

1707번: 이분 그래프

1707번 제출 맞은 사람 쏫코딩 풀이 풀이 작성 풀이 요청 재채점/수정 문제 추천
채점 현황 강의▼

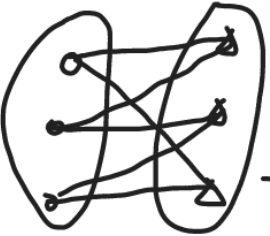
이분 그래프 풀이

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2 초	128 MB	9928	2202	1321	20.625%

문제

그래프의 정점의 집합을 둘로 분할하여, 각 집합에 속한 정점끼리는 서로 인접하지 않도록 분할할 수 있을 때, 그러한 그래프를 특별히 이분 그래프 (Bipartite Graph) 라 부른다.

그래프가 입력으로 주어졌을 때, 이 그래프가 이분 그래프인지 아닌지 판별하는 프로그램을 작성하시오.

이분그래프
인접 리스트 dfs or bfs + flag

how?
dfs? → 노드 방문 마다 flag switch
bfs? flag를 어디에? visit 어떨?
=> next == 0 진행 else 같X
다
↓
queue를 여러개 넣듯

문제 유형 : 그래프 , input 정보 분류하기

접근 방법은 한번 들은 문제라서 쉽게 생각해 냈다. DFS , BFS 둘 다 가능하며, dfs는 방문시마다 flag를 달리 해주며 확인하고, DFS는 인접 노드를 방문할 때 Flag를 달리 하여 방문하면 된다. 처음에 BFS는 flag를 번갈아 가며 방문해야 된다는 착각 때문에 구현이 어려울줄 알았지만, 방문 여부를 확인하는 저장 공간에 방문, flag를 저장하면 DFS와 같은 방법으로 풀리게 된다.

하지만 구현에 애가 먹었는데, 우선 재귀 구조에서 번갈아 들어오는 flag를 check할 때 오류가 났었다. 재귀 함수는 실수하기 좋은 부분이므로 세심하게 어디가 잘못 되었는지 확인해 가며 문제를 풀어야 한다. -> 구현시 생각 없이 코딩

두 번째 연결 그래프와 비연결 그래프 조건을 생각 못했다. 들어오는 모든 그래프가 한덩어리라는 편견을 갖고 코딩을 하였다. -> 문제의 요구사항 파악을 잘 못함

세 번째 그래프를 만들 때 양방향 그래프와 일 방향 그래프의 차이점을 염두해 두지 않았다. -> 인접행렬이나 인접 리스트로 그래프를 표현하는 것과 머리속에서 그려지는 그래프의 차이를 인지 못하고 일방향 그래프만 그리고 코딩을 하였다.

모든 문제는 문제를 읽고 생각, 코딩하면서 생각, feed -back이 부족해서 이다. 문제를 읽고 충분히 생각하고, 코드를 타이핑 해가면서 생각하면 작동하지 않는 프로그램 디버깅하는 시간과 더러운 코드 리팩토링하는 시간을 아낄 수 있다. 덤으로 사고력도 길러진다. 생각하고 한 글자 하나 하나에 피드백을 하자.

코드

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

public class Main {
    public static BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    public static ArrayList<Integer>[] cnj;
    public static int[] visit;
    public static void main(String[] Args)throws IOException {
        int t = Integer.parseInt(in.readLine());
        //in testCase
        while(t!=0) {
            String[] line = in.readLine().split(" ");
            int v = Integer.parseInt(line[0]) , e= Integer.parseInt(line[1]);
            cnj = new ArrayList[v];
            for(int i=0; i<v; i++) {
                visit = new int[v];
                cnj[i] = new ArrayList();
                visit[i]=0;
            }
            for(int i=0; i<e; i++) {
                line = in.readLine().split(" ");
                int cur = Integer.parseInt(line[0])-1 , nex = Integer.parseInt(line[1])-1;
```

```

        cnj[cur].add(nex);
        cnj[nex].add(cur);
    }
    find(v);
    t--;
}
}

public static boolean dfs (int cur, int flag) {
    if(visit[cur]==flag&&visit[cur]!=0)return true;
    if(visit[cur]!=flag&&visit[cur]!=0)return false;
    if(visit[cur]==0)visit[cur] = flag;
    for(int n : cnj[cur]) {
        if (!dfs(n,flag*-1)) return false;
    }
    return true;
}

public static void find(int v) {
    for( int node =0; node < v; node++) {
        if(visit[node] == 0) {
            if(!dfs(node,1)) {
                System.out.println("NO");
                return;
            }
        }
    }
    System.out.println("YES");
}
}
}

```

입력

입력은 여러 개의 테스트 케이스로 구성되어 있는데, 첫째 줄에 테스트 케이스의 개수 $K(2 \leq K \leq 5)$ 가 주어진다. 각 테스트 케이스의 첫째 줄에는 그래프의 정점의 개수 $V(1 \leq V \leq 20,000)$ 와 간선의 개수 $E(1 \leq E \leq 200,000)$ 가 빈 칸을 사이에 두고 순서대로 주어진다. 각 정점에는 1부터 N 까지 차례로 번호가 붙어 있다. 이어서 둘째 줄부터 E 개의 줄에 걸쳐 간선에 대한 정보가 주어지는데, 각 줄에 인접한 두 정점의 번호가 빈 칸을 사이에 두고 주어진다.

출력

K 개의 줄에 걸쳐 입력으로 주어진 그래프가 이분 그래프이면 YES, 아니면 NO를 순서대로 출력한다.

예제 입력

2

3 2

1 3

2 3

4 4

1 2

2 3

3 4

4 2

예제 출력

YES

NO

힌트

출처

- 문제를 번역한 사람: author5