프로그래머스 – 폰켓몬

■ for 문을 이용?

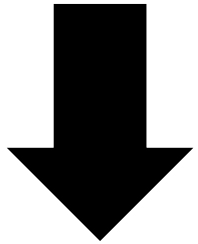
Nums를 1회 반복, 그 중에서 N/2만큼해서 종류를 계산, max와 교체
-> n2 + n/2

■ set 이용

Set에 집어넣어 중복을 제거하고 종류가 몇 개인지 셀 수 있음

프로그래머스 – 폰켓몬

[3,3,3,2,2,4]



Set

0	1	2
3	2	4

프로그래머스 – 폰켓몬

Set

0	1	2
3	2	4

Set size	N/2
3	2

고를 수 있는 종류의 수가 2개가 최대므로
총 3개의 종류를 고를 수 있어 정답은 2

프로그래머스 – 폰켓몬

```java
import java.util.*;

class Solution {
    public int solution(int[] nums) {
        int len = nums.length/2;

        HashSet<Integer> set = new HashSet<>
();
        for(int x: nums){
            set.add(x);
        }

        if(set.size() > len)
            return len;
        else
            return set.size();
    }
}
```

프로그래머스 – 완주하지 못한 선수

1. 참여자들을 HashMap에 넣는다.
    - 이때 중복되는 참여자는 +1

2. 완주자들을 참여자 명단에 있는 지 비교하고, 있으면 -1을 해 0을 만든다.

3. 참여자 HashMap을 조사해 0이 아니라면 0이 아닌 참여자 이름을 return

```java
import java.util.HashMap;


class Solution {
    public String solution(String[] participant, String[] completion)
{        String answer = "";
        HashMap<String, Integer> map = new HashMap<String, Integer>();

        int par_len = participant.length;
        int com_len = completion.length;

        for(String n : participant){

            if(map.containsKey(n)){
                map.put(n, map.get(n) +1);
            }
            else
                map.put(n,  1);
        }


        for (String n : completion){
            if(map.containsKey(n)){
                map.put(n,map.get(n) -1 );
            }
        }

        for(String n : participant){
            if(map.get(n) != 0)
                answer = n;
        }
        return answer;
    }
}
```

Hashmap에 넣기,
중복되면 value+1

완주자 명단을 비교해
value-1,
Value가 0이 아닌 사람을
return하고 종료

백준 – 바이러스

1. ArrayList를 가진 배열을 만들어 그래프 연결

2. DFS로 (1번부터 시작) 감염된 컴퓨터를 시작으로 DFS 시작할 때마다 +1

```java
package B2606바이러스;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.StringTokenizer;

public class B2606바이러스_서은효 {
    static int computer;
    static int pair;
    static ArrayList<Integer> [] graph;
    static boolean [] isVisited;
    static int cnt;

    public static void DFS(int nowNode) {

        if(isVisited[nowNode]) return;

        isVisited[nowNode] = true;
        cnt++;
        for(int nextNode : graph[nowNode]) {
            DFS(nextNode);

        }
    }
    public static void main(String[] args) throws NumberFormatException, IOException
{       BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer token;

        computer = Integer.parseInt(bf.readLine())+1;
        pair = Integer.parseInt(bf.readLine());

        graph = new ArrayList[computer];
        isVisited = new boolean[computer];

        for(int i =0; i<computer; i++) graph[i]= new ArrayList<>();


        for(int i =0; i<pair; i++) {
            token = new StringTokenizer(bf.readLine());

            int u = Integer.parseInt(token.nextToken());
            int v = Integer.parseInt(token.nextToken());

            graph[u].add(v);
            graph[v].add(u);

        }
        System.out.println(Arrays.toString(graph));

        DFS(1);
        System.out.println(cnt-1);



    }
}
```

백준 – 유기농 배추

1. 배열을 돌며 1을 만나면 DFS 시작

2. DFS에서 상,하,좌,우를 검사해 1이 있는 지 검사, DFS 한번 끝나면 +1

1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	1	1	1
0	0	0	0	1	0	0	1	1	1