#1403 거짓말

문제

문제

지민이는 파티에 가서 이야기 하는 것을 좋아한다. 파티에 갈 때마다, 지민이는 지민이가 가장 좋아하는 이야기를 한다. 지민이는 그 이야기를 말할 때, 있는 그대로 진실로 말하거나 엄청나게 과장해서 말한다. 당연히 과장해서 이야기하는 것이 훨씬 더 재미있기 때문에, 되도록이면 과장해서 이야기하려고 한다. 하지만, 지민이는 거짓말쟁이로 알려지기는 싫어한다. 문제는 몇몇 사람들은 그 이야기의 진실을 안다는 것이다. 따라서 이런 사람들이 파티에 왔을 때는, 지민이는 진실을 이야기할 수 밖에 없다. 당연히, 어떤 사람이 어떤 파티에서는 진실을 듣고, 또다른 파티에서는 과장된 이야기를 들었을 때도 지민이는 거짓말쟁이로 알려지게 된다. 지민이는 이런 일을 모두 피해야 한다.

사람의 수 N이 주어진다. 그리고 그 이야기의 진실을 아는 사람이 주어진다. 그리고 각 파티에 오는 사람들의 번호가 주어진다. 지민이는 모든 파티에 참가해야 한다. 이때, 지민이가 거짓말쟁이로 알려지지 않으면서, 과장된 이야기를 할 수 있는 파티 개수의 최댓값을 구하는 프로그램을 작성하시오.

입력

첫째 줄에 사람의 수 N과 파티의 수 M이 주어진다.

둘째 줄에는 이야기의 진실을 아는 사람의 수와 번호가 주어진다. 진실을 아는 사람의 수가 먼저 주어지고 그 개수만큼 사람들의 번호가 주어진다. 사람들의 번호는 1부터 N까지의 수로 주어진다.

셋째 줄부터 M개의 줄에는 각 파티마다 오는 사람의 수와 번호가 같은 방식으로 주어진다.

N, M은 50 이하의 자연수이고, 진실을 아는 사람의 수는 0 이상 50 이하의 정수, 각 파티마다 오는 사람의 수는 1 이상 50 이하의 정수이다.

출력

첫째 줄에 문제의 정답을 출력한다.

M개의 파티가 열리고 지민이는 모든 파티에 참석하여 이야기를 한다.

지민이는 최대한 과장해서 이야기를 하려고 한다. 그러나 그 이야기가 거짓임을 들키면 안된다.

따라서, 파티에 진실을 알고 있는 사람이 있는 경우, 지민이는 진실을 이야기하게 된다.

그리고 어떤 한 사람이 파티에서 진실인 이야기를 듣고, 다른 파티에선 거짓인 이야기를 들으면, 거짓말을 들키게 된다.

지민이가 거짓말을 하고 있다는 것을 들키지 않으면서, 거짓말을 할 수 있는 파티 개수의 최댓값을 구하시오.

▼ 입력 예시

예제 입력 6 복사

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

▼ 입력 데이터

- 파티 개수 : 50 이하의 자연수
- 사람 수 : 50 이하의 자연수
- 진실을 알고 있는 사람 수 : 0 이상 50 이하의 정수
- 각 파티에 오는 사람의 수 : 50 이하의 자연수

분석

파티에서 진실을 말해야 하는 경우 :

1. [진실을 알고 있는 사람]이 있는 경우
2. [[진실을 알고 있는 사람과] 함께 파티를 참여한 사람] 이 있는 경우
3. [[[진실을 알고 있는 사람]과 함께 파티를 참여한 사람]과 함께 파티를 참여한 사람]이 있는 경우
4. ...

⇒ 같은 파티에 참여한 사람끼리의 연결 관계를 알아야 함.

풀이

Union-Find 사용

Union-Find

: 합집합을 찾는 알고리즘. 주로 트리로 구현.

⇒ 같은 집합에 포함된 노드들은 같은 root를 가지고 있음.

find

현재 노드의 root을 반환하는 함수

```
int findParent(int idx){
    if(parent[idx] == idx)
        return idx;

    return parent[idx] = findParent(parent[idx]);
}
```

union

한 노드를 다른 노드를 하나의 집합으로 묶는 과정 (하나의 트리로 묶는 과정)

```
void unionParent(int idx1, int idx2){
    int p1 = findParent(idx1);
    int p2 = findParent(idx2);

    p1 < p2 ? parent[p2] = p1 : parent[p1] = p2;
}
```

코드

1. 진실을 알고 있는 노드의 값을 -1로 설정 (진실 아는 경우 root값이 -1)
2. 전체 파티당 참가인원 저장하며 union-find로 같은 집합 찾기
3. 파티 당 참가인원 조회하며 root 값이 -1인 경우 cnt ++ (진실말해야하는 경우 cnt++)
4. 거짓말할 수 있는 파티 개수 찾는 것이므로 [전체 파티개수 - cnt] 출력

```
#include <iostream>
#include <vector>

using namespace std;

vector<int> parent;
vector<vector<int>> party;

int N; // 사람 수
int M; // 파티 수
int trueN ; // 진실아는 사람 수
int i,j;
```

```
int cnt;

int findParent(int idx);
void unionParent(int idx1, int idx2);


int main()
{
    cin  >> N >>M >> trueN;

    // union-find 배열(parent 배열) 초기화  - 자기자신으로
    for(i=0;i<N+1;i++){
        parent.push_back(i);
    }

    // 진실 알고 있는 member - parent 배열 -1로 초기화
    for(i=0;i<trueN;i++){
        int trueMember;
        cin >> trueMember;
        parent[trueMember]=-1;
    }

    // 파티원 -> party 배열 초기화  + 한 파티에 참여하는 사람들 union 하기
    for(i=0;i<M;i++){
        party.push_back(vector<int>());

        int partyMemN;
        cin >> partyMemN;

        for(j=0;j<partyMemN;j++){
            int partyMem;
            cin >> partyMem;

            party[i].push_back(partyMem); // 파티 이차원 배열에 멤버 추가

            if(j>=1) // 3 5 6 7
                unionParent(party[i][j-1],partyMem); // 파티 멤버끼리 union 하기

        }
    }


    for(i=0;i<M;i++){
        int partyMemN = party[i].size();
        for(j=0;j<partyMemN;j++){
            if(findParent(party[i][j]) == -1){
                cnt++;
                break;
            }
        }
    }

    cout << M-cnt;
    return 0;
}

int findParent(int idx){
    if(idx==-1 || parent[idx] == idx)
```

```
        return idx;

    return parent[idx] = findParent(parent[idx]);
}

void unionParent(int idx1, int idx2){
    int p1 = findParent(idx1);
    int p2 = findParent(idx2);

    if(p1 == -1 && p2 == -1 )
        return;
    else
        p1 < p2 ? parent[p2] = p1 : parent[p1] = p2;
}
```