# Ch.11 탐욕 알고리즘(Greedy Alg.)

# 학습목표

- ✓ 그리디 알고리즘의 특징을 파악한다.
- ✓ 그리디 알고리즘으로 최적해가 보장되는 예와 그렇지 않은 예를 관찰한다.
- ✓ 매트로이드의 정의를 익힌다.
- ✓ 매트로이드가 만드는 문제 공간의 특성을 배운다.

타로는 자주 JOI잡화점에서 물건을 산다. JOI잡화점에는 잔돈으로 500엔, 100엔, 50엔, 10엔, 5엔, 1엔이 충분히 있고, 언제나 거스름돈 개수가 가장 적게 잔돈을 준다. 타로가 JOI잡화점에서 물건을 사고 카운터에서 1000엔 지폐를 한장 냈을 때, 받을 잔돈에 포함된 잔돈의 개수를 구하는 프로그램을 작성하시오.

## 입력

입력은 한줄로 이루어져있고, 타로가 지불할 돈(1 이상 1000미만의 정수) 1개가 쓰여져있다.

## 출력

제출할 출력 파일은 1행으로만 되어 있다. 잔돈에 포함된 매수를 출력하시오.

### 예제 입력 1 복사

```
380
```

### 예제 출력 1 복사

```
4
```

### 예제 입력 2 복사

```
1
```

### 예제 출력 2 복사

```
15
```

타로는 자주 JOI잡화점에서 물건을 산다. JOI잡화점에는 잔돈으로 500, 400, 100, 70, 10, 1 엔이 충분히 있고, 언제나 거스름돈 개수가 가장 적게 잔돈을 준다. 타로가 JOI잡화점에서 물건을 사고 카운터에서 1000엔 지폐를 한장 냈을 때, 받을 잔돈에 포함된 잔돈의 개수를 구하는 프로그램을 작성하시오.

## 입력

입력은 한줄로 이루어져있고, 타로가 지불할 돈(1 이상 1000미만의 정수) 1개가 쓰여져있다.

## 출력

제출할 출력 파일은 1행으로만 되어 있다. 잔돈에 포함된 매수를 출력하시오.

### 예제 입력 1 복사

```
380
```

### 예제 출력 1 복사

```
4
```

### 예제 입력 2 복사

```
1
```

### 예제 출력 2 복사

```
15
```

# Preview


사람이 살면서
눈앞의 이익만 좇으면
대개 결과가 안 좋지,
알고리즘도
그런가요?
대체로 그런데,
일부 알고리즘은 눈앞의
이익만 좇아가면 최적해를
구하게 되는 경우가 있긴 하지,
이미 배운 것 중에
그런 예가 있어,
최소 신장 트리를 구하는
알고리즘은 둘 다 그런 것 같고,
최단 경로를 구하는
다익스트라 알고리즘도?
빙 고

# 전형적인 그리디 알고리즘의 구조

- 눈앞의 이익만 취하고 보는 알고리즘
- 현재 시점에 가장 이득이 되어 보이는 해를 선택하는 행위를 반복한다
- 대부분 최적해와의 거리가 멀다
- 드물게 최적해가 보장되는 경우도 있다

**알고리즘 11-1** 전형적인 그리디 알고리즘

```
do
    가장 좋아 보이는 선택을 한다.
until (해 구성 완료)
```

1. 선택절차(Selection Procedure) : 현재 상태에서의 최적의 해답을 선택
2. 적절성 검사(Feasibility Check) : 선택된 해가 문제의 조건에 만족하는지 검사
3. 해답 검사(Solution Check) : 원래의 문제가 해결되었는지 검사하고, 해결되지 않았다면 선택 절차로 돌아가서 다시 반복

# 그리디 알고리즘을 쓰기 위해서는

- **일반적으로 그리디 알고리즘이 잘 동작하는 것은 두 조건을 만족**
  - 탐욕스러운 선택 조건(greedy choice property)
    - 앞의 선택이 이후의 선택에 영향을 주지 않는다.
  - 최적 부분 구조 조건(optimal substructure)
    - 문제에 대한 최종 해결 방법은 부분 문제에 대한 최적 문제 해결 방법으로 구성된다.

# 프림 알고리즘

## 최소 신장 트리를 찾는 그리디 알고리즘

**알고리즘 11-2** 프림 알고리즘

Prim $(G, r)$:

▷ $G=(V, E)$: 그래프, $r$: 시작 정점

  $S \leftarrow \emptyset$; $T \leftarrow \emptyset$  ▷ $S$: 정점 집합, $T$: 간선 집합

  정점 $r$을 집합 $S$에 더한다.

  while $(S \neq V)$

    $S$에서 $V-S$를 연결하는 간선들 중 최소 길이의 간선 $(x, y)$를 찾는다.

    ▷ $(x \in S, y \in V-S)$

    $T$에 간선 $(x, y)$를 더한다.

    정점 $y$를 집합 $S$에 더한다.

# 그리디 알고리즘

**알고리즘 11-3** 그리디 알고리즘

Greedy($C$):

▷ $C$: 원소들의 총 집합

$S \leftarrow \emptyset$

while ($C \neq \emptyset$ and $S$는 아직 온전한 해가 아님)

❶ $x \leftarrow C$에서 원소 하나 선택

집합 $C$에서 $x$를 제거한다. ▷ $C \leftarrow C-\{x\}$

❷ if ($S$에 $x$를 더해도 됨) $S \leftarrow S \cup \{x\}$

if ($S$가 온전한 해임) return $S$

else return "해 없음!"

# 최적해가 보장되지 않는 예

## 이진 트리의 최적합 경로 찾기

**알고리즘 11-4** 이진 트리의 탐색(그리디)

```
Greedy_Sum(T, w[], r):
▷ T: 이진 트리, r: 루트 노드, w[]: 노드들에 할당된 수
    x ← r; sum ← 0
    while (x가 리프 노드가 아님)
        sum ← sum + w[x]
        x ← x의 자식 중 가중치가 큰 자식
    sum ← sum + w[x]
    return sum
```

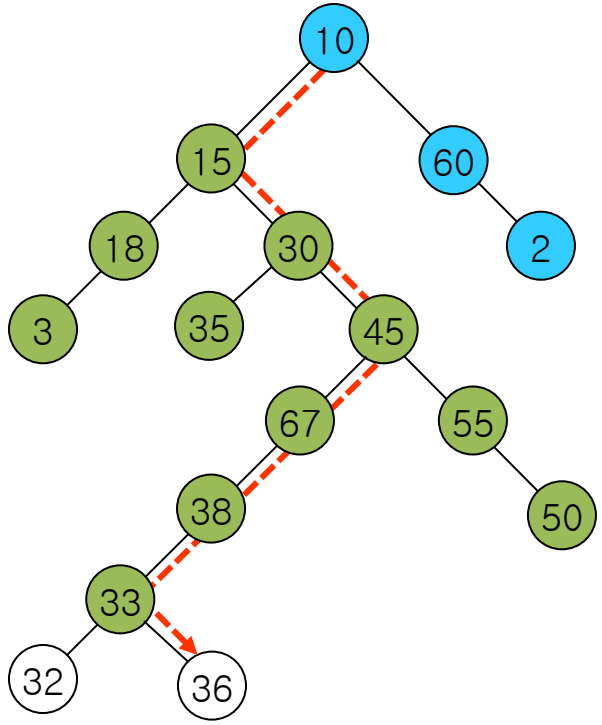



그림 11-1 그리디 알고리즘이 실패하는 예

# 최적해가 보장되지 않는 예

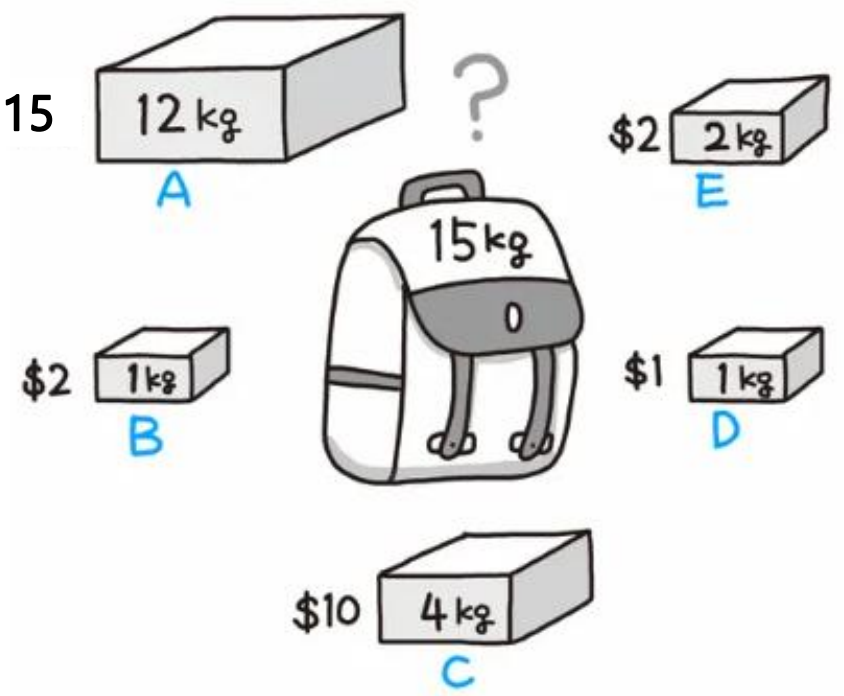
## 보따리 문제

**알고리즘 11-5** 보따리 문제(그리디)

```
Greedy_Knapsack(p[], W[], M):
▷ P[]: 가치 배열, W[]: 부피 배열
▷ X: 보따리에 담는 물건 집합, M: 보따리 부피

    ▷ P와 W를 P[i]/W[i]에 따라 내림차순(단위 부피당 가치가 큰 순서)으로 정렬한다.
    headRoom ← M; i ← 1
    while (W[i] ≤ headRoom and i≤n)
        X ← X ∪ {i}
        headRoom ← headRoom − W[i]
        i++
    return X
```

# 0/1 knapsack

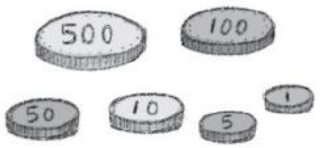


# 최적해가 보장되지 않는 예

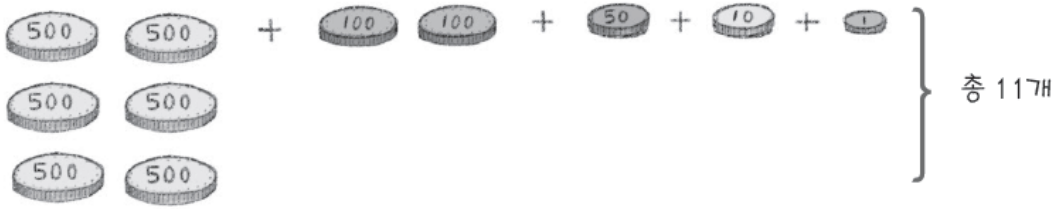
## 동전 바꾸기

- 액면이 바로 아래 액면의 배수가 되지 않으면 그리디 알고리즘으로 최적해가 보장되지 않는다.
- 예: 다음 페이지

동전의 액면




3,256원 만들기




동전 바꾸기

이렇게 동전의 액면이 모두 바로 아래 액면의 배수가 되면
그리디 알고리즘으로 최적해가 보장된다

타로는 자주 JOI잡화점에서 물건을 산다. JOI잡화점에는 잔돈으로 500엔, 100엔, 50엔, 10엔, 5엔, 1엔이 충분히 있고, 언제나 거스름돈 개수가 가장 적게 잔돈을 준다. 타로가 JOI잡화점에서 물건을 사고 카운터에서 1000엔 지폐를 한장 냈을 때, 받을 잔돈에 포함된 잔돈의 개수를 구하는 프로그램을 작성하시오.

## 입력

입력은 한줄로 이루어져있고, 타로가 지불할 돈(1 이상 1000미만의 정수) 1개가 쓰여져있다.

## 출력

제출할 출력 파일은 1행으로만 되어 있다. 잔돈에 포함된 매수를 출력하시오.

### 예제 입력 1 복사

```
380
```

### 예제 출력 1 복사

```
4
```

### 예제 입력 2 복사

```
1
```

### 예제 출력 2 복사

```
15
```

# 최적해가 보장되지 않는 예

## 동전 바꾸기

- 액면이 바로 아래 액면의 배수가 되지 않으면 그리디 알고리즘으로 최적해가 보장되지 않는다

# 최적해가 보장되는 예

## 최소 신장 트리 찾기를 위한 프림 알고리즘과 크루스칼 알고리즘

**알고리즘 11-6** 프림 알고리즘

```
Prim(G, r):
▷ G=(V, E): 주어진 그래프
▷ r: 시작 정점
    S ← ø
    for each u∈V
        u.cost ← ∞
    r.cost ← 0
    while (S≠V)
        u ← extractMin(V−S, d)
        S ← S∪{u}
        for each v∈u.adjlist
            if (v∈V−S and w(u, v) < v.cost)
                v.cost ← w(u, v)
                v.prev ← u
```

# 최적해가 보장되는 예

## 회의실 배정 문제

**알고리즘 11-7** 회의실 배정 문제(그리디)

```
Greedy_Schedule(S, n):
▷ S = {(s_i, t_i) | i = 1, 2, ···, n}, n: 신청 회의 수
▷ s_i: 시작 시간, t_i: 종료 시간
    t_i에 대한 오름차순으로 정렬하고, 이 순서대로 S = {(s_i, t_i) | i = 1, 2, ···, n}의
        번호를 다시 매긴다.        ▷ 즉, 종료 시간이 가장 이른 회의는 (s_1, t_1)이 된다.
    T ← {1}
    last ← 1
    for (i ← 2 to n)
        if (t_last ≤ s_i)
            T ← T ∪ {i}
            last ← i
    return T
```

# 그리디 알고리즘로 해결이 가능한 문제들

- Decision Tree
- Activity Selection Problem
- 거스름돈 문제(돈의 단위가 배수로 정해져 있을 때, 개수가 무제한일 때)
- Minimum Spanning Tree
- Dijkstra Algorithm
- Huffman Code

# 근사 알고리즘

- **근시안적 방법론**
  - 단기적인 목표를 중심으로 한 접근법
  - 현재 문제를 해결하기 위해 초점을 맞춤
- **근사 알고리즘(Approximation Algorithm)**
  - 최적의 해를 구할 수 없는 문제에서 근사한 해를 구하는 알고리즘
  - 최적해를 보장하진 않지만, 많은 경우에 최적해에 근접한 값을 구할 수 있음.

# 매트로이드

- **매트로이드 구조를 가지면 그리디 알고리즘으로 최적해가 보장된다**

**정의 11-1 매트로이드**

어떤 유한 집합 $S$의 부분집합들의 집합인 $I$(즉, $I \subseteq 2^S$)가 다음의 성질을 만족하면 매트로이드라 한다.

1. $A \in I$이고 $B \subseteq A$이면 $B \in I$이다. ▷ 상속성Heredity
2. $A, B \in I$이고 $|A| < |B|$이면, $A \cup \{x\} \in I$인 $x \in B - A$가 존재한다. ▷ 증강성Augmentation

# 그래픽 매트로이드

- **숲forest**
  - 하나 이상의 트리들로 이루어진 집합
  - 또는, 사이클을 이루지 않은 간선들의 집합
- **숲 집합 $F \subseteq 2^E$는 매트로이드다**

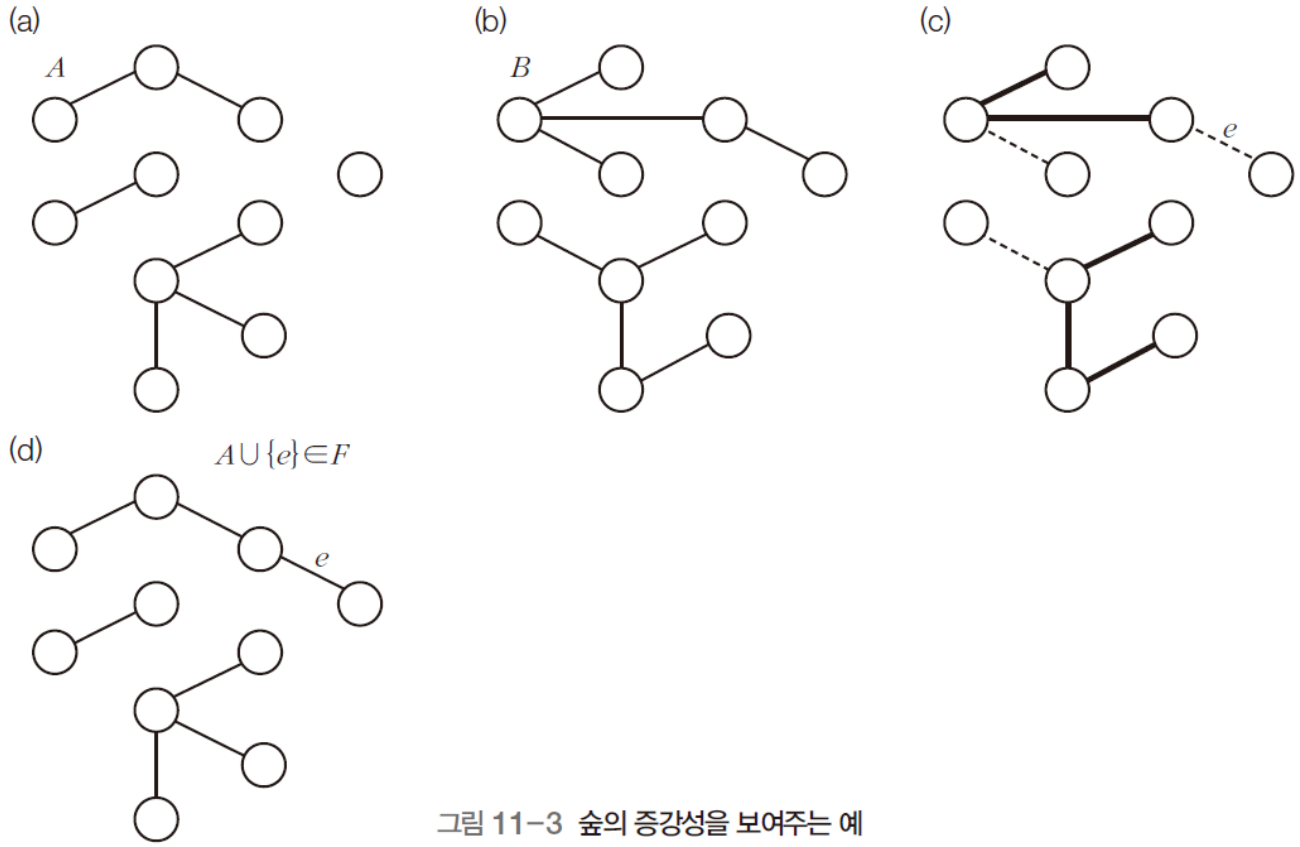



그림 11-3 숲의 증강성을 보여주는 예

# 매트로이드: 숲의 예

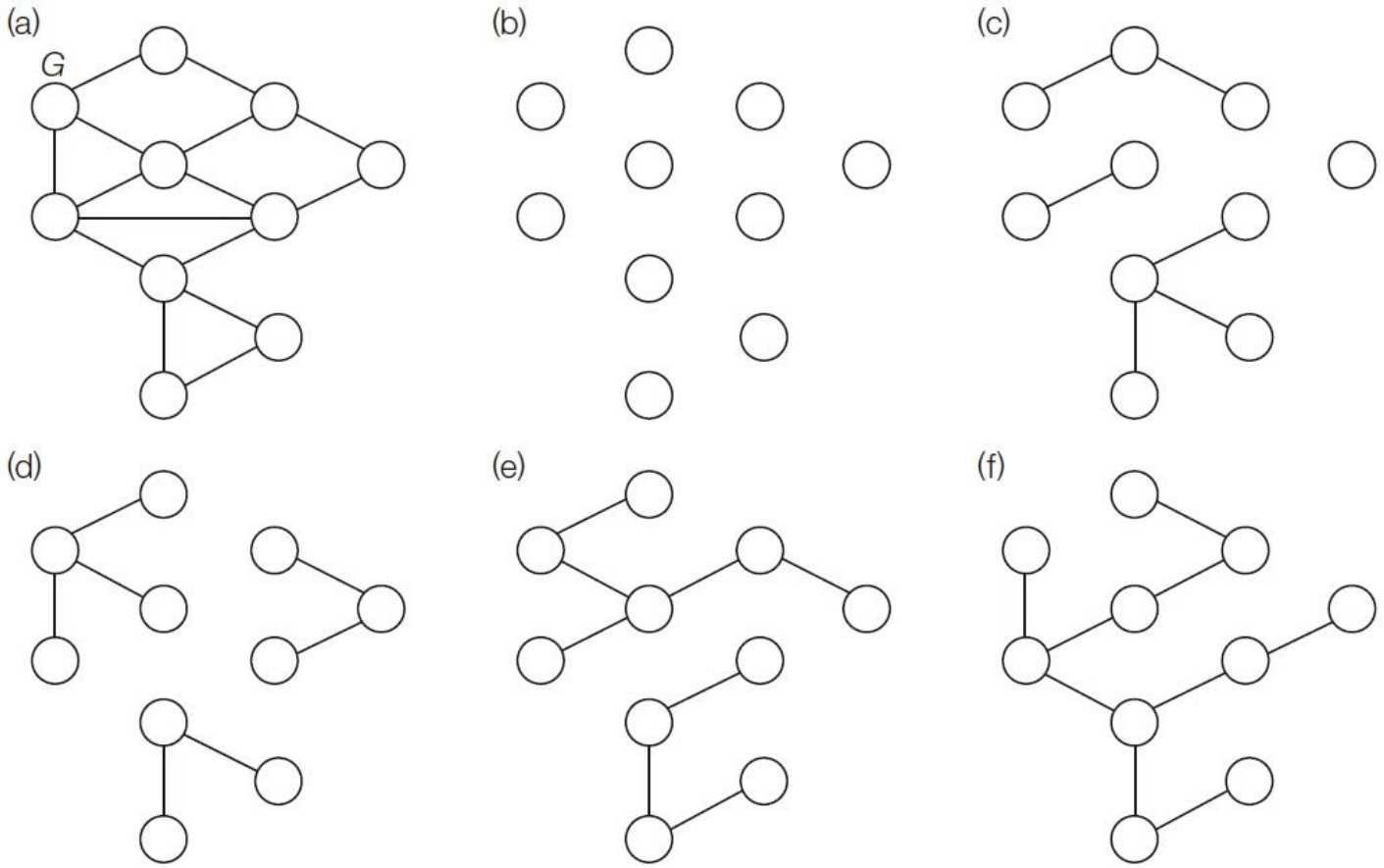



그림 11-2 (a)의 그래프로 만들 수 있는 숲의 예

# 매트로이드의 확장

**정의 11-2 확장Extension**

매트로이드 $I \subseteq 2^S$와 $A \in I$에서 $A$에 속하지 않는 어떤 원소 $x \in S$에 대하여 $A \cup \{x\} \in I$이면 $x$가 $A$를 확장한다고 한다. $A$가 더 이상 확장되지 않으면 $A$를 포화 집합Maximal Set이라 한다.

# 가중치 매트로이드

- 매트로이드의 원집합 S의 원소들이 (양의) 가중치를 갖고 있을 때 원소들의 합을 최대화하는 부분 집합 $A \in I$를 찾고자 한다
- 다음 그리디 알고리즘으로 최적해가 보장된다

**알고리즘 11-8** 최대 가중치 합 구하기(그리디)

Greedy$(I, w[])$:

▷ $I \subseteq 2^S$: 매트로이드, $w[]$: 가중치 배열

$A = \emptyset$

$S$의 원소들을 $w$의 가중치 크기로 내림차순 정렬한다.

for each $x \in S$(가중치 내림차순으로)

  if $(A \cup \{x\} \in I)$ $A \leftarrow A \cup \{x\}$

return $A$

# 문제 공간 탐색 관점에서 본 매트로이드

**알고리즘 11-9** 매트로이드 구조 문제(개선형 그리디)

Greedy2 $(I, A, w[\,])$

▷ $I \subseteq 2^S$: 매트로이드, $A \in I$, $w[\,]$: 가중치 배열

❶ while ($w(a) < w(x)$이고 $A \cup \{x\} - \{a\} \in I$인 $a \in A$, $x \in S - A$가 존재)

$A \leftarrow A \cup \{x\} - \{a\}$

return $A$

# 문제 공간 탐색 관점에서 본 매트로이드

**정의 11-3 인접성**

임의의 해 $A$, $B \in I$에 대해 $B = A \cup \{b\} - \{a\}$, $a \in A$, $b \in B - A$이면 $B$는 $A$에 인접하다고 한다.

**정의 11-4 지역 최적해 또는 끌개**

임의의 해 $A \in I$에 대해 $A$와 품질이 동일한 해들의 인접 관계의 체인을 따라 이를 수 있는 모든 해에서 품질이 더 좋은 인접해가 없으면 $A$는 지역 최적해 또는 끌개라고 한다.

# 문제 공간 탐색 관점에서 본 매트로이드

**정리 11-5**

매트로이드 $I$에서 $A \in I$보다 품질이 좋은 해가 존재한다면, $A$와 인접한 해 중에서 $A$보다 품질이 좋은 해가 반드시 하나 이상 존재한다.

**따름 정리 11-6**

매트로이드 $I$에서 $A \in I$와 인접한 해 중에 자신보다 품질이 더 좋은 해가 없으면 $A$는 (전역) 최적해다.

**정리 11-7**

매트로이드 $I$에서 서로 다른 두 (전역) 최적해 $A$, $B \in I$가 있다면 이들은 원소의 집합은 다르지만 가중치의 집합은 동일하다.

**정리 11-8**

서로 다른 어떤 두 최적해 $A$, $B \in I$든지 동일한 품질의 해들로 연결된 인접 관계의 체인을 따라 도달 가능하다.

**따름 정리 11-9**

매트로이드 $I$에서 최적해를 이루는 봉우리는 단 하나뿐이다.

# [참고] 재미있는 성질

- 가중치 매트로이드에서 서로 다른 최적해가 2개 이상 존재하면 그들의 원소 가중치 집합은 반드시 동일하다
- 가중치 매트로이드의 문제 공간에서는 단 하나의 봉우리만 존재하고 거기에 1개 또는 그 이상의 최적해가 존재한다 (이동 연산자와 관련이 있지만 직관적 이해를 위해 skip)