최소 비용 신장 트리 (MST : minimum spanning tree)

신장트리

* 그래프 내의 모든 정점을 포함하는 트리
* n개의 정점 (n-1)개의 간선

최소 비용 신장트리

* 모든 정점들을 가장 적은 수의 간선과 비용으로 연결

<Kruskal 의 MST 알고리즘>

* 조건 : 최소 비용 신장 트리가 간선의 가중치의 합의 최소

반드시 (n-1)개의 간선 사용, 사이클을 포함하면 안됨

* 그리디 알고리즘 사용

1. 그래프의 e개의 간선들을 가중치의 오름차순으로 정렬
2. 정렬된 간선 리스트에서 순서대로 사이클을 형성하지 않는 간선을 선택하여 현재의 최소비용 신장트리 집합에 추가

* 간선을 이미 선택된 간선들의 집합에 추가할 때 사이클을 생성하는지 확인해야 함 -> union-find 알고리즘 사용

<union-find 연산>

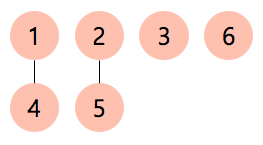
* union(x, y) 연산은 원소 x와y가 속해 있는 집합을 입력으로 받아 2개의 집합의 합집합을 만든다
* 집합을 구현하는 가장 효율적인 방법은 트리 형태를 사용
* find(x) 연산은 원소 x가 속해 있는 집합을 반환

ex)

S={1, 2, 3, 4, 5, 6} 집합의 원소를 하나씩 분리하여 독자적인 집합으로 만든다

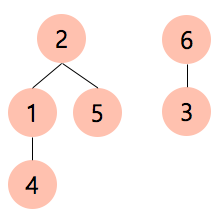
1. {1}, {2}, {3}, {4}, {5}, {6}
2. union(1, 4) 와 union(5, 2) 를 하면

{1, 4}, {5, 2}, {3}, {6}



1. union(4, 5) 와 union(3,6) 을 하면

{1, 4, 5, 2}, {3, 6}



1. find(4) 를 한다면 루트인 정점 2가 집합의 대표로 반환

* find연산에 소요되는 시간을 줄이려면 특정 정점과 루트와 의 거리가 짧아야 함 -> 모든 정점을 루트 노드 바로 밑에 놓는다
* union 연산 시 정점의 개수가 적은 트리의 루트가 큰 트리의 루트를 가리키도록

Kruskal의 알고리즘의 시간 복잡도는 간선들을 정렬하는 시간에 좌우

* 간선 n개를 퀵정렬과 같은 효율적인 알고리즘으로 정렬한다면 O(nlogn)

<Prime의 MST 알고리즘>

* 시작 정점에서부터 출발하여 신장 트리 집합을 단계적으로 확장해 나가는 방법
* 앞 단계에서 만들어진 신장 트리 집합에 인접한 정점들 중에서 최소 간선으로 연결 된 정점을 선택하여 트리를 확장
* 트리가 n-1개의 간선을 가질 때 까지 반복
* 시작 단계에서는 시작 정점만이 신장 트리 집합에 포함
* 주 반복문이 정점의 수 n 만큼 반복하고, 내부 반복문이 n번 반복하므로 시간복잡도 : O(n^2)
* Kruskal의 알고리즘
* 간선 선택 기반
* 이전 단계에서 만들어진 신장 트리와는 상관 없이 무조건 최소 간선만을 선택
* Prim의 알고리즘
* 정점 선택을 기반
* 이전 단계에서 만들어진 신장 트리를 확장하는 방식