https://course.acciojob.com/idle?question=f63b28d9-8782-4d16-8e73-52d370a3f1f5

MEDIUM

Max Score: 40 Points

Lexicographically Smallest Topological Ordering

Given a Directed Graph with `V` vertices (Numbered from `1` to `V`) and `E` edges. You need to find the lexicographically smallest topological ordering of the graph.

Note:- Print -1 if the topological sort does not exists for the graph.

Input Format

Input is managed for you. (You are given the graph in the form of adjacency list).

Output Format

Output is managed for you. (You need to complete topologicalSort() function).

Example 1

Input

```
4 3
2 1
3 4
2 4
```

Output

```
2 1 3 4
```

Explanation

```
The following five topological sort exist for the given graph:
(2,1,3,4),(2,3,1,4),(2,3,4,1),(3,2,1,4),(3,2,4,1). The lexicographically smallest among
them is (2,1,3,4)
```

Example 2

Input

```
2 3
1 2
1 2
2 1
```

Output

```
-1
```

Explanation

```
There does not exist any topological ordering of the graph since the graph is cyclic in
nature.
```

Constraints

2 ≤ V ≤ 10000

1 ≤ E ≤ (V*(V-1))/2

1 ≤ u, v ≤ V

The given graph is a directed graph.

Topic Tags

- **Graphs**
- **Heaps**
- **BFS**

My code

```
// in java
import java.util.*;
import java.io.*;
import java.util.*;
class Solution{
  public static ArrayList<Integer> topologicalSort(int V,
ArrayList<ArrayList<Integer>> graph){
    //write your code here
        PriorityQueue<Integer> pq = new PriorityQueue<>();

        int n = V + 1;
        // indegree
        int[] in = new int[n];
        Arrays.fill(in, 0);

        for(int i = 1; i < n; i++) {
            for(int j = 0; j < graph.get(i).size(); j++) {
                int v = graph.get(i).get(j);
		in[v]++;
	}
}

for(int i = 1; i < n; i++) {
	if(in[i] == 0) pq.add(i);
}

ArrayList<Integer> ans = new ArrayList<>();

while(pq.size() > 0) {
	int f = pq.poll();

	ans.add(f);

	// update nbrs indegrees
	for(int i = 0; i < graph.get(f).size(); i++) {
		int v = graph.get(f).get(i);

		in[v]--;

		if(in[v] == 0) {
			pq.add(v);
		}
	}
}

if(ans.size() != V) {
	ArrayList<Integer> t = new ArrayList<>();
	t.add(-1);
```

```
                        return t;
                }
                return ans;
    }
}


public class Main {
    public static void main (String[] args){
        Scanner sc = new Scanner(System.in);
        ArrayList<ArrayList<Integer>> adj = new ArrayList<>();
        int V = Integer.parseInt(sc.next());
        int E = Integer.parseInt(sc.next());
        for(int i = 0 ; i <= V ; ++i){
            adj.add(i , new ArrayList<Integer>());
        }
        for(int i = 0 ; i < E ; ++i){
            int u = Integer.parseInt(sc.next());
            int v = Integer.parseInt(sc.next());
            adj.get(u).add(v);
        }
        Solution ob = new Solution();
        ArrayList<Integer> ans = ob.topologicalSort(V,adj);
        for(int i = 0 ; i < ans.size() ; ++i){
            System.out.print(ans.get(i) + " ");
        }
    }
}
```