Practice Compete Jobs Rank Leaderboard gadhiya

Dashboard > Algorithms > Graph Theory > Breadth First Search: Shortest Reach

Badge Progress ★★★★

Points: 4727.88 Rank: 491

Breadth First Search: Shortest Reach

by pranav9413

Problem Submissions Leaderboard Discussions Editorial

Consider an undirected graph consisting of n nodes where each node is labeled from 1 to n and the edge between any two nodes is always of length 6. We define node s to be the starting position for a BFS.

Given q queries in the form of a graph and some starting node, s, perform each query by calculating the shortest distance from starting node s to all the other nodes in the graph. Then print a single line of $n - 1$ space-separated integers listing node s's shortest distance to each of the $n - 1$ other nodes (ordered sequentially by node number); if s is disconnected from a node, print -1 as the distance to that node.

Input Format

The first line contains an integer, q, denoting the number of queries. The subsequent lines describe each query in the following format:

- The first line contains two space-separated integers describing the respective values of n (the number of nodes) and m (the number of edges) in the graph.
- Each line i of the m subsequent lines contains two space-separated integers, u and v, describing an edge connecting node u to node v.
- The last line contains a single integer, s, denoting the index of the starting node.

Constraints

- $1 \leq q \leq 10$
- $2 \leq n \leq 1000$
- $1 \leq m \leq \frac{n \cdot (n-1)}{2}$
- $1 \leq u, v, s \leq n$

Output Format

For each of the q queries, print a single line of $n - 1$ space-separated integers denoting the shortest distances to each of the $n - 1$ other nodes from starting position s. These distances should be listed sequentially by node number (i.e., $1, 2, \ldots, n$), but *should not* include node s. If some node is unreachable from s, print -1 as the distance to that node.

Sample Input

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

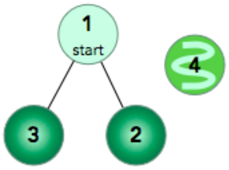
Sample Output

```
6 6 -1
-1 6
```

Explanation

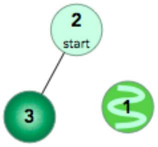
We perform the following two queries:

1. The given graph can be represented as:



where our *start* node, s, is node 1. The shortest distances from s to the other nodes are one edge to node 2, one edge to node 3, and an infinite distance to node 4 (which it's not connected to). We then print node 1's distance to nodes 2, 3, and 4 (respectively) as a single line of space-separated integers: `6, 6, -1`.

2. The given graph can be represented as:



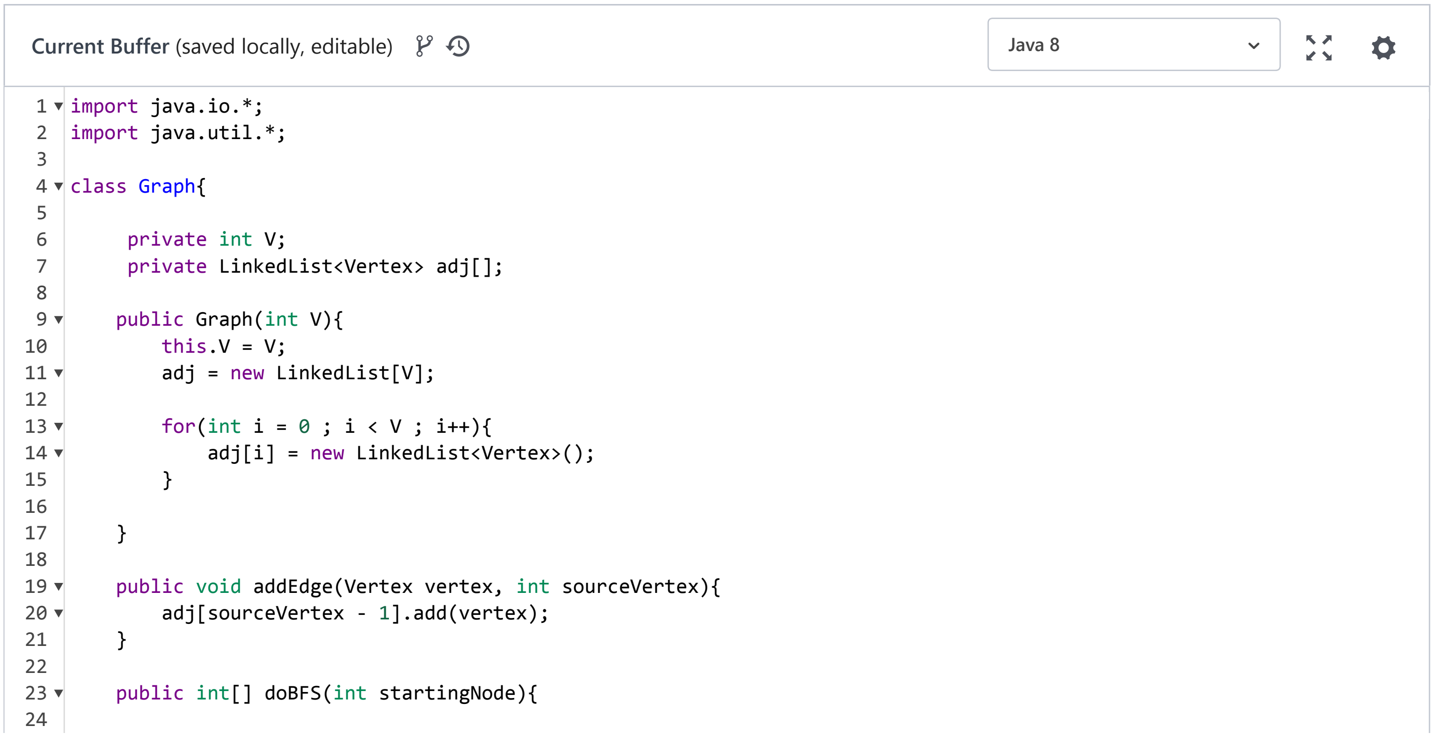
where our *start* node, s, is node 2. There is only one edge here, so node 1 is unreachable from node 2 and node 3 has one edge connecting it to node 2. We then print node 2's distance to nodes 1 and 3 (respectively) as a single line of space-separated integers: `-1 6`.

Note: Recall that the actual length of each edge is 6, and we print -1 as the distance to any node that's unreachable from s.

Submissions: 32435
Max Score: 55
Difficulty: Medium

Rate This Challenge:
☆ ☆ ☆ ☆ ☆

More



```java
import java.io.*;
import java.util.*;

class Graph{

     private int V;
     private LinkedList<Vertex> adj[];

    public Graph(int V){
        this.V = V;
        adj = new LinkedList[V];

        for(int i = 0 ; i < V ; i++){
            adj[i] = new LinkedList<Vertex>();
        }

    }

    public void addEdge(Vertex vertex, int sourceVertex){
        adj[sourceVertex - 1].add(vertex);
    }

    public int[] doBFS(int startingNode){

        LinkedList<Integer> queue = new LinkedList<Integer>();
        queue.add(startingNode - 1);

        int[] out = new int[V];
        boolean[] isVisited = new boolean[V];

        for(int i = 0 ; i < V ; i++){
            out[i] = Integer.MAX_VALUE;
        }

        out[startingNode - 1] = 0;

        while(queue.size() != 0){

            int s = queue.poll();
            isVisited[s] = true;
            Iterator<Vertex> i = adj[s].listIterator();

            while(i.hasNext()){
                Vertex ver = i.next();
                int v = ver.getVertex();

                if(!isVisited[v]){
                    queue.add(v);

                    if(out[v] > out[s] + 6){
                        out[v] = out[s] + 6;
                    }

                }
            }
        }

        return out;
    }
}

class Vertex{

    private int vertex;
    private int edge;

    public Vertex(int vertex){
        this.vertex = vertex - 1;
        edge = 6;
    }

    public int getVertex(){
        return vertex;
    }

    public int getEdge(){
        return edge;
    }
}

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);
        int tst = scan.nextInt();

        for(int i = 0 ; i < tst ; i++){

            int totalV = scan.nextInt();
            int totalE = scan.nextInt();

            Graph graph = new Graph(totalV);

            Set<String> set = new HashSet<String>();

            for(int j = 0 ; j < totalE ; j++){
                int source = scan.nextInt();
                int destination = scan.nextInt();

                Vertex vertex1 = new Vertex(destination);
                Vertex vertex2 = new Vertex(source);

                if(!set.contains(source + "-" + destination)){
                    graph.addEdge(vertex1, source);
                    graph.addEdge(vertex2, destination);
                    set.add(source + "-" + destination);
                    set.add(destination + "-" + source);
                }

            }

            int startingNode = scan.nextInt();

           int[] out = graph.doBFS(startingNode);

            for(int a = 0 ; a < out.length ; a++){

                if(out[a] != 0){
                    if(out[a] == Integer.MAX_VALUE){
                        System.out.print("-1 ");
                    }
                    else{
                        System.out.print(out[a] + " ");
                    }
                }

            }

            System.out.println("");
        }

    }
}
```

Line: 1 Col: 1

Upload Code as File □ Test against custom input Run Code Submit Code

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature