Started on	Wednesday, 3 April 2024, 9:49 PM
State	Finished
Completed on	Wednesday, 3 April 2024, 10:28 PM
Time taken	39 mins 17 secs
Grade	**10.00** out of 10.00 (**100**%)

Question 1

Correct

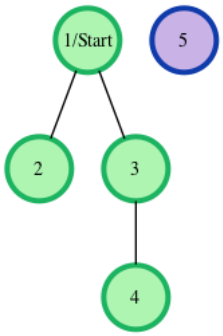
Mark 10.00 out of 10.00

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

```
        adjList[u].push_back(v);
        adjList[v].push_back(u); // Since the graph is undirected
    }

    // Breadth-first search
    queue<int> q;
    q.push(s);
    distances[s] = 0; // Distance to the starting node is 0

    while (!q.empty()) {
        int current = q.front();
        q.pop();

        for (int neighbor : adjList[current]) {
            if (distances[neighbor] == -1) { // If the neighbor is not visited
                distances[neighbor] = distances[current] + 6; // Update the distance
                q.push(neighbor);
            }
        }
    }

    // Prepare the result excluding the starting node
    vector<int> result;
    for (int i = 1; i <= n; ++i) {
        if (i != s) {
            result.push_back(distances[i]);
        }
    }

    return result;
}


int main()
{
    string q_temp;
    getline(cin, q_temp);

    int q = stoi(ltrim(rtrim(q_temp)));

    for (int q_itr = 0; q_itr < q; q_itr++) {
        string first_multiple_input_temp;
        getline(cin, first_multiple_input_temp);

        vector<string> first_multiple_input = split(rtrim(first_multiple_input_temp));

        int n = stoi(first_multiple_input[0]);

        int m = stoi(first_multiple_input[1]);

        vector<vector<int>> edges(m);

```

	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.