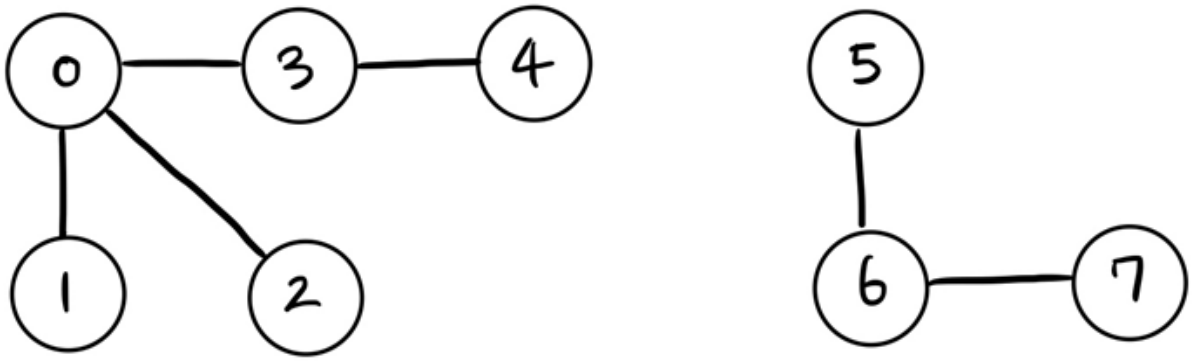
Union-Find

정의

- 서로소 집합을 찾는 알고리즘
  - 어떤 두 임의의 원소를 선택했을 때 그 두 원소가 같은 집합에 속하는지 판별하는 방법
- 최소신장트리 알고리즘에 많이 쓰임

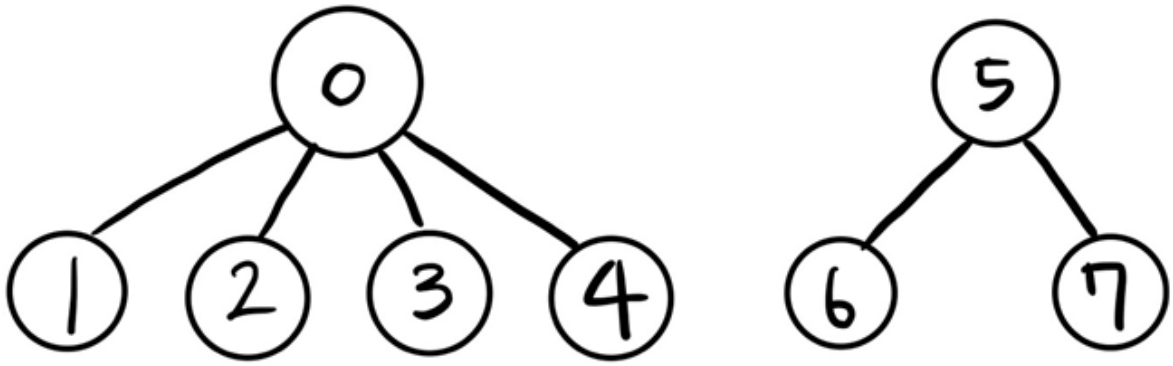
다음과 같은 두 그래프가 있다고 하자.



노드 2, 4가 같은 그래프에 있나?

맨 처음 생각나는 방법으로는 2번 노드에서 시작해 0 → 3 → 4의 과정을 거쳐 2와 4가 연결되어있음을 확인할 것이다.

하지만 2와 4 사이에 수억 개의 노드가 존재한다면 이 과정은 상당히 비효율적이다.

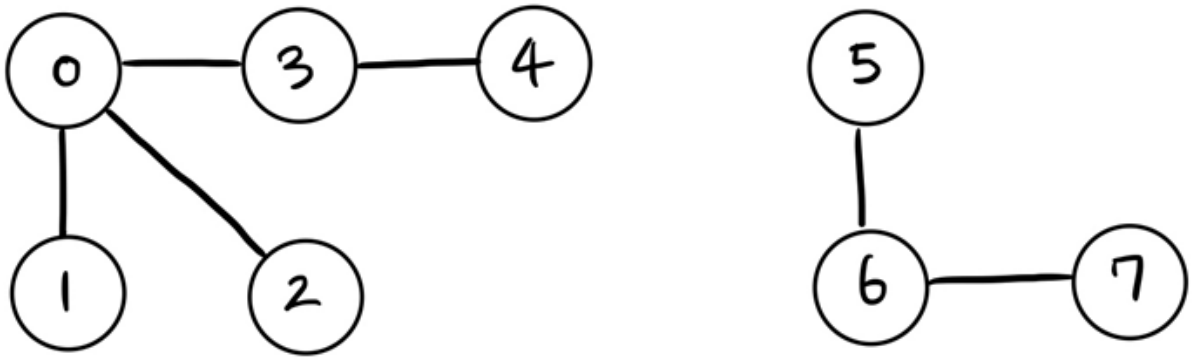


그래서 위와 같이, **각 집합의 부모(대표)를 정해놓고** 해당 멤버들은 **그 부모를 가리키도록** 한다. 이렇게 하면 같은 부모를 가진 노드들은 같은 그룹에 속한다는 것을 알 수 있다.

동작 방식

먼저, 각 노드들의 부모를 저장할 공간을 만들어, 처음에는 **부모를 자기 자신으로 초기화**한다.

노드번호	0	1	2	3	4	5	6	7
부모번호	0	1	2	3	4	5	6	7



다음 그래프에서 보면 0번 노드와 1번 노드는 연결되어 있다.

따라서 부모를 **Union**할 수 있다. 1번 노드의 부모를 0으로 바꿔주자.

노드번호	0	1	2	3	4	5	6	7
부모번호	0	**0**	2	3	4	5	6	7

위와 같은 이유로 2와 3의 부모 또한 0으로 바꿀 수 있다.

그렇다면, 4번 노드는?

4번 노드는 3번 노드와 연결되어있기 때문에, 4의 부모는 3이 될 수 있다.

그런데,

노드번호	0	1	2	3	4	5	6	7
부모번호	0	0	0	0	4	5	6	7

3번 노드의 부모가 자기 자신이 아니다.

즉, 3번 노드가 다른 노드를 부모로 가리키고 있다.

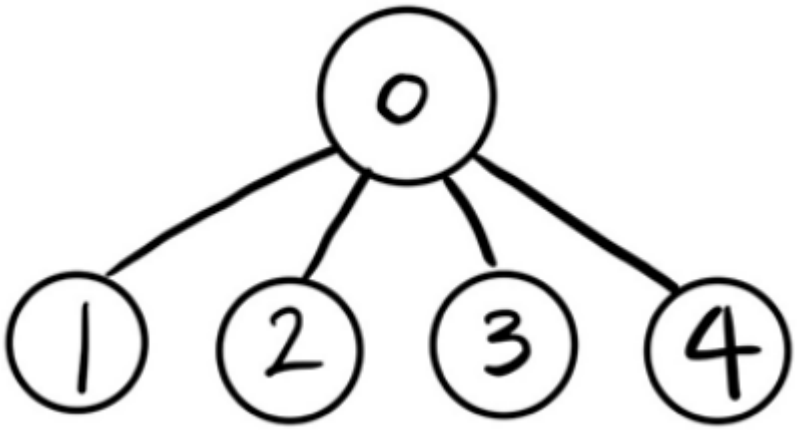
이 경우, **부모 노드로 가리키고 있는 다른 노드로 가서 확인**한다.

0번 노드를 확인해봤더니 부모가 자기 자신이다. 즉, 해당 노드가 부모 노드이다.

따라서 4번 노드의 부모는 최종적으로 0이 된다.

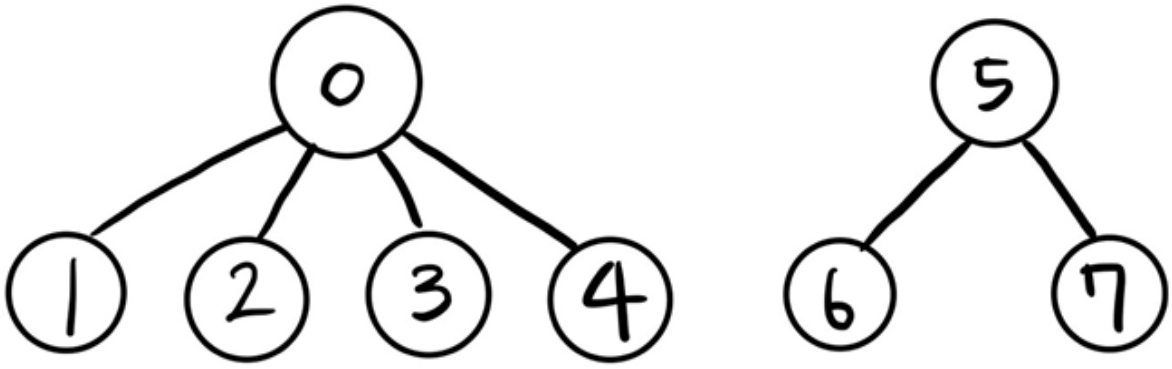
> Union-Find에서 부모 노드를 갱신할 때, **자기 자신을 부모로 가지는 노드를 찾을 때까지 깊이 탐색**을 하게 된다.

이 과정까지 마치면 다음와 같이 구성된다.



노드번호	0	1	2	3	4	5	6	7
부모번호	0	0	0	0	0	5	6	7

나머지 노드들도 같은 과정을 통해 부모를 갱신할 수 있다.



노드번호	0	1	2	3	4	5	6	7
부모번호	0	0	0	0	0	5	5	5

이렇게 같은 부모를 가지는 노드들을 하나의 집합으로 묶을 수 있게 되었다.

함수

여기서의 함수는 크게 두 가지로 나뉜다.

- Union - 두 노드를 같은 집합으로 합친다.
- Find - 해당 노드의 부모 노드를 찾는다.

부모번호를 저장하는 배열을 group이라 하자.

Union

```
void union(int a, int b) {
  int aGroup = find(a);
  int bGroup = find(b);

  group[bGroup] = aGroup;
}
```

Find

```
int find(int node) {
  if(node == group[node])
    return node;
  return group[node] = find(group[node]);
}
```