BFS: Shortest Reach

HackerRank

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

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

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
```

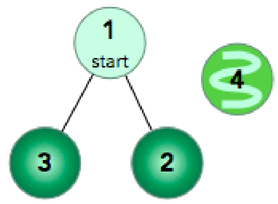
Sample Output

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

Explanation

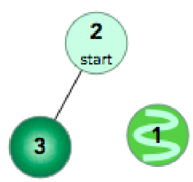
We perform the following two queries:

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.