알고리즘
그래프, 순회, 최소비용트리

충남대학교
이영석

Graph 문제

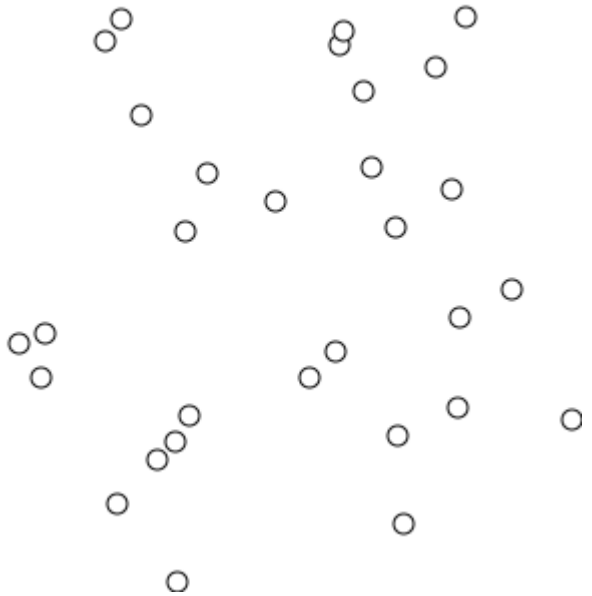
- 최소비용신장트리
  - Kruskal, Prim
  - 관련지식: union, find (집합관련)

그래프에서 모든 노드를 연결하는 트리

- 그래프: G=(V, E)
- 노드(정점, node, vertex): |V| = n
- 링크(간선, link, arc, edge): |E| = e
  - 방향은 없음
  - 링크의 가중치(weight)가 있음
- 모든 노드를 연결하는 트리
  - spanning tree: 트리의 노드 개수는? 링크 개수는?
- 그 중에서 링크의 가중치 합이 최소가 되는 트리
  - 최소 비용 신장 트리(minimum cost spanning tree: MST)

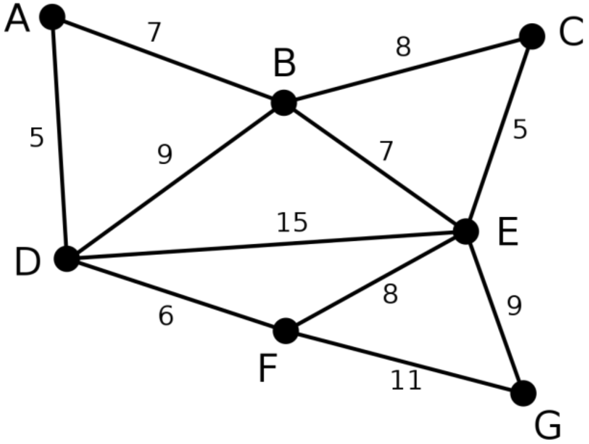
Kruskal's Algorithm

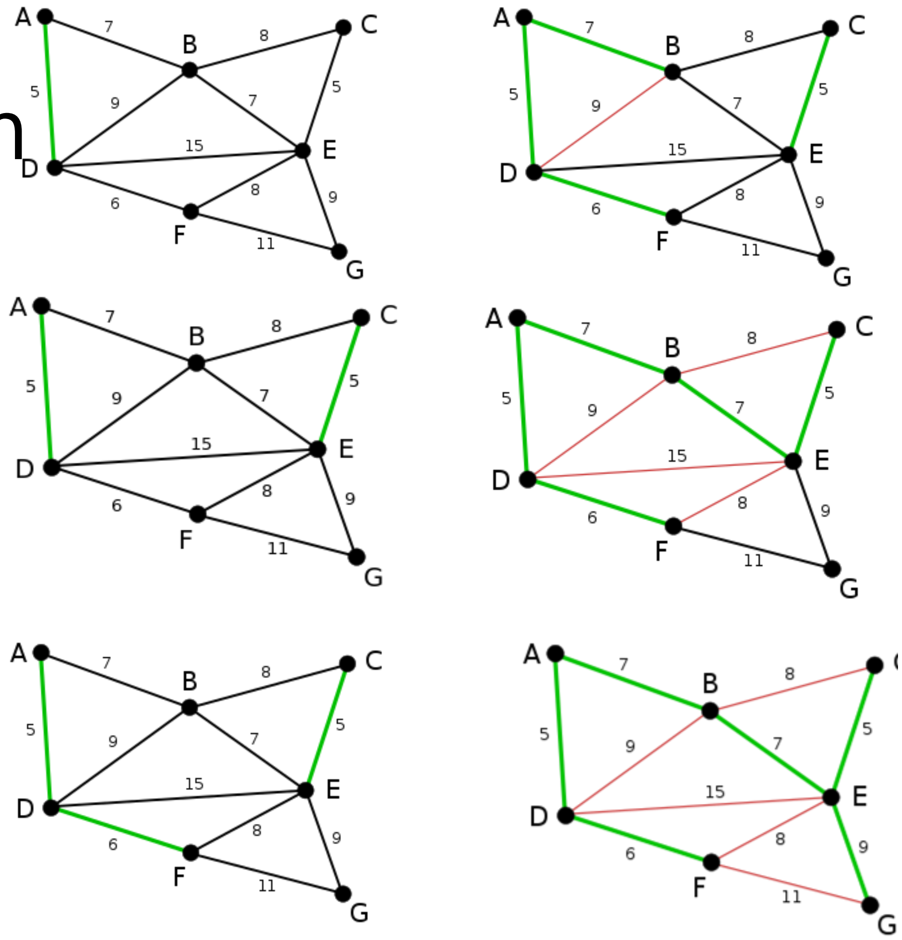
```
KRUSKAL(G):
1 A = ∅
2 foreach v ∈ G.V:
3    MAKE-SET(v)
4 foreach (u, v) in G.E ordered by weight(u,
v), increasing:
5    if FIND-SET(u) ≠ FIND-SET(v):
6       A = A ∪ {(u, v)}
7       UNION(FIND-SET(u), FIND-SET(v))
8 return A
```



$O(E \log E)$

Kruskal’s Algorithm





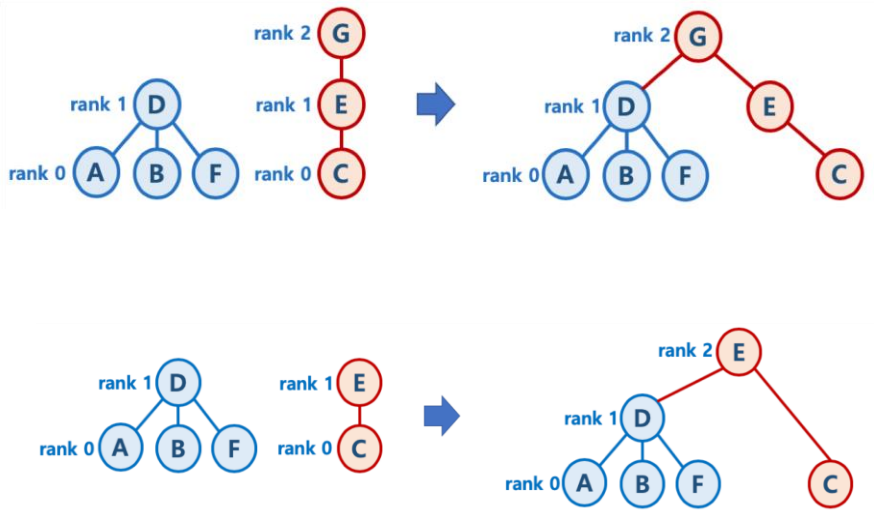
Union, Find, Disjoint Set

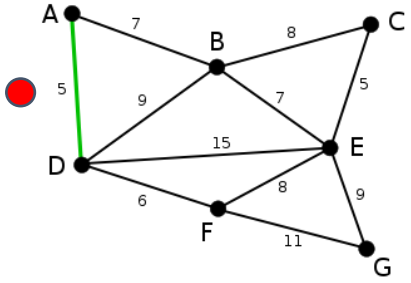
- Disjoint set
  - 분리된 집합
- Union
  - 집합 합치기
- Find
  - 원소를 집합에서 찾기
- 예
  - make-set(x)
  - find(x)
  - union(x, y)

- Union-Find 주의사항
  - linked list와 같이 O(n) 복잡도가 나오지 않도록 Set을 유지해야함

Union-by-rank

- Union-by-rank
  - 높이를 rank로 기억함
  - 두 트리의 높이가 다르면
    - 높이가 작은 트리를 높이가 큰 트리에 붙임
  - 두 트리의 높이가 같은 경우
    - 한 쪽 트리의 높이를 1 증가시켜 다른 트리의 루트 노드로 함

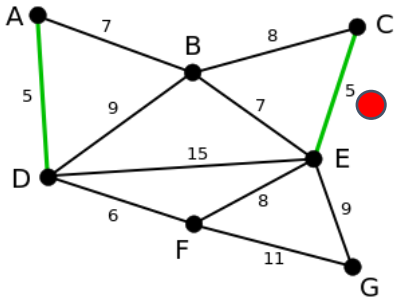




rank 1 D

↑

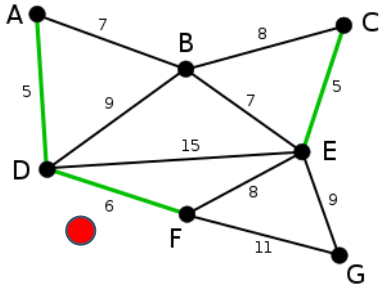
rank 0 A



rank 1 D E

↑ ↑

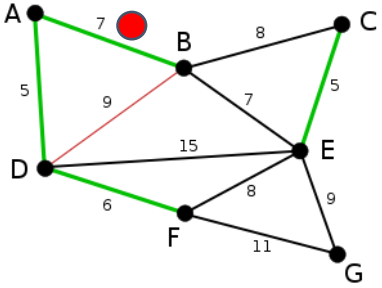
rank 0 A C



rank 1 D E

↑ ↖ ↑

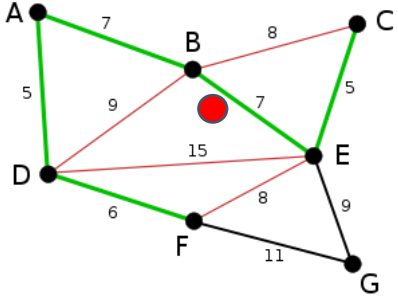
rank 0 A F C



rank 1 D E

↗ ↑ ↖ ↑

rank 0 A B F C



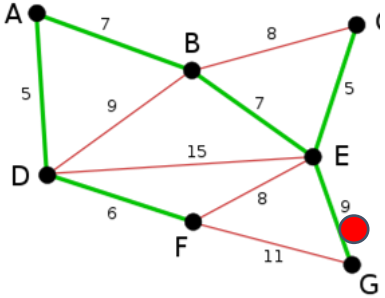
rank 2 E

↗ ↖

rank 1 D C

↗ ↑ ↖

rank 0 A B F



rank 2 E

↗ ↑ ↖

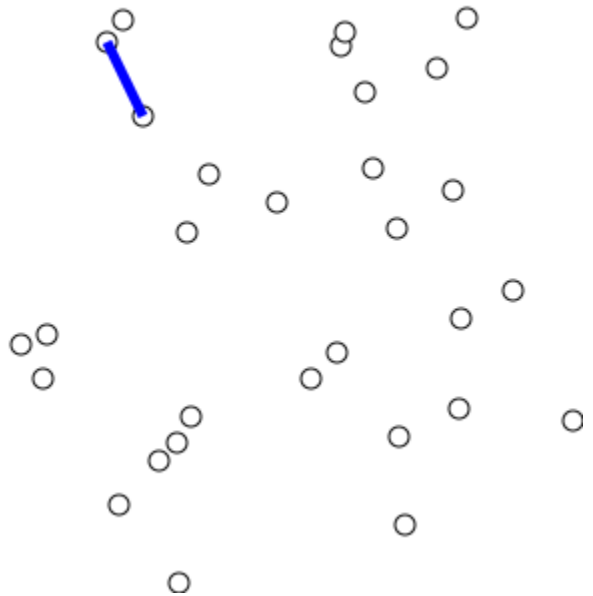
rank 1 D C G

↗ ↑ ↖

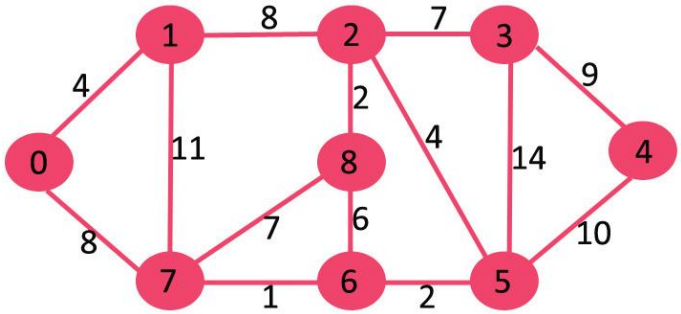
rank 0 A B F

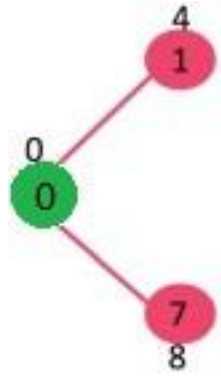
Prim's Algorithm

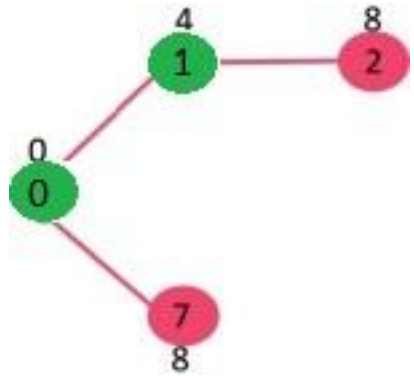
1. 그래프에서 하나의 노드를 선택하여 트리를 만든다.
2. 그래프의 모든 링크 집합을 만든다.
3. 모든 노드가 트리에 포함될 때까지
   - 트리와 연결된 링크 중에서 트리 속의 두 노드를 연결하지 않는 링크 중 최소값의 링크를 트리에 추가한다.

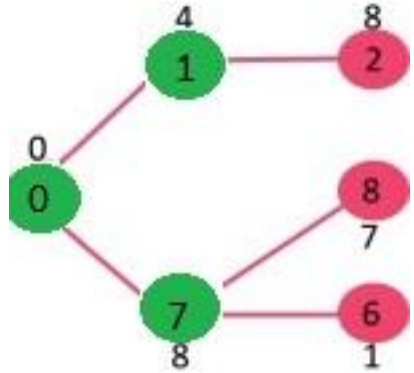


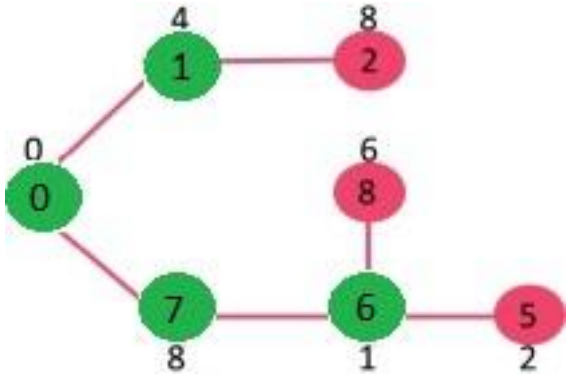
Minimum edge weight data structure	Time complexity (total)
adjacency matrix, searching	$O(\|V\|^2)$
binary heap and adjacency list	$O((\|V\| + \|E\|) \log \|V\|) = O(\|E\| \log \|V\|)$
Fibonacci heap and adjacency list	$O(\|E\| + \|V\| \log \|V\|)$

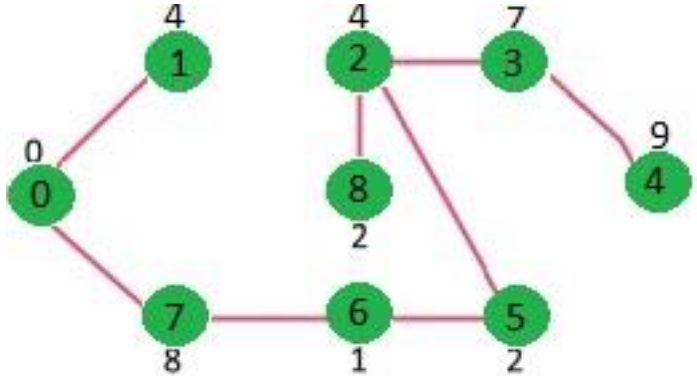

0
1
2
3
4
5
6
7
8
4
8
7
9
11
2
4
14
8
7
6
10
1
2


0
1
7
4
8


0
1
2
7
4
8
8


0
1
2
7
8
4
8
8
7
6
1


0
1
2
6
8
7
6
5
4
8
8
1
2


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

그래프 순회: 최소비용신장트리

- Kruskal's algorithm
- Prim's algorithm