10451 - 순열 사이클

난이도	실버 2

문제

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	256 MB	13916	8752	6284	62.596%

문제

1부터 N까지 정수 N개로 이루어진 순열을 나타내는 방법은 여러 가지가 있다. 예를 들어, 8개의 수로 이루어진 순열 (3, 2, 7, 8, 1, 4, 5, 6)을 배열을 이용해 표현하면 $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 2 & 7 & 8 & 1 & 4 & 5 & 6 \end{pmatrix}$ 와 같다. 또는, Figure 1과 같이 방향 그래프로 나타낼 수도 있다.

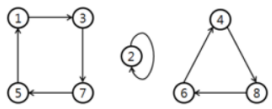

Figure 1.

순열을 배열을 이용해 $\begin{pmatrix} 1 & \dots & i & \dots & n \\ \pi_1 & \dots & \pi_i & \dots & \pi_n \end{pmatrix}$ 로 나타냈다면, i에서 π_i로 간선을 이어 그래프로 만들 수 있다.

Figure 1에 나와있는 것 처럼, 순열 그래프 (3, 2, 7, 8, 1, 4, 5, 6) 에는 총 3개의 사이클이 있다. 이러한 사이클을 "순열 사이클" 이라고 한다.

N개의 정수로 이루어진 순열이 주어졌을 때, 순열 사이클의 개수를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 테스트 케이스의 개수 T가 주어진다. 각 테스트 케이스의 첫째 줄에는 순열의 크기 N ($2 \le N \le 1,000$)이 주어진다. 둘째 줄에는 순열이 주어지며, 각 정수는 공백으로 구분되어 있다.

출력

각 테스트 케이스마다, 입력으로 주어진 순열에 존재하는 순열 사이클의 개수를 출력한다.

예제 입력 1 복사

```
2
8
3 2 7 8 1 4 5 6
10
2 1 3 4 5 6 7 9 10 8
```

예제 출력 1 복사

```
3
7
```

풀이과정

이 문제는 그래프 중 사이클의 개수를 세는 문제였다.

먼저 각 테케당 순열을 입력받은 뒤 따로 인접 리스트를 만들어주어 그래프를 나타냈고, DFS로 돌며 사이클을 하나씩 세주었다. 이 문제도 백준 2468, 2667번처럼영역 세주는 문제와 비슷해서 쉽게 풀 수 있었다.

소스코드

```
package BOJ10451;

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

public class Main {
    static int[] permutation;         // 순열
    static Vector<Integer>[] graph;   // 인접 리스트
    static boolean[] visited;         // 방문 여부
    static int T, N, count;

    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st;
        T = Integer.parseInt(br.readLine());
        for (int i = 0; i < T; i++) {
            N = Integer.parseInt(br.readLine());
            permutation = new int[N + 1];
            visited = new boolean[N + 1];
            graph = new Vector[N + 1];
            for (int j = 0; j <= N; j++)
                graph[j] = new Vector<>();
            permutation[0] = 0;
            st = new StringTokenizer(br.readLine());

            // 순열 입력받기
            for (int j = 1; j <= N; j++)
                permutation[j] = Integer.parseInt(st.nextToken());
```

```
            // 인접 리스트 입력하기
            for (int j = 1; j <= N; j++) {
                graph[j].add(permutation[j]);
            }

            count = 0;
            for (int j = 1; j <= N; j++) {
                if (!visited[permutation[j]]) {
                    dfs(permutation[j]);
                    count++;
                }
            }
            System.out.println(count);
        }
    }
    static void dfs(int num){
        visited[num] = true;

        int idx = 0;
        for (int node : graph[num]) {
            if (!visited[node]) dfs(node);
        }
    }
}
```