SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

APS 응용

SAMSUNG SW ACADEMY FOR YOUTH

목차

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

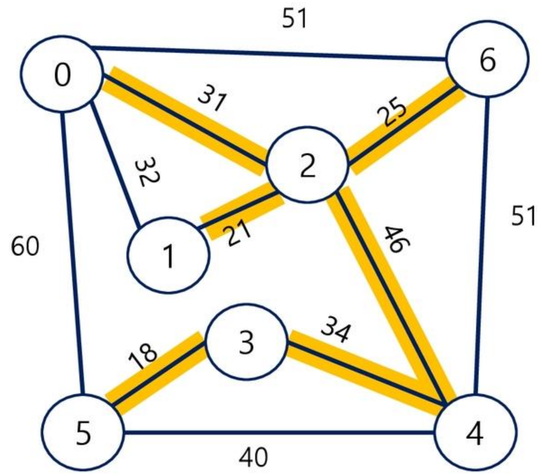
최소 신장 트리

최소 신장 트리(MST)

그래프에서 최소 비용 문제

① 모든 정점을 연결하는 간선들의 가중치의 합이 최소가 되는 트리

② 두 정점 사이의 최소 비용의 경로 찾기



신장 트리

- n개의 정점으로 이루어진 무향 그래프에서 n개의 정점과 n-1개의 간선으로 이루어진 트리

최소 신장 트리 (Minimum Spanning Tree)

- 무향 가중치 그래프에서 신장 트리를 구성하는 간선들의 가중치의 합이 최소인 신장 트리

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

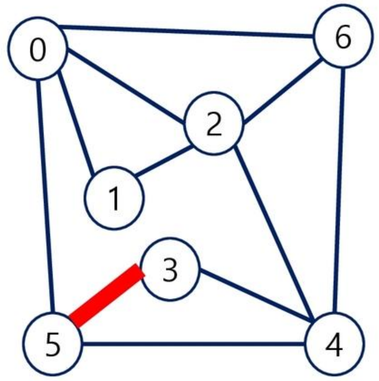
KRUSKAL 알고리즘

KRUSKAL 알고리즘

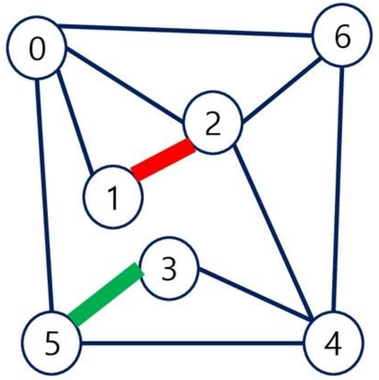
- **간선을 하나씩 선택해서 MST 를 찾는 알고리즘**
  - ① 최초, 모든 간선을 가중치에 따라 **오름차순**으로 정렬
  - ② 가중치가 가장 낮은 간선부터 선택하면서 트리를 증가시킴
    - 사이클이 존재하면 다음으로 가중치가 낮은 간선 선택
  - ③ n-1 개의 간선이 선택될 때까지 ②를 반복

KRUSKAL 알고리즘

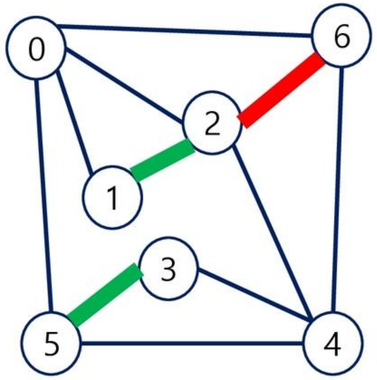
알고리즘 적용 예



5-3	**18**	5-3	18
1-2	21		
2-6	25		
0-2	31		
0-1	32		
3-4	34		
5-4	40		
2-4	46		
0-6	51		
4-6	51		
0-5	60		

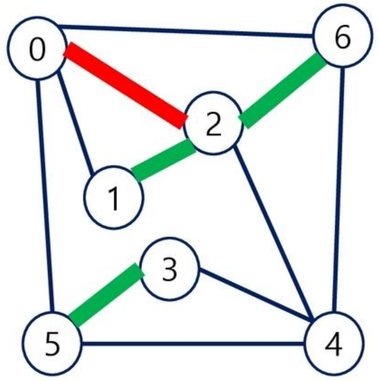


5-3	**18**	5-3	18
1-2	**21**	1-2	21
2-6	25		
0-2	31		
0-1	32		
3-4	34		
5-4	40		
2-4	46		
0-6	51		
4-6	51		
0-5	60		



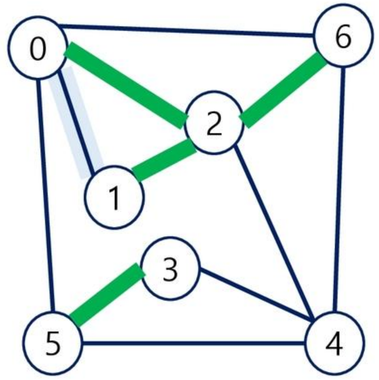
5-3	**18**	5-3	18
1-2	**21**	1-2	21
2-6	**25**	2-6	25
0-2	31		
0-1	32		
3-4	34		
5-4	40		
2-4	46		
0-6	51		
4-6	51		
0-5	60		

KRUSKAL 알고리즘



5-3	**18**
1-2	**21**
2-6	**25**
0-2	**31**
0-1	32
3-4	34
5-4	40
2-4	46
0-6	51
4-6	51
0-5	60

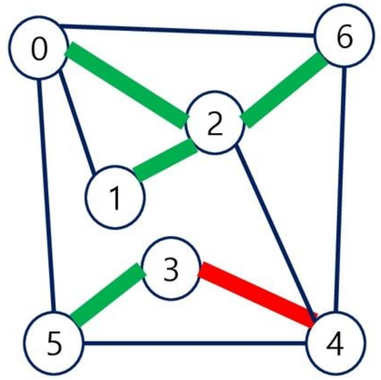
5-3	18
1-2	21
2-6	25
0-2	31



5-3	**18**
1-2	**21**
2-6	**25**
0-2	**31**
0-1	***32***
3-4	34
5-4	40
2-4	46
0-6	51
4-6	51
0-5	60

5-3	18
1-2	21
2-6	25
0-2	31

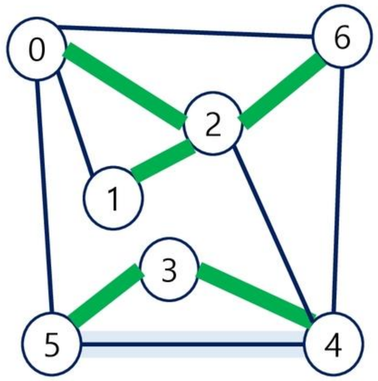
싸이클 존재
➔다음 간선선택



5-3	**18**
1-2	**21**
2-6	**25**
0-2	**31**
0-1	32
3-4	**34**
5-4	40
2-4	46
0-6	51
4-6	51
0-5	60

5-3	18
1-2	21
2-6	25
0-2	31
3-4	34

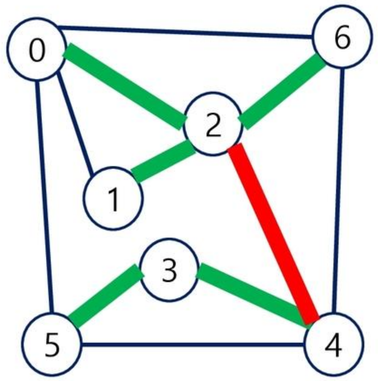
KRUSKAL 알고리즘



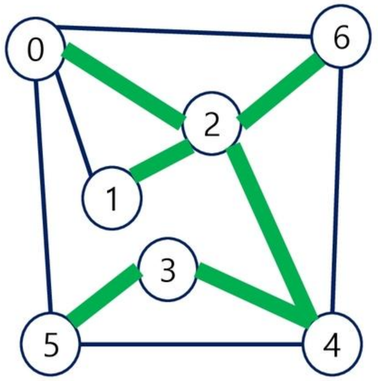
5-3	**18**
1-2	**21**
2-6	**25**
0-2	**31**
0-1	32
3-4	**34**
5-4	***40***
2-4	46
0-6	51
4-6	51
0-5	60

5-3	18
1-2	21
2-6	25
0-2	31
3-4	34

싸이클 존재
➔다음 간선선택



5-3	**18**
1-2	**21**
2-6	**25**
0-2	**31**
0-1	32
3-4	**34**
5-4	40
2-4	**46**
0-6	51
4-6	51
0-5	60



5-3	18
1-2	21
2-6	25
0-2	31
3-4	34
2-4	46

그래프의 정점의 수 N = 7
N-1 = 6 개의 간선이 선택됨

KRUSKAL 알고리즘

알고리즘

```
// G.V: 그래프의 정점 집합
// G.E: 그래프의 간선 집합

MST-KRUSKAL(G, w)
    FOR vertex v in G.V
        Make-Set(v)

    G.E에 포함된 간선들을 가중치 w를 이용한 오름차순 정렬

    FOR 가중치가 가장 낮은 간선 (u, v) ∈ G.E 선택(n-1개)
        IF  Find-Set(u) ≠ Find-Set(v)
            Union(u, v)
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

PRIM 알고리즘

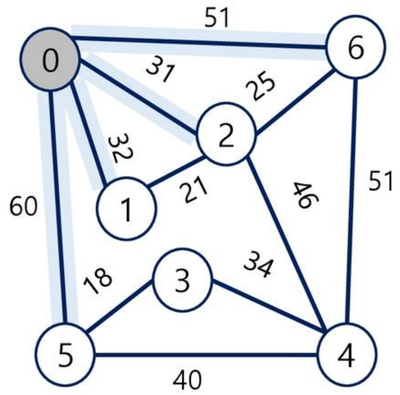
PRIM 알고리즘

- **하나의 정점에서 연결된 간선들 중에 하나씩 선택하면서 MST를 만들어 가는 방식**
  - ① 임의 정점을 하나 선택해서 시작
  - ② 선택한 정점과 인접하는 정점들 중의 최소 비용의 간선이 존재하는 정점을 선택
  - ③ 모든 정점이 선택될 때 까지 ①, ② 과정을 반복

- **서로소인 2개의 집합(2 disjoint-sets) 정보를 유지**
  - 트리 정점들(tree vertices) – MST를 만들기 위해 선택된 정점들
  - 비트리 정점들(non-tree vertices) – 선택 되지 않은 정점들

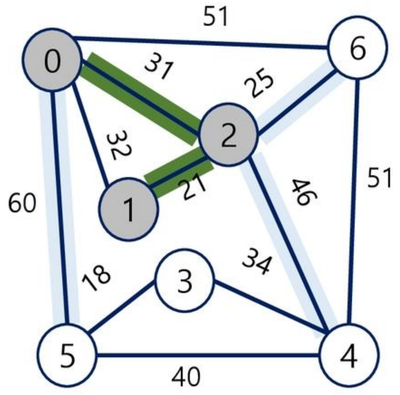
PRIM 알고리즘

알고리즘 적용 예



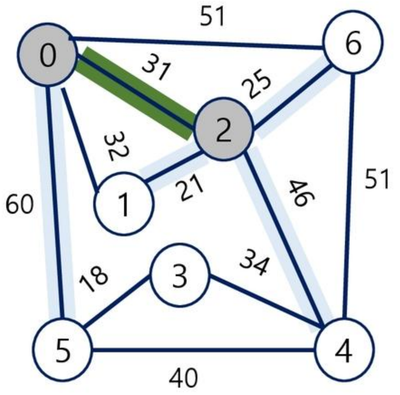
간선들> 0-1 0-2 0-5 0-6

누적 비용: 0



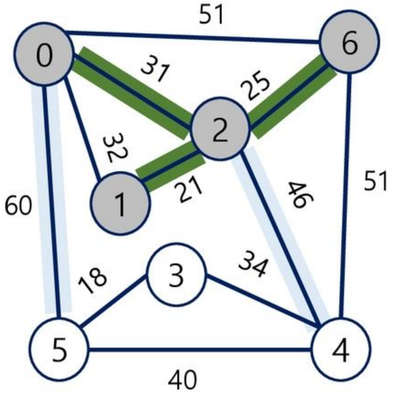
간선들> 0-5 2-4 2-6

누적 비용: 52

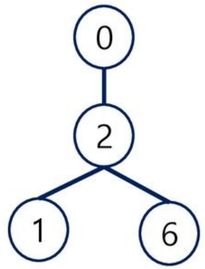


간선들> ~~0-1~~ 0-5 ~~0-6~~ 2-1 2-4 2-6

누적 비용: 31

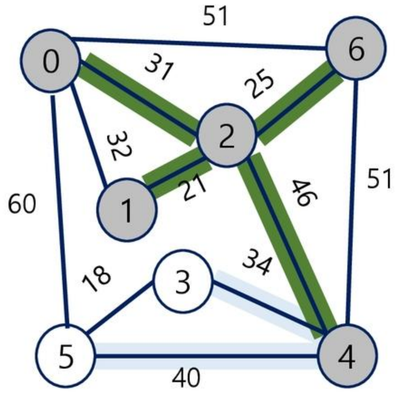


간선들> 0-5 2-4 ~~6-4~~

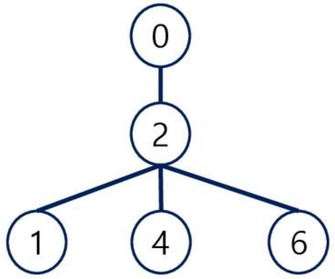


누적 비용: 77

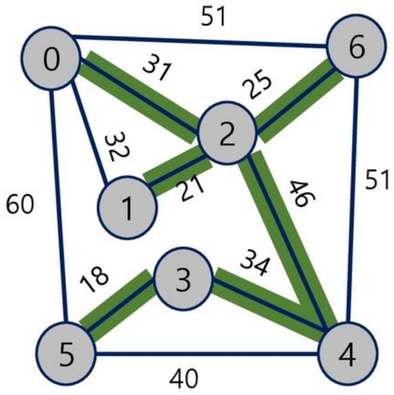
PRIM 알고리즘



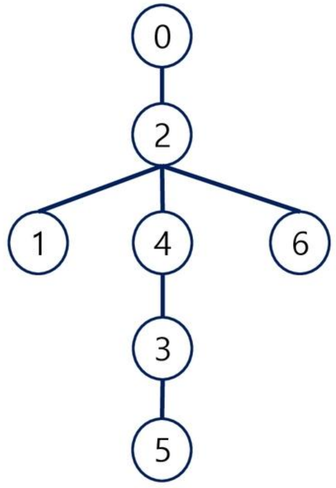
간선들> ~~0-5~~ 4-3 4-5



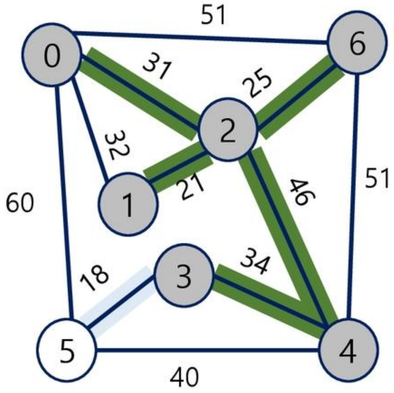
누적 비용: 123



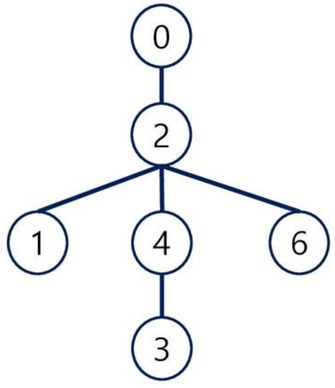
.51



누적 비용: 175



간선들> ~~5~~ ~~4-5~~ 3-5



누적 비용: 157

PRIM 알고리즘

알고리즘

```
MST-PRIM(G, r)              // G: 그래프, r:시작 정점, minEdge[]: 각 정점기준으로 타 정점과의 최소 간선 비용
    result ← 0, cnt ← 0  // result: MST비용, cnt: 처리한 정점수, visited[]: MST에 포함된 정점여부
    FOR u in G.V
        minEdge[u] ← ∞
    minEdge[r] ← 0                          // 시작 정점 r의 최소비용 0 처리
    WHILE true
        u ← Extract-MIN( )                  // 방문하지 않은(MST에 포함되지 않은 정점) 최소비용 정점 찾기
        visited[u] ← true                   // 방문 처리
        result ← result + minEdge[u];       // 비용 누적
        if(++cnt == N) break;               // 모든 정점이 다 연결되었으면 MST 완성
        FOR v in G.Adj[u]                   // u의 인접 정점들
                IF visited[v] == false AND w(u, v) < minEdge[v]   // u->v 비용이 v의 최소비용보다 작다면 갱신
                        minEdge[v] ← w(u, v)
    return result
end MST-PRIM
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

Appendix

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

PRIM 알고리즘 with PriorityQueue

PRIM 알고리즘 with PriorityQueue

알고리즘

```
MST-PRIM(G, r)              // G: 그래프, r:시작 정점, minEdge[]: 각 정점기준으로 타 정점과의 최소 간선 비용
    result ← 0, cnt ← 0     // result: MST비용, cnt: 처리한 정점수, visited[]: MST에 포함된 정점여부
    FOR u in G.V
        minEdge[u] ← ∞
    minEdge[r] ← 0                          // 시작 정점 r의 최소비용 0 처리
    WHILE true
        u ← Extract-MIN( )                  // 방문하지 않은(MST에 포함되지 않은 정점) 최소비용 정점 찾기
        ….
    return result
end MST-PRIM
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

다음 방송에서 만나요!

삼성 청년 SW 아카데미