ALGORITHM
알고리즘

Lecture 01

알고리즘 소개 (1)

컴퓨터과학과 | 이관용 교수

한국방송통신대학교

학습목차

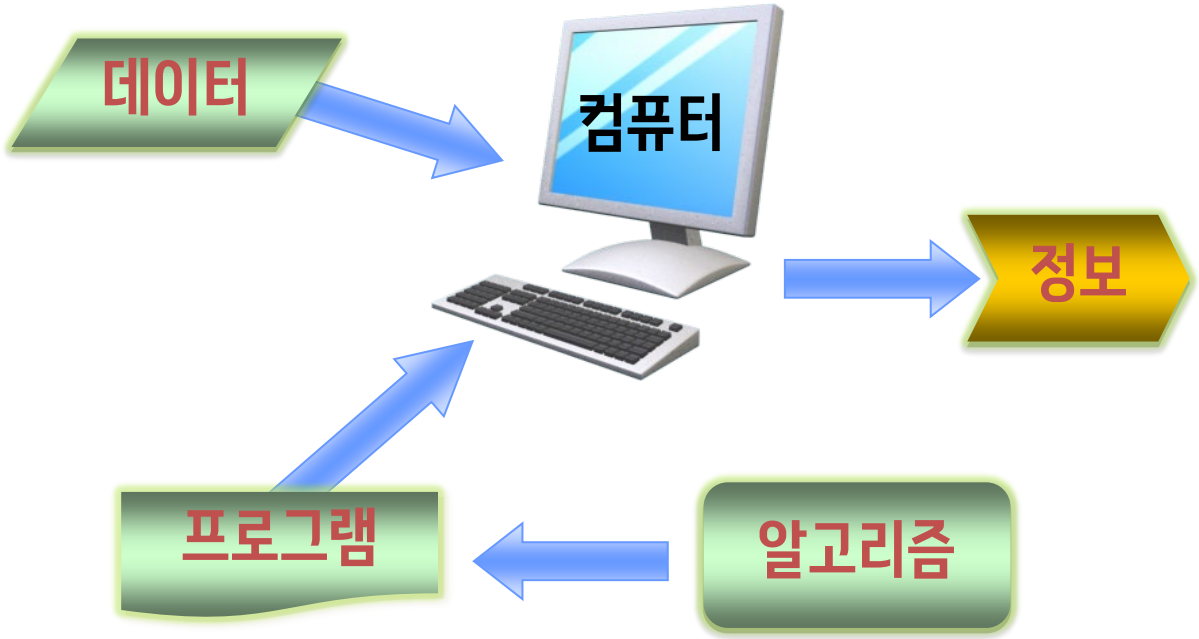
01 | 『알고리즘』 과목 소개

02 | 기본 개념

03 | 알고리즘 설계

01 『알고리즘』 과목 소개

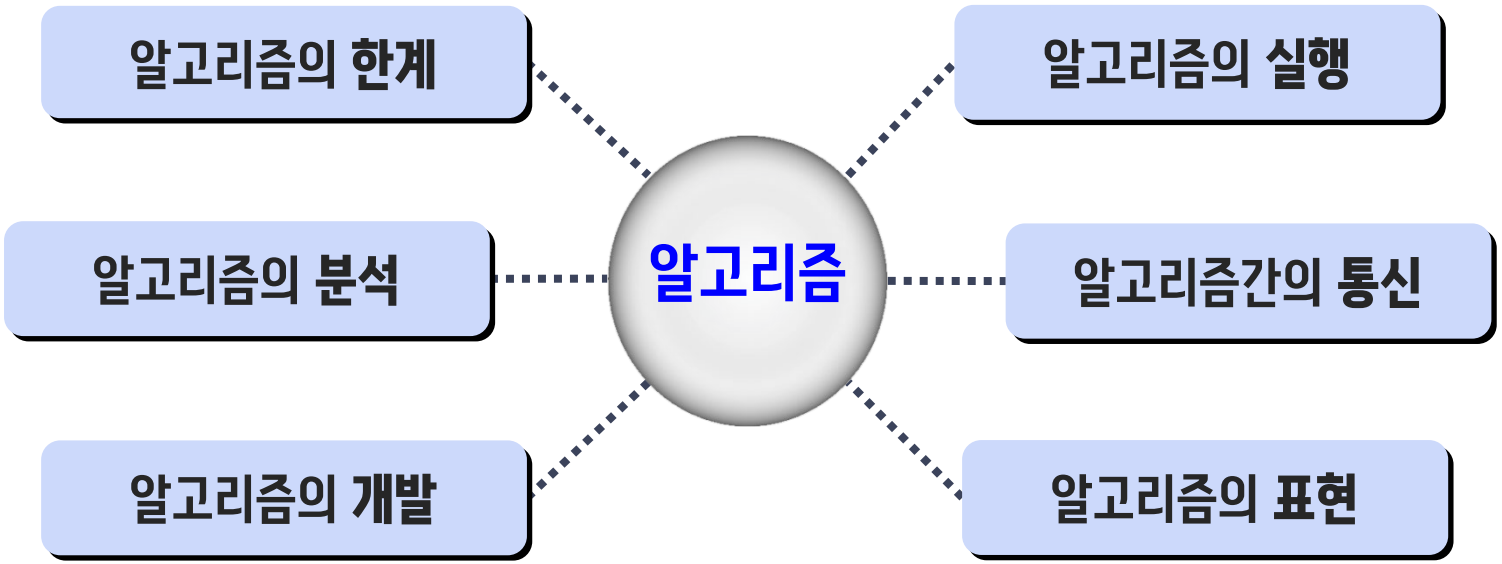
컴퓨터과학에서 알고리즘이란?



컴퓨터과학 = 컴퓨터 + 데이터 + 프로그램 + 알고리즘

컴퓨터과학에서 알고리즘이란?

- 컴퓨터의 한계 ≈ 알고리즘의 존재 여부
- "컴퓨터과학 = 알고리즘 과학"



학습목표

잘 알려진 특정 문제를 위한 알고리즘의 설계 및 분석 방법의 습득

컴퓨터를 이용한 문제 해결 방법에 대해 체계적으로 생각하는 훈련

주어진 문제에 대한 지적 추상화 능력 및 통찰력 향상

『알고리즘』 교재 및 강의 구성

1장. 알고리즘 소개	개념, 설계, 분석, 점근성능, 순환 알고리즘	1, 2 강	이관용 교수
2장. 정렬	선택, 버블, 삽입, 셸, 퀵, 합병, 힙, 계수, 기수, 버킷 정렬	3,4,5 강	
3장. 탐색	순차/이진탐색, 이진 탐색 트리, 2-3-4 트리, 레드-블랙 트리, B-트리, 해싱	6,7 강	
4장. 그래프	그래프 순회, 크루스칼/프림 알고리즘, 데이크스트라/벨만-포드/플로이드 알고리즘, 포드-풀커슨 알고리즘	8,9,10 강	
5장. 동적 프로그래밍	행렬의 연쇄적 곱셈, 최장 공통 부분 수열	11 강	
6장. 스트링 알고리즘	라빈-카프/KMP/보이어-무어 알고리즘, RLE/허프만 코딩/LZ77, 영상 압축	12,13,14 강	김진욱 교수
7장. NP-완전 문제	개념 및 용어, 근사 알고리즘(버텍스 커버 문제, 외판원 문제, 궤 채우기 문제)	15 강	

평가 방법

중간 평가 30점 + **기말 평가** 50점 + **형성 평가** 20점

출석수업

또는

출석수업대체시험

(교재 1~3장 + 해당 강의)

※ 변경 가능, 반드시 공고 참조

객관식 25문항

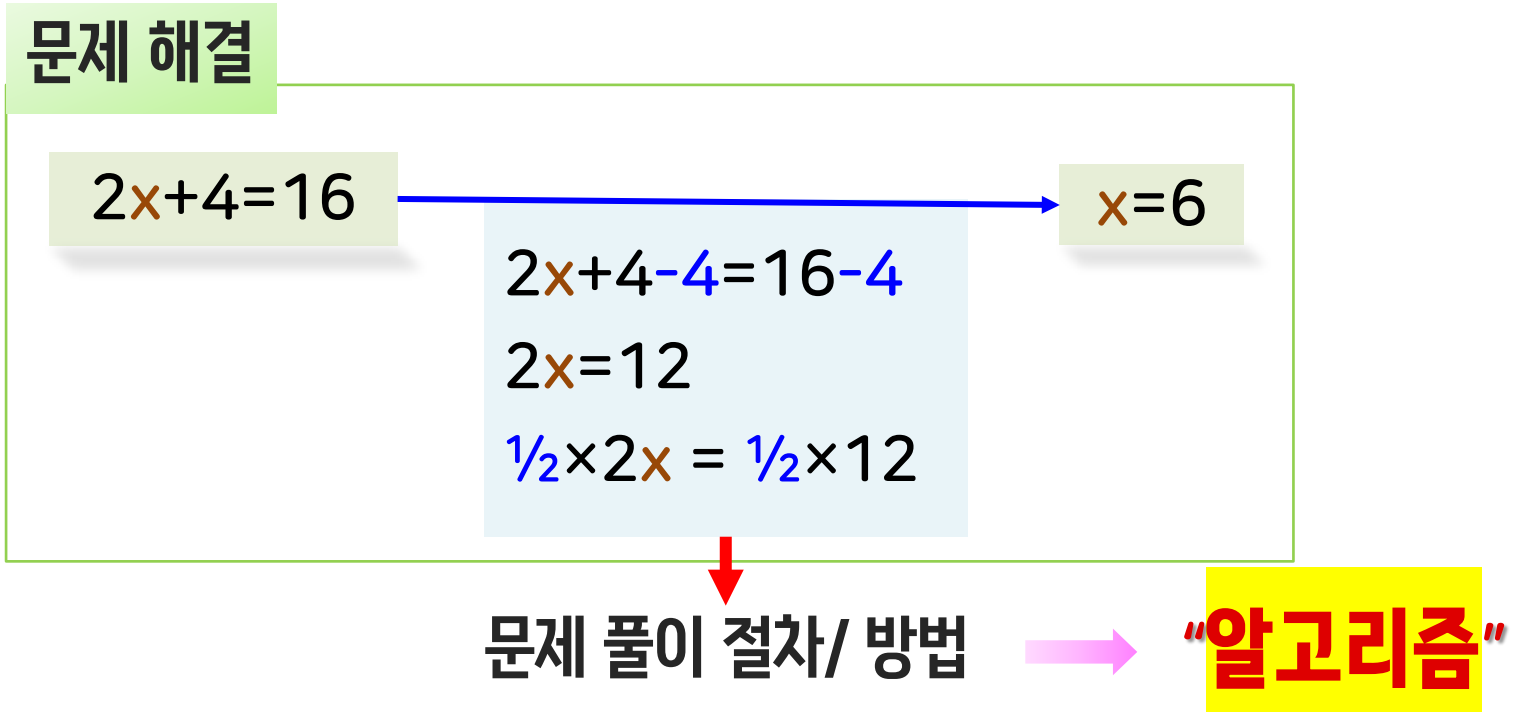
(교재/강의 전체)

02 기본 개념

문제를 해결하려면...

"컴퓨터과학"?

- 컴퓨터를 활용해서 주어진 문제를 해결하기 위한 학문



알고리즘 vs 레시피

알고리즘이란?

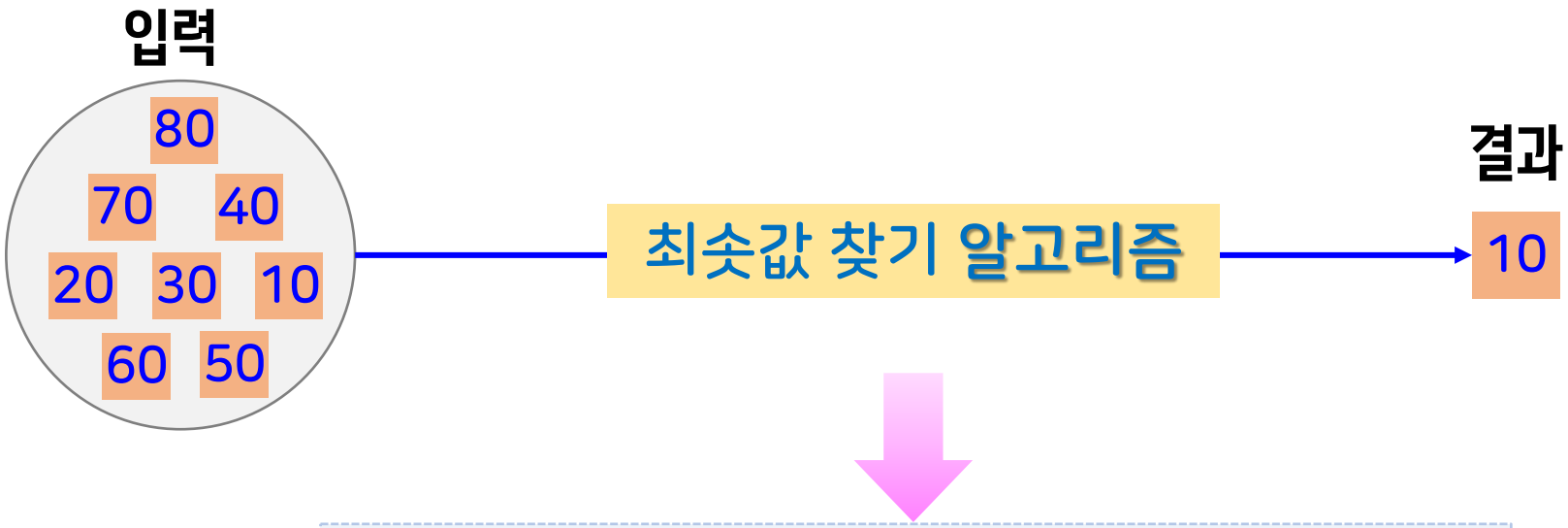
- 문제 해결을 위한 "레시피 recipe"

✓ 레시피의 단계적인 조리 절차를 따르면 음식을 만들 수 있듯이
알고리즘의 단계적인 처리 절차를 따르면 주어진 답을 구할 수 있음

✓ 레시피 → "맛있고 몸에 좋은 음식"

✓ 알고리즘 → "효율적인 알고리즘"

최솟값을 찾는 문제



[1] 첫 번째 데이터를 최솟값으로 저장한다.

[2] 다음 숫자를 읽고, 이것과 저장된 최솟값과 비교한다.

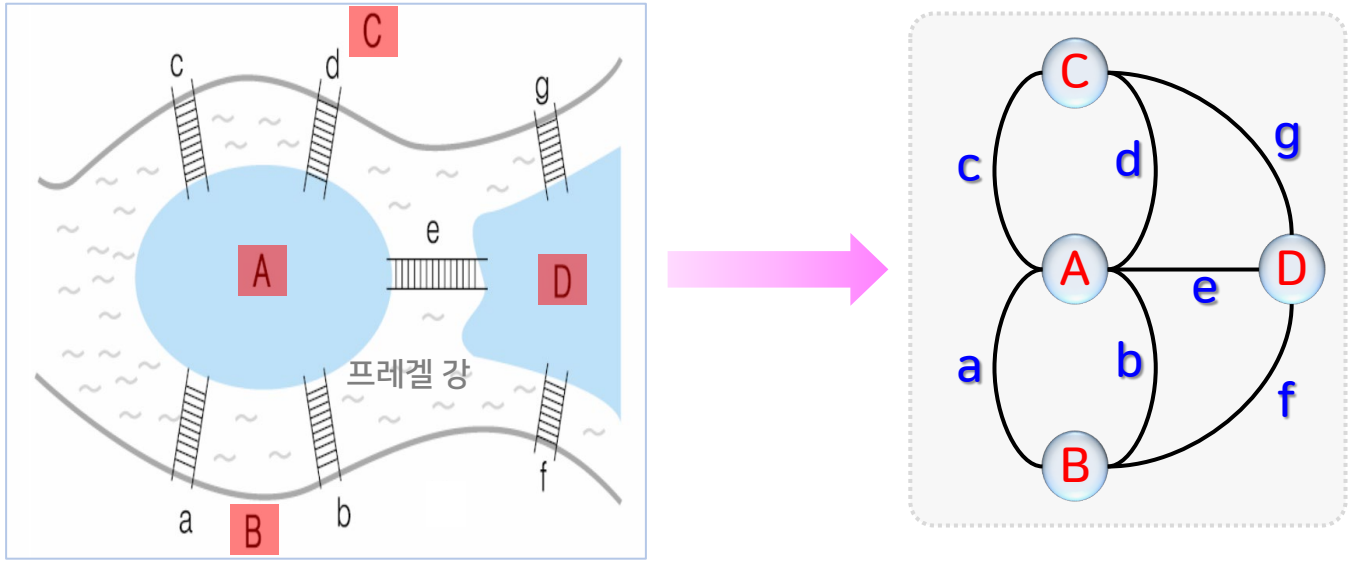
[3] 비교 후 더 작은 숫자를 최솟값으로 다시 저장한다.

[4] 다음에 처리할 데이터가 남아 있으면 [2]로 간다.

[5] 저장된 최솟값을 결과로 출력한다.

오일러 경로를 찾는 문제

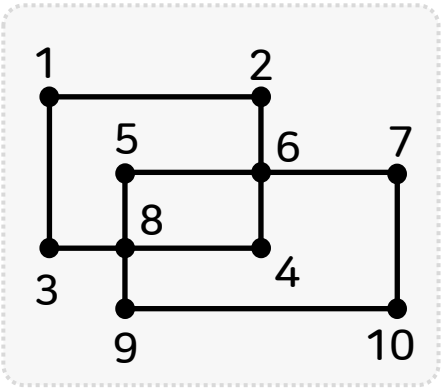
"쾨닉스버그 다리 문제"

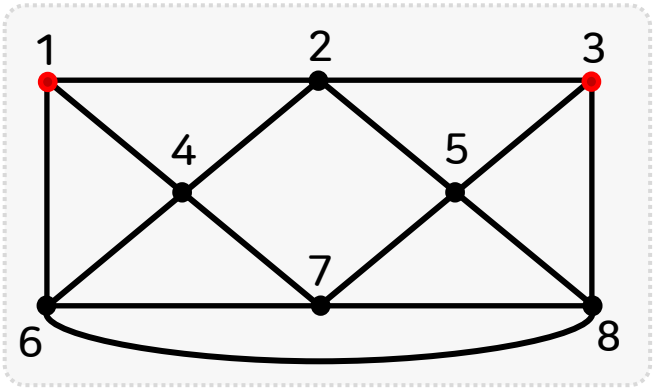


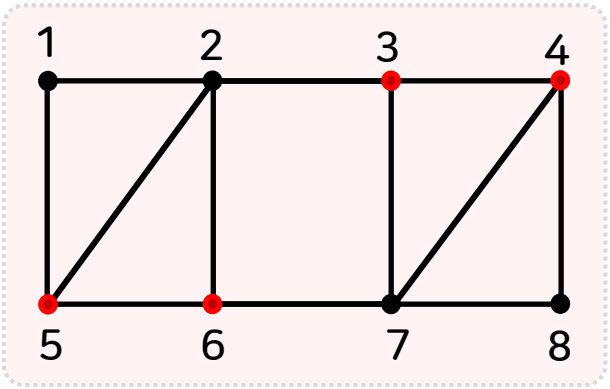
오일러 경로

- 그래프의 모든 간선을 오직 한 번씩만 지나가는 경로

오일러 경로를 찾는 문제





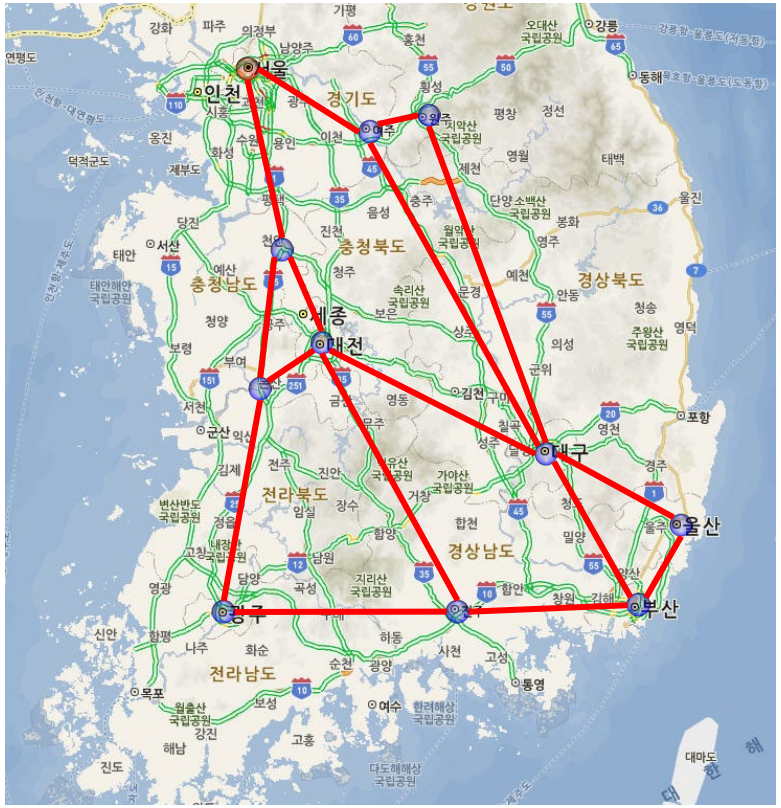


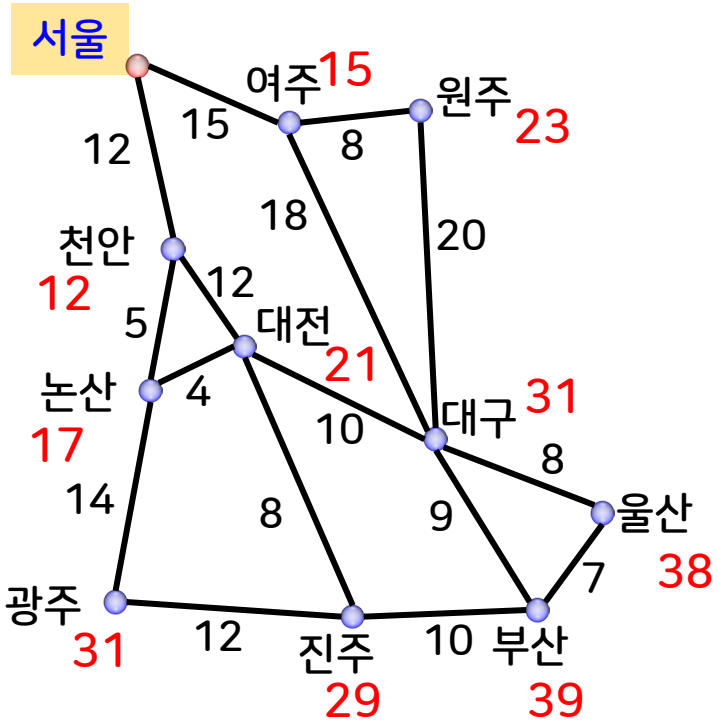
오일러 경로를 찾는 규칙

- 각 정점의 차수가 홀수인 정점이 0개 혹은 2개이어야 한다.
- 홀수점이 2개일 경우에는 홀수점에서 시작해야 한다.

최단 경로를 찾는 문제

고속도로를 이용해서 서울에서 부산까지 가장 짧게(거리, 시간) 가는 방법





단일 출발점 최단 경로 → 데이크스트라 알고리즘

알고리즘의 정의

주어진 문제를 해결하거나 함수를 계산하기 위해 따라야 할 명령어들을 단계적으로 나열한 것

- (입출력) 0개 이상의 외부 입력과 1개 이상의 출력을 생성
- (명확성) 각 명령은 모호하지 않고 단순 명확해야 함
- (유한성) 한정된 수의 단계를 거친 후에는 반드시 종료함
- (유효성) 모든 명령은 컴퓨터에서 수행할 수 있어야 함

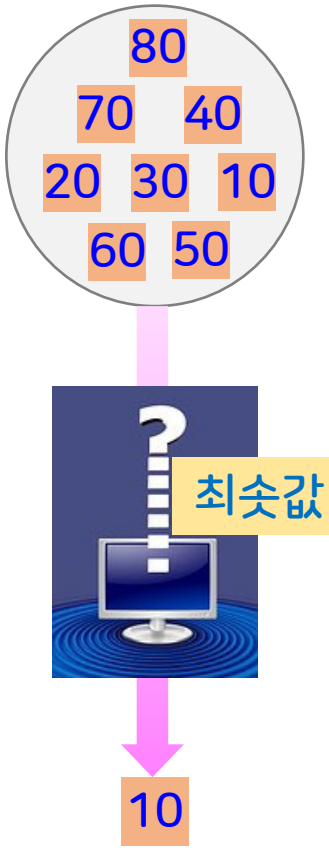
"주어진 문제에 대한 하나 이상의 결과를 생성하기 위해 모호하지 않고 단순 명확하며 컴퓨터가 수행할 수 있는 유한개의 일련의 명령어들을 순서에 따라 구성한 것"

- 실용적 관점 → (효율성) 알고리즘은 효율적이어야 함

알고리즘 생성 단계

설 계	표현/기술	정확성 검증	효율성 분석
상향식 설계 하향식 설계 ... 욕심쟁이 방법 분할정복 방법 동적 프로그래밍 방법 ...	일상 언어 순서도 의사코드 프로그래밍 언어 ...	수학적 증명	공간 복잡도 시간 복잡도

알고리즘의 표현/기술 방법



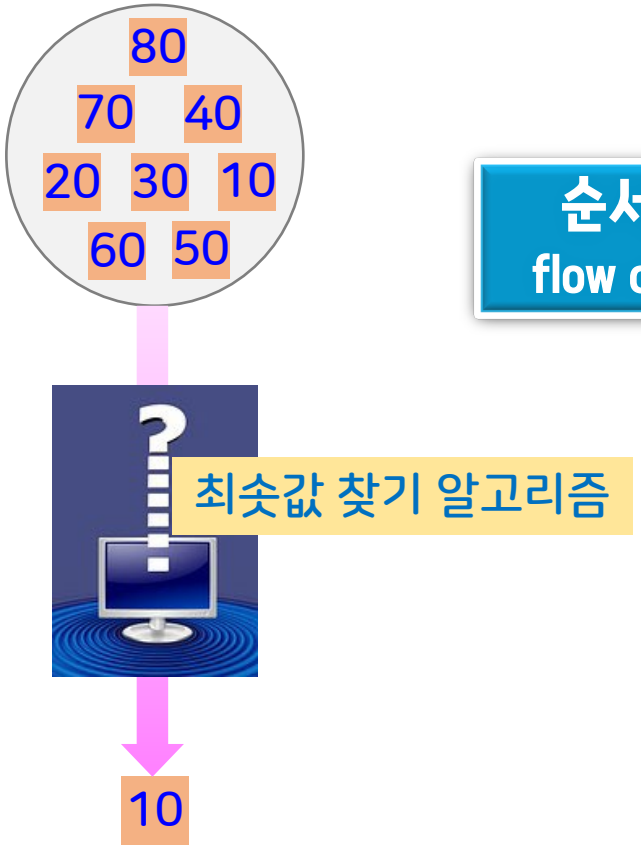
일상 언어

[1] 첫 번째 데이터를 최솟값으로 저장한다.
[2] 다음 숫자를 읽고, 이것과 저장된 최솟값과 비교한다.
[3] 비교 후 더 작은 숫자를 최솟값으로 다시 저장한다.
[4] 다음에 처리할 데이터가 남아 있으면 [2]로 간다.
[5] 저장된 최솟값을 결과로 출력한다.

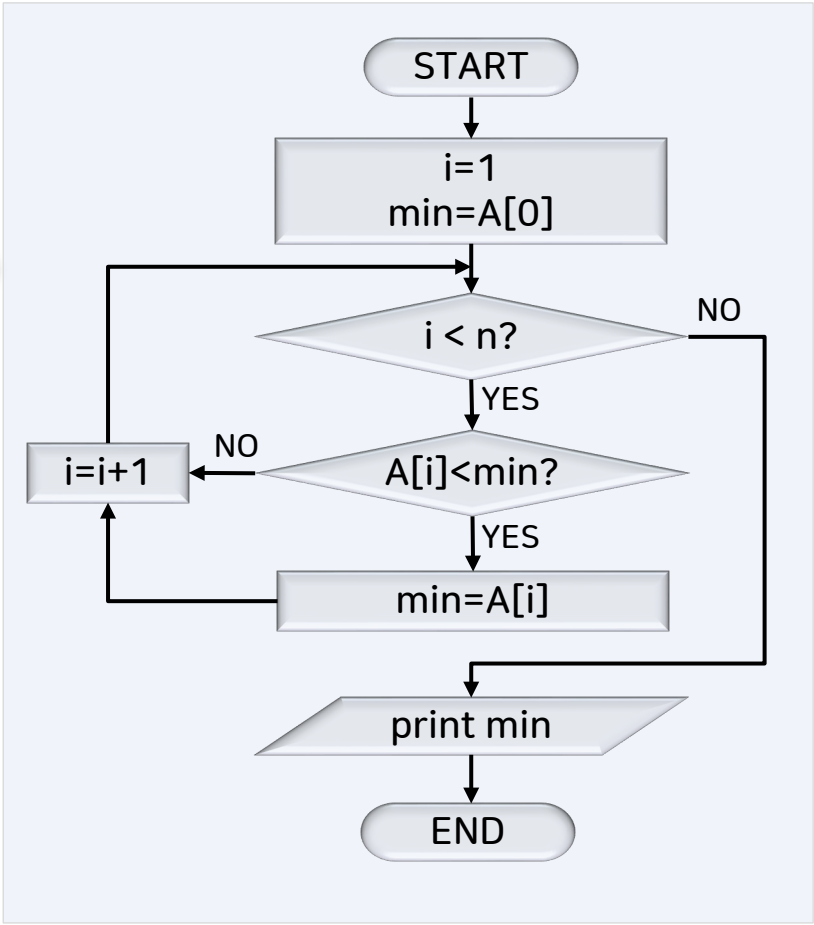
의사코드
pseudo code

```
i = 1;
min = A[0];      // 데이터 A[0..n-1]
while (i < n ) {
   A[i]가 min보다 작으면 min = A[i];
   i++;
}
최소값 min 출력;
```

알고리즘의 표현/기술 방법

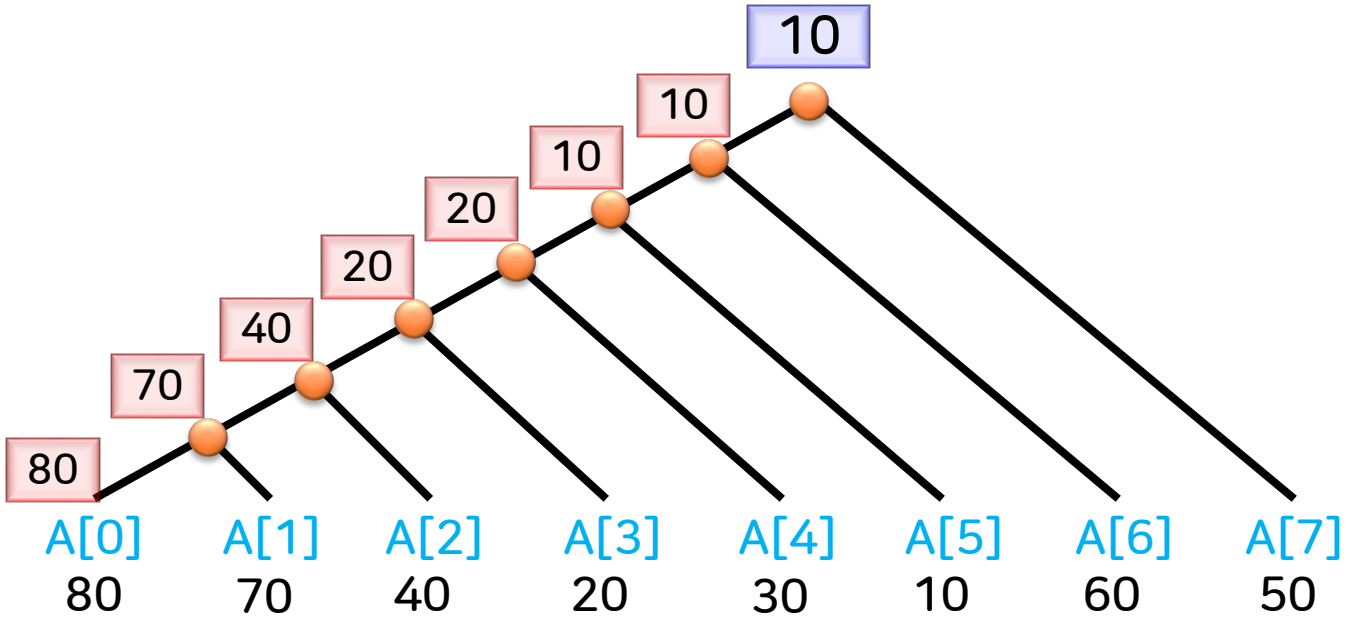


순서도
flow chart



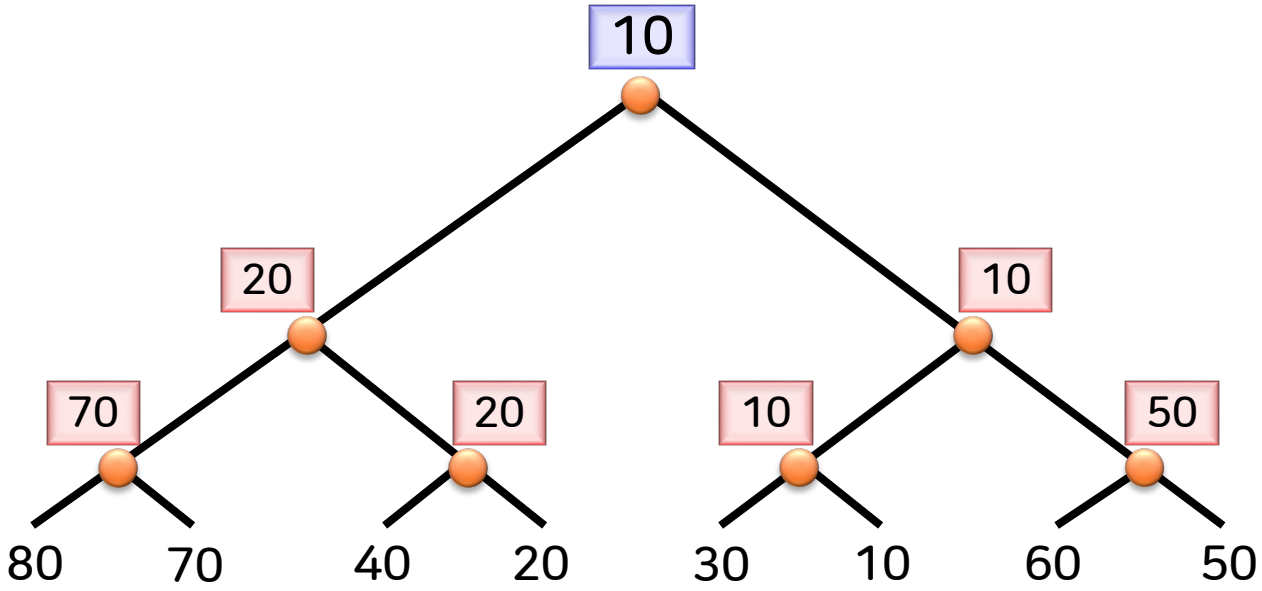
03 알고리즘 설계

최솟값 찾기_알고리즘 1



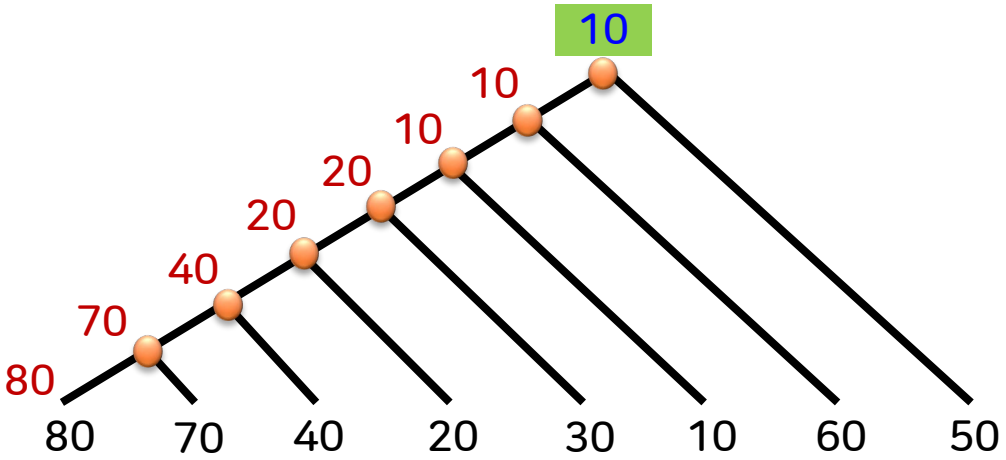
값들을 하나씩 모두 비교해 가면서 최솟값을 찾는 방법

최솟값 찾기_알고리즘 2

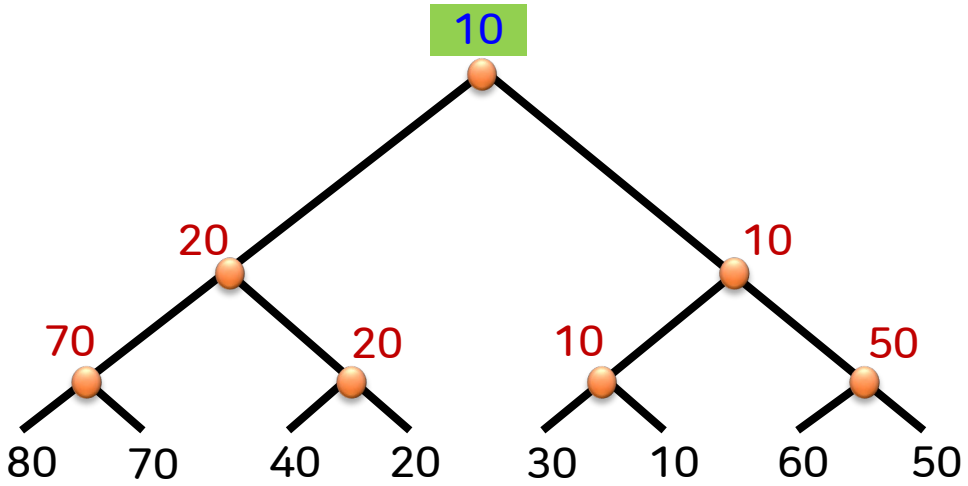


토너먼트 방식

최솟값 찾기 알고리즘



알고리즘1



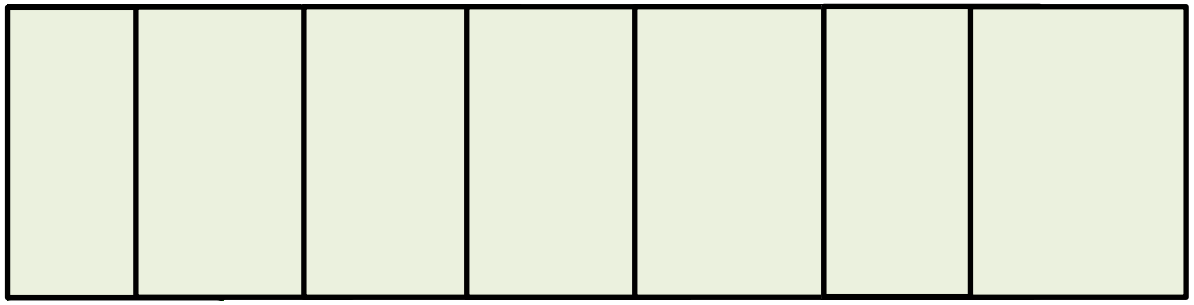
알고리즘2

최솟값 찾기에서 알고리즘1과 알고리즘2 중에서 어떤 것이 더 효율적인가?

힌트: 두 값을 비교하는 횟수?

탐색 알고리즘

뒤섞인 카드에서 10을 찾는 경우



순차 탐색
sequential search

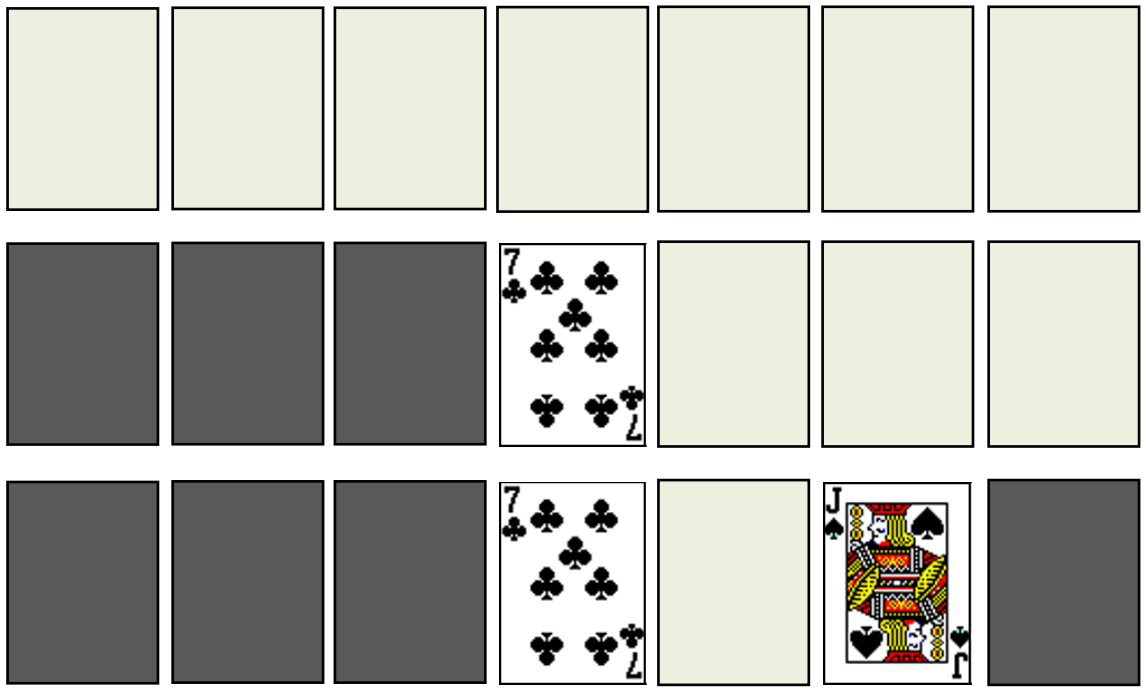
탐색 알고리즘

순차 탐색 알고리즘

```
SequentialSearch (A[], n, key)
// 배열 A[0..n-1]에서 key를 찾는 알고리즘
{
  i=0;
  while ( i < n && A[i] != key )
     i = i + 1;
  return (i);
}
```

탐색 알고리즘

순서대로 나열된 카드 중에서 10을 찾는 경우



이진 탐색
binary search

탐색 알고리즘

이진 탐색 알고리즘

```
BinarySearch (A[ ], key, Left, Right)
{
    if (Left > Right) return (-1);
    mid = | (Left + Right)/2 | ;
    if (A[Mid] == key) return (Mid);
    else if (key < A[Mid]) BinarySearch(A, key, Left, Mid-1)
            else BinarySearch(A, key, Mid+1, Right);
}
```

알고리즘 설계

주어진 문제와 조건 등이 매우 다양

→ 일반적/범용적인 설계 기법은 미존재

대표적인 알고리즘 설계 기법

- 욕심쟁이 greedy 방법
- 분할정복 divide-and-conquer 방법
- 동적 프로그래밍 dynamic programming 방법 (※교재 5장)

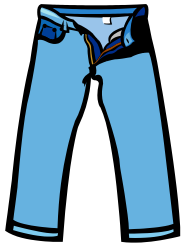
욕심쟁이 방법

- "탐욕적 방법", "탐욕 알고리즘", "그리디 greedy 알고리즘"
- 해를 구하는 일련의 선택 과정에서 전후 단계의 선택과는 상관없이 각 단계마다 '가장 최선'이라고 여겨지는 국부적인 최적해를 선택해 나가면 결과적으로 전체적인 최적해를 구할 수 있을 것이라는 희망적인 전략을 취하는 방법
  - "희망적" → 각 단계마다 선택한 국부적인 최적해가 항상 전체적인 최적해를 만들지 못할 수도 있음
  - ✓ 간단하면서 효율적인 알고리즘을 만들 수 있는 강력한 기법
  - 최솟값/최댓값을 구하는 최적화 문제에 주로 사용

욕심쟁이 방법

한계

선택 과정 1



선택 과정 2

선택 과정 3

멋쟁이씨가 세상에서 가장 멋진 바지, 가장 멋진 넥타이, 가장 멋진 쟈켓을 입었을 때 멋쟁이씨는 세상에서 가장 멋쟁이라고 할 수 있는가?

국부적인 최적해들이 전체 최적해를 구성하지 못하는 경우도 있다.

욕심쟁이 방법

대표적인 응용 문제

- 1장 → 거스름돈 문제, 배낭 문제
- 4장(그래프)
  - ✓ 최소 신장 트리 → 크루스칼 알고리즘, 프림 알고리즘
  - ✓ 단일 출발점 최단 경로 → 데이크스트라 알고리즘

욕심쟁이 방법_거스름돈 문제

"동전 문제", "동전 거스름돈 문제"

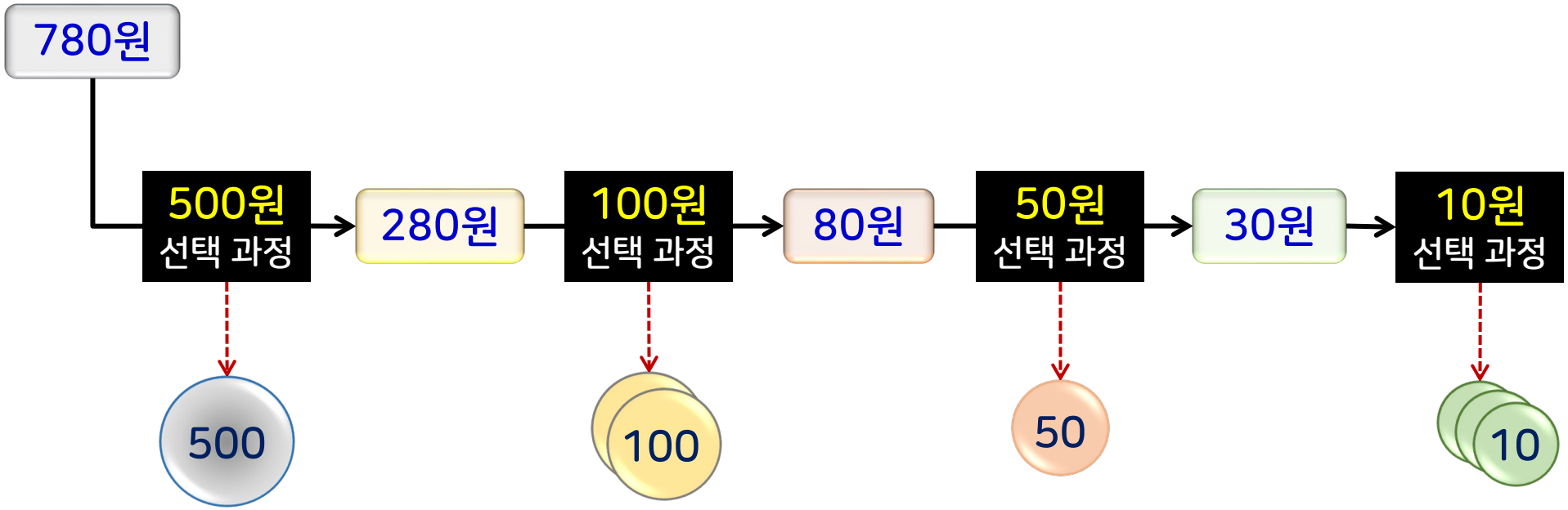
가게에서 고객에게 돌려줄 거스름돈이 T만큼 있을 때 고객이 받을 동전의 개수를 최소로 하면서 거스름돈을 돌려주는 방법을 찾는 문제

기본 해결 방법

- 거스름돈의 액수를 초과하지 않으면서 동전의 액면가가 단순히 큰 것부터 '욕심을 부려서' 최대한 뽑아서 거스름돈을 만듦

✓ 가정 → 동전의 종류: 500원, 100원, 50원, 10원

욕심쟁이 방법_거스름돈 문제

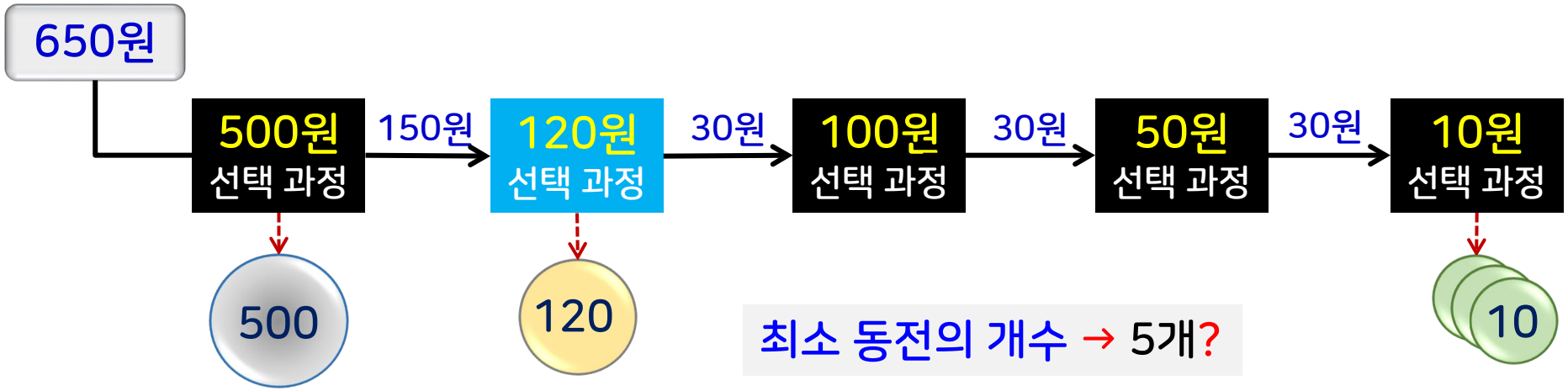


최소 동전의 개수 → 7개

욕심쟁이 방법_거스름돈 문제

동전의 종류

- 500원, 120원, 100원, 50원, 10원



→ 최적해 → 3개 500원×1개 + 100원 ×1개 + 50원 ×1개

→ 동전의 액면가가 일반적인 경우에는 욕심쟁이 방법으로 해결 불가

욕심쟁이 방법_배낭 문제

최대 용량 M인 하나의 배낭, n개의 물체가 있다고 가정

- 각 물체 i에는 물체의 무게 w_i와
  해당 물체를 배낭에 넣었을 때 얻을 수 있는 이익 p_i가 부여됨

배낭 문제

- 배낭의 용량을 초과하지 않는 범위 내에서
  배낭에 들어 있는 물체들의 이익의 합이 최대가 되도록
  물체를 넣는 방법(또는 최대 이익)을 찾는 문제

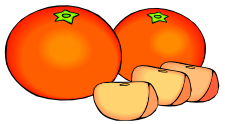
- 가정 → 물체를 쪼개서 넣을 수 있음

욕심쟁이 방법_배낭 문제

M = 10

n = 4

$p_1 = 15$
$w_1 = 3$



$p_2 = 20$
$w_2 = 5$

$p_3 = 9$
$w_3 = 3$

$p_4 = 14$
$w_4 = 4$

기본 해결 방안

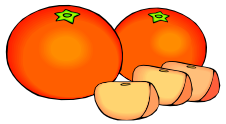
- 물체의 무게는 적으면서도 이익이 가장 큰 물체부터 골라서 '욕심을 내어' 최대한 넣는 과정을 반복
- 단위 무게당 이익이 가장 큰 물체부터 최대한 넣는 과정을 반복
  - ✓ 물체를 통째로 넣을 수 없으면 배낭의 남은 용량에 맞게 물체를 쪼개서 넣음

욕심쟁이 방법_배낭 문제

M = 10

n = 4

$p_1 = 15$
$w_1 = 3$



$p_2 = 20$
$w_2 = 5$

$p_3 = 9$
$w_3 = 3$

$p_4 = 14$
$w_4 = 4$

문제 표현

$M = 10, n = 4$

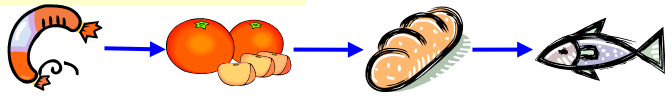
$(p_1, p_2, p_3, p_4) = (15, 20, 9, 14)$

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

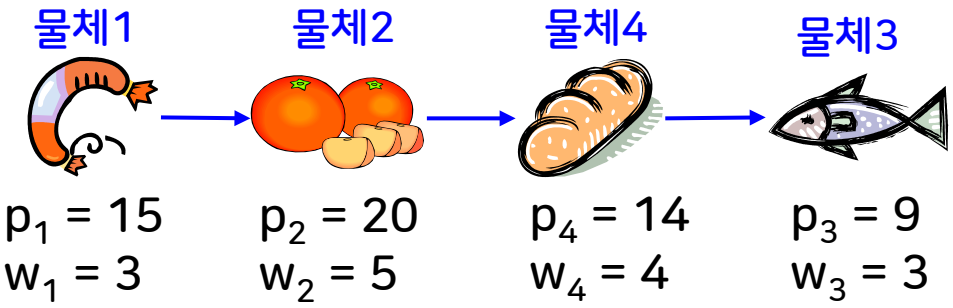
단위 무게당 이익

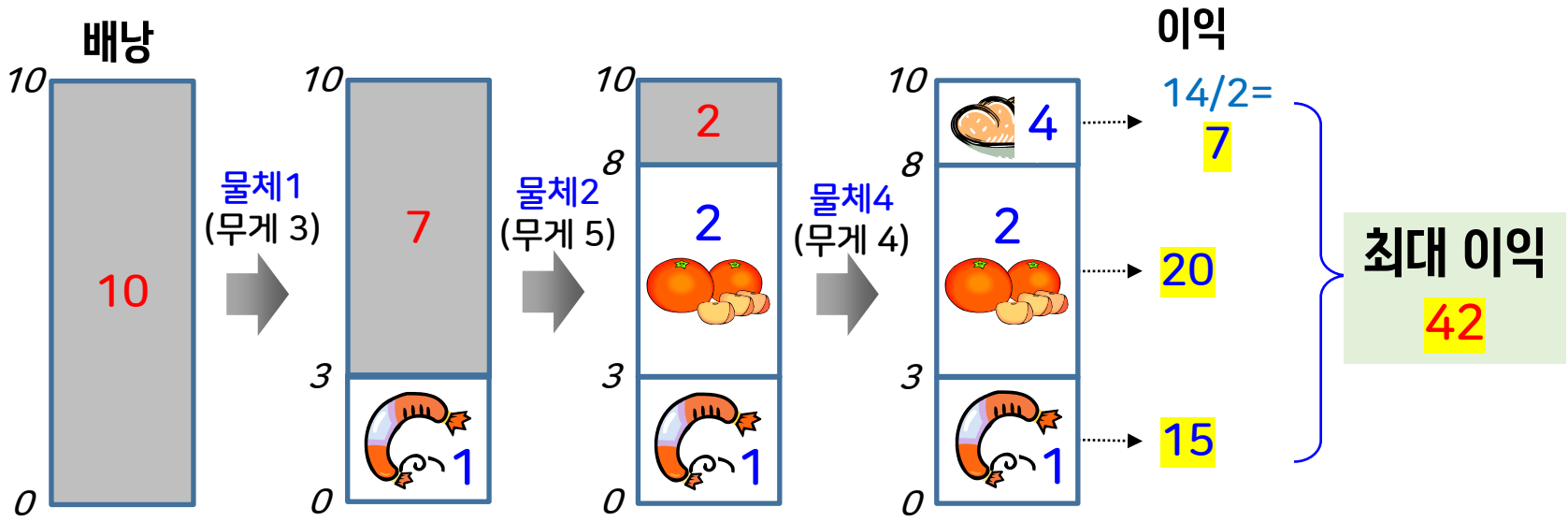
$$\left(\frac{p_1}{w_1}, \frac{p_2}{w_2}, \frac{p_3}{w_3}, \frac{p_4}{w_4}\right) = (5, 4, 3, 3.5)$$

1 2 4 3



욕심쟁이 방법_배낭 문제


물체1
물체2
물체4
물체3
$p_1 = 15$
$w_1 = 3$
$p_2 = 20$
$w_2 = 5$
$p_4 = 14$
$w_4 = 4$
$p_3 = 9$
$w_3 = 3$


배낭
이익
10
0
10
물체1
(무게 3)
7
3
1
물체2
(무게 5)
2
8
2
물체4
(무게 4)
4
14/2=
7
20
15
최대 이익
42

욕심쟁이 방법_배낭 문제

"0/1 배낭 문제"

- 물체를 쪼갤 수 없는 형태의 배낭 문제

M = 10, n = 4

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

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

단위 무게당 이익

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

욕심쟁이 방법 최대 이익 = $p_1 + p_2 = 35$

실제 최대 이익 = $p_1 + p_3 + p_4 = 38$

⇒ "0/1 배낭 문제는 욕심쟁이 방법으로 해결 불가"

분할정복 방법

순환적으로 문제를 푸는 하향식 top-down 접근 방법

- 주어진 문제의 입력을 더 이상 나눌 수 없을 때까지
  2개 이상의 작은 문제들로 순환적으로 분할하고,
  이렇게 분할된 작은 문제들을 각각 해결한 후,
  이들의 해를 결합하여 원래 문제의 해를 구하는 방식

특징

- 분할된 작은 문제는 원래 문제와 동일
  - ✓ 입력의 크기만 작아짐
- 분할된 문제는 서로 독립적
  - ✓ 순환적인 분할 및 결과의 결합이 가능

분할정복 방법

각 순환 호출마다 세 단계의 작업을 수행

분할 주어진 문제의 입력을 여러 개의 작은 문제로 분할

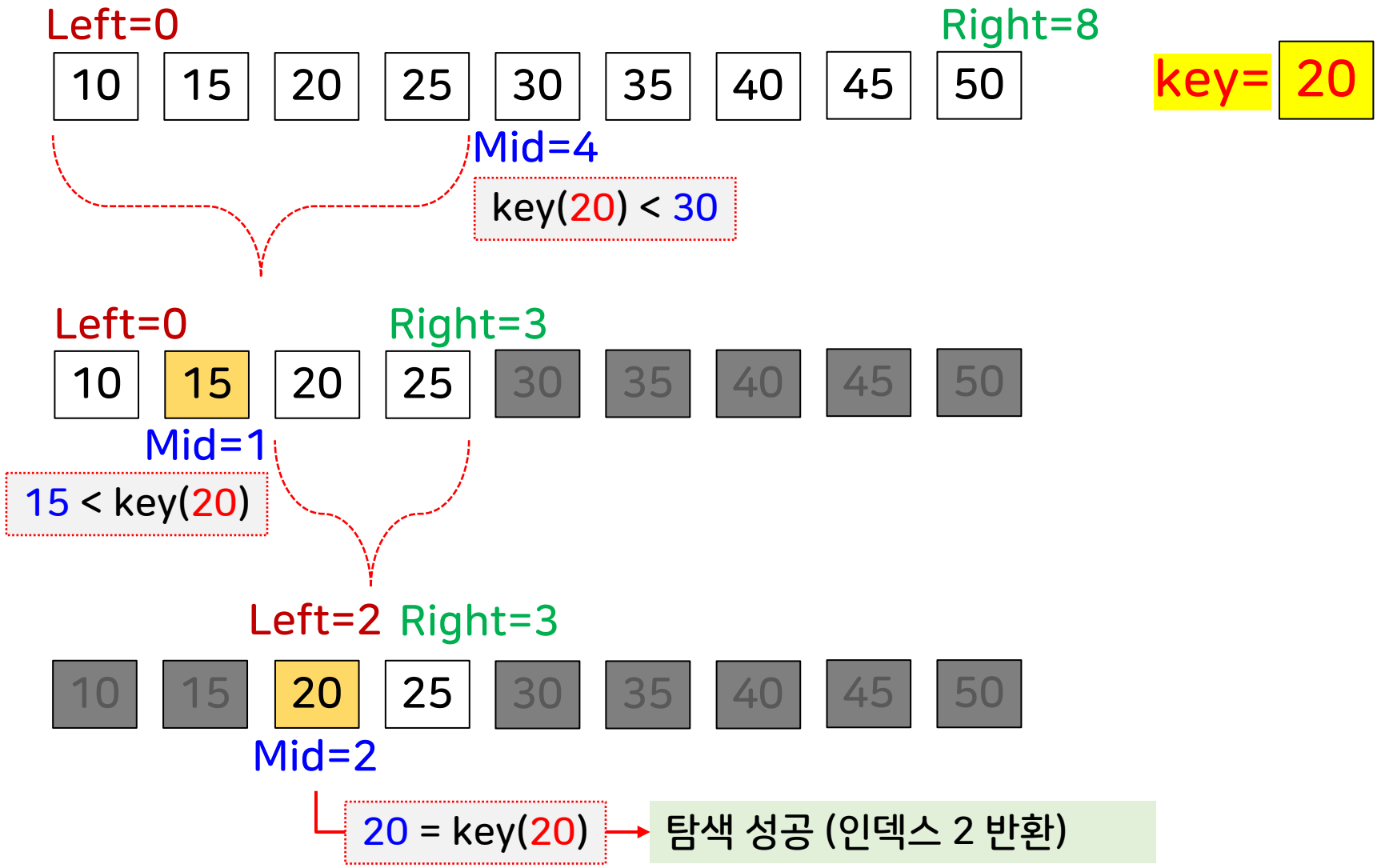
정복 작은 문제들을 순환적으로 분할.
만약 작은 문제가 더 이상 분할되지 않을 정도로 크기가
충분히 작으면 순환 호출 없이 작은 문제에 대한 해를 구함

결합 작은 문제에 대해 정복된 해를 결합하여 원래 문제의 해를 구함

대표적인 적용 문제

- 퀵 정렬, 합병 정렬, 이진 탐색

분할정복 방법_이진 탐색


Left=0
Right=8
10 15 20 25 30 35 40 45 50
key= 20
Mid=4
key(20) < 30
Left=0
Right=3
10 15 20 25 30 35 40 45 50
Mid=1
15 < key(20)
Left=2
Right=3
10 15 20 25 30 35 40 45 50
Mid=2
20 = key(20)
탐색 성공 (인덱스 2 반환)

분할정복 방법_이진 탐색

이진 탐색 알고리즘과 분할정복 방법의 관계

분할 배열의 가운데 원소를 기준으로 왼쪽과 오른쪽 부분배열로 절반씩 분할. 탐색 키와 가운데 원소가 같으면, 해당 원소의 배열 인덱스를 반환/종료

정복 탐색 키가 가운데 원소보다 작으면 왼쪽 부분배열을 대상으로 이진 탐색을 순환 호출, 크면 오른쪽 부분배열을 대상으로 이진 탐색을 순환 호출

결합 부분배열에 대한 탐색 결과가 직접 반환되므로 결합이 불필요

동적 프로그래밍 방법

입력의 크기가 가장 작은 부분 문제부터 해를 구하여 테이블에 저장해 놓고, 이를 이용해서 입력 크기가 보다 큰 문제의 해를 점진적으로 만들어 가는 상향식 bottom-up 접근 방법

- 각각의 작은 문제는 원래 문제와 동일, 입력의 크기만 작음
- 작은 문제들은 서로 독립일 필요가 없음

대표적인 적용 문제

- 모든 정점 쌍 간의 최단 경로 → 플로이드 알고리즘
- 5장 → 행렬의 연쇄적 곱셈 문제, 최장 공통 부분 수열 문제

정리하기

1. 기본 개념

- **알고리즘** → 주어진 문제를 해결하거나 함수를 계산하기 위해 따라야 할 명령어들을 순서적으로 나열한 것
- **조건** → 입출력, 명확성, 유한성, 유효성 + (실용적 조건) 효율성

2. 알고리즘 설계

- 대표적인 설계 기법 → 욕심쟁이 방법, 분할정복 방법, 동적 프로그래밍 방법
- **욕심쟁이 방법** → 거스름돈 문제, 배낭 문제, 최소 신장 트리 문제(크루스칼 알고리즘, 프림 알고리즘), 단일 출발점 최단 경로 문제(데이크스트라 알고리즘)
- **분할정복 방법** → 이진 탐색, 퀵 정렬, 합병 정렬
- **동적 프로그래밍 방법** → 모든 쌍 간의 최단 경로 문제(플로이드 알고리즘), 연쇄적 행렬 곱셈 문제, 최장 공통 부분 수열 문제

ALGORITHM
알고리즘

다음시간에는

Lecture 02

알고리즘 소개 (2)

컴퓨터과학과 | 이관용 교수