3).
There are 4 roads connecting the locations.
The roads and their distances are:
0 → 1 with distance 3
1 → 2 with distance 1
2 → 3 with distance 1
0 → 2 with distance 5
Initially, the distance matrix is set to infinity (INF) except for direct roads and diagonal values (0).
The Floyd-Warshall algorithm updates the matrix by checking intermediate paths.
The shortest path from 0 to 2 (via 1) is updated to 4 (0 → 1 → 2) instead of 5.
The shortest path from 0 to 3 is 5 (0 → 1 → 2 → 3).
The final distance matrix is printed, showing the shortest distances.
0 3 4 5
3 0 1 2
4 1 0 1
5 2 1 0



```java
import java.util.*;

public class Main {
    static final int INF = 1000000; // large value to represent infinity

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int N = sc.nextInt(); // number of locations
        int M = sc.nextInt(); // number of roads

        // Initialize distance matrix
        int[][] dist = new int[N][N];
        for (int i = 0; i < N; i++) {
            Arrays.fill(dist[i], INF);
            dist[i][i] = 0; // distance to itself is 0
        }

        // Read roads
        for (int i = 0; i < M; i++) {
            int u = sc.nextInt();
            int v = sc.nextInt();
            int w = sc.nextInt();
            dist[u][v] = w;
            dist[v][u] = w; // because roads are undirected
        }

        // Floyd-Warshall algorithm
        for (int k = 0; k < N; k++) {
            for (int i = 0; i < N; i++) {
                for (int j = 0; j < N; j++) {
                    if (dist[i][k] + dist[k][j] < dist[i][j]) {
                        dist[i][j] = dist[i][k] + dist[k][j];
```



```java
                for (int j = 0; j < N; j++) {
                    if (dist[i][k] + dist[k][j] < dist[i][j]) {
                        dist[i][j] = dist[i][k] + dist[k][j];
                    }
                }
            }
        }

        // Print final distance matrix
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                if (dist[i][j] >= INF)
                    System.out.print("INF");
                else
                    System.out.print(dist[i][j]);
            }
            System.out.println();
        }
    }
```