Data Structure

이종서(leejseo)

이 강의의 범위

- 자료구조들을 Black-box 형태로 가져다 쓸 수 있게 자료구조들이 지원하는 연산을 다룹니다.

- 일부 자료구조를 제외하고는 구현 방법에 대해서는 다루지 않습니다. (대다수 언어의 내장 라이브러리로 이미 구현이 되어 있기 때문)

Heap

- Max-heap, Min-heap 등이 존재 (Max-heap 기준으로 설명)

- Top(): Heap의 원소들 가운데 최댓값을 반환
- Push(x): Heap에 새로운 원소를 추가
- Pop(): Heap에서 최댓값을 제거

Heap

- Max-heap, Min-heap 등이 존재 (Max-heap 기준으로 설명)

- Top(): Heap의 원소들 가운데 최댓값을 반환 – O(1)
- Push(x): Heap에 새로운 원소를 추가 – O(log n)
- Pop(): Heap에서 최댓값을 제거 – O(log n)

Heap

- Max-heap, Min-heap 등이 존재 (Max-heap 기준으로 설명)

- Top(): Heap의 원소들 가운데 최댓값을 반환 – O(1)
- Push(x): Heap에 새로운 원소를 추가 – O(log n)
- Pop(): Heap에서 최댓값을 제거 – O(log n)

- std::priority_queue(C++), heapq(Python)

Set

- add(x): 원소 x를 추가
- remove(x): 원소 x를 제거
- contains(x): x를 원소로 가지고 있는지 판별
- 원소들을 크기 순으로 순회 가능

Set

- add(x): 원소 x를 추가 – O(log n)
- remove(x): 원소 x를 제거 – O(log n)
- contains(x): x를 원소로 가지고 있는지 판별 – O(log n)
- 원소들을 크기 순으로 순회 가능

Set

- add(x): 원소 x를 추가 – O(log n)
- remove(x): 원소 x를 제거 – O(log n)
- contains(x): x를 원소로 가지고 있는지 판별 – O(log n)
- 원소들을 크기 순으로 순회 가능

- std::set(C++)

Hash Set

- add(x): 원소 x를 추가 – O(1)
- remove(x): 원소 x를 제거 – O(1)
- contains(x): x를 원소로 가지고 있는지 판별 – O(1)

Hash Set

- add(x): 원소 x를 추가 – O(1)
- remove(x): 원소 x를 제거 – O(1)
- contains(x): x를 원소로 가지고 있는지 판별 – O(1)

- std::unordered_set(C++), set(Python)

Map

- set(k, v): key k에 해당하는 value를 v로 설정
- get(k): key k에 해당하는 value를 반환
- remove(k): key k를 삭제
- key의 크기 순으로 순회 가능

Map

- set(k, v): key k에 해당하는 value를 v로 설정 – O(log n)
- get(k): key k에 해당하는 value를 반환 – O(log n)
- remove(k): key k를 삭제 – O(log n)
- key의 크기 순으로 순회 가능

Map

- set(k, v): key k에 해당하는 value를 v로 설정 – O(log n)
- get(k): key k에 해당하는 value를 반환 – O(log n)
- remove(k): key k를 삭제 – O(log n)
- key의 크기 순으로 순회 가능

- std::map(C++)

Map

- set(k, v): key k에 해당하는 value를 v로 설정 – O(1)
- get(k): key k에 해당하는 value를 반환 – O(1)
- remove(k): key k를 삭제 – O(1)

Hash Map

- set(k, v): key k에 해당하는 value를 v로 설정 – O(1)
- get(k): key k에 해당하는 value를 반환 – O(1)
- remove(k): key k를 삭제 – O(1)

- std::unordered_map(C++), dict(Python)

Union-Find

N개의 집합 {1}, {2}, ..., {N}이 있다.

- union(x, y): x가 속한 집합과 y가 속한 집합을 하나로 합친다.
- find(x): x가 어느 집합에 속해있는지 반환

Union-Find

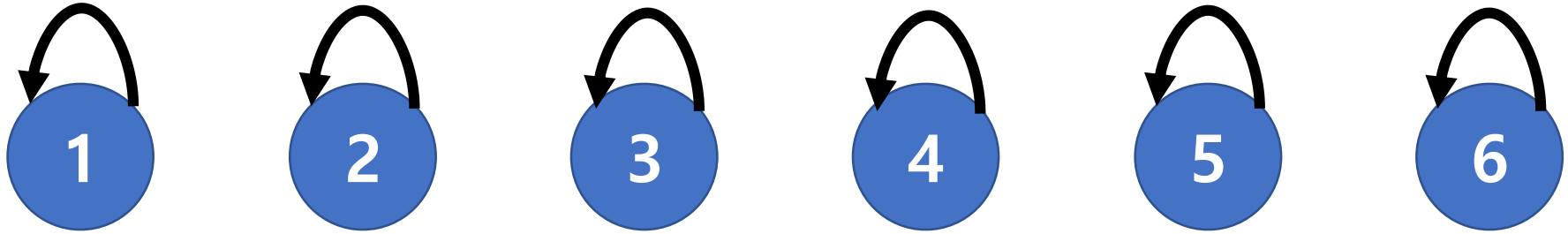
N개의 집합 {1}, {2}, ..., {N}이 있다.

- union(x, y): x가 속한 집합과 y가 속한 집합을 하나로 합친다.
- find(x): x가 어느 집합에 속해있는지 반환

Union-Find 자료구조는 위의 두 가지 연산을 효율적으로 해주는 자료구조이다.

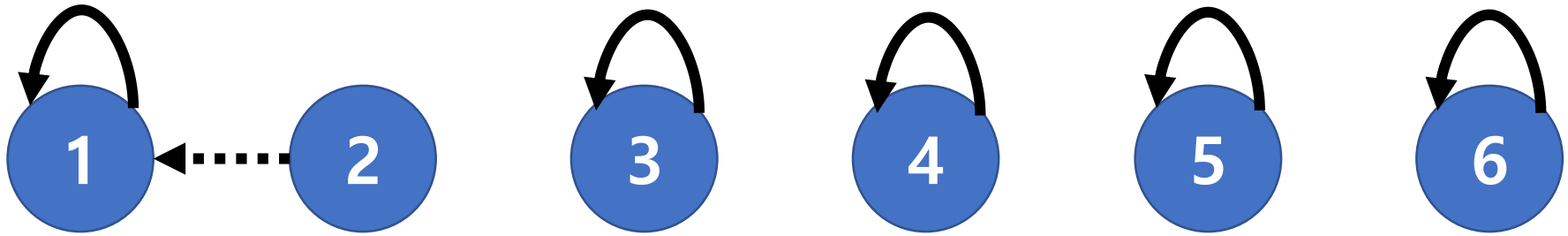
Union-Find

- N개의 집합을 N개의 정점이 있는 트리로 나타낼 것이다. 하나의 트리는 하나의 집합에 대응된다.



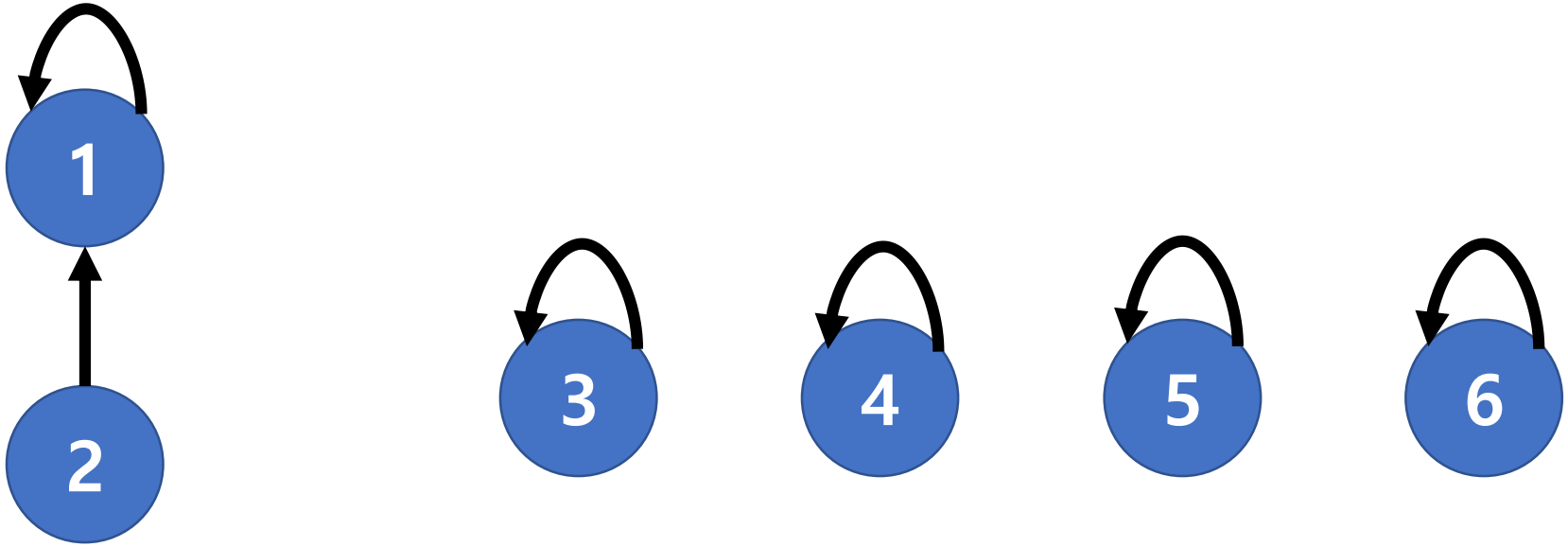
Union-Find

- union(1, 2): 1번 정점이 속한 집합과 2번 정점이 속한 집합을 합친다.



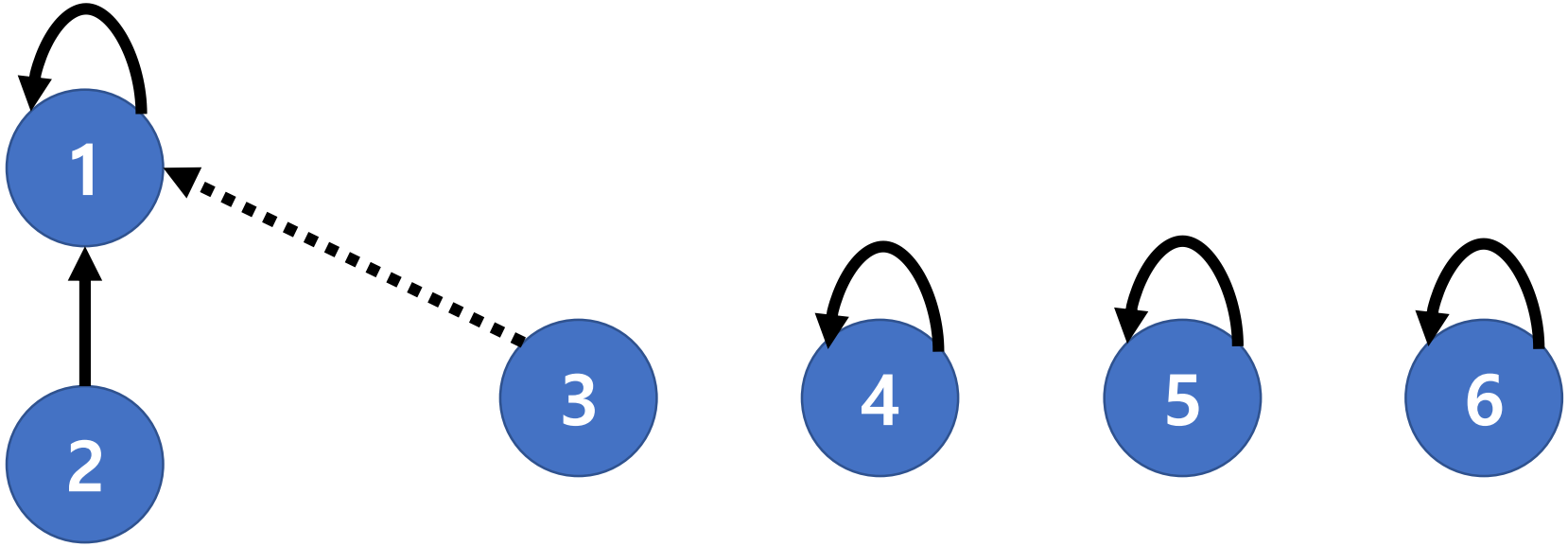
Union-Find

- union(1, 2): 1번 정점이 속한 집합과 2번 정점이 속한 집합을 합친다.



Union-Find

- union(2, 3): 2번 정점이 속한 집합과 3번 정점이 속한 집합을 합친다.



Union-Find

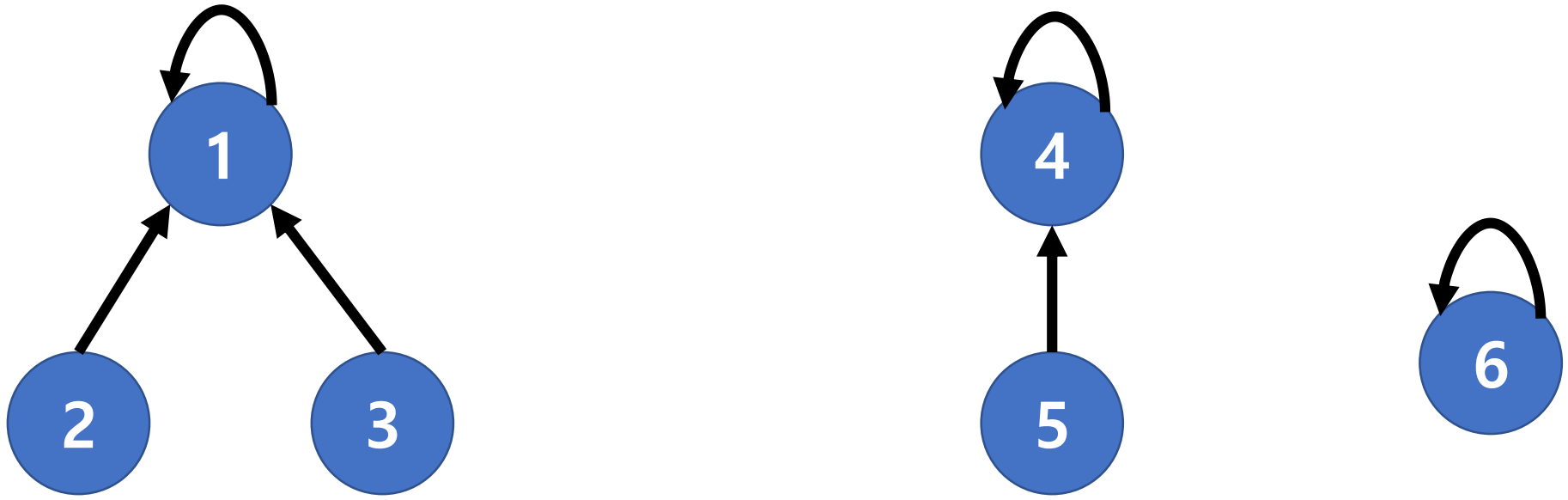
- union(2, 3): 2번 정점이 속한 집합과 3번 정점이 속한 집합을 합친다.

Union-Find

- union(4, 5): 4번 정점이 있는 집합과 5번 정점이 있는 집합을 합친다.

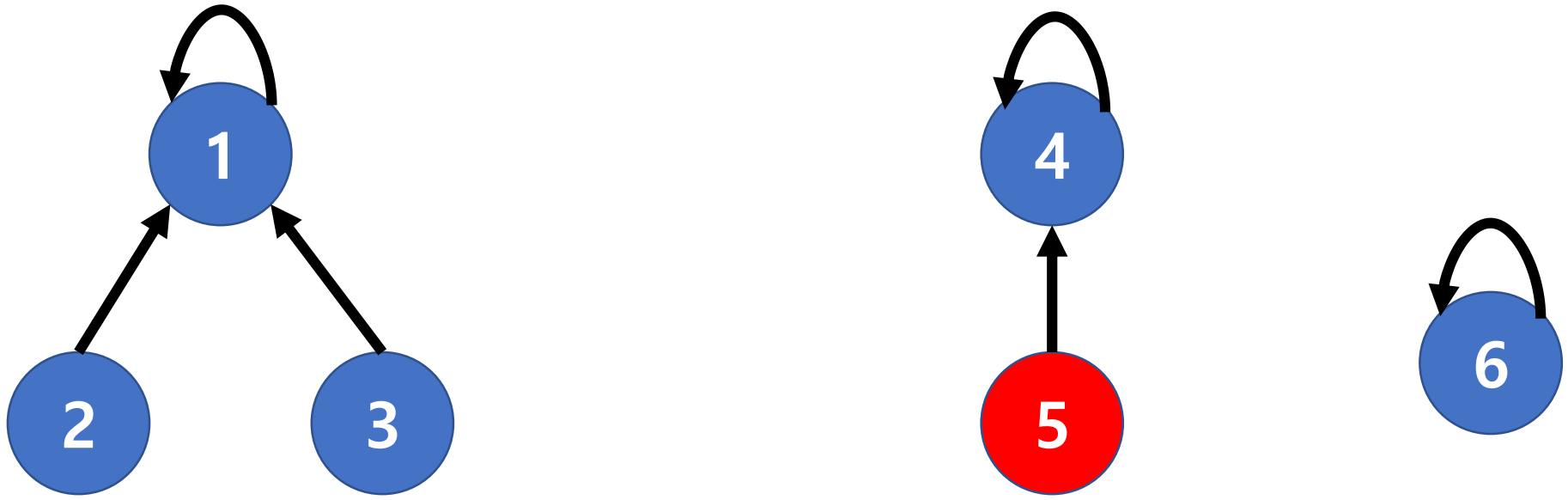
Union-Find

- union(4, 5): 4번 정점이 있는 집합과 5번 정점이 있는 집합을 합친다.



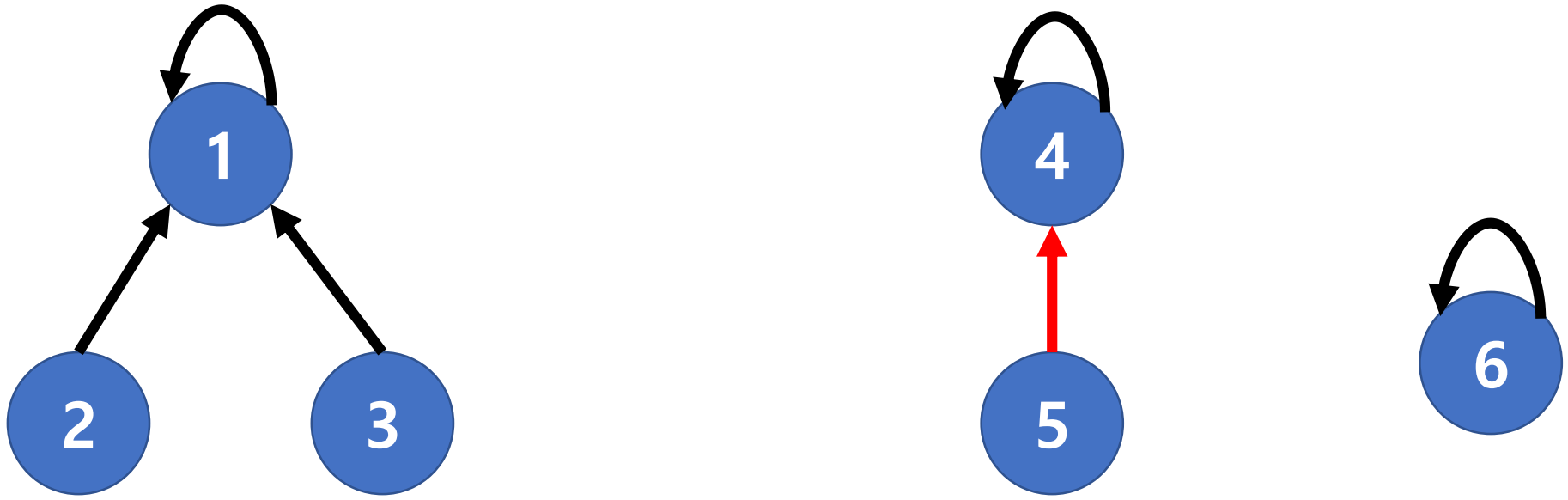
Union-Find

- find(5): 5가 속해있는 집합의 루트인 4를 반환한다.



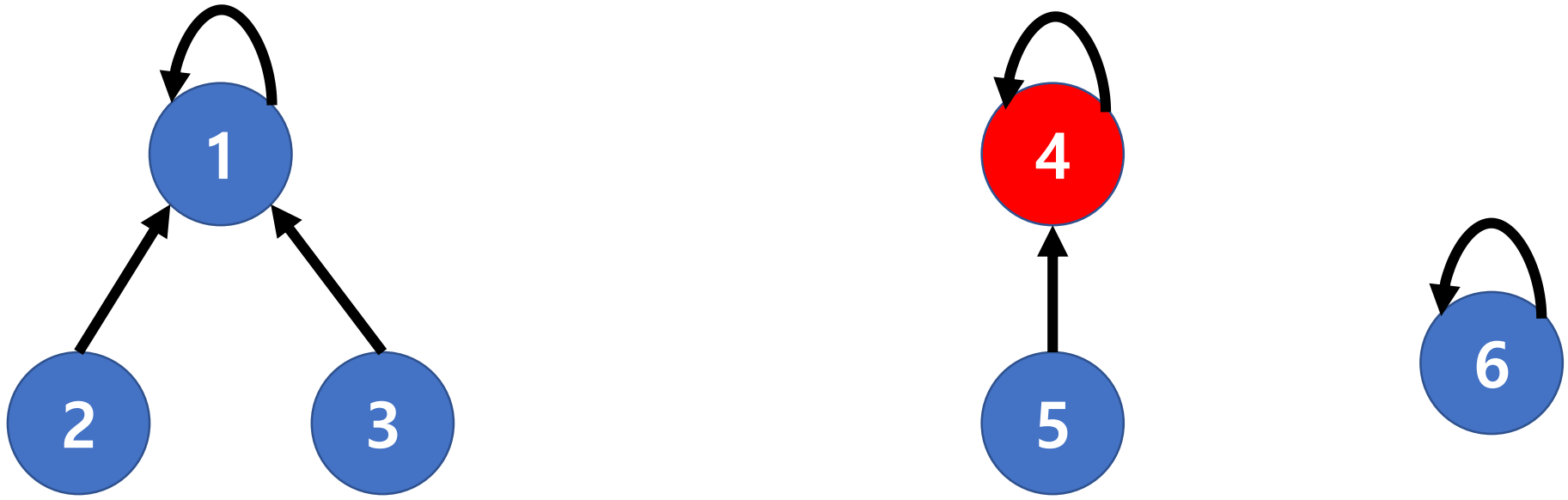
Union-Find

- find(5): 5가 속해있는 집합의 루트인 4를 반환한다.



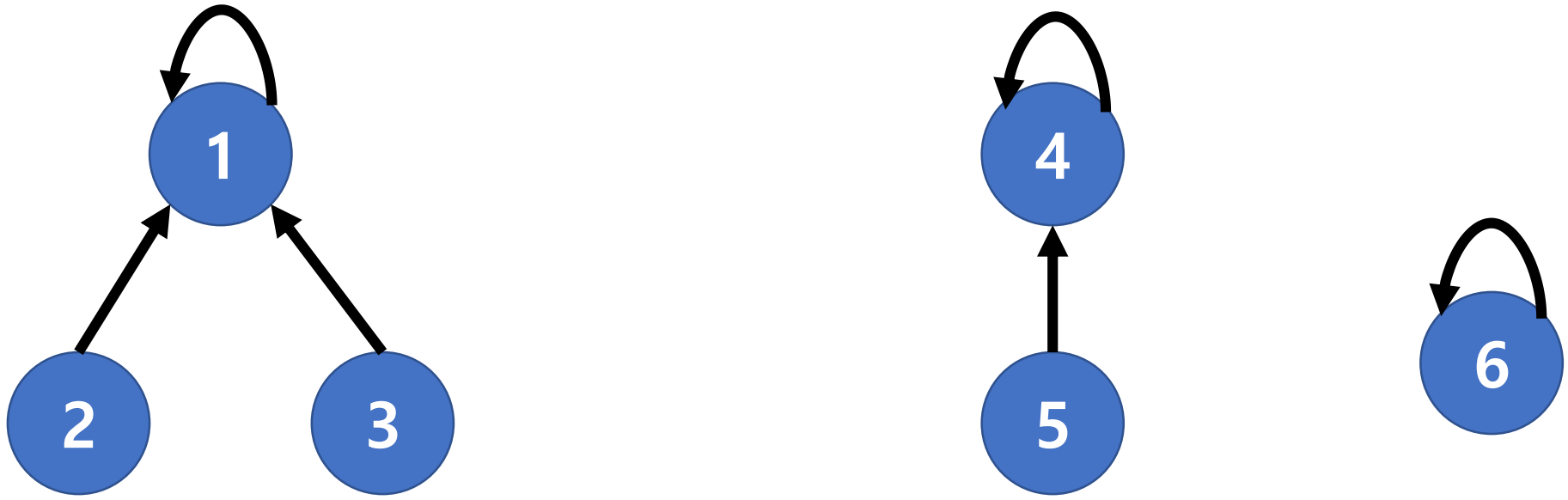
Union-Find

- find(5): 5가 속해있는 집합의 루트인 4를 반환한다.



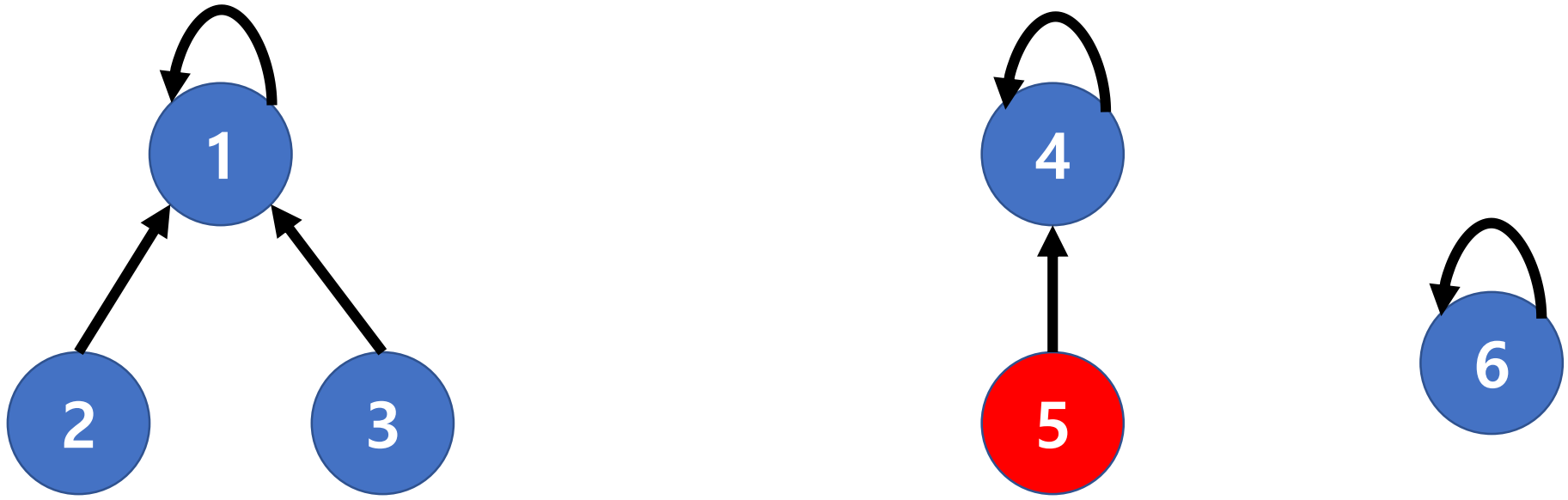
Union-Find

- union(3, 5): 3번 정점이 속한 집합과 5번 정점이 속한 집합을 합친다.



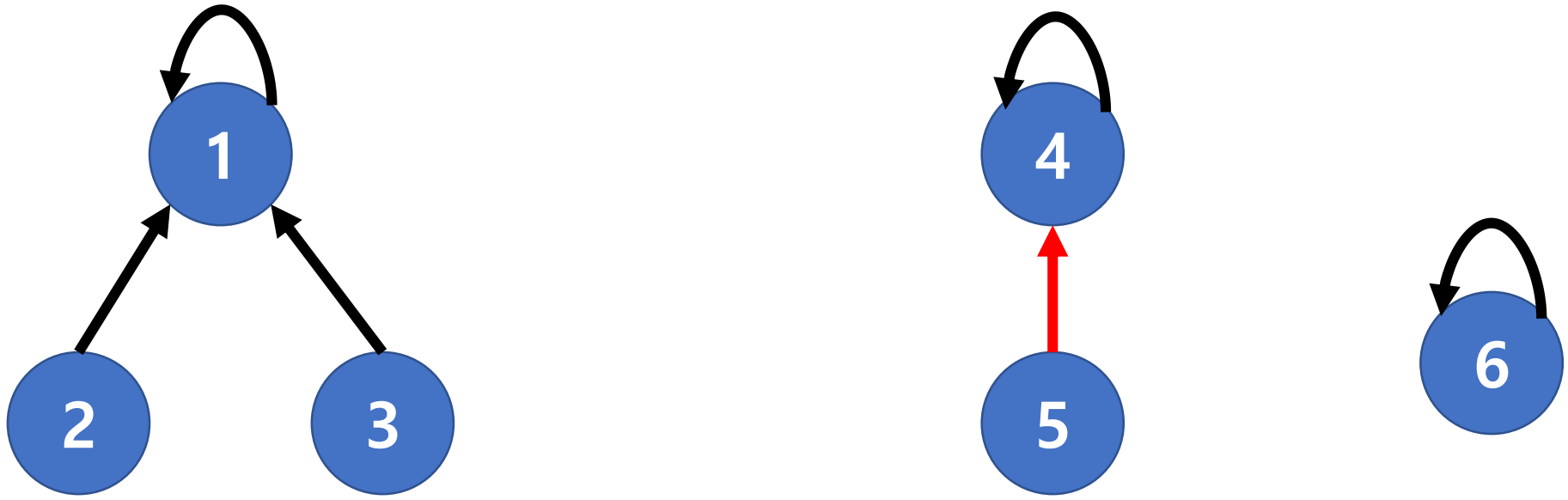
Union-Find

- union(3, 5): 3번 정점이 속한 집합과 5번 정점이 속한 집합을 합친다.



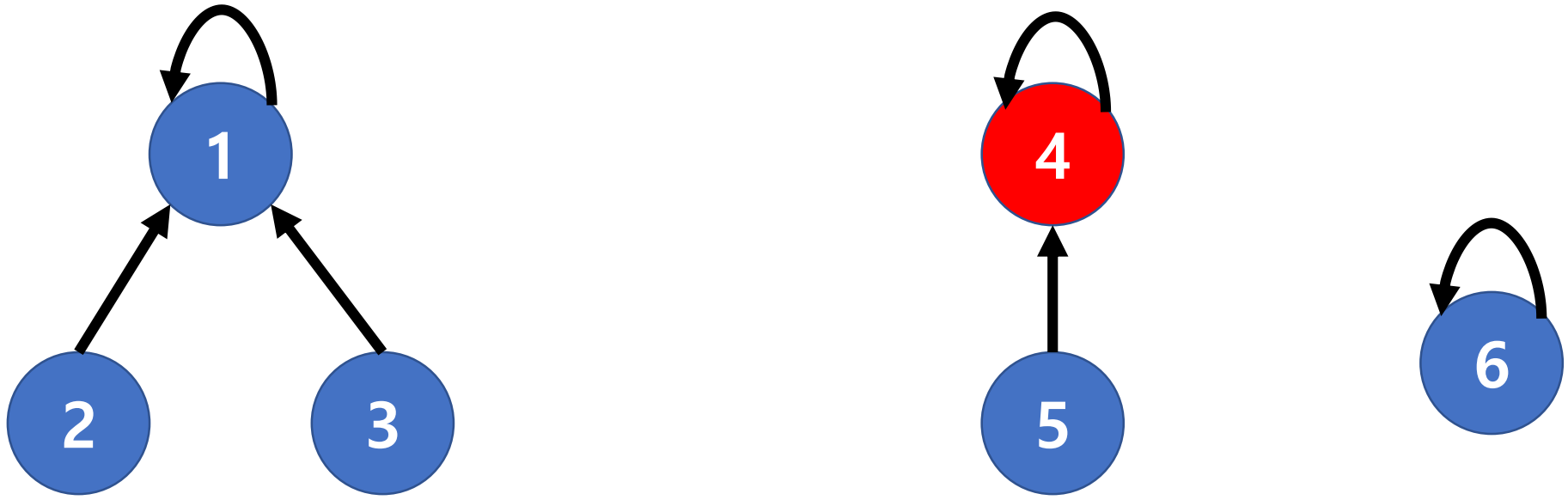
Union-Find

- union(3, 5): 3번 정점이 속한 집합과 5번 정점이 속한 집합을 합친다.



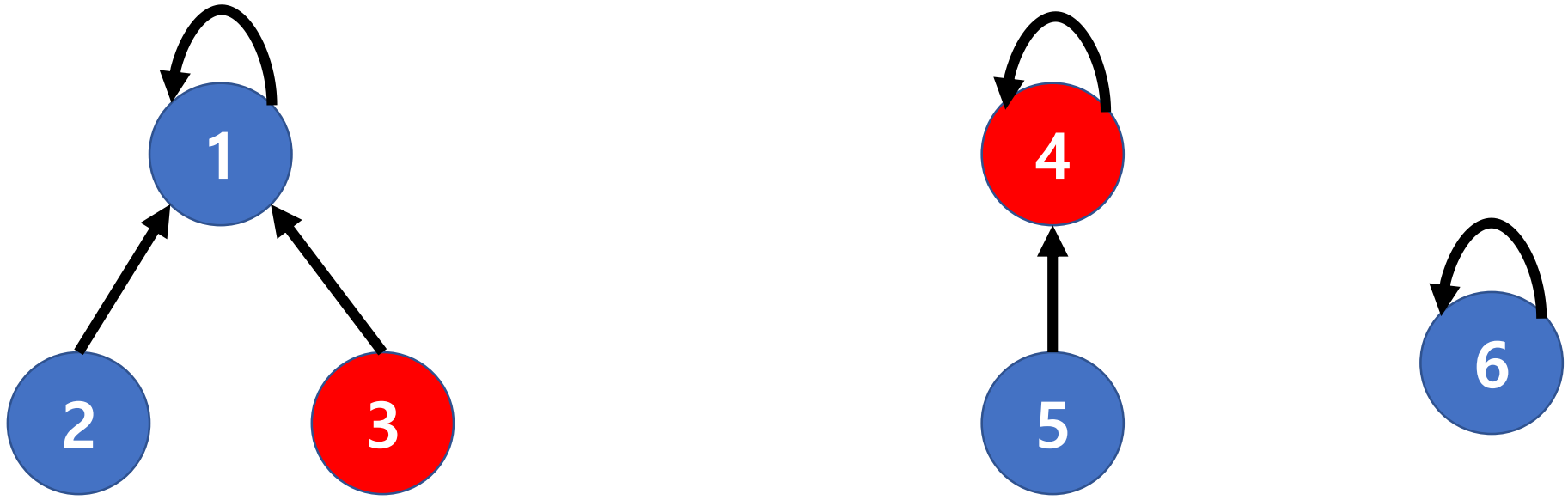
Union-Find

- union(3, 5): 3번 정점이 속한 집합과 5번 정점이 속한 집합을 합친다.



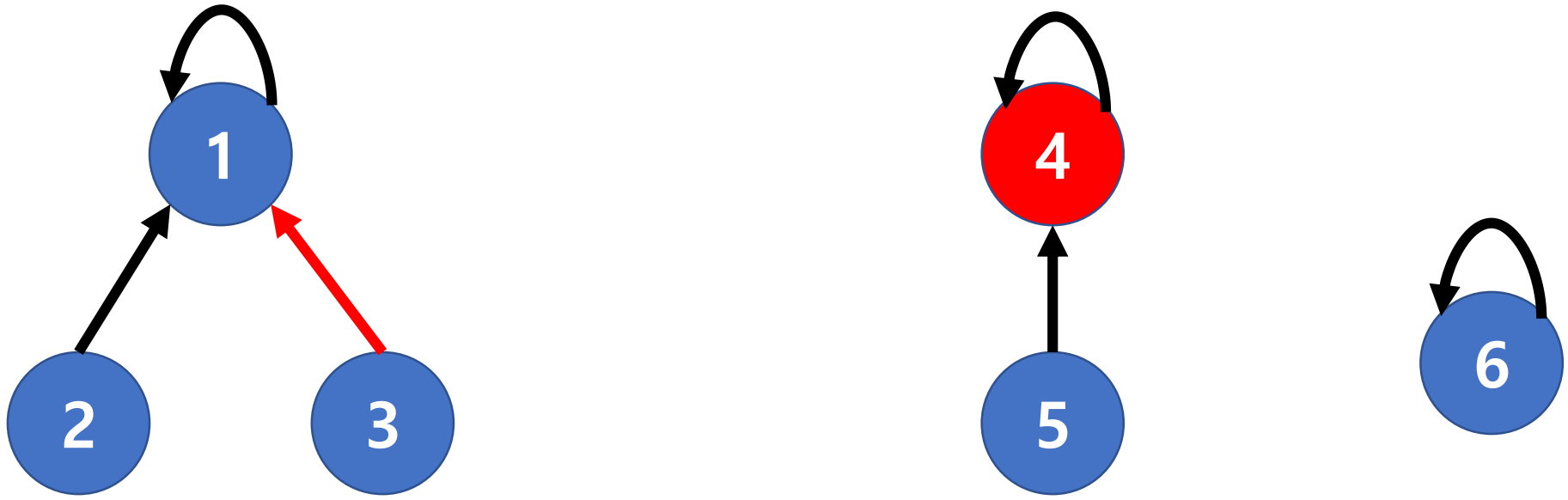
Union-Find

- union(3, 5): 3번 정점이 속한 집합과 5번 정점이 속한 집합을 합친다.



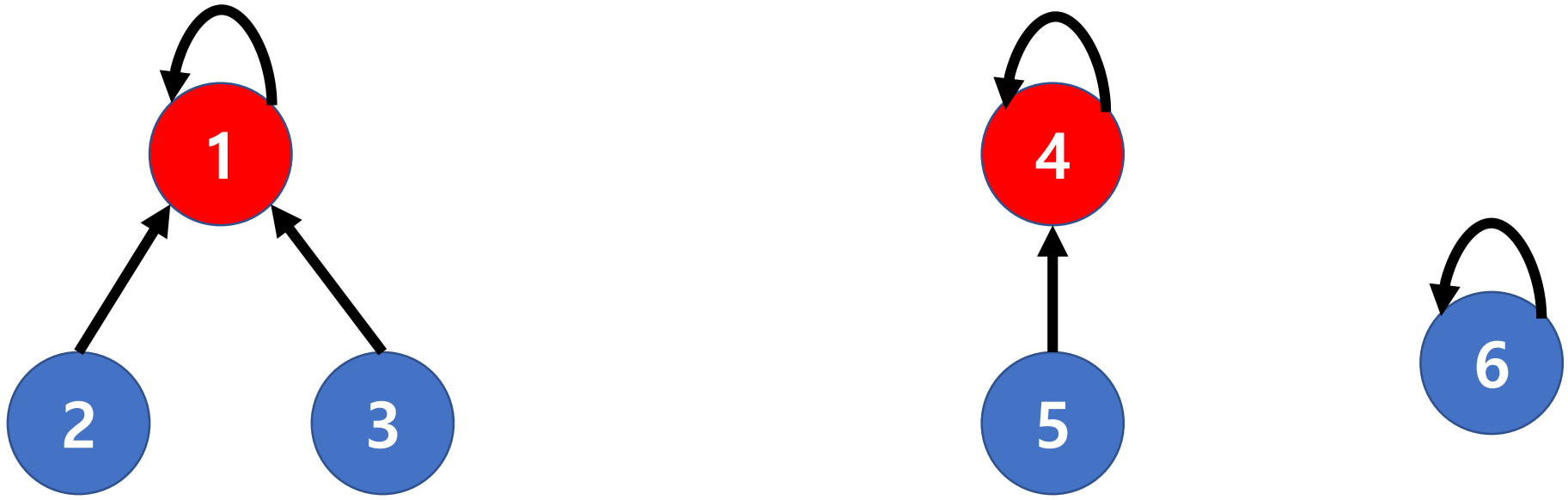
Union-Find

- union(3, 5): 3번 정점이 속한 집합과 5번 정점이 속한 집합을 합친다.



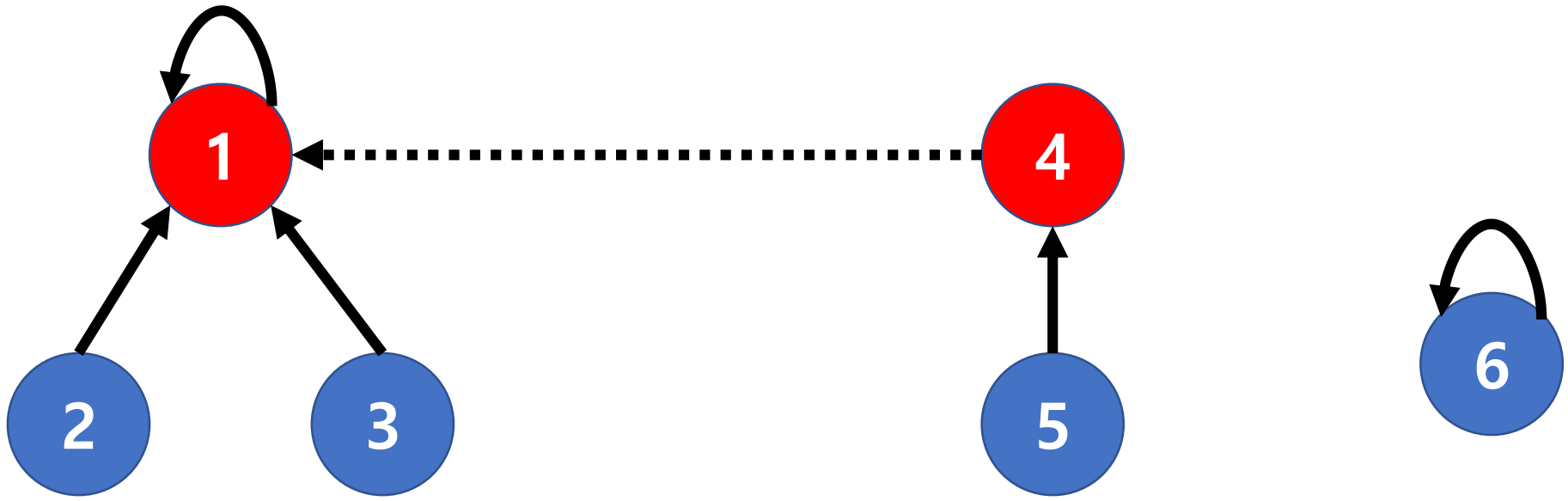
Union-Find

- union(3, 5): 3번 정점이 속한 집합과 5번 정점이 속한 집합을 합친다.



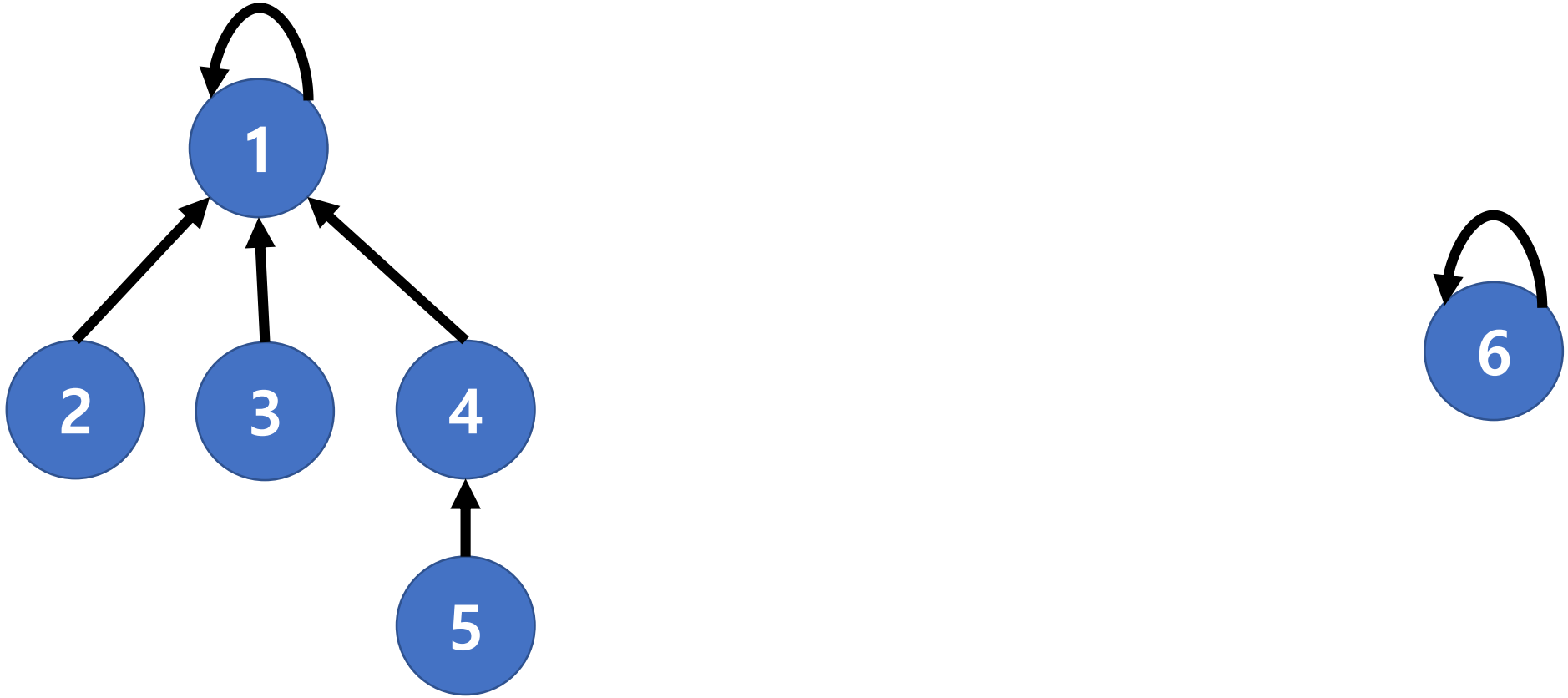
Union-Find

- union(3, 5): 3번 정점이 속한 집합과 5번 정점이 속한 집합을 합친다.



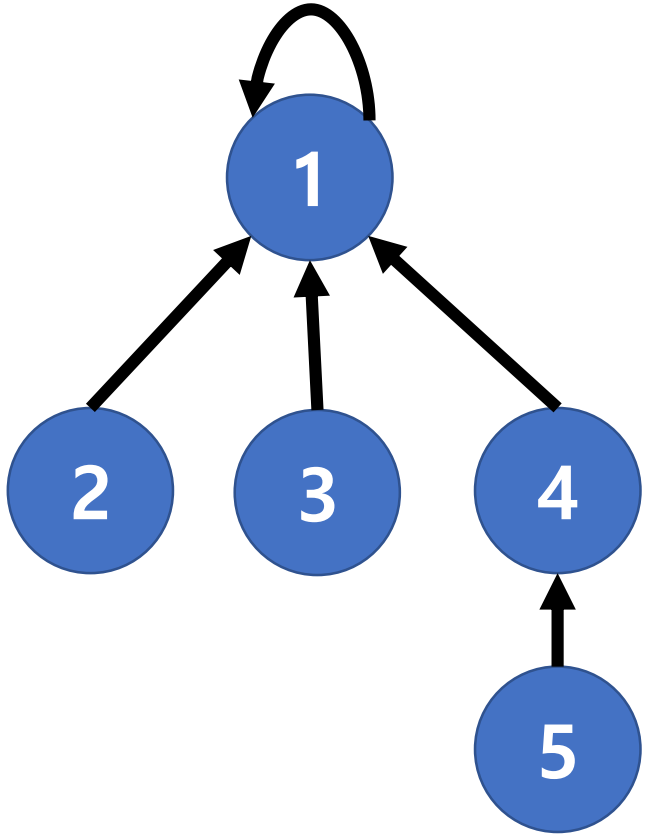
Union-Find

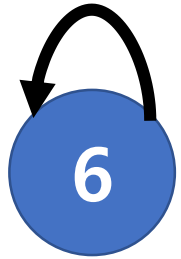
- union(3, 5): 3번 정점이 속한 집합과 5번 정점이 속한 집합을 합친다.



Union-Find

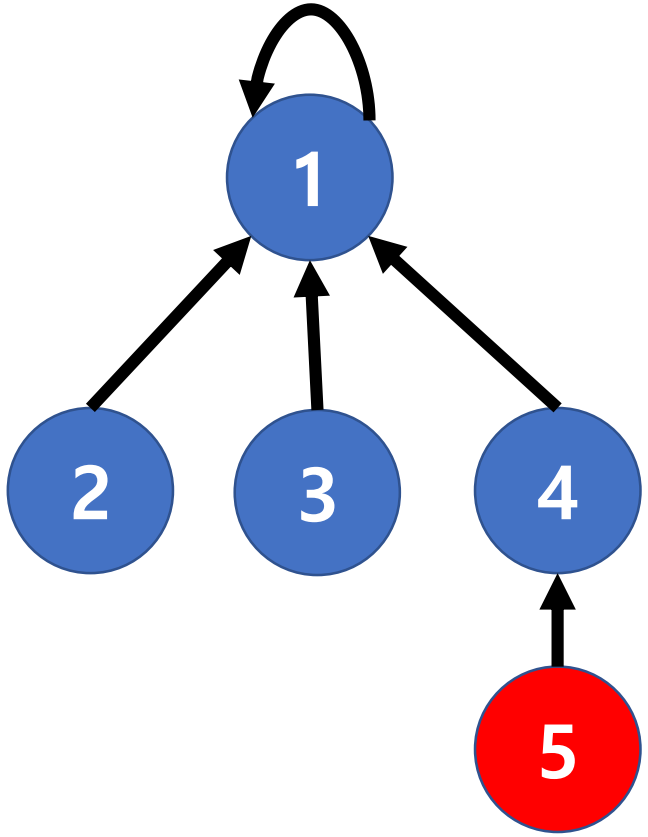
- find(5): 정점 5가 속하는 트리의 루트를 반환한다.

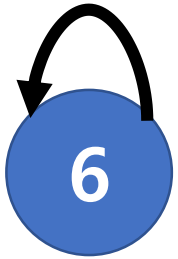




Union-Find

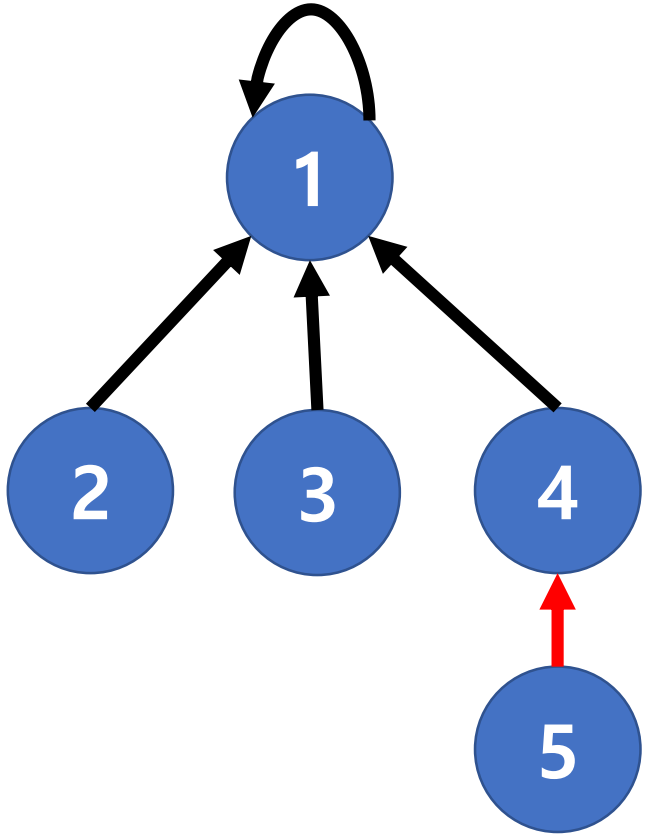
- find(5): 정점 5가 속하는 트리의 루트를 반환한다.

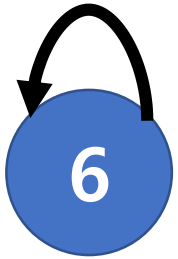




Union-Find

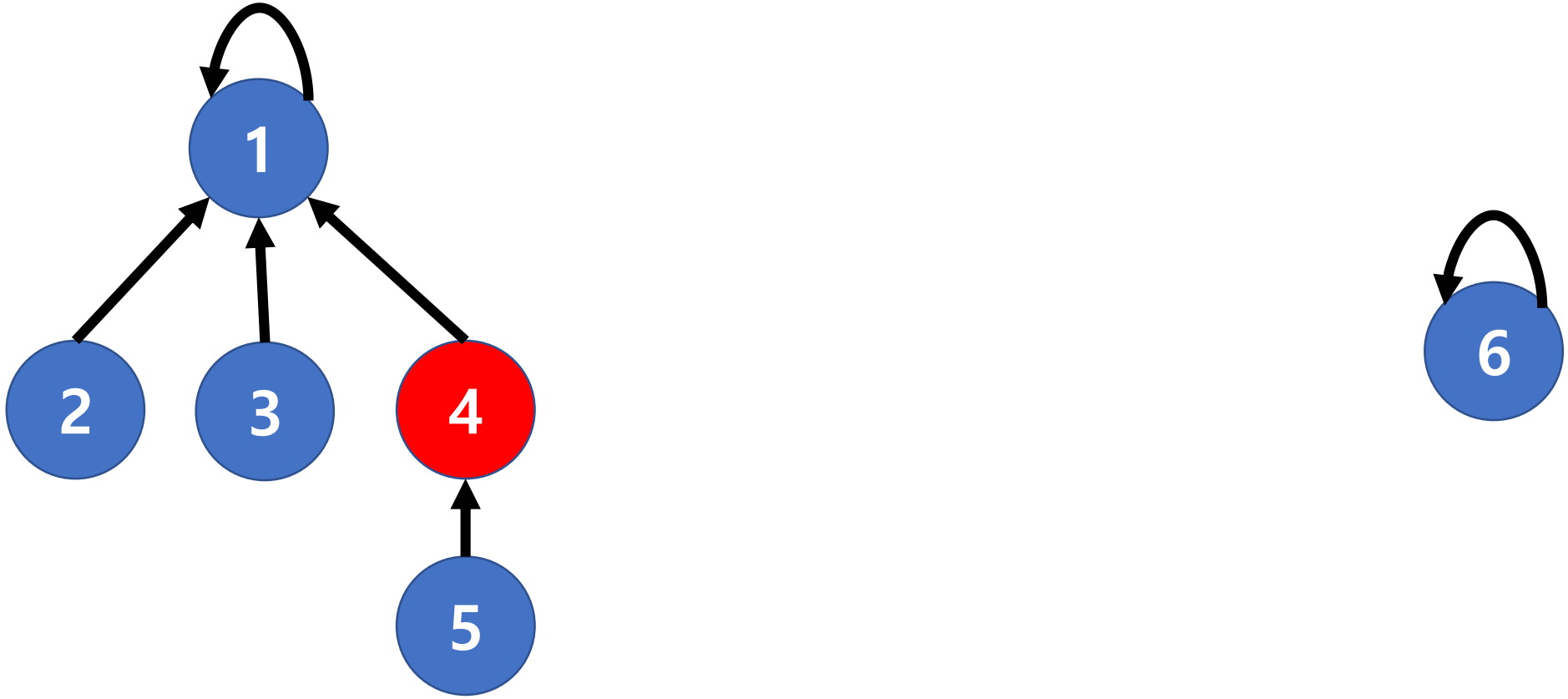
- find(5): 정점 5가 속하는 트리의 루트를 반환한다.





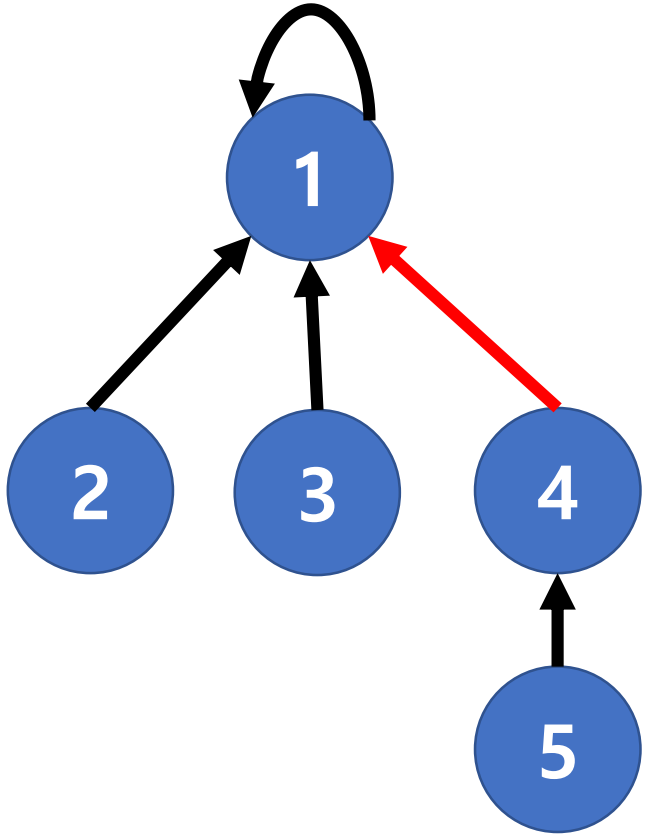
Union-Find

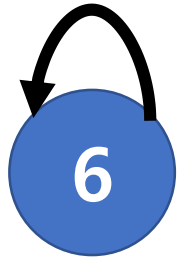
- find(5): 정점 5가 속하는 트리의 루트를 반환한다.



Union-Find

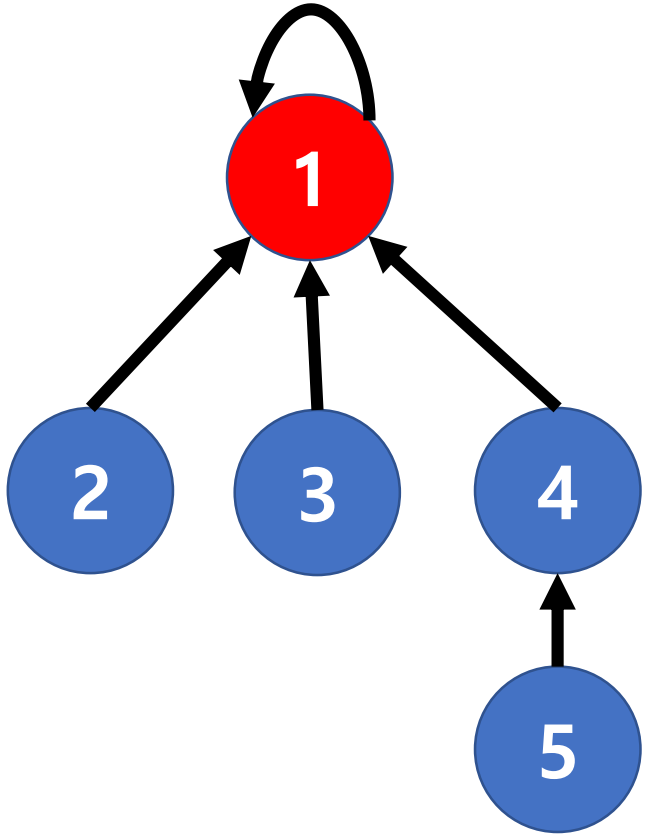
- find(5): 정점 5가 속하는 트리의 루트를 반환한다.

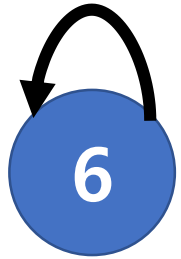




Union-Find

- find(5): 정점 5가 속하는 트리의 루트를 반환한다.





Union-Find

- pseudo code:

```
int par[MAXN];
function find(int x) {
	if (x == par[x]) return x;
	return find(par[x]);
}
function union(int x, int y) {
	x = find(x), y = find(y);
	if (x != y) par[x] = y;
}
```

Union-Find

- pseudo code:

```
int par[MAXN];
function find(int x) { // O(n) => 트리의 깊이를 줄여보자
	if (x == par[x]) return x;
	return find(par[x]);
}
function union(int x, int y) {
	x = find(x), y = find(y);
	if (x != y) par[x] = y;
}
```

Union-Find

- pseudo code:

```
int par[MAXN];
function find(int x) { // O(n) => 트리의 깊이를 줄여보자
	if (x == par[x]) return x;
	return par[x] = find(par[x]);
}
function union(int x, int y) {
	x = find(x), y = find(y);
	if (x != y) par[x] = y;
}
```

연습 문제

- BOJ 그룹에 있는 연습
- https://leetcode.com/problems/insert-delete-getrandom-o1/
- https://leetcode.com/problems/insert-delete-getrandom-o1-duplicates-allowed/
- https://leetcode.com/problems/ugly-number-ii/