18352 - 특정 거리의 도시 찾기

난이도	실버 2

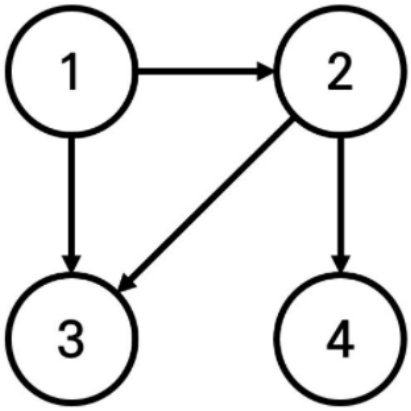
문제

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	256 MB	15440	4515	2861	28.246%

어떤 나라에는 1번부터 N번까지의 도시와 M개의 단방향 도로가 존재한다. 모든 도로의 거리는 1이다.

이 때 특정한 도시 X로부터 출발하여 도달할 수 있는 모든 도시 중에서, 최단 거리가 정확히 K인 모든 도시들의 번호를 출력하는 프로그램을 작성하시오. 또한 출발 도시 X에서 출발 도시 X로 가는 최단 거리는 항상 0이라고 가정한다.

예를 들어 N=4, K=2, X=1일 때 다음과 같이 그래프가 구성되어 있다고 가정하자.



이 때 1번 도시에서 출발하여 도달할 수 있는 도시 중에서, 최단 거리가 2인 도시는 4번 도시 뿐이다. 2번과 3번 도시의 경우, 최단 거리가 1이기 때문에 출력하지 않는다.

입력

첫째 줄에 도시의 개수 N, 도로의 개수 M, 거리 정보 K, 출발 도시의 번호 X가 주어진다. ($2 \le N \le 300,000$, $1 \le M \le 1,000,000$, $1 \le K \le 300,000$, $1 \le X \le N$) 둘째 줄부터 M개의 줄에 걸쳐서 두 개의 자연수 A, B가 공백을 기준으로 구분되어 주어진다. 이는 A번 도시에서 B번 도시로 이동하는 단방향 도로가 존재한다는 의미다. ($1 \le A, B \le N$) 단, A와 B는 서로 다른 자연수이다.

출력

X로부터 출발하여 도달할 수 있는 도시 중에서, 최단 거리가 K인 모든 도시의 번호를 한 줄에 하나씩 오름차순으로 출력한다.

이 때 도달할 수 있는 도시 중에서, 최단 거리가 K인 도시가 하나도 존재하지 않으면 -1을 출력한다.

풀이과정

이 문제는 그렇게 어려운 편은 아니었는데, 숫자 잘못 보고 왜 안 되지하고 고민을 많이 했던 문제다... 다시 한번 느끼지만 **문제 제발 잘 보자!!**

이 문제 역시 최단 거리라는 키워드를 보고 BFS로 풀었다.

소스코드

```
package BOJ18352;

import java.io.*;
import java.util.*;

public class Main {

    static boolean[] visited = new boolean[300001];
    static Vector<Integer>[] graph;
    static int[] route;
    static int N, M, K, X;
    public static void main(String[] args) throws IOException {

        Vector<Integer> answer = new Vector<>();

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String str = br.readLine();
        StringTokenizer st = new StringTokenizer(str);
        N = Integer.parseInt(st.nextToken());
        M = Integer.parseInt(st.nextToken());
        K = Integer.parseInt(st.nextToken());
        X = Integer.parseInt(st.nextToken());

        // Graph
        int A, B;
        graph = new Vector[N+1];
        for(int i=0 ; i<=N ; i++)
            graph[i] = new Vector<>();
        for(int i=0 ; i<M ; i++){
            String cityNum = br.readLine();
            st = new StringTokenizer(cityNum);
            A = Integer.parseInt(st.nextToken());
            B = Integer.parseInt(st.nextToken());
            graph[A].add(B);
        }

        route = new int[300001];
        bfs(X);

        for(int i=1 ; i<=N ; i++) {
            if(route[i] == K) answer.add(i);
        }

        if(answer.isEmpty()) System.out.println(-1);
        else{
            answer.sort((o1, o2) -> o1-o2);
            for(int i=0 ; i<answer.size() ; i++){
                System.out.println(answer.get(i));
            }
        }

    }

    public static void bfs(int num) {
```

```
        Queue<Integer> city = new LinkedList<>();
        visited[num] = true;
        city.add(num);

        while(!city.isEmpty()){
            int node = city.poll();
            for(int i=0 ; i<graph[node].size() ; i++){
                int tmp = graph[node].get(i);
                if(!visited[tmp]){
                    visited[tmp] = true;
                    city.add(tmp);
                    route[tmp] = route[node] + 1;
                }
            }
        }
    }
}
```

💡 주요 포인트

1. route[300001] 배열

일단 출발 도시에서 방문할 수 있는 다른 도시들까지 다 방문하면서 그 최단 거리를 route 배열에 기록해주었다. 300001로 크기를 설정해준 이유는 도시의 개수 N의 범위가 2 이상 300000 이하였기 때문에, 해당 도시 번호에 해당하는 위치에 거리를 적어주기 위함이었다.

그렇게 거리를 다 기록해두고, route 배열에서 최단 거리가 K로 기록된 도시 번호를 찾아 vector에 넣어준 뒤 오름차순으로 정렬하여 출력했다. 만약 해당 vector의 size가 0이라면 -1을 출력하도록 하였다.

위 배열 이외에는 기본적인 BFS 구현 문제였다.

> 사담

저 route 배열의 크기를 30만1로 해줬어야했는데, 30001로 해줘서 에러가 났었다...ㅎ 이제 보니 문제는 읽었지만 내 코드에서 0을 하나 빼먹은 거였다. 실수 좀 줄이자...