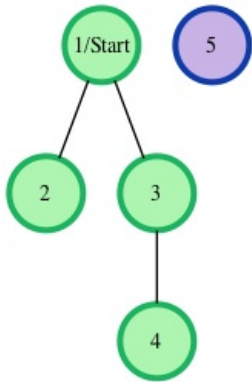


Consider an undirected graph where each edge is the same weight. Each of the nodes is labeled consecutively.

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). Distances are to be reported in node number order, ascending. If a node is unreachable, print -1 for that node. Each of the edges weighs 6 units of distance.

For example, given a graph with 5 nodes and 3 edges, [1, 2], [1, 3], [3, 4], a visual representation is:



The start node for the example is 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. It must return an array of integers representing distances from the start node to each other node in node ascending order. If a node is unreachable, its distance is -1 .

`bfs` has the following parameter(s):

- `n`: the integer number of nodes
- `m`: the integer number of edges
- `edges`: a 2D array of start and end nodes for edges
- `s`: the node to start traversals from

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`, 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, ..., `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
```

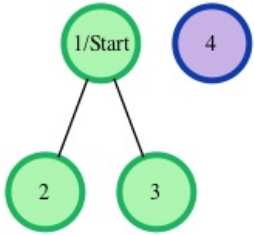
Sample Output

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

Explanation

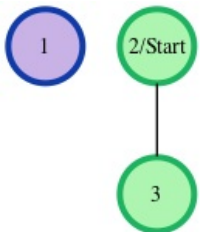
We perform the following two queries:

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*.