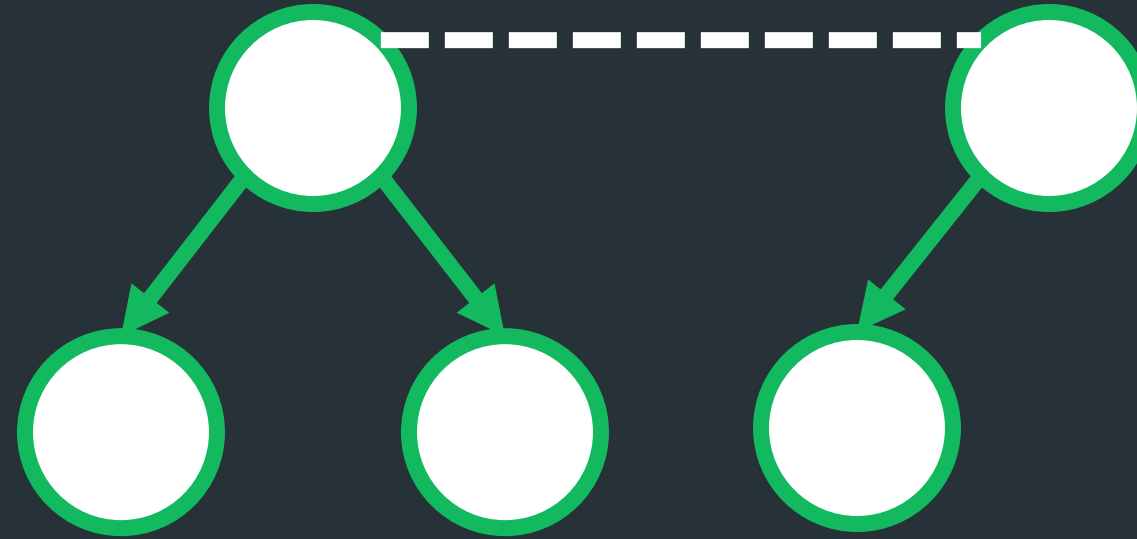


# 알튜브 유니온 파인드

서로 다른 두 원소가 같은 집합에 있는지 판별하는 유니온 파인드 알고리즘입니다.  
시간 복잡도가  $O(1)$ 에 가까운 아주 빠른 알고리즘입니다.



## Union Find

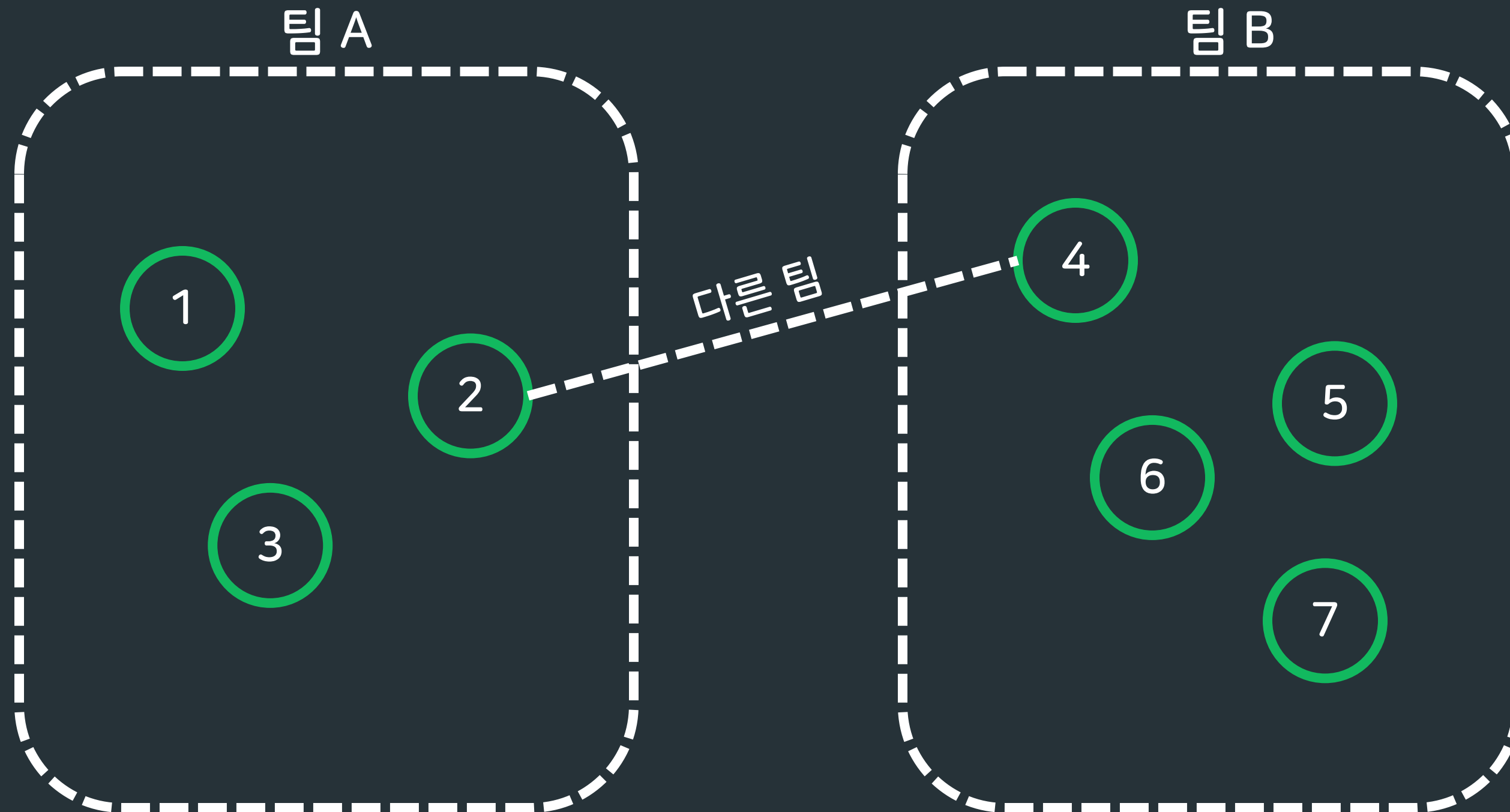
- 서로 다른 두 원소가 같은 집합에 속해 있는지 판별하는 알고리즘
- Find 연산과 Union 연산으로 이루어짐
  - Find : 각 원소가 속한 집합을 판별하는 연산
  - Union : 서로 다른 집합에 속한 두 원소를 같은 집합에 속하도록 합치는 연산
- 분리 집합(Disjoint-set)으로 불리기도 함
- 시간 복잡도는  $O(a(N))$ 으로  $O(1)$ 에 가까움



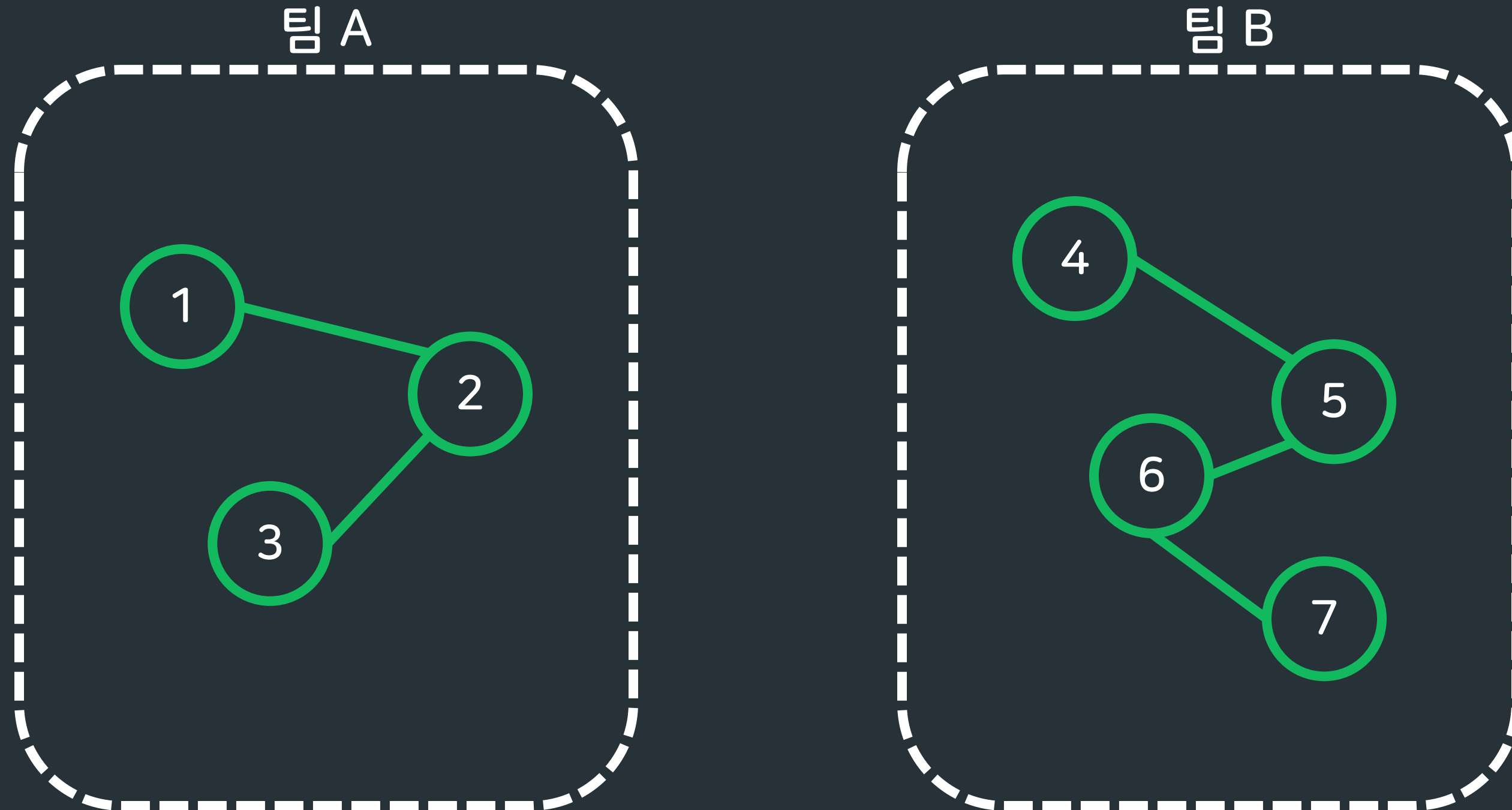
임의의 두 선수가 같은 팀에 속해 있는가?



임의의 두 선수가 같은 팀에 속해 있는가?

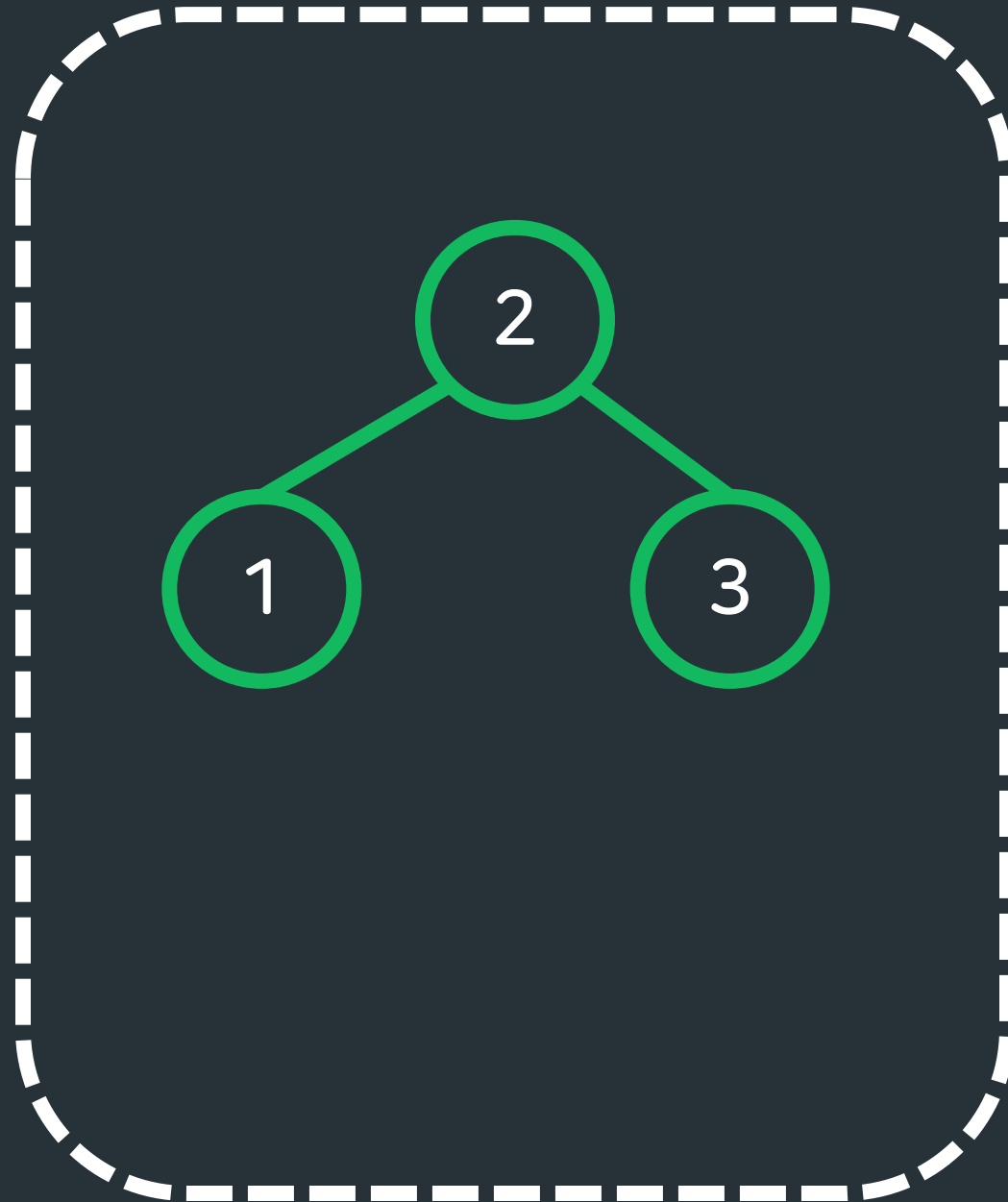


임의의 두 선수가 **같은 팀**에 속해 있는가?

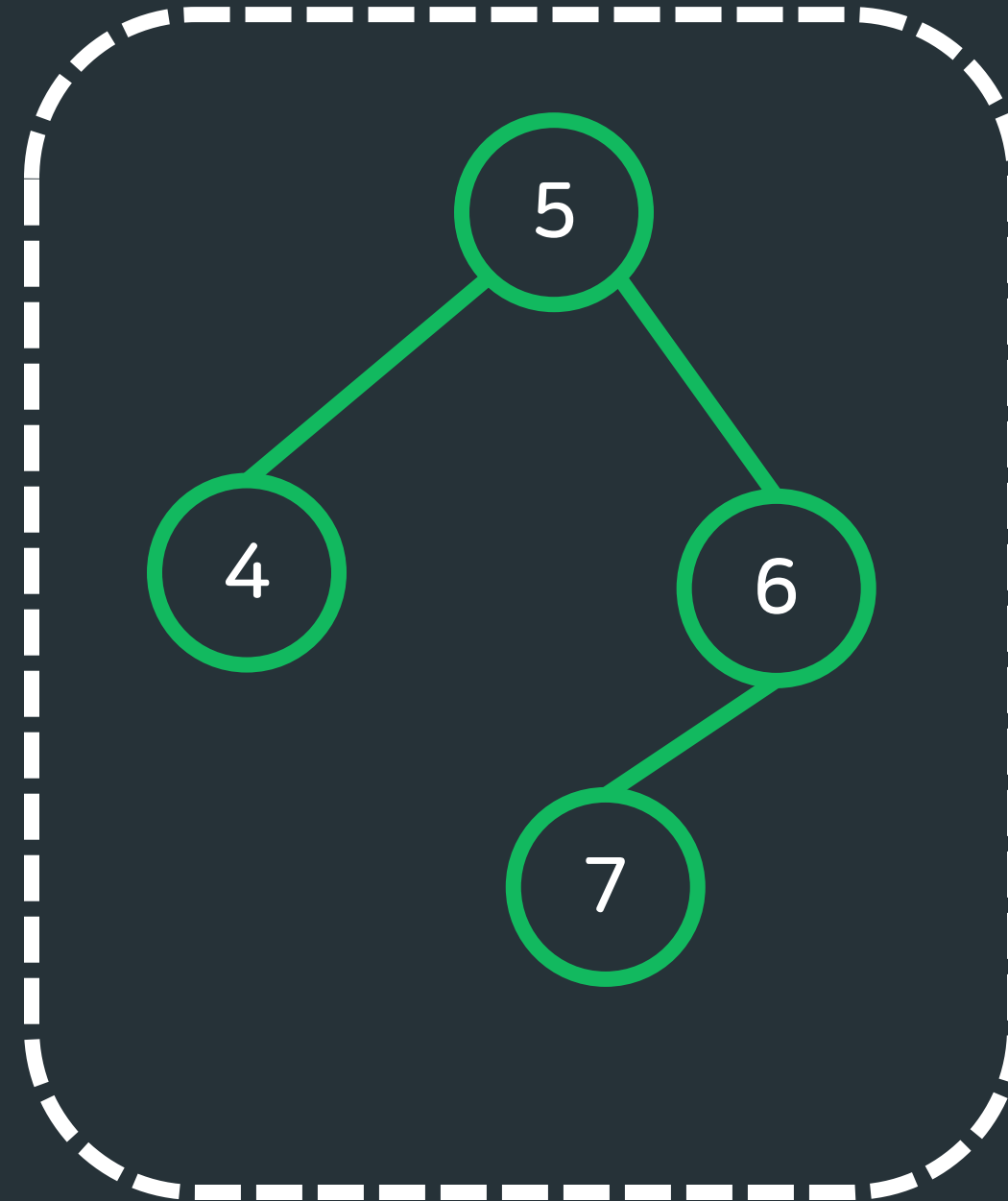


같은 팀끼리 연결

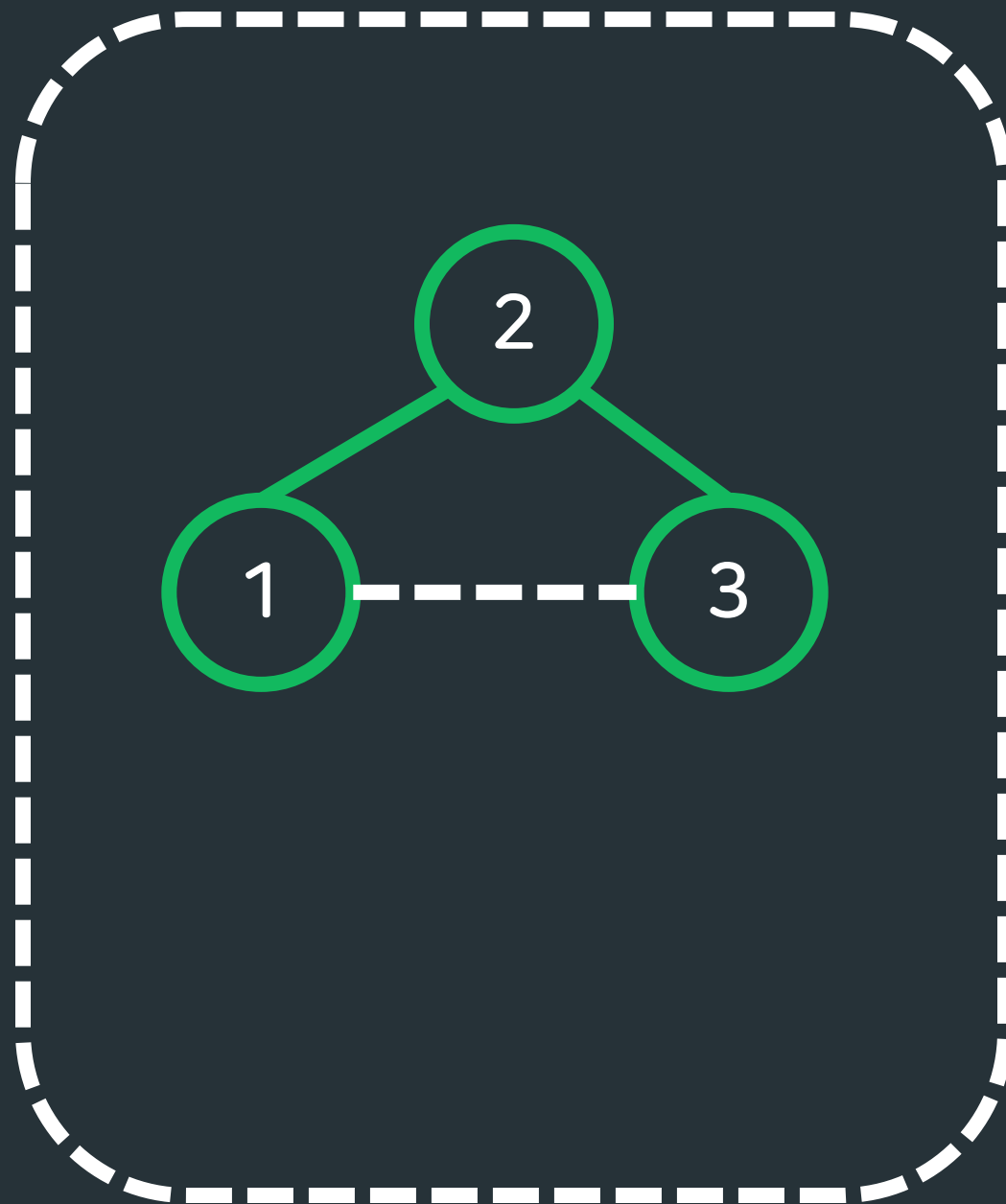
팀 A의 대표 선수는 2번



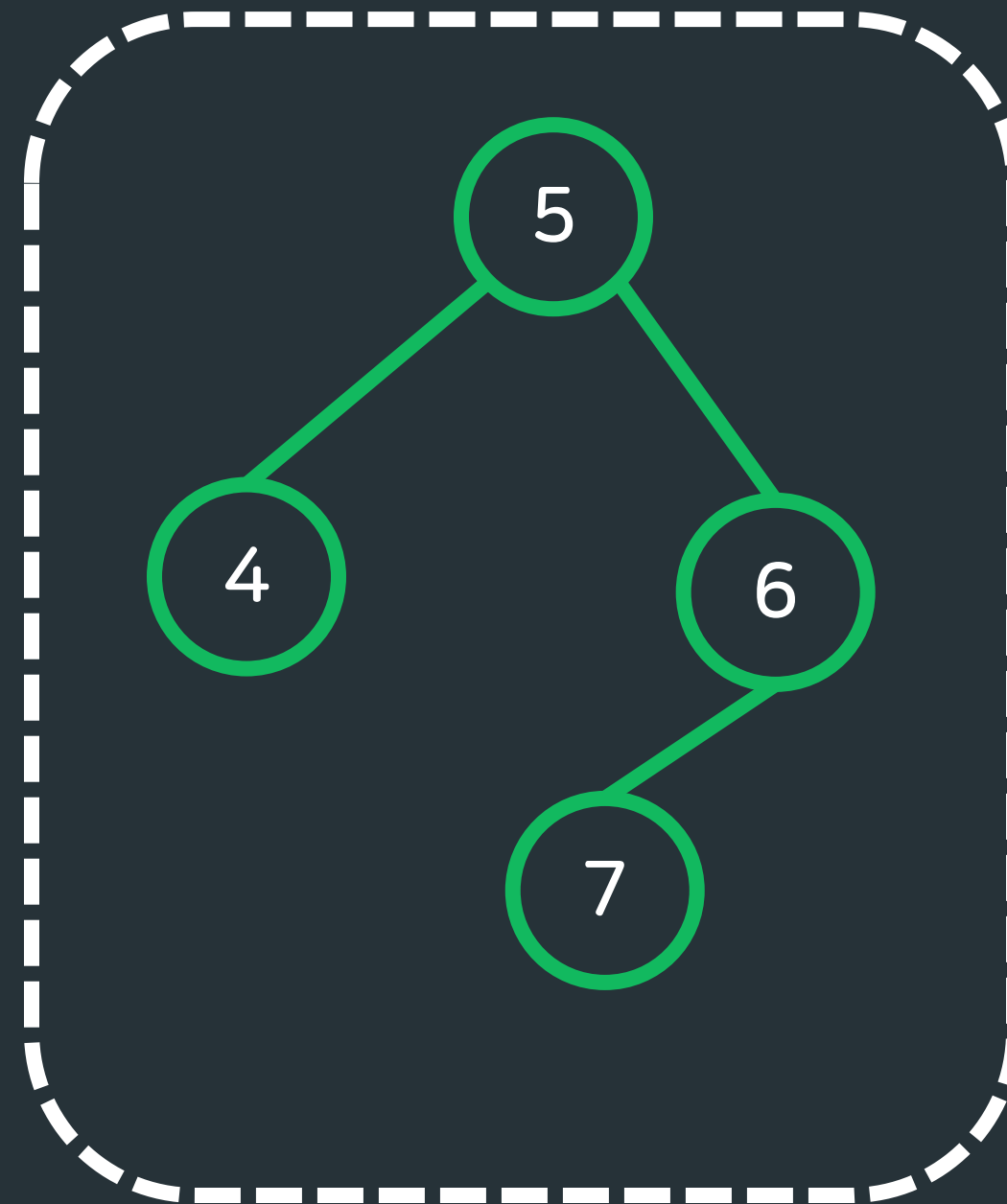
팀 B의 대표 선수는 5번



팀 A의 대표 선수는 2번

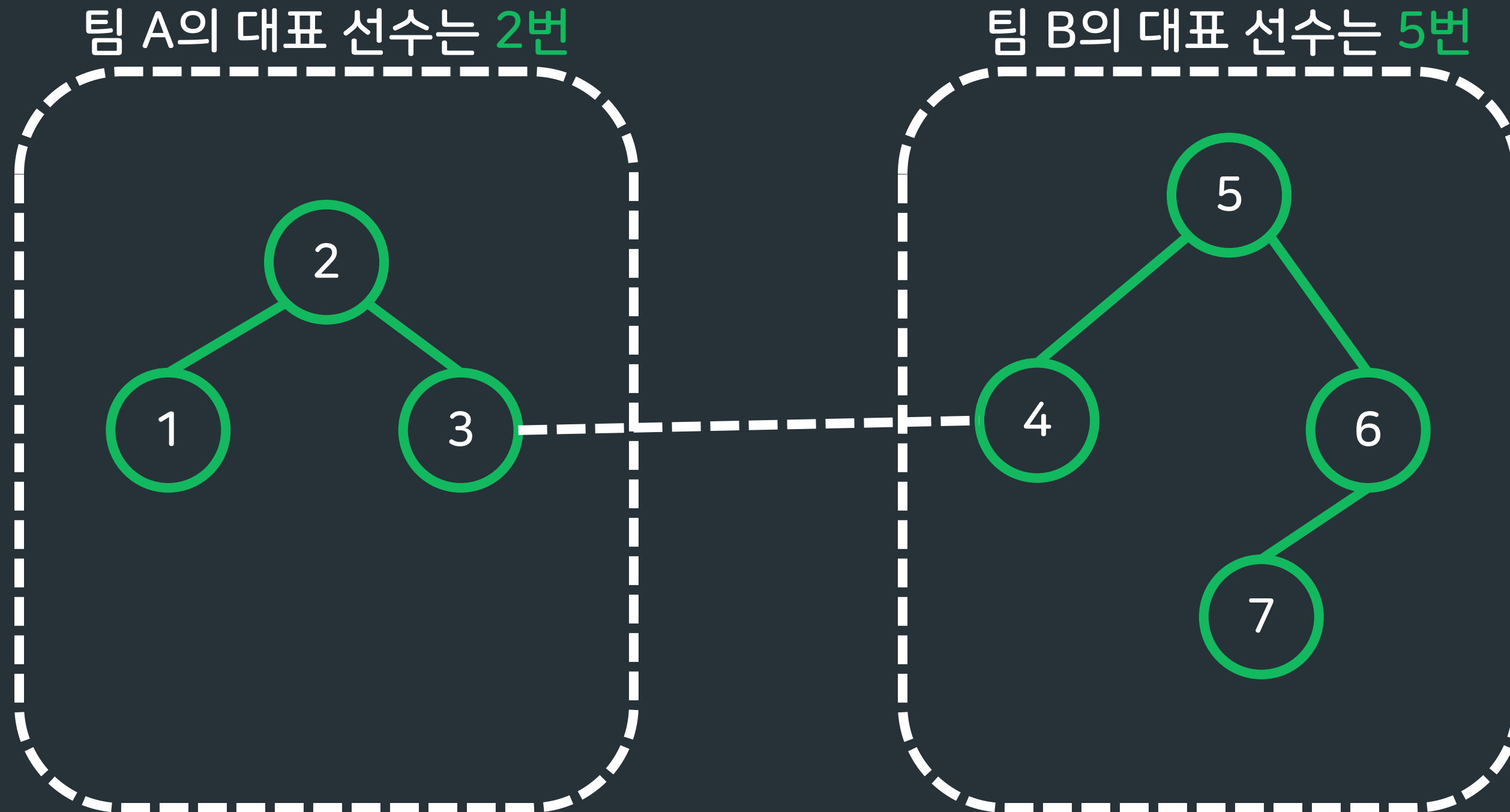


팀 B의 대표 선수는 5번

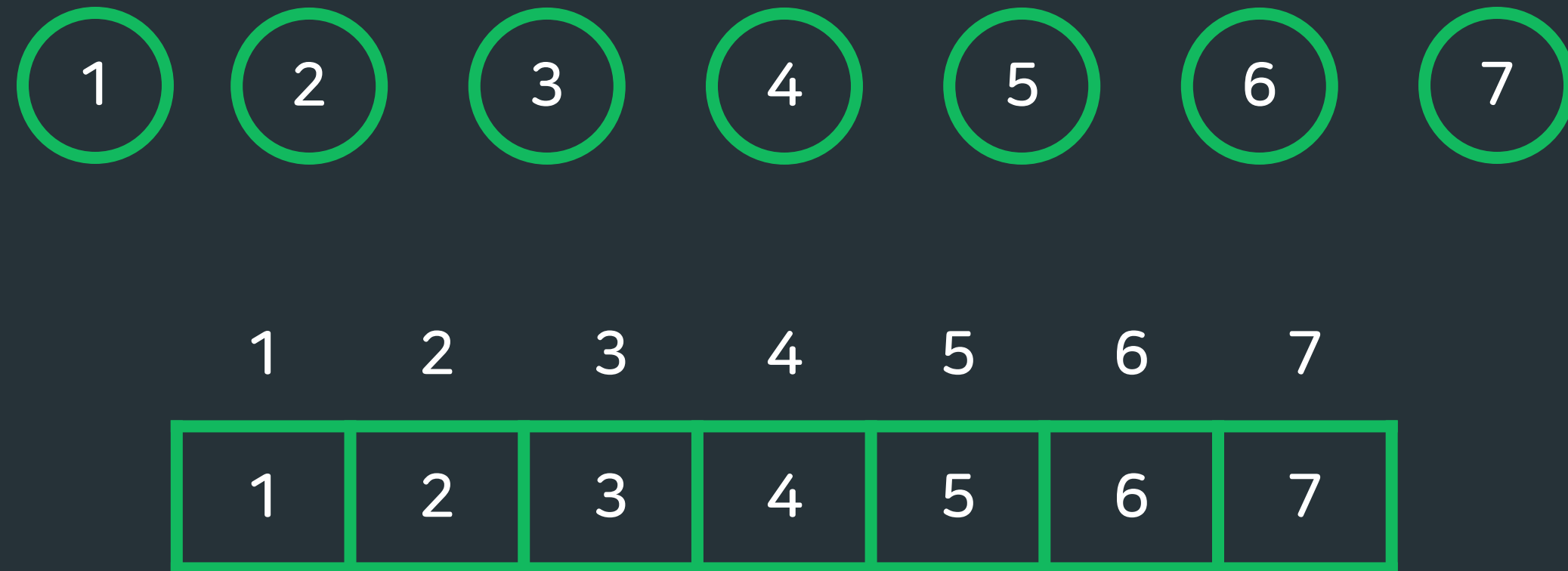


대표 선수가 같으므로 같은 팀!  
(연결하면 **사이클 발생**)

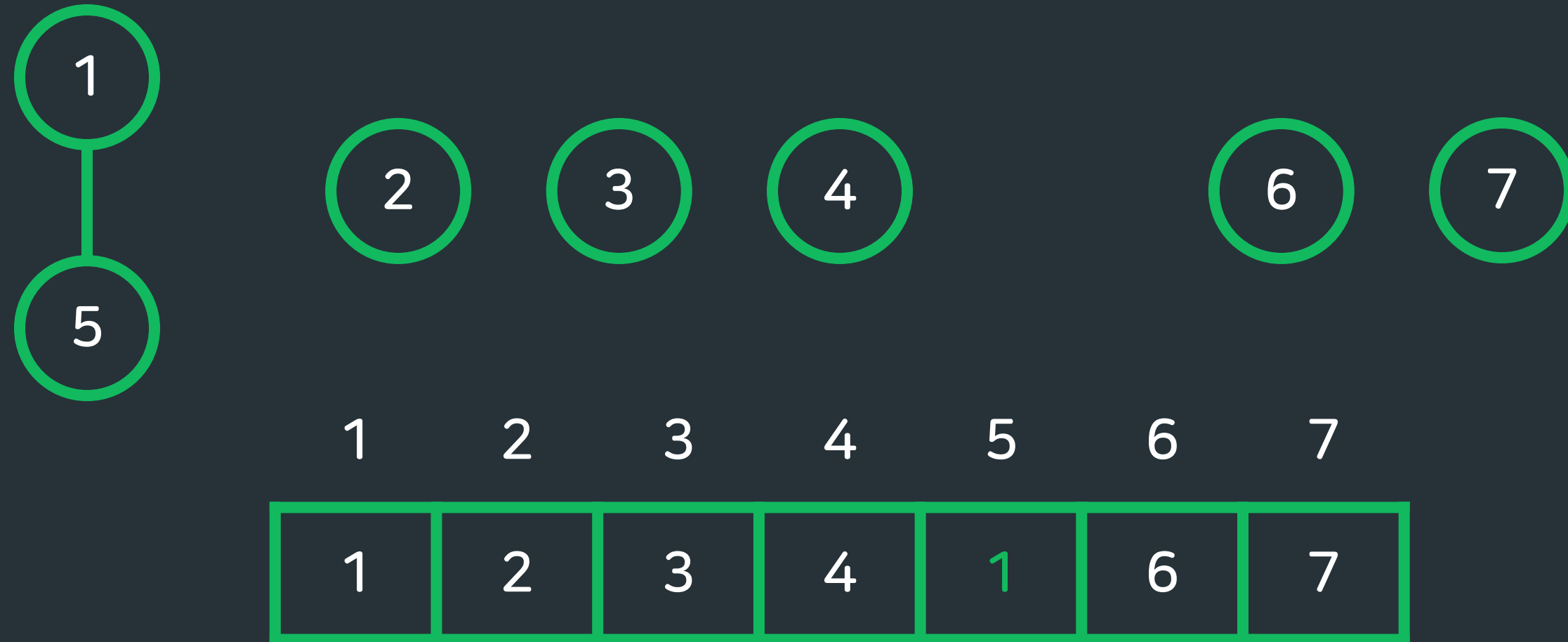


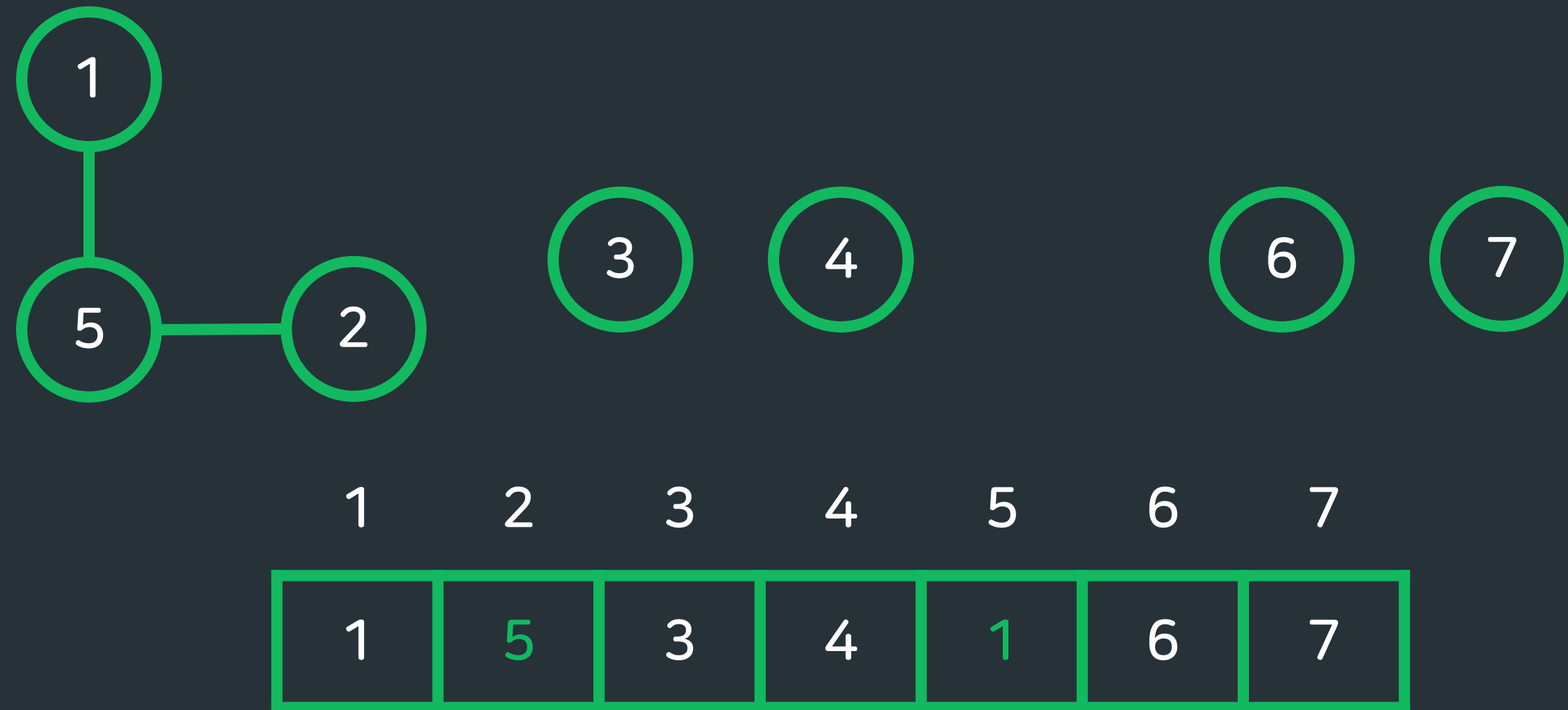


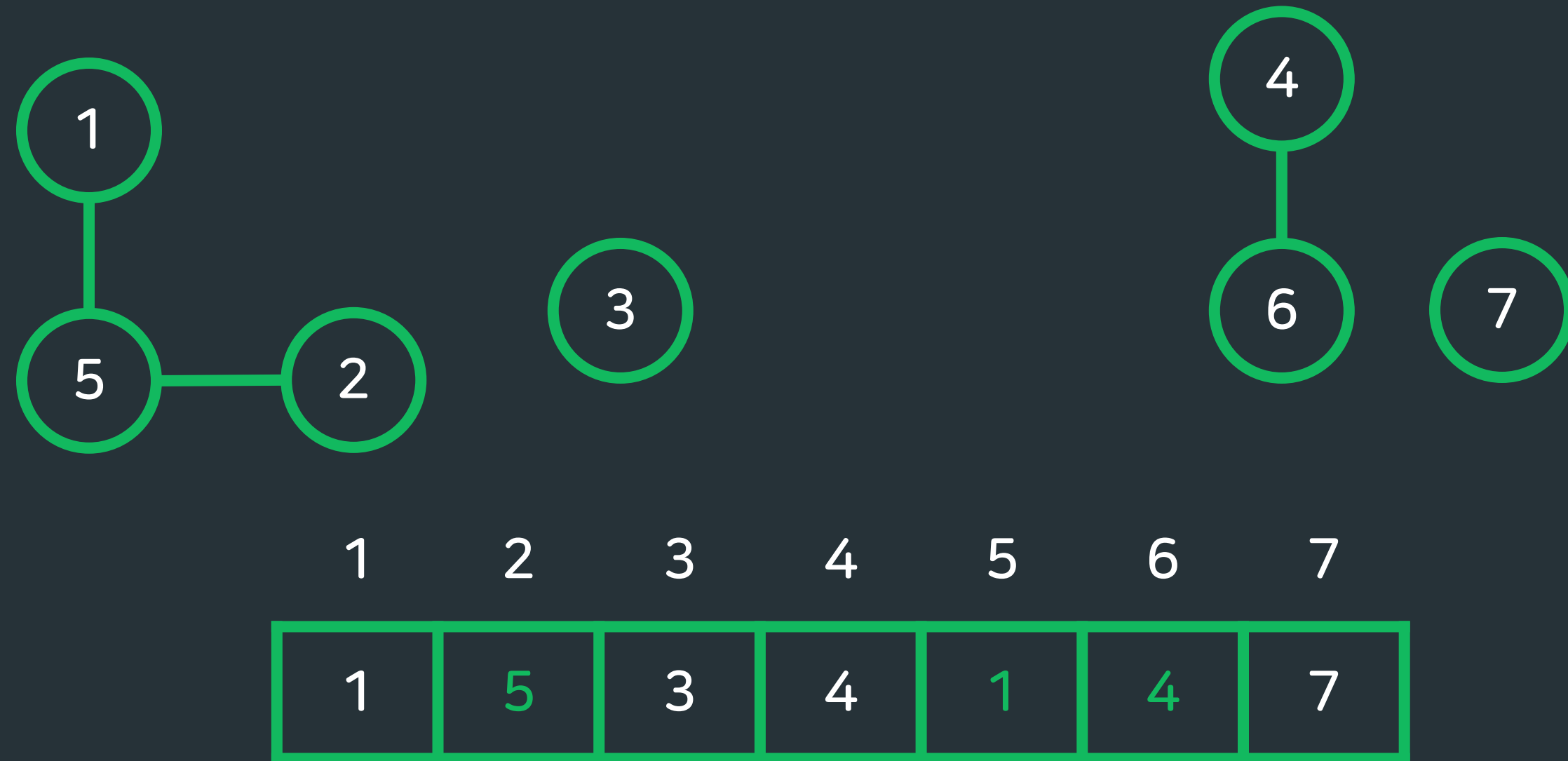
대표 선수가 다르므로 다른 팀!  
(연결해도 사이클 발생하지 않음)

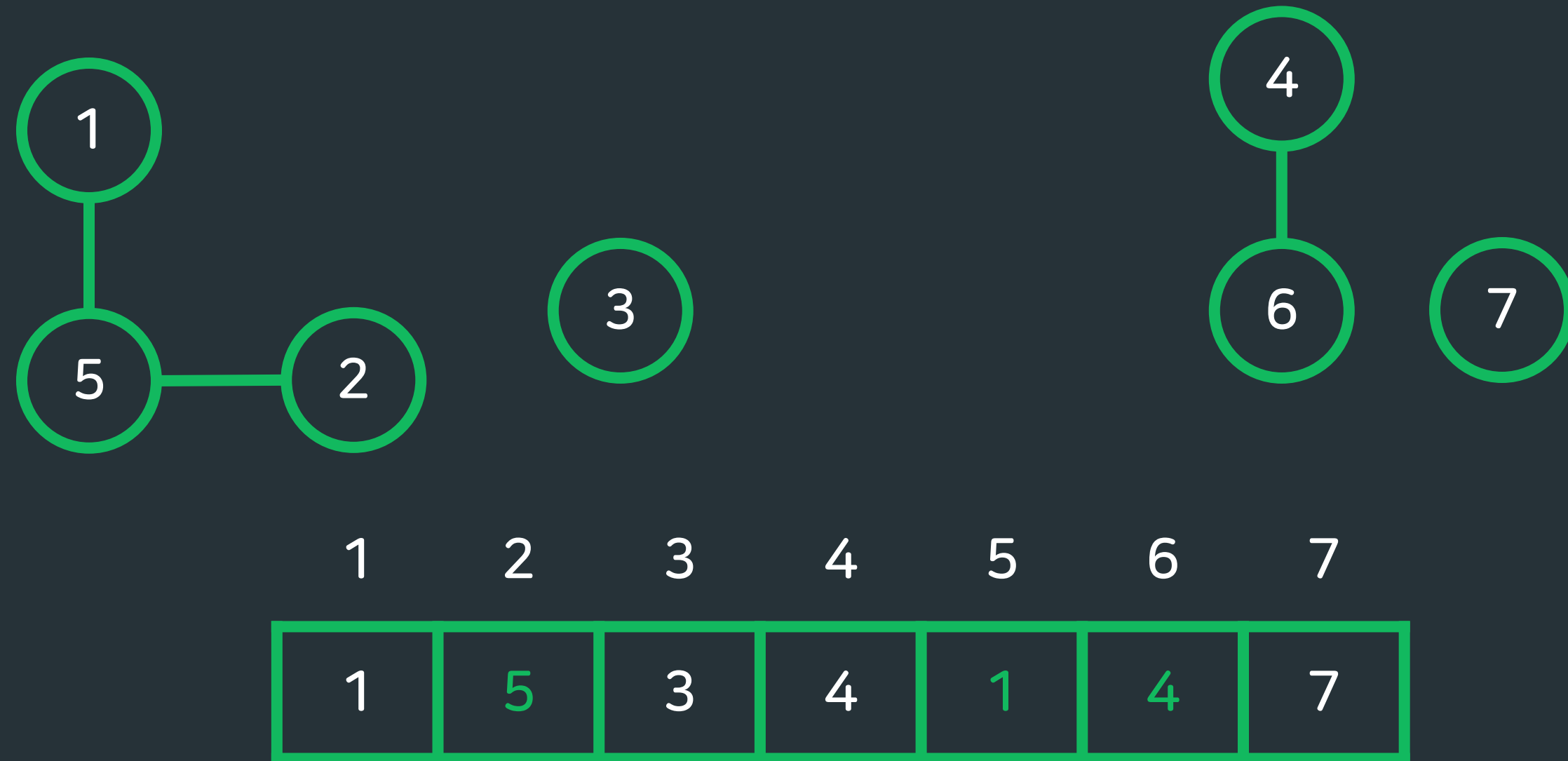


모든 정점의 부모를 자기 자신으로 초기화

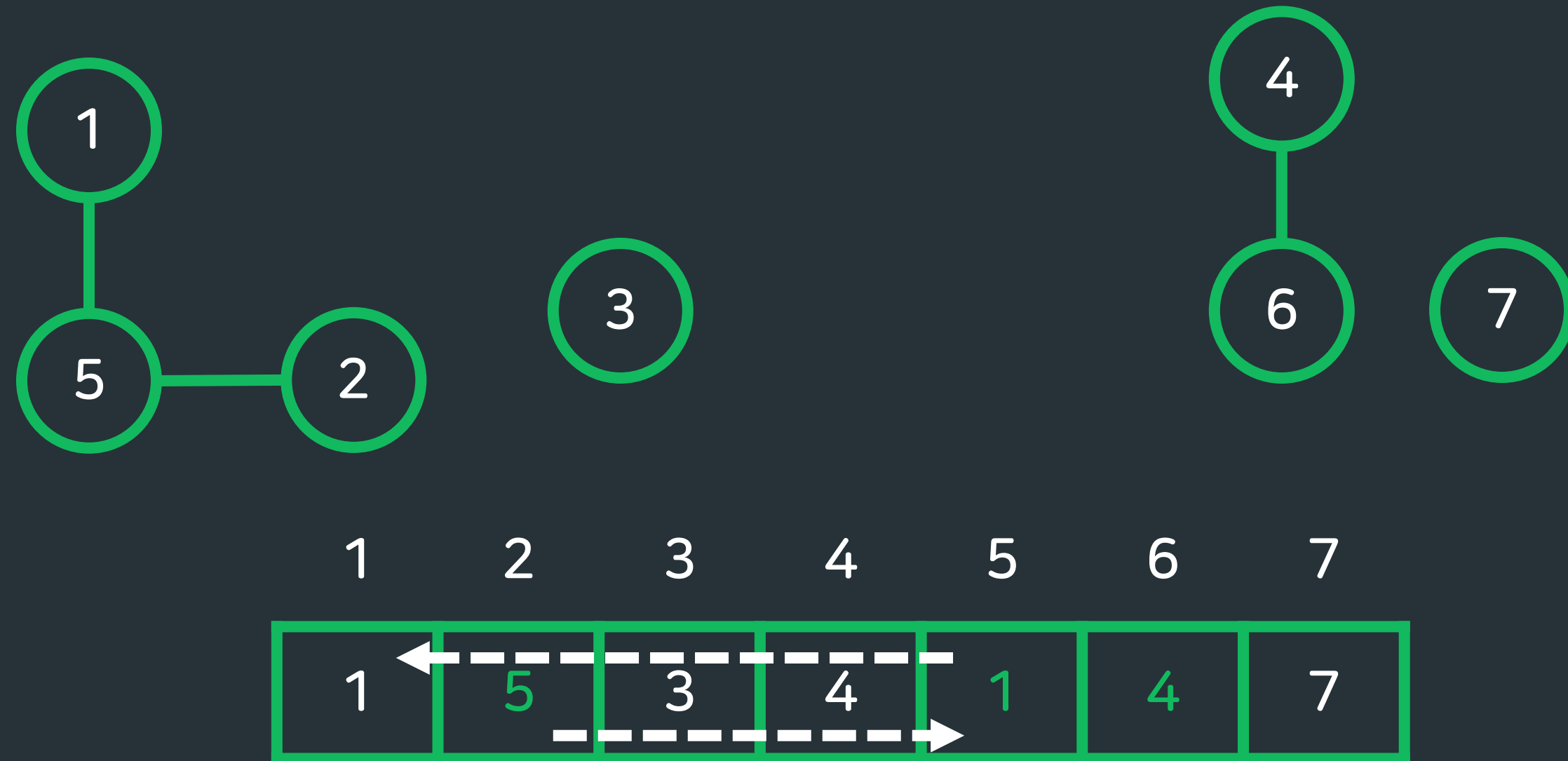








2번 정점이 속한 트리의 루트 정점은?



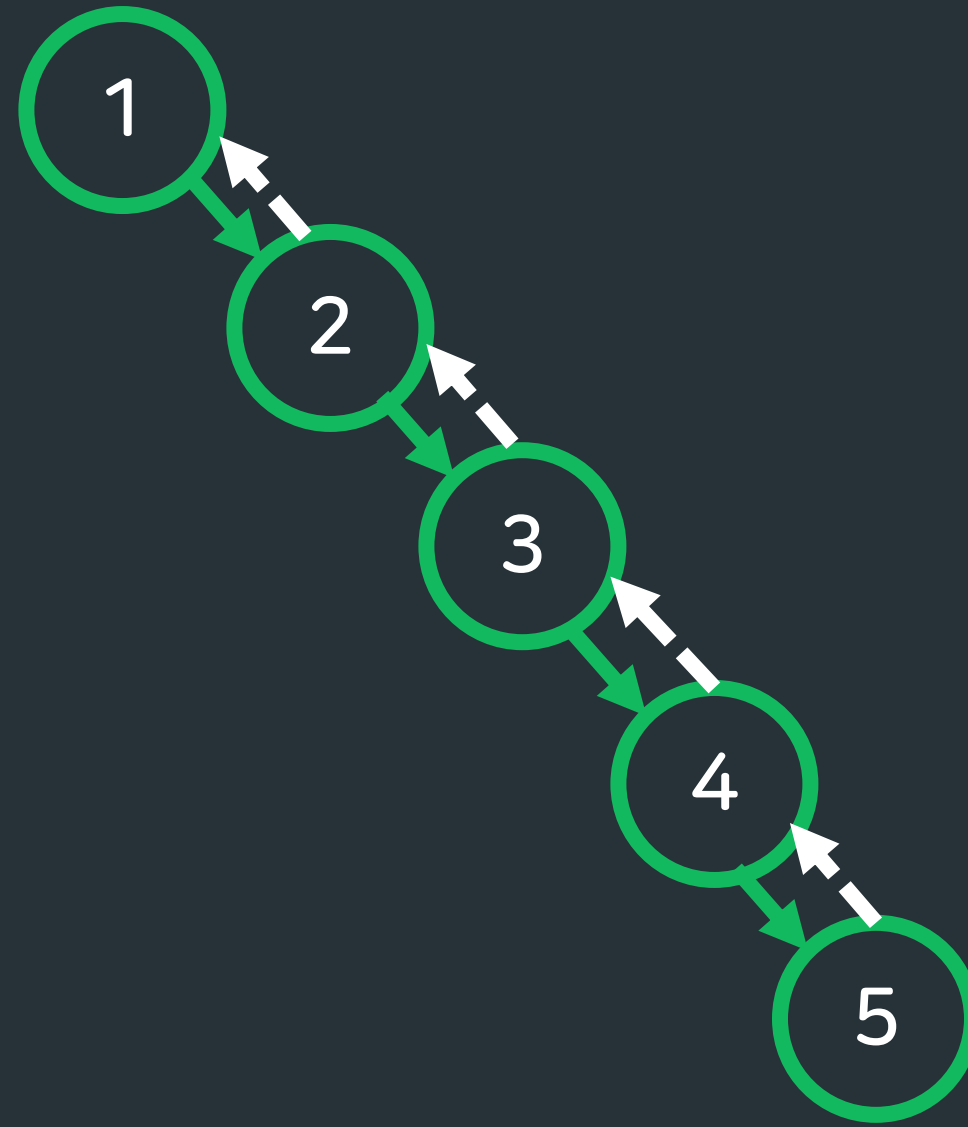
2번 정점이 속한 트리의 루트 정점은?



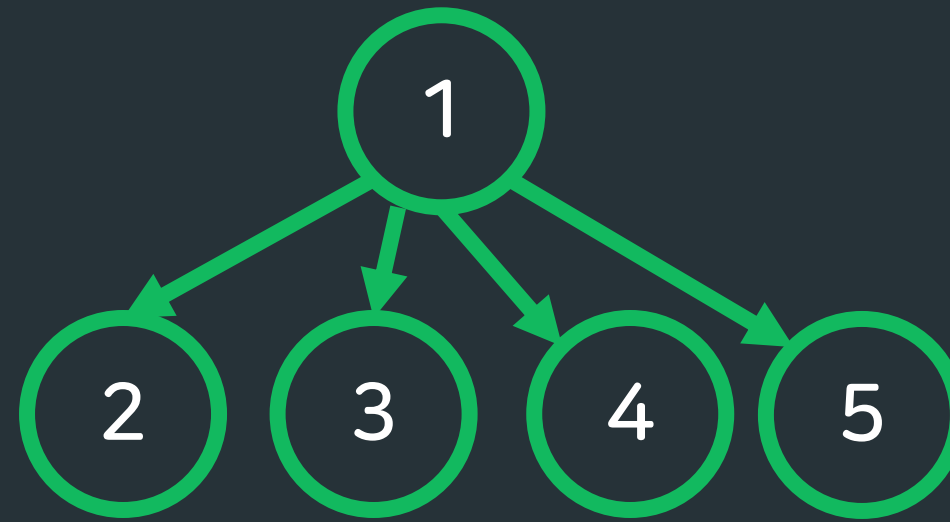
```
//초기값은 parent[x] = x 라고 가정  
int find(int node) {  
    if(node == parent[node])  
        return node;  
    return find(parent[node]);  
}
```

이대로 괜찮나?





최악의 경우 Find 연산의 시간 복잡도는  $O(n)$



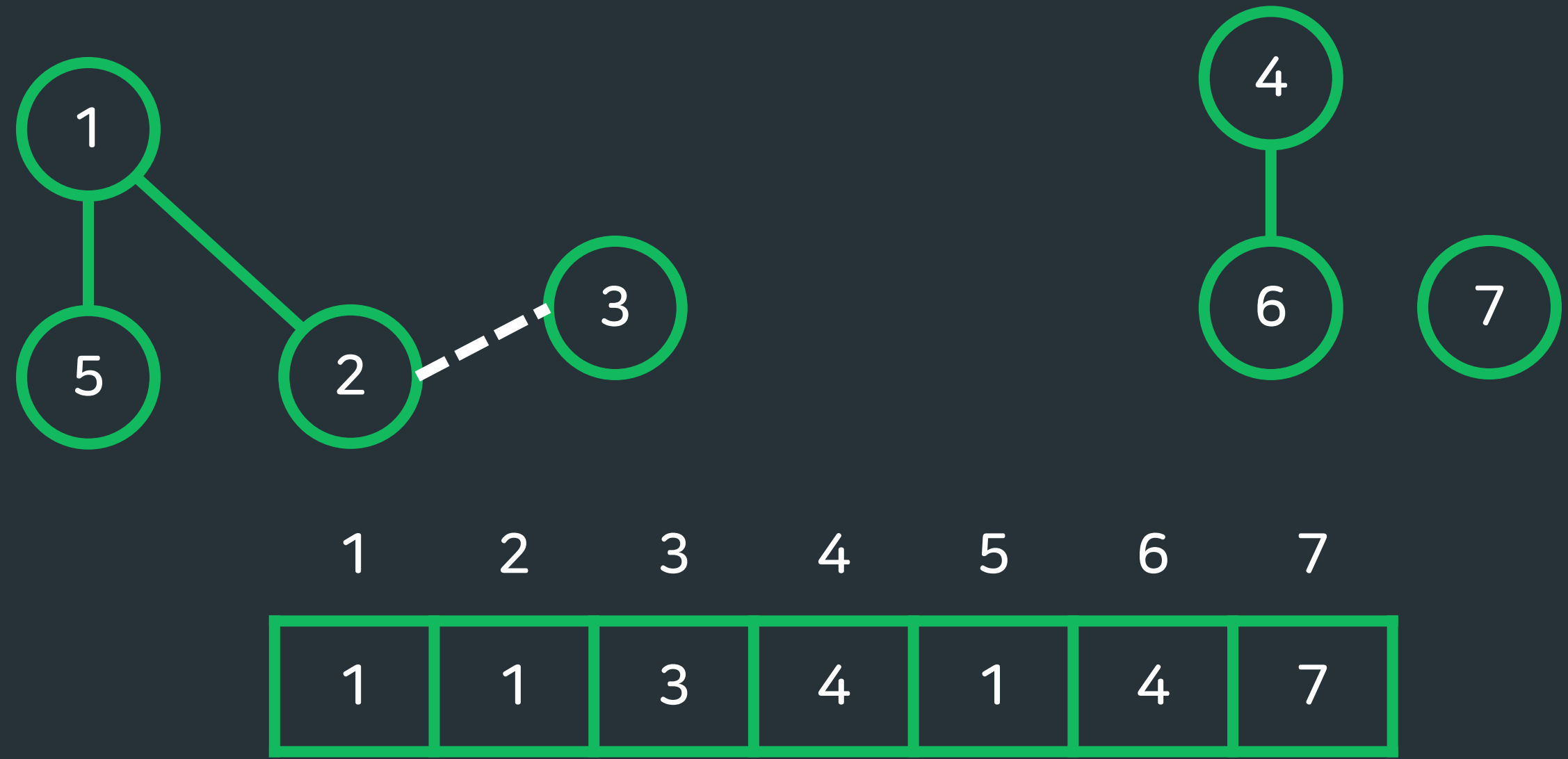
Find 연산을 하면서 각 정점의 부모를 루트 정점으로 설정  
= 그래프 압축



```
//초기값은 parent[x] = x 라고 가정
int find(int node) {
    if(node == parent[node])
        return node;
    return find(parent[node]);
}
```



```
//초기값은 parent[x] = x 라고 가정
int find(int node) {
    if(node == parent[node])
        return node;
    return parent[node] = find(parent[node]);
}
```

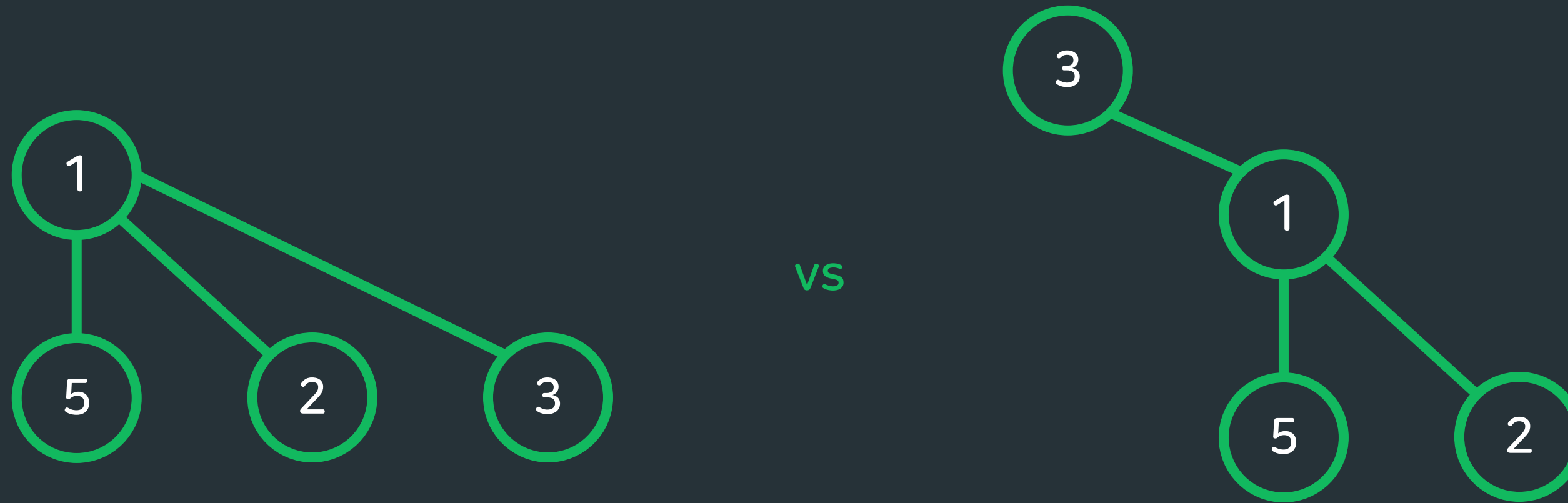


Find 연산으로 다른 집합인 것이 확인된 **2번, 3번 정점**을 Union 할 때...  
어떤 집합의 루트가 **새로운 루트**가 되어야 하나?



```
void union(int x, int y) {  
    int xp = find(x);  
    int yp = find(y);  
  
    if(xp == yp)  
        return;  
    parent[yp] = xp;  
}
```

그냥 연결해도 괜찮을까?



원래 정점이 더 많았던 집합의 루트가 새로운 집합의 루트가 되는 것이 더 효율적  
-> 각 집합의 크기를 저장해야 함



	1	2	3	4	5	6	7
Parent	1	1	3	4	1	4	7
Rank	3	1	1	2	1	1	1



```
//초기값은 rank[x] = 1 이라고 가정
void union(int x, int y) {
    int xp = find(x);
    int yp = find(y);

    if(xp == yp)
        return;
    if(rank[xp] > rank[yp]) {
        rank[xp] += rank[yp];
        parent[yp] = xp;
    }
    else {
        rank[yp] += rank[xp];
        parent[xp] = yp;
    }
}
```

부모 정점 저장 + 정점의 수 저장  
-> 메모리 낭비

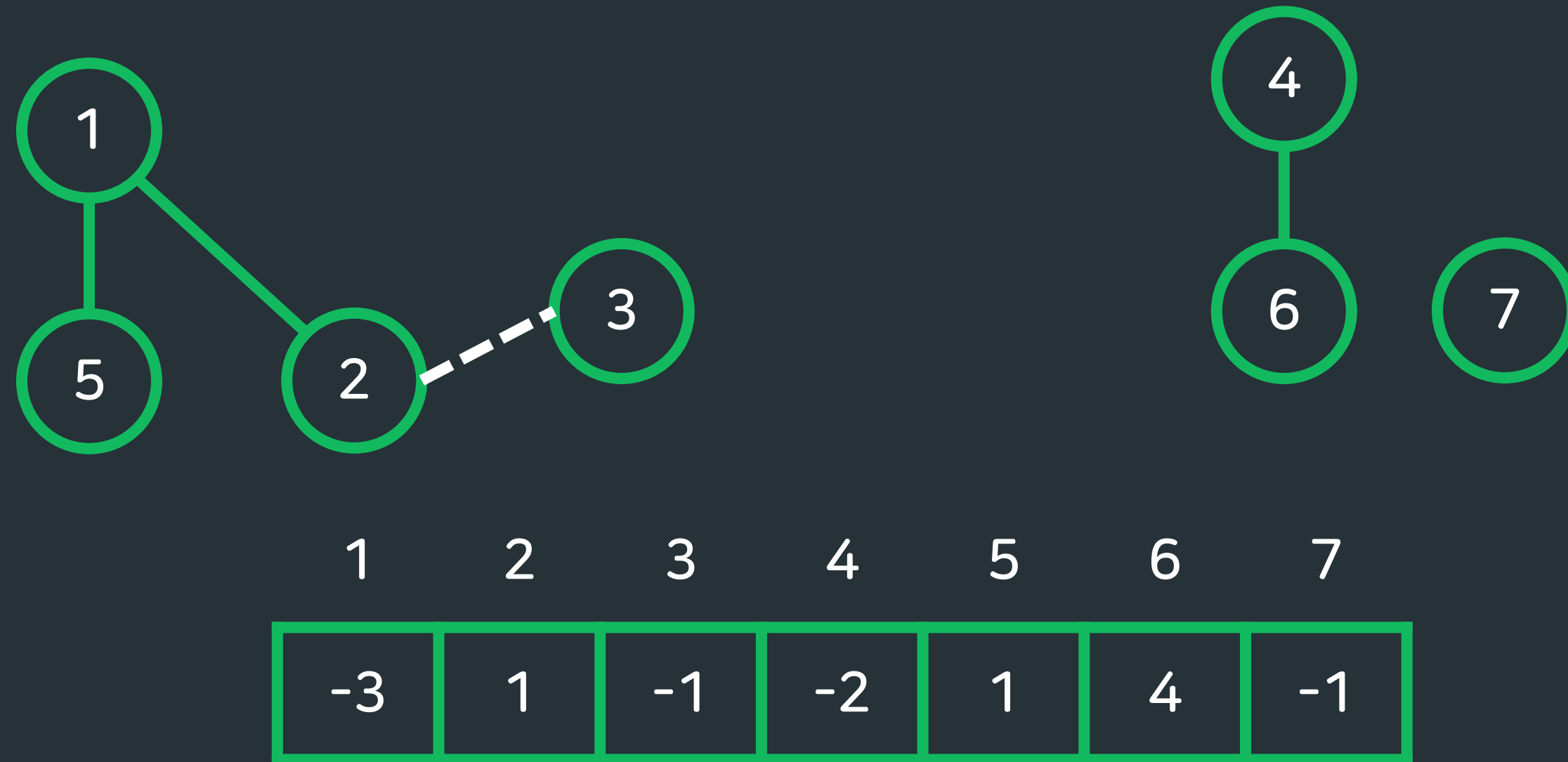


# Weighted Union Find



저장된 값이 음수 : 해당 정점이 루트 정점이며, 저장된 값의 절댓값이 집합의 크기  
저장된 값이 양수 : 저장된 값을 따라가면 해당 정점이 속한 집합의 루트를 알 수 있음

# Weighted Union Find



저장된 값이 음수 : 해당 정점이 루트 정점이며, 저장된 값의 절댓값이 집합의 크기  
저장된 값이 양수 : 저장된 값을 따라가면 해당 정점이 속한 집합의 루트를 알 수 있음

## /<> 1717번 : 집합의 표현 - Gold 4

### 문제

- $n+1$ 개의 원소에 대해, 다음의 연산을 하라
  - 두 원소가 같은 집합에 속해 있는지 확인하는 연산
  - 두 원소를 같은 집합으로 합치는 연산

### 제한 사항

- 원소의 개수  $n$ 은  $1 \leq n \leq 1,000,000$
- 연산의 개수  $m$ 은  $1 \leq m \leq 100,000$

## 예제 입력 1

```
7 8
0 1 3
1 1 7
0 7 6
1 7 1
0 3 7
0 4 2
0 1 1
1 1 1
```

## 예제 출력 1

```
NO
NO
YES
```

## /<> 4803번 : 트리 - Gold 4

### 문제

- 그래프가 주어질 때, **트리의 개수**를 출력하라

### 제한 사항

- 정점의 개수  $n$ 은  $0 \leq n \leq 500$
- 간선의 개수  $m$ 은  $0 \leq m \leq n(n-1)/2$
- 입력은 **무방향 그래프**

## 예제 입력 1

```
6 3
1 2 2 3 3 4
6 5
1 2 2 3 3 4 4 5 5 6
6 6
1 2 2 3 1 3 4 5 5 6 6 4
0 0
```

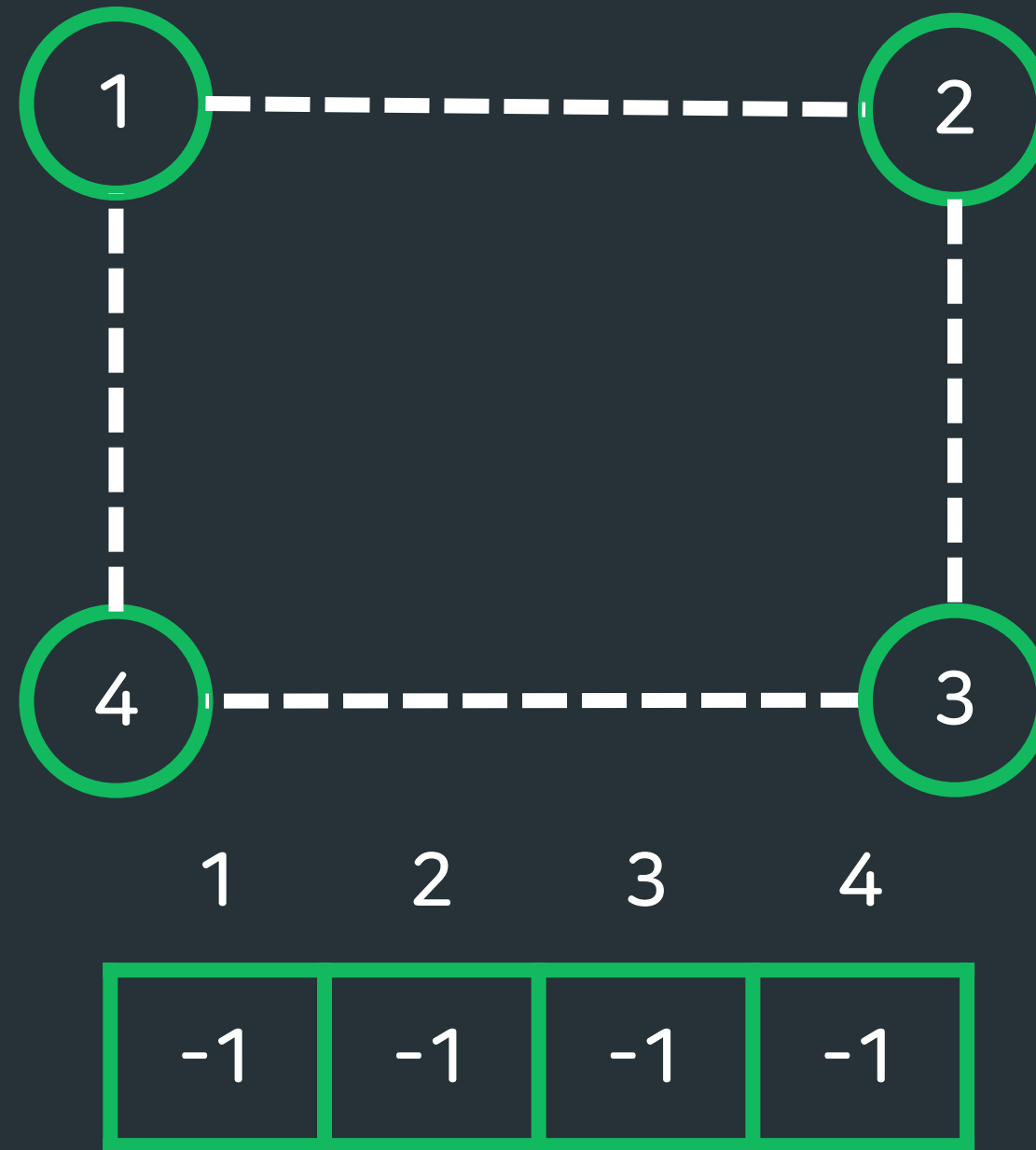
## 예제 출력 1

Case 1: A forest of 3 trees.  
Case 2: There is one tree.  
Case 3: No trees.

## Hint

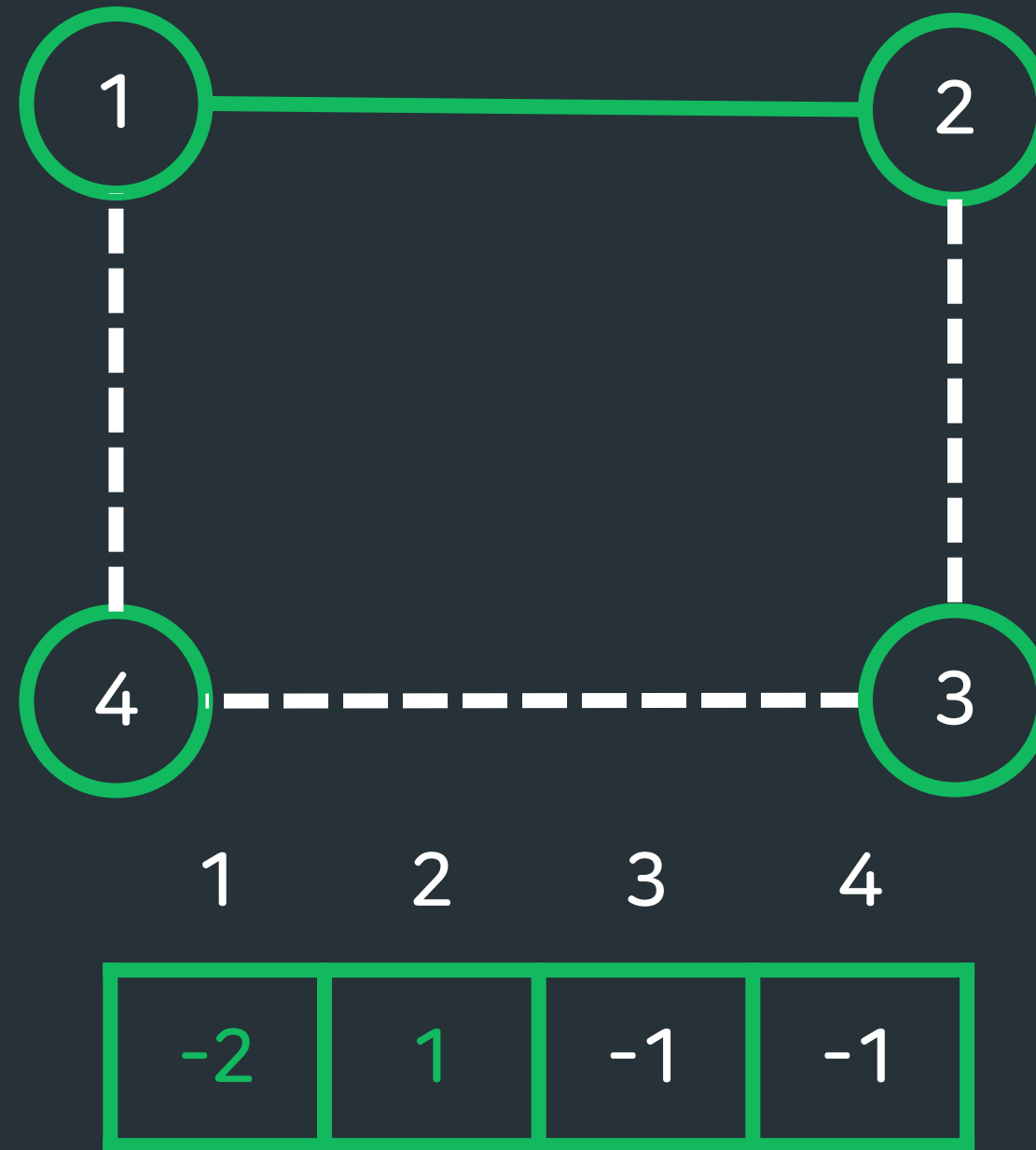
1. 유니온 파인드에서 각각의 집합을 **트리**로 표현했었어요!
2. **사이클**이 발생하는 순간은 언제일까요?

# 사이클이 생성되는 순간

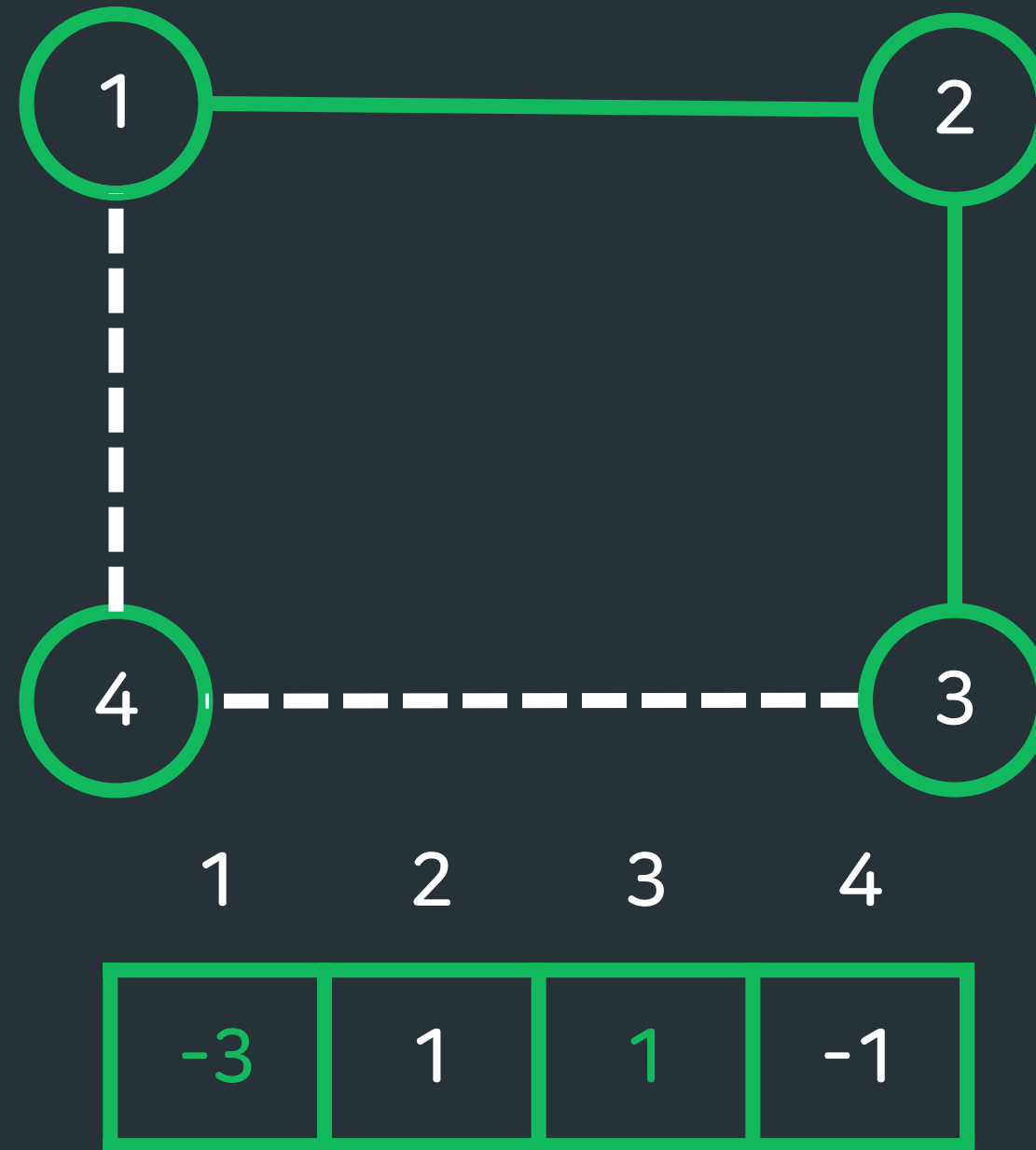




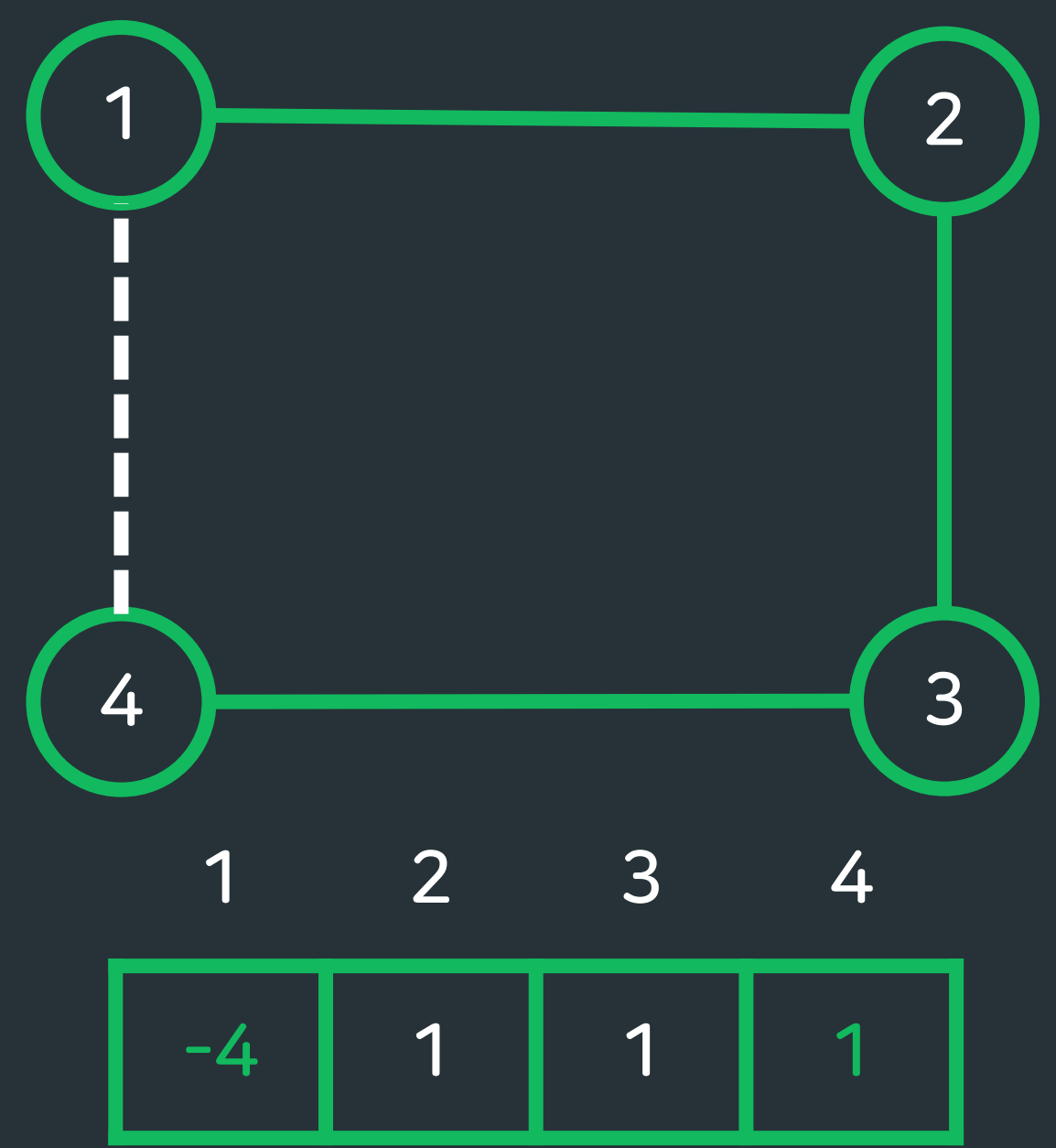
# 사이클이 생성되는 순간



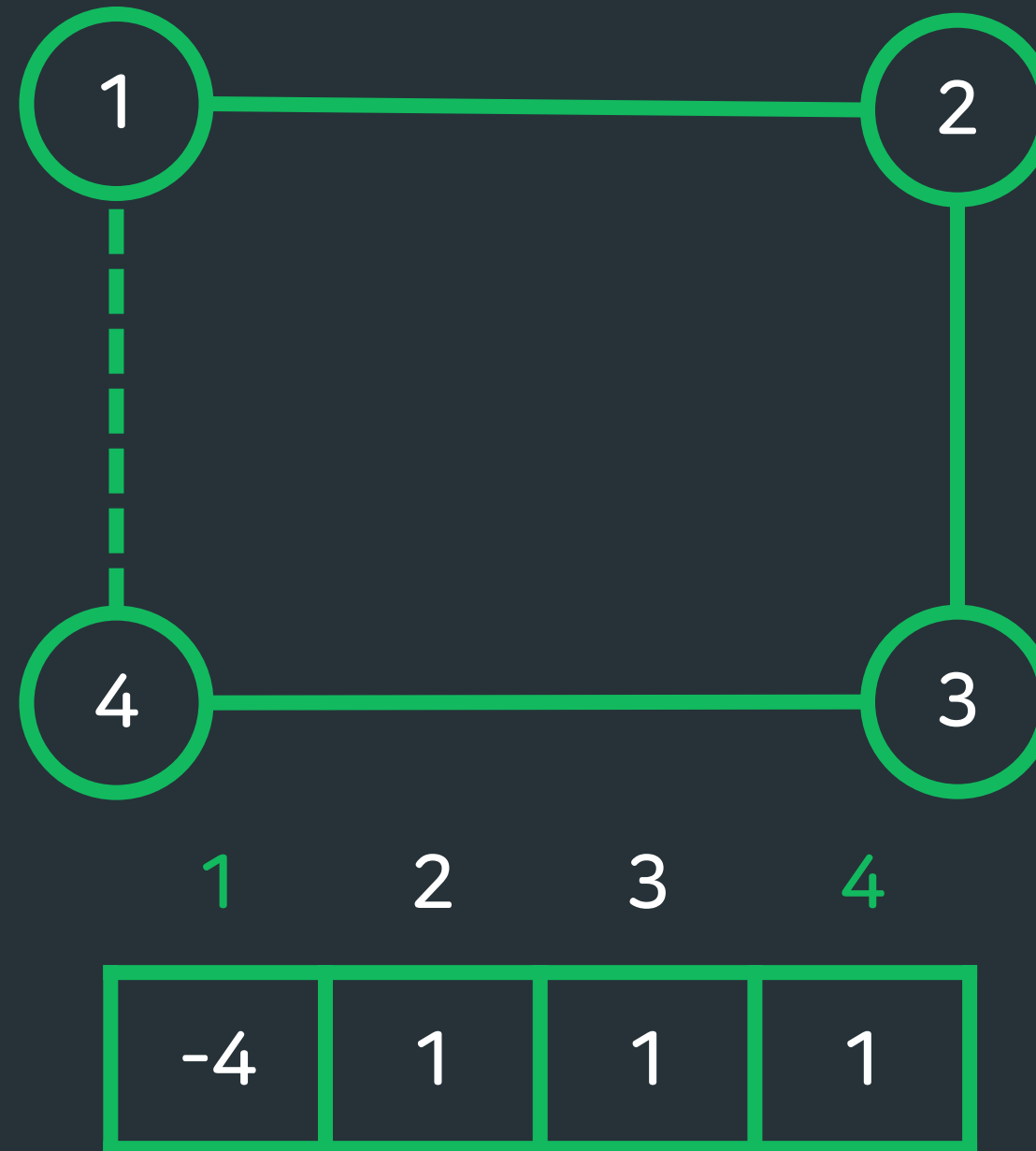
# 사이클이 생성되는 순간



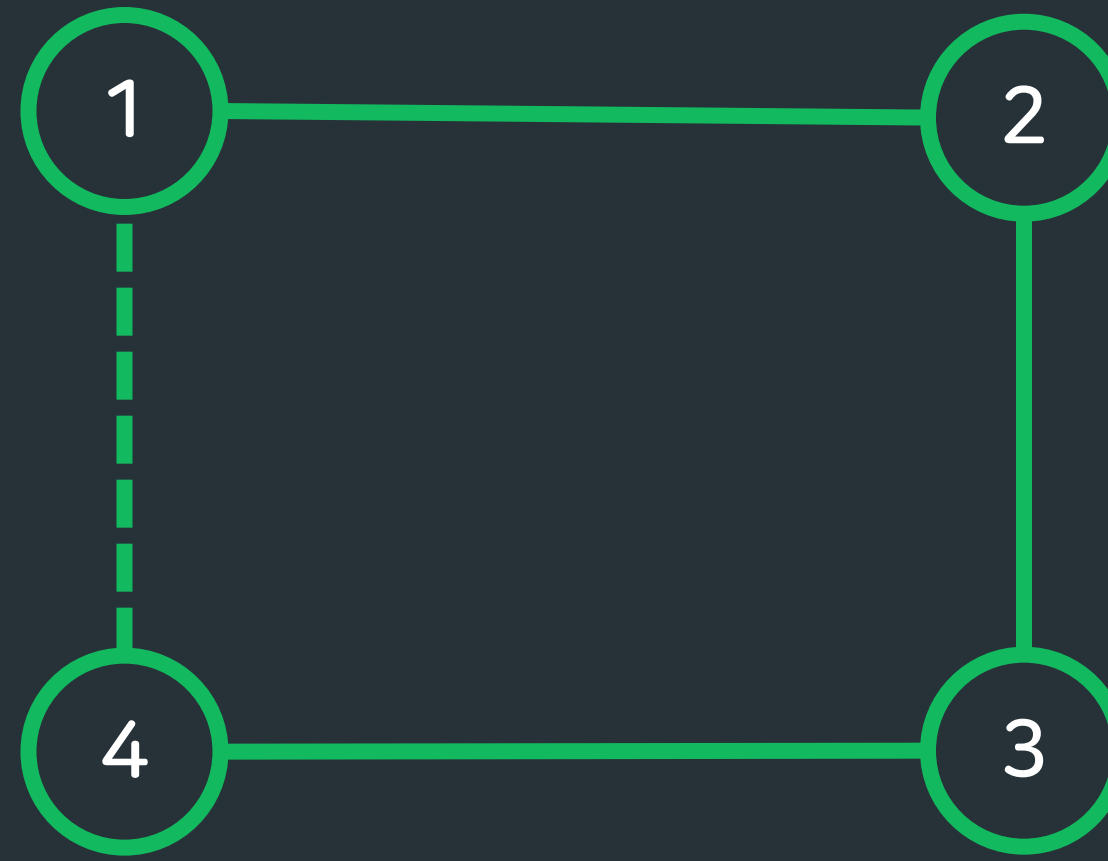
# 사이클이 생성되는 순간



# 사이클이 생성되는 순간



## 사이클이 생성되는 순간



이미 같은 집합에 속한 두 정점을 잇는 순간 **사이클** 발생!

## /<> 16562번 : 친구비 - Gold 3

### 문제

- 친구를 사귀기 위해선 친구비가 필요하다.
- 친구의 친구는 친구다.
- 가장 적은 비용으로 모든 사람과 친구가 되는 방법을 출력하라

### 제한 사항

- 학생의 수  $N$ 은  $1 \leq N \leq 10,000$
- 친구 관계 수  $M$ 은  $0 \leq M \leq 10,000$
- 가지고 있는 돈  $k$ 는  $1 \leq k \leq 10,000,000$
- 각 학생이 원하는 친구비  $A_i$ 는  $1 \leq A_i \leq 10,000$

## 예제 입력 1

```
5 3 20
10 10 20 20 30
1 3
2 4
5 4
```

## 예제 출력 1

20

## 예제 입력 2

```
5 3 10
10 10 20 20 30
1 3
2 4
5 4
```

## 예제 출력 2

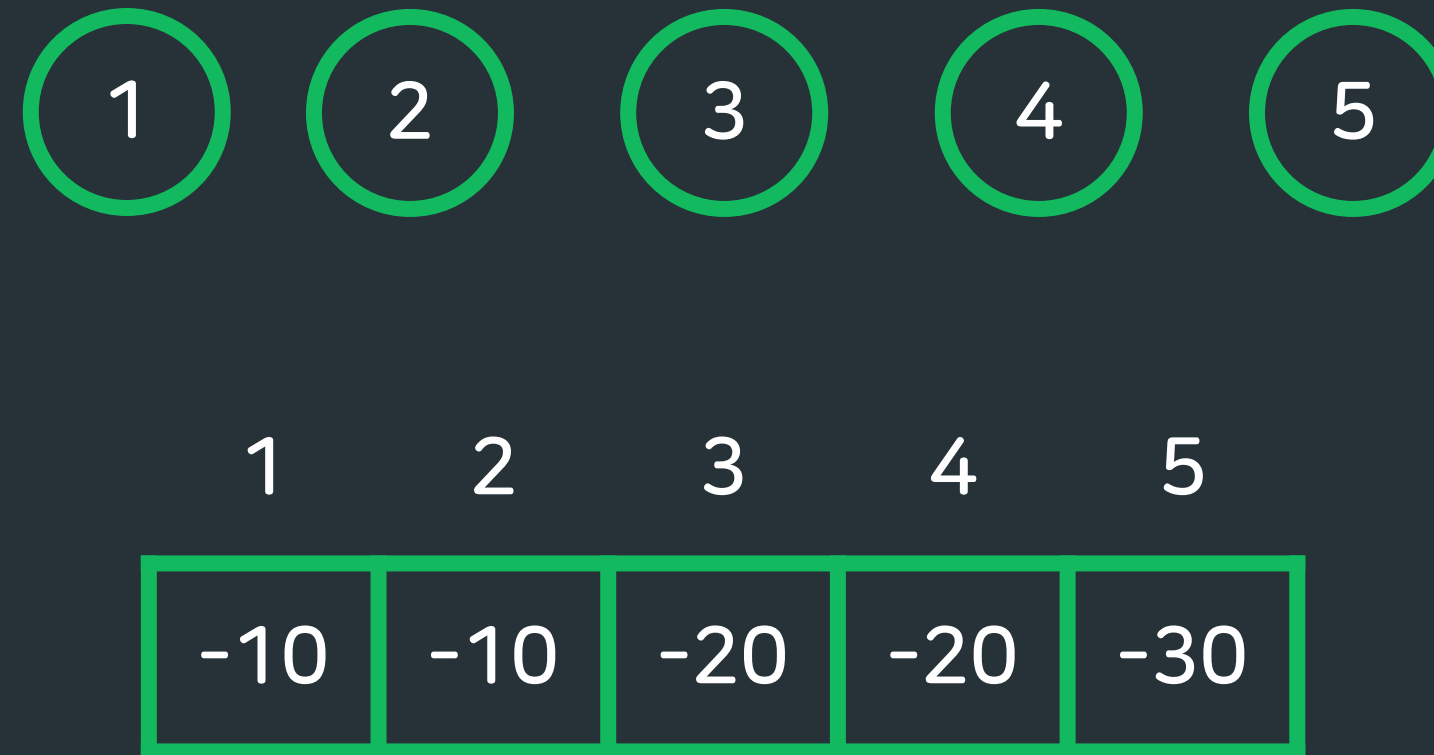
Oh no

## Hint

1. 친구들을 연결 관계로 표현할 수 있겠네요.
2. Weighted Union Find에서 정점이 많은 집합의 루트가 새로운 루트가 됐어요.  
이번엔 어떻게 될까요?

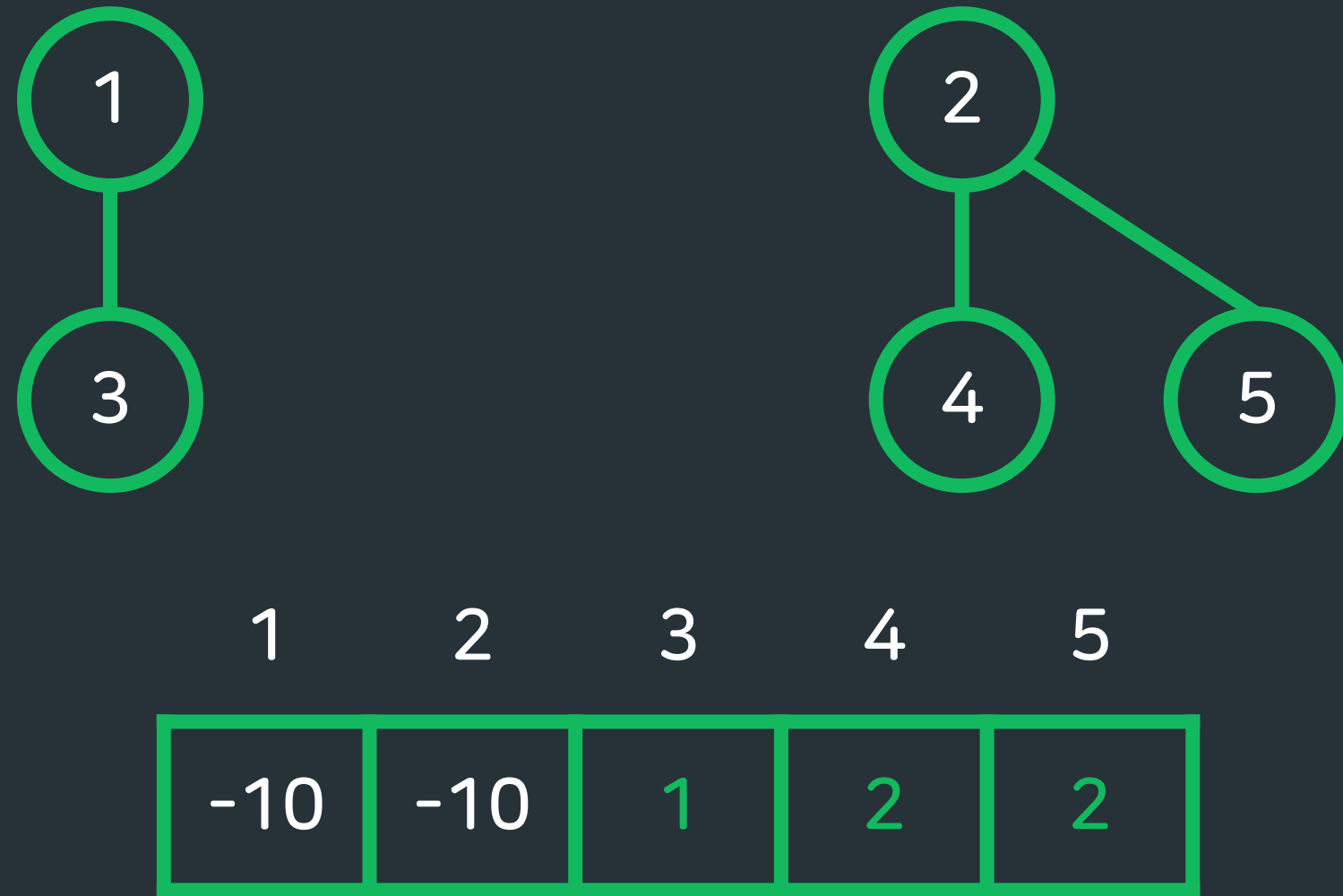


# Weighted?



집합에 속한 정점의 수가 많은 루트를 선택하는 것이 아니라  
친구비가 더 작은 루트를 선택!

# Weighted?



집합에 속한 정점의 수가 많은 루트를 선택하는 것이 아니라  
친구비가 더 작은 루트를 선택!

## 정리

- 트리를 활용해 각 집합에 속한 원소를 빠르게 파악하는 유니온 파인드 알고리즘
- 각 집합의 루트 정점을 찾아 속해 있는 집합을 판별하는 Find 연산
- 서로 다른 두 집합을 하나의 집합으로 합치는 Union 연산
- 처음의 비효율적인 방법에서 Weighted Union Find까지 발전하는 과정을 잘 이해하기


## 이것도 알아보세요

- 유니온 파인드 알고리즘은 트리의 사이클을 판별하기에 좋은 알고리즘입니다.  
최소 신장 트리(MST) 알고리즘 중에 유니온 파인드를 활용하는 알고리즘이 있는데, 예습하시면 좋아요!

## 필수

- /<> 16236번 : 아기 상어 - Gold 4
- /<> 18111번 : 마인크래프트 - Silver 3

## 3문제 이상 선택

- /<> 1043번 : 거짓말 - Gold 4
- /<> 1976번 : 여행 가자 - Gold 4
- /<> 10775번 : 공항 - Gold 2
- /<> 16724번 : 피리 부는 사나이 - Gold 2
- /<> 20040번 : 사이클 게임 - Gold 4
-  연습문제 : 가장 큰 정사각형 찾기 - Level 2