CSE2010 자료구조론

Week 10: Minimum Spanning Tree

ICT융합학부 한진영

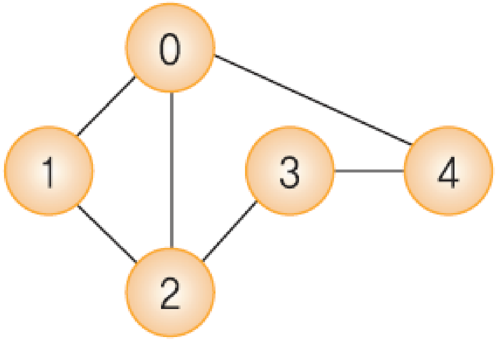
신장 트리(Spanning Tree)

- 그래프내의 모든 정점을 포함하는 트리
- 모든 정점들이 연결되어 있어야 하고 사이클을 포함해서는 안됨
- n개의 정점을 가지는 그래프의 신장트리는 n-1개의 간선을 가짐
- 최소의 링크를 사용하는 네트워크 구축 시 사용: 통신망, 도로망, 유통망 등

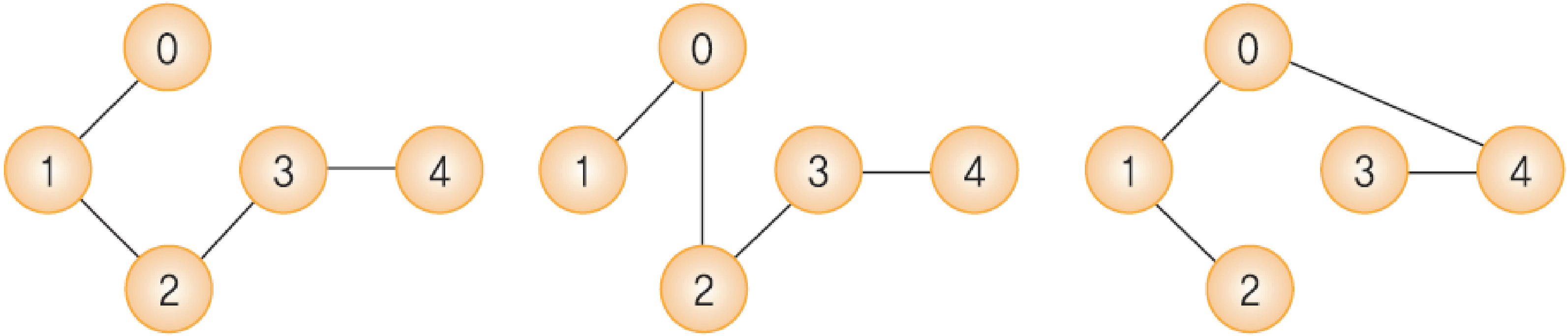
- 신장 트리 알고리즘

```
depth_first_search(v)

v를 방문되었다고 표시;
for all u ∈ (v에 인접한 정점) do
        if (u가 아직 방문되지 않았으면)
                then        (v,u)를 신장트리 간선이라고 표시;
                            depth_first_search(u);
```

신장 트리 예



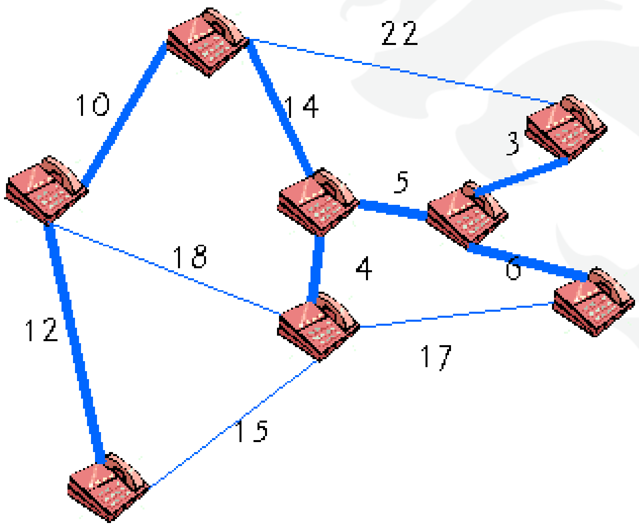
(a) 연결 그래프



(b) 신장 트리 중의 일부

최소비용 신장트리(MST: Minimum Spanning Tree)

- 네트워크에 있는 모든 정점들을 가장 적은 수의 간선과 비용으로 연결
- MST의 응용
  - 도로 건설: 도시들을 모두 연결하면서 도로의 길이를 최소가 되도록 하는 문제
  - 전기 회로: 단자들을 모두 연결하면서 전선의 길이를 가장 최소로 하는 문제
  - 통신: 전화선의 길이가 최소가 되도록 전화 케이블 망을 구성하는 문제
  - 배관: 파이프를 모두 연결하면서 파이프의 총 길이를 최소로 하는 문제



Kruskal의 MST 알고리즘(1)

- 탐욕적인 방법(greedy method)
  - 주요 알고리즘 설계 기법
  - 각 단계에서 최선의 답을 선택하는 과정을 반복함으로써 최종적인 해답에 도달
  - 탐욕적인 방법은 항상 최적의 해답을 주는지 검증 필요
  - Kruskal MST 알고리즘은 최적의 해답임이 증명됨

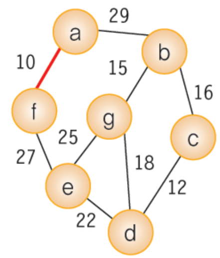
Kruskal의 MST 알고리즘(2)

- MST는 최소 비용의 간선으로 구성됨과 동시에 사이클을 포함하지 않아야 함
- 각 단계에서 사이클을 이루지 않는 최소 비용 간선 선택
  - 그래프의 간선들을 가중치의 오름차순으로 정렬
  - 정렬된 간선 중에서 사이클을 형성하지 않는 간선을 현재의 MST 집합에 추가
  - 만약 사이클을 형성하면 그 간선은 제외

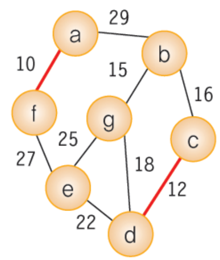
```
// 최소비용 스패닝트리를 구하는 Kruskal의 알고리즘
// 입력: 가중치 그래프 G = (V, E), n은 노드의 개수
// 출력: E_T, 최소비용 신장 트리를 이루는 간선들의 집합
kruskal(G)

    E를 w(e_1) ≤ ··· ≤ w(e_e) 가 되도록 정렬한다.
    E_T ← Φ; ecounter ← 0
    k ← 0
    while ecounter < (n − 1) do
        k ← k + 1
        if E_T ∪ {e_k} 가 사이클을 포함하지 않으면
            then E_T ← E_T ∪ {e_k}; ecounter ← ecounter + 1
    return E_T
```

Kruskal의 MST 알고리즘 예

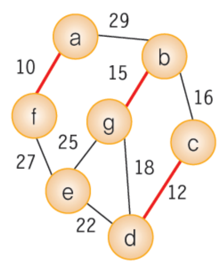
af	cd	bg	bc	dg	de	eg	ef	ab
10	12	15	16	18	22	25	27	29



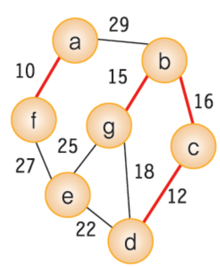
af	**cd**	bg	bc	dg	de	eg	ef	ab
10	12	15	16	18	22	25	27	29



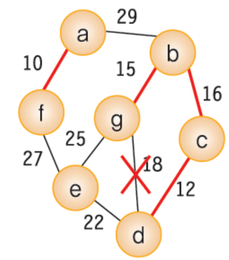
af	cd	**bg**	bc	dg	de	eg	ef	ab
10	12	15	16	18	22	25	27	29



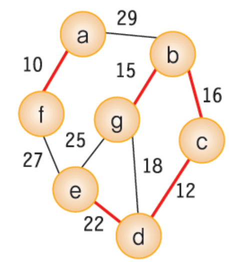
af	cd	bg	**bc**	dg	de	eg	ef	ab
10	12	15	16	18	22	25	27	29



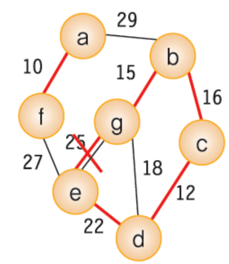
af	cd	bg	bc	**dg**	de	eg	ef	ab
10	12	15	16	18	22	25	27	29



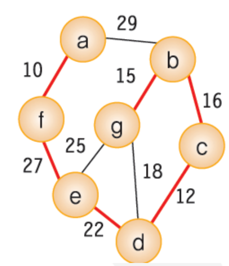
af	cd	bg	bc	dg	**de**	eg	ef	ab
10	12	15	16	18	22	25	27	29



af	cd	bg	bc	dg	de	**eg**	ef	ab
10	12	15	16	18	22	25	27	29



af	cd	bg	bc	dg	de	eg	**ef**	ab
10	12	15	16	18	22	25	27	29



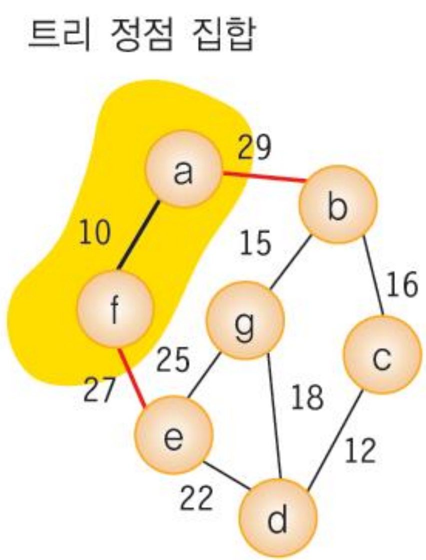
Kruskal의 MST 알고리즘 복잡도

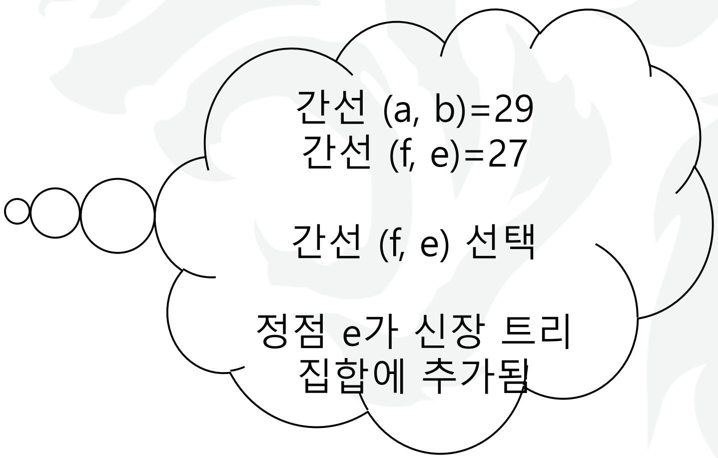
- Kruskal 알고리즘은 대부분 간선들을 정렬하는 시간에 좌우됨
  - 사이클 테스트 등의 작업은 정렬에 비해 매우 신속하게 수행됨

- 네트워크의 간선 e개를 퀵정렬과 같은 효율적인 알고리즘으로 정렬한다면 Kruskal 알고리즘의 시간 복잡도는 O(e*log(e))

Prim의 MST 알고리즘(1)

- 시작 정점에서부터 출발하여 신장 트리 집합을 단계적으로 확장해 나감
- 시작 단계에서는 시작 정점만이 신장 트리 집합에 포함됨
- 신장 트리 집합에 인접한 정점 중에서 최저 간선으로 연결된 정점 선택하여 신장 트리 집합에 추가함
- 이 과정은 신장 트리 집합이 n-1개의 간선을 가질 때까지 반복

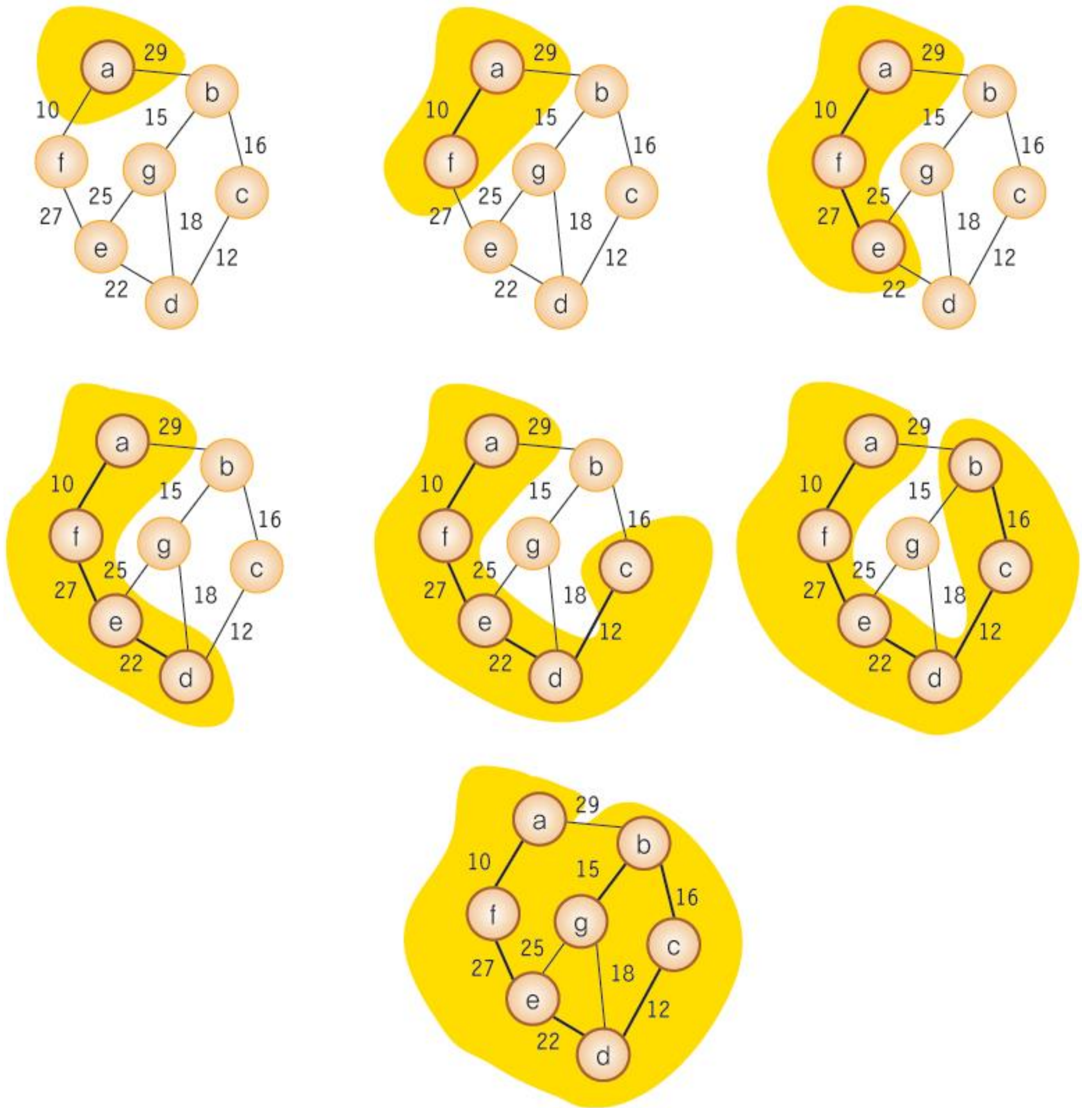




Prim의 MST 알고리즘(2)

```
// 최소 비용 신장 트리를 구하는 Prim의 알고리즘
// 입력: 네트워크 G=(V, E), s는 시작 정점
// 출력: 최소 비용 신장 트리를 이루는 정점들의 집합
Prim(G, s)

    for each u∈V do
        dist[u] ← ∞
    dist[s] ← 0
    우선 순위큐 Q에 모든 정점을 삽입(우선순위는 dist[])
    for i ← 0 to n-1 do
        u ← delete_min(Q)
        화면에 u를 출력
        for each v∈(u의 인접 정점)
            if( v∈Q and weight[u][v]<dist[v] )
                then dist[v] ← weight[u][v]
```

Prim의 MST 알고리즘 예



Prim의 MST 알고리즘 복잡도

- 주 반복문이 정점의 수 n만큼 반복하고, 내부 반복문이 n번 반복하므로 Prim의 알고리즘은 $O(n^2)$ 의 복잡도를 가짐

- 희박한 그래프
  - O(e*log(e)) 인 Kruskal의 알고리즘이 유리
- 밀집한 그래프
  - $O(n^2)$ 인 Prim의 알고리즘이 유리

Week 10: Minimum Spanning Tree