텀 프로젝트(문제 _ 9466)

이번 가을학기에 '문제 해결' 강의를 신청한 학생들은 텀 프로젝트를 수행해야 한다. 프로젝트 팀원 수에는 제한이 없다. 심지어 모든 학생들이 동일한 팀의 팀원인 경우와 같이 한 팀만 있을 수도 있다. 프로젝트 팀을 구성하기 위해, 모든 학생들은 프로젝트를 함께하고 싶은 학생을 선택해야 한다. (단, 단 한 명만 선택할 수 있다.) 혼자 하고 싶어하는 학생은 자기 자신을 선택하는 것도 가능하다.

학생들이(s1, s2, ..., sr)이라 할 때, r=1이고 s1이 s1을 선택하는 경우나, s1이 s2를 선택하고, s2가 s3를 선택하고,..., sr-1이 sr을 선택하고, sr이 s1을 선택하는 경우에만 한 팀이 될 수 있다.

예를 들어, 한 반에 7명의 학생이 있다고 하자. 학생들을 1번부터 7번으로 표현할 때, 선택의 결과는 다음과 같다.

1	2	3	4	5	6	7
3	1	3	7	3	4	6

위의 결과를 통해 (3)과 (4, 7, 6)이 팀을 이룰 수 있다. 1, 2, 5는 어느 팀에도 속하지 않는다.

주어진 선택의 결과를 보고 어느 프로젝트 팀에도 속하지 않는 학생들의 수를 계산하는 프로그램을 작성하라.

입력

첫째 줄에 테스트 케이스의 개수 T가 주어진다. 각 테스트 케이스의 첫 줄에는 학생의 수가 정수 n (2 ≤ n ≤ 100,000)으로 주어진다.

각 테스트 케이스의 둘째 줄에는 선택된 학생들의 번호가 주어진다. (모든 학생들은 1부터 n까지 번호가 부여된다.)

출력

각 테스트 케이스마다 한 줄에 출력하고, 각 줄에는 프로젝트 팀에 속하지 못한 학생들의 수를 나타내면 된다.

예제

```
2
7
3 1 3 7 3 4 6
```

```
8
1 2 3 4 5 6 7 8

answer :
2
0
```

풀이

설명 참고 - https://hibee.tistory.com/152

사이클이 형성 되었다면 그 사이클에 속해있는 학생들은 한 팀이 된다. 즉 사이클이 형성 되었는지 판단하는 게 이번 문제의 풀이법이다.

사이클 판단은 dfs을 사용하면 된다. dfs는 더이상 나아갈 수 없을 때까지 한 정점을 파고들기 때문이다. (깊이우선)

하지만 이번 문제는 테스트 케이스의 크기가 크기 때문에 단순히 dfs를 이용한다면 시간 초과가 나게 된다. (학생 하나마다 dfs를 사용하는 경우 - visited 배열 초기화 과정에서 시간이 많이 든다)

그렇기 때문에 visited 배열 초기화 없이 사이클 판단을 할 필요가 있다.

알아야 할점

모든 학생이 한명을 꼭 지목하기 때문에, 어떤 정점에서 dfs를 시작하던 사이클을 포함하는 경로가 나오게 된다.
그렇기 때문에 다음 노드가 이미 사이클을 형성 했는지, 형성을 실패 한 건지 확인하면 굳이 끝까지 확인할 필요가 없다.

```
학생 1 , 2 가 있는 경우

1->1 / 2->1 -----> [1] 사이클
1->1 / 2->2 -----> [1],[2] 사이클
1->2 / 2->1 -----> [1,2] 사이클
1->2 / 2->2 -----> [2] 사이클
```

배열

1. visited[] == false : dfs 탐색을 진행하지 않은 노드
2. finished[] : 이미 방문했던 노드를 또 방문하니까, 현재 노드를 포함한 사이클이 생겼거나, 현재 노드를 제외한 사이클이 이미 생겨있는 경우

3. finished[] == false : 새로운 사이클이 생기는 경우. cnt 증가 필요

4. finished[] == true : 사이클 판단이 되어 있는 경우. 이미 사이클 판단이 되어있다면 현재 노드가 그 사이클에 포함되어 있었다면 이미 dfs를 진행해 visited가 true였을 것임. 하지만 방문조차 하지 않았기 때문에 해당 사이클에 이번 노드 포함 안되어 있음. 사이클 형성 실패

내가 헷갈려서 하는 정리

❗ for문에서 if(!visited[])하는 이유 ❗

만약 1->2->3->4->5->4 이라면 dfs(1)을 진행할 때 1,2,3,4,5 노드를 모두 방문하고 사이클 판단까지 마친다. 그렇기 때문에 방문한 노드는 사이클 판단이 끝나 있다고 생각해도 됨. 더이상 dfs로 탐색할 이유가 없다.

❗ 마지막에 finished[]= true 해주는 이유 ❗

dfs 인자를 포함한 사이클이 형성됐던 아니던, dfs를 끝까지 실행하며 사이클을 만났을 것이다. 모든 학생이 한명을 꼭 지목하기 때문에 어떤 정점에서 시작하던 사이클을 만나고 끝난다는 점을 생각해보면 함수가 끝나기전에는 무조건 사이클을 형성하거나 이미 생성된 사이클을 만났을 것이고, 현재 dfs 인자를 가리키는 학생이 dfs를 시작할 때 finished 배열이 true 라면 해당 학생을 제외한 사이클이 이미 형성 되었다고 판단할 수 있으므로 finished를 마지막에 꼭 true로 바꿔준다.

```
3->4->5->3 : 사이클 형성 -> finished[true]

2->3->4->5->3 : 3이 이미 방문한 노드라 finished 확인. true라면 2를 제외한 사이클 이미 존

1->2->3->4->5->3 : 2이 이미 방문한 노드라 finished 확인. true라면 1을 제외한 사이클 이미
```

❗ cnt + 1 해주는 이유 ❗

- 만약 `1 → 2 → 3 → 1` 의 사이클이 형성되었다면, 1 부터 반복문을 돌린다 (now: 3, nxt: 1)
- 사이클의 마지막 노드 전까지 반복하며 cnt++ 을 수행한다
- 마지막 노드를 포함하면 곧바로 종료 되기 때문에 마지막에 cnt에 +1 을 수행해준다

❗ 예시 Input을 넣었을 때의 코드 작동 ❗

dfs(num)	Dfs 함수 내용	cur	Next	배열 변화
dfs(1)	! visited[3] -> dfs(3) 종료	1	3	(1) visited[1] = true (5) finished[1] = true
dfs(3)	visited[3] -> else 문 진행 ! finished[3] -> cnt 증가 종료 후 dfs(1)로 돌아감	3	3	(2) visited[3] = true (3) Cnt 0 -> 1 (4) Finished[3] = true
dfs(2)	visited[1] -> else 문 진행 finished[1] -> 이미 사이클 탐색이 끝난 경우 종료	2	1	(6) visited[2] = true (7) finished[2] = true
dfs(3)	이미 방문한 노드임으로 dfs 함수 실행 안함			
dfs(4)	!visited[7] -> dfs(7) 종료	4	7	(8) visited[4] = true (14) finished[4] = true
dfs(7)	!visited[6] -> dfs(6) 종료 후 dfs(4)로 돌아감	7	6	(9) visited[7] = true (13) finished[7] = true
dis(6)	visited[4] -> else 문 진행 ! finished[4] -> cnt 증가 종료 후 dfs(7)로 돌아감	6	4	(10) visited[6] = true (11) cnt 1 -> 4 (12) finished[6] = true
dfs(5)	visited[3] -> else 문 진행 finished[3] -> 이미 사이클 판단이 끝난 경우 종료	5	3	(15) visited[5] = true (16) finished[5] = true

코드

```
int  arr[100001];

bool  visited[100001];

bool  finished[100001];

int cnt;


void  dfs(int  cur){
```

```
    visited[cur]= true;
    int next = arr[cur];

    if(!visited[next]){ // 아직 탐색하지 못한 노드의 경우
        dfs(next); // 더 진행해본다
    }
    else{ // 사이클이거나, dfs 끝까지 가서 사이클 판단이 끝난 경우
        if(!finished[next]){ // 새로운 사이클이 생기는 경우
            for(int i=next;i != cur ; i=arr[i]){
                cnt++;
            }
        cnt ++; // 사이클의 마지막 노드는 포함이 안되므로 +1
        }
    }
    finished[cur]=true; // 사이클 판단이 끝남(이 노드와 연결되는 노드는 사이클 형성 못함)
}
int  main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    int t;
    cin >> t;

    for(int test=0;test<t;test++){
```

```
        int n;
        cin >> n;

        for(int i=1;i<=n;i++){ //1번부터 n번까지
            cin >> arr[i];
        }

        memset(visited, false, sizeof(visited));
        memset(finished, false, sizeof(finished));

        cnt = 0; // 팀을 형성한 학생들

        for(int i=1;i<=n;i++){
            if(!visited[i]) dfs(i);
        }

        cout << n - cnt<< "\n"; // 총 학생 - 팀 구성된 학생 수 = 팀 못 구한 학생 수

    }
}
```