

## 알고리즘설계

- 정렬
- 분할정복 알고리즘
- 탐욕적 알고리즘

		강의 주제	주차
1부 컴퓨팅 사고력	컴퓨팅 사고력의 소개	IT 사회, 소프트웨어 세상, 컴퓨팅 사고력의 소개, 컴퓨팅 사고력의 개념	1
		컴퓨팅 사고력의 개념, 주위에서 볼 수 있는 컴퓨팅 사고력, 문제해결 방법	2
	문제해결 방법, 컴퓨터	문제해결 방법, 문제해결 과정 예, 문제해결을 위한 소프트웨어 설계 사상, 컴퓨터의 특징, 소프트웨어, 유한상태기계	3
2부 소프트웨어	알고리즘	알고리즘 소개, 알고리즘의 표현 방법, 의사코드, 흐름도	4
	프로그램	프로그램의 기능, 함수, 컴파일러	5
	파이썬	파이썬 소개 및 설치, 변수에 값 저장, 입력, 출력, 조건부 수행	6
		반복, 리스트, 함수, 출력 형식	7
3부 컴퓨팅 사고력 활용하기	데이터의 표현	이진수, 아스키코드, 오디오 데이터, 이미지 데이터, 자료구조	8
		인코딩 및 압축, 오류확인	9
	데이터의 저장과 검색	배열 및 연결 리스트, 선형검색, 이분검색, 색인순차검색, 해싱	10
		이진검색트리, 최대값 및 최소값 검색	11
	알고리즘설계	정렬, 분할정복 알고리즘, 탐욕적 알고리즘	12

정렬 알고리즘

Sorting Algorithm

## 정렬 알고리즘

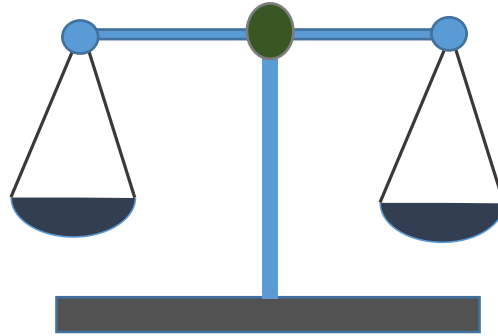
- 데이터의 분석을 위해서는 데이터가 일정한 순서로 되어 있어야 할 필요가 있다.
  - ✓ 학생들의 성적순으로 정렬
  - ✓ 직원들의 명단을 입사 순으로 정렬
  - ✓ 백화점에서 구매 금액이 큰 순서로 고객들을 정렬

- 데이터는 숫자, 문자로 구성되어 있고, 각 구성 요소는 순서가 미리 정해져 있다. – 알파벳 순서 또는 숫자 0, 1, 2, ....
- 정렬: 데이터 값이 증가 또는 감소하는 순서로 데이터를 정리하는 작업
- 예 – 증가하는 순서(오름차순)
  - ✓ 101 129 250 295 307 408
  - ✓ base cartoon fire moon park young
  - ✓ 김 바둑 사회 우주 카누
- 감소하는 순서(내림차순)

- 다양한 정렬 알고리즘이 있는데, 각 알고리즘마다 특징을 갖고 있다.
- 정렬 알고리즘의 특징에 따라, 수행 시간이 크게 다르다.
- 효과적인 정렬 알고리즘의 선택이 중요하다.



- 정렬은 한 번에 두 개의 데이터의 대, 소 값을 비교하는 과정을 통해 진행된다.
- 두 개의 데이터를 천칭에 놓고 대, 소를 비교하는 개념

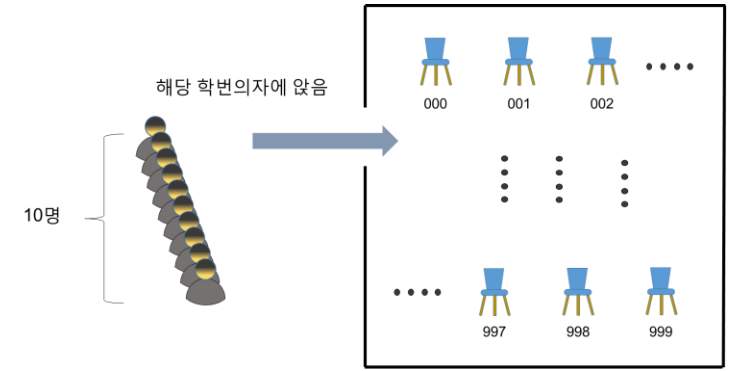


- 오름차순 정렬인 경우 base가 hot 보다 먼저 위치한다.  $\text{base} < \text{hot}$



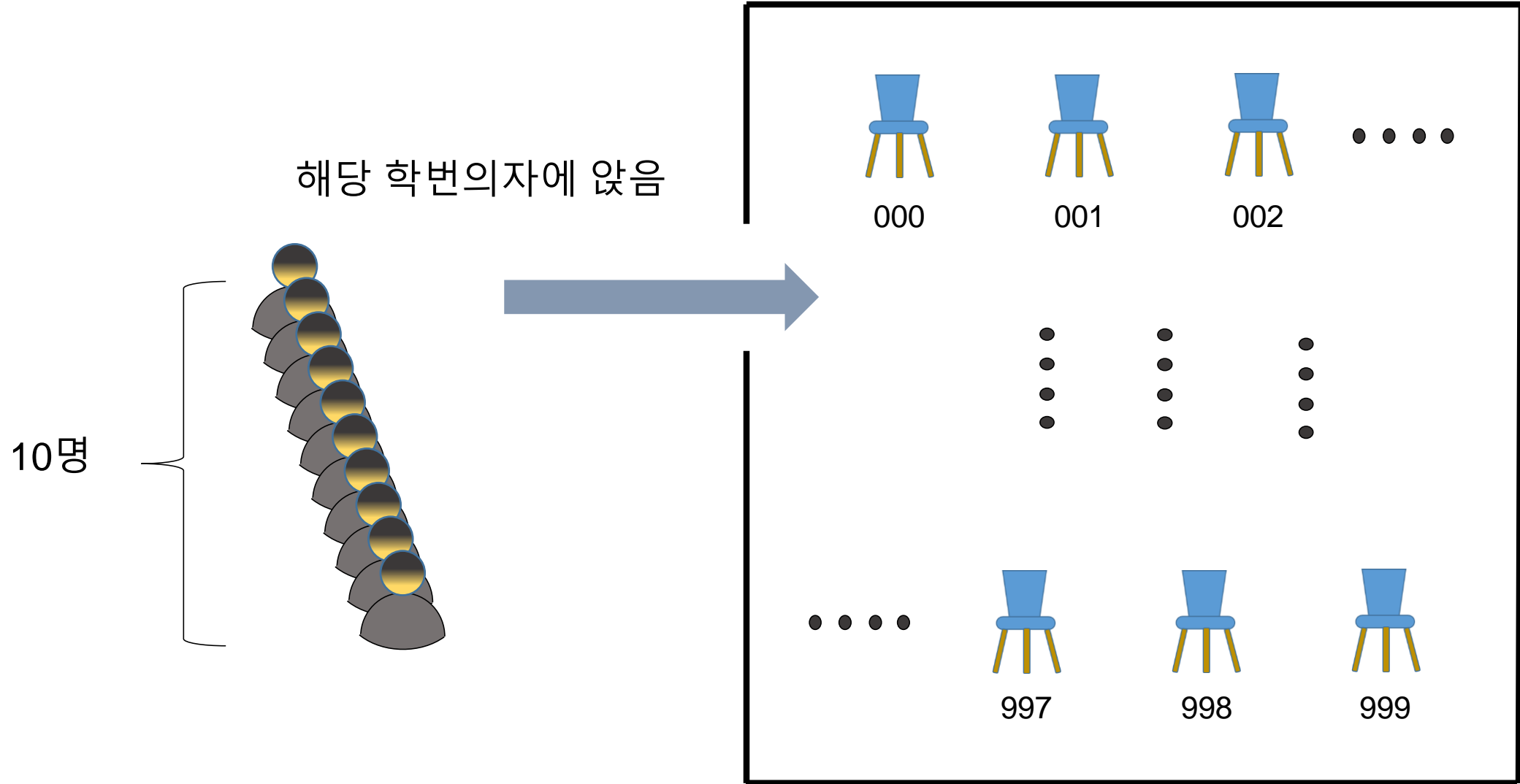
- 데이터의 값 자체를 이용해서 정렬하는 것은 아니다.

- ✓ 예를 들어, 학생이 10명 있고, 십진수 3자리 학번을 갖고 있다고 가정하자. 학번은 000~999.
- ✓ 이 경우 교실에 의자를 1,000개를 준비하고, 각 의자에 000~999 번호를 붙여놓은 후, 학생들에게 한 번에 한 명씩 교실에 들어와, 학번에 해당하는 자리에 앉게 한다.
- ✓ 모든 학생이 자리를 앉게 되면, 학생들은 학번에 따른 정렬이 이루어 지지만, 여기서는 이러한 방법은 고려하지 않는다.
- ✓ 학번 숫자 정보를 직접 사용하는 방법 – 두 데이터의 비교를 이용하고 있지 않다.



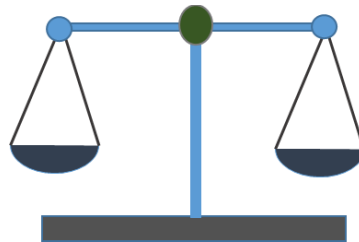
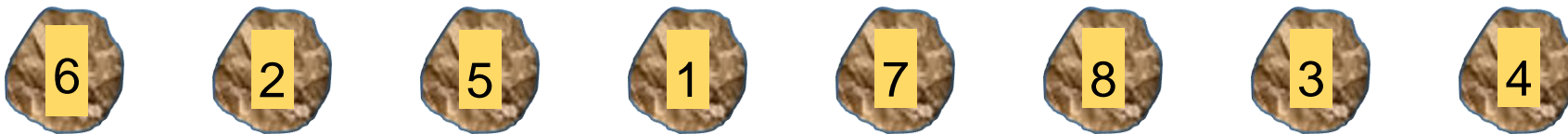


❖ 이 방법도 정렬의 일종이지만, 여기서는 다루지 않는다



[problem]

- 서로 다른 무게의 돌 8개가 있고, 두 개의 돌 무게를 비교할 수 있는 천칭(balance)이 있다.
- 천칭을 사용하여, 한 번에 두 개의 돌 무게를 비교하는 방법만을 이용하여 8개 돌들의 무게가 오름차순이 되도록 돌들을 정렬한다.
- 돌의 모양은 동일하다고 가정. 설명을 위해 돌의 무게를 표시. 작업자는 돌의 무게숫자를 볼 수 없다고 가정.



## 문제의 이해

- 알고 있는 데이터나 정보는 무엇인가?
  - ✓ 데이터의 개수
- 모르고 있는 것은 무엇인가?
  - ✓ 데이터의 대소 순서
- 조건들은 무엇인가?
  - ✓ 한 번에 두 개의 데이터의 대, 소 비교만 가능하다



## ● 문제 해결 과정

- 1.문제를 이해한다
- 2.계획을 세운다
- 3.계획을 실행한다
- 4.풀이과정을 재점검한다

## ● Computational Thinking

- 분해(decomposition)
- 패턴 확인
- 추상화(abstraction)
- 알고리즘 설계
- 데이터 모음
- 데이터 표현
- 데이터 분석
- 모의실험
- 자동화
- 병렬화
- 패턴 일반화

## ● 문제 해결 전략

- 추정과 확인
- 순서 리스트 작성
- 가능성들을 제거
- 대칭 성질을 이용
- 특별한 경우를 고려
- 직접 추론 사용
- 방정식을 푼다
- 패턴을 찾는다
- 그림을 그린다
- 단순한 문제 해결 시도
- 모델 사용
- 역방향으로 시도
- 공식 사용
- 창조적 생각
- 잔피를 사용



# 선택정렬

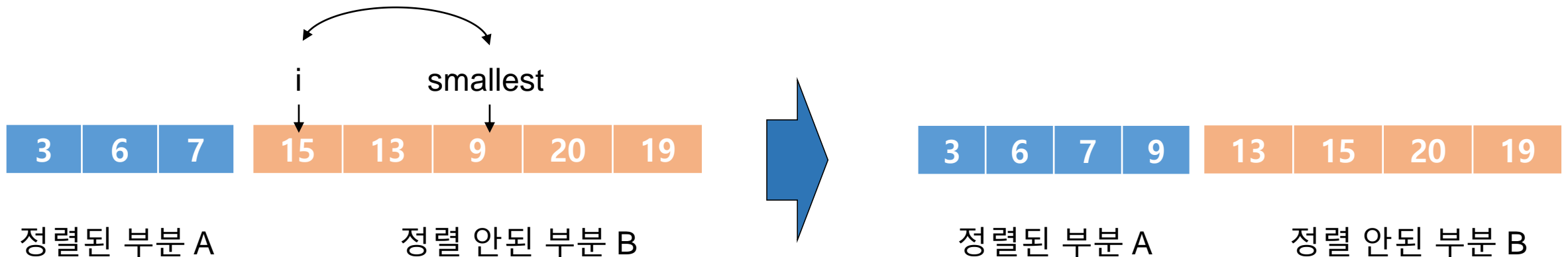
## Selection Sort

<https://www.youtube.com/watch?v=Ns4TPTC8whw>



## [해결 방법] 기본 개념

- 매 단계에서 정렬된 부분 A 와 정렬되지 않은 부분 B로 나눈다.
- B 부분에서 가장 작은 데이터를 찾아 B 부분의 제일 왼쪽의 데이터와 교환한 후 그 데이터를 A 부분으로 편입한다.
- 이 방법을 제일 처음(왼쪽) 데이터부터 끝까지에 적용한다.
- 초기에는 모든 데이터는 B 부분에 속한다.



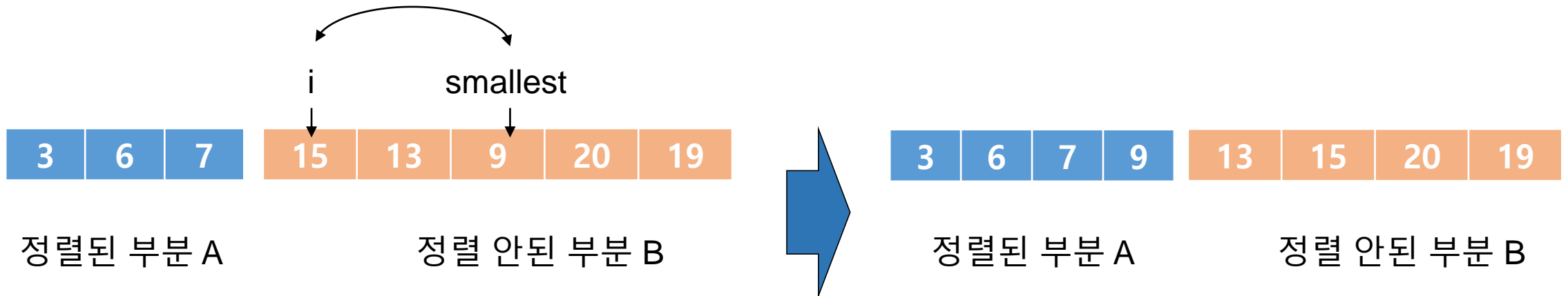
[해결 방법] 데이터가  $a[1] \dots a[n]$ 에 저장되어 있다고 가정

$i=1$

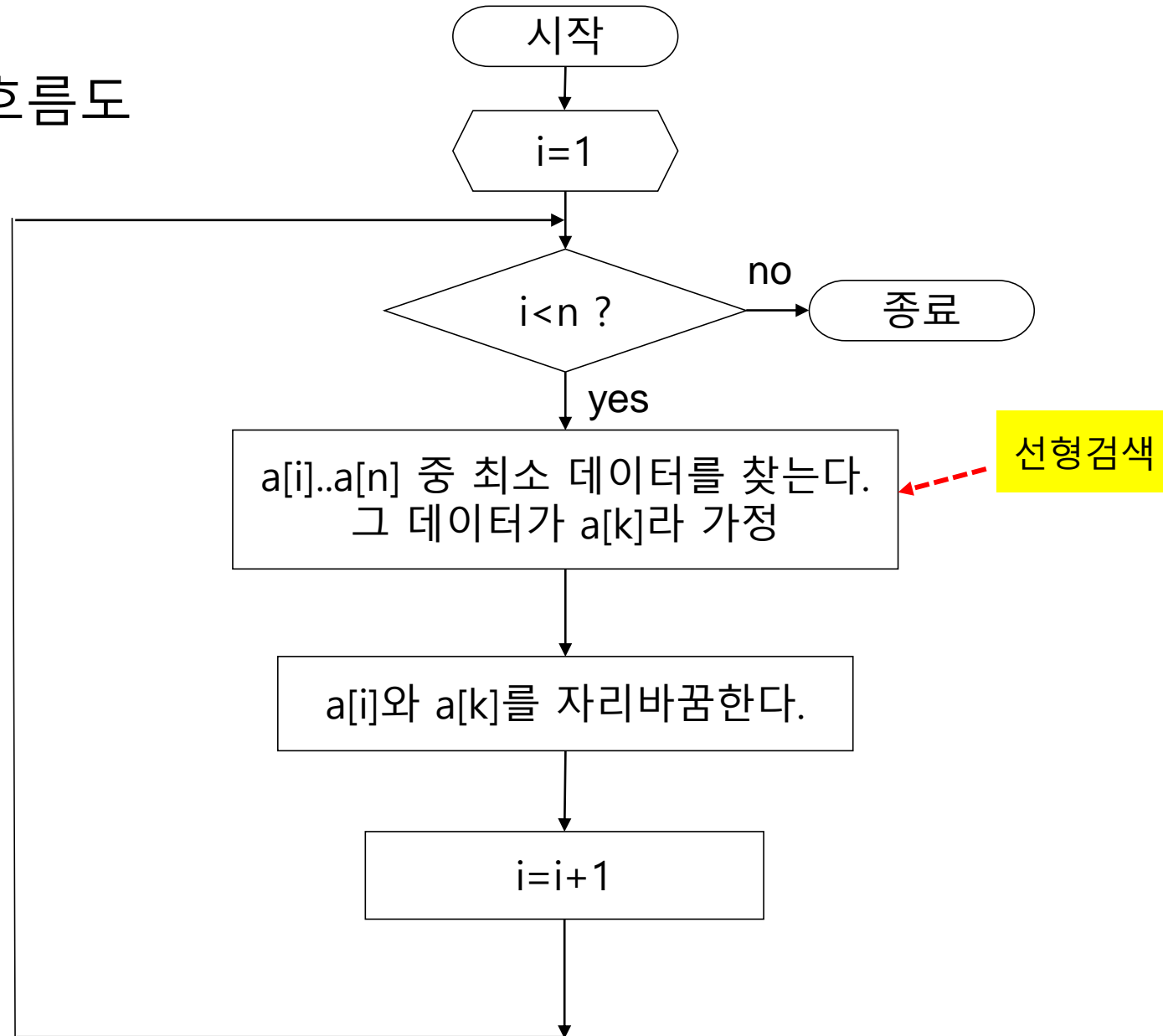
while( $i < n$ )

$a[i] \dots a[n]$  중 제일 작은 값을 찾는다. 그 값이  $a[k]$ 라 가정  
 $a[i]$ 와  $a[k]$ 의 저장위치를 서로 바꾼다.

$i = i + 1$



## 선택정렬의 흐름도





시작

천칭 비교

알고리즘  
진행 과정

M=6

M=2

M=1

이동

7회 천칭작업

최소

교환

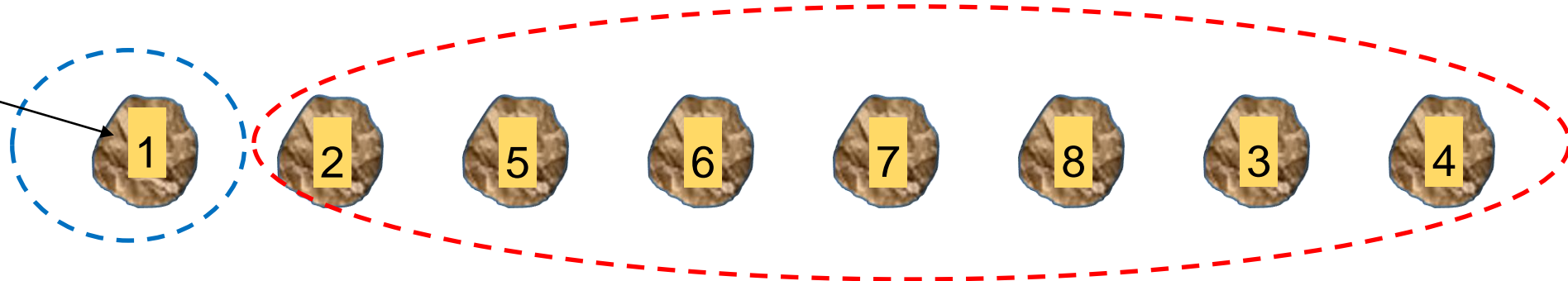


정렬된 데이터

아직 정렬 안된 데이터

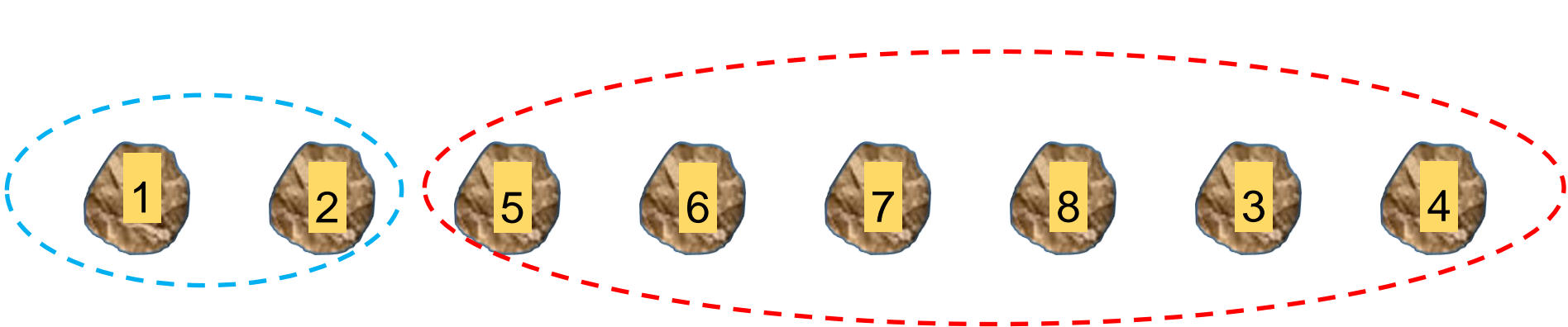
1.

최소



2.

6회 천칭작업



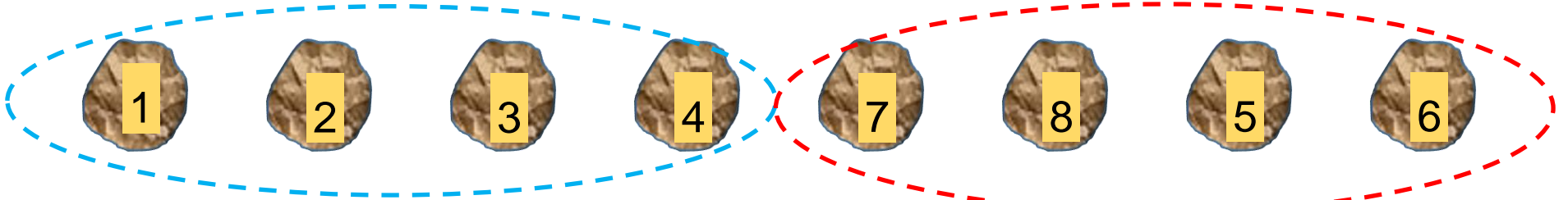
정렬된 데이터

아직 정렬 안된 데이터

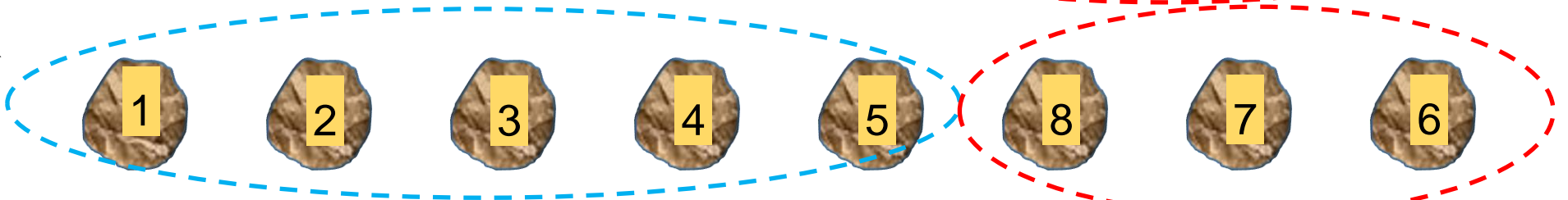
5회 천칭작업



4회 천칭작업



3회 천칭작업

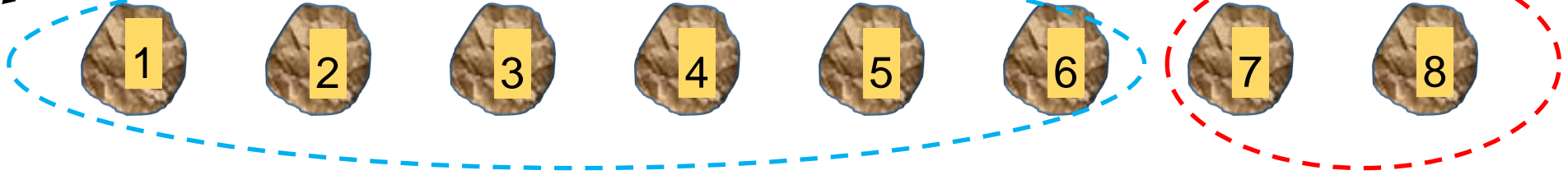


정렬된 데이터

아직 정렬 안된 데이터

2회 천칭작업

6.

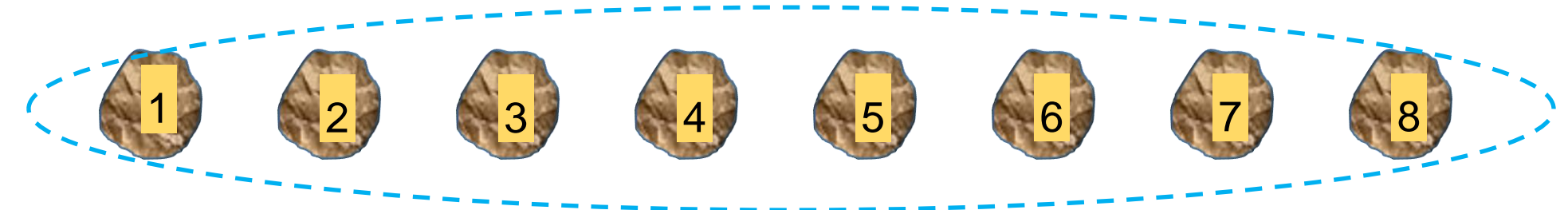


1회 천칭작업

7.



정렬 완료



- 데이터의 개수가 8개 이었다. 총 천칭 작업 횟수는 얼마인가?  
✓  $7+6+5+4+3+2+1 = 28$ 회

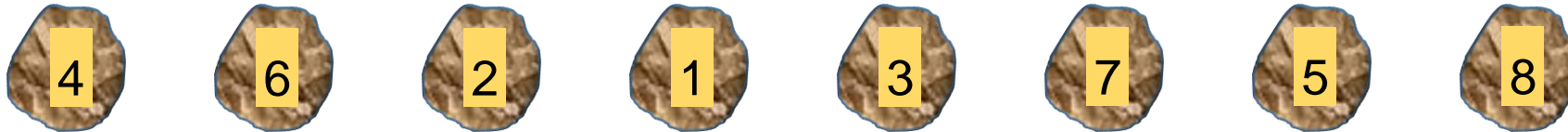
## Computational thinking

- 이미 정렬되어 있는 데이터에 대해 선택정렬을 수행해 보시오.

- 선택정렬은 데이터의 개수가  $n$ 일 때 총  $n(n-1)/2$ 회의 데이터 비교가 필요하다.
- 이미 정렬되어 있는 데이터라고 하더라도 총  $n(n-1)/2$ 회의 데이터 비교가 필요하다.



[연습문제] 다음을 선택정렬 방법을 이용하여 오름차순으로 정렬하시오.



# 삽입정렬

## Insertion Sort

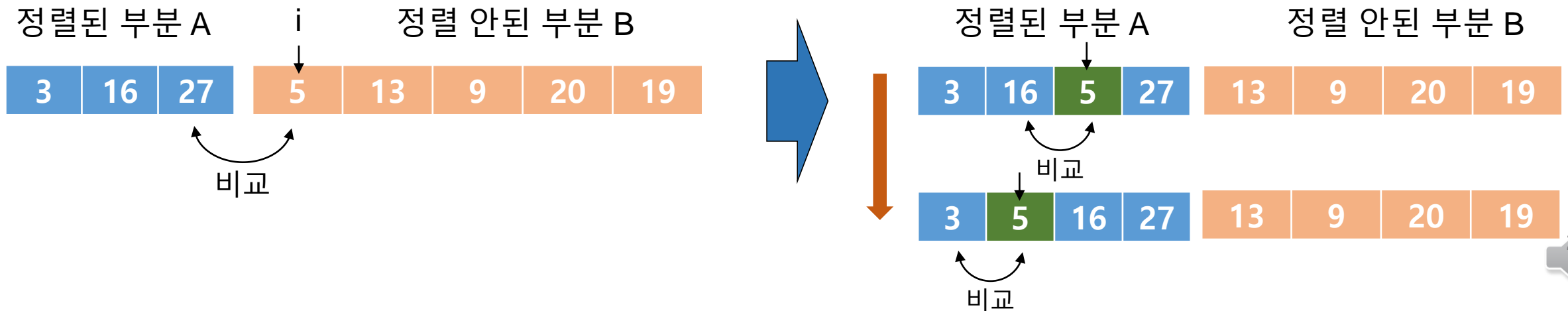
<https://www.youtube.com/watch?v=ROalU379l3U>





## [해결 방법] 기본 개념

- 매 단계에서 정렬된 부분 A와 정렬되지 않은 부분 B로 나누어진다.
- B 부분에서 가장 왼쪽 데이터를 A 부분으로 왼쪽으로 이동하면서 위치할 수 있는 자리를 찾아 준다.
- 자리를 찾았으면, 그 데이터를 A 부분으로 통합한다.
- 이 방법을 제일 처음(왼쪽) 데이터부터 끝까지에 적용한다.
- 초기에는 제일 왼쪽 데이터는 A 부분에 속하고, 나머지 모든 데이터는 B 부분에 속한다.



[해결 방법] 의사코드. 데이터가  $a[1] \dots a[n]$ 에 저장되어 있다고 가정

$i=2$

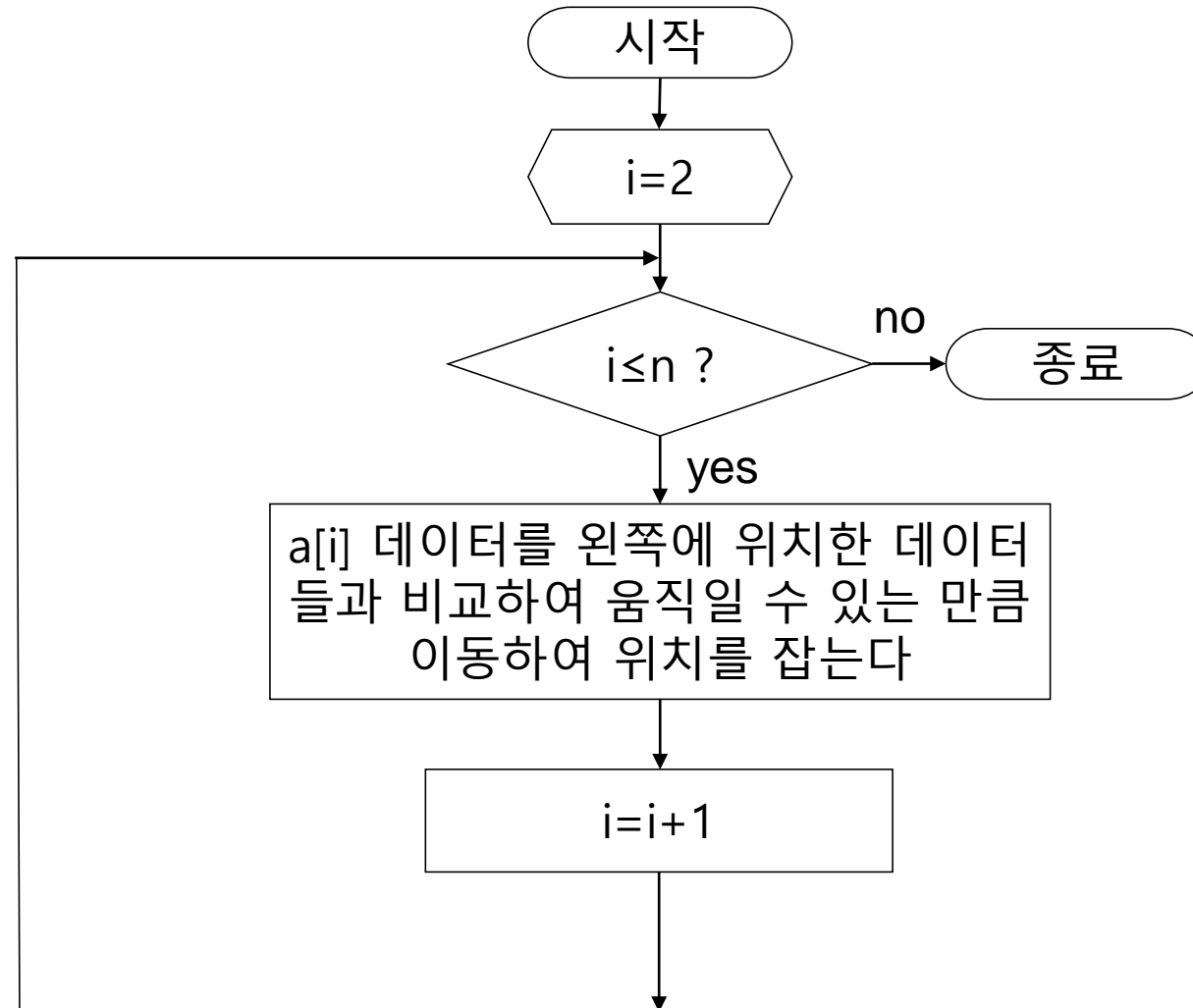
while( $i \leq n$ )

$a[i]$  데이터를 왼쪽에 위치한 데이터들과 비교하여 움직일 수 있는 만큼 이동하여 위치를 잡는다.

- 왼쪽의 데이터 보다 작으면 왼쪽으로 이동
- 왼쪽의 데이터 보다 크면 정지

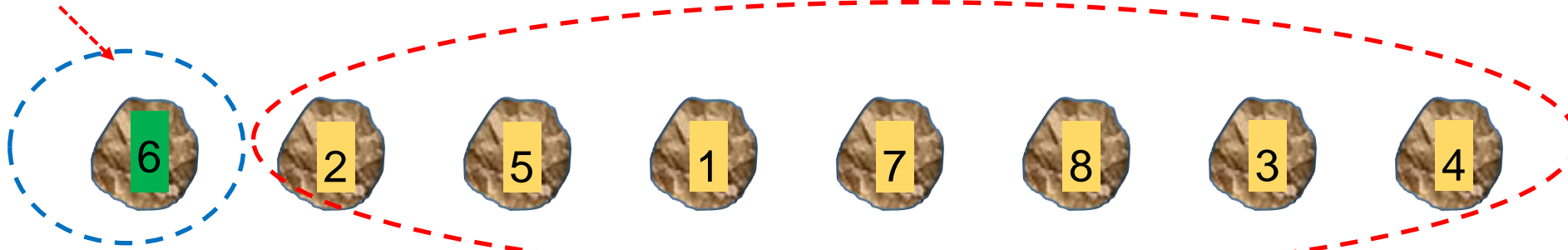
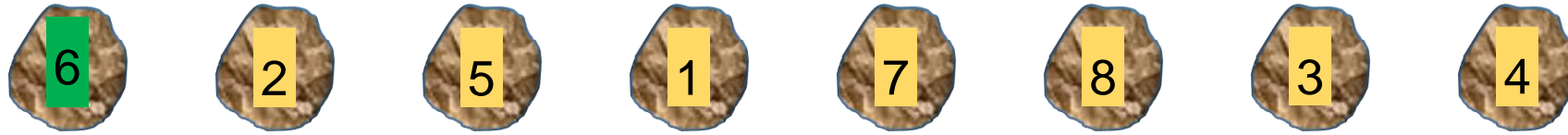
$i = i+1$





시작

천칭 비교



정렬된 데이터

아직 정렬 안된 데이터

- 첫 번째 데이터에 대한 작업 수행은 비교 대상이 없으므로 의미 없다.



천칭 비교

1.



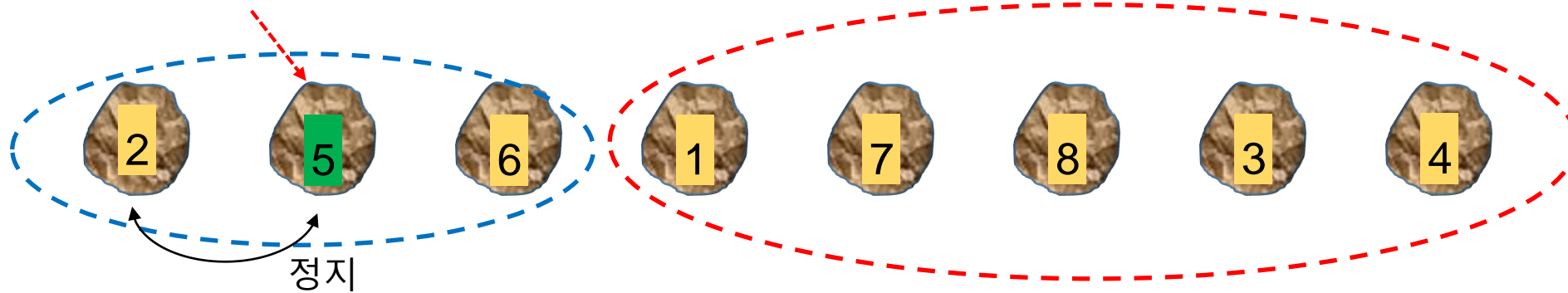
1회 천칭작업

2.



1회 천칭작업

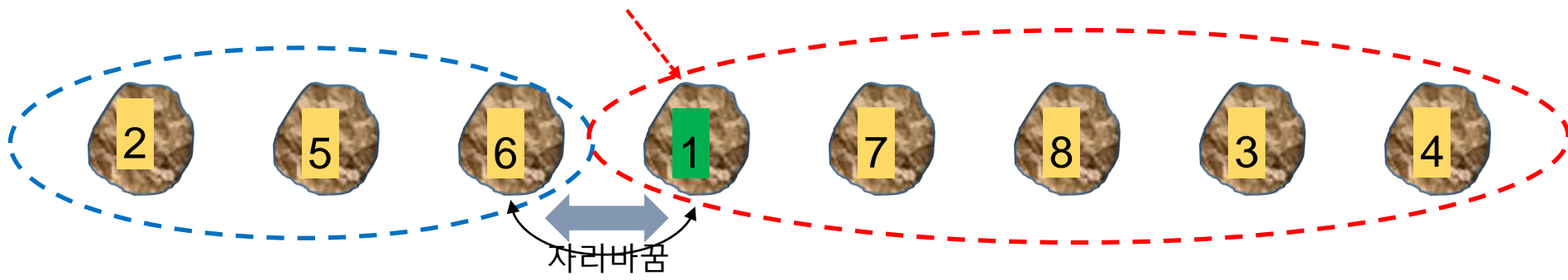
3.



1회 천칭작업



4.



1회 천칭작업

5.



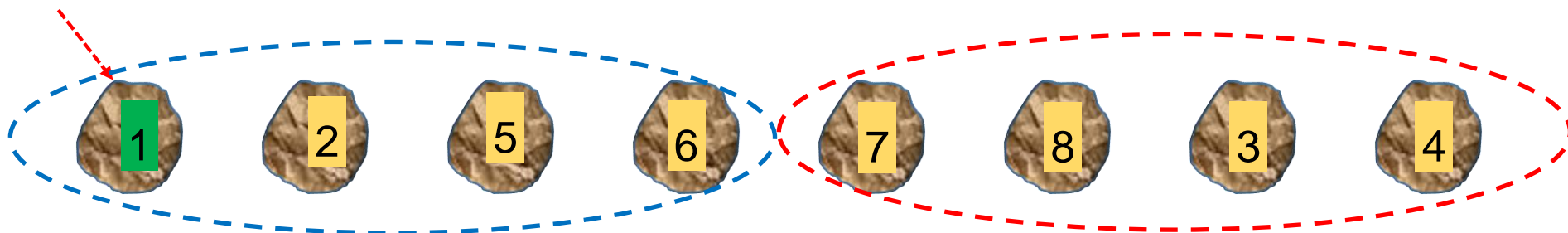
1회 천칭작업

6.

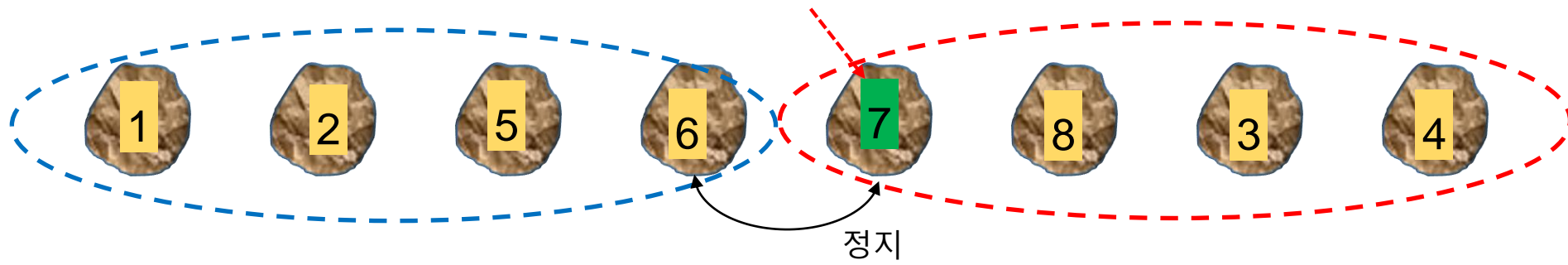


1회 천칭작업

7.

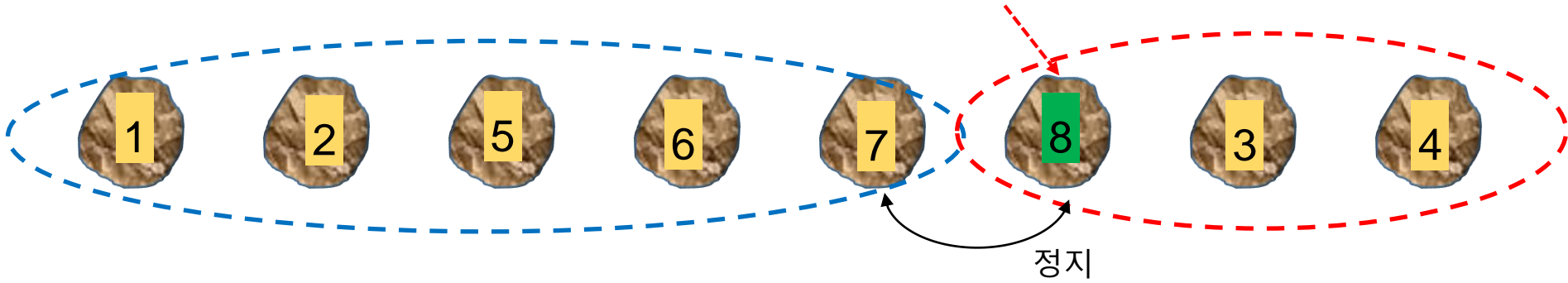


8.



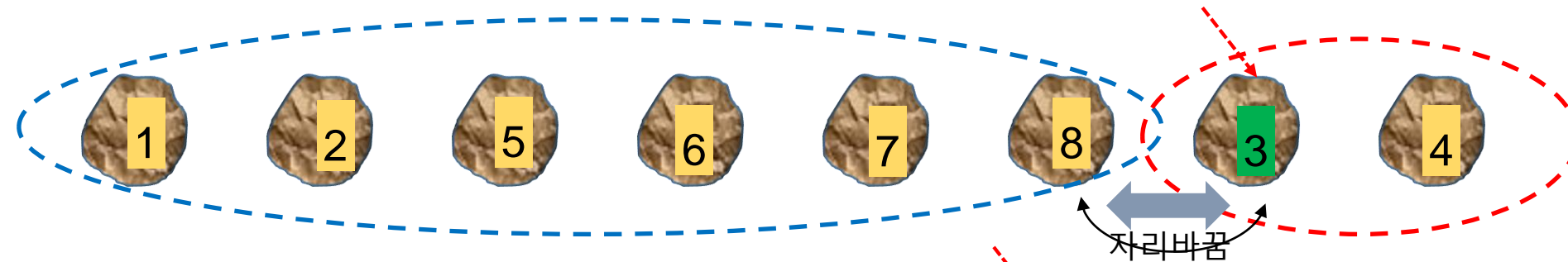
1회 천칭작업

9.



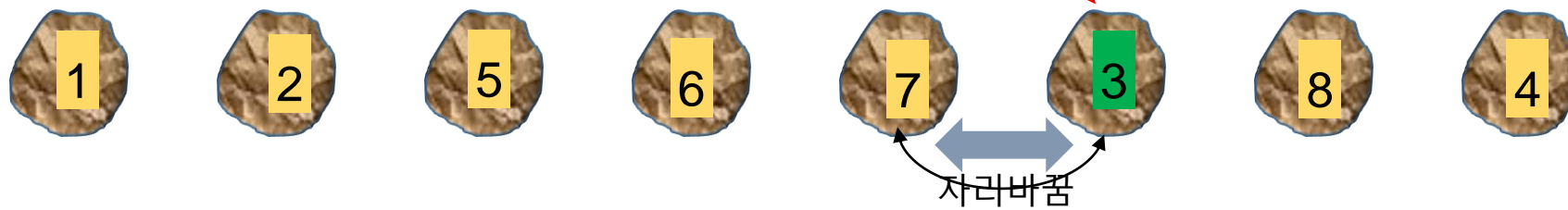
1회 천칭작업

10.



1회 천칭작업

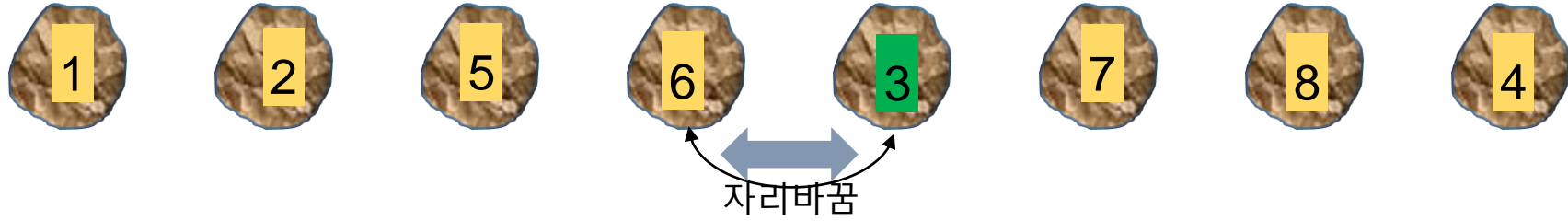
11.



1회 천칭작업

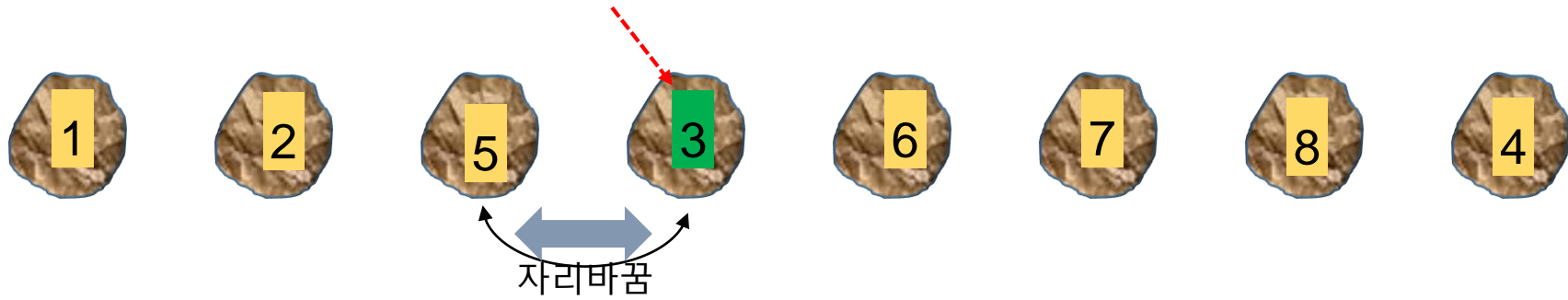


12.



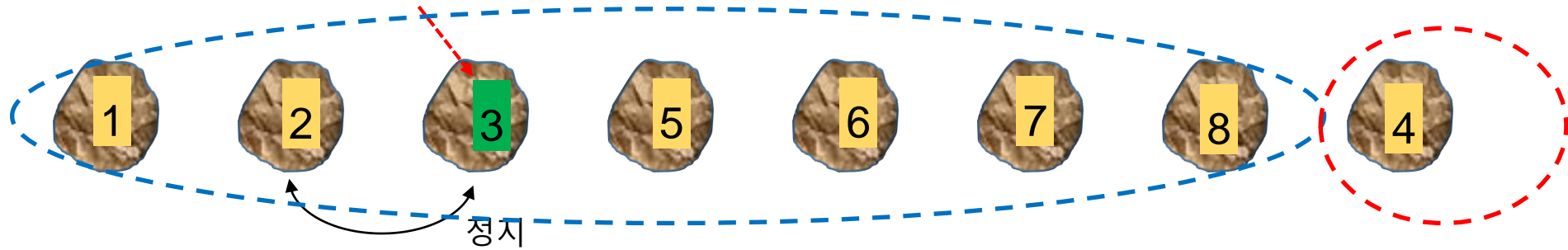
1회 천칭작업

13.



1회 천칭작업

14.

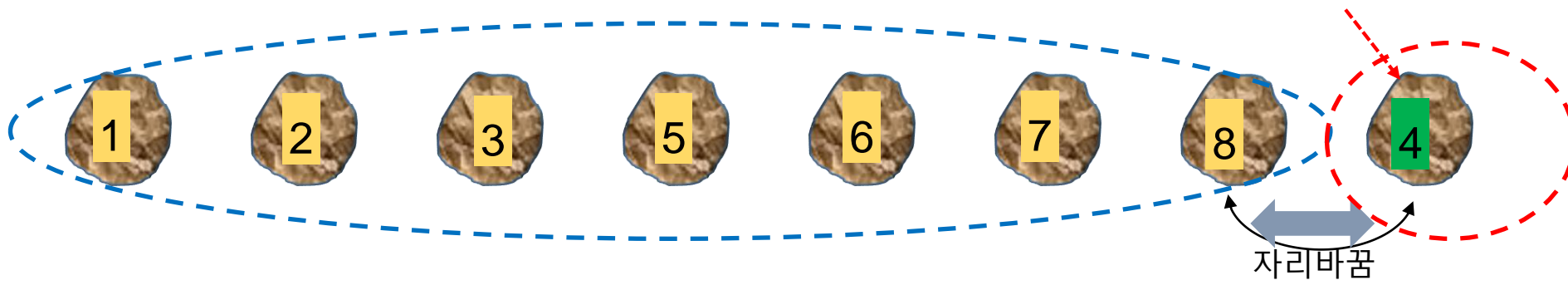


1회 천칭작업



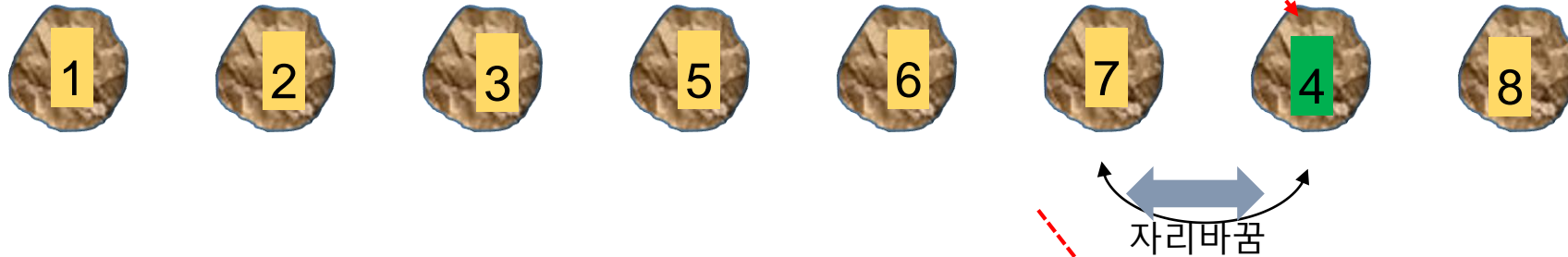


15.



1회 천칭작업

16.



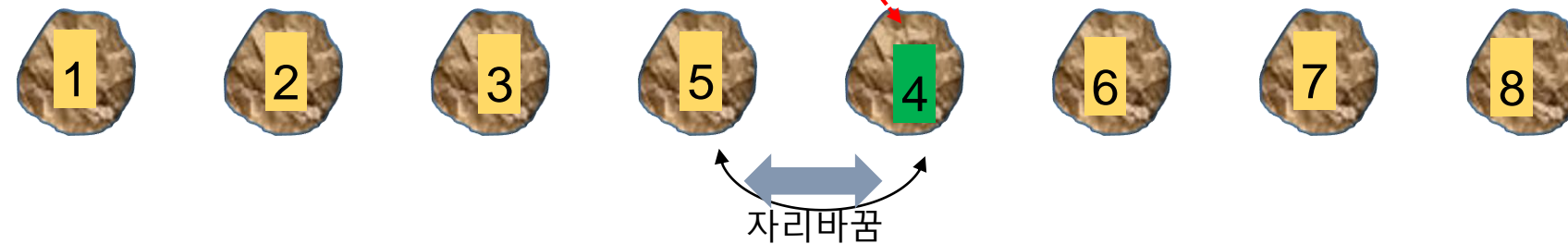
1회 천칭작업

17.



1회 천칭작업

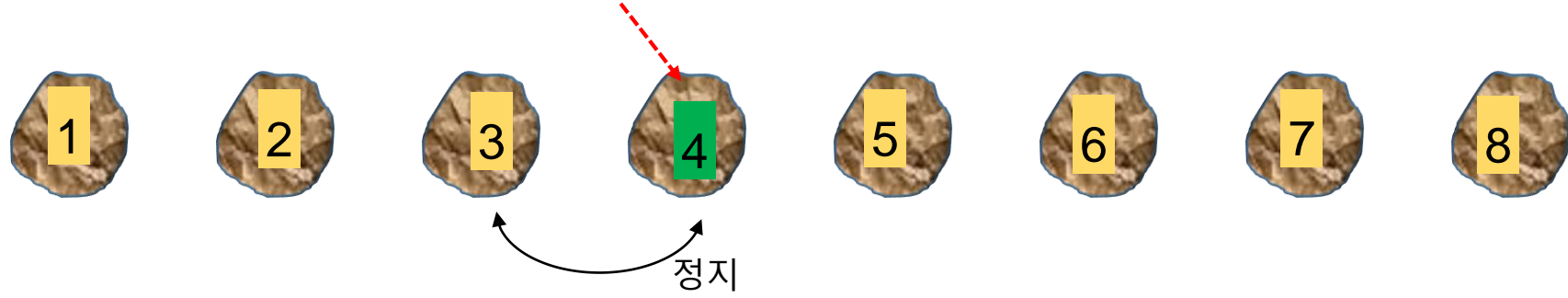
18.



1회 천칭작업



정렬완료



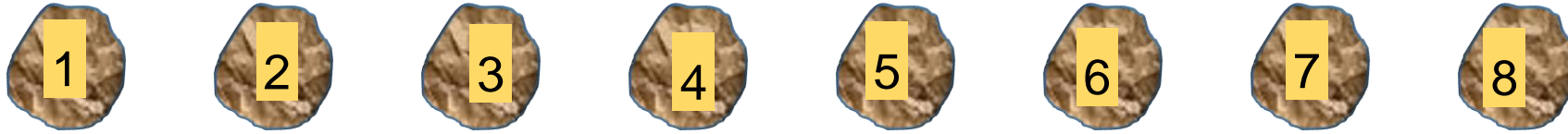
1회 천칭작업

- 데이터의 개수가 8개 였다. 총 천칭 작업 횟수는 얼마인가?
  - ✓ 18회
  - ✓ 동일한 데이터에 대해 선택정렬은 28번의 비교가 필요했다.
  - ✓ 일반적으로 삽입정렬이 선택정렬보다 우수하다.

## Computational thinking

- 서로 다른 정렬 알고리즘은 성능의 차이가 있을 수 있다.
- 이미 정렬되어 있는 데이터에 대해 삽입정렬을 수행해 보시오.

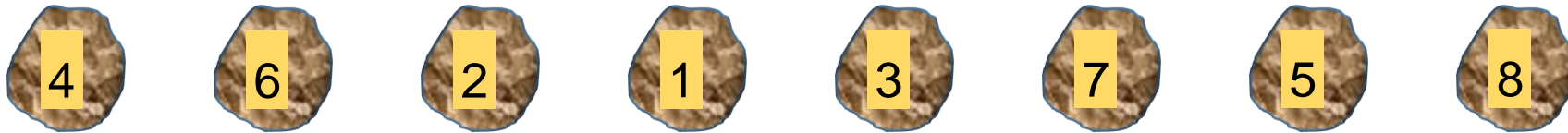
데이터가 이미 정렬되어 있는 경우의 삽입정렬



- 삽입정렬은 데이터의 개수가  $n$ 일 때 총 최대  $n(n-1)/2$ 회의 데이터 비교, 최소  $(n-1)$ 회의 데이터 비교가 필요하다.
- 평균적으로  $n(n-1)/4$ 회의 데이터 비교가 필요하다.
- 일반적으로 선택정렬보다 우수하다고 할 수 있다.



[연습] 다음을 삽입정렬 방법을 이용하여 오름차순으로 정렬하시오.



분할정복 알고리즘

Divide and Conquer Algorithm



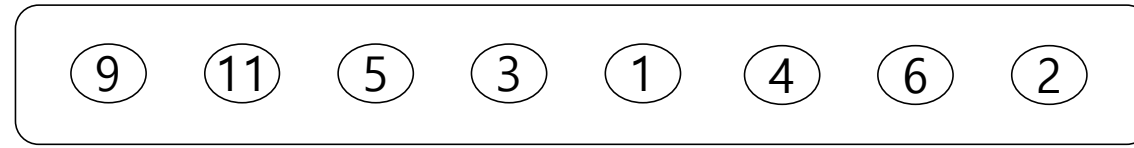


# 분할정복 알고리즘

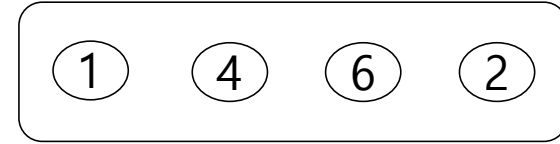
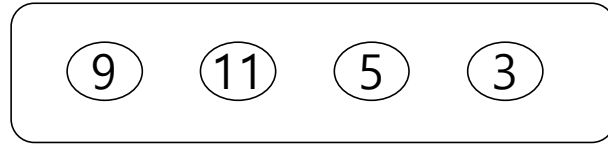
- 문제를 해결하는 알고리즘 유형 중의 하나
- 분할정복 알고리즘의 일반적인 해결 단계
  1. 문제를 작은 문제로 분할한다.
  2. 분할된 문제의 해를 구한다.
  3. 분할된 문제의 해를 이용해서 원래의 분할 이전의 문제의 해를 구한다.
- 이분검색, 빠른정렬(quick sort), 합병정렬(merge sort) 등에 분할정복 알고리즘이 적용된다



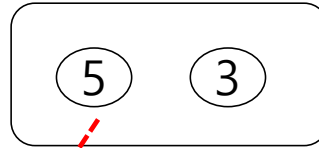
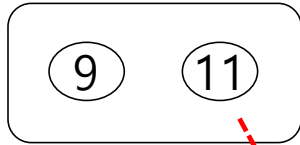
# 제일 큰 수 찾기



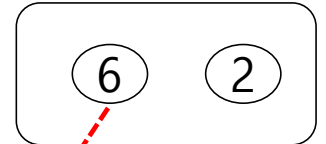
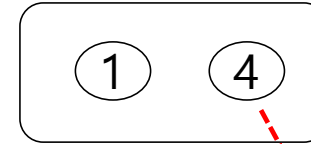
분할



분할

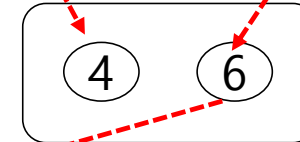
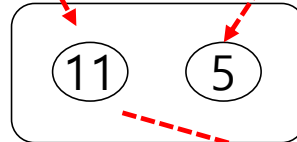


분할

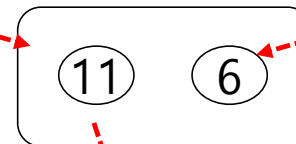


큰 수를 넘겨줌

큰 수를 넘겨줌



큰 수를 넘겨줌



## ● Computational Thinking

- 분해
- 패턴 확인
- 추상화
- 알고리즘 설계
- 데이터 모음
- 데이터 표현
- 데이터 분석
- 모의실험
- 자동화
- 병렬화
- 패턴 일반화



# 이분 검색

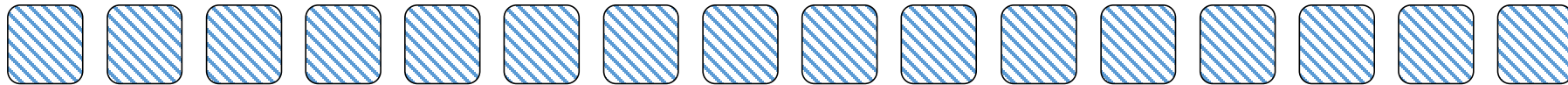
## Binary Search



## [problem]

- 문자가 표시되어 있는 카드 16장을 뒤집어 놓는다. 이 중 특정 카드(예, G)가 표시된 카드를 찾는다
- 이 때 카드는 오름차순으로 정렬되어 있다.

A B C E G H I K M N P R S T V Y



## [해결 방법]

단계 1: 카드열의 중앙에 있는 카드를 뒤집어 카드의 문자를 확인한다. 다음  
의 3가지 경우가 발생한다. ----> 분할

(경우 1) 찾는 문자가 지금 보고 있는 문자보다 작으면 현 카드부터 오른쪽에 있는 카드를 없앤다.

(경우 2) 찾는 문자가 맞다

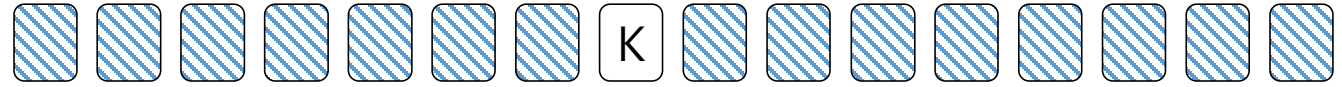
(경우 3) 찾는 문자가 지금 보고 있는 문자보다 크면 카드부터 현 카드부터 왼쪽에 있는 카드를 없앤다.

단계 2: 남아 있는 카드들에 대해 단계 1을 반복하여 문자를 찾는다. ----> 정복



## 문제: G를 찾는다

1. 중앙에 위치한 카드의 문자를 확인한다. 중앙의 위치가 두 개일 때 앞의 것을 선택



2. 찾는 카드가 K보다 작으므로, K가 있는 카드부터 오른쪽에 있는 모든 카드를 제거한다.



3. 중앙에 위치한 카드의 문자를 확인한다.



4. 찾는 카드가 E보다 크므로, E가 있는 카드부터 왼쪽에 있는 모든 카드를 제거한다.



5. 중앙에 위치한 카드의 문자를 확인한다.



6. 찾는 카드가 H보다 작으므로, H가 있는 카드부터 오른쪽에 있는 모든 카드를 제거한다.

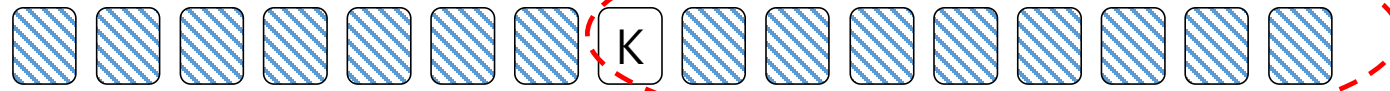


7. 남은 한 장 카드의 문자를 확인한다.



## 문제: G를 찾는다

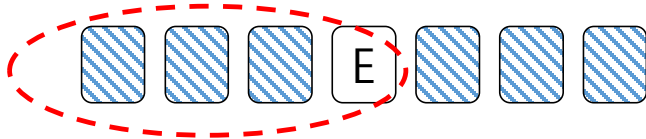
제외



- K를 기준으로 문제를 나눈다: **분할**



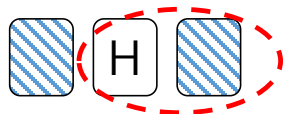
- G가 없는 것이 확실한 데이터를 제외한다.
- 작아진 문제에서 해를 찾는다. **정복**



- E를 기준으로 문제를 나눈다 **분할**



- G가 없는 것이 확실한 데이터를 제외한다.
- 작아진 문제에서 해를 찾는다. **정복**



- H를 기준으로 문제를 나눈다 **분할**



- G가 없는 것이 확실한 데이터를 제외한다.
- 작아진 문제에서 해를 찾는다. **정복**





## ● Computational Thinking

- 분해 (decomposition)
- 패턴 확인
- 추상화(abstraction)
- 알고리즘 설계
- 데이터 모음
- 데이터 표현
- 데이터 분석
- 모의실험
- 자동화
- 병렬화
- 패턴 일반화

## ● 문제 해결 전략

- 추정과 확인
- 순서 리스트 작성
- 가능성들을 제거
- 대칭 성질을 이용
- 특별한 경우를 고려
- 직접 추론 사용
- 방정식을 푼다
- 패턴을 찾는다
- 그림을 그린다
- 단순한 문제 해결 시도
- 모델 사용
- 역방향으로 시도
- 공식 사용
- 창조적 생각
- 잔피를 사용



## Computational thinking

- 문제를 어떻게 나누었나?
- 나누어진 문제는 원래 문제보다 해결하기 쉬운가?
- 꼭 가운데 위치하는 데이터를 이용해야 하나?
- 분할된 문제는 원래의 문제와 같은 형태인가?
- 여기서는 2개의 문제로 분할된다. 3개로 분할할 수 있을까?
- 실생활에서 유사한 해결 방법을 찾을 수 있는가?

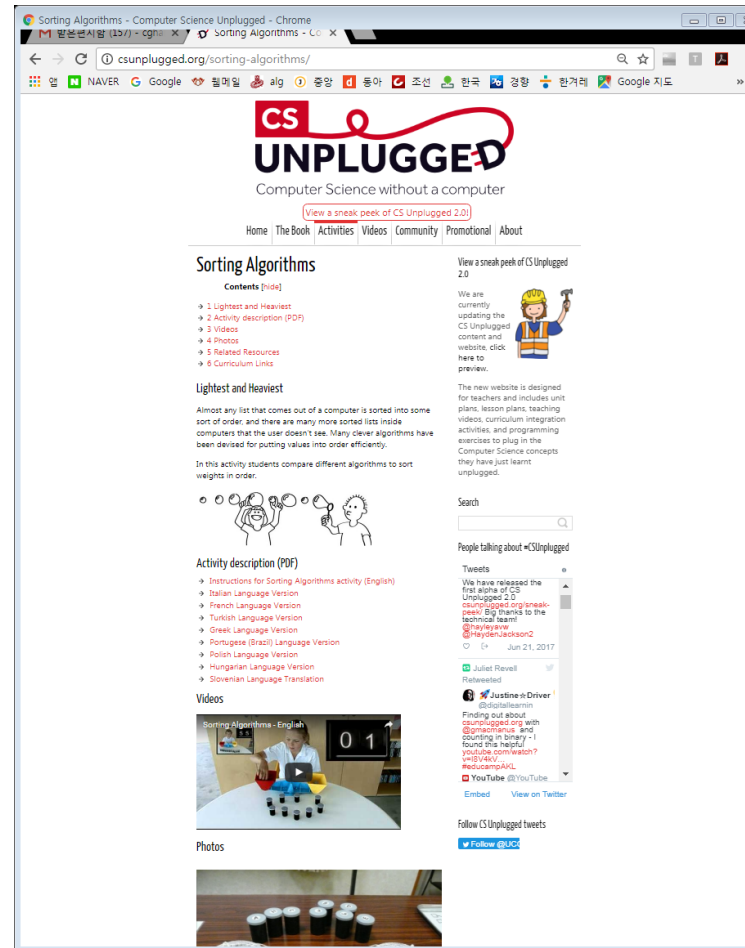


# 빠른정렬

## Quick Sort

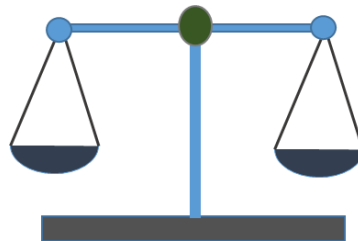
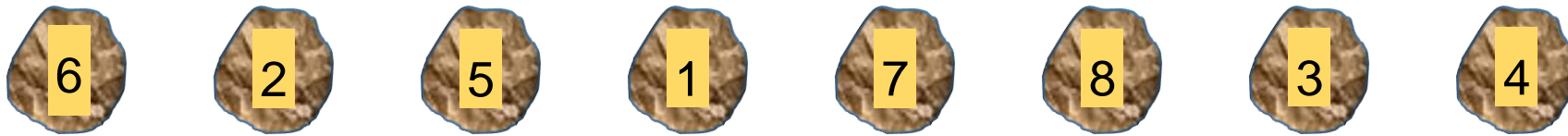
# 선택정렬과 빠른정렬 비교 동영상

<http://csunplugged.org/sorting-algorithms/>



[problem]

- 서로 다른 무게의 돌 8개가 있고, 두 개의 돌 무게를 비교할 수 있는 천칭(balance)이 있다.
- 천칭을 사용하여, 한 번에 두 개의 돌 무게를 비교하는 방법만을 이용하여 8개 돌들의 무게가 오름차순이 되도록 돌들을 정렬한다.
- 돌의 모양은 동일하다고 가정. 설명을 위해 돌의 무게를 표시. 작업자는 돌의 무게숫자를 볼 수 없다고 가정.



- 알고 있는 데이터나 정보는 무엇인가?
  - ✓ 돌의 개수
- 모르고 있는 것은 무엇인가?
  - ✓ 돌의 무게 순서
- 조건들은 무엇인가?
  - ✓ 한 번에 두 개의 돌의 무게를 비교해서, 어느 돌이 더 무거운지를 알 수 있다.

## ● 문제 해결 과정

- 1.문제를 이해한다
- 2.계획을 세운다
- 3.계획을 실행한다
- 4.풀이과정을 재점검한다

## ● Computational Thinking

- 분해(decomposition)
- 패턴 확인
- 추상화(abstraction)
- 알고리즘 설계
- 데이터 모음
- 데이터 표현
- 데이터 분석
- 모의실험
- 자동화
- 병렬화
- 패턴 일반화

## ● 문제 해결 전략

- 추정과 확인
- 순서 리스트 작성
- 가능성들을 제거
- 대칭 성질을 이용
- 특별한 경우를 고려
- 직접 추론 사용
- 방정식을 푼다
- 패턴을 찾는다
- 그림을 그린다
- 단순한 문제 해결 시도
- 모델 사용
- 역방향으로 시도
- 공식 사용
- 창조적 생각
- 잔피를 사용



## [해결 방법]

(단계 1) 임의의 한 개 돌을 집어 천칭에 올린다. 그 돌을 A라 한다. 여기서는 제일 왼쪽에 있는 돌을 선택한다.

(단계 2) 나머지 돌들에 대해, 돌 한 개를 집어 천칭의 반대쪽에 올린다. 그 돌을 B라 한다. 천칭 결과는 다음 두 가지 경우다.

(경우 1) B가 A보다 무거우면, B를 무거운 그룹 H에 넣는다.

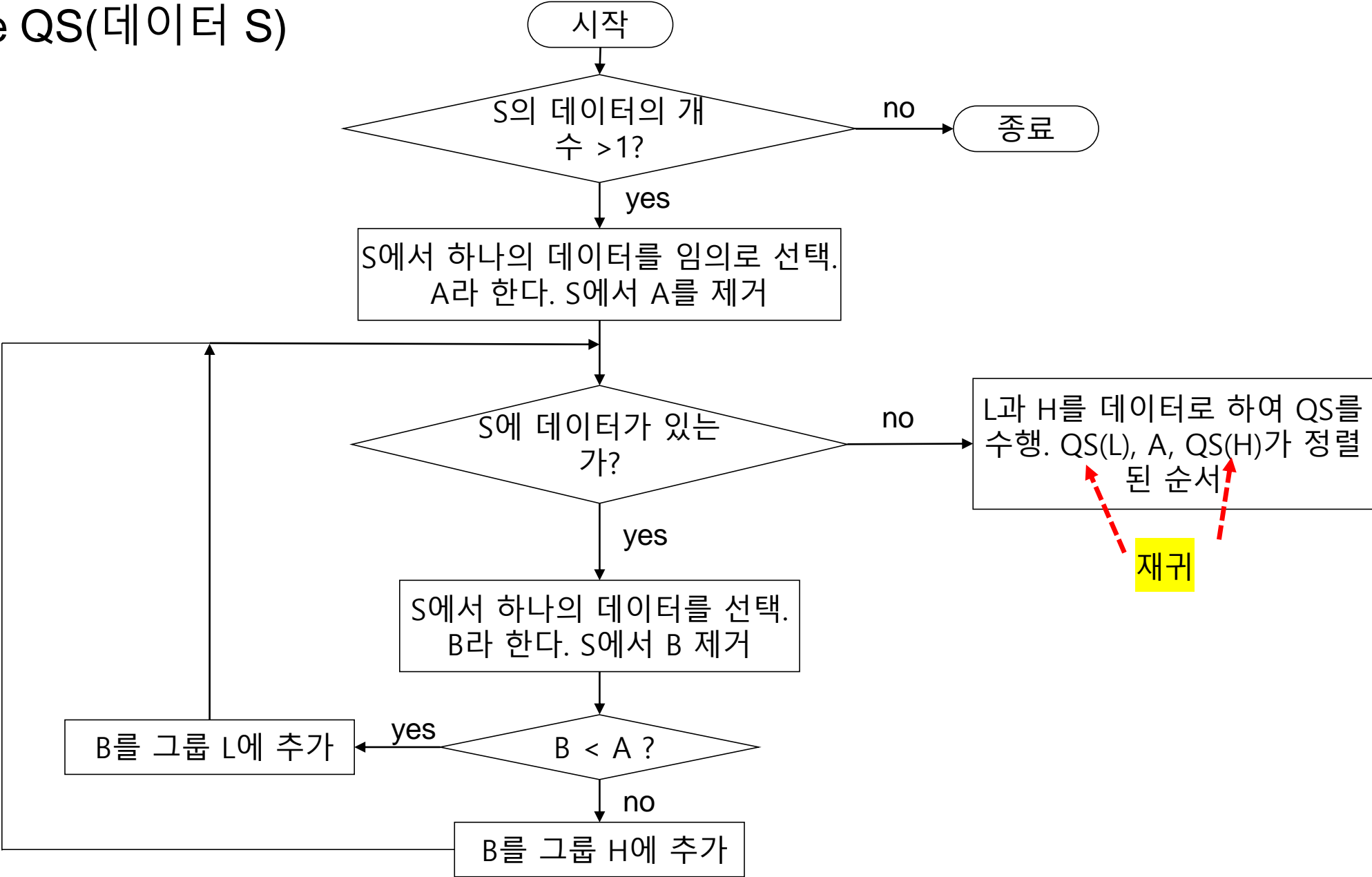
(경우 2) B가 A보다 가벼우면, B를 가벼운 그룹 L에 넣는다.

(단계 3) 돌들은 그룹 L, A, 그리고 그룹 H로 나뉜다. ---> 분할

(단계 4) (단계 1)~(단계 3)을 그룹 L, 그룹 H에 대해 각각 수행한다. ---> 정복



procedure QS(데이터 S)



# 빠른정렬 진행 과정

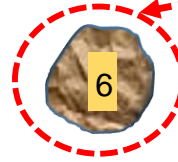
기준돌

(1)

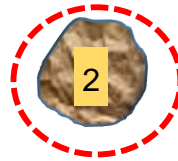


자리고정

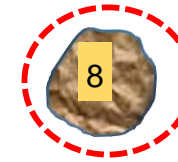
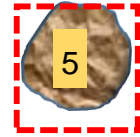
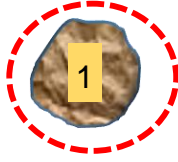
(2)



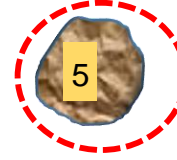
(3)



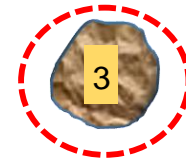
(4)



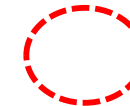
(5)



(6)

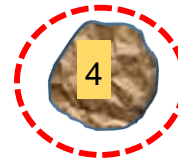


기준돌



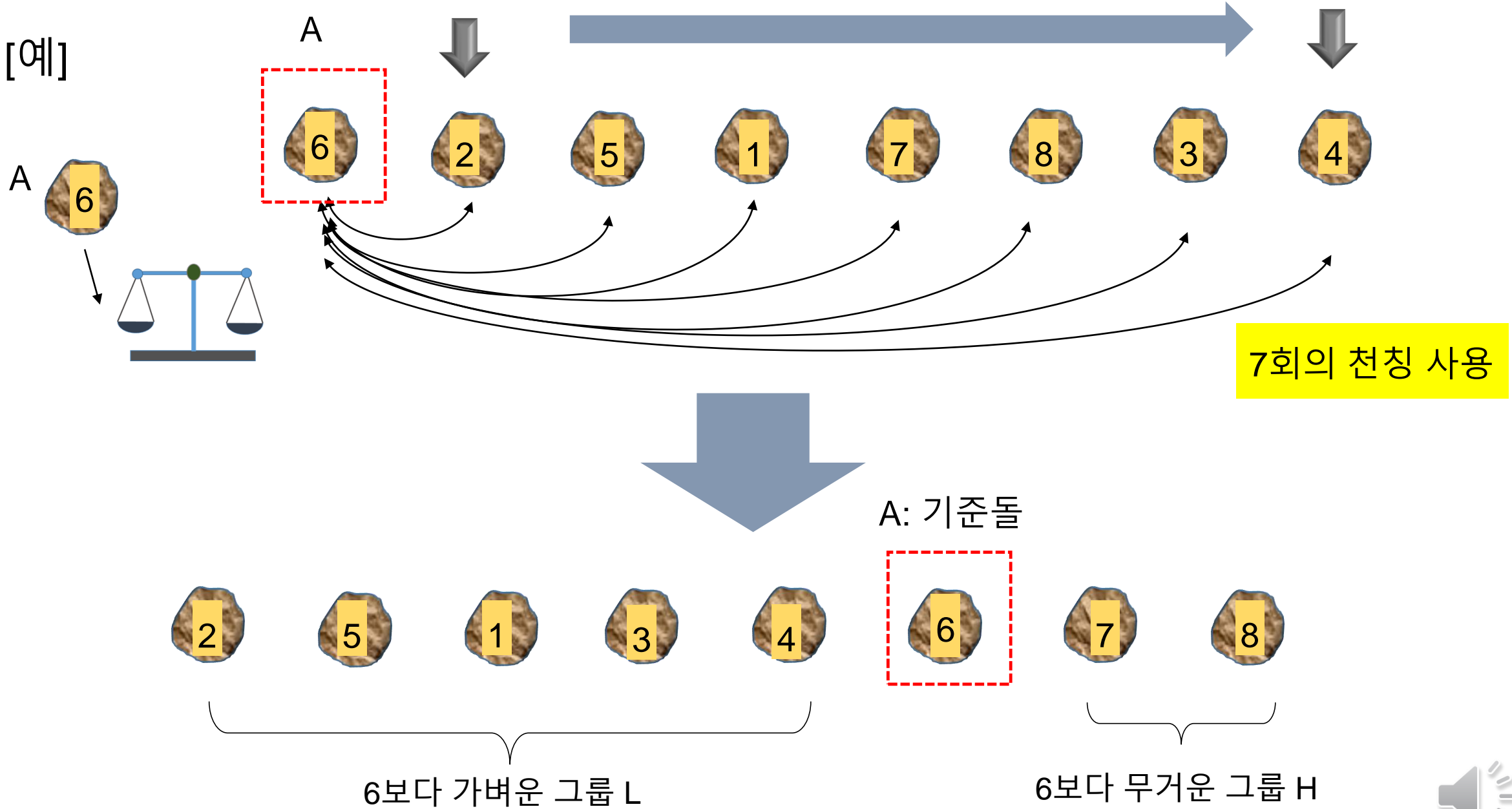
자리고정

(7)



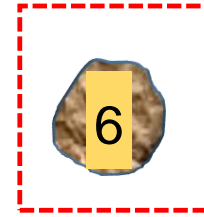
(1)

[예]



(2)

원래의 문제는 다음의 두 문제로 분할된다.



6보다 가벼운 돌들을 정렬하는 문제




6보다 무거운 돌들을 정렬하는 문제



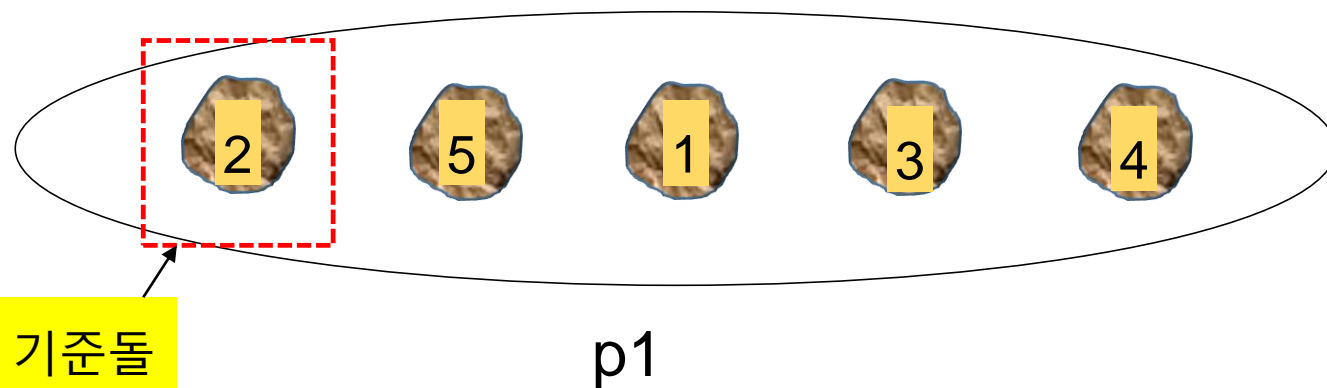
같은 방법으로 빠른정렬 방법을 두 문제에 적용

(3)

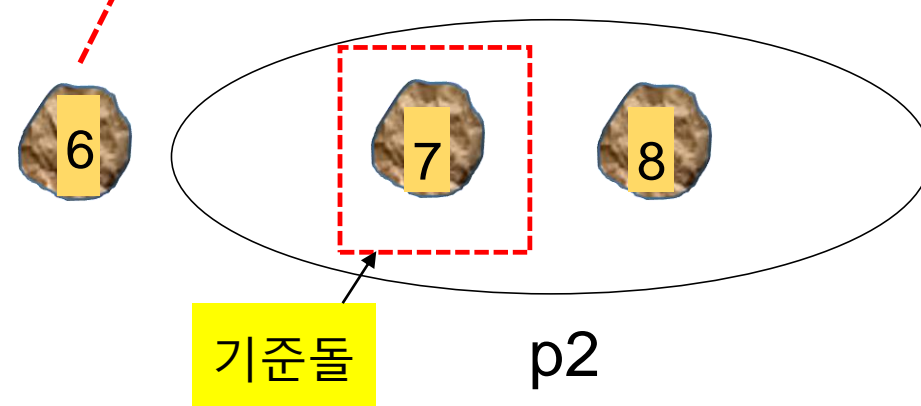
제자리 결정

1	2	3	4	5	6	7	8
							

6보다 가벼운 돌들을 정렬하는 문제



6보다 무거운 돌들을 정렬하는 문제





✓ 같은 방법으로 빠른정렬 방법을 두 문제에 적용



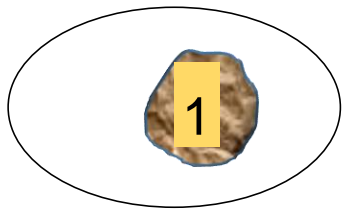
(4)

제자리 결정

1	2	3	4	5	6	7	8
							

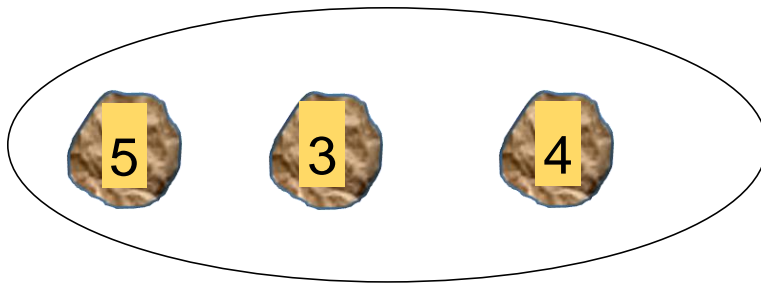
p1은 p3, p4 두 문제로 분할된다.

2보다 가벼운 돌들을  
정렬하는 문제



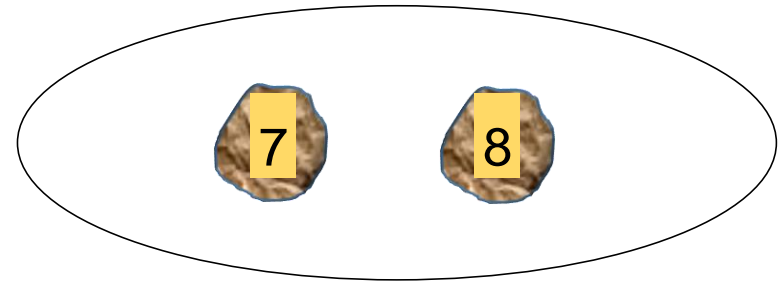
p3

2보다 무겁고, 6보다 가벼운  
돌들을 정렬하는 문제



p4

6보다 무거운 돌들을  
정렬하는 문제



p2




기준돌

4회의 천칭 사용



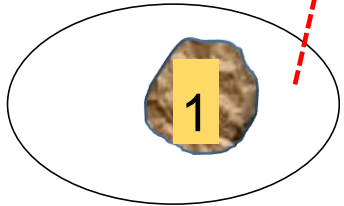
(5)

제자리 결정

1	2	3	4	5	6	7	8
							

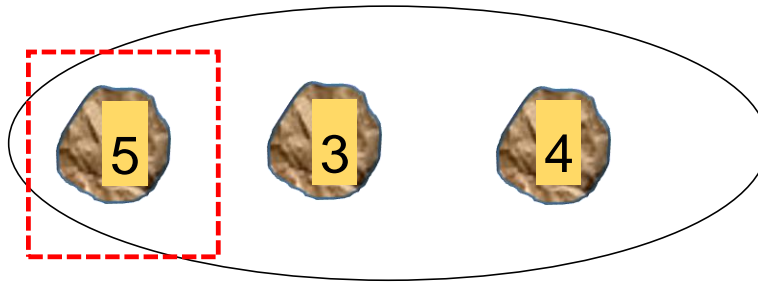
p3은 하나의 돌만 있으므로 제자리 결정

2보다 가벼운 돌들을  
정렬하는 문제



p3

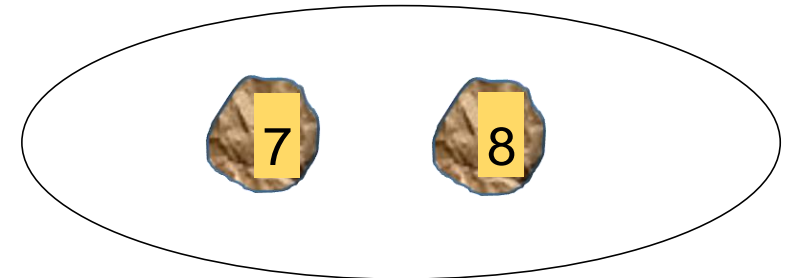
2보다 무겁고 6보다 가벼운  
돌들을 정렬하는 문제



기준돌

p4

6보다 무거운 돌들을  
정렬하는 문제







p2



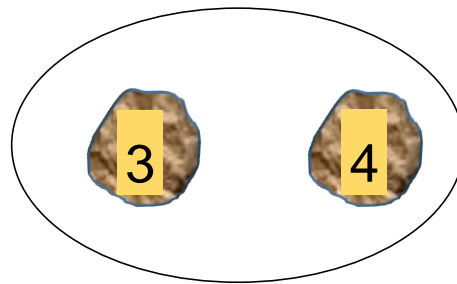
(6)

제자리 결정

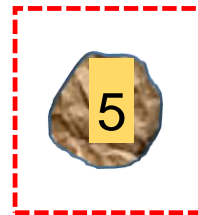
1	2	3	4	5	6	7	8
							

p4는 기준돌 5와 p5로 나뉜다.

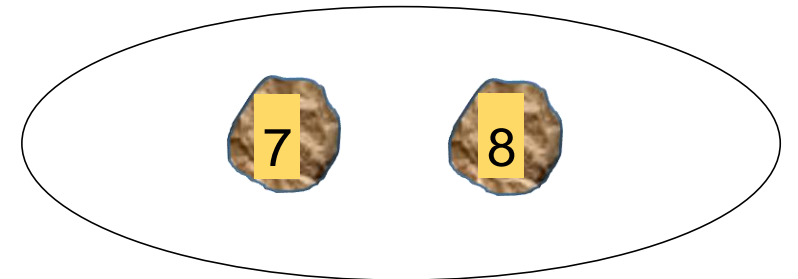
5보다 가벼운 돌들을  
정렬하는 문제



p5



기준돌







p2

2회의 천칭 사용

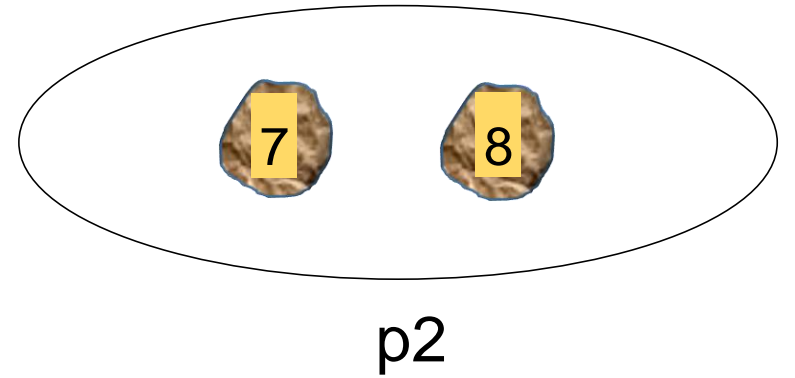
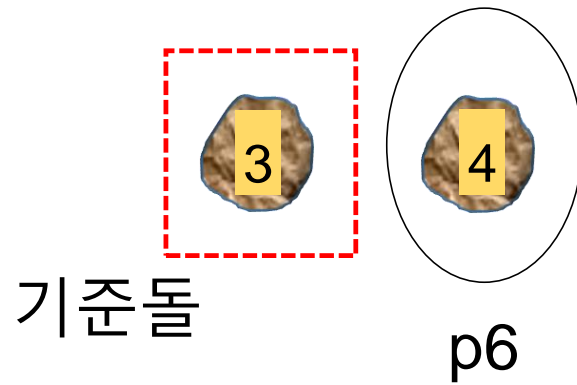


(7)

제자리 결정







1	2	3	4	5	6	7	8
							

p5는 기준돌 3 과 p6로 나뉜다.

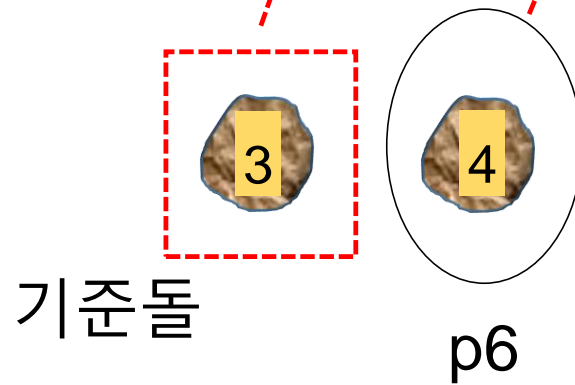


(8)

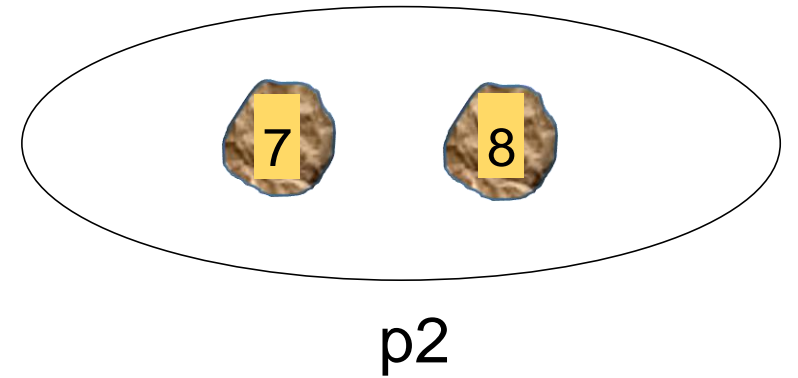
제자리 결정

1	2	3	4	5	6	7	8
							

p5는 기준돌 3 과 p6로 나뉜다. p6은 하나의 돌만 있으므로, 제자리를 찾았다.











1회의 천칭 사용

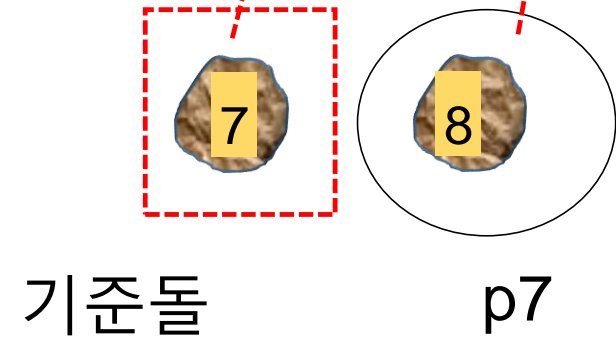


(9)

제자리 결정

1	2	3	4	5	6	7	8
							

p2는 기준돌 7 과 p7로 나뉜다. p7은 하나의 돌만 있으므로, 제자리를 찾았다.



1회의 천칭 사용



- 총 15회의 천칭 비교가 있었다.
- 선택정렬(selection sort)인 경우는  $8*7/2=28$ 회의 천칭 비교가 필요하였다.
- 삽입정렬(insertion sort)인 경우는 18회의 천칭 비교가 필요하였다.
- 분할정복 알고리즘을 이용한 빠른 정렬은 데이터들이 무게로 정렬하였을 때 가운데 위치하는 데이터가 기준돌이 될 때 속도가 빨라진다.
- 그러나, 돌의 무게를 미리 알 수 없으므로, 기준돌을 임의로 선택할 수 밖에 없다.

## Computational thinking

- 빠른정렬 방법은 데이터의 개수  $n=1,000,000$  일 때, 선택정렬보다 약 50,000 배 빠르게 정렬을 수행한다.
- 예를 들어, 휴대폰서비스 회사의 가입자가 백 만명이라고 가정한다. 가입자의 이름 순서로 정렬하는 문제인 경우, 빠른정렬 알고리즘이 1초에 수행하는 일을 선택정렬 알고리즘을 적용하여 문제를 해결한다면, 약 50,000초, 즉 13시간 이상이 걸리게 된다.
- 문제의 규모, 성질에 따라 적당한 알고리즘의 선택이 중요하다.



- 문제를 어떻게 나누었나?
- 기준 데이터로 꼭 가운데 위치하는 데이터를 이용해야 하나?
- 분할된 문제는 원래의 문제와 같은 형태인가?
- 데이터를 어느 한 군데에서도 직접 정렬을 실시하지는 않았지만, 전체가 정렬되는 효과가 나타났다. 기준 데이터를 중심으로 가벼운 데이터, 무거운 데이터로 나눈 것이 정렬의 본질이다.
- 실생활에서 유사한 개념으로 처리할 수 있는 일을 생각해 보자.

- 빠른정렬은 여러 정렬 알고리즘 중에서 우수한 성능을 보이는 정렬 알고리즘이다. 데이터의 개수가  $n$ 일 때, 데이터의 비교 횟수는  $n \log n$ 에 비례한다.
- 문제를 분할할 때, 기준 데이터가 데이터들 중 가장 작은 값이거나, 가장 큰 값일 경우, 나쁜 성능을 보여 준다. 그러나 이러한 가능성은 적으므로, 평균적으로 좋은 성능을 보인다.
- 전형적인 분할정복 알고리즘의 예이다.

# 탐욕적 알고리즘 Greedy Algorithm





## 탐욕적 알고리즘

- 탐욕적 알고리즘은 문제를 해결하는 방법 유형 중의 하나
- 탐욕적 알고리즘은 단계 별로 문제를 해결할 때, 이후에 발생하는 상황을 고려하지 않고, 현재 단계에서 가장 좋다고 판단되는 해를 선정하는 방식이다.
- 현재 단계에서 가장 좋다고 생각되는 해는 가능한 해(feasible solution)인지를 확인하고, 가능한 해가 아닌 경우는 차선의 해를 선정한다.
- 전체 해는 각 단계 별로 얻어진 해를 모아서 구성된다.
- 한 단계에서 결정된 해는 이후 단계에서 다시 고려하지 않으며, 전체 해에서 더 이상 변경되지 않는다.
- 탐욕적 알고리즘이 항상 문제의 최적해를 찾는 것은 아니다.



# 탐욕적 알고리즘 구성 요소

## 1. 선택과정

현재 단계에서 가장 적합한 해를 선택한 후,

## 2. 적정성점검

그 해가 가능한지를 확인한다. 가능하지 않을 경우는 다음의 차선택해를 선택한다.

## 3. 해 점검

문제를 해결했는지를 확인한다.

# 최소 동전 개수 문제

[problem]

동전의 개수가 최소가 되도록 거스름돈을 주는 문제

## Computational thinking

- 우리는 일상 생활을 통해 거스름돈을 동전으로 받을 때 어떻게 받아야 거스름돈의 동전 개수가 최소가 되는지를 잘 알고 있다.
- 그 안에 있는 알고리즘 요소는 무엇일까?
- 어떤 방식으로 알고리즘이 진행되는 것일까?

- 알고 있는 데이터나 정보는 무엇인가?
  - ✓ 거스름돈의 액수
  - ✓ 각 동전의 액수
- 모르고 있는 것은 무엇인가?
  - ✓ 최소 동전의 개수
- 조건들은 무엇인가?
  - ✓ 최소 동전 개수가 되도록 해야 한다

## ● 문제 해결 과정

- 1.문제를 이해한다
- 2.계획을 세운다
- 3.계획을 실행한다
- 4.풀이과정을 재점검한다

## ● Computational Thinking

- 분해(decomposition)
- 패턴 확인
- 추상화(abstraction)
- 알고리즘 설계
- 데이터 모음
- 데이터 표현
- 데이터 분석
- 모의실험
- 자동화
- 병렬화
- 패턴 일반화

## ● 문제 해결 전략

- 추정과 확인
- 순서 리스트 작성
- 가능성들을 제거
- 대칭 성질을 이용
- 특별한 경우를 고려
- 직접 추론 사용
- 방정식을 푼다
- 패턴을 찾는다
- 그림을 그린다
- 단순한 문제 해결 시도
- 모델 사용
- 역방향으로 시도
- 공식 사용
- 창조적 생각
- 잔피를 사용



## [해결 방법] 탐욕적 알고리즘

- 거스름돈을  $p$ 라 하자.
- 동전을 가장 금액이 큰 동전부터 작은 동전 순서로 나열한다.  $c_1 > c_2 > c_3 \dots$
- 다음을  $i=1,2,\dots$ 에 대해 반복 수행한다. 동전금액이 큰 순서로 확인 - 선택과정

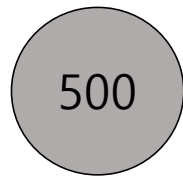
(단계 1)  $p$ 를 초과하지 않는 한도에서  $c_i$ 원 짜리 동전을 1개, 2개, ...를 준다.  $y$ 개까지 줄 수 있다고 가정한다.  $p$ 를 초과하는지 확인- 적정성점검

(단계 2) 거스름돈을 갱신한다.  $p = p - y \times c_i$ .

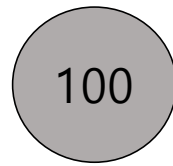
(단계 3)  $p=0$  인지 확인한다.  $p=0$ 이면 종료.  $p \neq 0$ 이면 (단계 1)로 가서 반복  $p=0$ 인지 확인- 해 점검

- 현재 우리나라에서 사용되는 동전 체계에서 대해 이 알고리즘을 적용하여 거스름돈을 동전으로 준다면, 최소한의 동전 수를 얻을 수 있다.
- 탐욕적 방법을 기반으로 한 알고리즘이 최적해를 구할 수 있다.

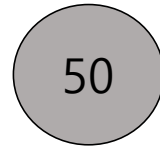
## 사용되는 동전



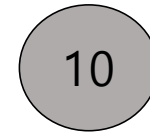
500원



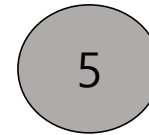
100원



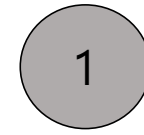
50원



10원



5원



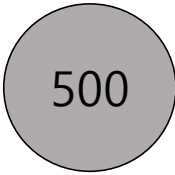
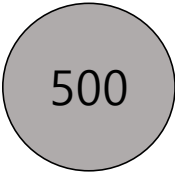
1원

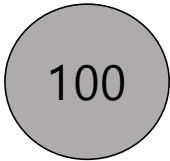
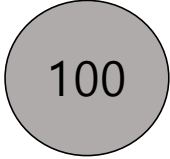




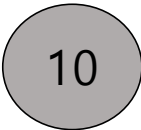
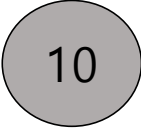
## (예) 거스름돈 1,278원

세 개는 불가능

(1) 1,278원에 대해   1,000원 까지 지불 가능. 나머지 278원

(2) 278원에 대해   200원 까지 지불 가능. 나머지 78원

(3) 78원에 대해  50원 까지 지불 가능. 나머지 28원

(4) 28원에 대해   20원 까지 지불 가능. 나머지 8원

(5) 8원에 대해  5원 까지 지불 가능. 나머지 3원

(6) 3원에 대해    3원 까지 지불 가능. 나머지 0원. 완료 동전 수=11개



- (1)에서 1,278원을 거슬러 주기 위해 최대로 줄 수 있는 500원짜리 동전의 개수를 정하는 작은 문제가 단계별로 있는 작은 문제이다.
- 이 단계의 문제의 답은 500원짜리 2개가 되는데, 3개가 되면 거스름돈을 초과한다는 사실을 확인하고, 단계의 답으로 확정하게 된다.
- 이후에 500원짜리 동전의 개수를 정하는 문제는 더 이상 고려하지 않게 되며, 이 답은 이후에 변경되지 않는다.



## [풀이과정 재점검]

- 최소의 동전 개수가 되었는지 확인하시오.
- 큰 액수의 동전부터 확인하는 것이 올바른 방법이라는 것을 어떻게 알 수 있나?
- 매 단계의 문제는 이전 단계의 문제와 거스름돈의 액수, 고려하는 동전의 금액만 다르고, 계산 방식은 동일하다.

## Computational thinking

- 문제를 어떻게 나누었나?
- 한 번 결정된 동전의 개수(이전 예에서, 500원 동전의 개수 2개)는 이후 알고리즘이 진행하면서 바뀌지 않는다. 이 점은 탐욕적 알고리즘의 특징이다.
- 다음 단계의 동전의 개수가 적절한지 어떻게 검증했나?
- 알고리즘이 끝나는 것을 어떻게 확인할 수 있나?
- 하루 일과를 결정하는 문제, 미래의 인생을 준비하는 문제를 해결하는 데, 탐욕적 방법을 사용하는 것이 적절할까?

# [탐욕적 알고리즘의 예]

## 후프만코드

- 주어진 데이터를 가장 짧게 해주는 전치(prefix)코드로 만드는 방법

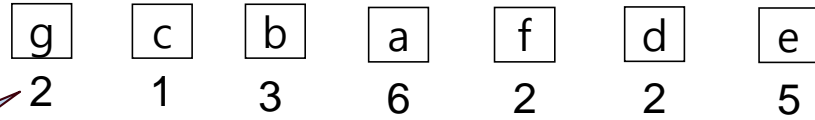
- 후프만코드 생성 방법

- (1) 인코딩하려는  $n$ 개의 데이터에 대해 빈도수를 표시하여  $n$ 개의 노드 생성
- (2) 두 노드의 빈도수의 합이 최소가 되는 노드를 찾는다.
- (3) 두 노드를 합병시켜서 이진트리로 만든다.
- (4) 모든 노드가 하나의 이진트리로 합쳐질 때까지 단계(2) (3)을 반복



(예) 파일=cbbgaaaffdeebgaaadeeee

[1]

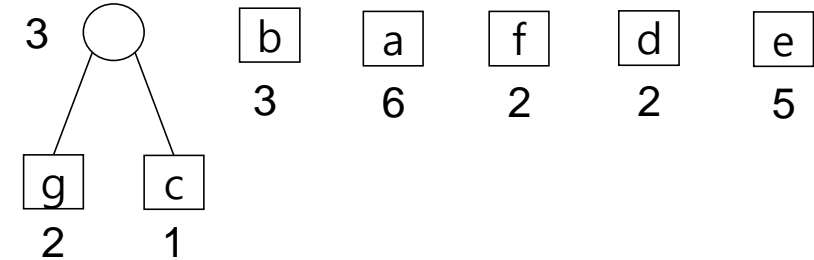


빈도수

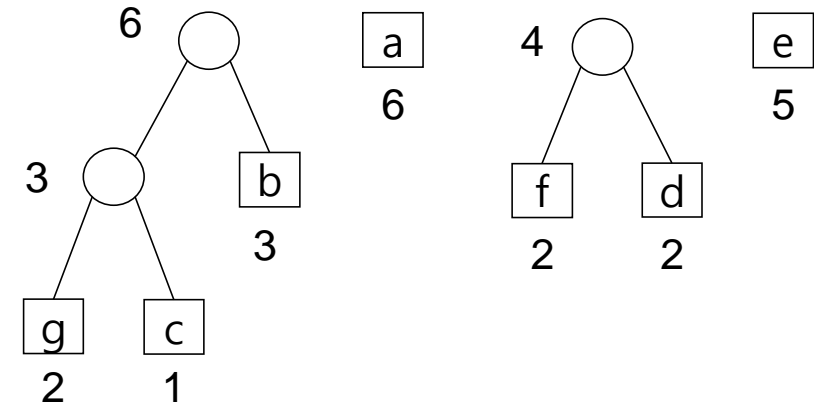
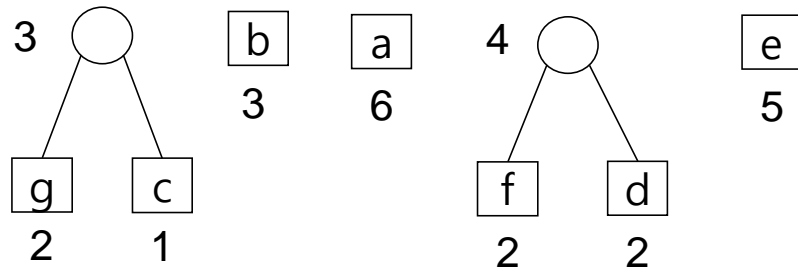
g는 2번 나타난다는 뜻

[2]

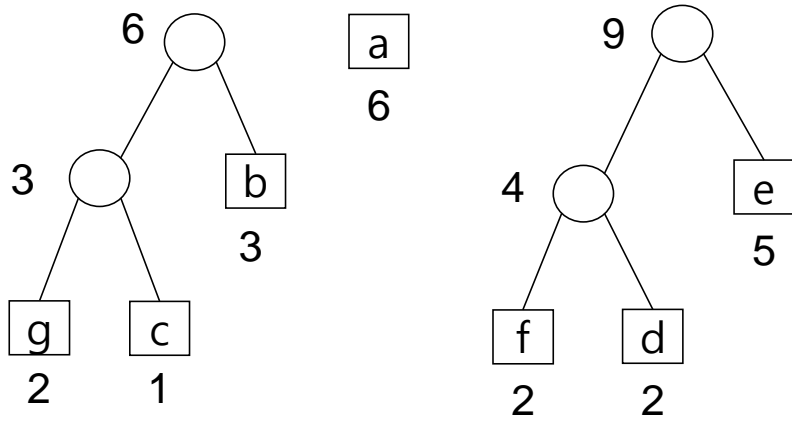
2+1=3이 두 수를  
합칠 때 가장 작은 경우



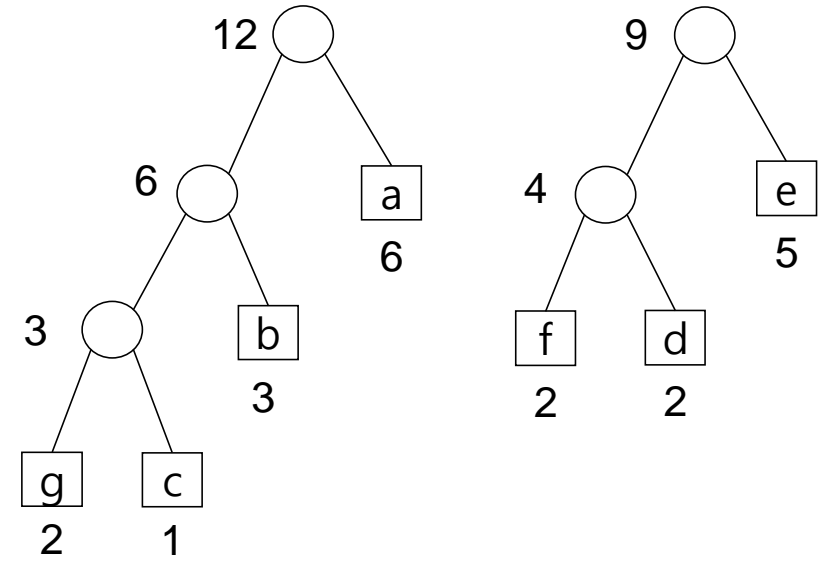
[4]



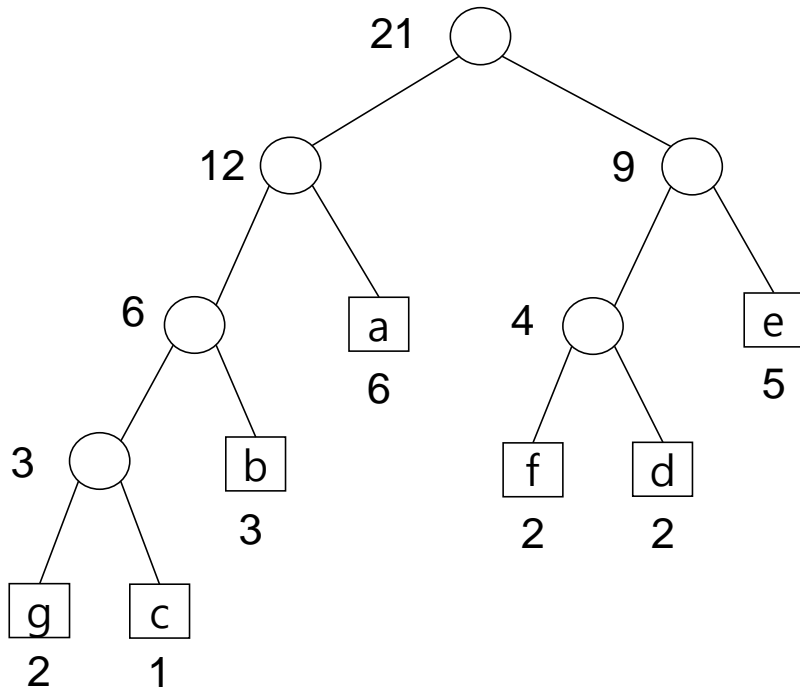
[5]



[6]



[7]

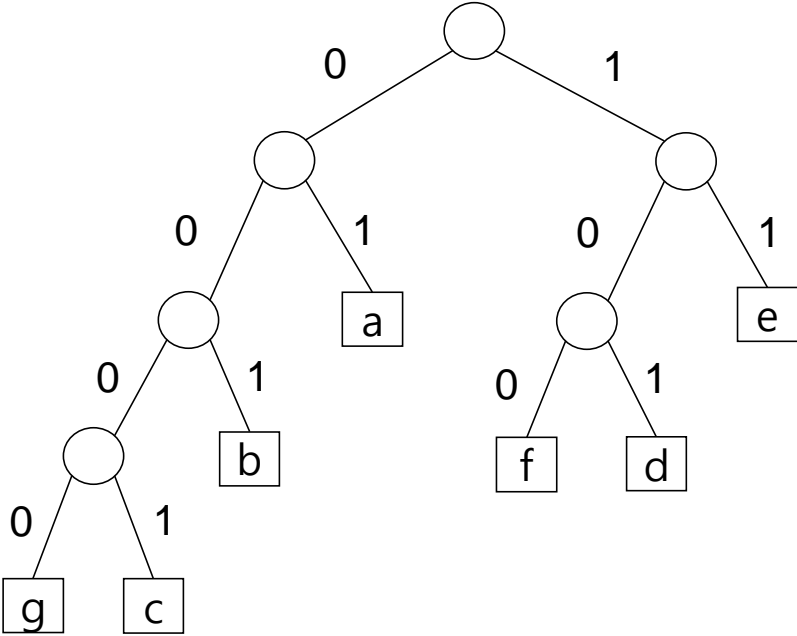


종료



왼쪽으로 이동 시 0 부여, 오른쪽으로 이동 시 1 부여

[8]



후프만코드





- 코드를 구축해 가는 매 단계에서 결정된 사항은 이후에는 다시 고려하지 않는다.

✓ 탐욕적 알고리즘

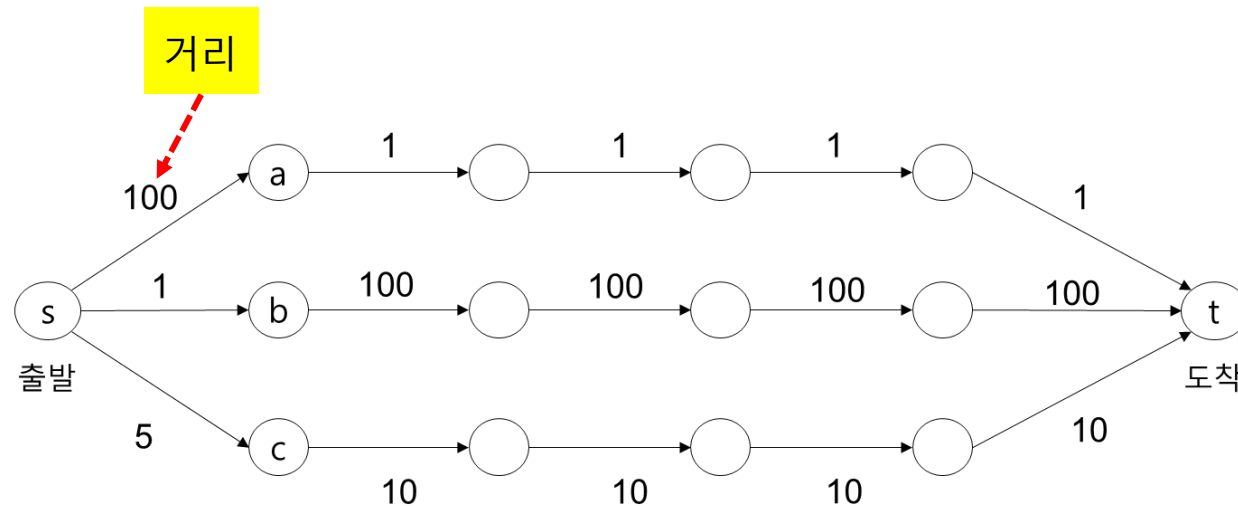


## 탐욕적 알고리즘이 최적해를 찾지 못하는 경우

문제: 출발지점에서 도착지점까지의 최단 경로 찾기

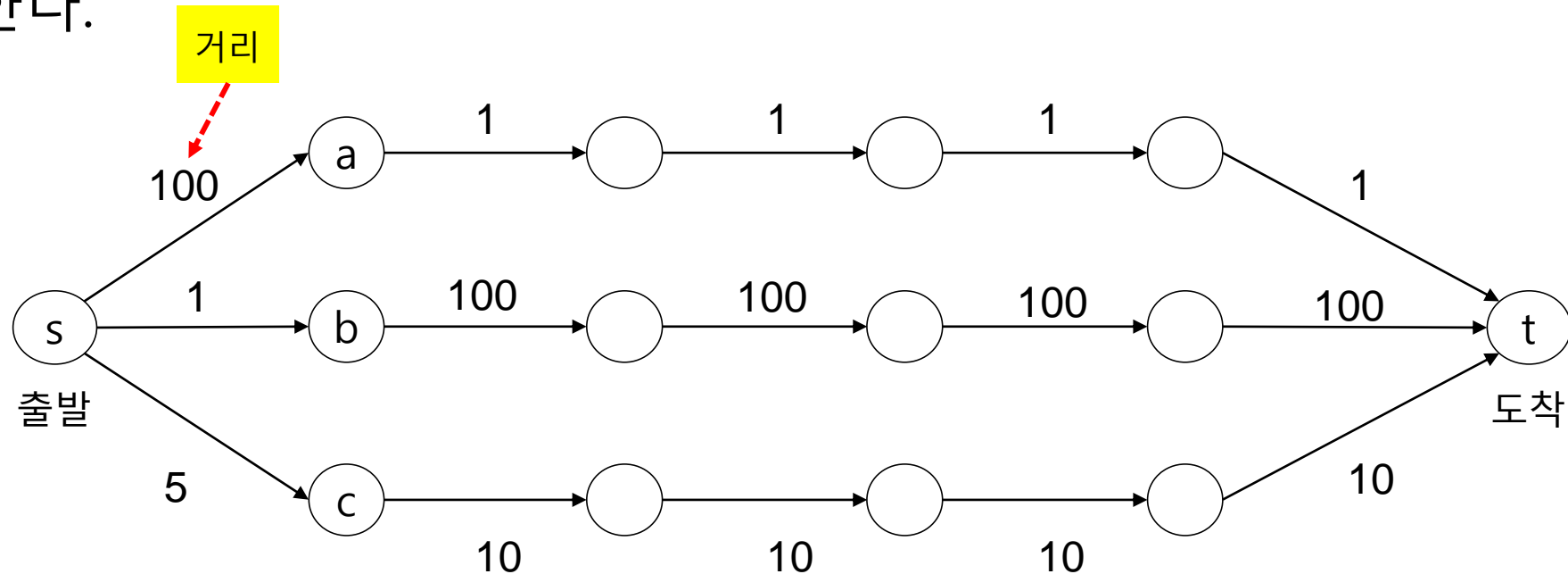
[최단 경로 알고리즘]

출발점에서 다음으로 직접 갈 수 있는 지점(노드) 중 가장 가까운 지점을 찾는다. 그 지점으로 이동한 후, 도착지점에 도달할 때 까지 같은 방법을 수행한다.



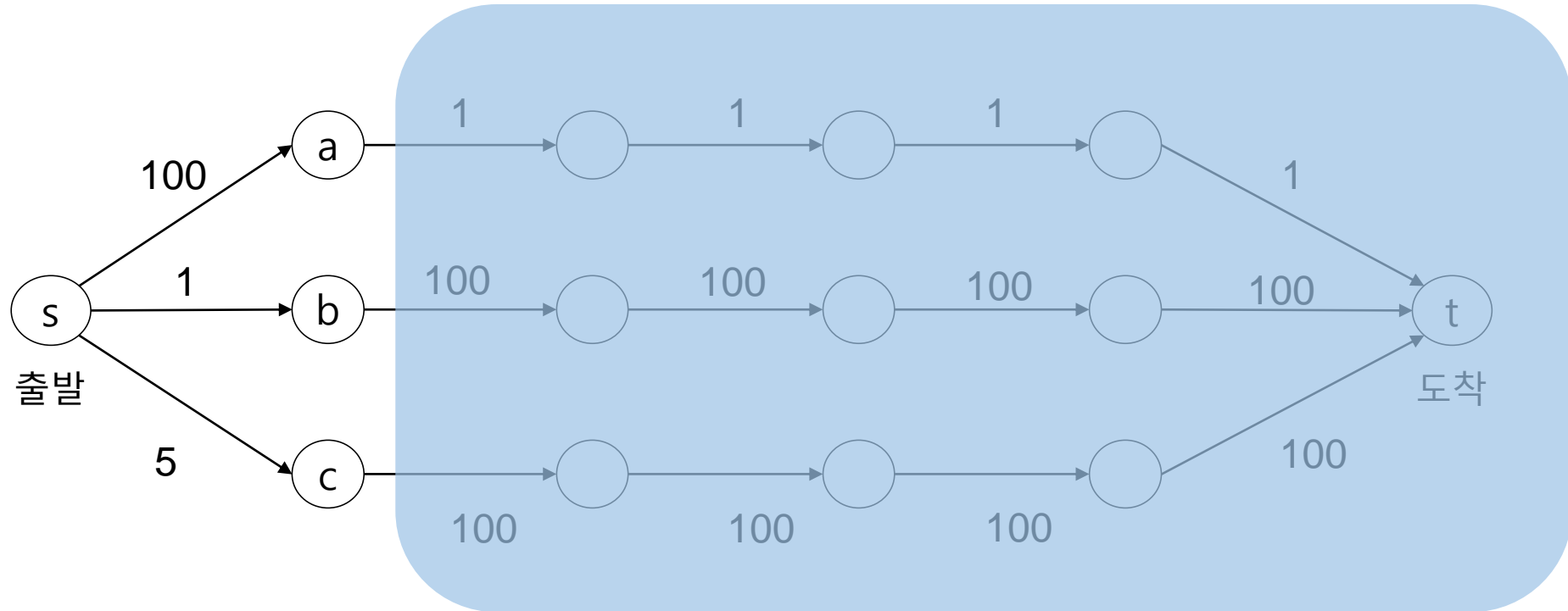
## [탐욕적 알고리즘]

출발점에서 다음으로 직접 갈 수 있는 지점(노드) 중 가장 가까운 지점을 찾는다. 그 지점으로 이동한 후, 도착지점에 도달할 때 까지 같은 방법을 수행한다.



## [탐욕적 알고리즘]

만일 출발지점에서 100, 1, 5 중 제일 가까운 b를 선택한다면, 최적해를 못 찾음



- 탐욕적 알고리즘은 해를 쉽게 구한다는 장점이 있다.
- 그러나, 모든 문제에 대해 탐욕적 알고리즘이 최적해를 찾는 것은 아니다.
- 다른 금액의 동전을 이용하여 최적해가 되지 않는 경우를 구성해 보시오.



## 12주차 강의 요약

- 정렬
- 분할정복 알고리즘
- 탐욕적 알고리즘



12주차 끝

마지막 강의입니다.

한 학기 수고하셨습니다.

