알고리즘 0401

BFS

* 연결상태, 연결되지 않은 컴포넌트 확인가능
* 트리인지 아닌지 확인가능
* 무작위로 트리를 탐색하기 때문에, 비용 문제를 생각하지 않는다.

Spanning Subgraph(신장 그래프)

* 모든 정점을 활용해야한다.
* Tree는 연결되어야 한다.

Tree

* 낭비가 가장 적은 그래프

Spanning Tree(신장 트리)

* BFS를 활용하여 확인 가능
* 신장 트리는 1개만 존재한다.

완전 그래프

* BFS가 필요없고, 신장 트리가 여러 개 존재한다.
* Prufer Code를 통해 트리가 몇 개(10^n-2) 있는지 계산할 수 있다.
* 어떤 트리를 선택하든지 똑같다.

Minimal Spanning Trees

* 트리마다 경로가 다르다.
* 최적 경로를 선택하고자 한다.
* 간선을 추가하면, 신장 트리가 여러 개 생긴다.
* 신장 트리가 여러 개인 경우는 간선에 문제가 생겨도 연결이 끊기지 않는다.
* 간선엔 가중치가 있다.
* 비용 문제를 해결하기 위한 알고리즘

Boruvka’s Algorithm (1926)

* 결합 간선 중에 가장 저렴한 경로를 선택한다.
* 가중치가 제일 저렴한 간선의 방향을 표시
* 방향을 연결하면, 트리가 여러 개 생긴다.
* 여러 트리가 생긴 후에 다른 트리와 연결되는 간선 중에 제일 저렴한 간선을 선택한다. (회로가 되지 않게 하는 것이 중요하다.)
* 최소 신장 트리가 된다.

Anlysis of Boruvka’s Algorithm(1)

* 왜 트리만 만드는가?
* 여러 트리의 간선 값이 다 같으면 회로가 될 수 있다.
* 여러 트리의 간선 값이 다르면 가장 비싼 간선은 선택되지 않기 때문에 회로가 되지 않는다.

Dealing With Duplicate Edge Values

* 회로를 만드는 상황을 피하려면 Top10(a, b, c, d, ….)이 필요하다.
* 간선의 값이 같은 경우는 순위를 선택하면 된다.
* 간선의 값을 기준으로 순서를 정한다.
* 간선의 값이 같더라도 정해둔 선택을 정하면 순환 선택을 피할 수 있다.

Uniqueness

* 증명은 복잡해서 pass
* 간선의 가중치가 다른 경우는 유일한 최소 트리는 1개만 존재한다.

Prim’s Algorithem (1930 Jarnik, 1957 Prim)

* Mini Tree, 주변 간선 1개 추가하고, 주변 간선 1개 추가한다…
* 가장 저렴한 간선을 신장 트리가 만들어질 때까지 계속 추가한다.
* 트리 기준으로 주변의 전체 가중치 중에 가장 작은 간선을 선택해서 한다.
* 단, 회로가 되면 안된다.

Exercise

* 같은 가중치가 있기 때문에 순위를 먼저 정해준다.
* 가격은 연결된 간선 가중치의 합이다.
* 순서에 따라 트리가 달라질 수 있지만, 전체 가중치의 합은 같다.

Kruskals’s Algorithm (1959)

* 가장 저렴한 가중치부터 선택하여 연결하는 알고리즘
* 회로를 피해서 연결해야한다.
* 순위를 정하고, 저렴한 가중치가 여러 개 존재해도 연결할 수 있다.