Multi Source

Mr. Panda has been told that in lecture, you as a CS2040 student has learnt how to solve the Single-Source Shortest Path problem on an unweighted graph using Breadth First Search (BFS). Now, Mr. Panda wants you to solve *multiple instances* of the Single Source Shortest Path problem on the same directed, unweighted graph. However, the **destination** vertex in all these instances is **always fixed at vertex T**.

In essence, given a directed, unweighted graph with **V** vertices labelled 0 to (**V – 1**) and **E** edges, you are to compute the shortest distance from each of the **Q** source vertices $S_1, S_2, \ldots, S_Q$ to a destination vertex **T**.

Hint: It is **too slow** to run a **Breadth First Search** for every source vertex. Instead, consider what happens if you flip the direction of each edge in the provided graph. What does the problem become?

Input

The first line of input contains 2 integers **V** and **E** where **V** is the number of vertices and **E** is the number of edges in the graph.

The next **E** lines each contains 2 integers **u** and **v**, representing a directed edge from **u** to **v**. It is guaranteed that **u** and **v** are different and that no 2 edges are the same.

The next line of input contains 2 integers **Q** and **T** respectively. **Q** is the number of shortest path instances you are asked to solve, and **T** is the destination vertex for all the shortest path instances.

The next **Q** lines of input each contain an integer S_i representing a source vertex for the **i-th** instance of the Single Source Shortest Path problem.

Output

For every instance with source vertex S_i, output the shortest distance from S_i to **T**.

Limits

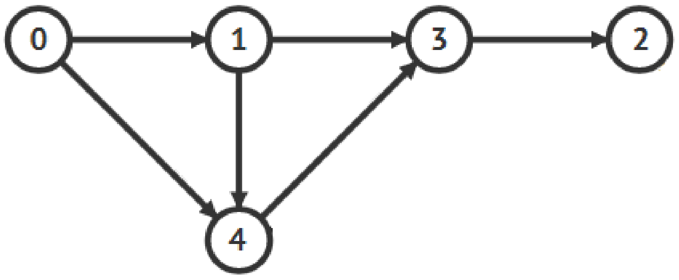
- $1 \le V, E, Q \le 200{,}000$
- $0 \le u, v, S_i, T \le (V - 1)$
- It is guaranteed all the vertices in the graph can reach vertex **T**

Sample Input (**multisource1.in**)	Sample Output (**multisource1.out**)

5 6
0 1
1 3
3 2
1 4
4 3
0 4
8 2
3
4
2
0
2
4
1
4
	1
	2
	0
	3
	0
	2
	2
	2

Explanation

In the input, **V** = 5 and **E** = 6. The graph provided is shown below.



There are **Q** = 8 instances of the single source shortest path problem to be solved, all of them ending at destination vertex **T** = 2.

Shortest path from **0** to 2:	**0 -> 1 -> 3 -> 2**	(Distance 3)
Shortest path from **1** to 2:	**1 -> 3 -> 2**	(Distance 2)
Shortest path from **2** to 2:	**2**	(Distance 0)
Shortest path from **3** to 2:	**3 -> 2**	(Distance 1)
Shortest path from **4** to 2:	**4 -> 3 -> 2**	(Distance 2)

The source vertices are 3, 4, 2, 0, 2, 4, 1, 4 respectively.
As such, the output should be 1, 2, 0, 3, 0, 2, 2, 2.

Notes:

1. You should develop your program in the subdirectory **ex3** and use the skeleton java file provided. You should not create a new file or rename the file provided.
2. You are free to define your own helper methods and classes (or remove existing ones).
3. Please be reminded that the marking scheme is:
   a. Public Test Cases (1%) - 1% for passing **all** test cases, 0% otherwise
   b. Hidden Test Cases (1%) - Partial scoring depending on test cases passed
   c. Manual Grading (1%)
      i. Overall Correctness (correctness of algorithm, severity of bugs)
      ii. Coding Style (meaningful comments, modularity, proper indentation, meaningful method and variable names)
4. Your program will be tested with a time limit of not less than **2 sec** on Codecrunch.

Skeleton File – Multisource.java

You are given the below skeleton file `Multisource.java`. You should see a non-empty file when you open the skeleton file. Otherwise, you might be in the wrong working directory.

```
/**
 * Name          :
 * Matric. No    :
 * PLab Acct.    :
 */
import java.util.*;
public class Multisource {
    private void run() {
        //implement your "main" method here
    }
    public static void main(String[] args) {
        Multisource newMultisource = new Multisource();
        newMultisource.run();
    }
}
```