ALGORITHM 알고리즘

Lecture 09

그래프 (2)

컴퓨터과학과 | 이관용 교수

한국방송통신대학교

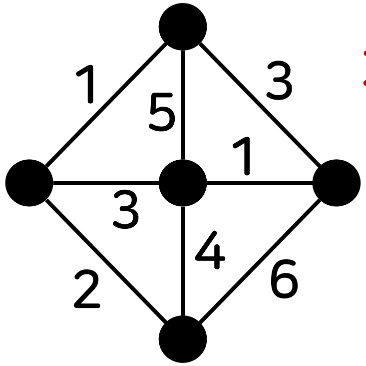
학습목차

01. 최소 신장 트리: 크루스칼 알고리즘

최소 신장 트리?

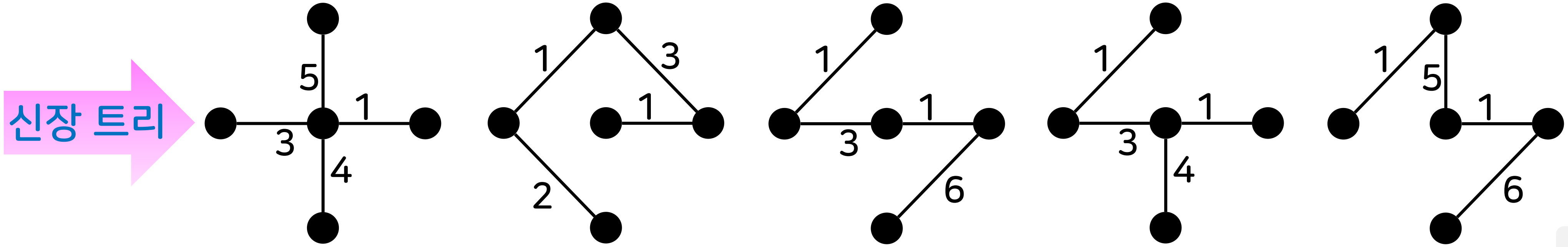
신장 트리 spanning tree

- 가중 무방향 그래프에서 모든 정점을 포함하는 트리



가중 무방향 그래프

|V|=n이면, 트리에는 정확히 (n-1)개의 간선이 존재



최소 신장 트리

최소 (비용) 신장 트리 minimum spanning tree

- 간선 (u, v)마다 가중치 w(u, v)를 가진
  연결된 무방향 그래프 G=(V, E)에 대해서 다음을 만족하는 트리 G'=(V, E')

$$E' \subseteq E \qquad w(E') = \min\left\{ \sum_{(u,v)\in E'} w(u,v) \right\}$$

→ 신장 트리 중에서 가중치의 합이 가장 작은 것

최소 신장 트리를 구하는 알고리즘

모든 간선 중에서 정점을 모두 연결하면서 가중치의 합을 가장 작게 만드는 (n-1)개의 간선을 고르는 과정

```
Greedy_MST ( G )
{
  T ← Ø ;       // 최소 신장 트리의 간선 집합
  while ( T가 신장 트리를 만들지 않았음 ) {
     최선의 간선 (u, v) 선택;
     T ← T ∪ { (u, v) };
  }
  return (T);
}
```

사이클을 형성하지 않으며
최소의 가중치를 갖는 간선

- 욕심쟁이 방법 적용 → 크루스칼 Kruskal 알고리즘, 프림 Prim 알고리즘

크루스칼 알고리즘?

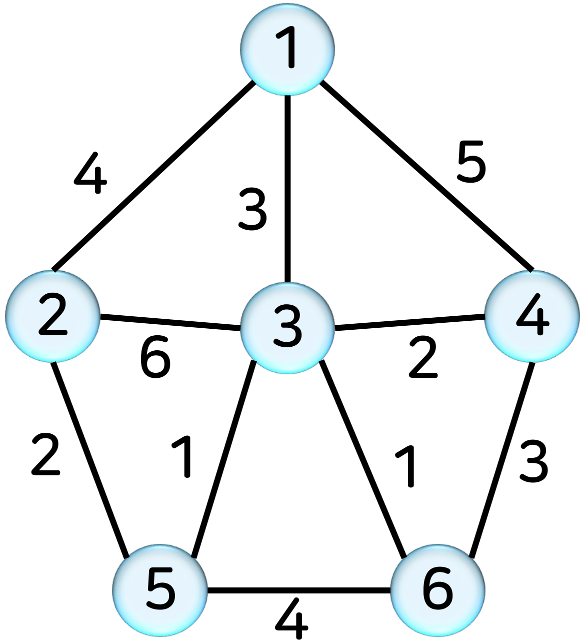
간선이 하나도 없는 상태에서 시작해서 가중치가 가장 작은 간선부터 하나씩 골라서 사이클을 형성하지 않으면 해당 간선을 추가하는 방식

- 사이클 형성 여부의 판단
  - ✓ 간선 (u, v)의 두 정점 u, v가 서로 다른 연결 성분에 속하면 사이클을 형성하지 않음
- |V|=n개의 정점이 각각의 서로 다른 연결 성분으로 구성된 상태에서 시작해서 간선을 추가할 때마다 연결 성분들이 하나씩 합쳐지고 최종적으로 하나의 연결 성분을 형성함

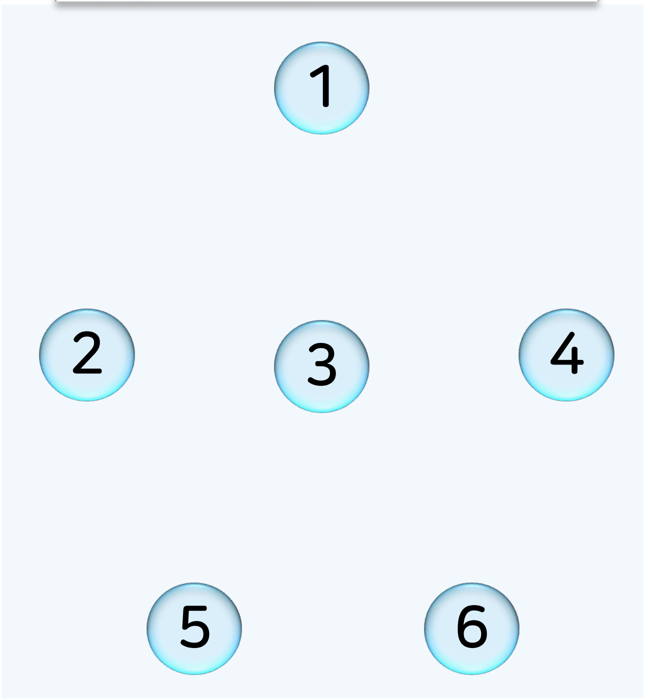
크루스칼 알고리즘


```
Kruskal ( G )
{
   T = Ø;
①  for ( G의 각 정점 v에 대해 )
      정점 v로 구성된 연결 성분 초기화;
②  가중치가 증가하는 순으로 모든 간선을 정렬;
③  for ( 가중치가 가장 작은 간선부터 모든 간선 (u, v)∈E에 대해서 )
      if ( u와 v가 서로 다른 연결 성분에 속하면 ) {    //사이클을 형성하지 않으면
         T = T ∪ { (u, v) };
         u가 속한 연결 성분과 v가 속한 연결 성분을 합침;
      }
      else 간선 (u, v)를 버림;
   return (T);
}
```

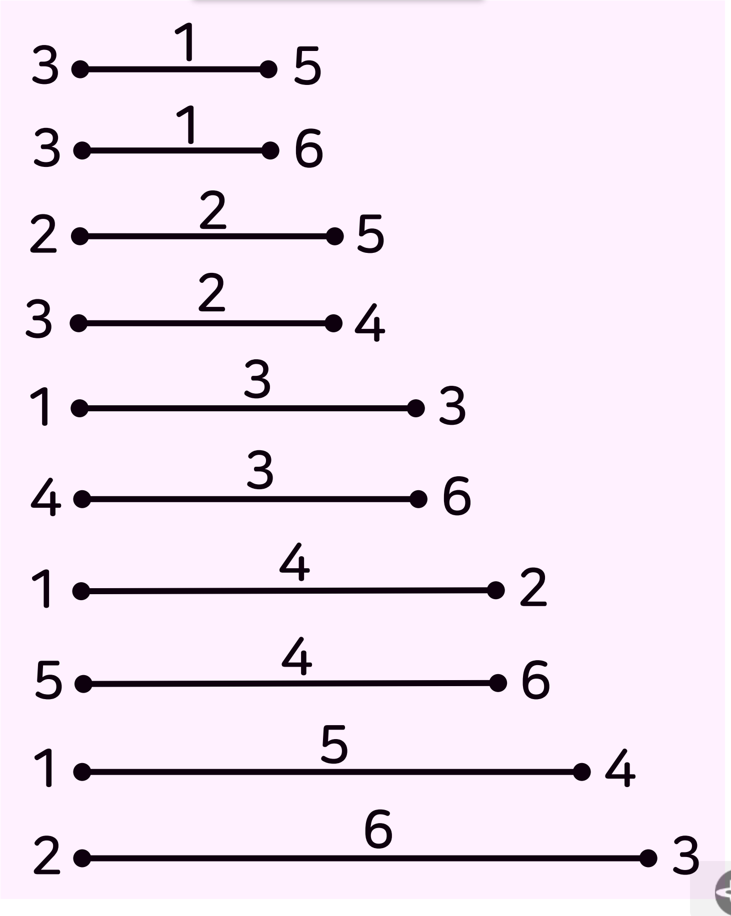
크루스칼 알고리즘_예_1



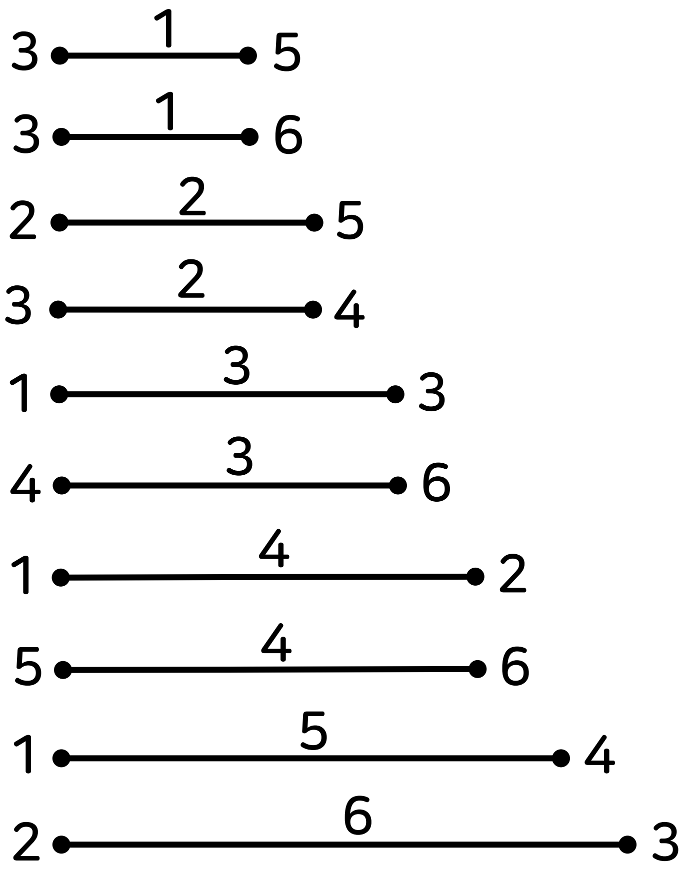
연결 성분의 초기화

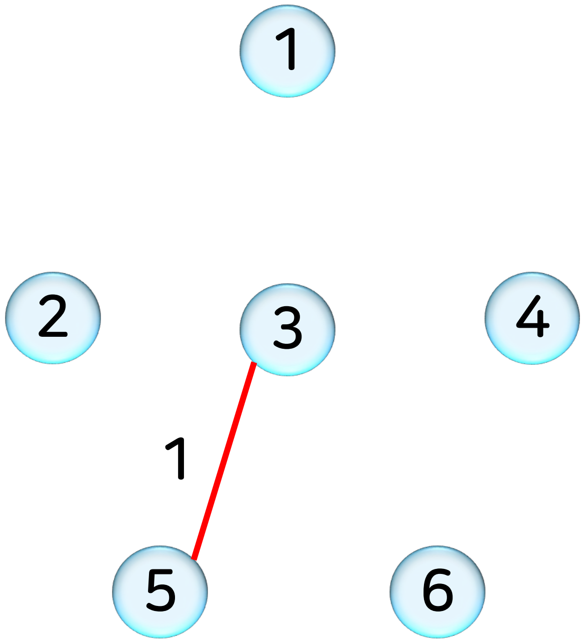


간선 정렬



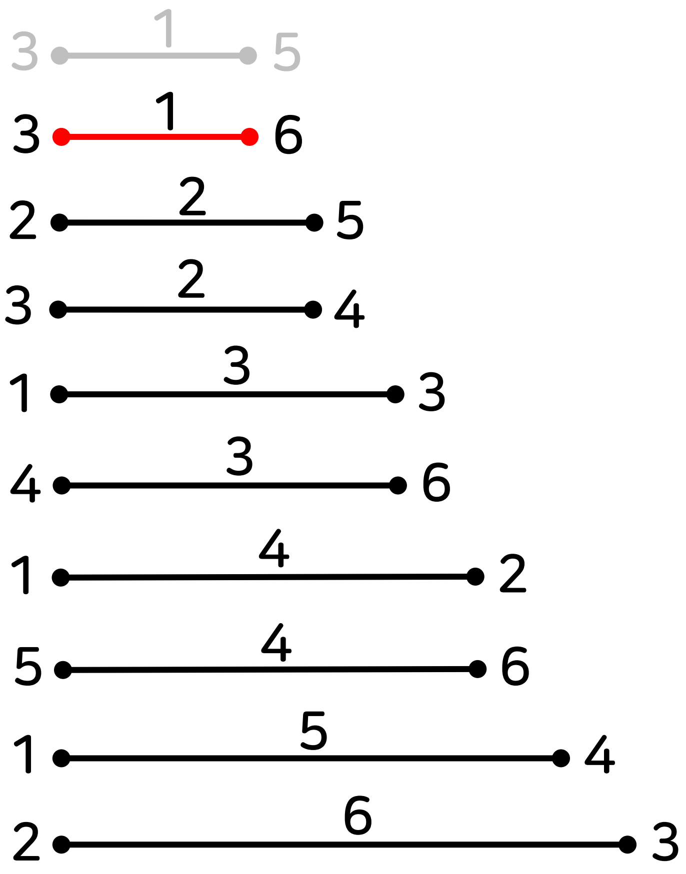
크루스칼 알고리즘_예_1

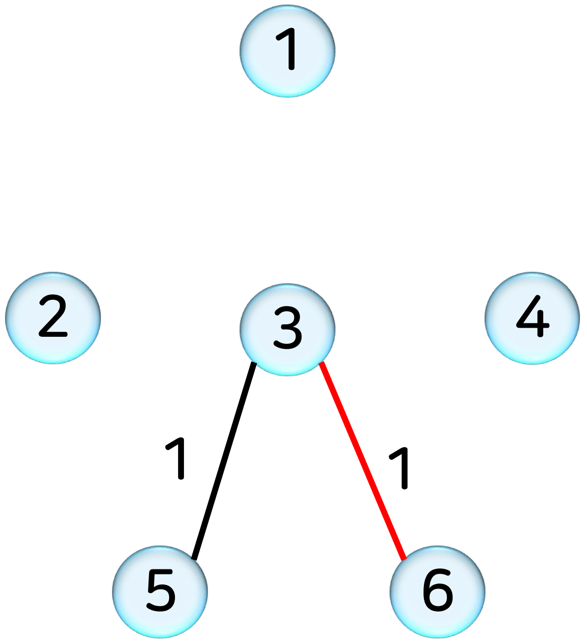




간선 (3, 5) 추가

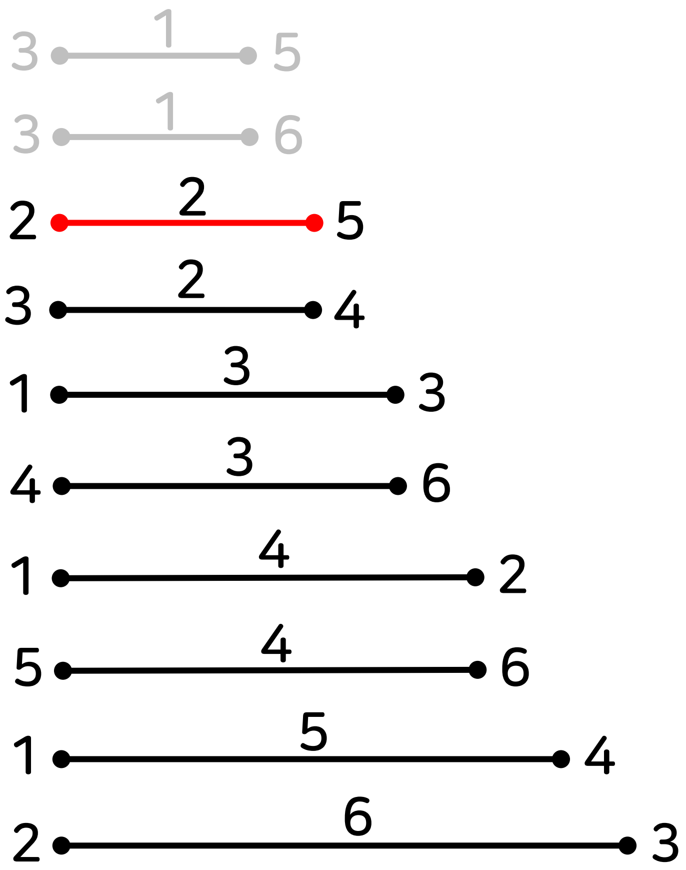
크루스칼 알고리즘_예_1

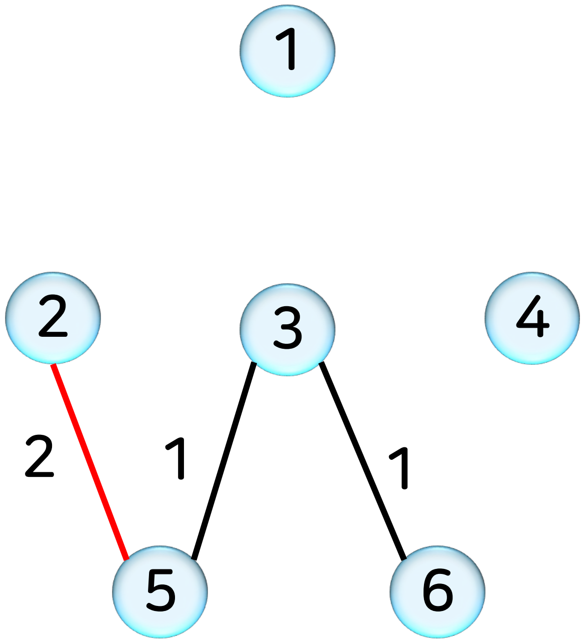




간선 (3, 6) 추가

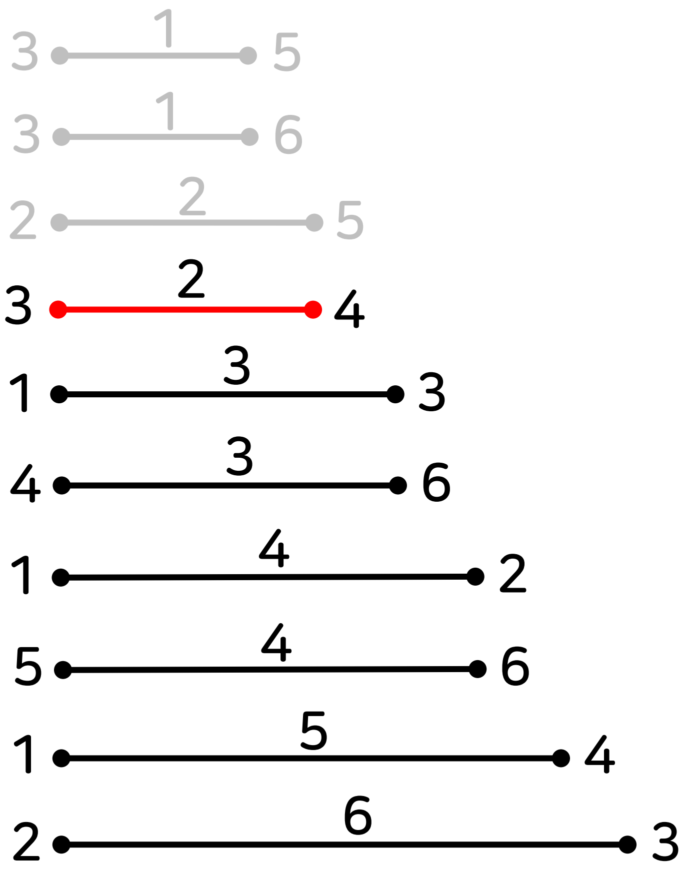
크루스칼 알고리즘_예_1

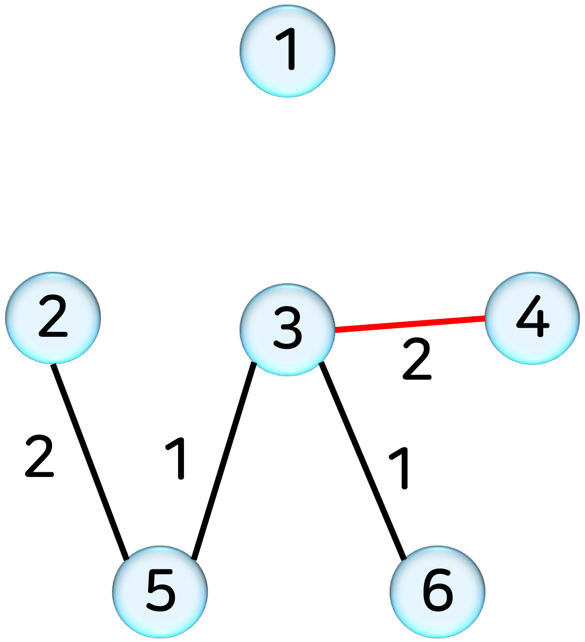




간선 (2, 5) 추가

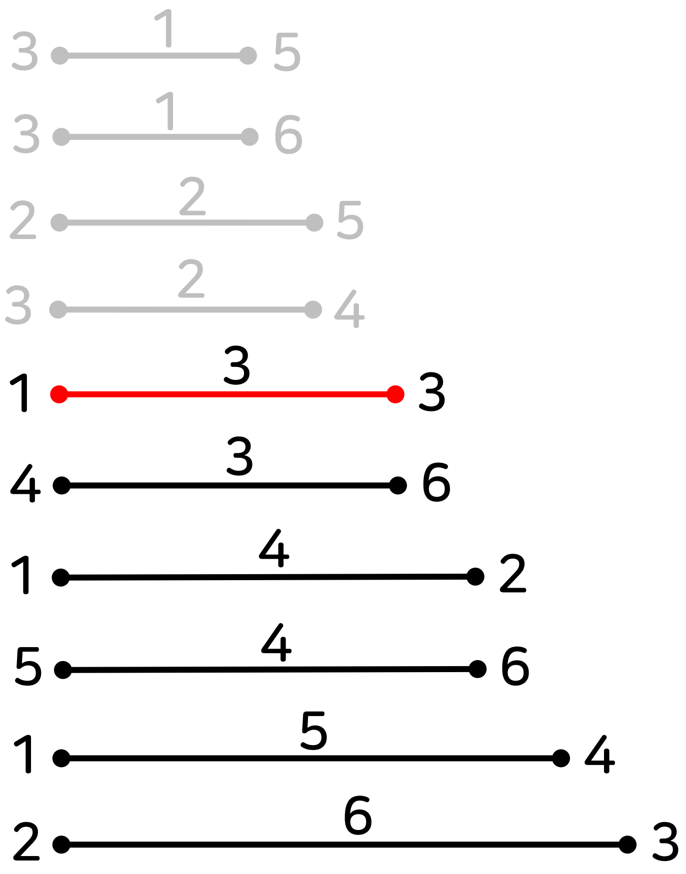
크루스칼 알고리즘_예_1

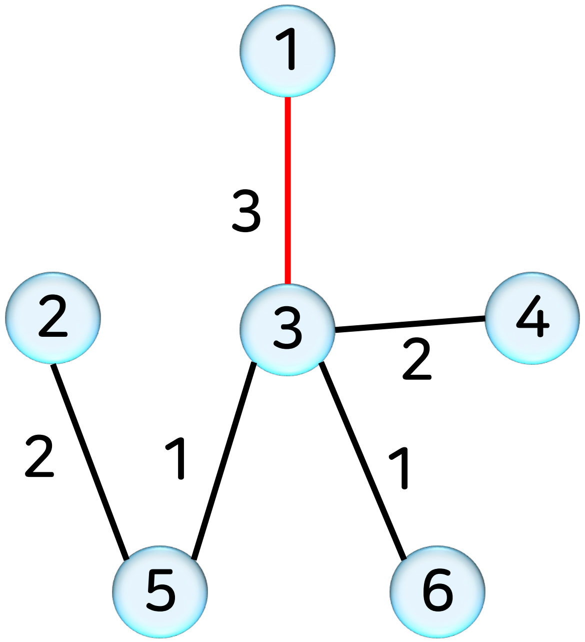




간선 (3, 4) 추가

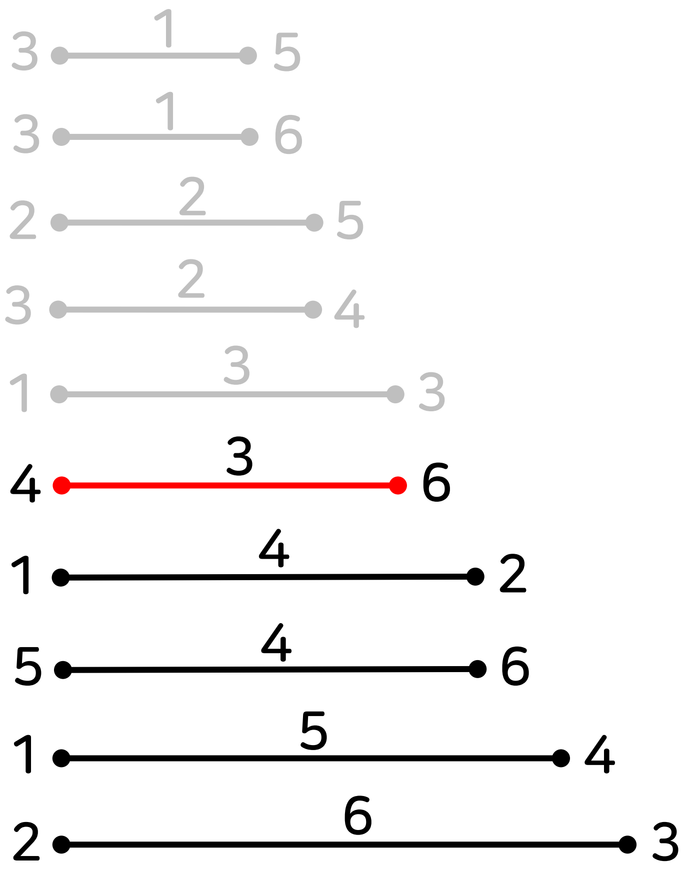
크루스칼 알고리즘_예_1

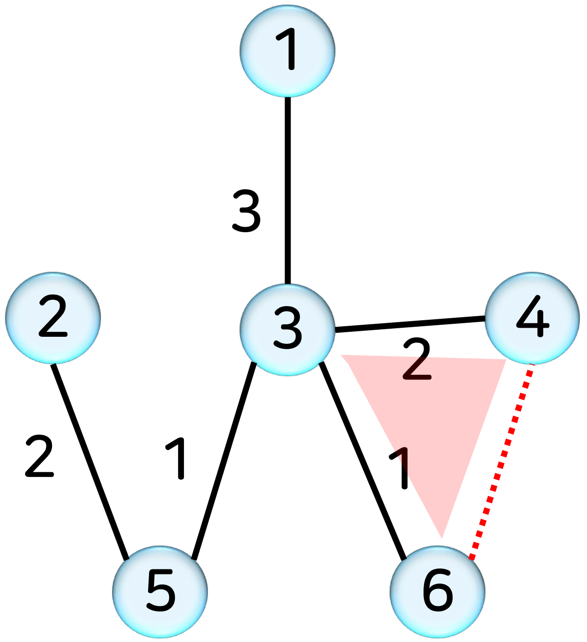




간선 (1, 3) 추가

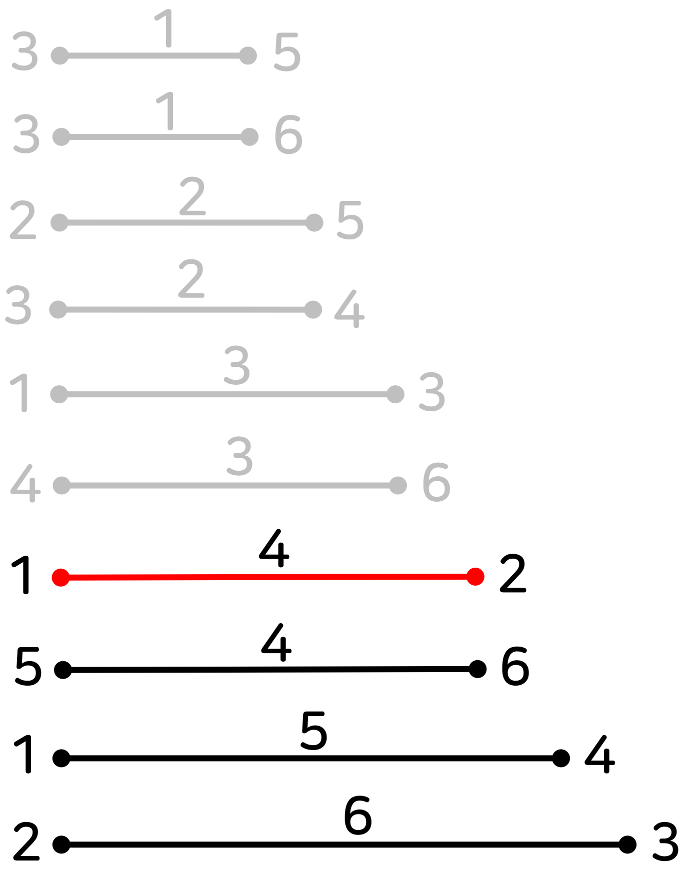
크루스칼 알고리즘_예_1

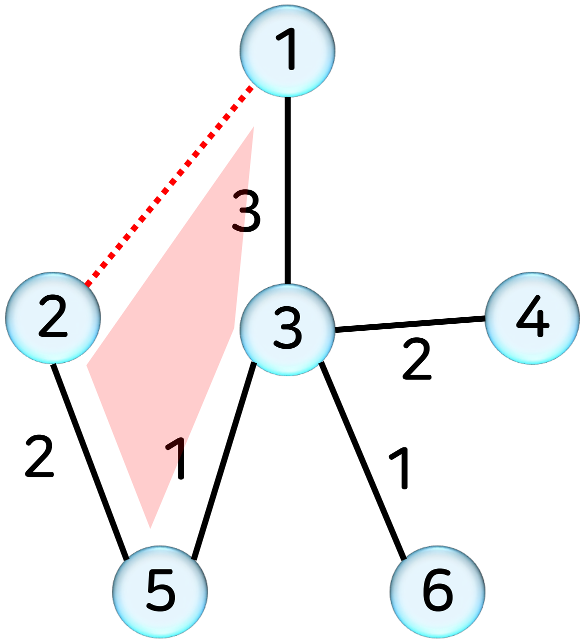




간선 (4, 6) → 제외

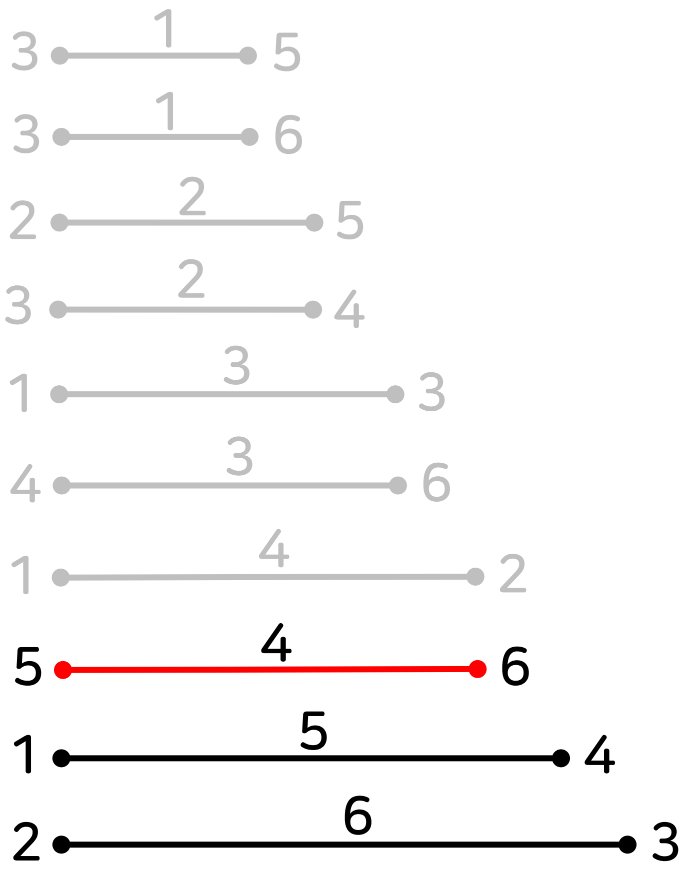
크루스칼 알고리즘_예_1

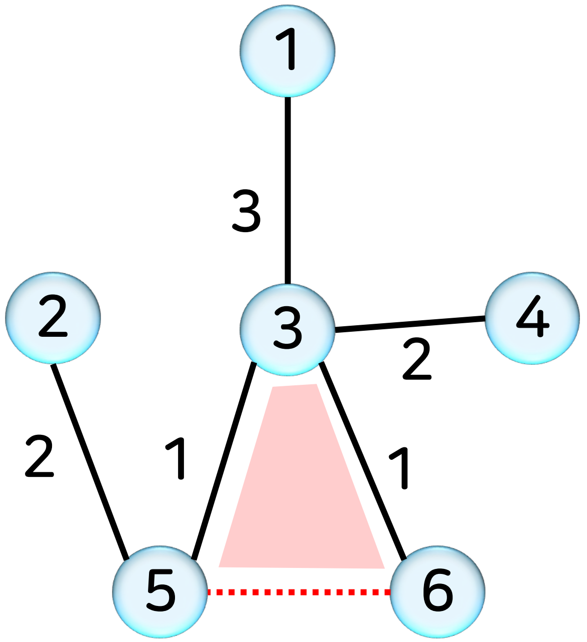




간선 (1, 2) → 제외

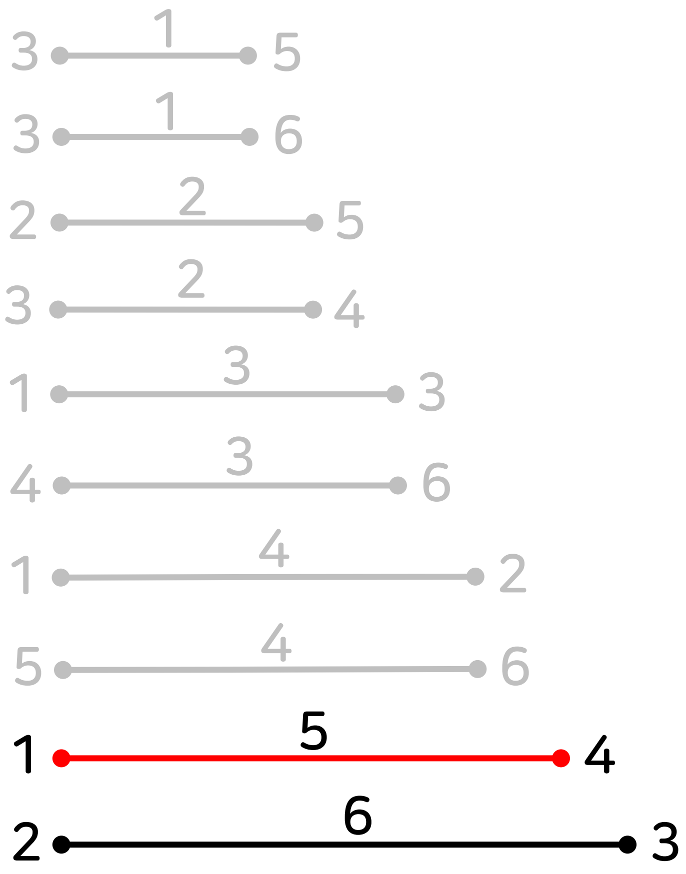
크루스칼 알고리즘_예_1

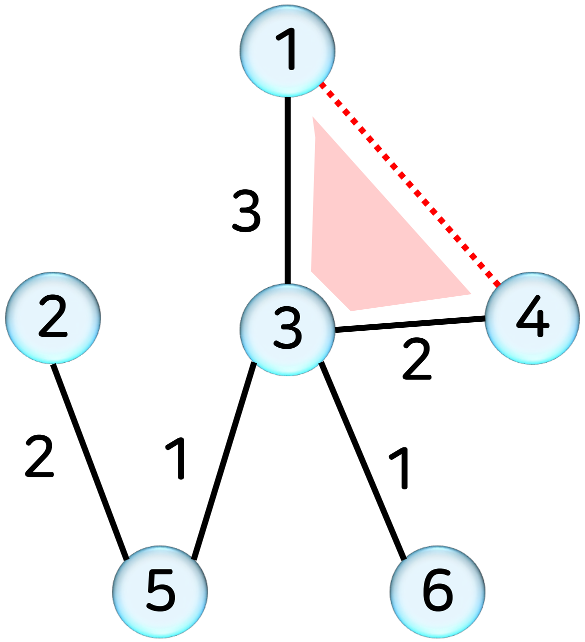




간선 (5, 6) → 제외

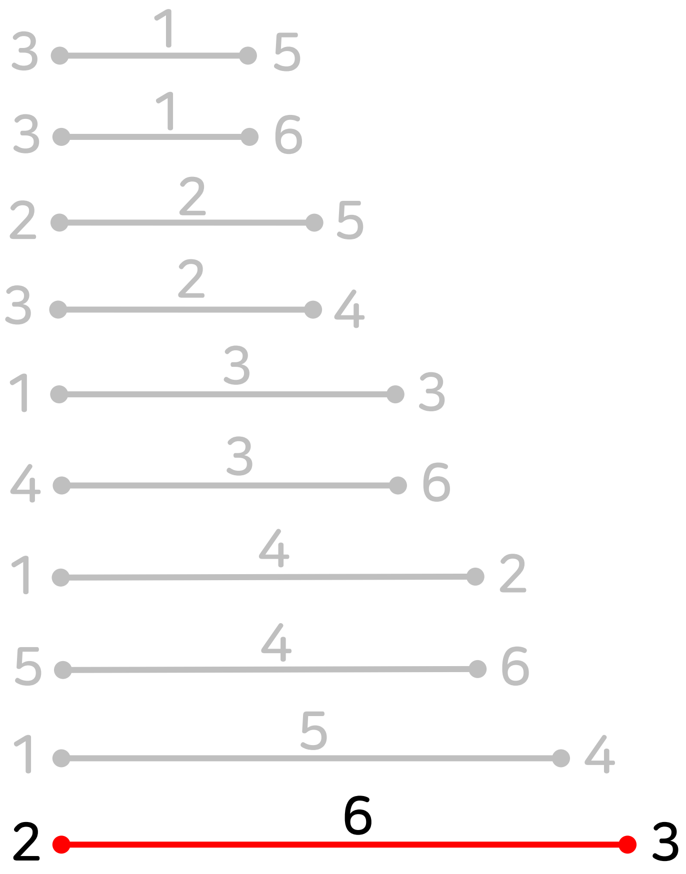
크루스칼 알고리즘_예_1

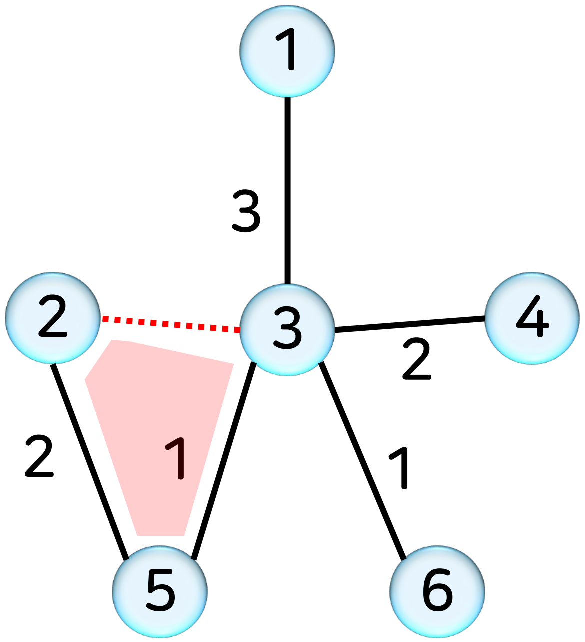




간선 (1, 4) → 제외

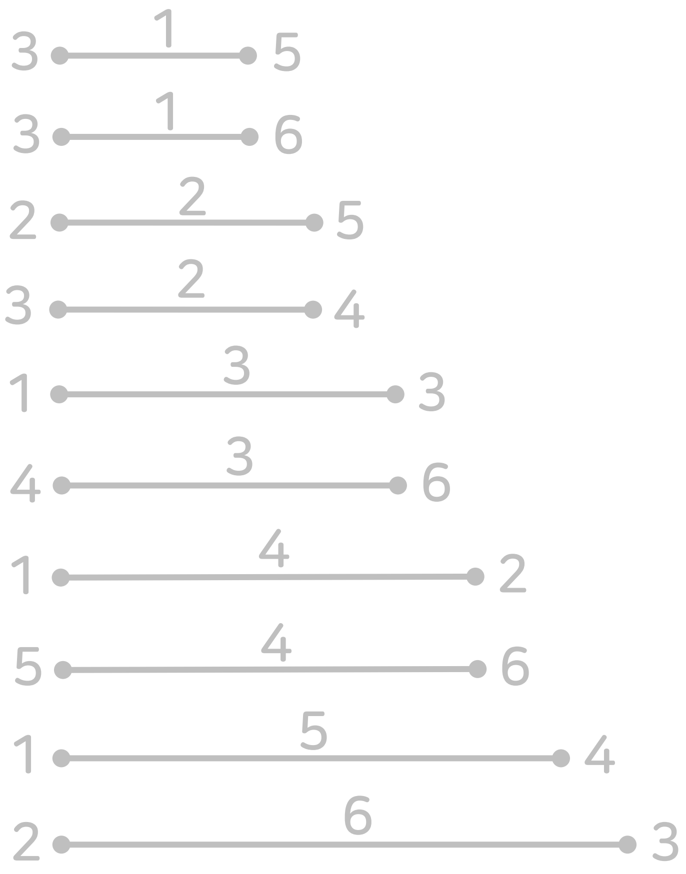
크루스칼 알고리즘_예_1

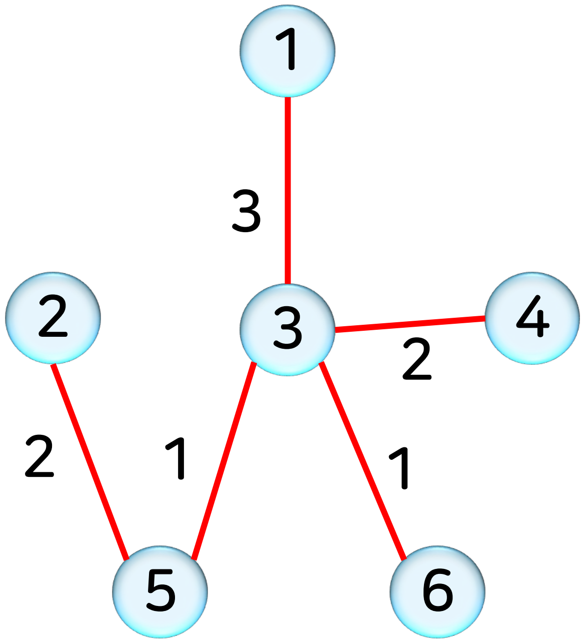




간선 (2, 3) → 제외

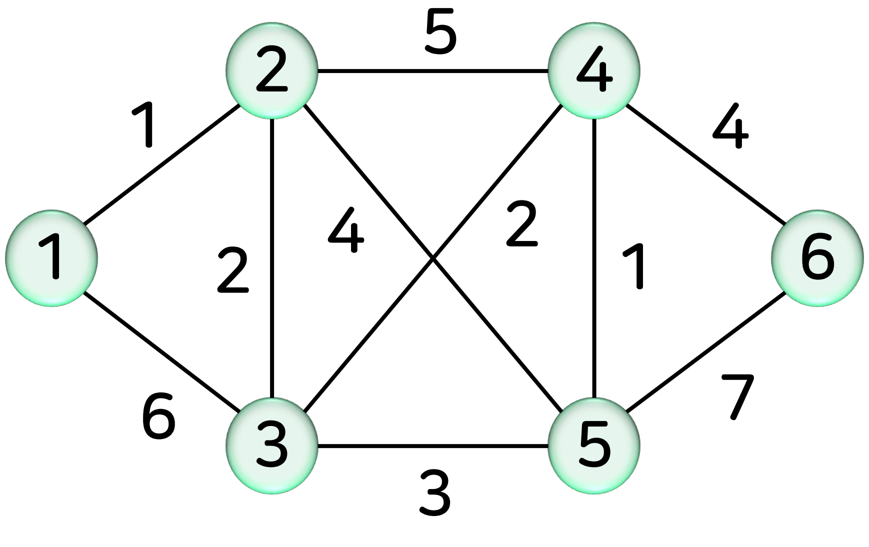
크루스칼 알고리즘_예_1

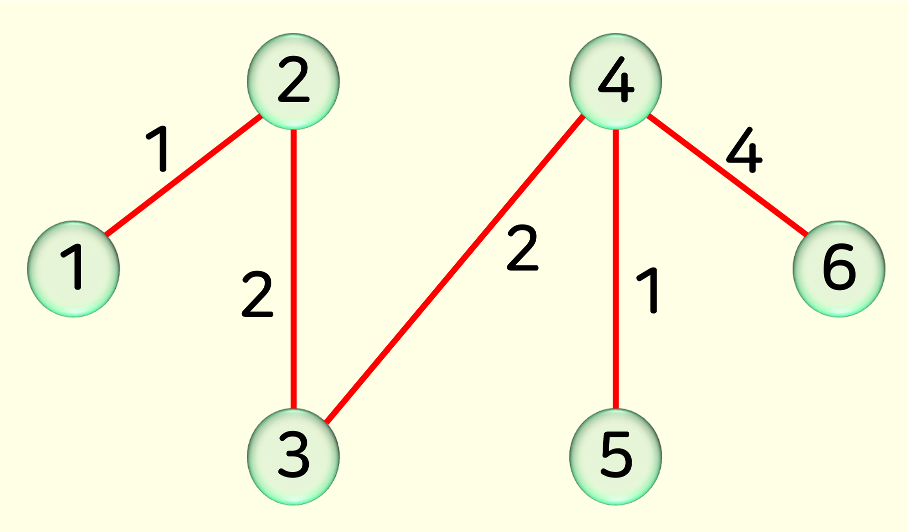




가중치의 합 → 1+1+2+2+3 = 9

크루스칼 알고리즘_예_2





가중치의 합 → 1+1+2+2+4 = 10

크루스칼 알고리즘의 성능


```
Kruskal ( G )
{
   T = Ø;                                         O(1)
   for ( G의 각 정점 v에 대해 )                      O(|V|)
      정점 v로 구성된 연결 성분 초기화;
   가중치가 증가하는 순으로 모든 간선을 정렬;            O(|E|log|E|)
   for ( 가중치가 가장 작은 간선부터 모든 간선 (u, v)∈E에 대해서 )   O(|E|)
      if ( u와 v가 서로 다른 연결 성분에 속하면 ) {       O(log|E|)
         T = T ∪ { (u, v) };
         u가 속한 연결 성분과 v가 속한 연결 성분을 합침;
      }
      else 간선 (u, v)를 버림;
   return (T);
}
```

O(|E|) · O(log|E|) → O(|E|log|E|)

O(|E|log|E|)

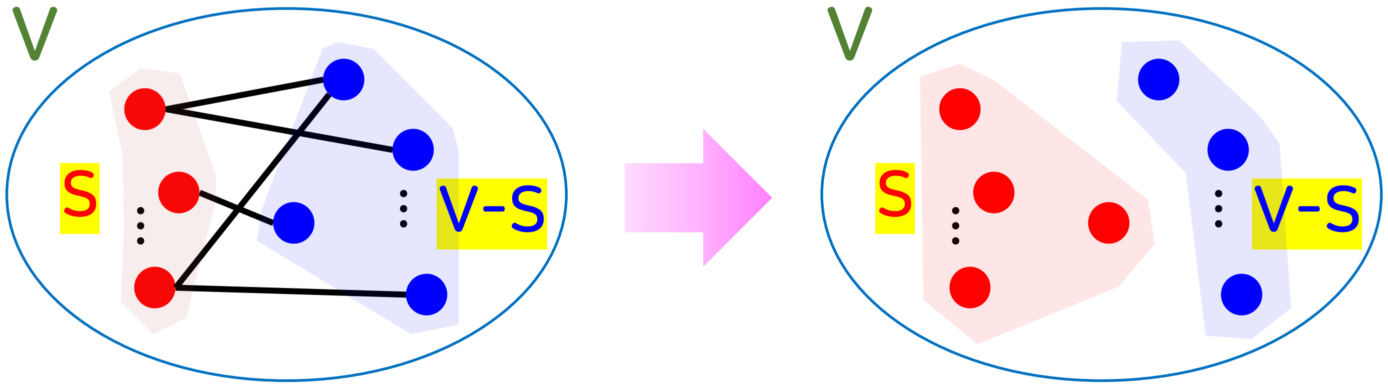
02. 최소 신장 트리: 프림 알고리즘

프림 알고리즘?

임의의 한 정점에서 시작해서 연결된 정점을 하나씩 선택해 나가는 방법

- 이미 선택된 정점들에 부수된 가중치가 가장 작은 간선을 선택해서 추가
- 어떤 순간에 이미 선택된 정점의 집합 S와 선택되지 않은 정점의 집합 V-S를 잇는 간선 중에서 가중치가 가장 작은 간선을 선택해서 추가하는 방법

✓ 임의의 정점 하나를 S로 지정한 후 시작해서 S=V가 될 때까지 S를 점점 키워 나가는 방법



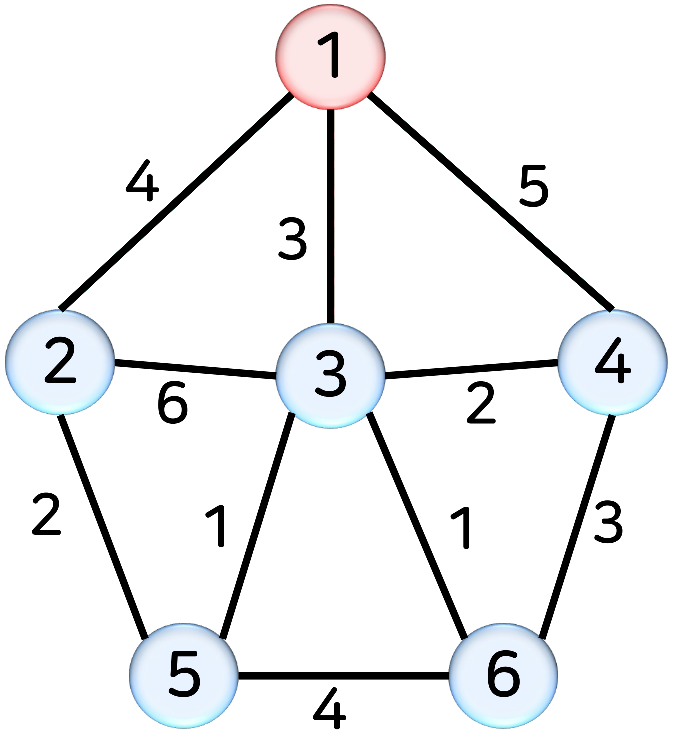
프림 알고리즘


```
Prim ( G )
{
   T = Ø;
   S = { 1 };         // 임의의 정점(예, 여기서는 1)으로 초기화
   while ( S != V ) {
       u∈S, v∈V-S인 것 중 가중치가 최소인 간선 (u, v) 선택;
       T = T ∪ { (u, v) };
       S = S ∪ { v };
   }
   return (T);
}
```

프림 알고리즘_예_1

S = { 1 }

V-S = { 2, 3, 4, 5, 6 }



S = { 1, 3 }

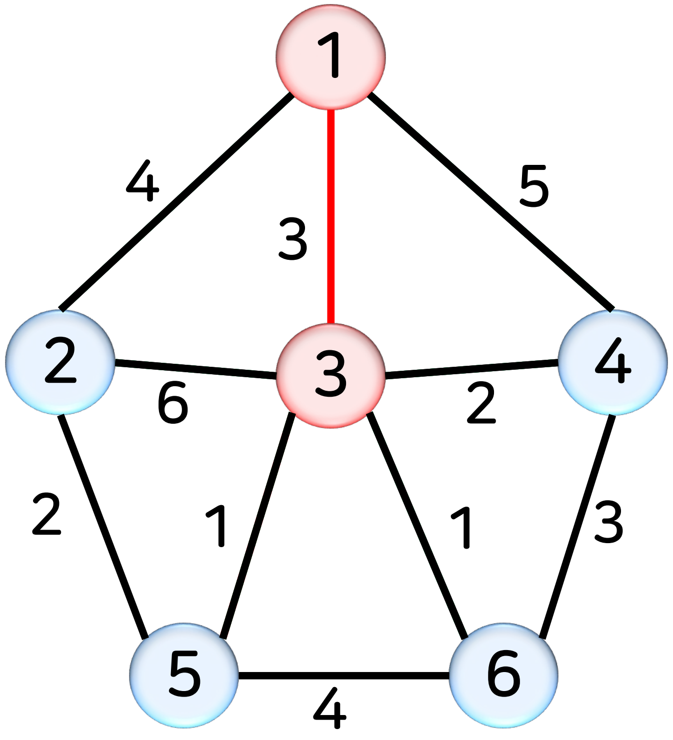
V-S = { 2, 4, 5, 6 }

T = { (1,3) }

프림 알고리즘_예_1

S = { 1, 3 }

V-S = { 2, 4, 5, 6 }



S = { 1, 3, 5 }

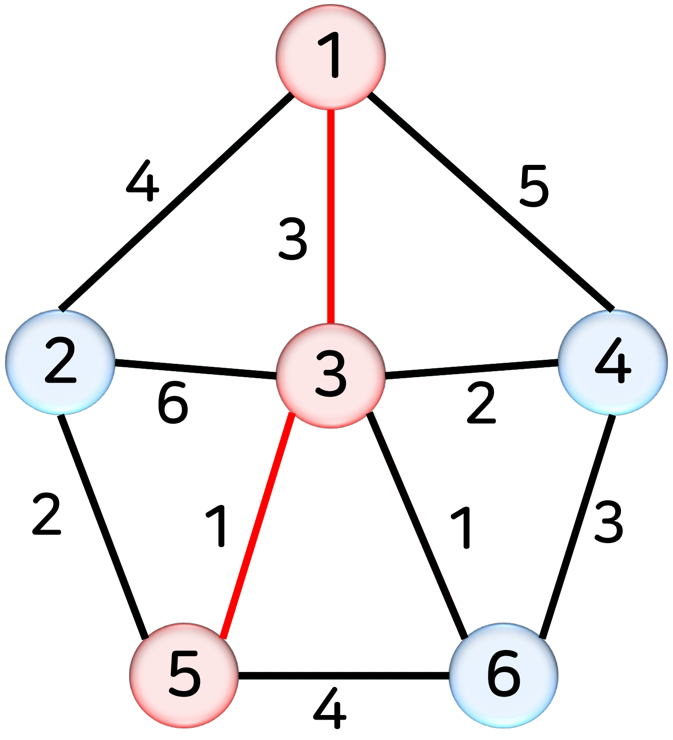
V-S = { 2, 4, 6 }

T = { (1,3), (3,5) }

프림 알고리즘_예_1

S = { 1, 3, 5 }

V-S = { 2, 4, 6 }



S = { 1, 3, 5, 6 }

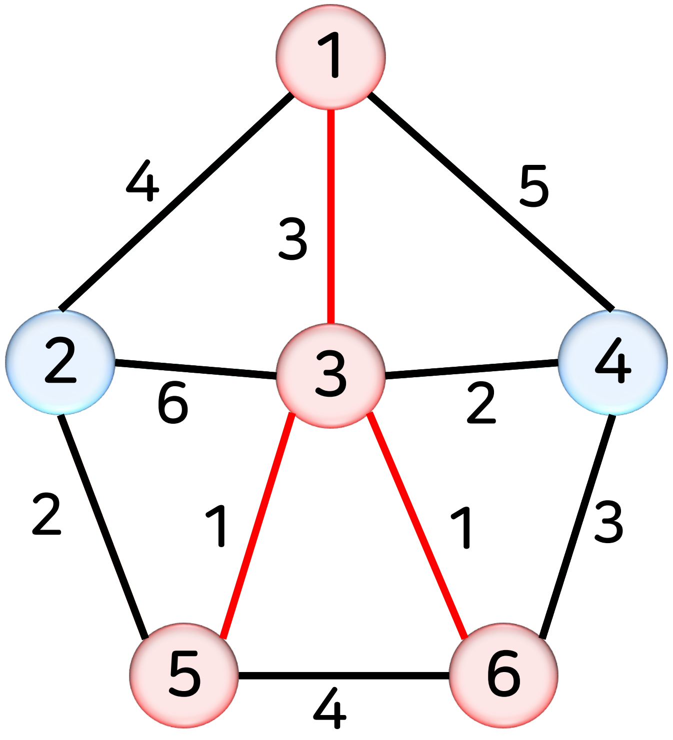
V-S = { 2, 4, 6 }

T = { (1,3), (3,5), (3,6) }

프림 알고리즘_예_1

S = { 1, 3, 5, 6 }

V-S = { 2, 4 }



S = { 1, 2, 3, 5, 6 }

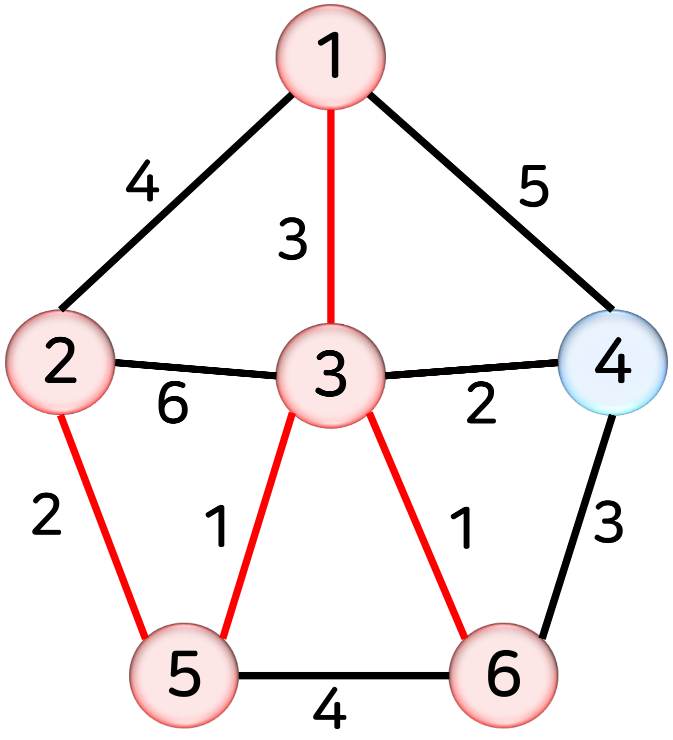
V-S = { 4 }

T = { (1,3), (3,5), (3,6), (2,5) }

프림 알고리즘_예_1

S = { 1, 2, 3, 5, 6 }

V-S = { 2 }



S = { 1, 2, 3, 4, 5, 6 }

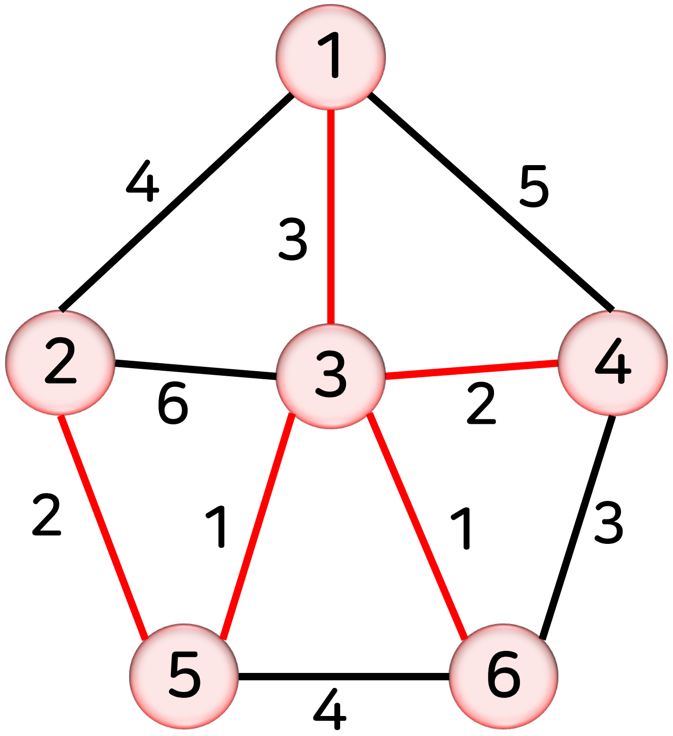
V-S = { }

T = { (1,3), (3,5), (3,6), (2,5), (3,4) }

프림 알고리즘_예_1

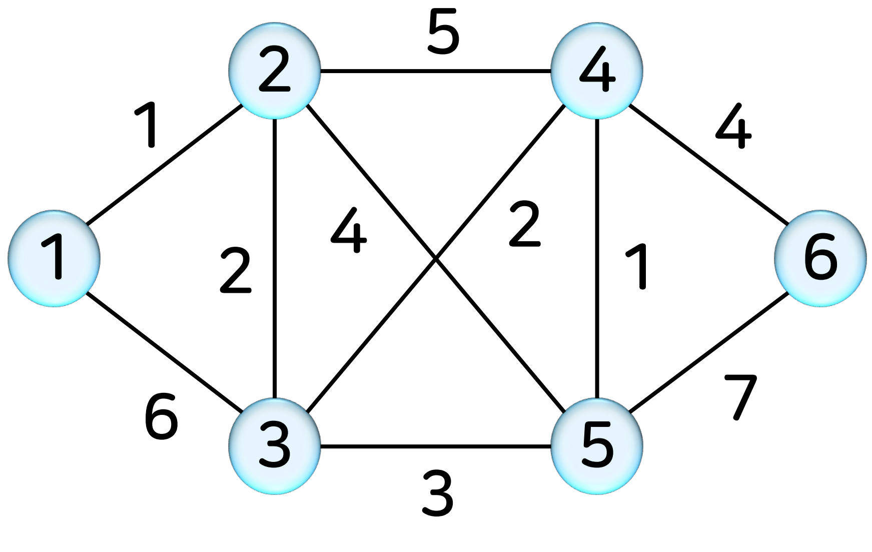
S = { 1, 2, 3, 4, 5, 6 }

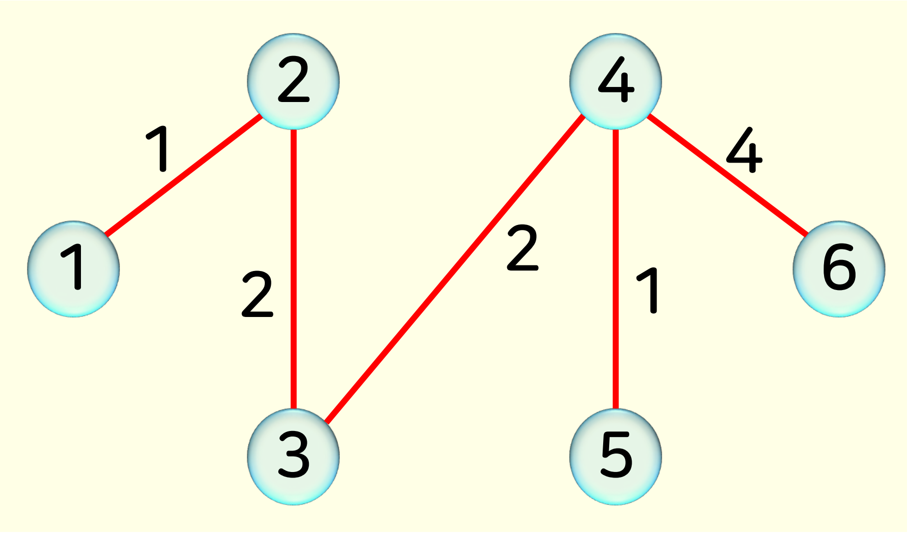
V-S = { }



가중치의 합 → 1+1+2+2+3 = 9

프림 알고리즘_예_2





가중치의 합 → 1+1+2+2+4 = 10

프림 알고리즘의 성능


```
Prim ( G )
{
  T = Ø;
  S = { 1 };      // 임의의 정점(예, 여기서는 1)으로 초기화
  while ( S != V ) {
     u∈S, v∈V-S인 것 중 가중치가 최소인 간선 (u, v) 선택;
     T = T ∪ { (u, v) };
     S = S ∪ { v };
  }
  return (T);
}
```

인접 행렬의 경우 → $O(|V|^2)$

인접 리스트와 힙을 사용한 경우 → $O((|V|+|E|)\log|V|)$

03. 최단 경로: 데이크스트라 알고리즘

최단 경로?

두 정점 u와 v간의 최단 경로 shortest path

- 가중 그래프에서 두 정점 u에서 v를 연결하는 경로 중 간선의 가중치의 합이 가장 작은 경로

최단 경로 문제의 유형

- 단일 출발점 최단 경로 single-source shortest path 문제
  - ✓ 데이크스트라 Dijkstra 알고리즘, 벨만-포드 Bellman-Ford 알고리즘
- 단일 도착점 최단 경로 문제
- 단일 쌍 최단 경로 문제
- 모든 쌍 최단 경로 all-pairs shortest path 문제
  - ✓ 플로이드 Floyd 알고리즘

데이크스트라 알고리즘?

Dijkstra, 다익스트라

- 단일 출발점 최단 경로 알고리즘

  → 하나의 출발 정점에서 다른 모든 정점으로 최단 경로를 찾는 알고리즘

  ✓ 욕심쟁이 방법이 적용된 알고리즘

  ✓ 가정 → 음의 가중치를 갖는 간선 없음

- 거리 d[v]

  ✓ 출발점에서 현재까지 선택된 정점 집합 S를 경유하여 정점 v에 이르는 최소 경로의 길이

데이크스트라 알고리즘

출발점에서 시작하여 거리 d[ ]가 최소인 정점을 차례대로 선택하여 최단 경로를 구하는 방법

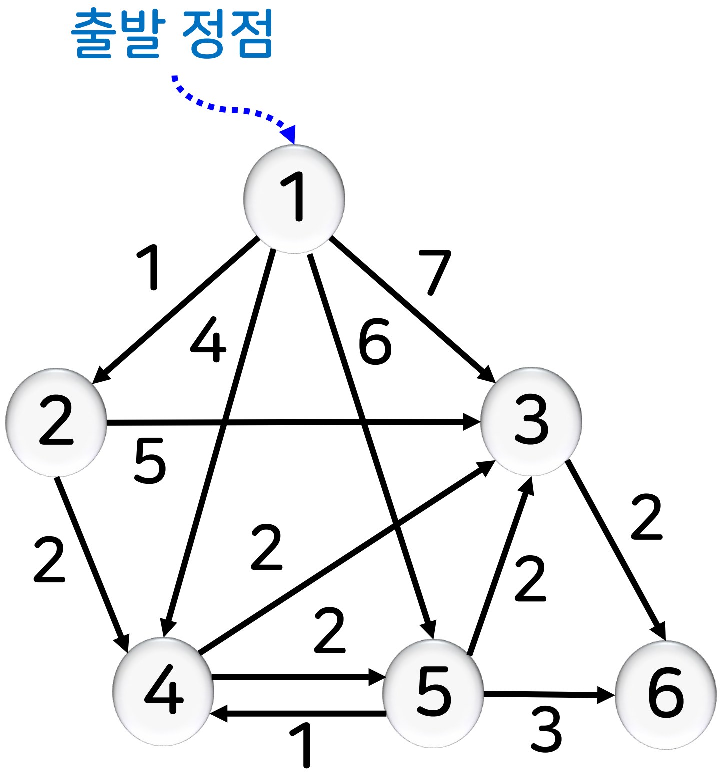
- 초기화 → 출발점 s의 거리 d[s]=0, 나머지 모든 정점 v의 거리 d[v]=∞,
  선택된 정점의 집합 S={ }
- S=V가 될 때까지 반복
  - 미선택 정점 집합 V-S에서 거리 d[ ]가 가장 작은 정점 u를 선택
  - u의 인접 정점에 대해서
    u를 경유하는 거리와 기존 거리를 비교해서
    작은 값을 새로운 거리값으로 조정

데이크스트라 알고리즘

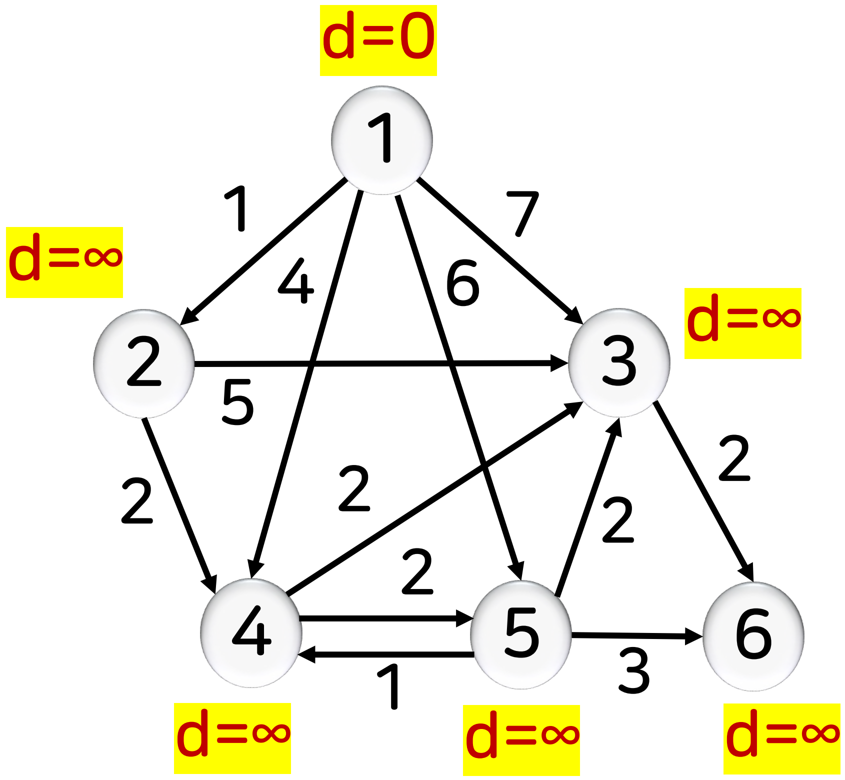
입력: G=(V,E), s : 시작 정점
출력: d[] : s로부터 다른 모든 정점으로의 최단 경로의 길이
prev[] : 최단 경로를 만드는 선행 정점

```
Dijkstra (G, s)
{
  S = { }; d[s] = 0;
  for ( 모든 정점 v∈V ) {
    d[v] = ∞;
    prev[v] = NULL;
  }
  while (S != V) {
    d[u]가 최소인 정점 u∈V-S를 선택;
    S = S ∪ { u };
    for ( u에 인접한 모든 정점 v )
      if ( d[v] > d[u] + W(u, v) ) {
        d[v] = d[u] + W(u, v);
        prev[v] = u;
      }
  }
  return (d[ ], prev[ ]);
}
```

데이크스트라 알고리즘_예_1

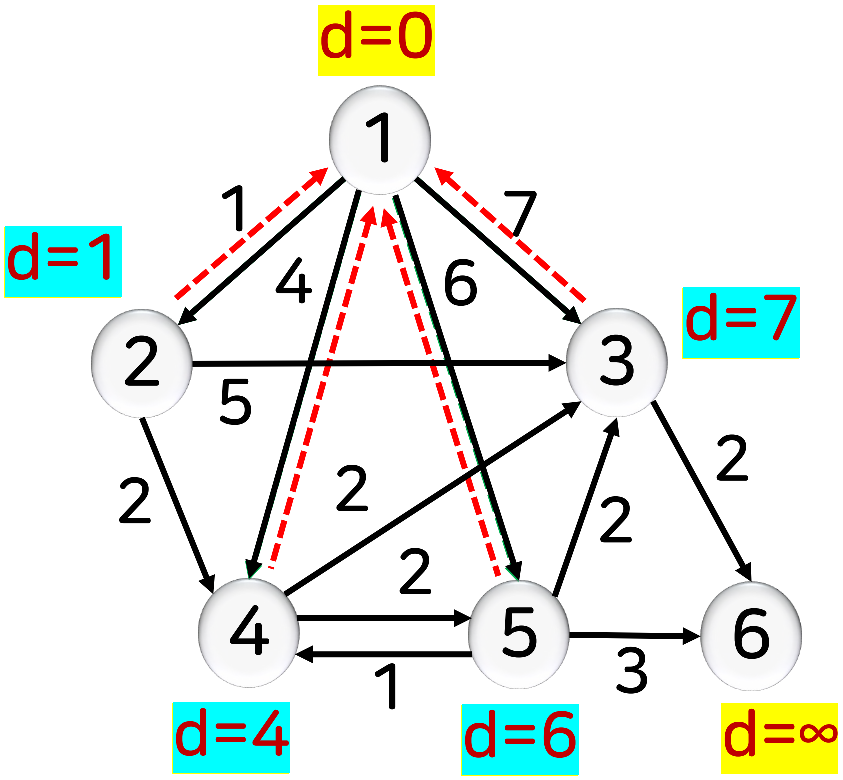


초기화



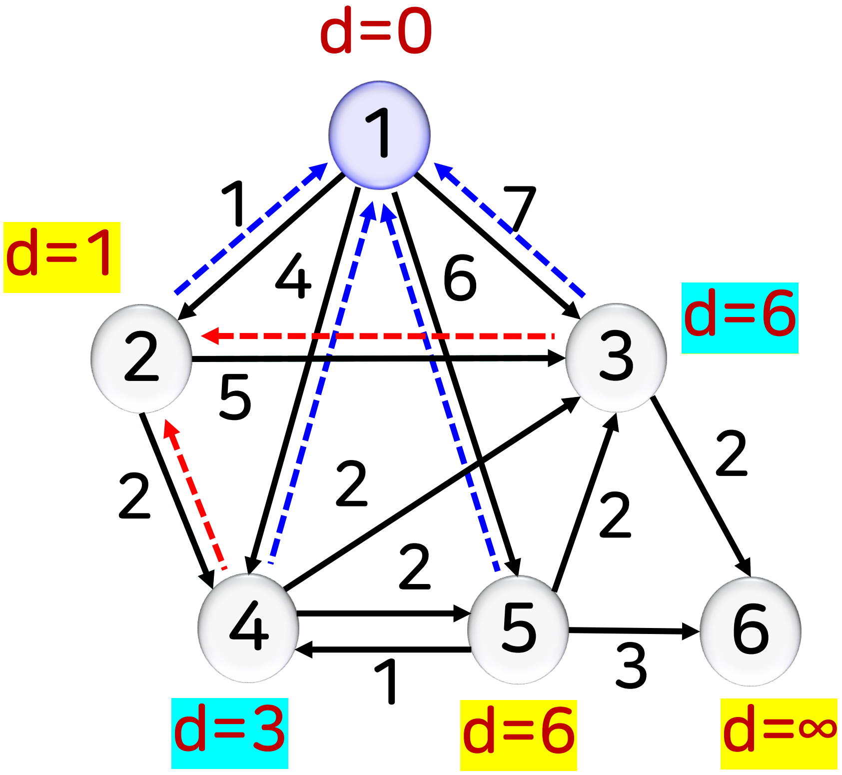
S = { }

데이크스트라 알고리즘_예_1



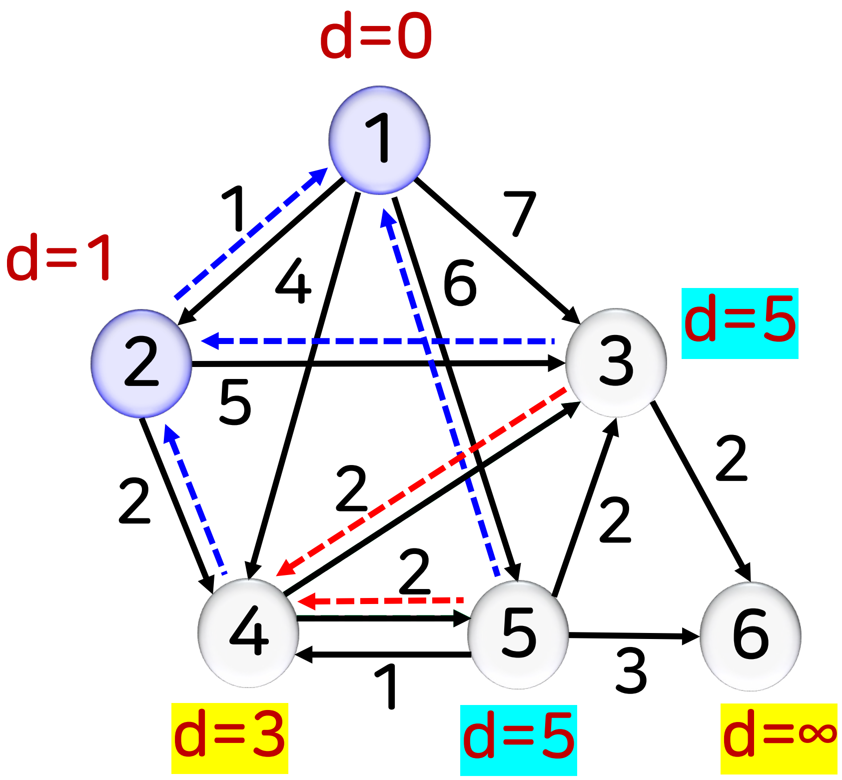
S = { 1 }

데이크스트라 알고리즘_예_1



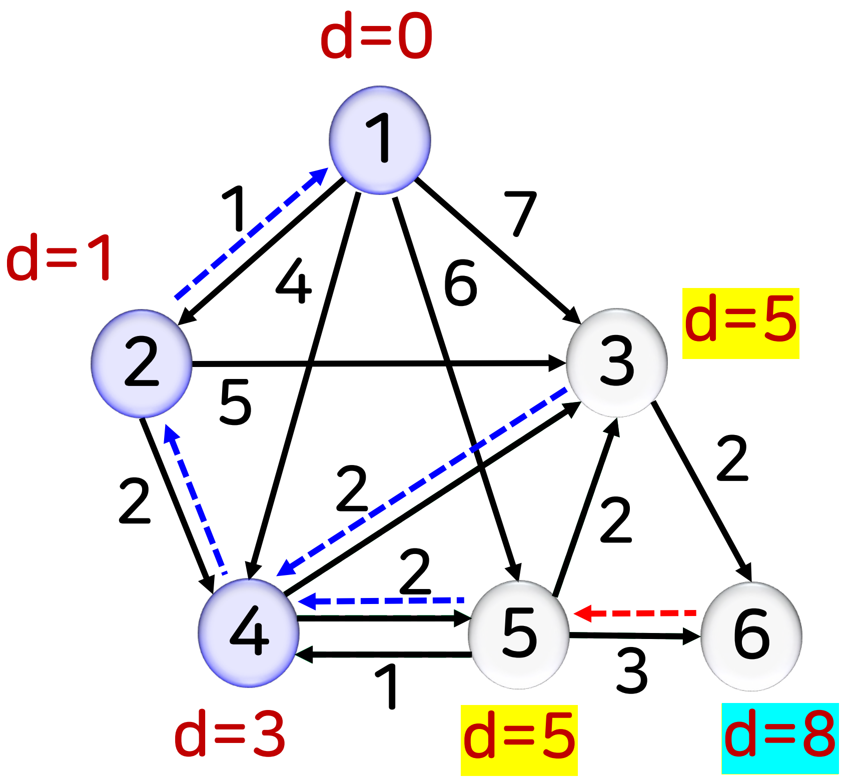
S = { 1, 2 }

데이크스트라 알고리즘_예_1



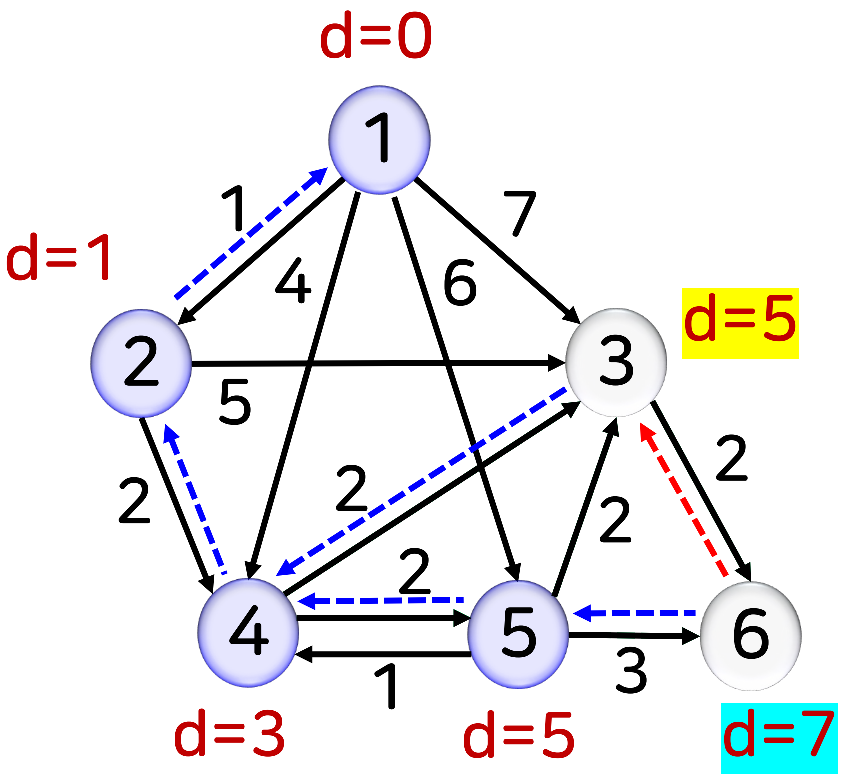
S = { 1, 2, 4 }

데이크스트라 알고리즘_예_1



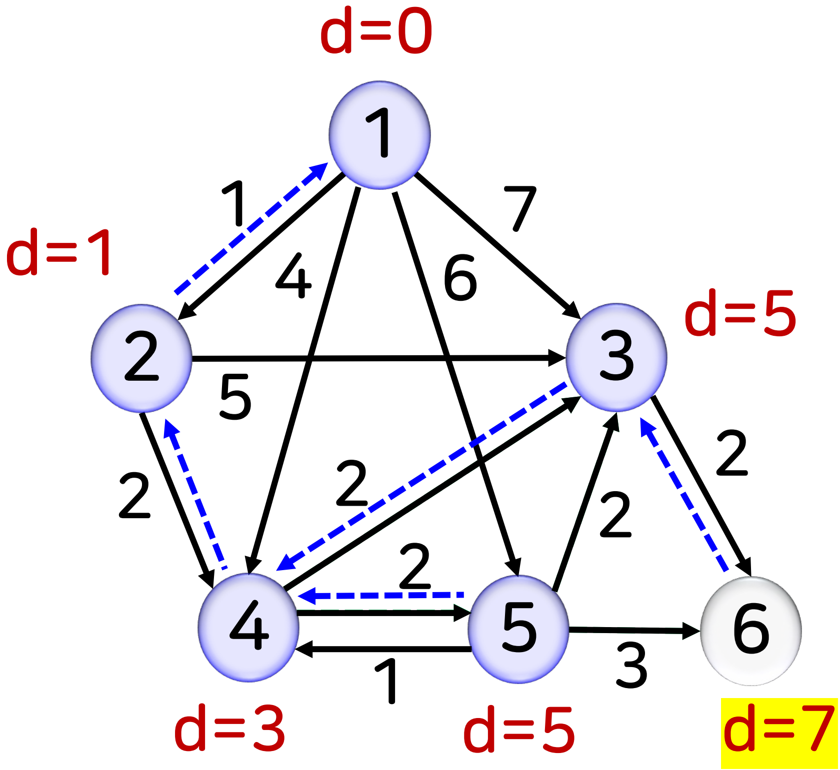
S = { 1, 2, 4, 5 }

데이크스트라 알고리즘_예_1



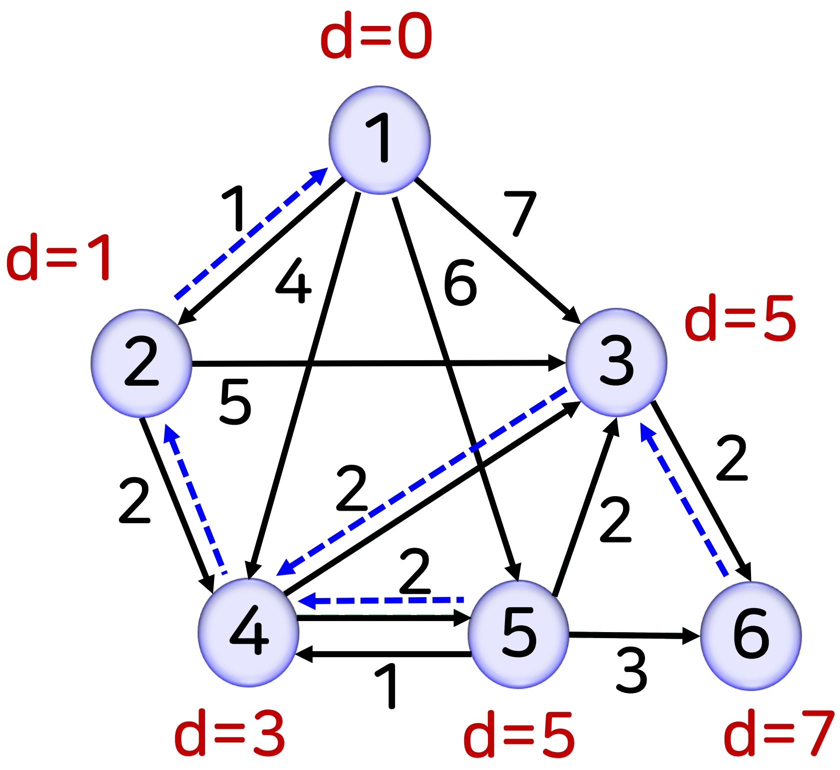
S = { 1, 2, 3, 4, 5 }

데이크스트라 알고리즘_예_1



S = { 1, 2, 3, 4, 5, 6 }

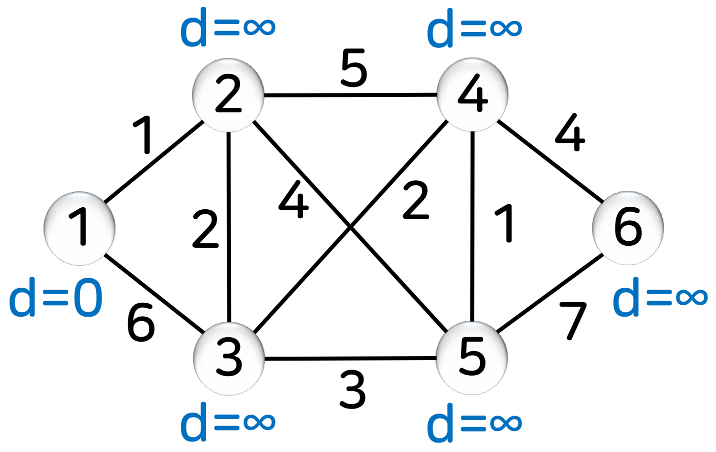
데이크스트라 알고리즘_예_1

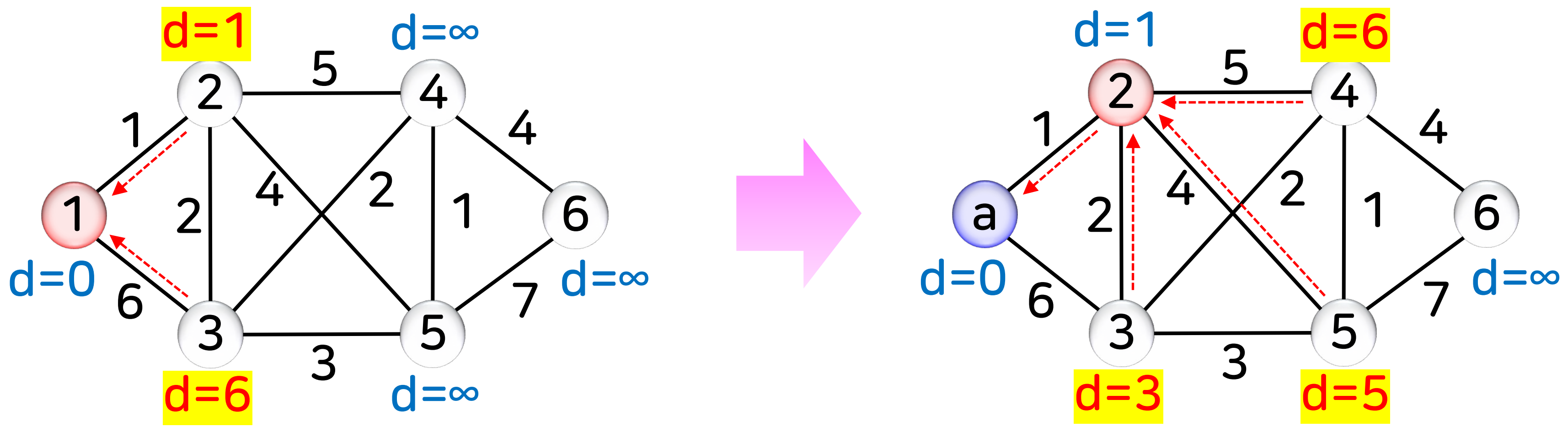


S = { 1, 2, 3, 4, 5, 6 } → 종료

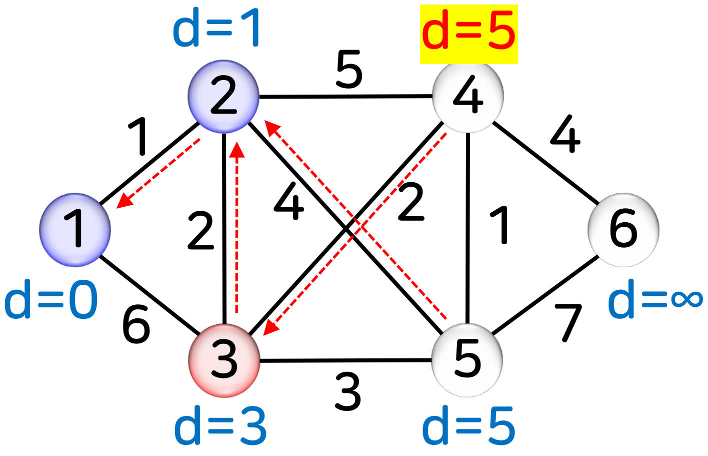
데이크스트라 알고리즘_예_2

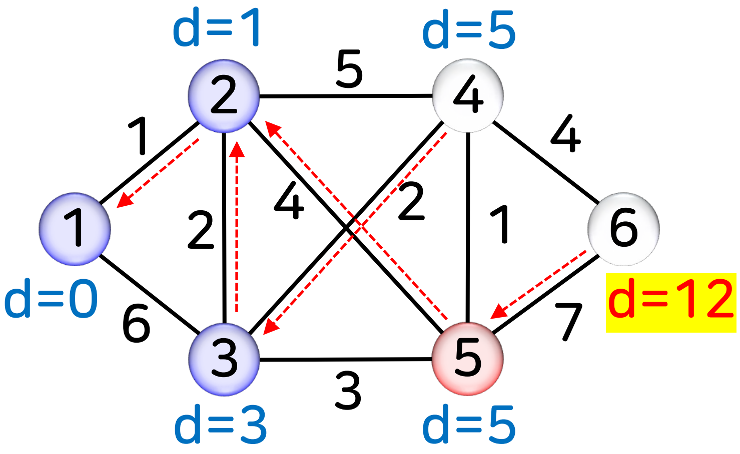
무방향 그래프

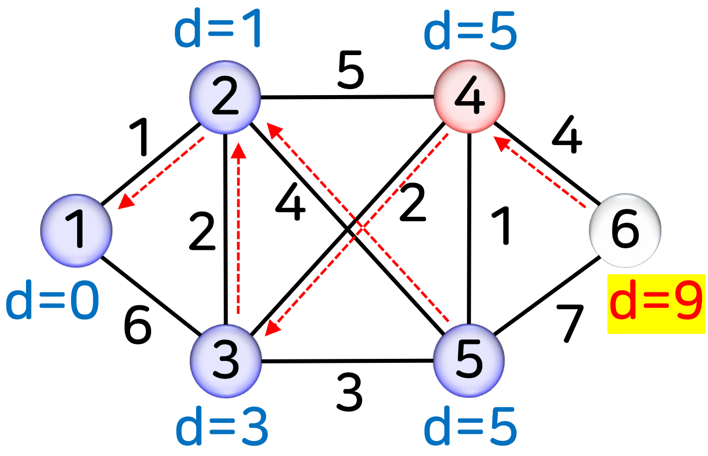


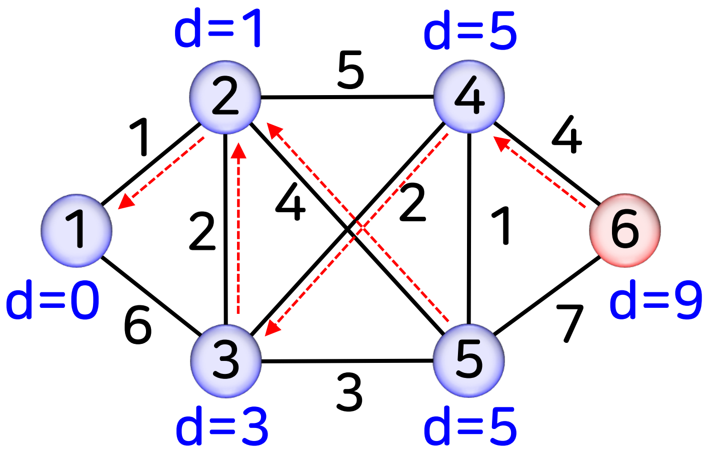


데이크스트라 알고리즘_예_2



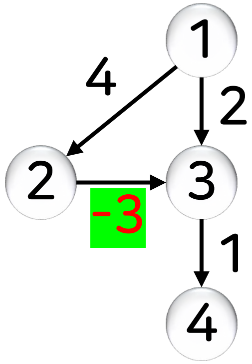


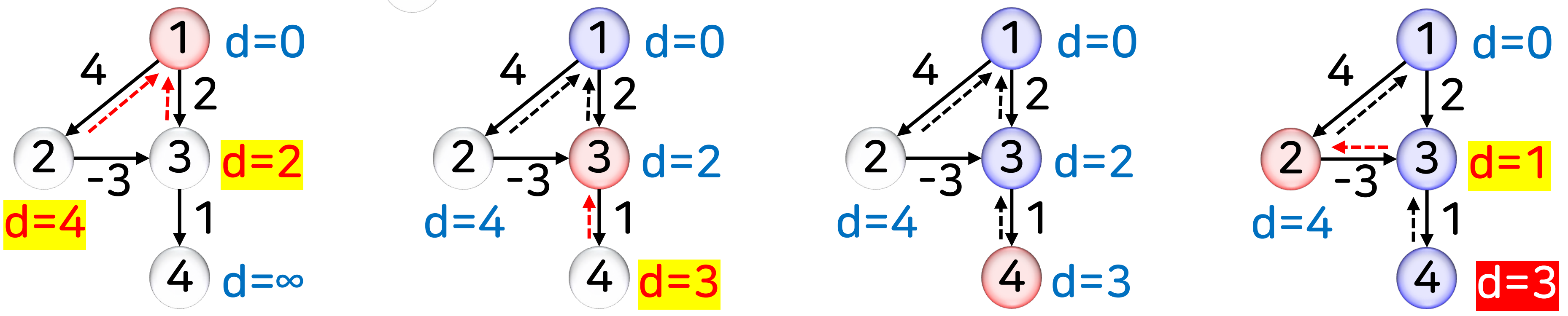




성능과 특징

- 인접 행렬 $O(|V|^2)$, 인접 리스트 + 힙 $O((|V|+|E|)\log|V|)$
- 음의 가중치를 갖는 간선이 없어야 함





$S = \{1\}$ $S = \{1,3\}$ $S = \{1,3,4\}$ $S = \{1,2,3,4\}$

정리하기

1. 크루스칼 알고리즘

- 최소 신장 트리, 욕심쟁이 방법
- 가중치가 가장 작은 간선부터 사이클을 형성하지 않으면 추가하는 방식
- $O(|E|\log|E|)$

2. 프림 알고리즘

- 최소 신장 트리, 욕심쟁이 방법
- S와 V-S를 잇는 간선 중에서 가중치가 가장 작은 간선을 선택해서 추가하는 방식
- $O((|V|+|E|)\log|V|)$, $O(|V|^2)$

3. 데이크스트라 알고리즘

- 단일 출발점 최단 경로, 욕심쟁이 방법, 음의 가중치를 갖는 간선이 없어야 함
- 하나의 출발점에서 시작하여 거리 d[ ]가 최소인 정점을 선택한 후 인접 정점에 대해 경유 거리와 기존 거리 중에서 작은 값을 새로운 거리값으로 조정
- $O((|V|+|E|)\log|V|)$, $O(|V|^2)$

ALGORITHM
알고리즘

다음시간에는

Lecture 10

그래프 (3)

컴퓨터과학과 | 이관용 교수