Dijkstra's Algorithm 4

From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm. Write the program in Java.

Input Format

4 0 15 10 9999 9999 0 15 9999 20 9999 0 20 9999 10 9999 0 3

Constraints

--

Output Format

Shortest path from 3 to all other vertices To 0 is 45 To 1 is 10 To 2 is 25 To 3 is 0

Sample Input 0

```
4
0 15 10 9999
9999 0 15 9999
20 9999 0 20
9999 10 9999 0
3
```

Sample Output 0

```
Shortest path from 3 to all other vertices
To 0 is 45
To 1 is 10
To 2 is 25
To 3 is 0
```

Contest ends in 9 days

Submissions: 98
Max Score: 10
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

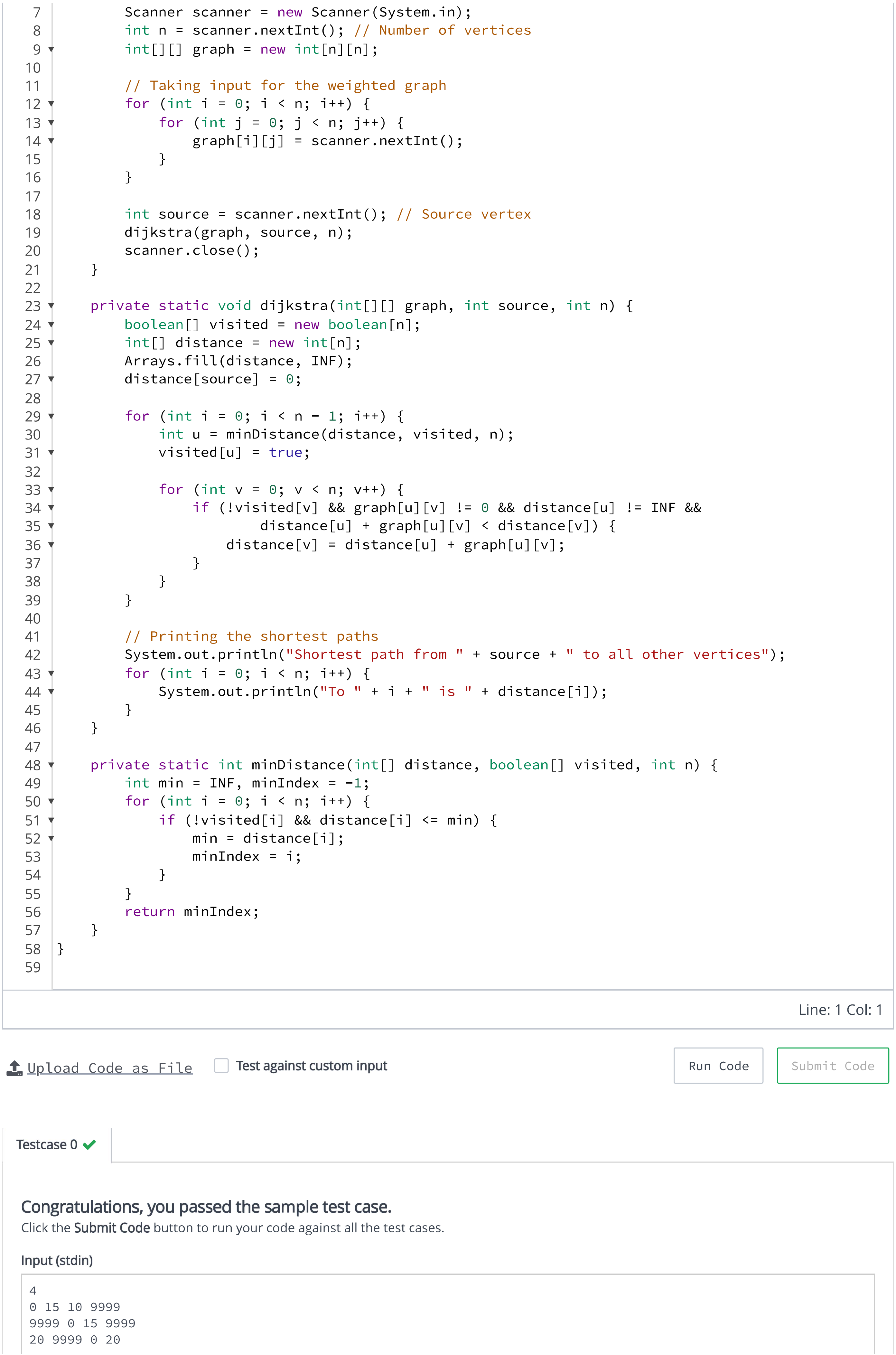
More

Java 7

```java
import java.util.*;

public class DijkstraAlgorithm {
    private static final int INF = 9999;

    public static void main(String[] args) {
```



```java
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt(); // Number of vertices
        int[][] graph = new int[n][n];

        // Taking input for the weighted graph
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                graph[i][j] = scanner.nextInt();
            }
        }

        int source = scanner.nextInt(); // Source vertex
        dijkstra(graph, source, n);
        scanner.close();
    }

    private static void dijkstra(int[][] graph, int source, int n) {
        boolean[] visited = new boolean[n];
        int[] distance = new int[n];
        Arrays.fill(distance, INF);
        distance[source] = 0;

        for (int i = 0; i < n - 1; i++) {
            int u = minDistance(distance, visited, n);
            visited[u] = true;

            for (int v = 0; v < n; v++) {
                if (!visited[v] && graph[u][v] != 0 && distance[u] != INF &&
                        distance[u] + graph[u][v] < distance[v]) {
                    distance[v] = distance[u] + graph[u][v];
                }
            }
        }

        // Printing the shortest paths
        System.out.println("Shortest path from " + source + " to all other vertices");
        for (int i = 0; i < n; i++) {
            System.out.println("To " + i + " is " + distance[i]);
        }
    }

    private static int minDistance(int[] distance, boolean[] visited, int n) {
        int min = INF, minIndex = -1;
        for (int i = 0; i < n; i++) {
            if (!visited[i] && distance[i] <= min) {
                min = distance[i];
                minIndex = i;
            }
        }
        return minIndex;
    }
}

```

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
4
0 15 10 9999
9999 0 15 9999
20 9999 0 20
```

```
9999 10 9999 0
3
```

Your Output (stdout)

```
Shortest path from 3 to all other vertices
To 0 is 45
To 1 is 10
To 2 is 25
To 3 is 0
```

Expected Output

```
Shortest path from 3 to all other vertices
To 0 is 45
To 1 is 10
To 2 is 25
To 3 is 0
```