https://course.acciojob.com/idle?question=92610f2e-18e2-4b4b-838c-bdebed8f07de

- MEDIUM
- Max Score: 40 Points
-
-

Shortest Path

Given a weighted, undirected graph of `V` vertices and an adjacency list `adj` where adj[i] is a list of lists containing two integers where the first integer of each list `j` denotes there is edge between `i` and `j` , second integers corresponds to the weight of that edge . You are given the source vertex `S` and You have to Find the shortest distance of all the vertex's from the source vertex `S`. Return a list of integers denoting shortest distance between each node and Source vertex `S`.

Note 1: The Graph doesn't contain any negative weight cycle.

Note 2: If the node is not reachable from `S`, return distance as `-1`.

Input Format

The first line of input contains two integers, `V` and `E` respectively.

The next `E` lines of input contains three integers `u`, `v`, and `w`, representing there's a edge between vertex `v` and `u` with weight `w`.

The last line of input contains an integer representing `S`.

Output Format

Return a list of integers denoting shortest distance between each node and Source vertex `S`.

Example 1

Input

```
2 1
1 0 9
0
```

Output

```
0 9
```

Example 2

Input

```
3 3
0 1 1
0 2 6
1 2 3
2
```

Output

```
4 3 0
```

Constraints

1 <= V <= 1000

0 <= adj[i][j] <= 1000

1 <= E <= (V*(V-1))/2

0 <= S < V

Topic Tags

- **Graphs**

My code

```
// n java
import java.io.*;
import java.util.*;
import java.lang.*;

public class Main {
  public static int[] dijkstra(int V,
ArrayList<ArrayList<ArrayList<Integer>>> adj, int S)
  {
     // Write your code here
          PriorityQueue<int[]> pq = new PriorityQueue<>(new
Comparator<int[]>() {
                        public int compare(int[] a, int[] b) {
                                return Integer.compare(a[1], b[1]);
                        }
                });

                int n = V;

                int dist[] = new int[n];
                boolean vis[] = new boolean[n];

                Arrays.fill(dist, Integer.MAX_VALUE);
                Arrays.fill(vis, false);

                pq.add(new int[]{S, 0});
```

```
        while(pq.size() > 0) {
            int[] curr = pq.poll();

            int u = curr[0];
            int dist_u = curr[1];

            if(vis[u] == true) continue;

            vis[u] = true;
            dist[u] = dist_u;

            for(int i = 0; i < adj.get(u).size(); i++) {
                int v = adj.get(u).get(i).get(0);
                int g_uv = adj.get(u).get(i).get(1);

                if(vis[v] == false) {
                    pq.add(new int[]{v, dist_u + g_uv});
                }
            }
        }

        for(int i = 0; i < n; i++) {
            if(dist[i] == Integer.MAX_VALUE) dist[i] = -1;
        }

        return dist;

    }

    public static void main(String args[]) throws IOException {
        BufferedReader read =
            new BufferedReader(new InputStreamReader(System.in));
            String str[] = read.readLine().trim().split(" ");
            int V = Integer.parseInt(str[0]);
            int E = Integer.parseInt(str[1]);

            ArrayList<ArrayList<ArrayList<Integer>>> adj = new
ArrayList<ArrayList<ArrayList<Integer>>>();
            for(int i=0;i<V;i++)
            {
                adj.add(new ArrayList<ArrayList<Integer>>());
            }

            int i=0;
            while (i++<E) {
                String S[] = read.readLine().trim().split(" ");
                int u = Integer.parseInt(S[0]);
                int v = Integer.parseInt(S[1]);
                int w = Integer.parseInt(S[2]);
                ArrayList<Integer> t1 = new ArrayList<Integer>();
                ArrayList<Integer> t2 = new ArrayList<Integer>();
                t1.add(v);
                t1.add(w);
                t2.add(u);
                t2.add(w);
                adj.get(u).add(t1);
                adj.get(v).add(t2);
            }
            int S = Integer.parseInt(read.readLine());

            int[] ptr = dijkstra(V, adj, S);

            for(i=0; i<V; i++)
                System.out.print(ptr[i] + " ");
            System.out.println();
        }


}
```