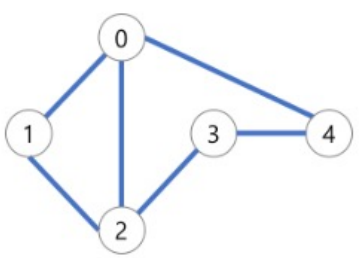
# Minimum Spanning Trees

202020988 조아영

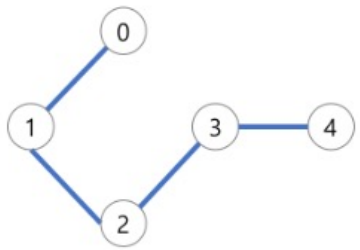
# Spanning Trees

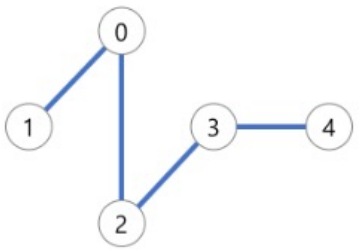
- 방향성이 없는 그래프 내의 모든 정점을 포함하는 트리
- No cycle

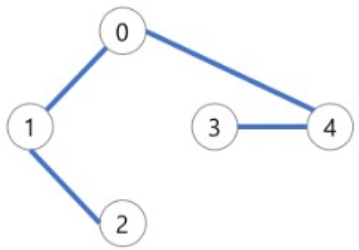
- 그래프에서 일부 간선을 선택해서 만든 트리
- N개의 정점을 정확히 N-1개의 간선으로 연결함
- 하나의 그래프에는 여러 개의 신장 트리가 존재할 수 있음




(a)연결 그래프
정점: 5개, 간선: 6개




(b) 신장 트리 중의 일부
정점: 5개, 간선: 4개

# Minimum Spanning Trees

- 그래프의 모든 정점들을 가장 적은 수의 간선과 비용으로 연결

특징

- 간선의 가중치의 합이 최소여야 함
- N개의 정점을 가지는 그래프에 대해 반드시 N-1개의 간선을 사용해야 함
- 사이클이 포함되어서는 안됨

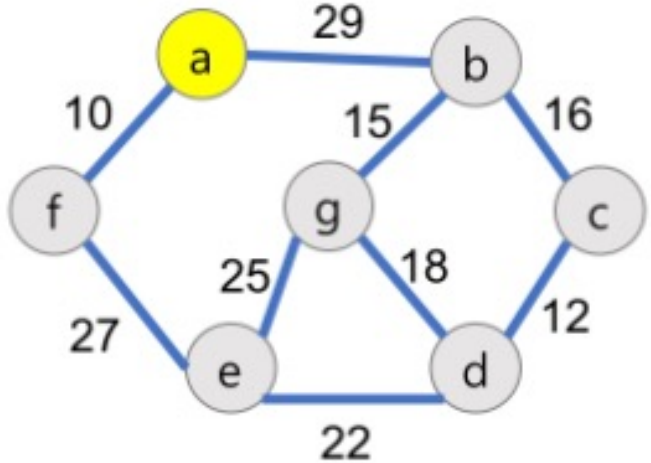
# Prim Algorithm

- 정점 선택을 기반으로 하는 알고리즘
- 이전 단계에서 만들어진 신장 트리를 확장해 나가는 방법

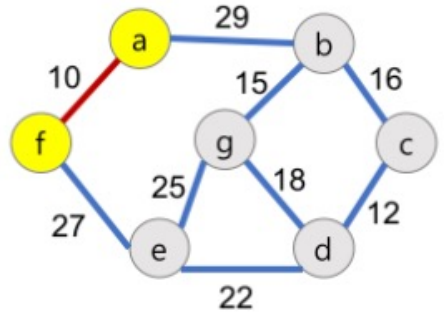
과정)

1. 시작 정점을 선택된 정점 집합에 넣어준다
2. 앞 단계의 선택된 정점 집합에 속한 정점과 인접한 정점들 중 가중치가 작은 간선으로 연결된 정점을 선택하여 트리를 확장
3. 위의 과정을 모든 노드가 연결될 때까지 반복함

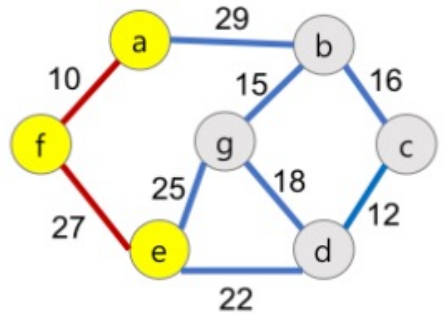
1) 시작 단계: 시작 정점만 포함



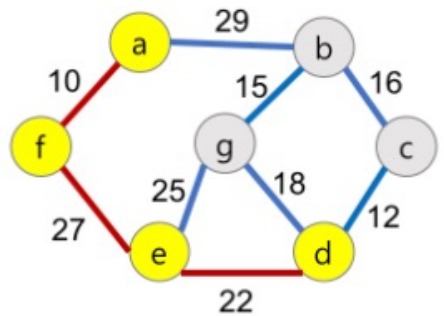

2) 인접 정점 중 최소 간선으로 연결된 정점 선택



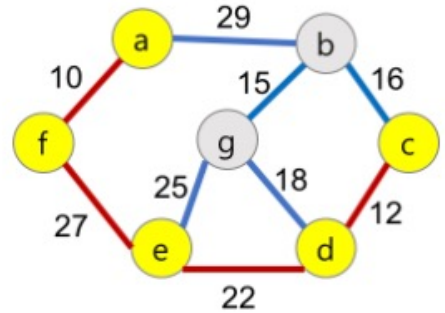

3)



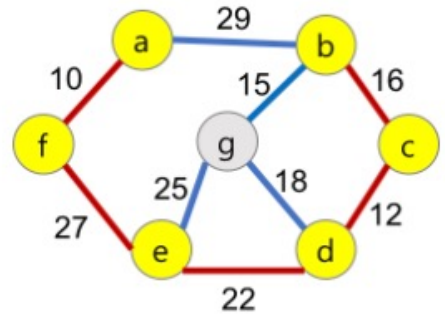

4)



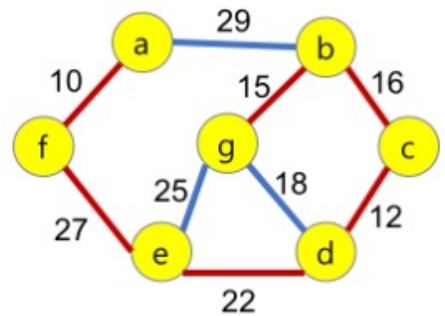

5)




6)




7) N-1개의 간선 생성. 종료



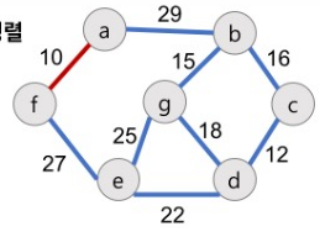
# Kruscal Algorithm

- Greedy algorithm를 이용하여 최적해를 구함
- 간선 선택을 기반으로 하는 알고리즘
- 최소 간선을 선택

과정)

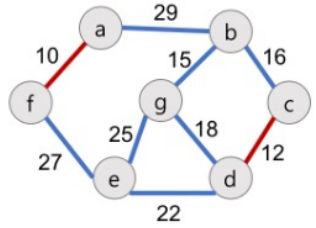
1. 간선들의 가중치를 오름차순으로 정렬
2. 가장 낮은 가중치부터 사이클을 형성하지 않는 간선을 선택
3. 해당 간선이 연결된 정점과 간선의 가중치를 저장

1) 간선들의 가중치 오름차순 정렬



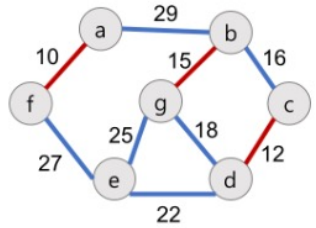

| | af | cd | bg | bc | dg | de | eg | ef | ab |
|---|---|---|---|---|---|---|---|---|---|
| 간선의 가중치 | 10 | 12 | 15 | 16 | 18 | 22 | 25 | 27 | 29 |

2)



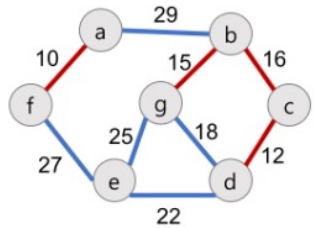

| | af | cd | bg | bc | dg | de | eg | ef | ab |
|---|---|---|---|---|---|---|---|---|---|
| 간선의 가중치 | 10 | 12 | 15 | 16 | 18 | 22 | 25 | 27 | 29 |

3)



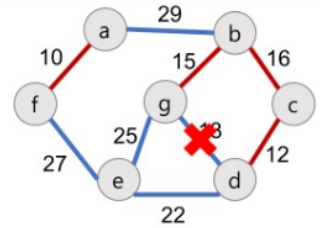

| | af | cd | bg | bc | dg | de | eg | ef | ab |
|---|---|---|---|---|---|---|---|---|---|
| 간선의 가중치 | 10 | 12 | 15 | 16 | 18 | 22 | 25 | 27 | 29 |

4)



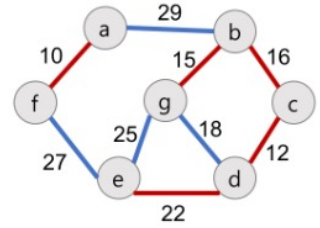

| | af | cd | bg | bc | dg | de | eg | ef | ab |
|---|---|---|---|---|---|---|---|---|---|
| 간선의 가중치 | 10 | 12 | 15 | 16 | 18 | 22 | 25 | 27 | 29 |

5) 사이클 형성. dg는 제외



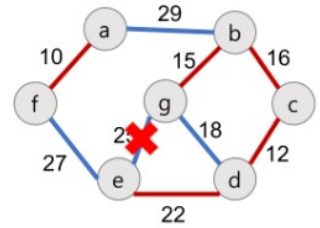

| | af | cd | bg | bc | dg | de | eg | ef | ab |
|---|---|---|---|---|---|---|---|---|---|
| 간선의 가중치 | 10 | 12 | 15 | 16 | 18 | 22 | 25 | 27 | 29 |

6)



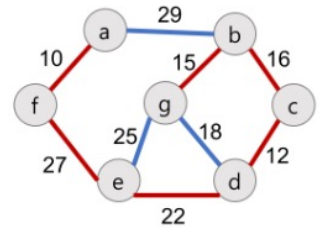

| | af | cd | bg | bc | dg | de | eg | ef | ab |
|---|---|---|---|---|---|---|---|---|---|
| 간선의 가중치 | 10 | 12 | 15 | 16 | 18 | 22 | 25 | 27 | 29 |

7) 사이클 형성. eg는 제외




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

8) N-1개의 간선 생성. 종료




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

# Time Complexity

**Prim**

모든 노드 탐색 : O(V)

우선순위 큐를 사용하여 매 노드마다 최소 간선을 찾는 시간 : O(log V)

탐색과정에는 O(Vlog V)

노드의 인접 간선을 찾는 시간 : O(E)

각 간선에 대해 힙에 넣는 과정이 O(log V)

= > O(Vlog V) + O(Elog V)

그래프에 간선이 많이 존재하는 '밀집 그래프(Dense Graph)' 의 경우 적합함

**Kruscal**

- union-find 알고리즘을 이용하면
  Kruskal 알고리즘의 시간 복잡도는 정렬의 시간 복잡도와 동일
- 간선 n개를 퀵 정렬과 같은 효율적인 알고리즘으로 정렬

=> O(nlogn)

그래프 내에 적은 숫자의 간선만을 가지는 '희소 그래프(Sparse Graph)'의 경우 적합