욕심쟁이 방법_1

2020년도 2학기 최 희석

목차

- 욕심쟁이 방법이란
- 동전 거스름돈 문제
- 배낭 문제
- 최소 신장 트리

욕심쟁이 방법이란

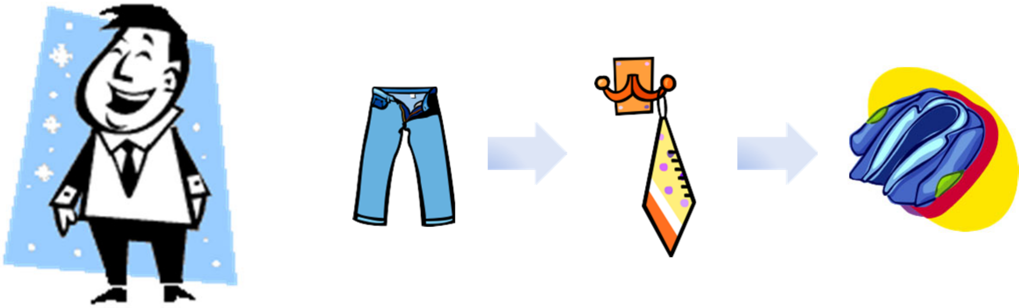
욕심쟁이 방법이란

- 해를 구하는 일련의 선택 단계마다 전후 단계의 선택과는 무관하게 해당 단계에서 **가장 최선**이라고 여겨지는 국부적인 최적해를 선택함으로써 전체적인 최적해를 구하는 방법
  - "greedy" → 탐욕적 방법, 탐욕 알고리즘, 그리디 알고리즘

동적 프로그래밍 방법	욕심쟁이 방법
최적화 문제 해결에 주로 사용	
최적성의 원리가 적용된 방법	
소문제의 여러 최적해로부터 다음 크기의 문제에 대한 최적해가 결정 → 항상 전체적인 최적해를 구함	소문제(각 단계)에 대해서 하나의 최적해만 고려 → 전체적인 최적해를 구하지 못할 수 있음

욕심쟁이 방법이란

- 욕심쟁이 방법의 한계



멋쟁이씨가 세상에서 가장 멋진 바지, 가장 멋진 넥타이, 가장 멋진 쟈켓을 입는다고 할 때, 그렇다면 그런 옷을 입은 멋쟁이씨는 세상에서 가장 멋쟁이라고 할 수 있는가?

욕심쟁이 방법으로 해를 구할 수 없는 문제도 많다.

동전 거스름돈 문제

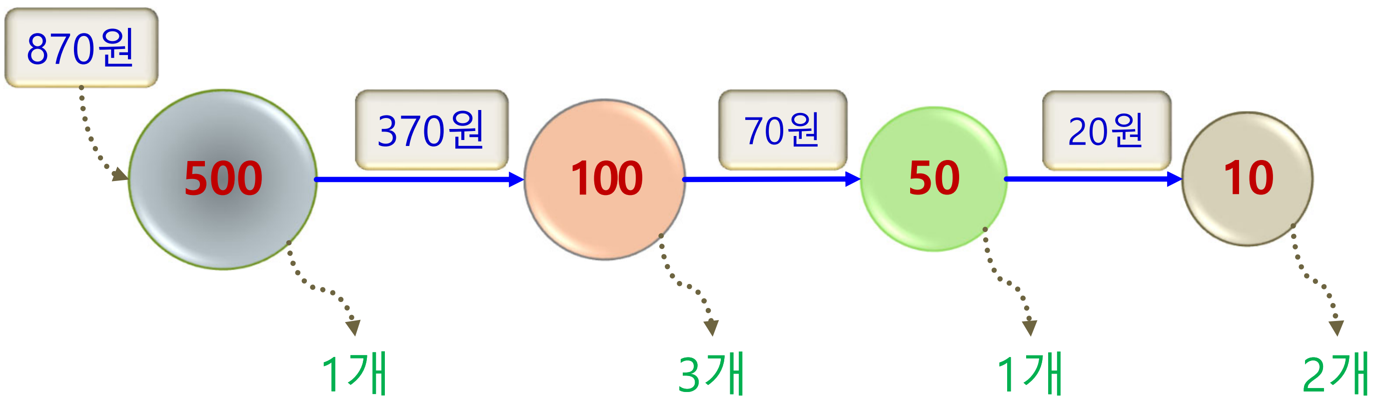
동전 거스름돈 문제의 개념과 원리

- 고객에게 돌려줄 거스름돈이 있을 때 고객이 받을 동전의 개수를 최소로 하여 거스름돈을 돌려주는 방법을 찾는 문제
  - "동전 문제", "거스름돈 문제"
  - 동전의 종류 → 500원, 100원, 50원, 10원

- 기본 아이디어
  - 거스름돈의 액수를 초과하지 않는 조건하에서 단순히 액면가가 가장 큰 동전부터 '욕심을 부려서' 최대한 사용해서 거스름 돈을 만듦

동전 거스름돈 문제의 개념과 원리

- 고객에게 돌려줄 거스름돈이 T만큼 있을 때 고객이 받을 동전의 개수를 최소로 하면서 거스름돈을 돌려주는 방법을 찾는 문제



동전 거스름돈 문제의 알고리즘과 성능 분석

```
CoinChange (T, n, C[], X[])
입력: T : 고객에게 돌려줄 거스름돈, n : 동전의 종류
      C[0..n-1] : 동전의 액면가를 감소순으로 저장(500→100→50→10)
출력: X[0..n-1] : 거스름돈으로 돌려줄 i번째 동전의 개수
{
   for (i=0; i<n; i++) {
      X[i] = ⌊T/C[i]⌋;
      T = T - C[i]*X[i];
   }
}
```

O(n)

동전 거스름돈 문제의 특징

- 동전의 액면가가 임의로 주어지는 일반적인 경우
  - 동전의 종류: 500원, **120원**, 100원, 50원, 10원
    - 거스름돈 650원에 대해서 욕심쟁이 방법을 적용하면

500원짜리 동전 → 1개

120원짜리 동전 → 1개

10원짜리 동전 → 3개

5개의 동전이 필요

최적해 → 500원x 1개, 100원x 1개, 50원x 1개 ⇒ 3개

일반적인 경우의 거스름돈 문제는 욕심쟁이 방법으로 해결 불가

배낭 문제

배낭 문제의 개념과 원리

- 최대 용량 M인 하나의 배낭과 n개의 물체
  - 각 물체 i 에는 물체의 무게 w_i 와
    해당 물체를 배낭에 넣을 때 얻을 수 있는 이익 pi

- 배낭의 용량을 초과하지 않는 범위 내에서 배낭에 들어 있는 물체의 이익의 합이 최대가 되도록 물체를 넣는 방법 (또는 최대 이익)을 구하는 문제
  - **물체를 쪼개서 넣을 수 있다고 가정**

배낭 문제의 개념과 원리

- 기본 아이디어
  - 물체의 무게는 적으면서도 이익이 가장 큰 물체부터 골라서 '**욕심을 내어**' 최대한 넣음
  - **단위 무게당 이익**이 가장 큰 물체부터 최대한 배낭에 넣는 과정을 반복, 물체를 통째로 넣을 수 없으면 남은 배낭의 용량만큼 물체를 쪼개서 넣음


용량 10kg


4kg, 이익 14


3kg, 이익 15


5kg, 이익 20


3kg, 이익 9

배낭 문제의 알고리즘

```
Knapsack (p[], w[], M, n, x[])
{
   for (i=0; i<n; i++) x[i] = 0;
   r = M;
   for (i=0; i<n; i++)
      if ( w[i] <= r ) {
         x[i] = 1;
         r = r - w[i];
      }
      else break;
   if ( i<n ) x[i] = r / w[i];
}
```

입력: p[i], w[i] : 물체 i의 이익과 무게

(단위 무게당 이익이 감소하는 순으로 정렬)

n, M : 각각 물체의 개수와 배낭의 용량

출력: X[i] : 배낭에 들어간 물체 i의 비율

배낭 문제의 적용 예

용량 10kg

4kg, 이익 14

3kg, 이익 15

5kg, 이익 20

3kg, 이익 9

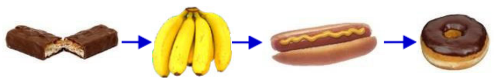
M = 10, n = 4

$(p_1, p_2, p_3, p_4) = (14, 15, 20, 9)$

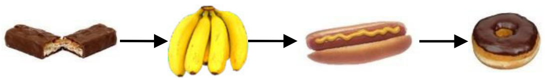
$(w_1, w_2, w_3, w_4) = (4, 3, 5, 3)$

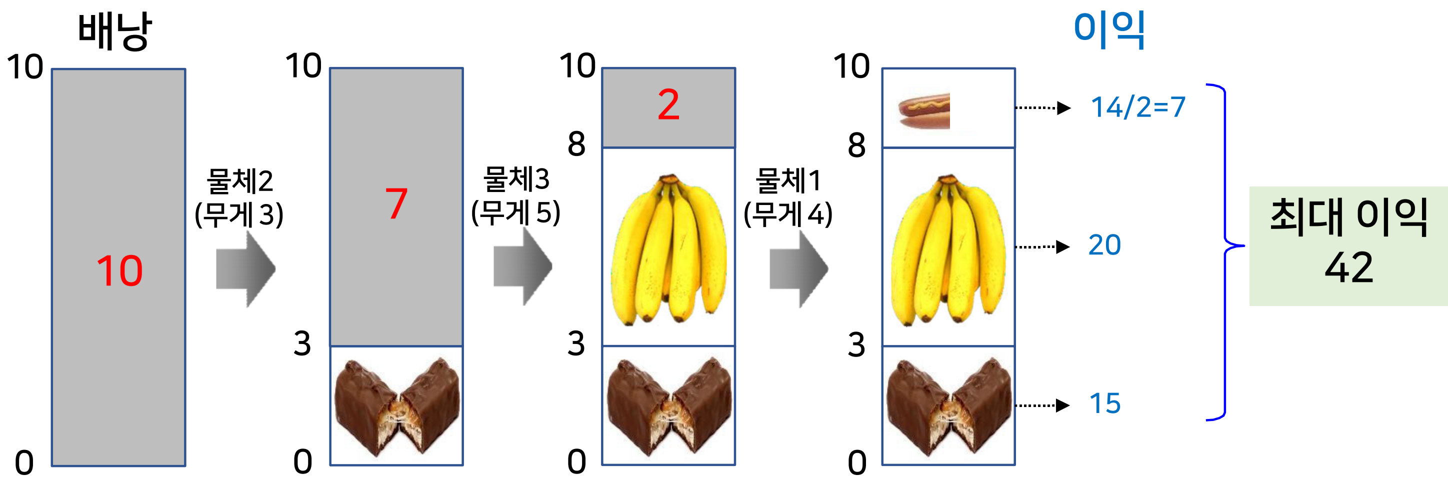
단위 무게당 이익

$$\left(\frac{p_1}{w_1}, \frac{p_2}{w_2}, \frac{p_3}{w_3}, \frac{p_4}{w_4}\right) = (3.5,5,4,3)$$



배낭 문제의 적용 예





배낭
10
10
0
물체2
(무게 3)
10
7
3
0
물체3
(무게 5)
10
2
8
3
0
물체1
(무게 4)
10
8
3
0
이익
14/2=7
20
15
최대 이익
42

배낭 문제의 성능 분석

```
Knapsack (p[], w[], M, n, x[])
{
   for (i=0; i<n; i++)
     x[i] = 0;
   r = M;
   for (i=0; i<n; i++)
      if ( w[i] <= r ) {
         x[i] = 1;
         r = r - w[i];
      }
      else break;
   if ( i<n ) x[i] = r / w[i];
}
```

O(n)

O(n)

입력: p[i], w[i] : 물체 i의 이익과 무게
(단위 무게당 이익이 감소하는 순으로 정렬)
n, M : 각각 물체의 개수와 배낭의 용량
출력: X[i] : 배낭에 들어간 물체 i의 비율

정렬을 고려한다면 ➡ O(nlogn)

배낭 문제

- 0/1 배낭 문제 **-> 욕심쟁이 방법 적용 불가**
  - 물체를 쪼갤 수 없는 형태의 배낭 문제

M = 10, n = 4 $(p_1, p_2, p_3, p_4) = (14, 15, 20, 9)$ $(w_1, w_2, w_3, w_4) = (4, 3, 5, 3)$

단위 무게당 이익 $\left(\frac{p_1}{w_1}, \frac{p_2}{w_2}, \frac{p_3}{w_3}, \frac{p_4}{w_4}\right) = (3.5, 5, 4, 3)$

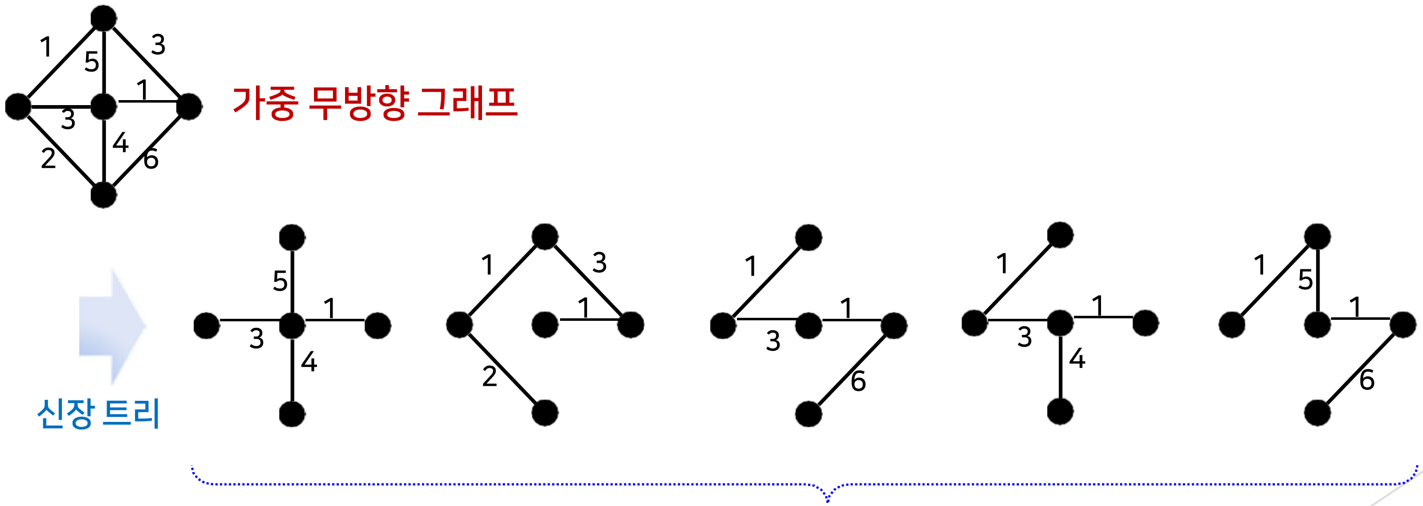
욕심쟁이 방법 적용 최대 이익 $= p_2 + p_3 =$ 35

실제 최대 이익 $= p_1 + p_2 + p_4 =$ 38

최소 신장 트리

최소 신장 트리의 개념과 원리

- 신장 트리 spanning tree
  - 가중 무방향 그래프에서 모든 정점을 포함하는 연결된 트리



정점이 n개이면, 트리에는 n-1개의 간선이 존재

최소 신장 트리의 개념과 원리

- 최소 (비용) 신장 트리minimum 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′)

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

  - 신장 트리 중에서 간선의 가중치의 합이 가장 작은 트리

최소 신장 트리의 개념과 원리

- 최소 신장 트리 알고리즘
  - 크루스칼Kruskal 알고리즘
  - 프림Prim 알고리즘

↓

욕심쟁이 방법

```
Greedy_MST ( G=(V, E) )
{
  T ← Ø ;
  while ( T가 신장 트리를 만들지 않았음 ) {
     최선의 간선 (u, v)를 선택;
     T ← T ∪ { (u, v) };
  }
  return (T);
}
```

사이클을 형성하지 않으며 최소의 가중치를 갖는 간선

크루스칼 알고리즘의 개념과 원리

- 간선이 하나도 없는 상태에서 시작해서 가중치가 가장 작은 간선부터 하나씩 사이클을 만들지 않으면 추가시키는 방법
- 서로 다른 연결 성분에 속하는 정점을 잇는 최소 가중치의 간선을 선택
  - n개의 정점이 서로 다른 연결 성분으로 구성된 상태에서 시작해서,
    간선이 추가될 때마다 연결 성분들이 합쳐지게 되고,
    최종적으로 하나의 연결 성분을 형성

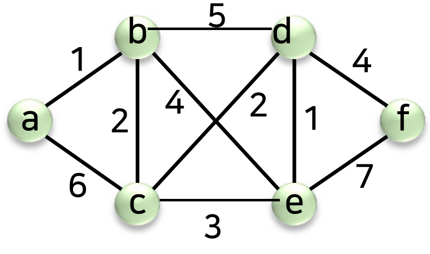
크루스칼 알고리즘

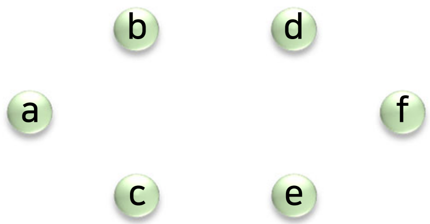
입력: G : 가중 무방향 그래프 G=(V,E)

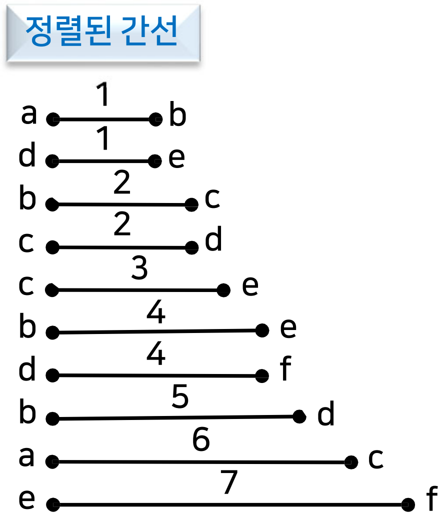
출력: T : 최소 신장 트리의 간선 집합 T

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

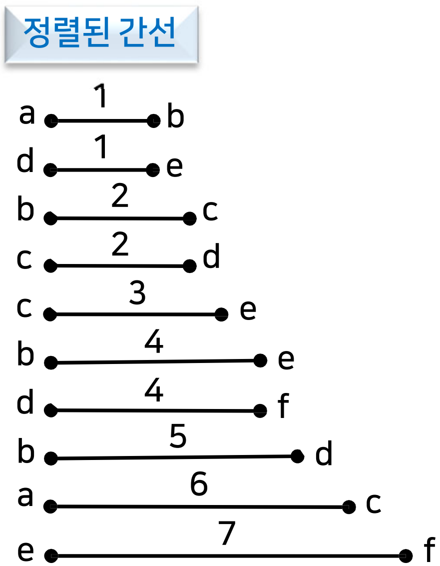
크루스칼 알고리즘의 적용 예

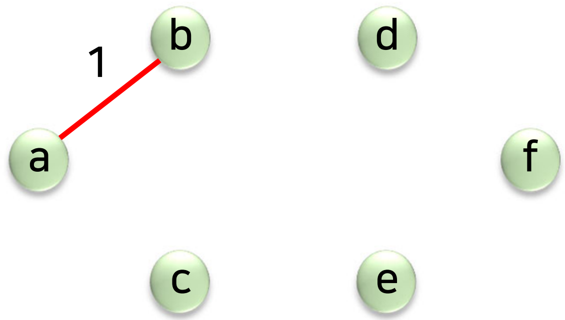




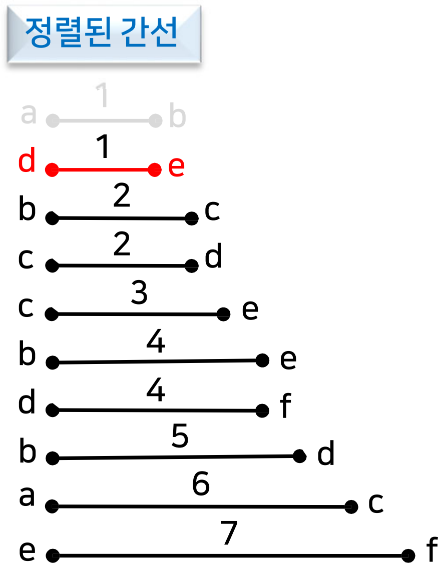


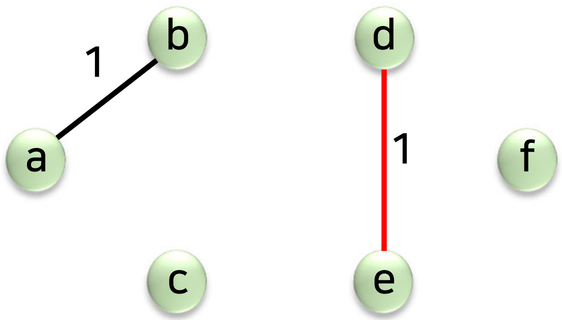
크루스칼 알고리즘의 적용 예



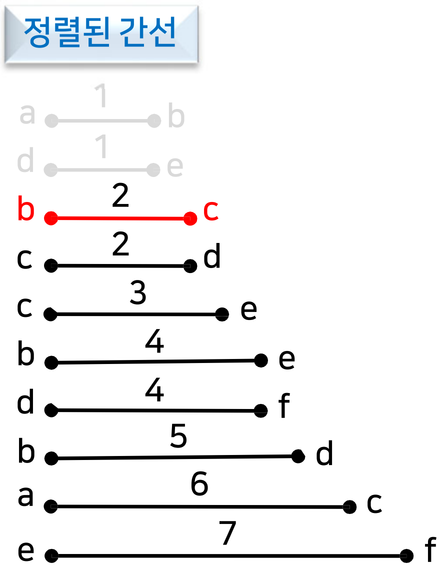


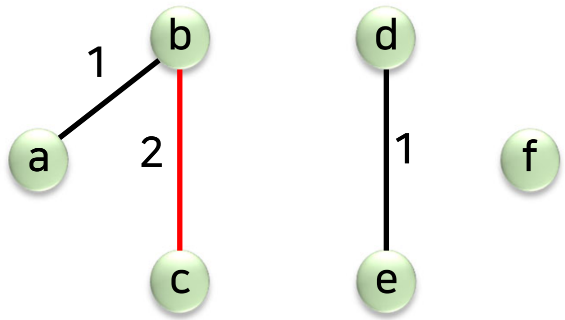
크루스칼 알고리즘의 적용 예



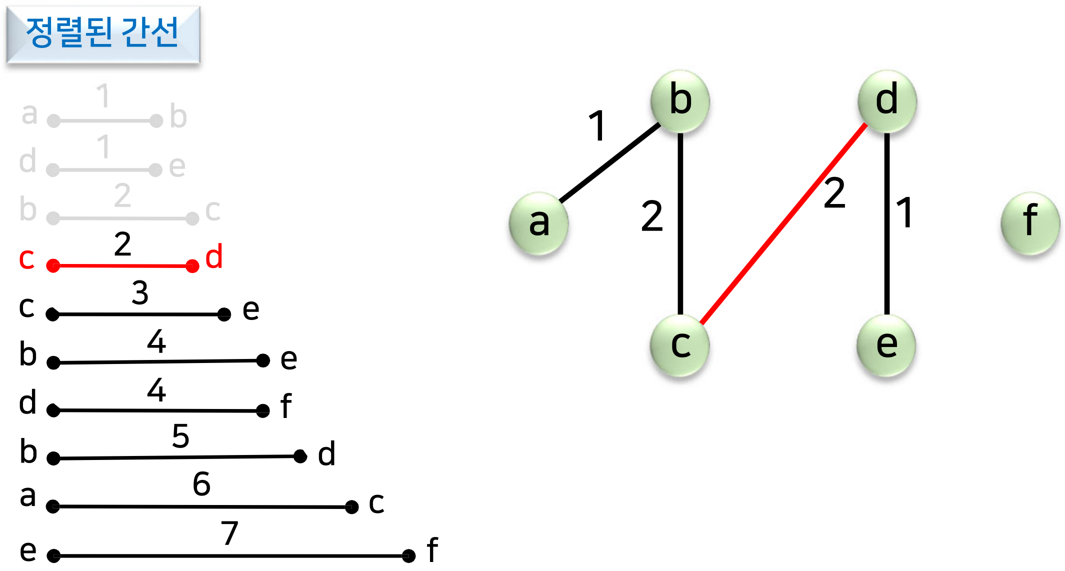


크루스칼 알고리즘의 적용 예

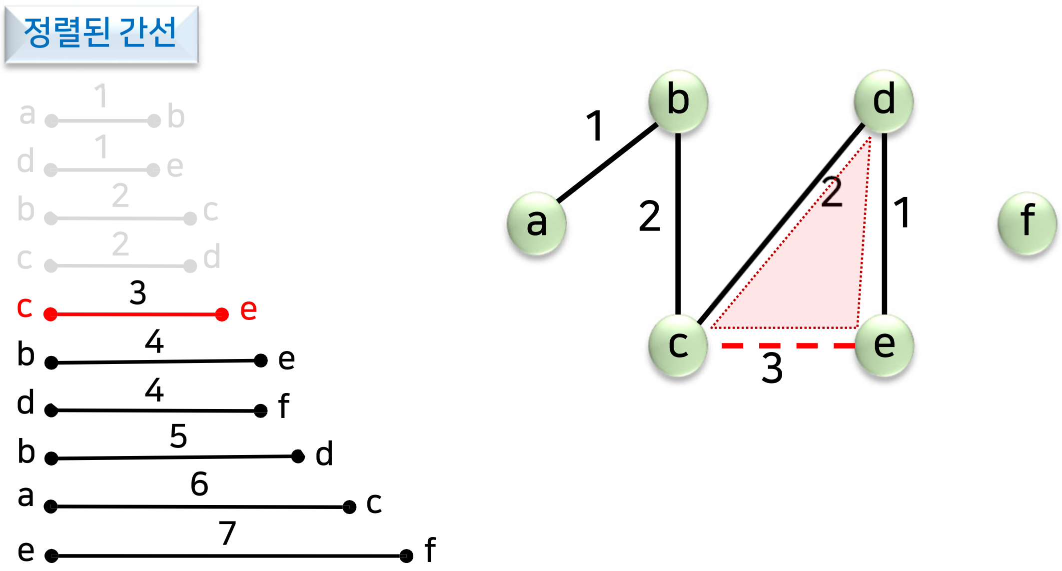




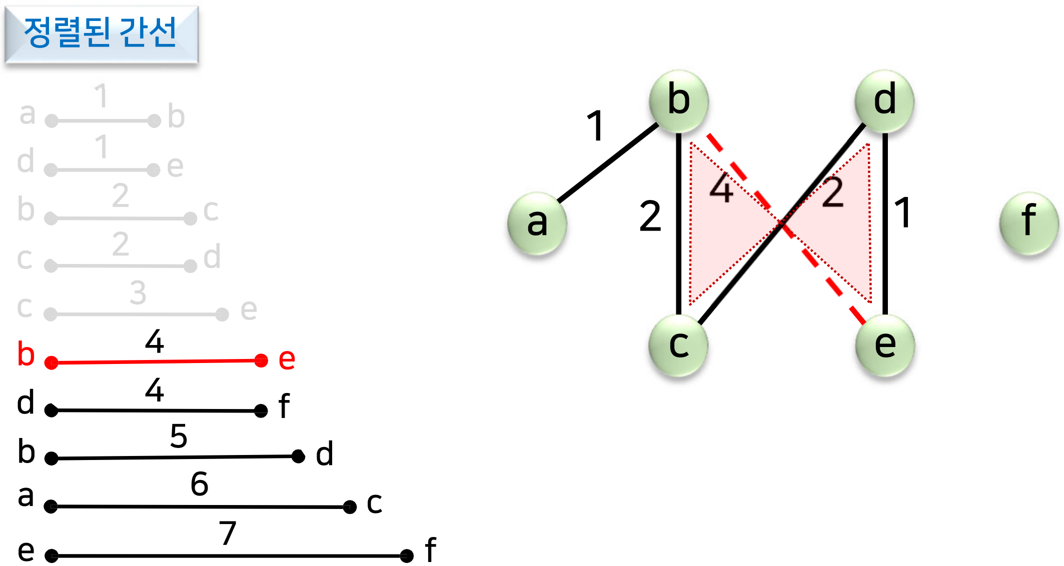
크루스칼 알고리즘의 적용 예



크루스칼 알고리즘의 적용 예

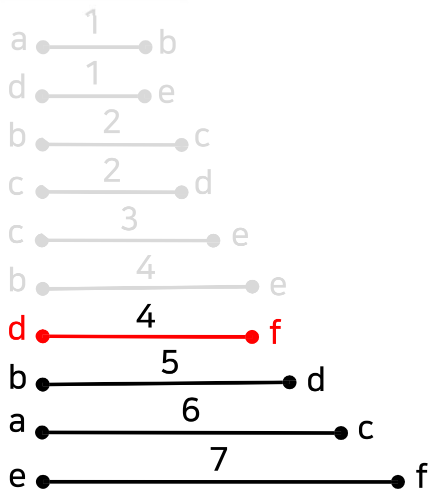


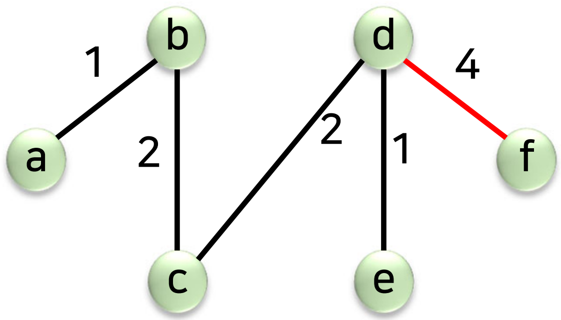
크루스칼 알고리즘의 적용 예



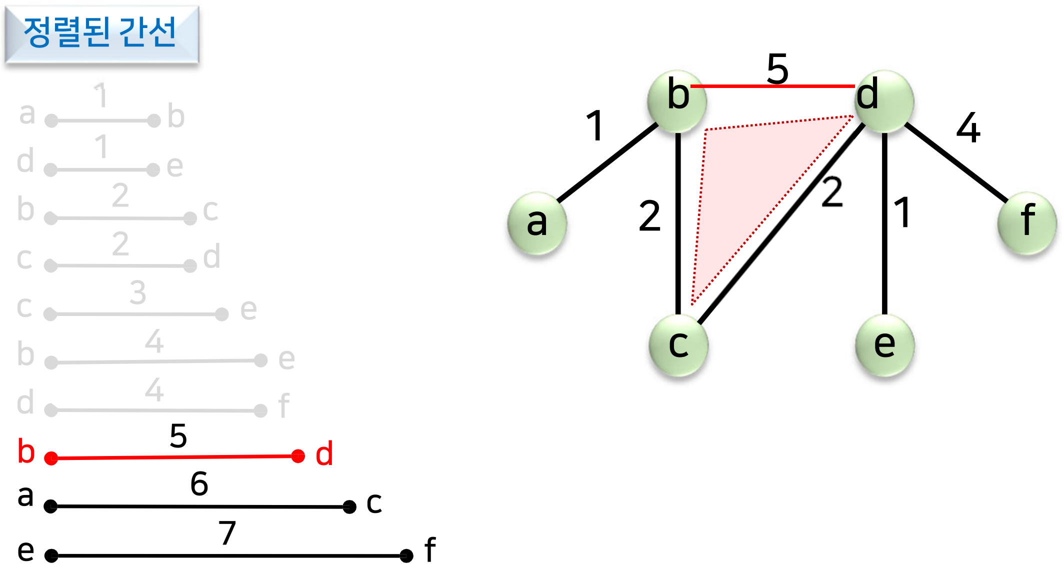
크루스칼 알고리즘의 적용 예

정렬된 간선

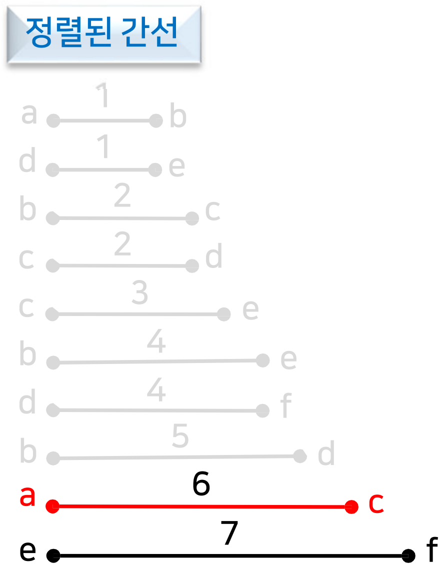


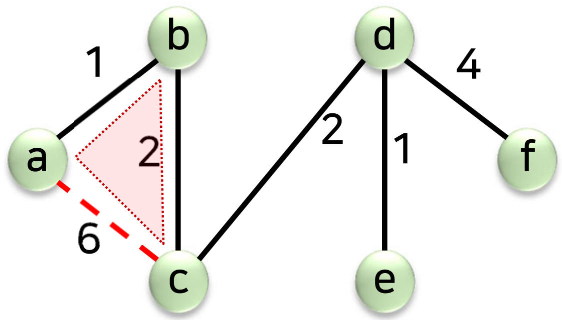


크루스칼 알고리즘의 적용 예

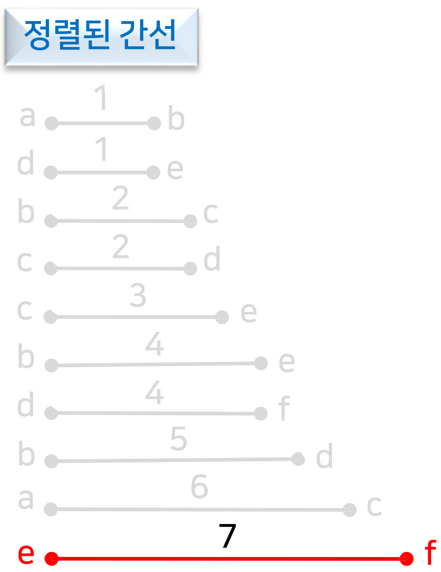


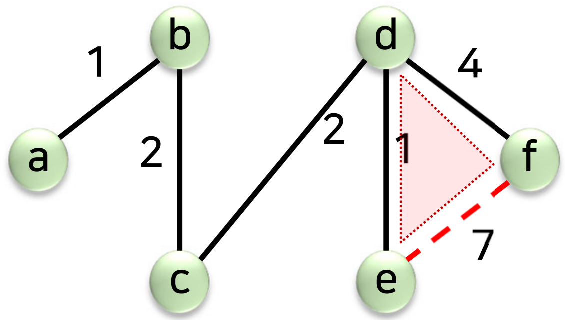
크루스칼 알고리즘의 적용 예





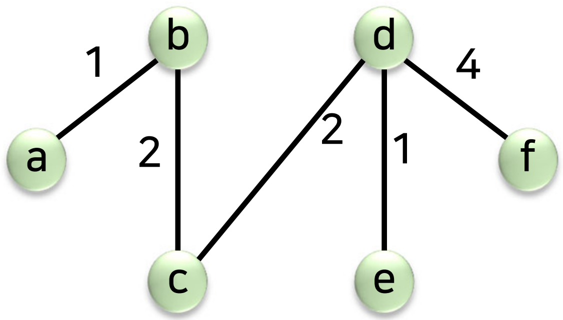
크루스칼 알고리즘의 적용 예





크루스칼 알고리즘의 적용 예

정렬된 간선



가중치의 합: 1+1+2+2+4=10

크루스칼 알고리즘의 성능 분석

```
Kruskal ( G )
{
   T = Ø;
   for (각 정점 v에 대해)
     정점 v로 구성된 연결 성분 초기화;
   가중치가 증가하는 순으로 모든 간선을 정렬;
   for ( 가중치가 가장 작은 간선부터 모든 간선 (u,v)∈E에 대해서 )
     if ( u와 v가 다른 연결 성분에 있으면 ) {
       T = T ∪ { (u,v) };
       u의 연결 성분과 v의 연결 성분을 합침;
     }
     else 간선 (u,v)를 버림;
   return T;
}
```

O(|V|)

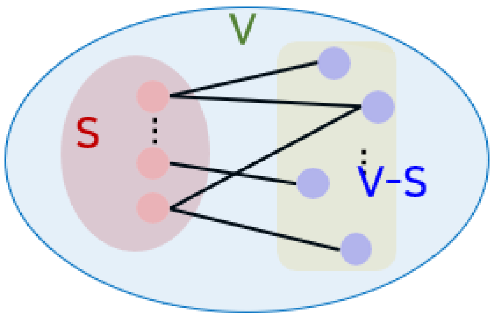
O(|E|log|E|)

O(|E|α(|E|,|V|))

O(|E| log|E|)

프림 알고리즘의 개념과 원리

- 임의의 한 정점에서 시작해서 연결된 정점을 하나씩 선택해 나가는 방법
  - 이미 선택된 정점에 부수된 가중치가 최소인 간선을 추가하는 방법
    - 이미 선택된 정점의 집합 S와 미선택 정점의 집합 V-S를 잇는 간선 중에서 가중치가 최소인 간선을 선택해서 추가해 나가는 방법



프림 알고리즘의 알고리즘

입력: G : 가중 무방향 그래프 G=(V,E)

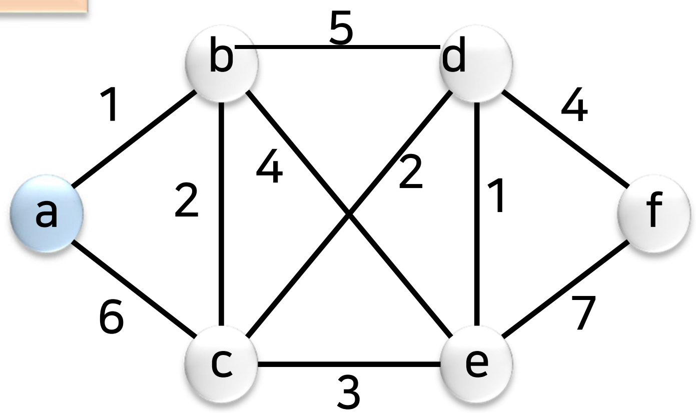
출력: T : 최소 신장 트리의 간선 집합 T

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

프림 알고리즘의 적용 예

S = { a }

V-S = { b, c, d, e, f }



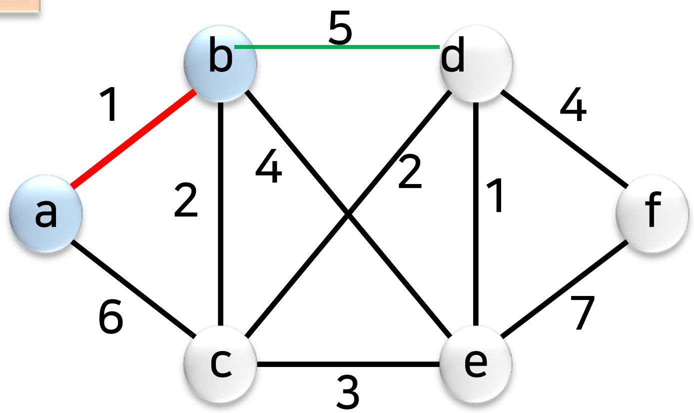
S = { a, b }

V-S = { c, d, e, f }

프림 알고리즘의 적용 예

S = { a, b }

V-S = { c, d, e, f }



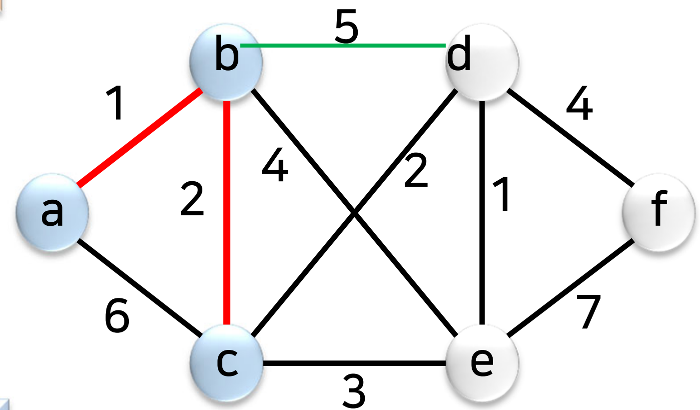
S = { a, b, c }

V-S = { d, e, f }

프림 알고리즘의 적용 예

S = { a, b, c }

V-S = { d, e, f }



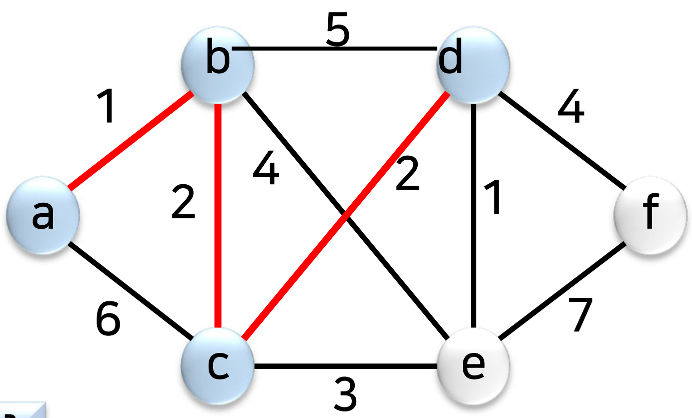
S = { a, b, c, d }

V-S = { e, f }

프림 알고리즘의 적용 예

S = { a, b, c, d }

V-S = { e, f }



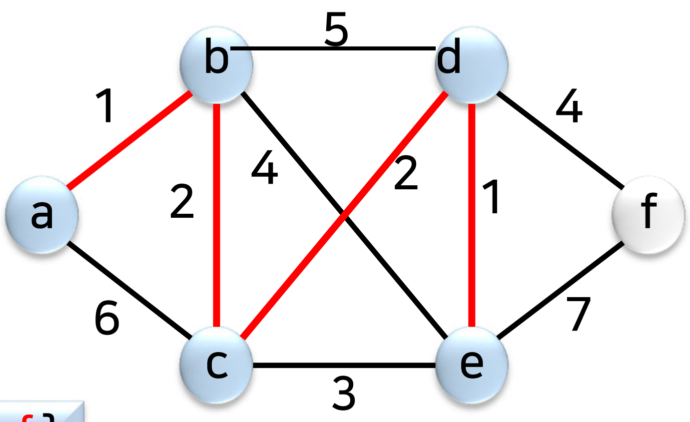
S = { a, b, c, d, e }

V-S = { f }

프림 알고리즘의 적용 예

S = { a, b, c, d, e }

V-S = { f }



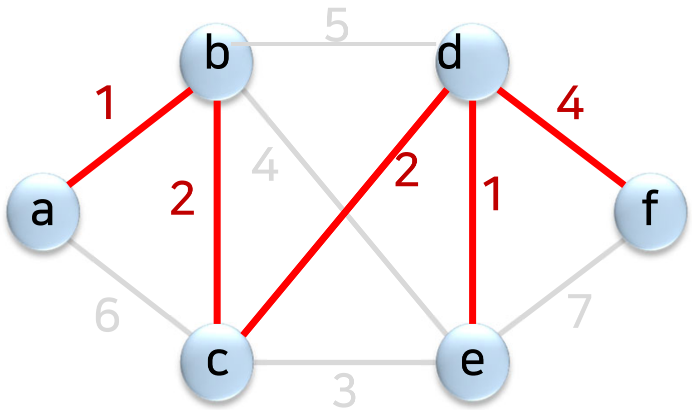
S = { a, b, c, d, e, f }

V-S = { }

프림 알고리즘의 적용 예

S = { a, b, c, d, e, f } = V = { a, b, c, d, e, f }

V-S = { }



가중치의 합: 1+1+2+2+4=10

프림 알고리즘의 성능 분석

```
Prim ( G )
{
   T = Ø;
   S= { a };
   while ( S != V ) {
       u∈S, v∈V-S 인  것 중 가중치가 최소인 간선(u,v) 선택;
       T = T∪  { (u,v) };
       S= S∪  { v };
   }
   return T;
}
```

인접 행렬의 경우: $O(|V|^2)$

인접 리스트 + 힙 사용: $O((|V|+|E|) \log|V|)$

과제 안내

7주차 과제

- 크루스칼 알고리즘, 프림 알고리즘 구현하기
  - e-Class 업로드
- 양식 (한글, 워드,PDF -> 자유)
- 파일명 (이름_학번_전공)
  - 예) 최희석_2014182009_게임공학

- 질의 응답은 e-Class 질의응답 게시판에 남겨 주시길 바랍니다.