Started on	Wednesday, 3 April 2024, 9:52 PM
State	Finished
Completed on	Wednesday, 3 April 2024, 10:28 PM
Time taken	36 mins 2 secs
Grade	**10.00** out of 10.00 (**100**%)

Question 1

Correct

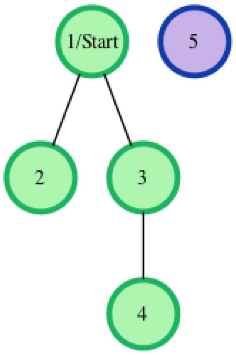
Mark 10.00 out of 10.00

Consider an undirected graph where each edge weighs 6 units. Each of the nodes is labeled consecutively from 1 to n.

You will be given a number of queries. For each query, you will be given a list of edges describing an undirected graph. After you create a representation of the graph, you must determine and report the shortest distance to each of the other nodes from a given starting position using the *breadth-first search* algorithm (BFS). Return an array of distances from the start node in node number order. If a node is unreachable, return -1 for that node.

Example

The following graph is based on the listed inputs:



$n = 5$ // number of nodes
$m = 3$ // number of edges
$edges = [1, 2], [1, 3], [3, 4]$
$s = 1$ // starting node

All distances are from the start node 1. Outputs are calculated for distances to nodes 2 through 5: $[6, 6, 12, -1]$. Each edge is 6 units, and the unreachable node 5 has the required return distance of -1.

Function Description

Complete the *bfs* function in the editor below. If a node is unreachable, its distance is -1.

bfs has the following parameter(s):

- *int n*: the number of nodes
- *int m*: the number of edges
- *int edges[m][2]*: start and end nodes for edges
- *int s*: the node to start traversals from

Returns
int[n-1]: the distances to nodes in increasing node number order, not including the start node (-1 if a node is not reachable)

Input Format

The first line contains an integer q, the number of queries. Each of the following q sets of lines has the following format:

- The first line contains two space-separated integers n and m, the number of nodes and edges in the graph.
- Each line i of the m subsequent lines contains two space-separated integers, u and v, that describe an edge between nodes u and v.
- The last line contains a single integer, s, the node number to start from.

Constraints

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

For example:

Input	Result
2 4 2 1 2 1 3 1 3 1 2 3 2	6 6 -1 -1 6
1 5 3 1 2 1 3 3 4 1	6 6 12 -1

Answer: (penalty regime: 0 %)

Reset answer

```
#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &);
string rtrim(const string &);
vector<string> split(const string &);

/*
 * Complete the 'bfs' function below.
 *
 * The function is expected to return an INTEGER_ARRAY.
 * The function accepts following parameters:
 *  1. INTEGER n
 *  2. INTEGER m
 *  3. 2D_INTEGER_ARRAY edges
 *  4. INTEGER s
 */

vector<int> bfs(int n, int m, vector<vector<int>> edges, int s) {
    // Creating adjacency list representation of the graph which stores the vertices connected to each verte
    vector<vector<int>> adjList(n + 1);
    for (int i = 0; i < m; i++) {//going through all eddges
        int u = edges[i][0]; // getting the starting node of the edge
        int v = edges[i][1]; // getting the ending node of the edge
        adjList[u].push_back(v); // add v to the adjacency list of u
        adjList[v].push_back(u); // add u to the adjacency list of v (undirected graph)
    }

    // Array to store distances from the start node
    vector<int> distances(n + 1, -1);

    // BFS traversal
    queue<int> q;
    q.push(s); // Enqueue the start node
    distances[s] = 0; // Distance to itself is 0

    while (!q.empty()) {
        int node = q.front(); // Dequeue a node
        q.pop();

        // Traverse all neighbors of the current node
        for (int neighbor : adjList[node]) {
            // If the neighbor is not visited yet
            if (distances[neighbor] == -1) {
                distances[neighbor] = distances[node] + 6; // Update distance
                q.push(neighbor); // Enqueue the neighbor for further traversal
            }
        }
    }

    // Converting distances to the required format
```

	Input	Expected	Got	
✔	2 4 2 1 2 1 3 1 3 1 2 3 2	6 6 -1 -1 6	6 6 -1 -1 6	✔
✔	1 5 3 1 2 1 3 3 4 1	6 6 12 -1	6 6 12 -1	✔

Passed all tests! ✔

▸ **Show/hide question author's solution (Cpp)**

Correct

Marks for this submission: 10.00/10.00.