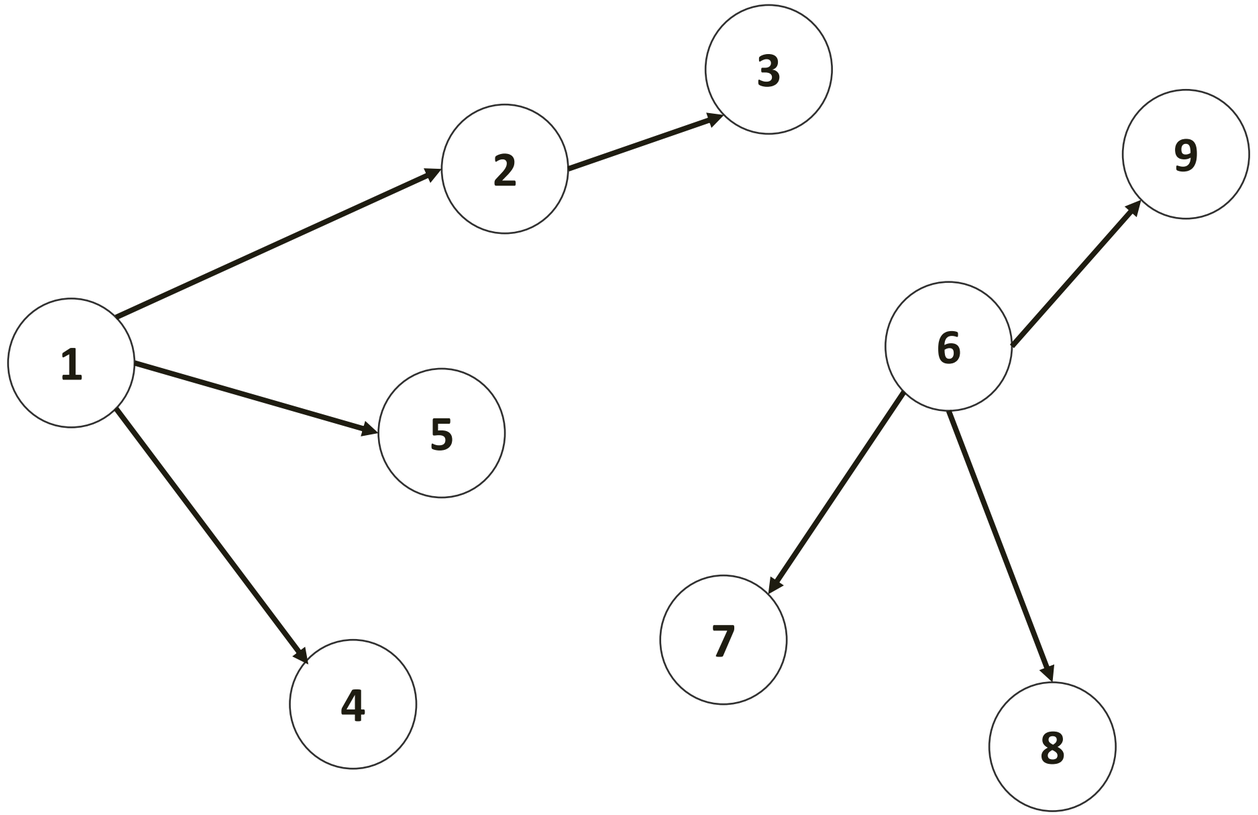
2025 겨울방학 알고리즘 스터디

분리 집합

컴퓨터 공학과 20230546 서보경

목차

분리 집합이 필요한 이유



가령, 위와 같은 그래프에서 계속 정점과 간선이 추가된다고 가정하자.
이때, 문제 상황으로 A 정점과 B 정점이 같은 컴포넌트인지 묻는 질문이 들어온다고 하자.
문제를 어떻게 풀 수 있을까? 그래프 탐색? dp?

분리 집합이 필요한 이유

그래프 탐색

일단 정점이 추가될 때 마다 계속 모든 컴포넌트를 탐색한다고 가정하자.
정점 개수가 적은 상황에선 유효할지 모르지만 이런 방식이면 무식하게 브루트포스 돌리는 것과 다를게 없다.

시간 복잡도 = O(Q*V) (Q는 쿼리의 개수, V는 정점의 개수)

다이나믹 프로그래밍

다이나믹 프로그래밍은 직전 상태를 저장(메모이제이션)하면서 해를 구하는 최적화 기법 중 하나이다.
그러나 이런 문제를 풀기엔 적합 하지 않다. 기본적으로 다이나믹 프로그래밍은 최적화된 브루트포스 기법이기 때문에 상태 전이와 갱신에 시간이 꽤 많이 든다.

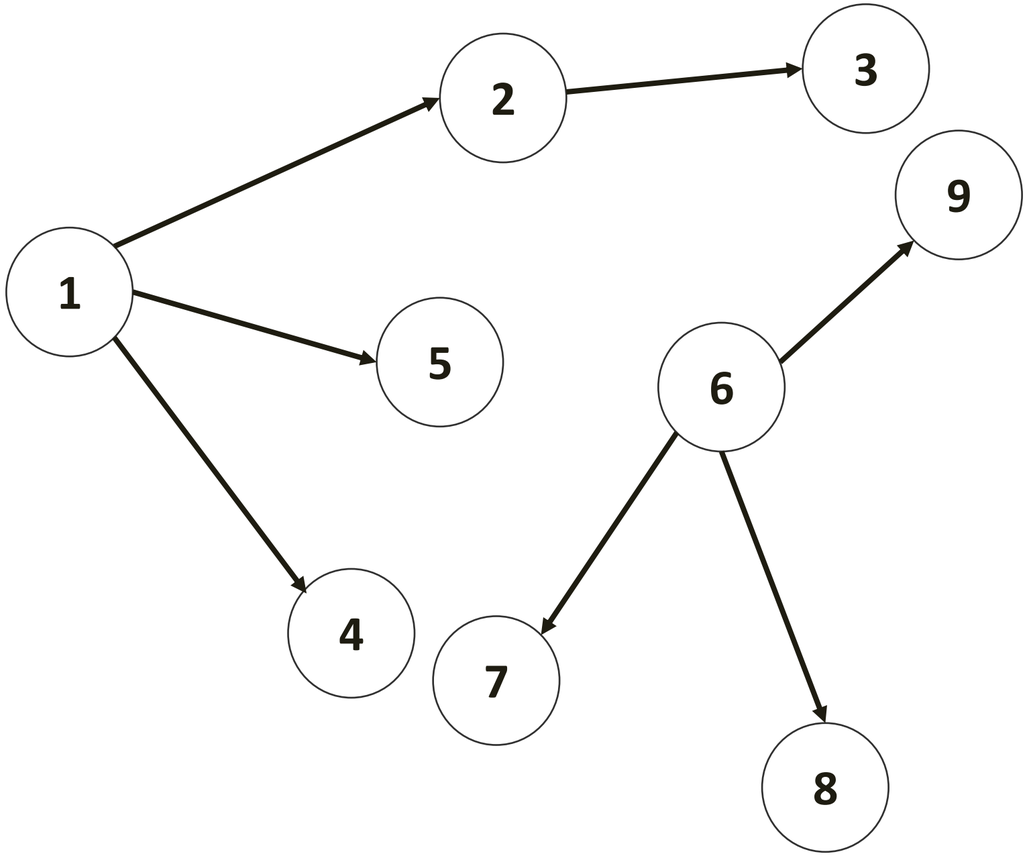
시간 복잡도 = O(Q*V) (Q는 쿼리의 개수, V는 정점의 개수)

분리 집합

분리 집합(disjoint set)은 주로 여러 개의 집합을 관리하면서, 집합 간의 연산을 빠르게 수행하기 위해 사용 되는 자료구조이다.
특히 온라인 쿼리(쿼리가 주어지고 쿼리에 대해서 연산을 수행한 다음 답을 요구하는 문제) 문제에 특화 되어 있으며 시간복잡도도 매우 낮다!

시간 복잡도 = O(α(V)) (역 아커만 함수. 보통 어떤 V에 대해서도 5미만의 값이다. / 경로 압축 썼을 때)

분리 집합 만들기



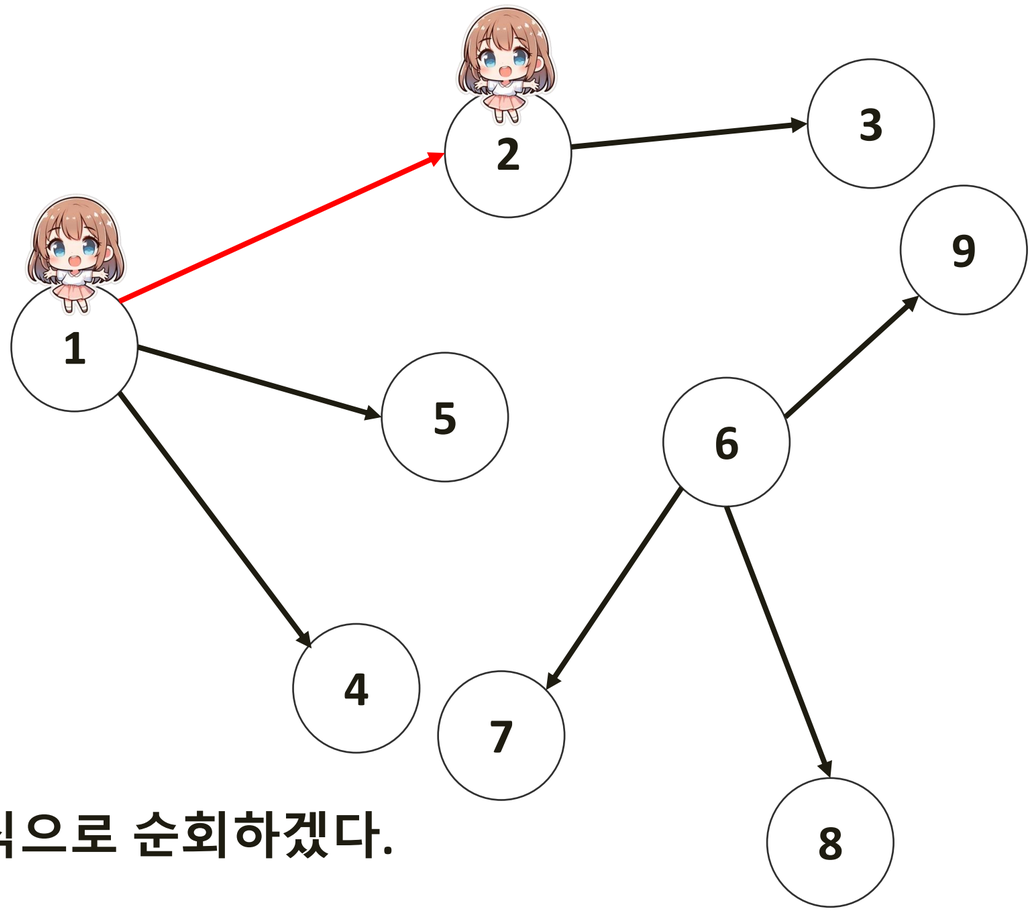
정점	1	2	3	4	5	6	7	8	9
부모	1	2	3	4	5	6	7	8	9

아까 본 그래프를 재활용 하겠다.

먼저, 분리 집합을 만들기에 앞서 정점 수 만큼 배열을 만들어 준다.
분리집합도 일종의 트리이기 때문에 부모를 가지는데 이는 배열의 값으로 구현하도록 하겠다.

처음엔 모두 리프 노드이기 때문에 부모를 자신으로 초기화 해준다.

분리 집합 만들기



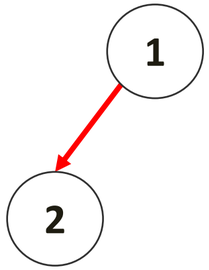
정점	1	2	3	4	5	6	7	8	9
부모	1	2	3	4	5	6	7	8	9

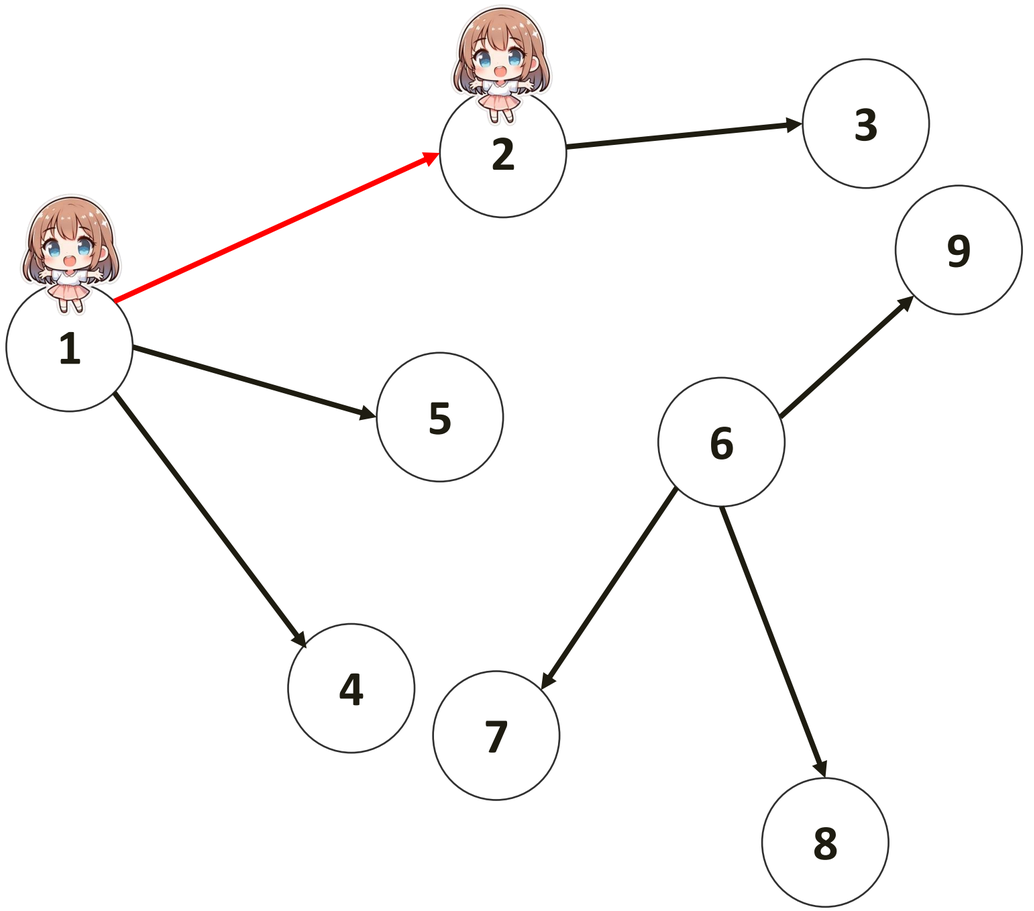
1부터 그래프 탐색을 시작하겠다. 편의를 위해 깊이 우선 탐색 방식으로 순회하겠다.

현재 정점은 1번이고 그에 연결된 2번 정점이 있다.

두 정점의 부모가 모두 본인이다. 부모가 다르니 합쳐준다!

분리 집합 만들기





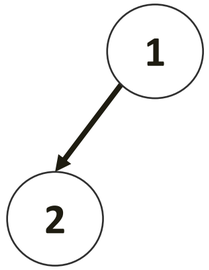
정점	1	2	3	4	5	6	7	8	9
부모	1	1	3	4	5	6	7	8	9

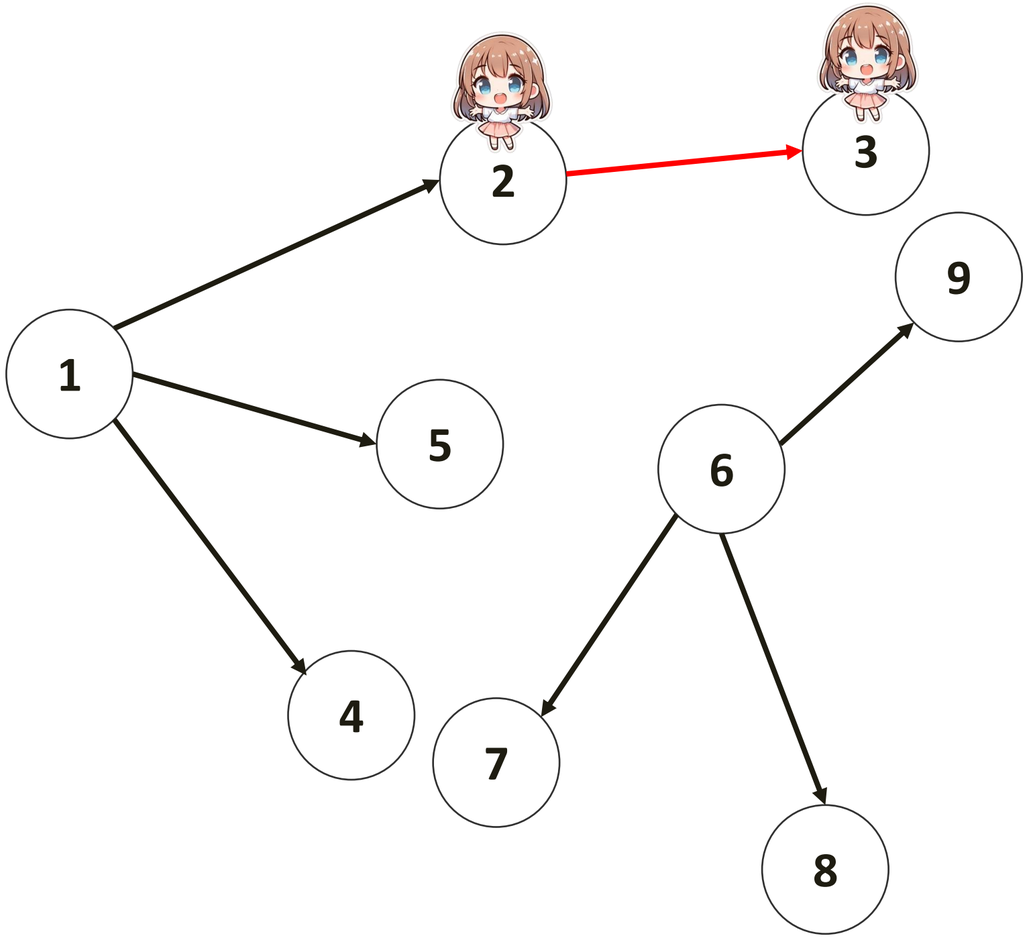
작은 정점이 큰 정점의 부모가 된다고 가정하겠다.

고로 트리는 위의 형태처럼 만들어 지게 되고, 2번 정점의 부모가 1번 정점으로 갱신된다.

이후 그래프 탐색을 재개해준다.

분리 집합 만들기



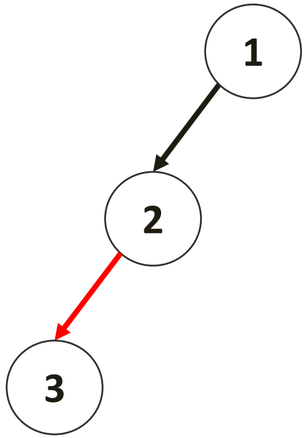


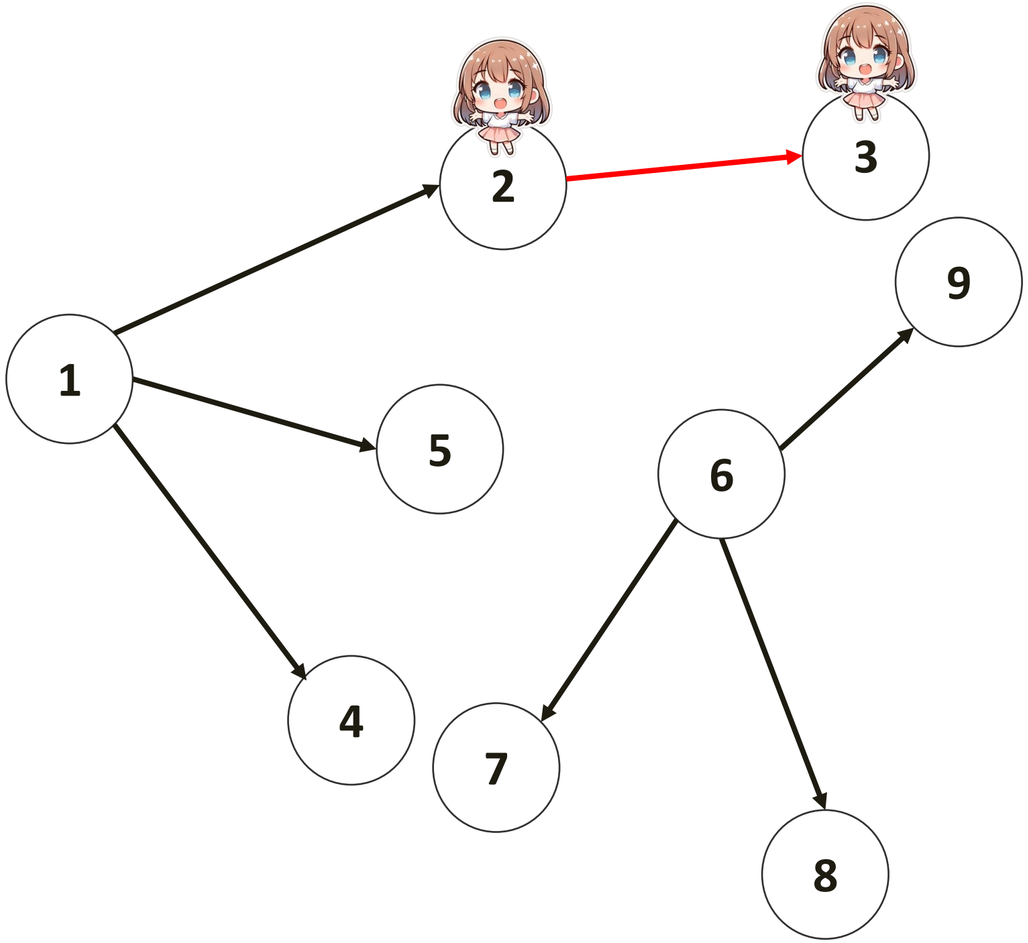
정점	1	2	3	4	5	6	7	8	9
부모	1	1	3	4	5	6	7	8	9

현재 정점은 2번이고 그에 연결된 3번 정점이 있다.

두 정점의 부모가 다르다. 고로 합쳐준다!

분리 집합 만들기





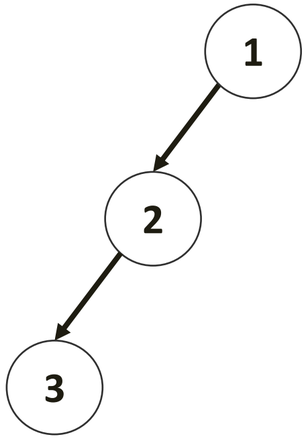
정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	4	5	6	7	8	9

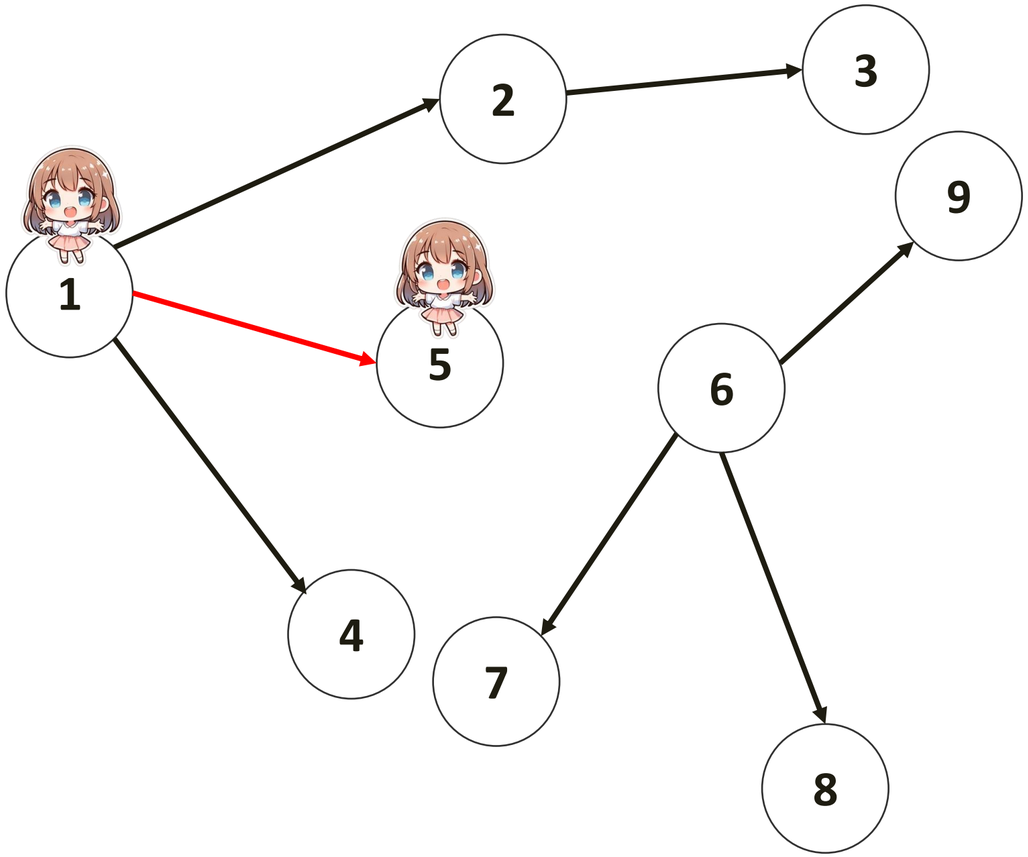
작은 정점이 큰 정점의 부모가 된다는 룰을 지켜야 한다.

고로 3번의 부모는 2번 정점이 되도록 트리를 만들어 주고, 배열 값을 갱신해준다.

이후 그래프 탐색을 재개해준다.

분리 집합 만들기



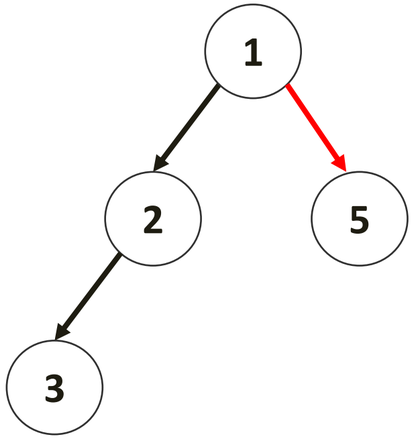


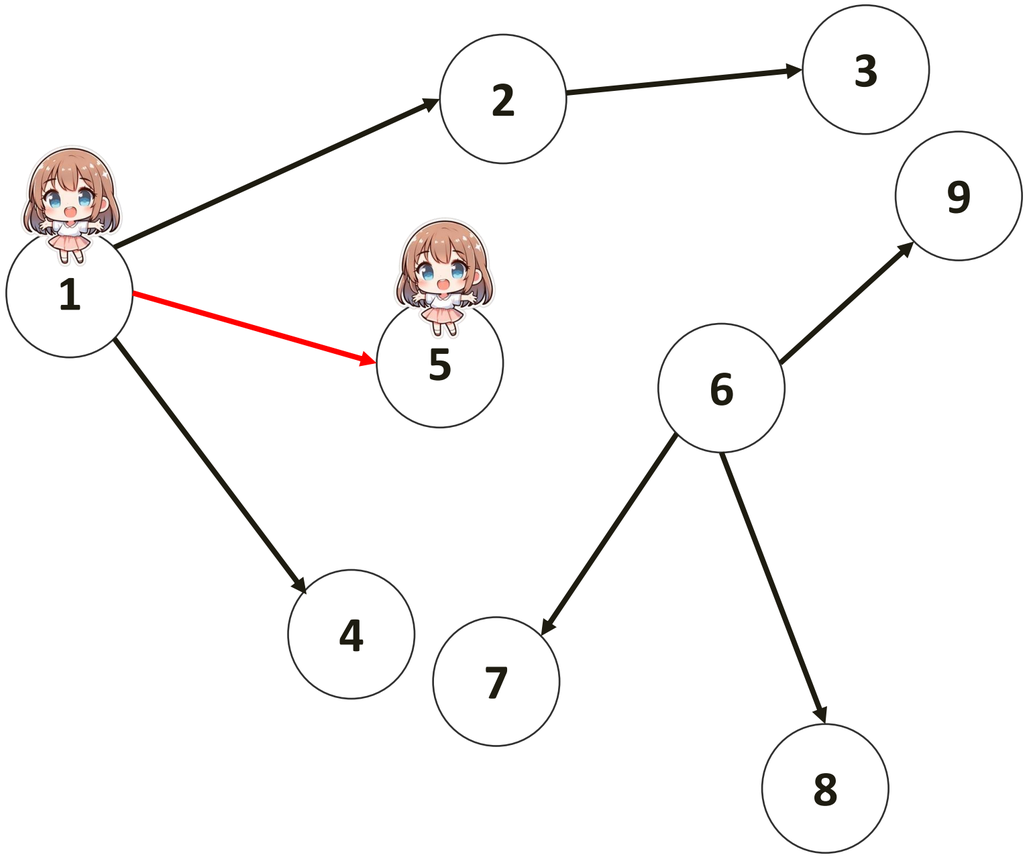
정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	4	5	6	7	8	9

현재 정점은 1번이고 그에 연결된 5번 정점이 있다.

두 정점의 부모가 다르다. 고로 합쳐준다!

분리 집합 만들기





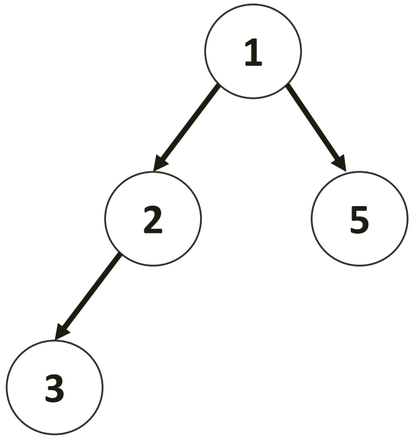
정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	4	1	6	7	8	9

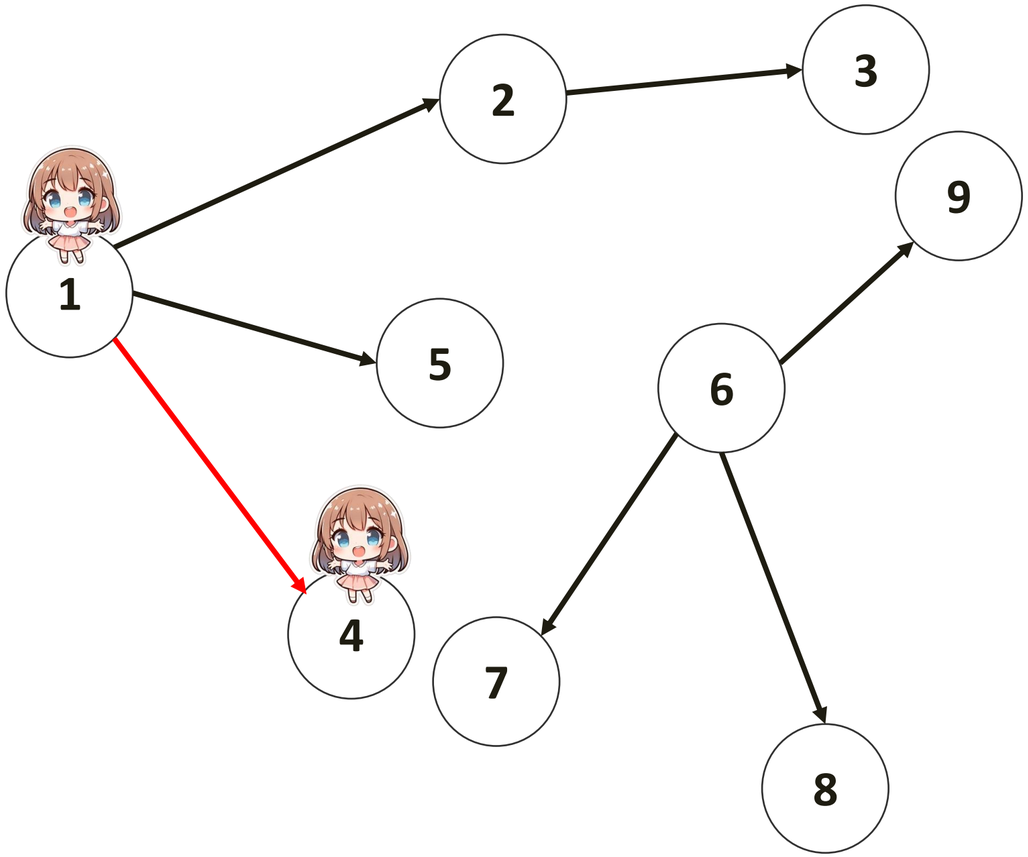
작은 정점이 큰 정점의 부모가 된다는 룰을 지켜야 한다.

고로 5번의 부모는 1번 정점이 되도록 트리를 만들어 주고, 배열 값을 갱신해준다.

이후 그래프 탐색을 재개해준다.

분리 집합 만들기



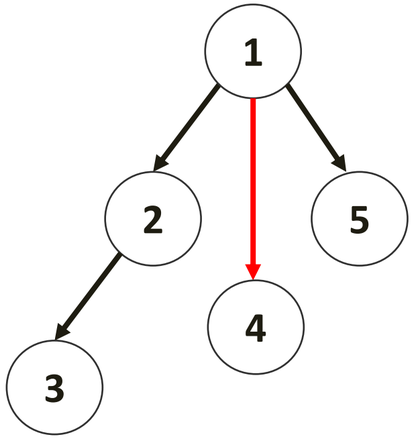


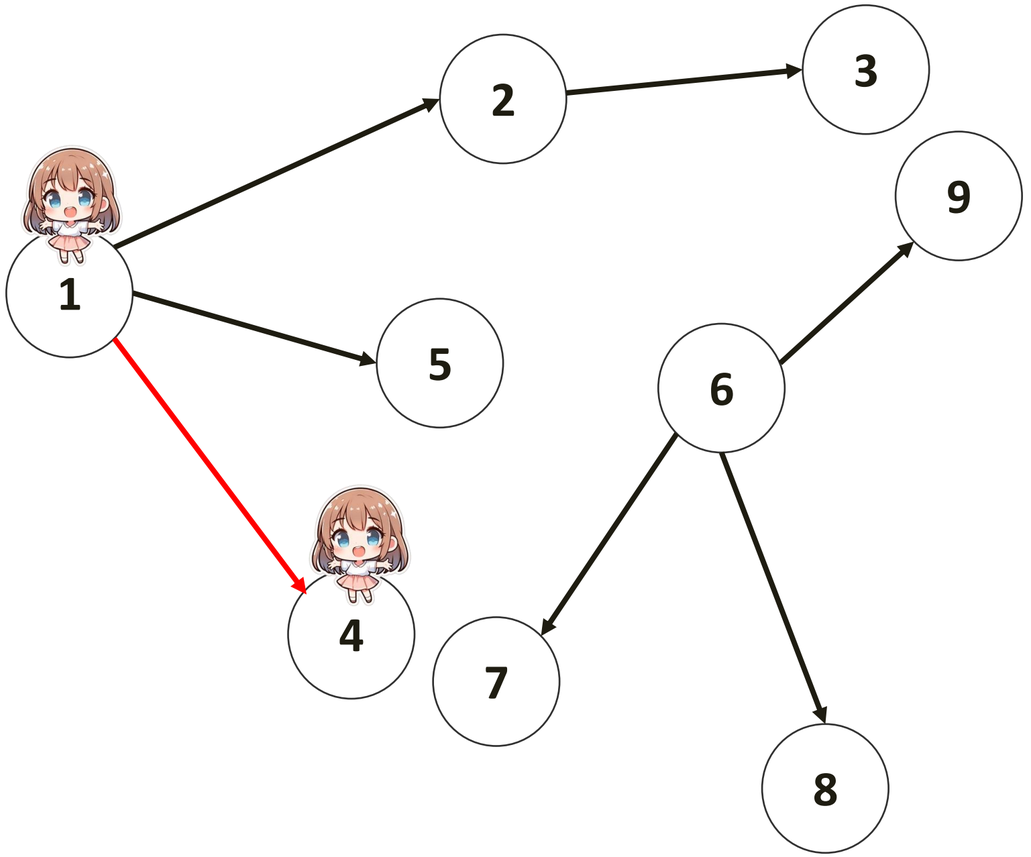
정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	4	1	6	7	8	9

현재 정점은 1번이고 그에 연결된 4번 정점이 있다.

두 정점의 부모가 다르다. 고로 합쳐준다!

분리 집합 만들기





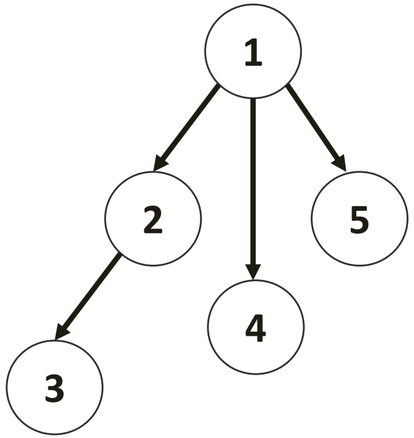
정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	7	8	9

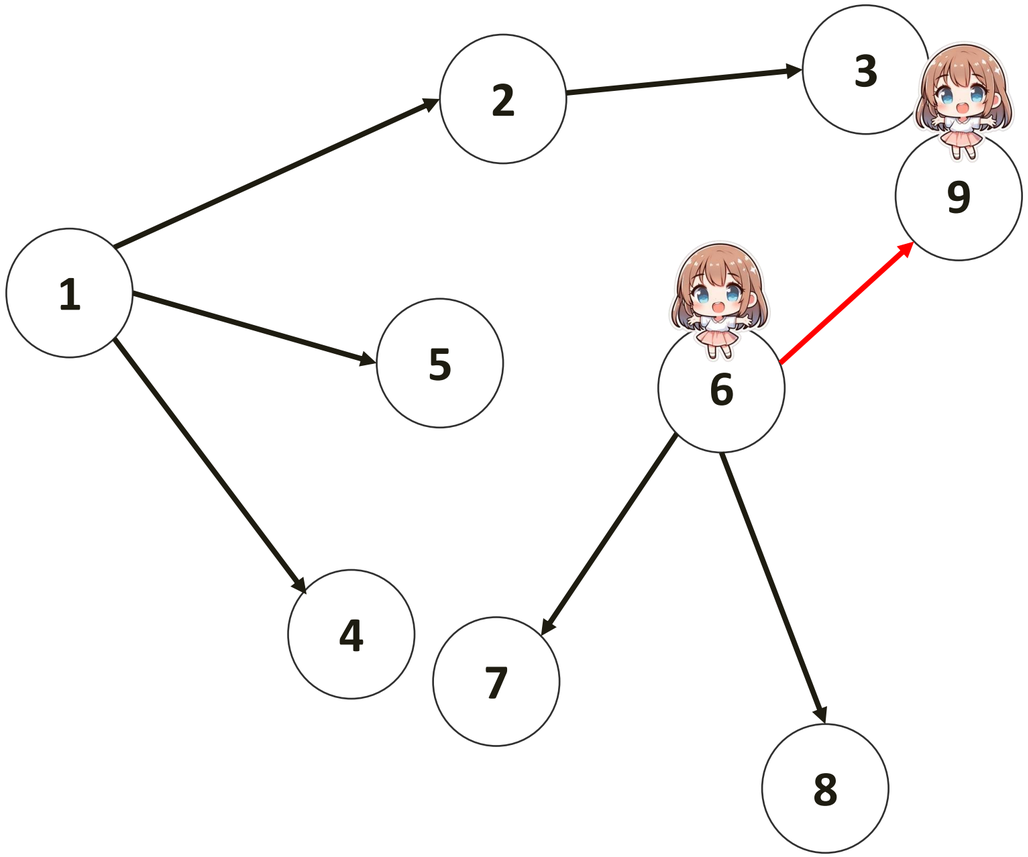
작은 정점이 큰 정점의 부모가 된다는 룰을 지켜야 한다.

고로 4번의 부모는 1번 정점이 되도록 트리를 만들어 주고, 배열 값을 갱신해준다.

한 컴포넌트의 탐색이 끝났다. 이제 6번 정점으로 이동하겠다.

분리 집합 만들기



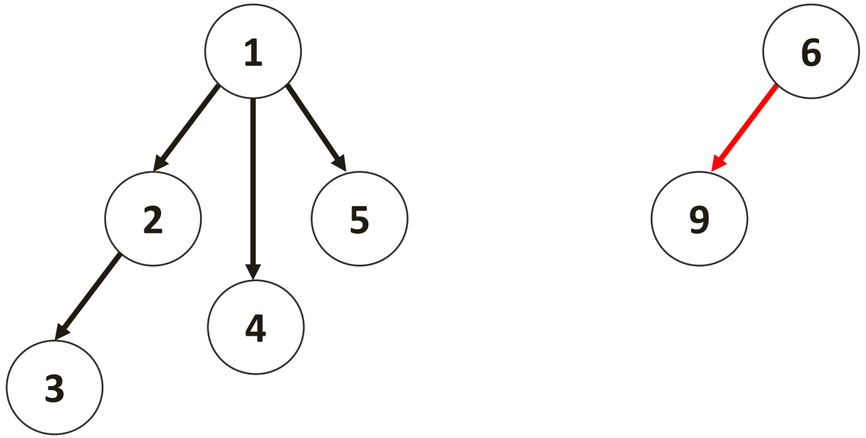


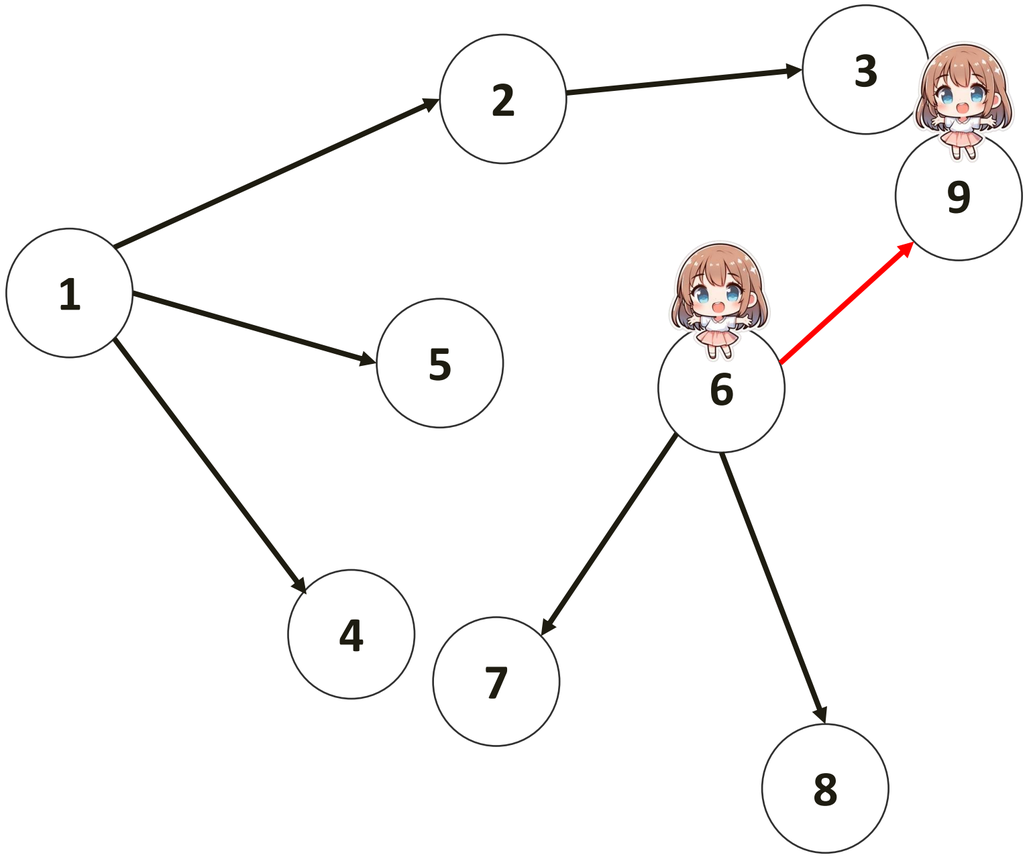
정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	7	8	9

현재 정점은 6번이고 그에 연결된 9번 정점이 있다.

두 정점의 부모가 다르다. 고로 합쳐준다!

분리 집합 만들기





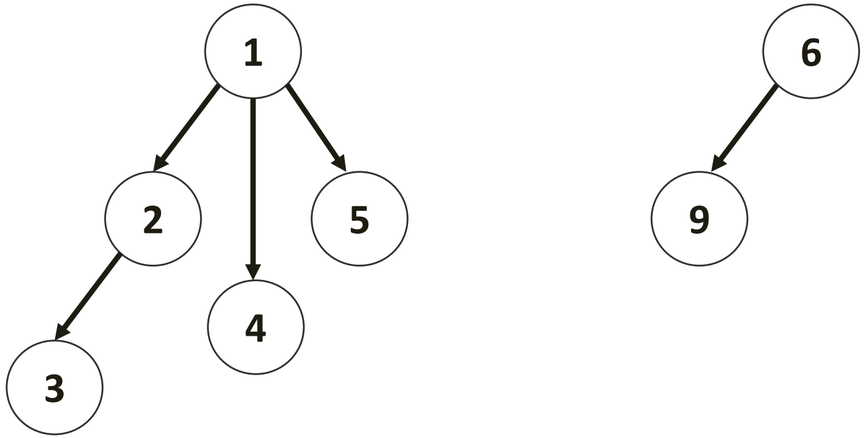
정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	7	8	6

작은 정점이 큰 정점의 부모가 된다는 룰을 지켜야 한다.

고로 9번의 부모는 6번 정점이 되도록 트리를 만들어 주고, 배열 값을 갱신해준다.

이후 그래프 탐색을 재개해준다.

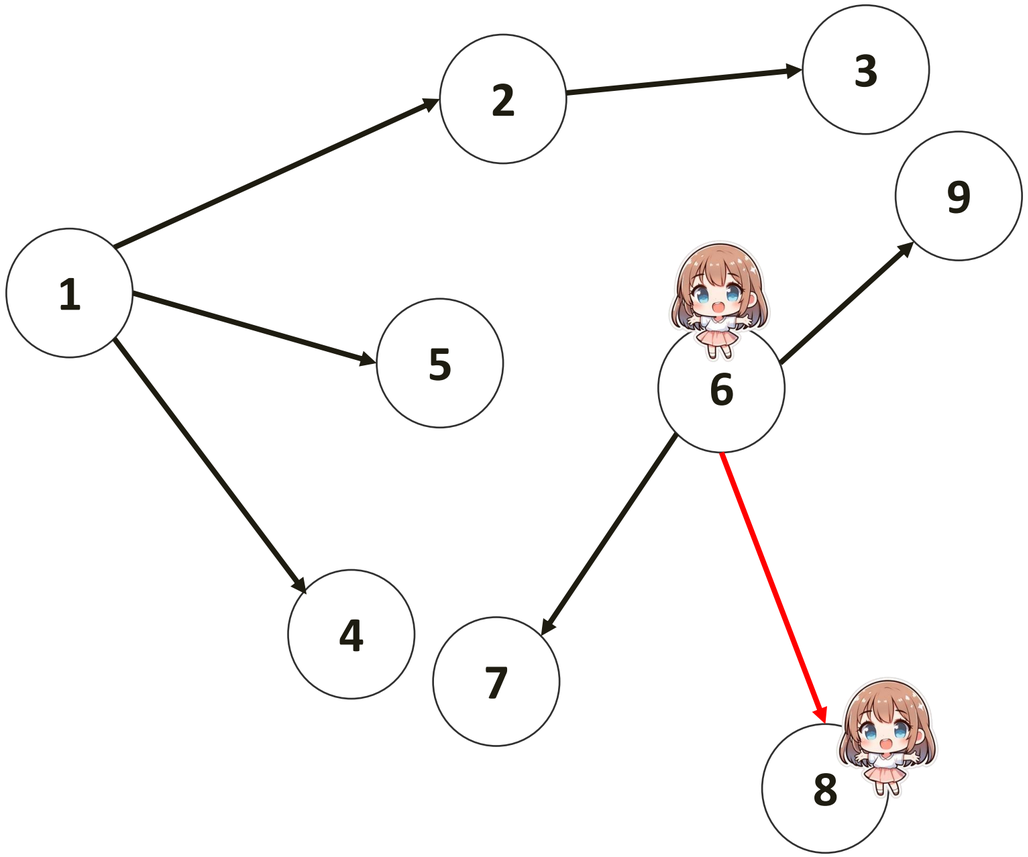
분리 집합 만들기



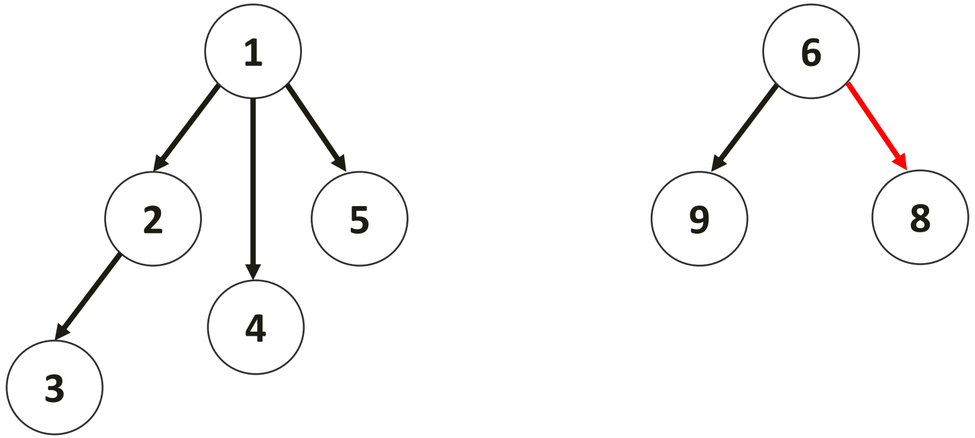
정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	7	8	6

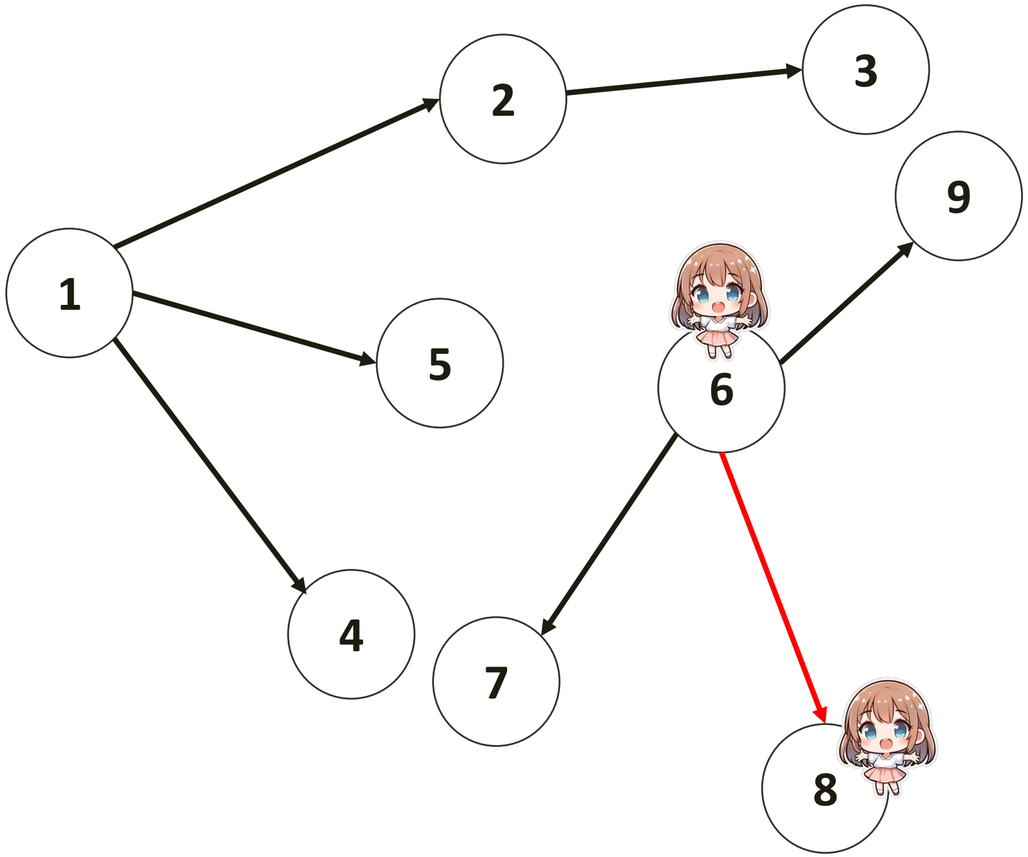
현재 정점은 6번이고 그에 연결된 8번 정점이 있다.

두 정점의 부모가 다르다. 고로 합쳐준다!



분리 집합 만들기





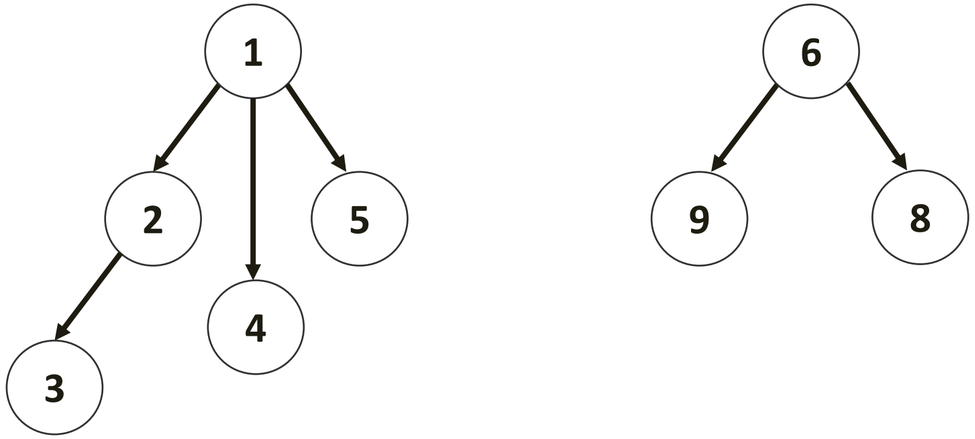
정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	7	6	6

작은 정점이 큰 정점의 부모가 된다는 룰을 지켜야 한다.

고로 8번의 부모는 6번 정점이 되도록 트리를 만들어 주고, 배열 값을 갱신해준다.

이후 그래프 탐색을 재개해준다.

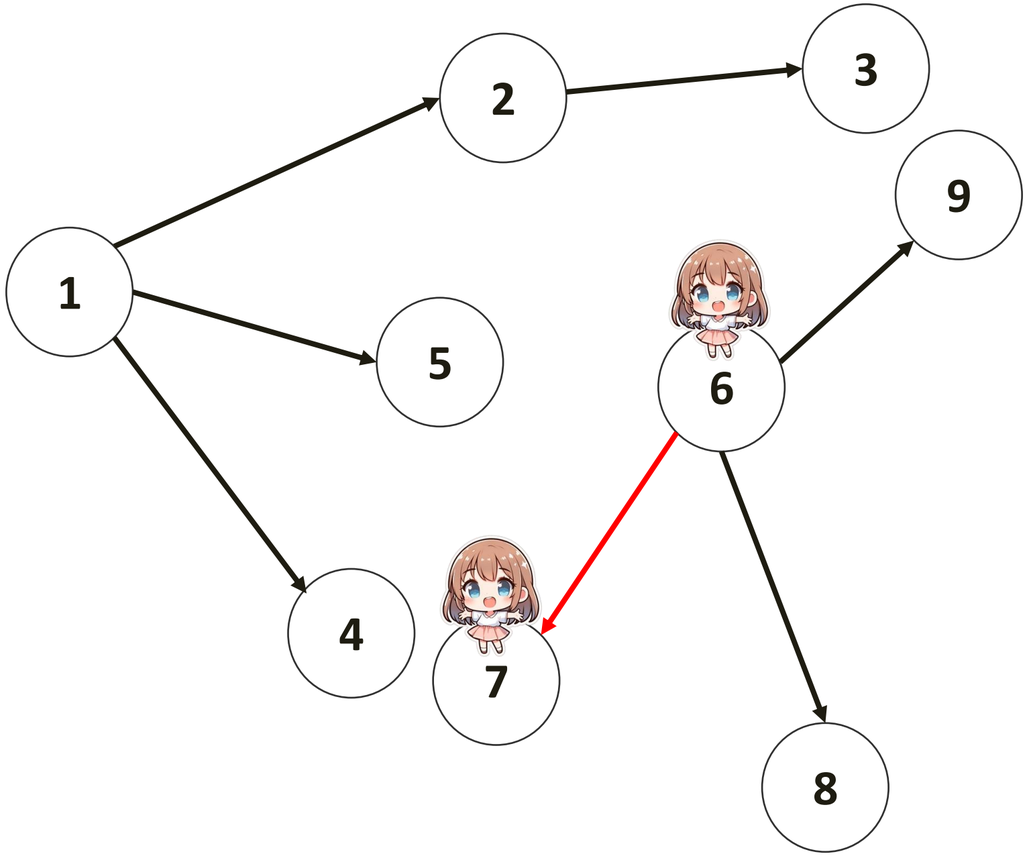
분리집합 만들기



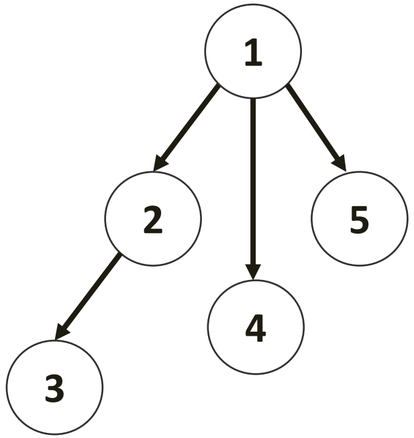
정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	7	6	6

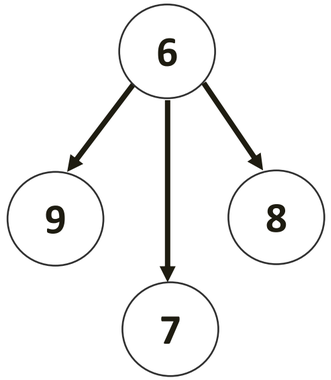
현재 정점은 6번이고 그에 연결된 7번 정점이 있다.

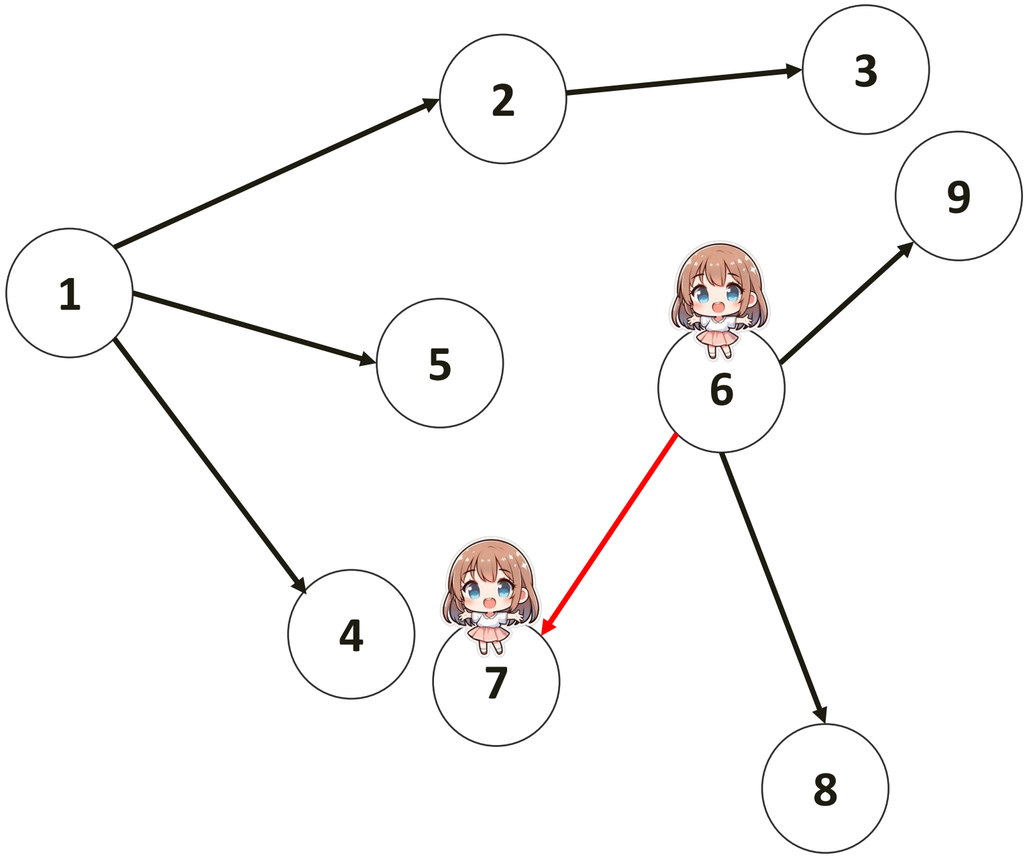
두 정점의 부모가 다르다. 고로 합쳐준다!



분리 집합 만들기







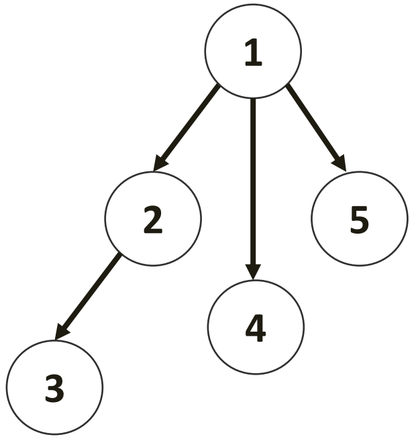
정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	6	6	6

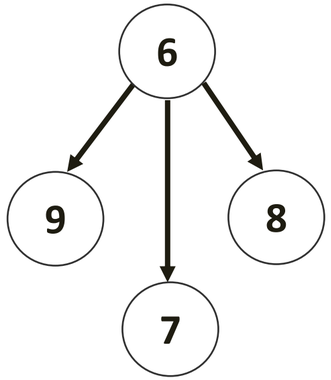
작은 정점이 큰 정점의 부모가 된다는 룰을 지켜야 한다.

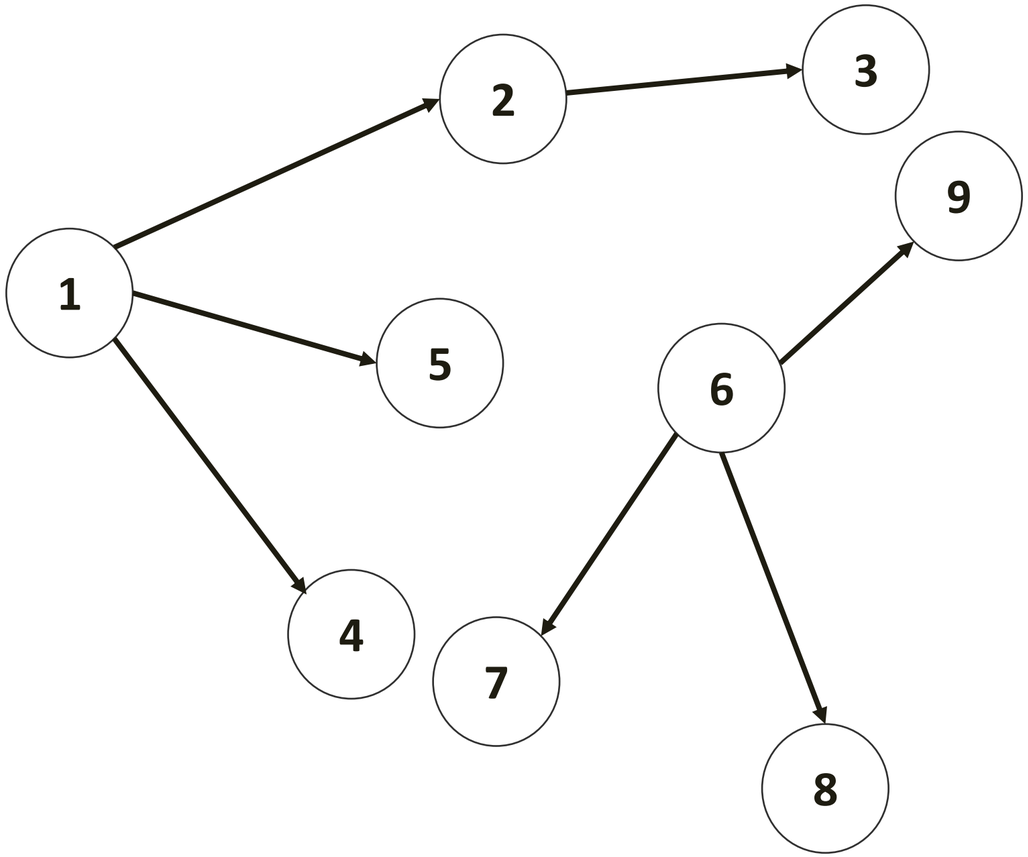
고로 7번의 부모는 6번 정점이 되도록 트리를 만들어 주고, 배열 값을 갱신해준다.

분리 집합 만들기가 끝났다.

분리 집합 만들기







정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	6	6	6

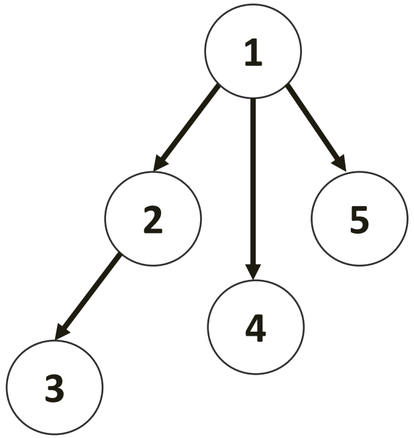
query : 1번 == 4번?

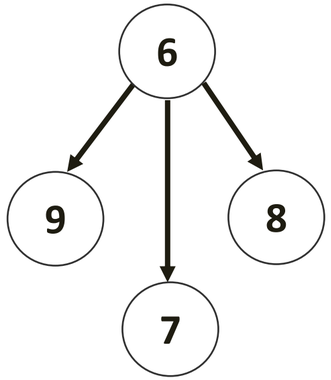
1번과 4번 정점이 같은 컴포넌트에 있는지 묻는 쿼리다.

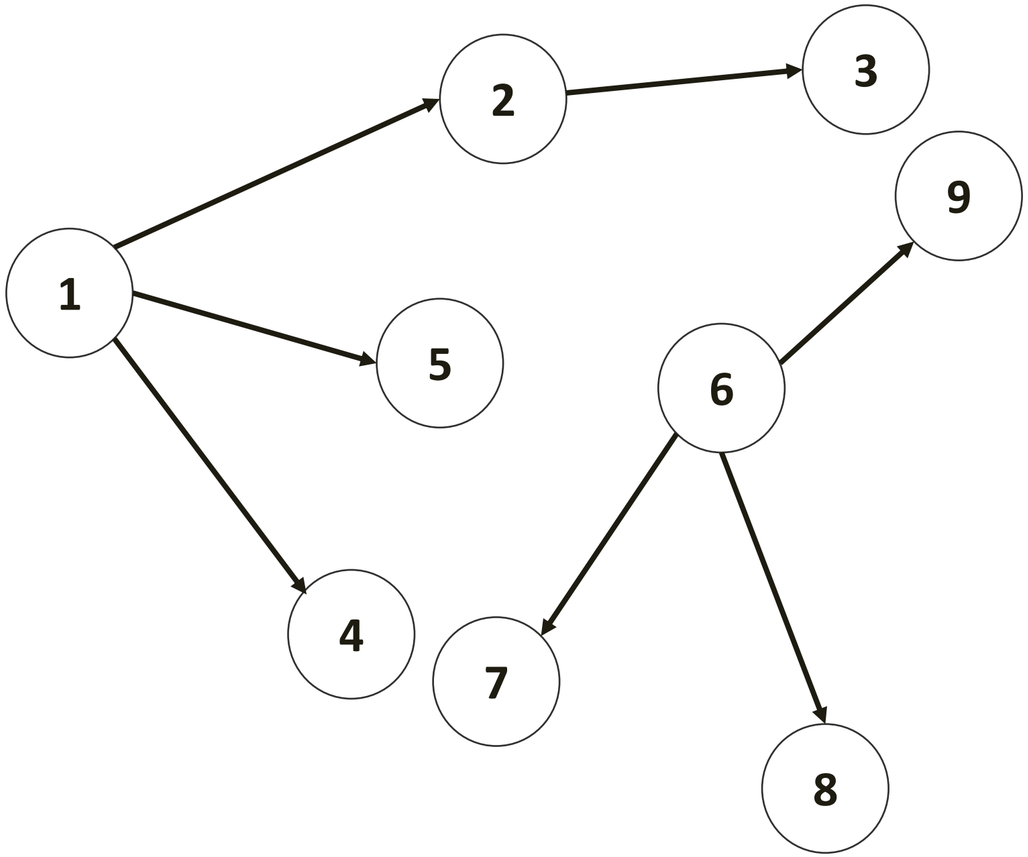
먼저 배열의 값(부모)를 확인해준다.

서로 같다! 고로 같은 컴포넌트가 맞다!

분리 집합 만들기







정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	6	6	6

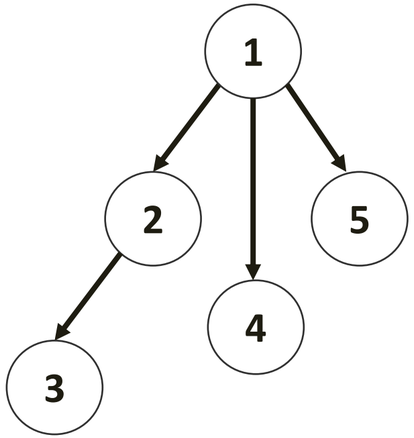
query : 2번 == 3번?

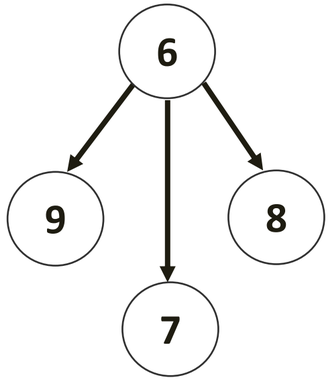
2번과 3번 정점이 같은 컴포넌트에 있는지 묻는 쿼리다.

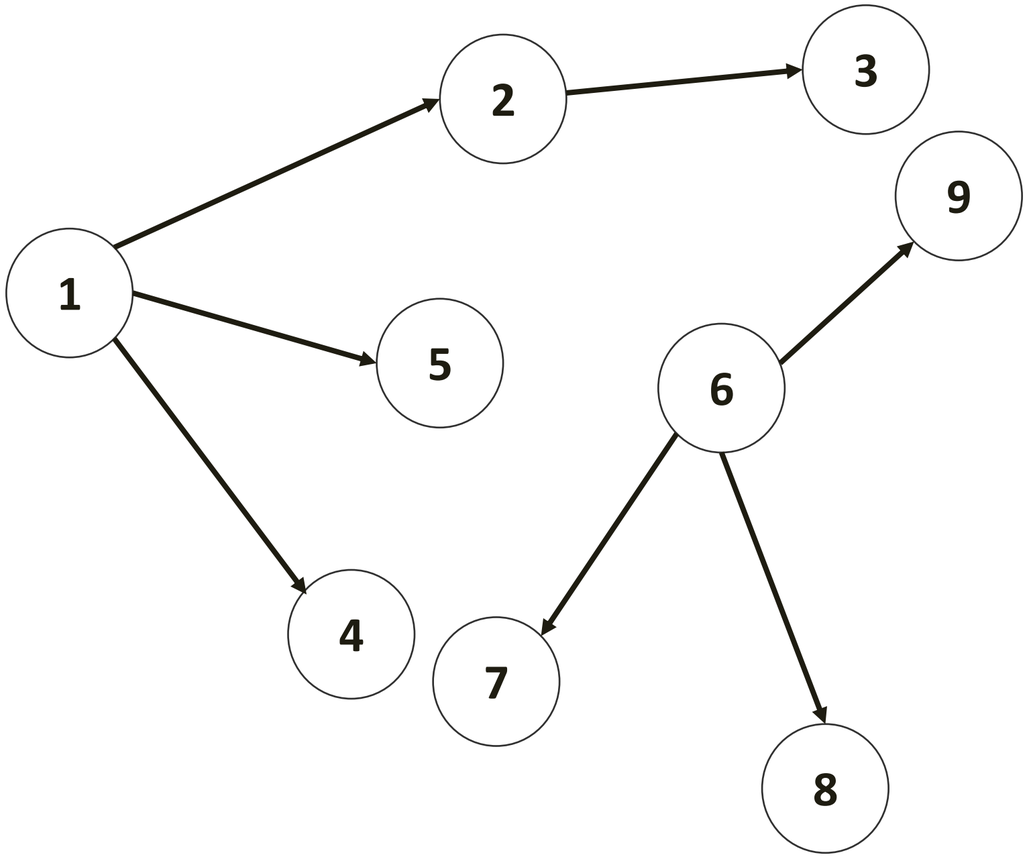
먼저 배열의 값(부모)를 확인해준다.

서로 다르다. 같은 컴포넌트가 아닌걸까?

분리 집합 만들기







정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	6	6	6

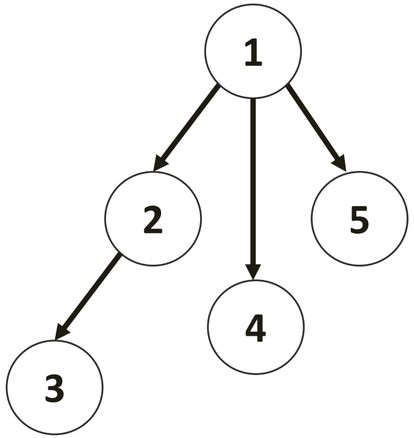
query : 2번 == 3번?

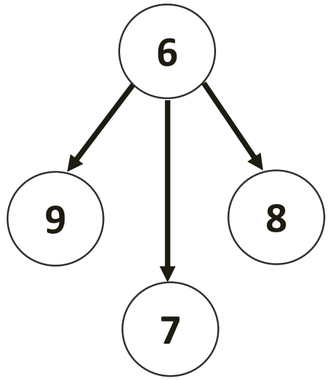
한 트리에서 부모가 다를 수는 있지만 루트 노드가 다를 수 없다.

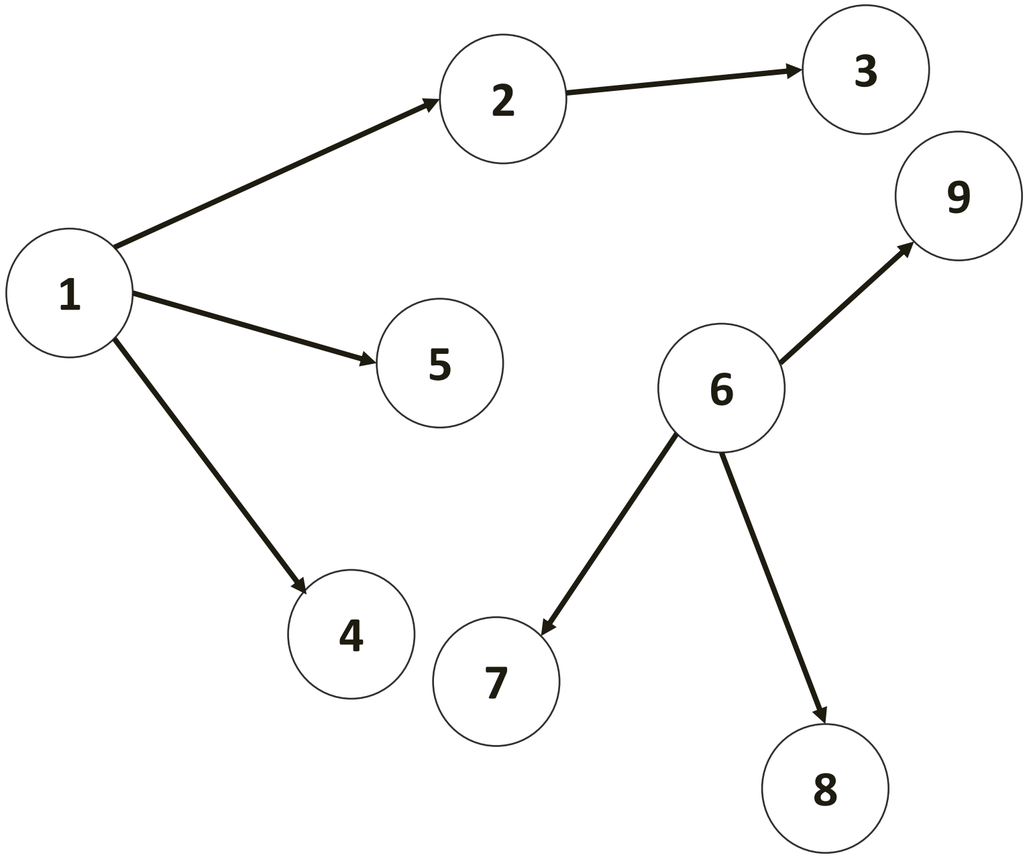
그렇기 때문에 계속 거슬러 올라가면 된다.

이때 2번 정점의 루트는 1번, 3번 정점의 루트는 1번이기 때문에 같은 컴포넌트에 속한다!

분리 집합 만들기







정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	6	6	6

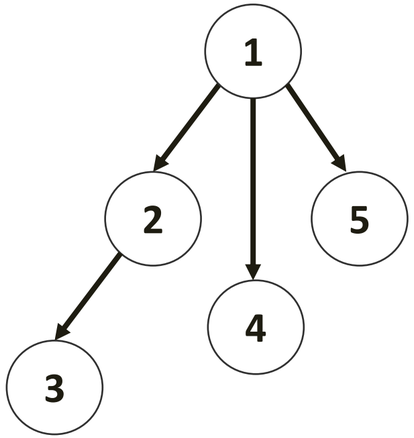
query : 7번 == 3번?

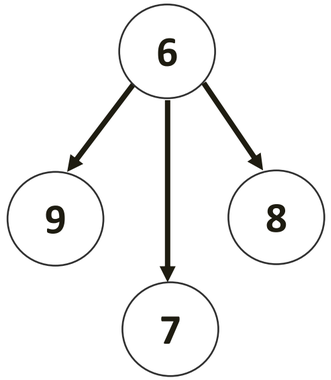
7번과 3번 정점이 같은 컴포넌트에 있는지 묻는 쿼리다.

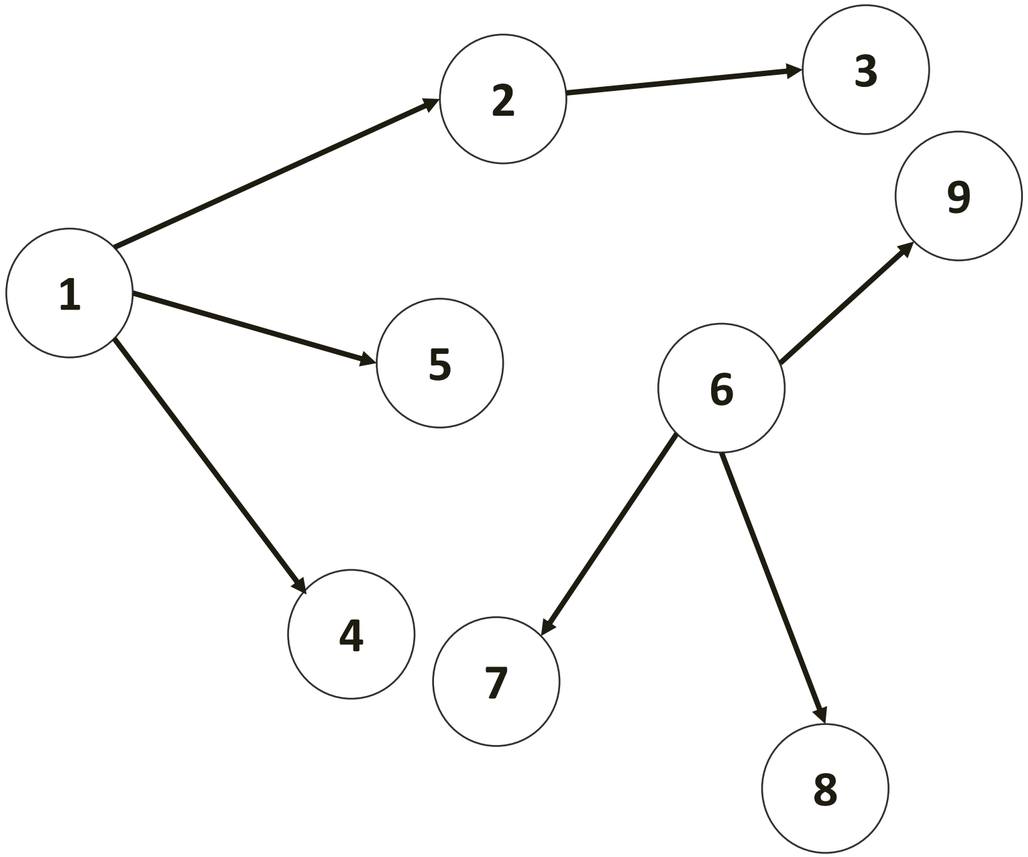
먼저 배열의 값(부모)를 확인해준다.

서로 다르기 때문에 루트 노드를 추적해준다.

분리 집합 만들기







정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	6	6	6

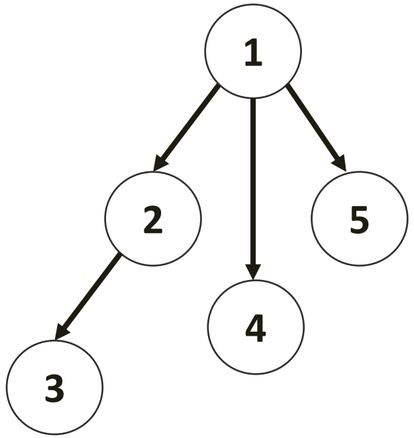
query : 7번 == 3번?

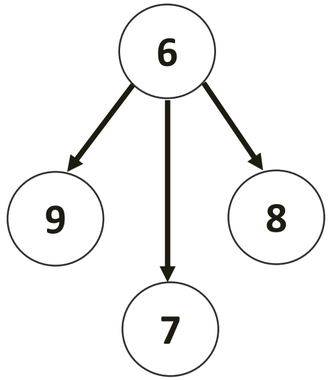
7번을 거슬러 올라갔을 때 루트 노드는 6번이다.

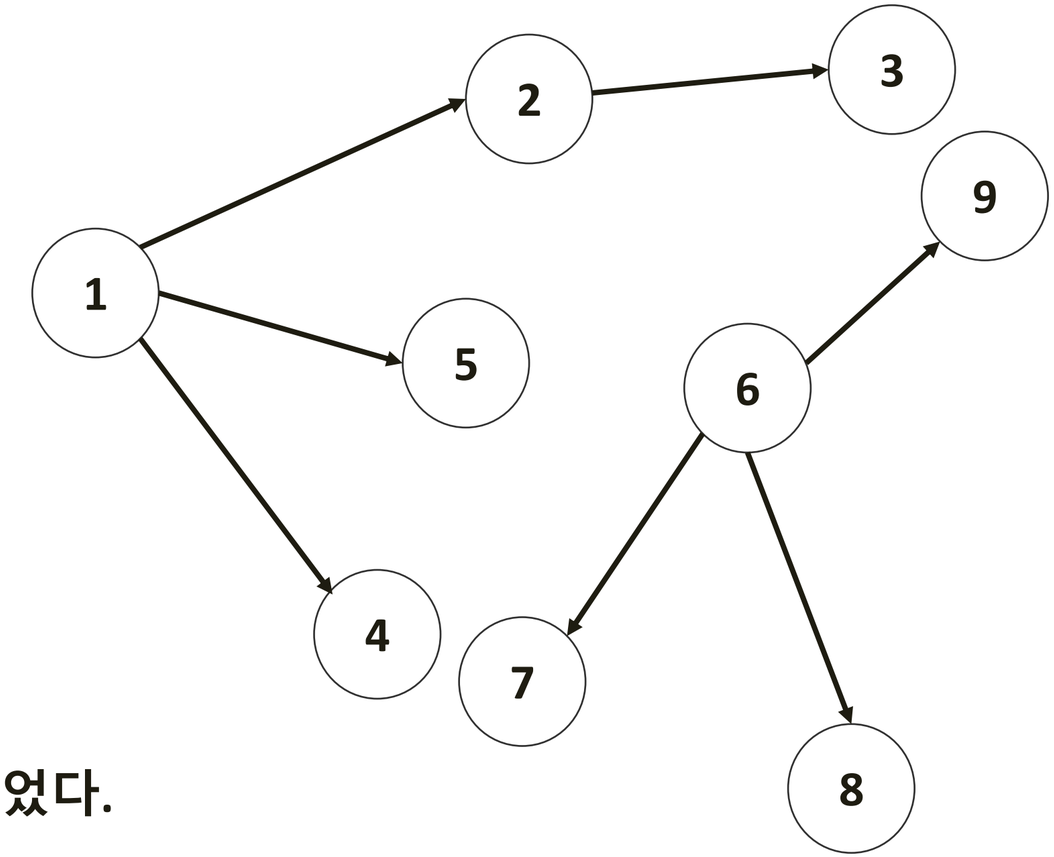
3번을 거슬러 올라갔을 때 루트 노드는 1번이다.

서로 루트 노드가 다르다. 고로 같은 컴포넌트가 아니다.

분리 집합 만들기







정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	6	6	6

보여준 예시에서는 그래프 탐색을 하면서 분리 집합(트리)를 만들었다.

하지만 실시간 입력을 받으면서 충분히 트리를 만들 수 있으며, 다른 여러가지 방면으로 활용이 가능하다.

그래프 탐색과 같은 다른 방법에 비해서 효율적으로 보이...나? 정말 그럴까?

구현한 예시의 문제점

선형 트리

만약 앞서 제시한 그래프와 같이 이쁜 트리가 안 나오고 선형으로 나온다면 조상(루트)를 찾아가기 위해서 얼마나 올라가야 할까?

만약 분리 집합 트리에서 맨 밑에 있는 친구만 무작정 호출하는 쿼리를 Q번 주게 된다면??

시간 복잡도 : O(Q*V)

트리 간의 병합

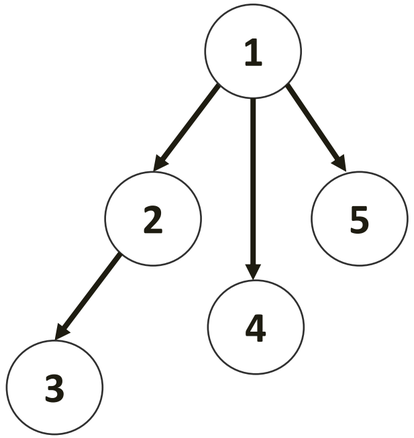
선형 트리라고 아니라고 가정하자. 앞서 보여준 예시는 루트가 이미 확보된 상태에서 리프 정점들만 흡수하는 형식이다. 과연 리프 정점만 흡수 할 수 있을까?
그렇지 않다. 리프 정점, 트리 등등 여러가지를 흡수할 수 있다!

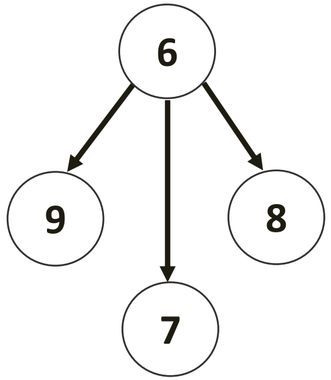
흡수를 했을 때 배열의 부모를 갱신해 주어야 하는데, 만약 정점의 수가 큰놈일수록 서브 트리의 정점 개수가 많고 작은놈일수록 정점 개수가 적고 아까 처럼 정점이 작은놈이 우선적으로 루트가 되는 경우라면?

너무 비효율적이다! 크기가 큰 트리에 작은 트리를 합치는게 합당하다.

앞서 제시한 두가지 문제를 어떻게 해결할 수 있을까?

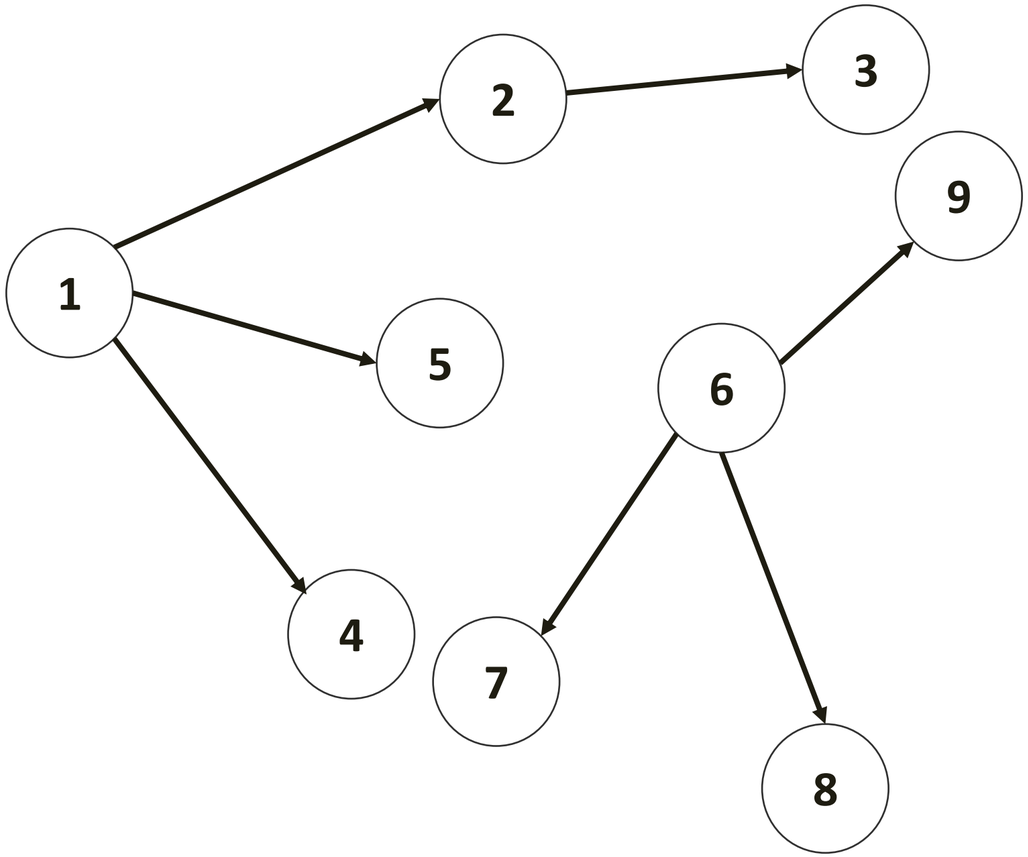
경로 압축과 랭크 기반 합치기





정점	1	2	3	4	5	6	7	8	9
랭크	3	1	0	0	0	3	0	0	0

정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	6	6	6

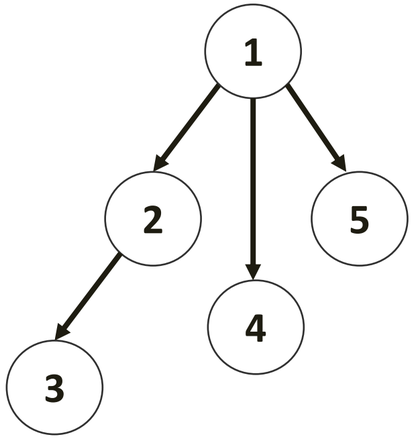


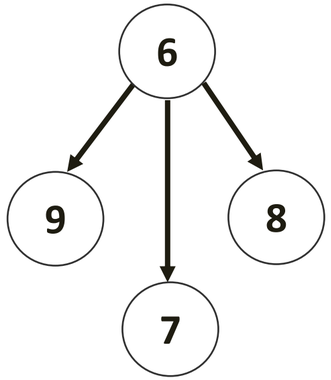
아까 본 그래프와 분리 집합의 트리를 재활용 하겠다.

각 정점의 부모를 저장할 배열은 이미 자신의 부모를 가지고 있다.

또한 랭크도 아까 기준에 맞춰서 초기화를 해주겠다.

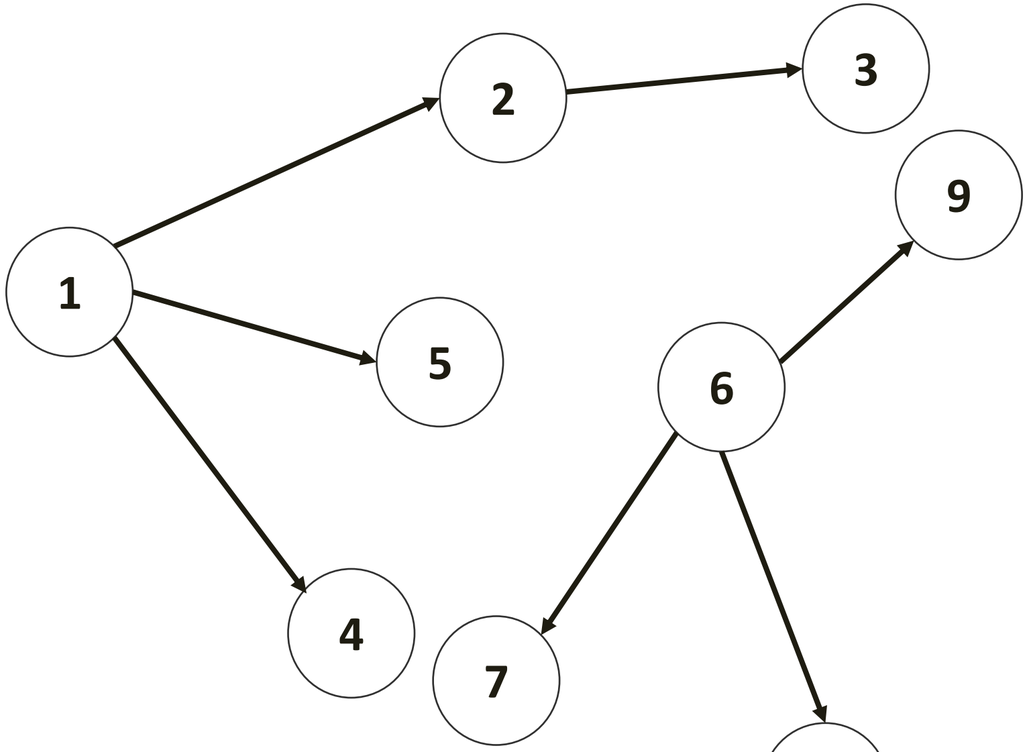
경로 압축과 랭크 기반 합치기





정점	1	2	3	4	5	6	7	8	9
랭크	3	1	0	0	0	3	0	0	0

정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	6	6	6

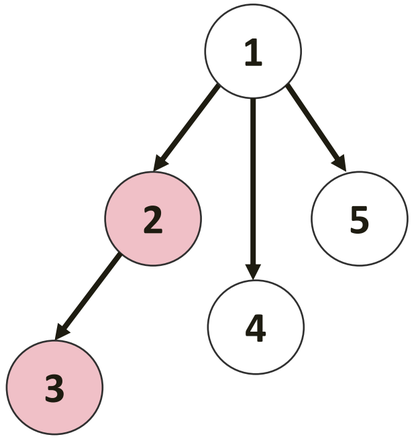


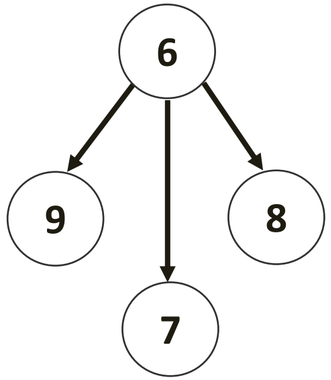
아까 문제 상황에서는 3번의 부모가 2번이기 때문에 같은 컴포넌트에 속하는지 한번에 알 수 없었다.

그러면 이런 상황을 어떻게 해결할까?

바로 **경로 압축**을 통해 해결할 수 있다.

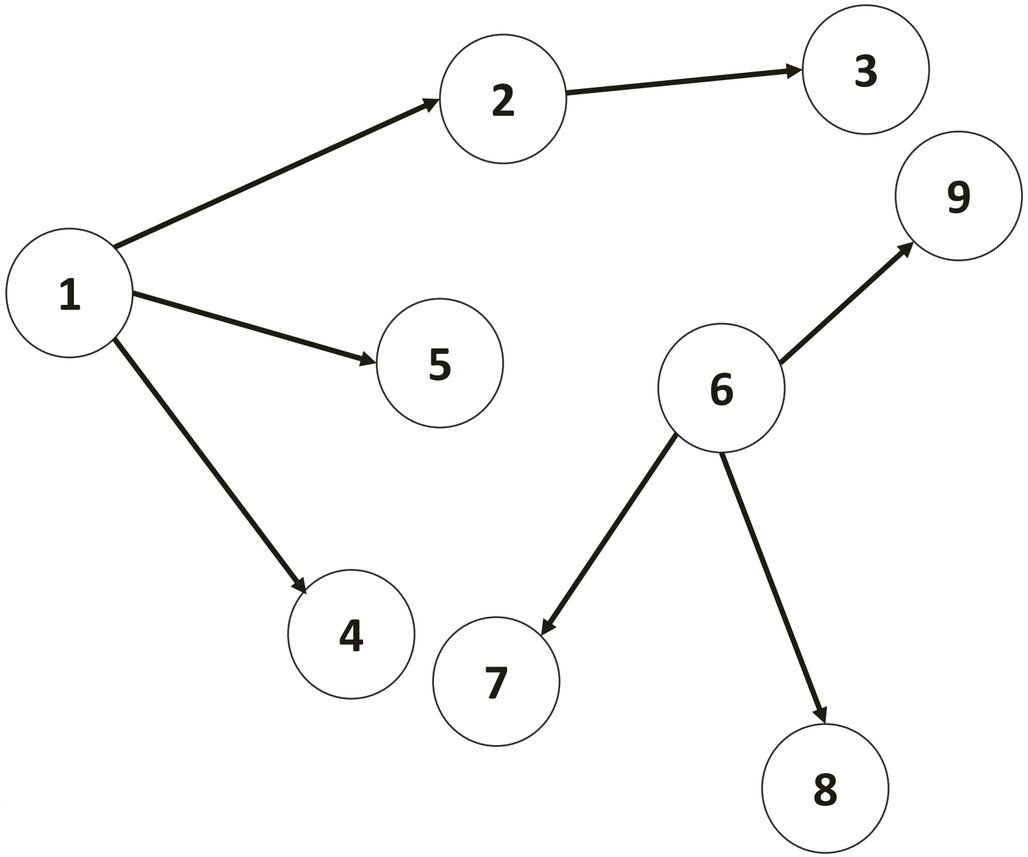
경로 압축과 랭크 기반 합치기





정점	1	2	3	4	5	6	7	8	9
랭크	3	1	0	0	0	3	0	0	0

정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	6	6	6

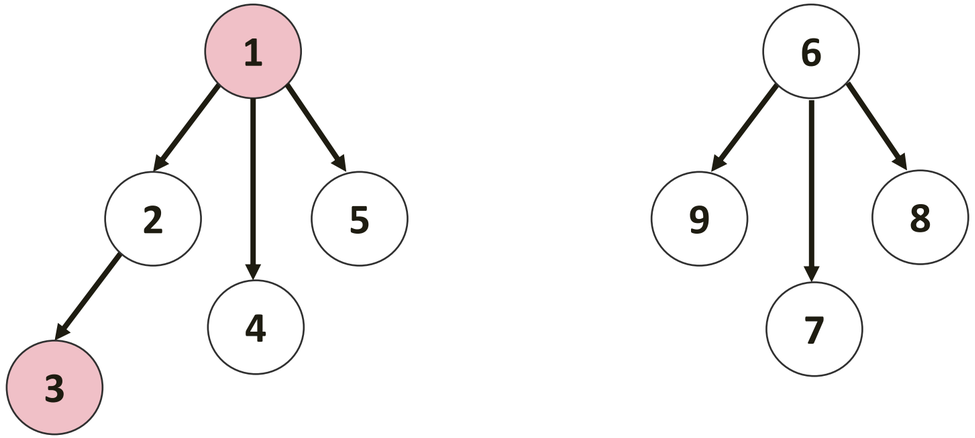


현재 3번 정점이 기준이며 그의 부모인 2번 정점이 연결 되어있다.

이때, 2번 정점은 루트 노드가 아니다.

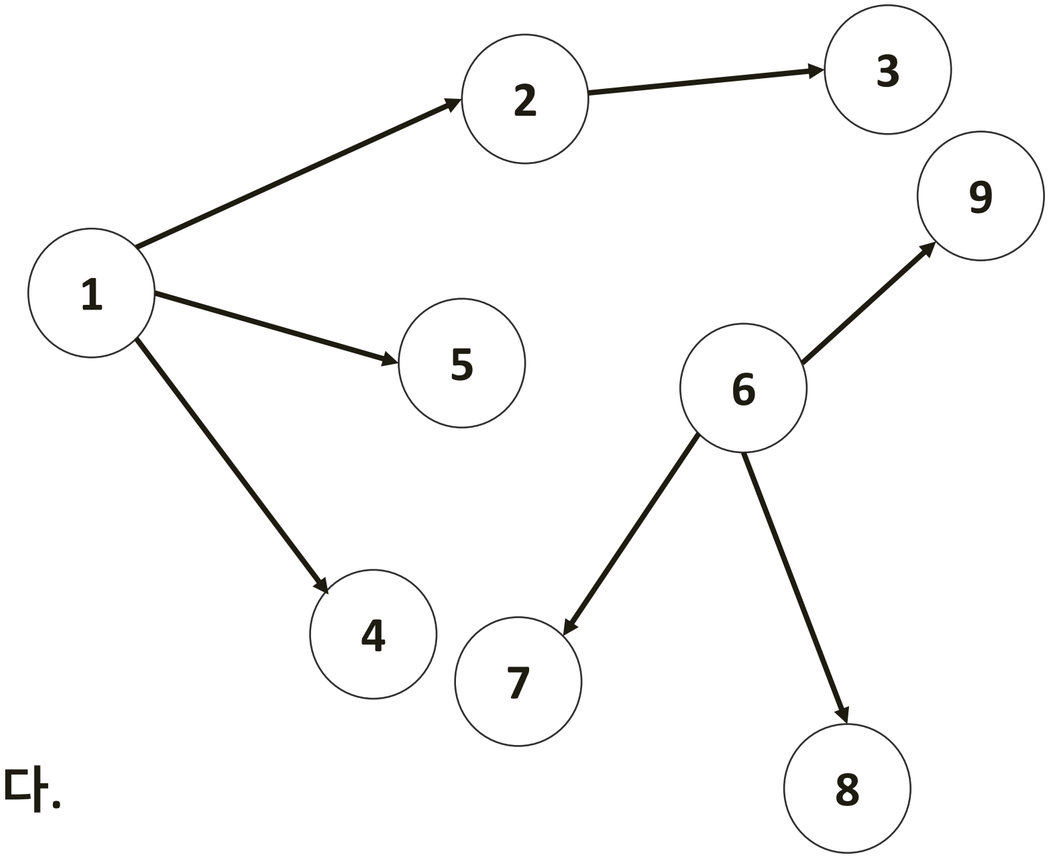
고로 2번 정점에서 다시 한 칸 올라가 준다.

경로 압축과 랭크 기반 합치기



정점	1	2	3	4	5	6	7	8	9
랭크	3	1	0	0	0	3	0	0	0

정점	1	2	3	4	5	6	7	8	9
부모	1	1	2	1	1	6	6	6	6

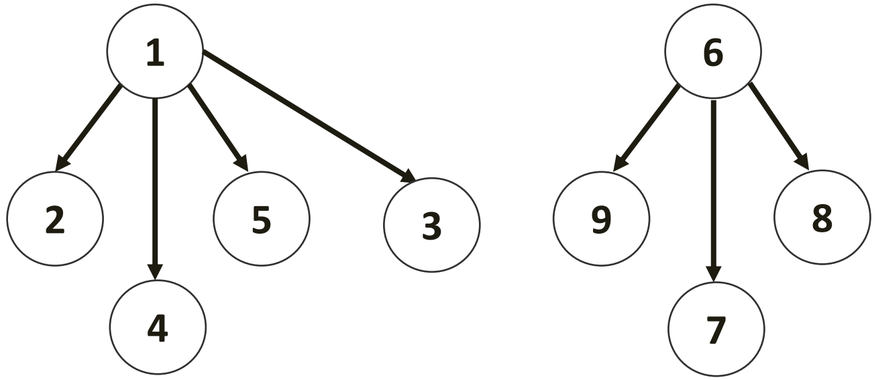


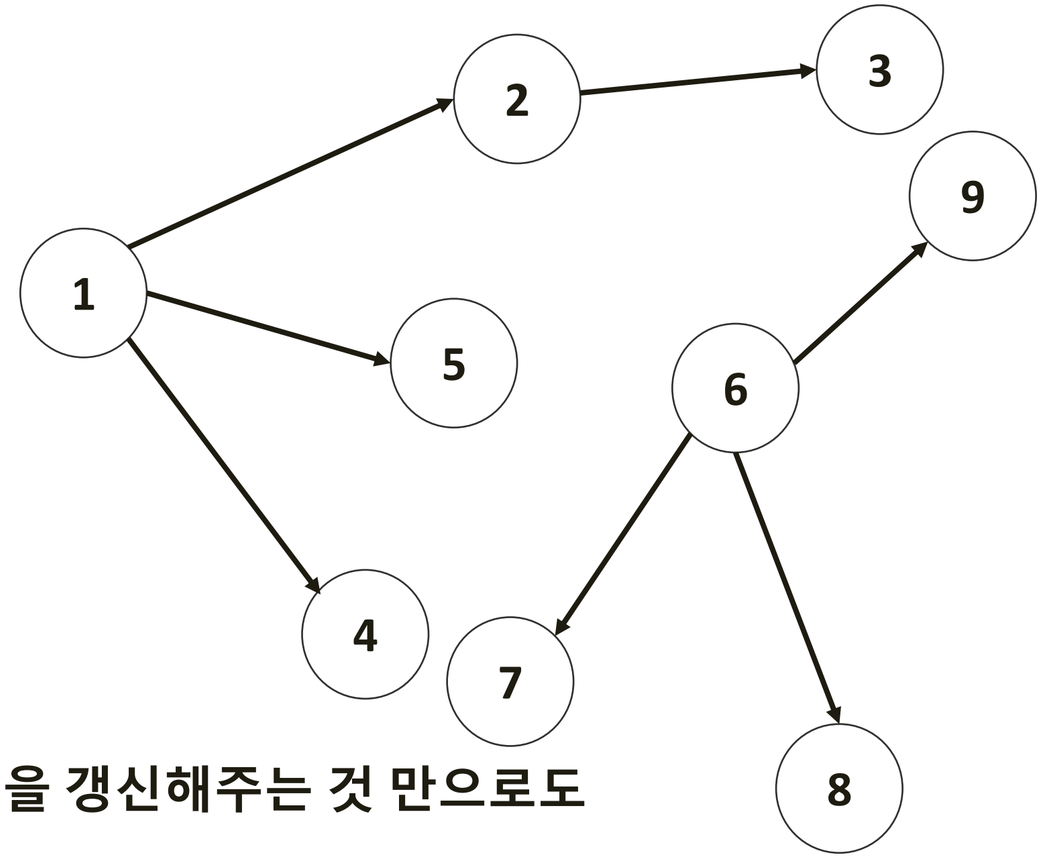
현재 3번 정점이 기준이며 1번 정점과 간접적으로 연결된 상태이다.

이때, 1번 정점은 루트이다.

고로 3번 정점의 부모를 1번이 되도록 고쳐 주어야한다.

경로 압축과 랭크 기반 합치기





정점	1	2	3	4	5	6	7	8	9
랭크	4	0	0	0	0	3	0	0	0

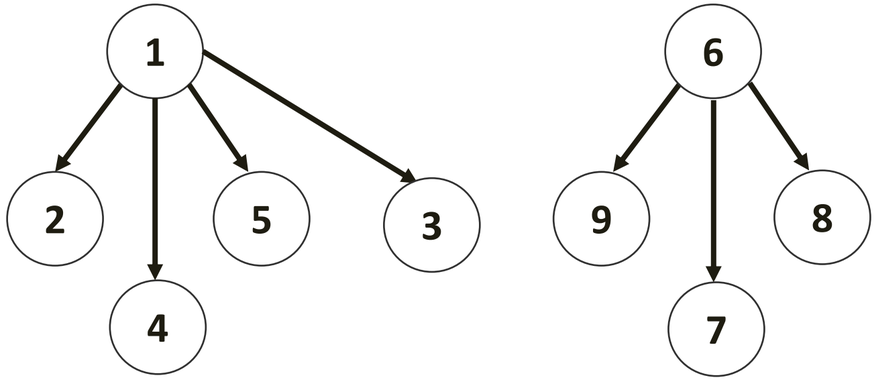
정점	1	2	3	4	5	6	7	8	9
부모	1	1	1	1	1	6	6	6	6

포인터로 트리를 구현한 것이 아니기 때문에 단지 부모 배열의 값을 갱신해주는 것 만으로도 트리 구조를 손쉽게 바꿀 수 있다.

경로 압축이란 트리 구조를 손쉽게 바꾸는 구조이다.

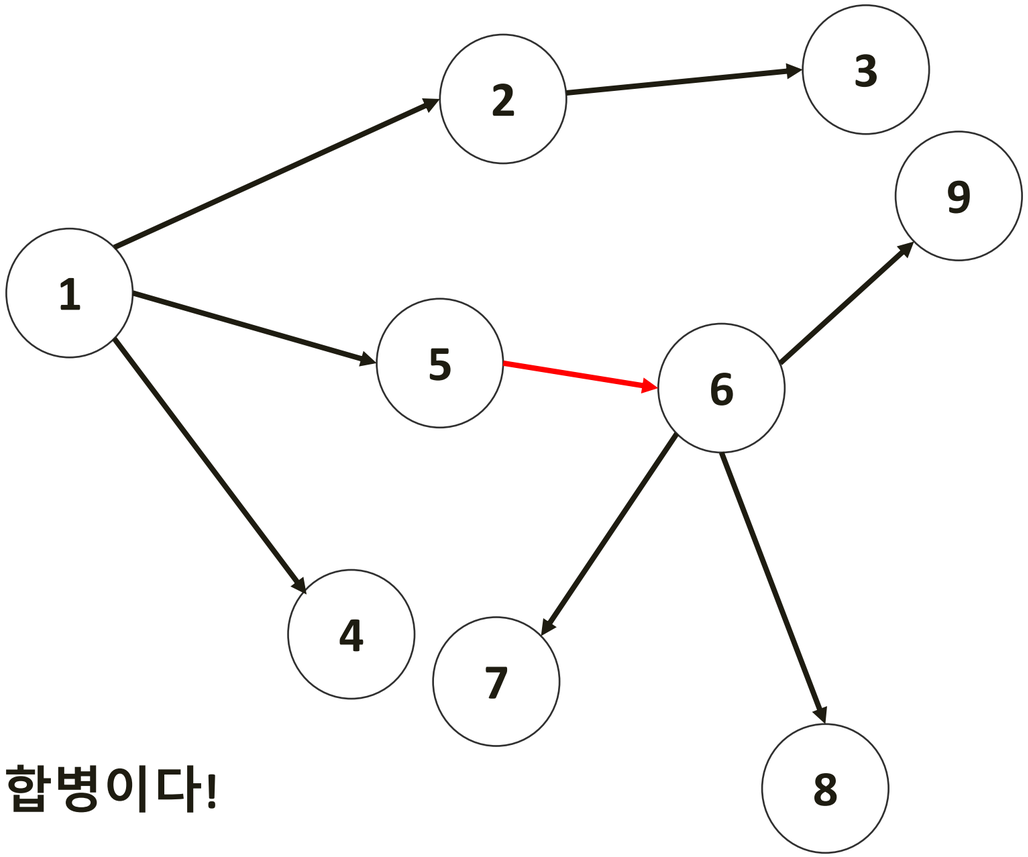
즉 루트 노드 밑에 하위 노드들이 직접적으로 연결된 자식 노드의 형태를 가지게끔 바꾸는 테크닉이다.

경로 압축과 랭크 기반 합치기



정점	1	2	3	4	5	6	7	8	9
랭크	4	0	0	0	0	3	0	0	0

정점	1	2	3	4	5	6	7	8	9
부모	1	1	1	1	1	6	6	6	6

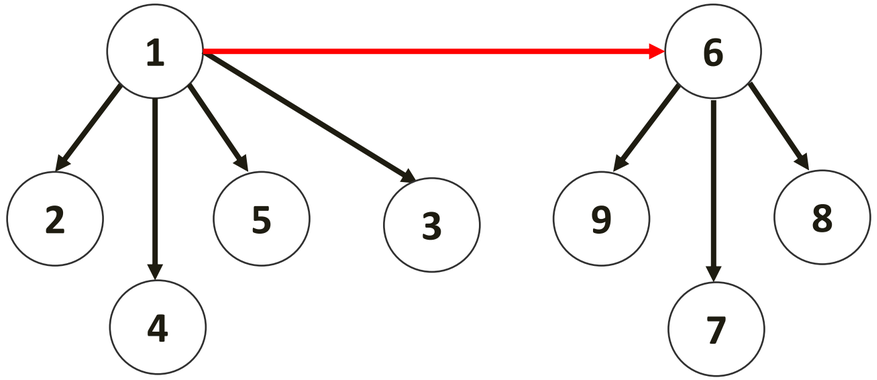


여기서 5번 정점과 6번 정점이 연결된다고 가정하자. 이제 트리의 합병이다!

먼저 부모의 랭크를 먼저 비교한다.

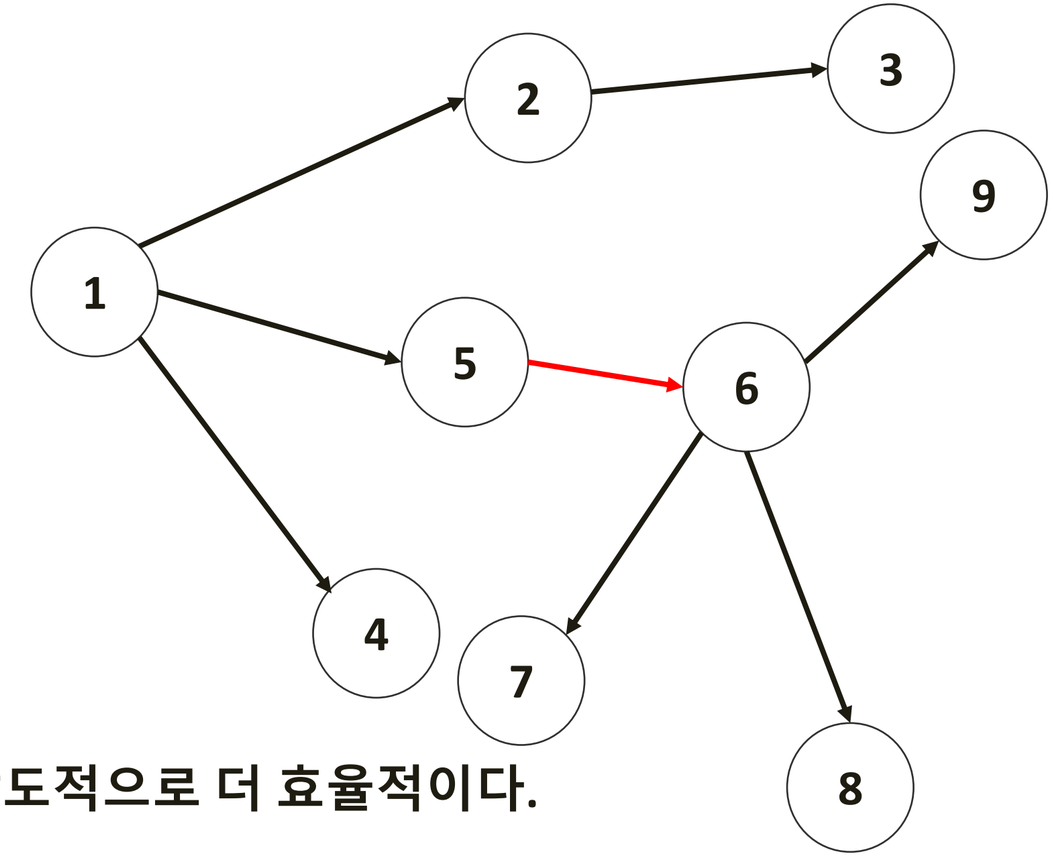
5번 정점의 부모인 1번은 랭크가 4, 6번 정점의 부모는 자기 자신이므로 랭크가 3이다.

경로 압축과 랭크 기반 합치기



정점	1	2	3	4	5	6	7	8	9
랭크	4	0	0	0	0	3	0	0	0

정점	1	2	3	4	5	6	7	8	9
부모	1	1	1	1	1	1	6	6	6

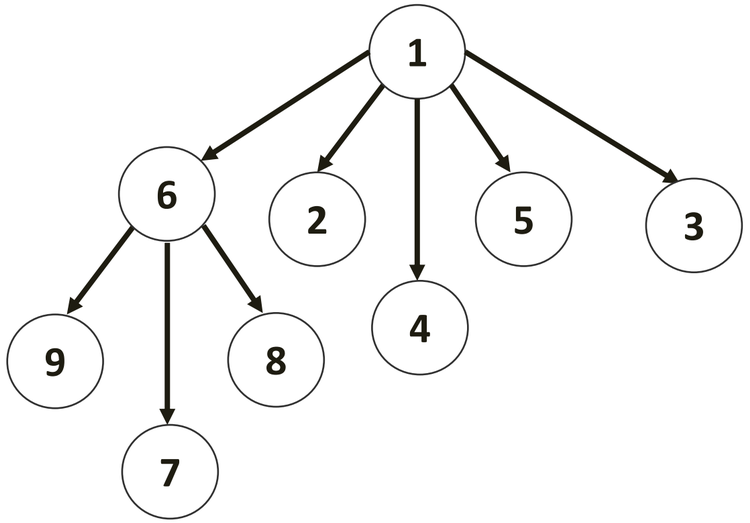


당연하게도 랭크가 큰 트리에 작은 트리를 합병하는게 시간 복잡도적으로 더 효율적이다.

고로 6번 노드를 1번 노드에(5번 이랑 합병이지만 경로 압축에 의해 루트 바로 밑의 자식으로 들어감) 연결 시켜준다. 즉 경로 압축에서의 합병은 노드와 노드가 아닌, 노드의 부모와 다른 노드의 부모의 합병이다.

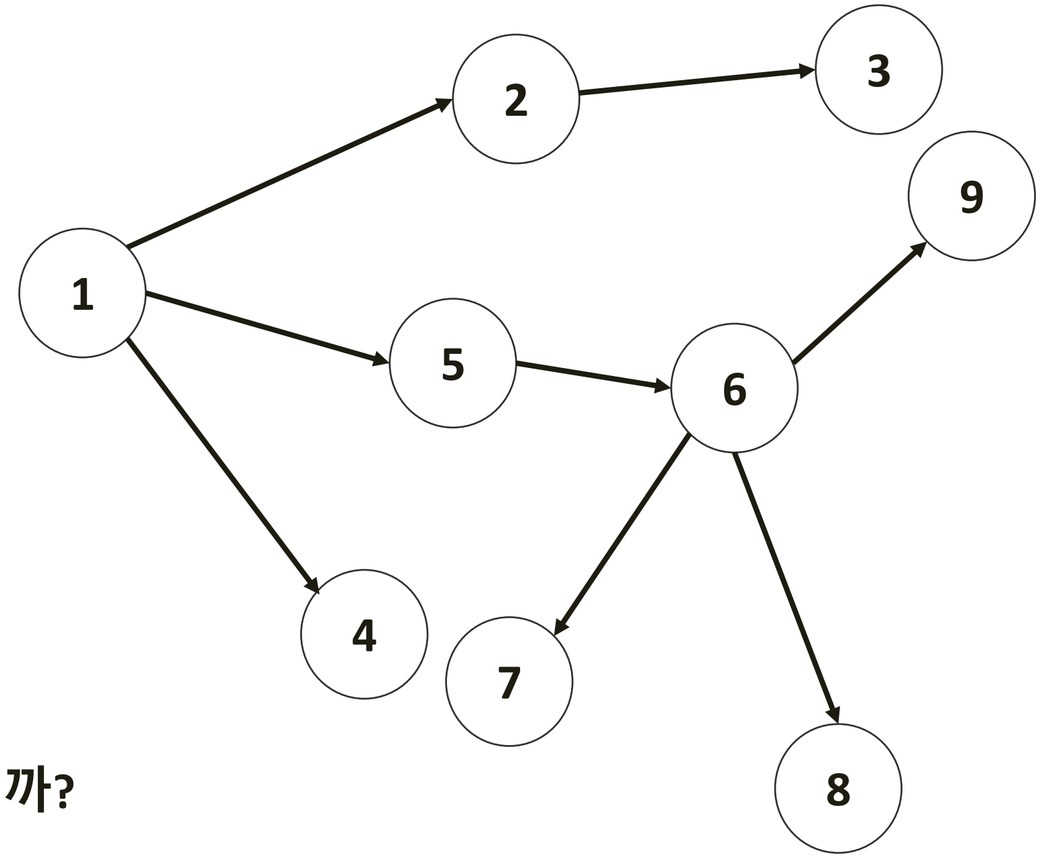
이렇게 모든 노드가 한 컴포넌트에 있다는 것을 빠르게 알 수 있다!

경로 압축과 랭크 기반 합치기



정점	1	2	3	4	5	6	7	8	9
랭크	4	0	0	0	0	3	0	0	0

정점	1	2	3	4	5	6	7	8	9
부모	1	1	1	1	1	1	6	6	6

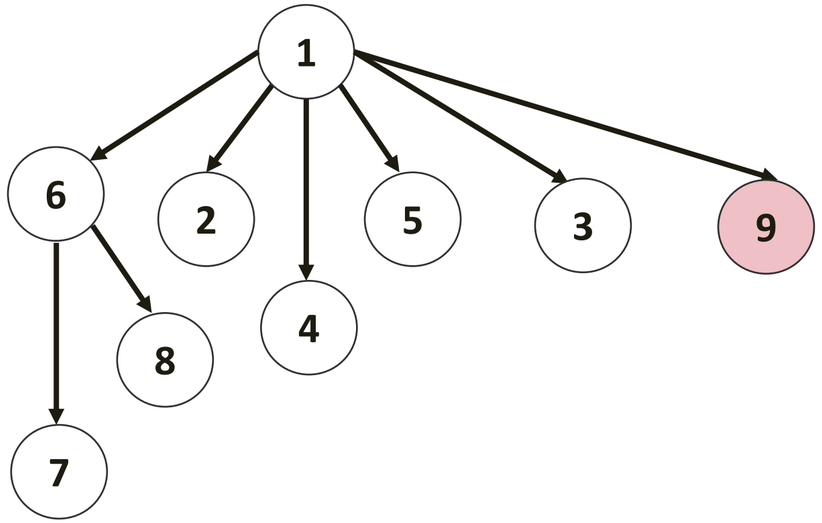


하지만, 7번 8번 9번 노드의 부모는 1번 노드가 아니다. 왜 그런 걸까?

경로 압축의 특성상 부모와 부모만 연결을 하기 때문에 그 하위 노드들의 값은 다시 갱신을 해주어야 한다.

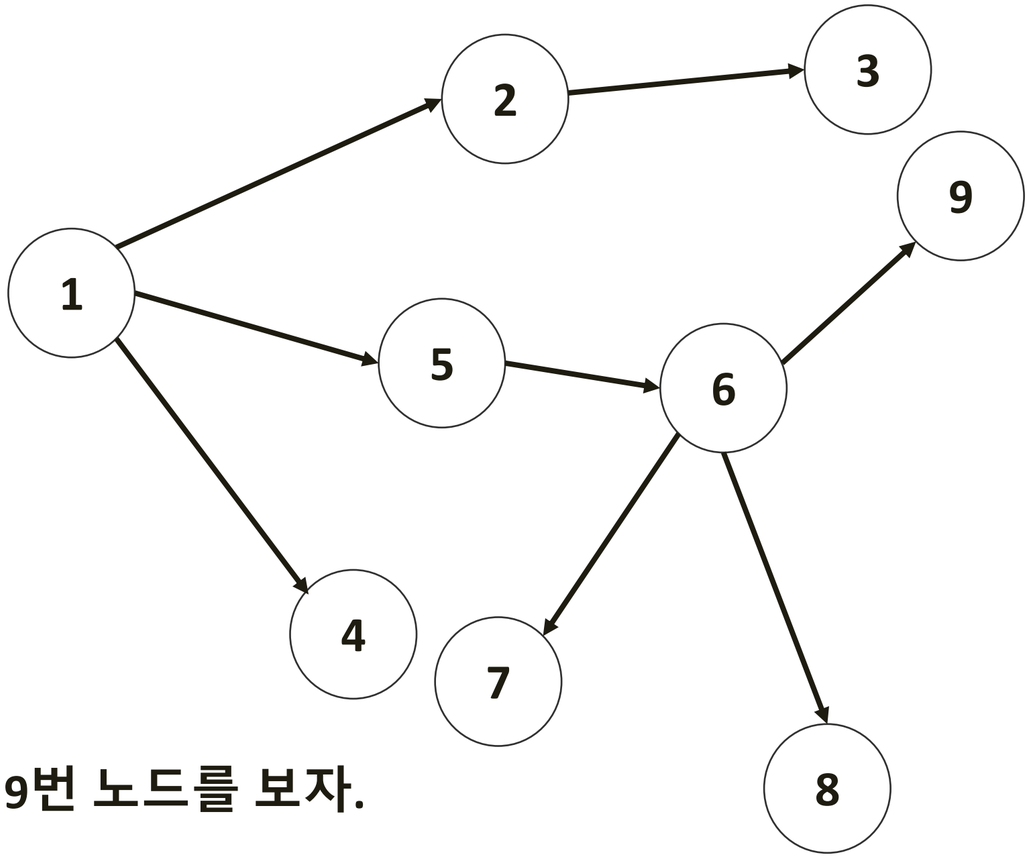
하지만 지금 당장 할 필요가 없다! 경로 압축을 통해 자연적으로 루트 하나에 자식이 주렁주렁 달린 형태가 된다.

경로 압축과 랭크 기반 합치기



정점	1	2	3	4	5	6	7	8	9
랭크	4	0	0	0	0	3	0	0	0

정점	1	2	3	4	5	6	7	8	9
부모	1	1	1	1	1	1	6	6	1

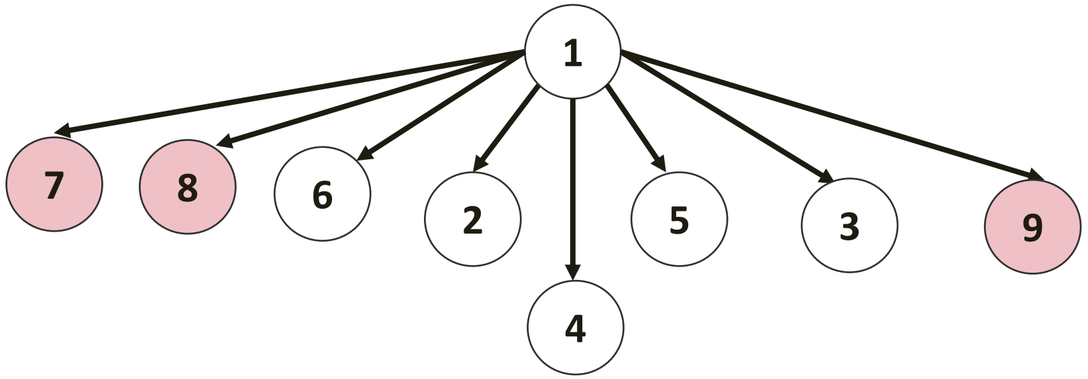


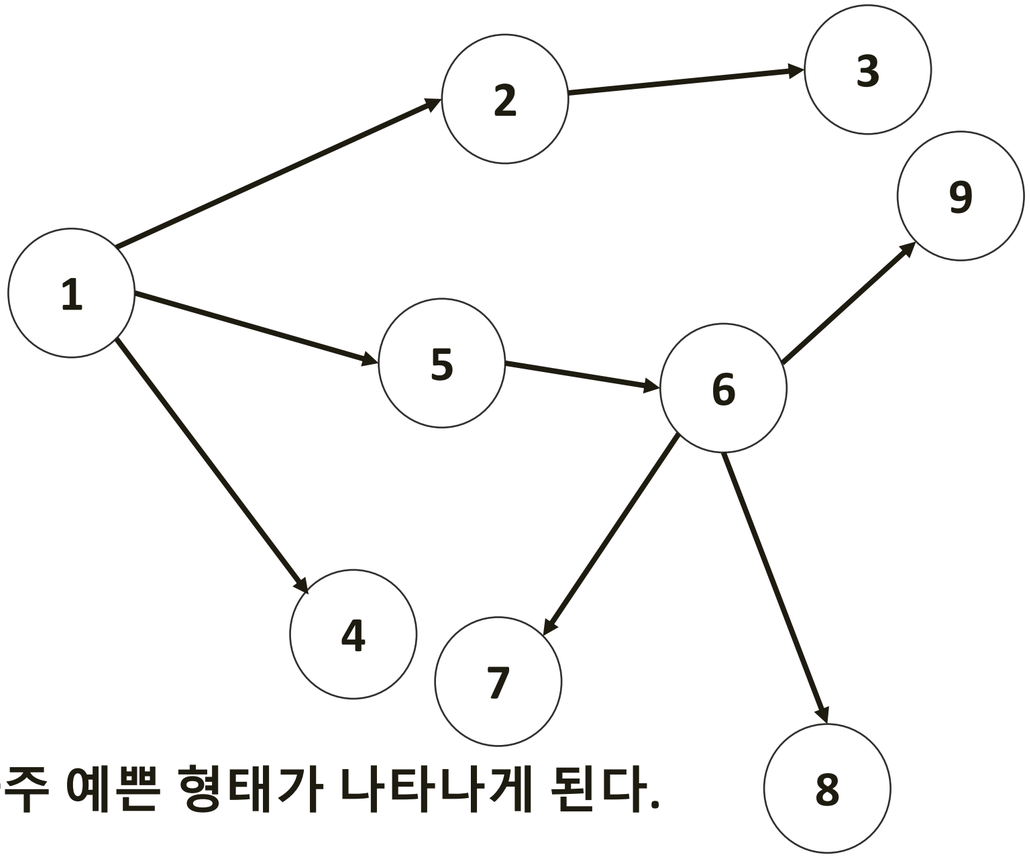
하위 노드에 대해서 어떻게 구조가 바뀌는지 예시를 보겠다. 먼저 9번 노드를 보자.

9번의 부모는 6번 노드 지만 루트가 아니다. 계속 부모의 부모의 부모.... 를 타고 넘어가면 루트인 1번 노드가 보인다.

단지 부모배열을 1로 고쳐 주기만 하면 된다! 그러면 분리 집합 트리가 위와 같이 바뀐다.

경로 압축과 랭크 기반 합치기





정점	1	2	3	4	5	6	7	8	9
랭크	4	0	0	0	0	3	0	0	0

정점	1	2	3	4	5	6	7	8	9
부모	1	1	1	1	1	1	1	1	1

나머지 노드들도 차례 대로 경로 압축을 진행 해주면 위와 같은 아주 예쁜 형태가 나타나게 된다.

그러면 노드를 병합할 때 마다 하위 노드를 모두 경로압축을 시켜 주어야 할까?

정답은 No이다. 나중에 저 노드의 직접적인 부모가 필요할 때만 경로 압축을 시켜주면 된다! 이런 기법은 세그트리의 지연전파 기법에서도 볼 수 있으며, 컴퓨터 구조의 쓰기 정책에서도 나오는 테크닉이다.