#1403 거짓말💬

202010542 최정윤

목차

- 문제 분석
- Union-Find 설명
- 코드

문제

- 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.
- 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

예제 입력 6 복사

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

- 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

진실을 아는 사람	1
파티 1	2 5
파티 2	3 4 6
파티 3	1 3
파티 4	4 7

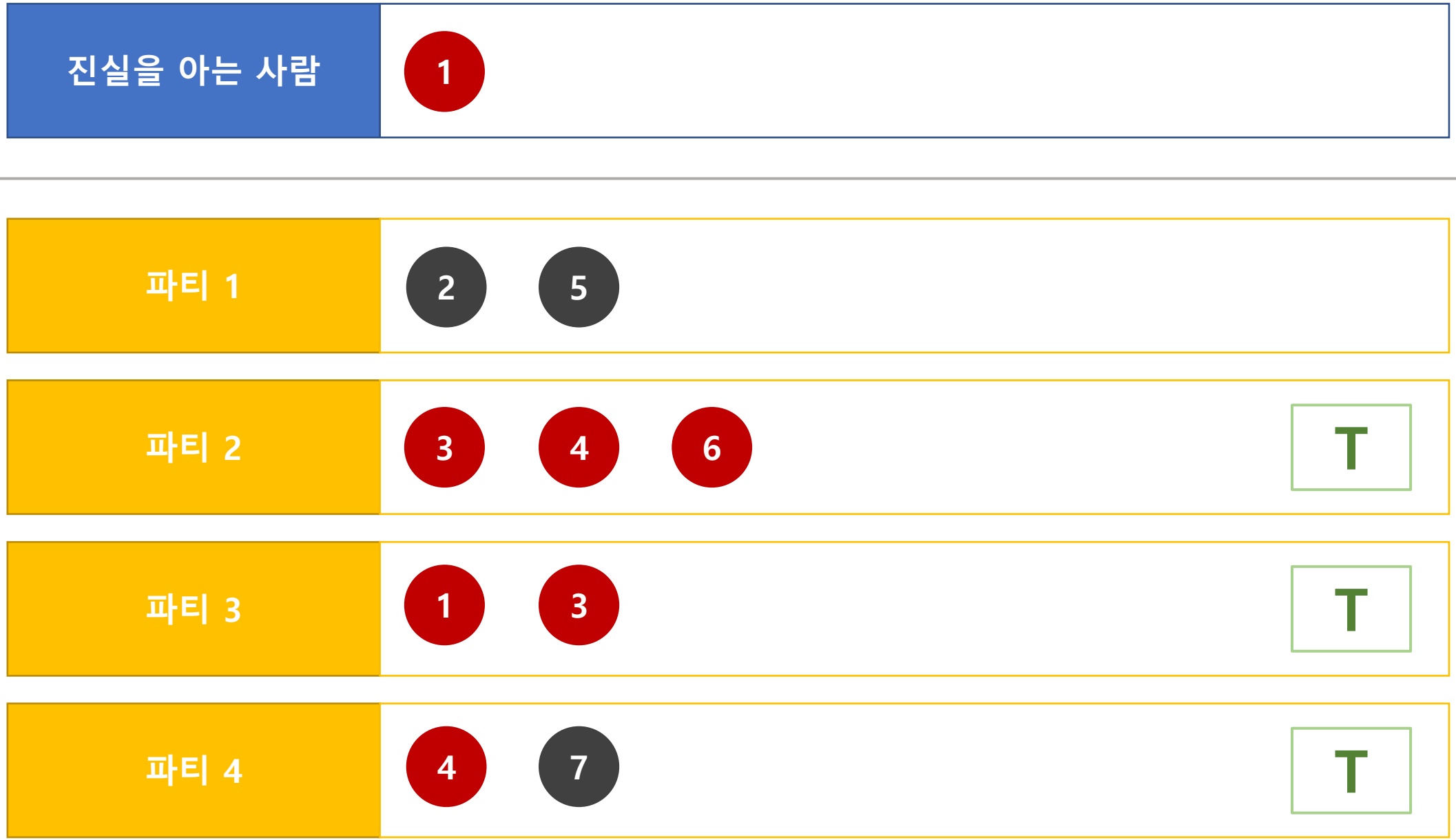
진실을 아는 사람	1

파티 1	2 5
파티 2	3 4 6
파티 3	1 3 T
파티 4	4 7

진실을 아는 사람	1
파티 1	2 5
파티 2	3 4 6
파티 3	1 3 T
파티 4	4 7

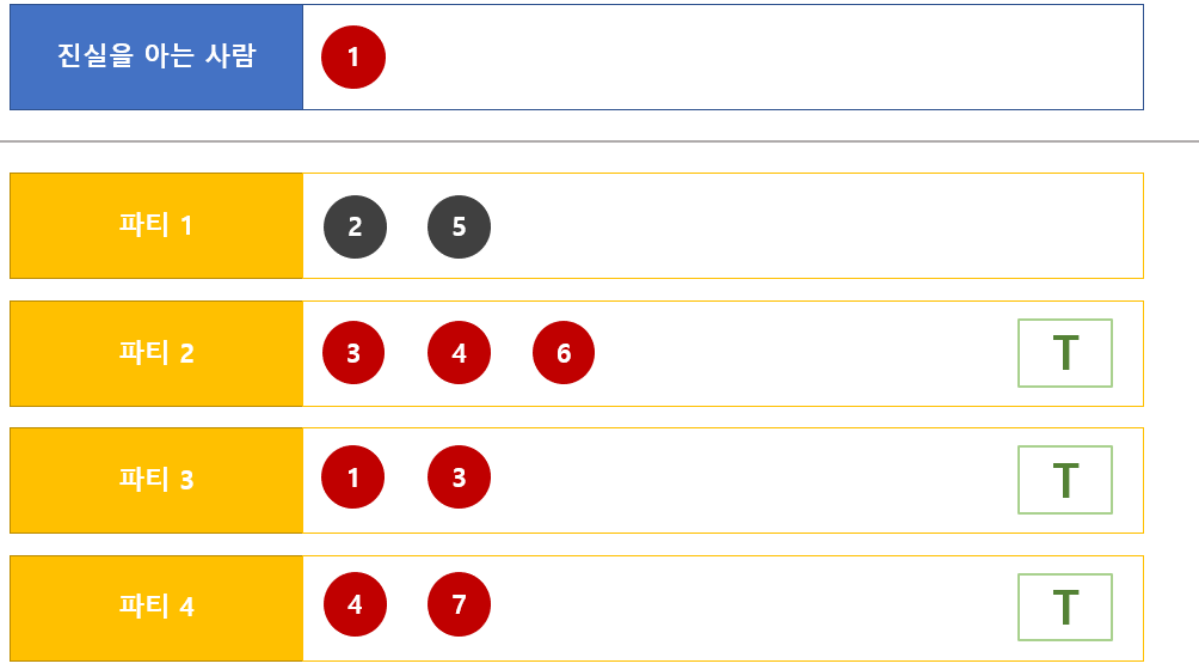
진실을 아는 사람	1	
파티 1	2 5	
파티 2	3 4 6	T
파티 3	1 3	T
파티 4	4 7	

진실을 아는 사람	1	
파티 1	2 5	
파티 2	3 4 6	T
파티 3	1 3	T
파티 4	4 7	


진실을 아는 사람
1
파티 1
2
5
파티 2
3
4
6
T
파티 3
1
3
T
파티 4
4
7
T

진실을 아는 사람	1	
파티 1	2 5	
파티 2	3 4 6	T
파티 3	1 3	T
파티 4	4 7	T

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

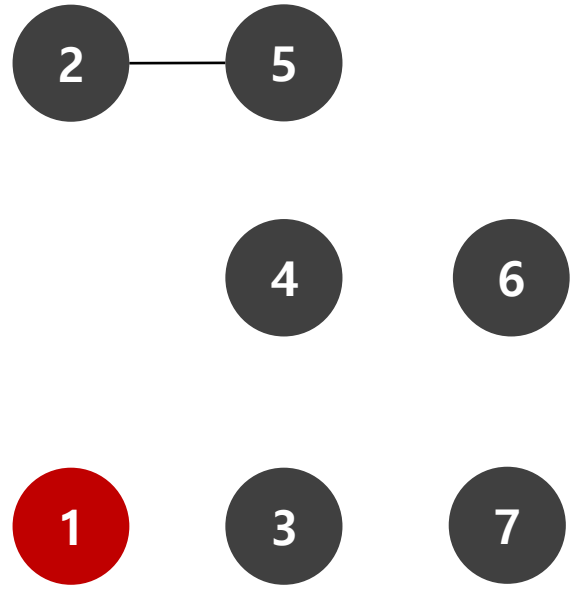
3.[[[진실을 알고 있는 사람]과 함께 파티를 참여한 사람]과 함께 파티를 참여한 사람]이 있는 경우

4....

⇒ 같은 파티에 참여한 사람끼리의 연결 관계를 알아야 함.

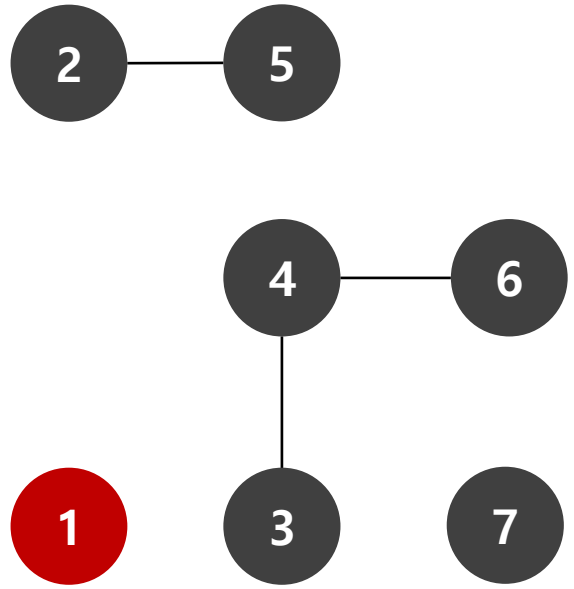
진실을 아는 사람	1

파티 1	2 5
파티 2	3 4 6
파티 3	1 3
파티 4	4 7



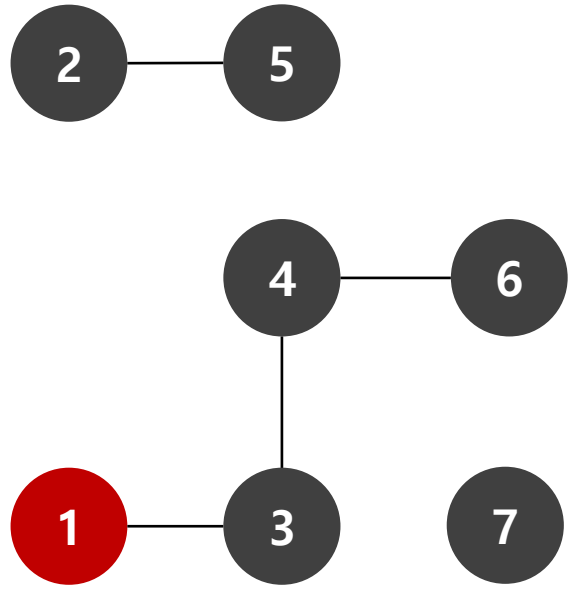
진실을 아는 사람	1

파티 1	2 5
파티 2	3 4 6
파티 3	1 3
파티 4	4 7



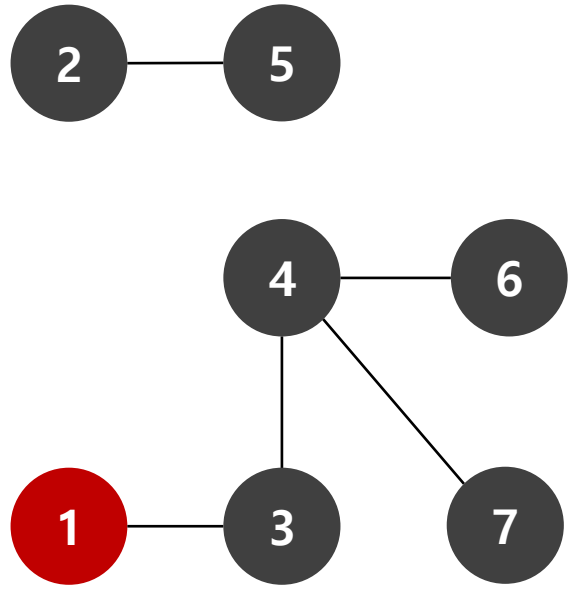
진실을 아는 사람	1

파티 1	2	5
파티 2	3	4	6
파티 3	1	3
파티 4	4	7



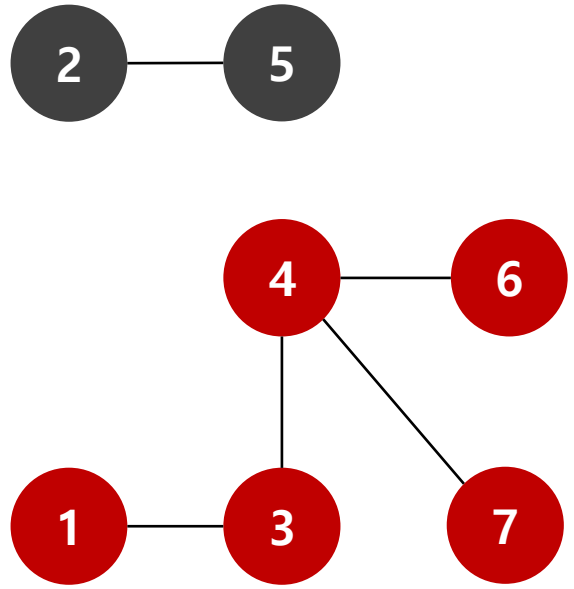
진실을 아는 사람	1

파티 1	2	5
파티 2	3	4	6
파티 3	1	3
파티 4	4	7



진실을 아는 사람	1

파티 1	2	5		
파티 2	3	4	6	T
파티 3	1	3		T
파티 4	4	7		T



풀이

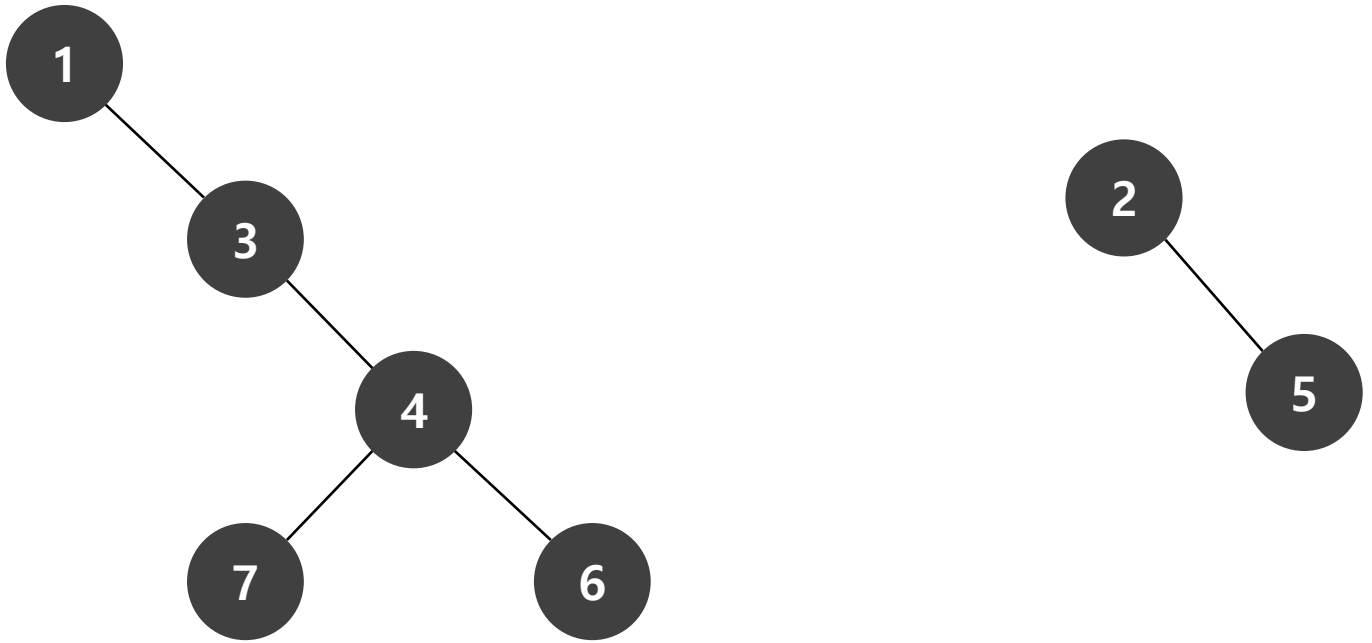
Union-Find 사용

: 합집합을 찾거나, 노드들을 같은 집합으로 만들어주는 알고리즘. 주로 트리로 구현.

⇒ 같은 집합에 포함된 노드들은 같은 root를 가지고 있음.

Parent 배열 (트리 역할)

인덱스	1	2	3	4	5	6	7
저장된 값 (부모 인덱스)	1	2	1	3	2	4	4



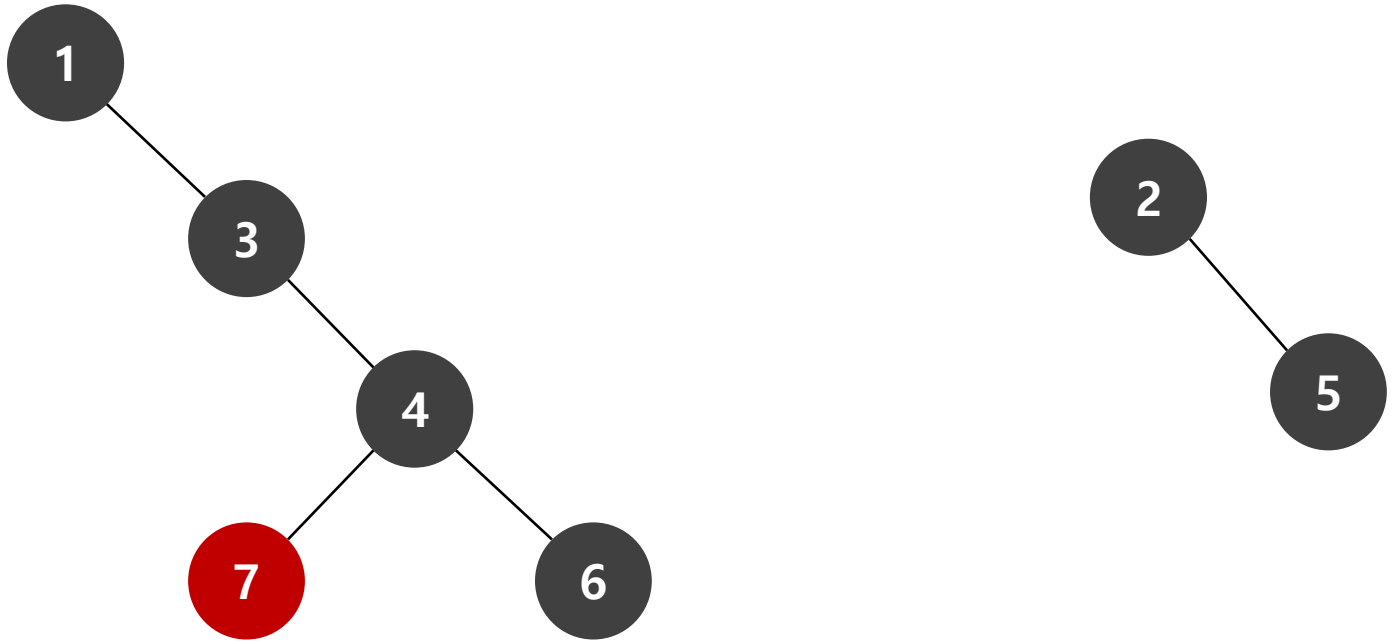
Union-Find

- Find 알고리즘 : root를 찾는 알고리즘. 같은 root를 가지고 있으면 같은 집합이다.
- Union 알고리즘 : 두 노드를 하나의 집합으로 합쳐주는 알고리즘.

find Root 찾기

재귀적으로 부모 노드를 호출 -> (인덱스 값 == 부모 노드의 값) 일때까지 (root 일 때 까지)

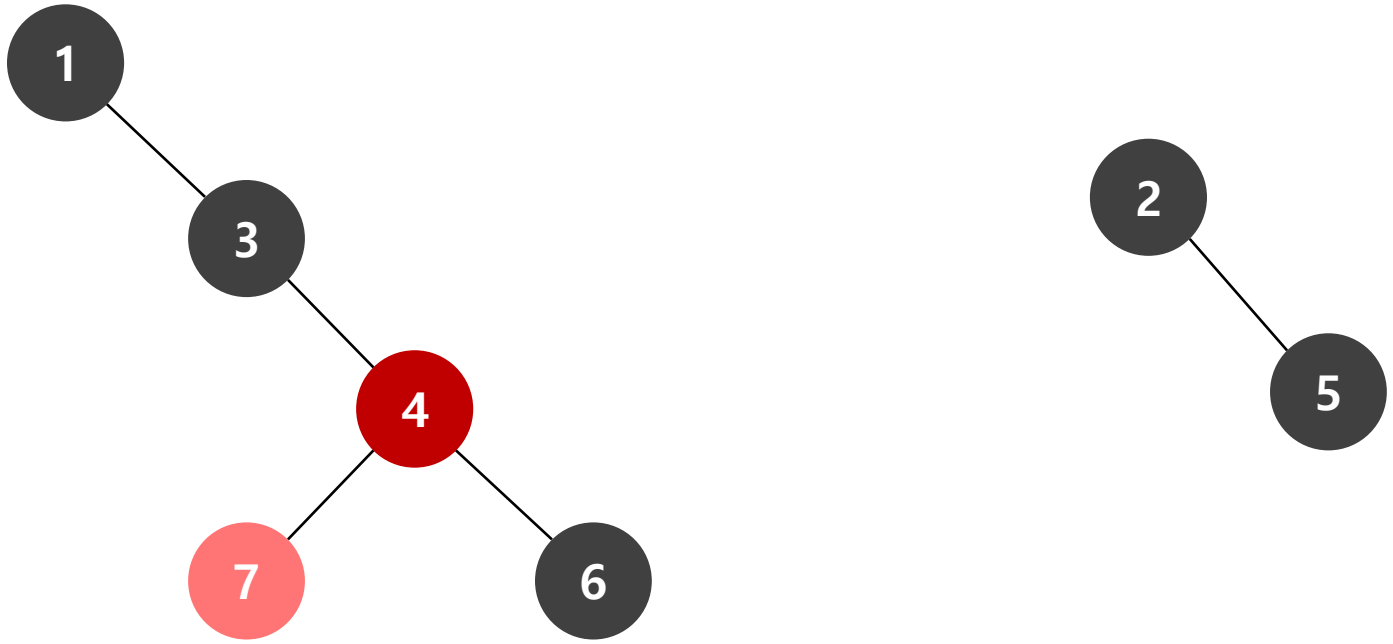
인덱스	1	2	3	4	5	6	7
저장된 값 (부모 인덱스)	1	2	1	3	2	4	4



find Root 찾기

재귀적으로 부모 노드를 호출 -> 인덱스 값 == 부모 노드의 값 일때까지 (root 일 때 까지)

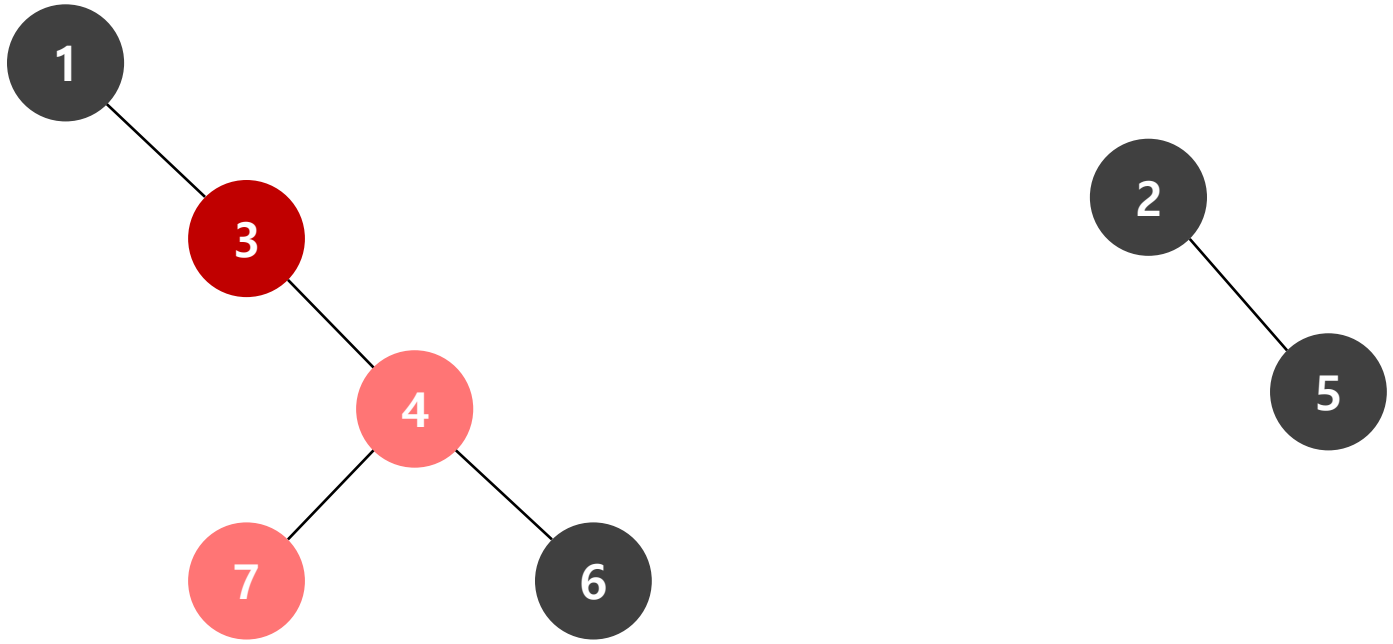
인덱스	1	2	3	4	5	6	7
저장된 값 (부모 인덱스)	1	2	1	3	2	4	4



find Root 찾기

재귀적으로 부모 노드를 호출 -> 인덱스 값 == 부모 노드의 값 일때까지 (root 일 때 까지)

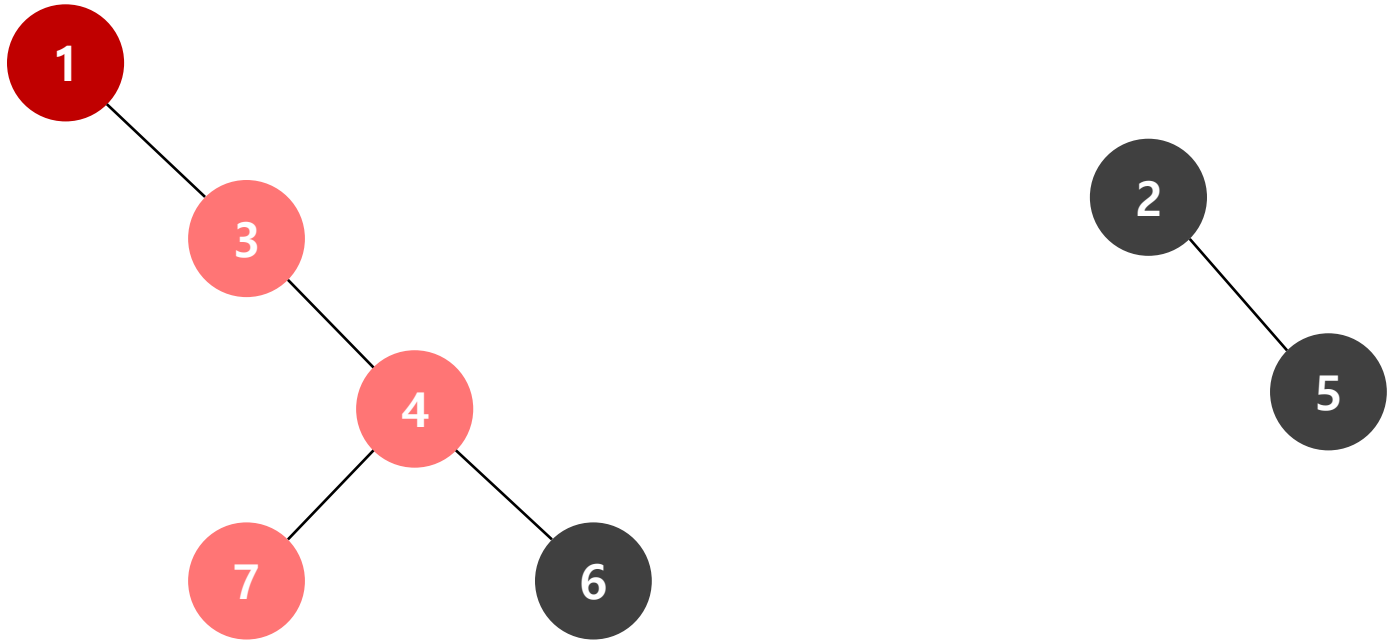
인덱스	1	2	3	4	5	6	7
저장된 값 (부모 인덱스)	1	2	1	3	2	4	4



find Root 찾기

재귀적으로 부모 노드를 호출 -> 인덱스 값 == 부모 노드의 값 일때까지 (root 일 때 까지)

인덱스	1	2	3	4	5	6	7
저장된 값 (부모 인덱스)	1	2	1	3	2	4	4



find

재귀적으로 부모 노드를 호출 -> 인덱스 값 == 부모 노드의 값 일때까지 (root 일 때 까지)

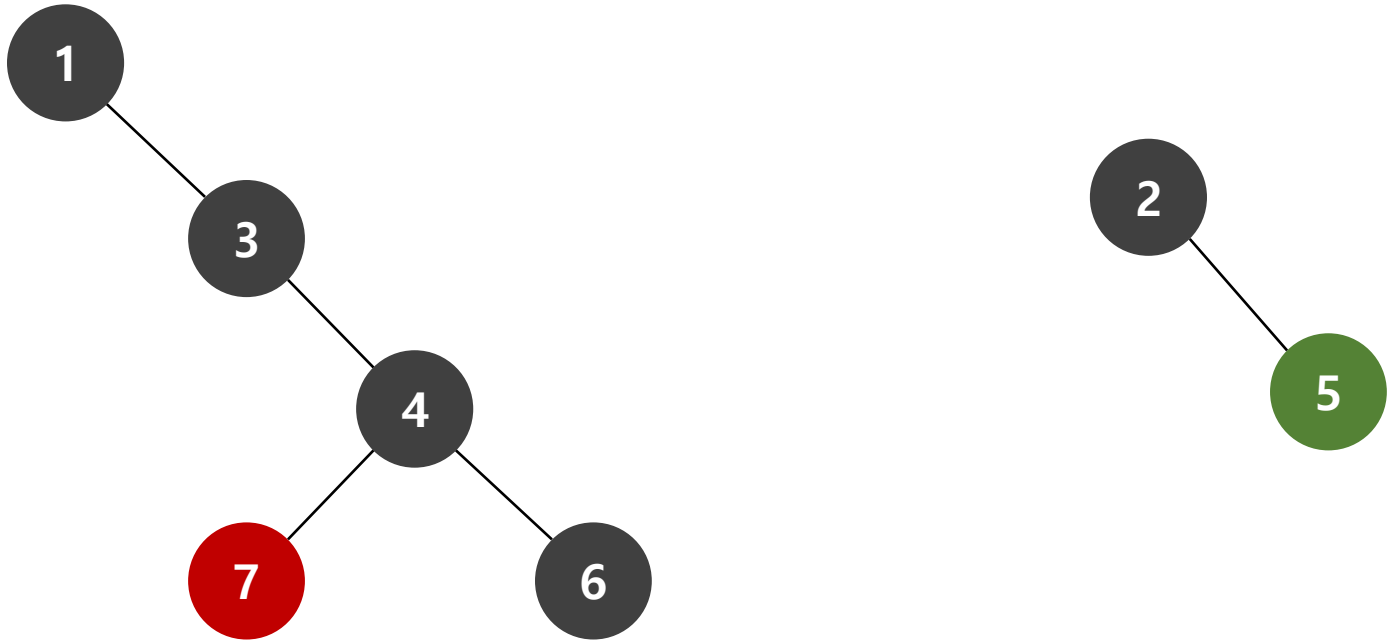
```
int findParent(int idx){
    if(parent[idx] == idx)
        return idx;

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

Union 하나의 집합으로 만들기

루트 노드 값을 비교하여 더 작은 값을 새로운 루트 노드로 삼음

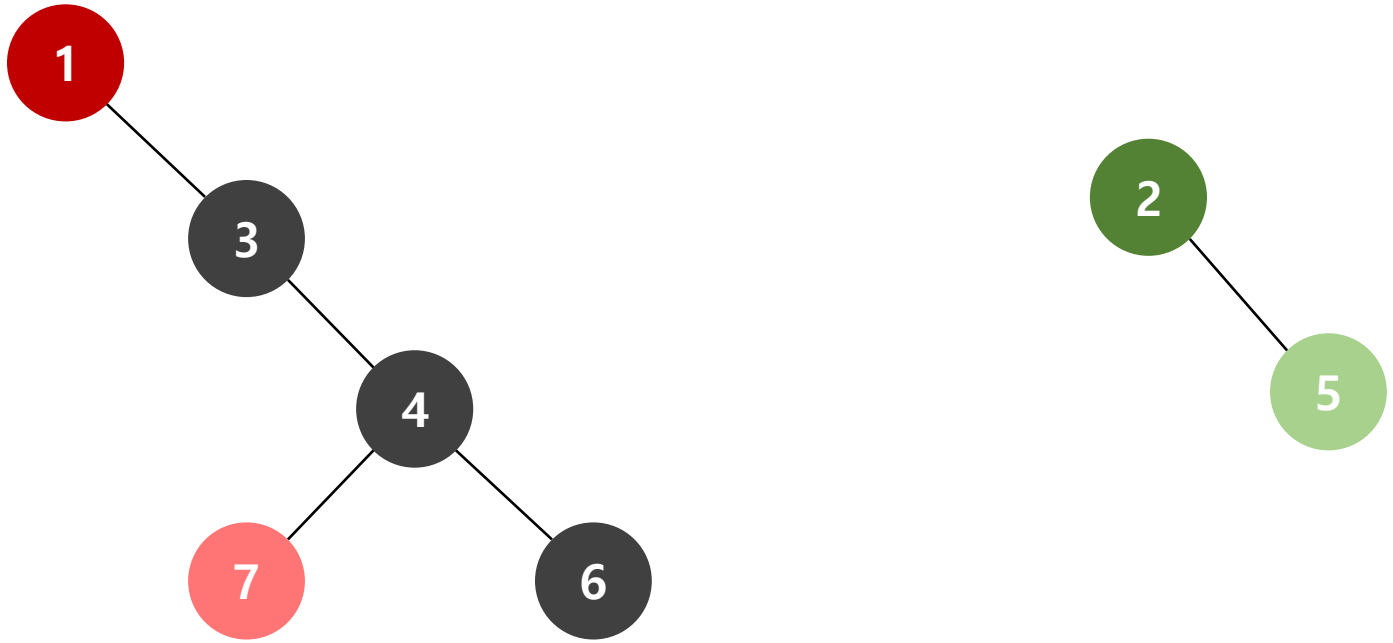
인덱스	1	2	3	4	5	6	7
저장된 값 (부모 인덱스)	1	2	1	3	2	4	4



Union

루트 노드 값을 비교하여 더 작은 값을 새로운 루트 노드로 삼음

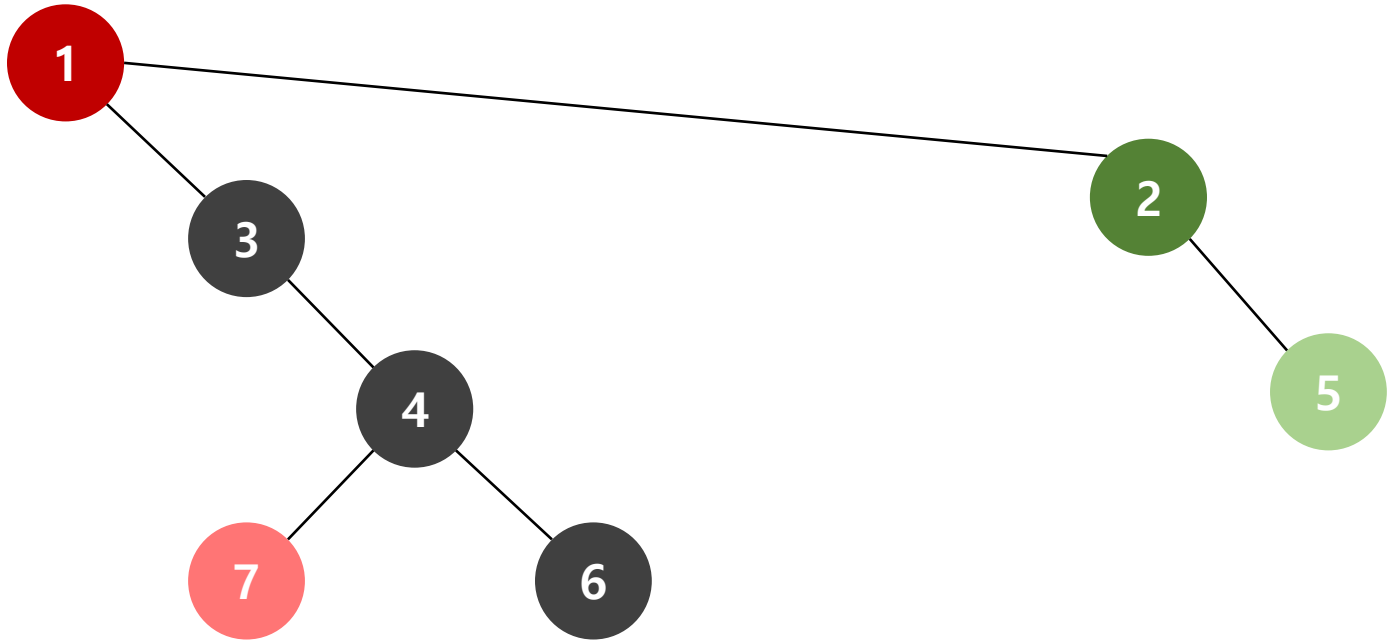
인덱스	1	2	3	4	5	6	7
저장된 값 (부모 인덱스)	1	2	1	3	2	4	4



Union

루트 노드 값을 비교하여 더 작은 값을 새로운 루트 노드로 삼음

인덱스	1	2	3	4	5	6	7
저장된 값 (부모 인덱스)	1	**1**	1	3	2	4	4



Union

루트 노드 값을 비교하여 더 작은 값을 새로운 루트 노드로 삼음

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

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

문제 풀이 방법

1. 진실을 알고 있는 사람들의 parent[idx] 값을 -1로 설정 (진실 아는 경우 root값이 -1)
2. 전체 파티당 참가인원 저장하며 union-find로 같은 파티에 참여한 사람들을 하나의 집합으로 만들기
3. 파티 당 참가인원 조회하며 root 값이 -1인 경우 cnt ++ (진실 말해야하는 경우 cnt++)
4. 거짓말할 수 있는 파티 개수 찾는 것이므로 [전체 파티개수 - cnt] 출력