2기 / 4주차_유니온 파인드

🌟 유니온 파인드

연결되어 있는 노드를 같은 집합으로 묶어주고, 각 노드들이 연결되어 있는지 (같은 집합에 있는지) 판별하는 알고리즘이다.
두 노드를 같은 집합으로 묶는 Union과 어느 집합에 포함되어 있는지 찾는 Find 연산으로 이루어져 있다.

- Union

```
if (parent[x] <= parent[y]) parent[y] = parent[x];
else parent[x] = parent[y];
```

- Find

```
if (parent[x] == x) return x;
return findParent(parent[x]);
```

1. 백준 17352_여러분의 다리가 되어 드리겠습니다!

💡 접근법

섬의 부모를 결정하고 부모가 다른 섬을 출력하면 되겠다고 생각했다.

🔑 풀이

유니온-파인드 알고리즘을 사용했다.

연결되어야 할 두 개의 섬의 부모 노드를 확인하고, 부모 노드가 다르다면 둘 중 더 작은 값으로 부모 노드를 바꾼다.

이 때, 부모 노드가 같다는 것은 연결되어 있다는 것을 의미한다.

정답은 무조건 1이 들어갈 수 있다는 것을 이용해서 1과 연결되어 있지 않은 섬을 찾아서 출력했다.

출력 과정에서 부모 노드를 찾는 것을 한 번 더 진행하는 이유는 아래와 같은 반례 때문이다.

입력 순서가 이 반례처럼 섞여 있을 경우 부모 노드가 제대로 들어가지 않는 경우가 생긴다.

💻 코드

648 ms, 81048 KB

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

public class BOJ_17352 {
    private static int countIsland;
    private static int[] parent;
    private static int answer1, answer2;

    public static void main(String[] args) throws IOException {
        input();
        findNotConnected();
        output();
    }

    private static void output() {
        System.out.println(answer1 + " " + answer2);
    }

    private static void findNotConnected() {
        answer1 = 1;
        for (int i = 2; i < parent.length; i++) {
            if (findParent(i) != answer1) {
                answer2 = i;
                break;
```

```
            }
        }
    }

    private static void input() throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        countIsland = Integer.parseInt(br.readLine());

        parent = new int[countIsland + 1];
        for (int i = 1; i <= countIsland; i++) {
            parent[i] = i;
        }
        for (int i = 0; i < countIsland - 2; i++) {
            StringTokenizer st = new StringTokenizer(br.readLine());
            int firstIsland = Integer.parseInt(st.nextToken());
            int secondIsland = Integer.parseInt(st.nextToken());

            union(firstIsland, secondIsland);
        }
    }

    private static void union(int firstIsland, int secondIsland) {
        firstIsland = findParent(firstIsland);
        secondIsland = findParent(secondIsland);

        if (firstIsland == secondIsland) return;

        if (firstIsland <= secondIsland) parent[secondIsland] = firstIsland;
        else parent[firstIsland] = secondIsland;
    }

    private static int findParent(int island) {
        if (parent[island] == island) return island;
        return findParent(parent[island]);
    }
}
```

2. 백준 24391_귀찮은 해강이

💡 접근법

🔑 풀이

17352번과 동일한 방법으로 풀었다.

건물을 모두 연결한 후 강의 순서대로 연결되지 않은 강의를 들을 때 정답을 1씩 더해주었다.

💻 코드

484 ms, 51252 KB

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

public class BOJ_24391 {
    private static int countLecture;
    private static int[] parent;
    private static int answer = 0;
    private static String schedule;

    public static void main(String[] args) throws IOException {
        input();
        findAnswer();
        output();
    }

    private static void output() {
        System.out.println(answer);
    }

    private static void findAnswer() {
        StringTokenizer st = new StringTokenizer(schedule);
        int firstLecture = Integer.parseInt(st.nextToken());
        int start = findParent(firstLecture);

        while (st.hasMoreTokens()) {
            int lecture = Integer.parseInt(st.nextToken());
            int parent = findParent(lecture);
            if (parent != start) {
                answer++;
                start = parent;
            }
        }
    }

    private static void input() throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st = new StringTokenizer(br.readLine());

        countLecture = Integer.parseInt(st.nextToken());
        int countLink = Integer.parseInt(st.nextToken());
        parent = new int[countLecture + 1];

        for (int i = 1; i < parent.length; i++) {
            parent[i] = i;
        }

        for (int i = 0; i < countLink; i++) {
            st = new StringTokenizer(br.readLine());
            int firstBuilding = Integer.parseInt(st.nextToken());
            int secondBuilding = Integer.parseInt(st.nextToken());
```

```
            union(firstBuilding, secondBuilding);
        }

        schedule = br.readLine();
    }

    private static void union(int firstBuilding, int secondBuilding) {
        firstBuilding = findParent(firstBuilding);
        secondBuilding = findParent(secondBuilding);

        if (firstBuilding == secondBuilding) return;

        if (firstBuilding <= secondBuilding) parent[secondBuilding] = firstBuilding;
        else parent[firstBuilding] = secondBuilding;
    }

    private static int findParent(int firstBuilding) {
        if (parent[firstBuilding] == firstBuilding) return firstBuilding;
        return findParent(parent[firstBuilding]);
    }
}
```

3. 백준 12893_적의 적

💡 접근법

🔑 풀이

💻 코드

1228 ms, 263976 KB

```
```