

*2주차 : Insertion & Merge sort*

# 알 고 리 즈

2016. 9. 8.

충남대학교 컴퓨터공학과 임베디드 시스템 연구실  
조교 김주엽

# Overview

- ▶ 알고리즘의 수행 시간

- 1) 시간 복잡도

- 2)  $O$ -,  $\Omega$ -, and  $\Theta$ -notation

- ▶ 두 가지 정렬 방법 소개 및 Time Complexity 계산

- 1) Insertion sort

- 2) Merge sort

- ▶ 실습 / 과제

- 파일 입출력을 사용한 Insertion & Merge sort 구현

# Time Complexity

## ▶ Time Complexity (시간 복잡도)

알고리즘을 구성하는 모든 명령어들에 대해서  
각각의 [ 수행에 필요한 Cost x 수행 횟수 ] 의 총합



# Notation

▶ **O-notation (최악의 경우) :  $f(n) = O(g(n))$**

모든  $n \geq n_0$ 에 대해  $0 \leq f(n) \leq cg(n)$ 인 양의 상수  $n$ ,  $c$ 이 존재할 때  
e.g.  $2n^2 = O(n^3)$  ( $c=1, n_0=2$ )

▶  **$\Omega$ -notation (최상의 경우) :  $f(n) = \Omega(g(n))$**

모든  $n \geq n_0$ 에 대해  $0 \leq cg(n) \leq f(n)$ 인 양의 상수  $n$ ,  $c$ 이 존재할 때  
e.g.  $\sqrt{n} = \Omega(\lg n)$  ( $c=1, n_0=16$ )

▶  **$\Theta$ -notation (평균인 경우) :  $f(n) = \Theta(g(n))$**

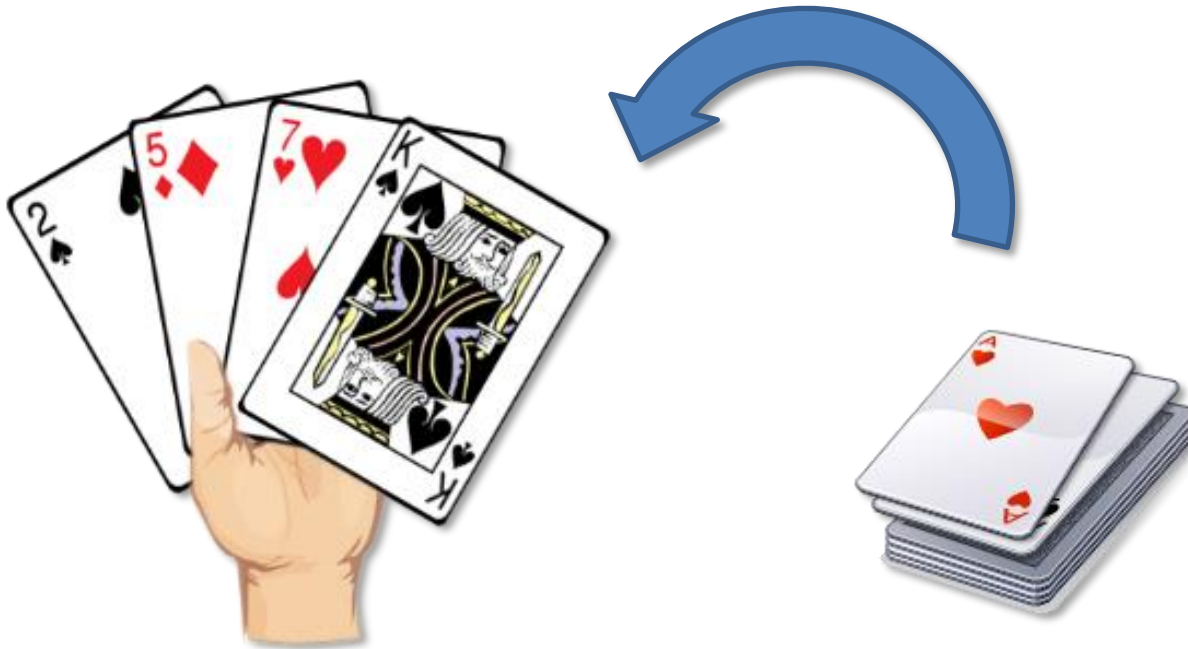
$\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$  일 때  
e.g.  $\frac{1}{2}n^2 - 2n = \Theta(n^2)$

※ 양의 상수  $n$ 과  $c$ , 계산 방법에 따라 여러 가지  $g(n)$ 을 구할 수 있다.  
단, 일반적으로 가장 근접한 값을 찾으려 한다.

# Insertion Sort

## ▶ Insertion Sort (삽입 정렬)

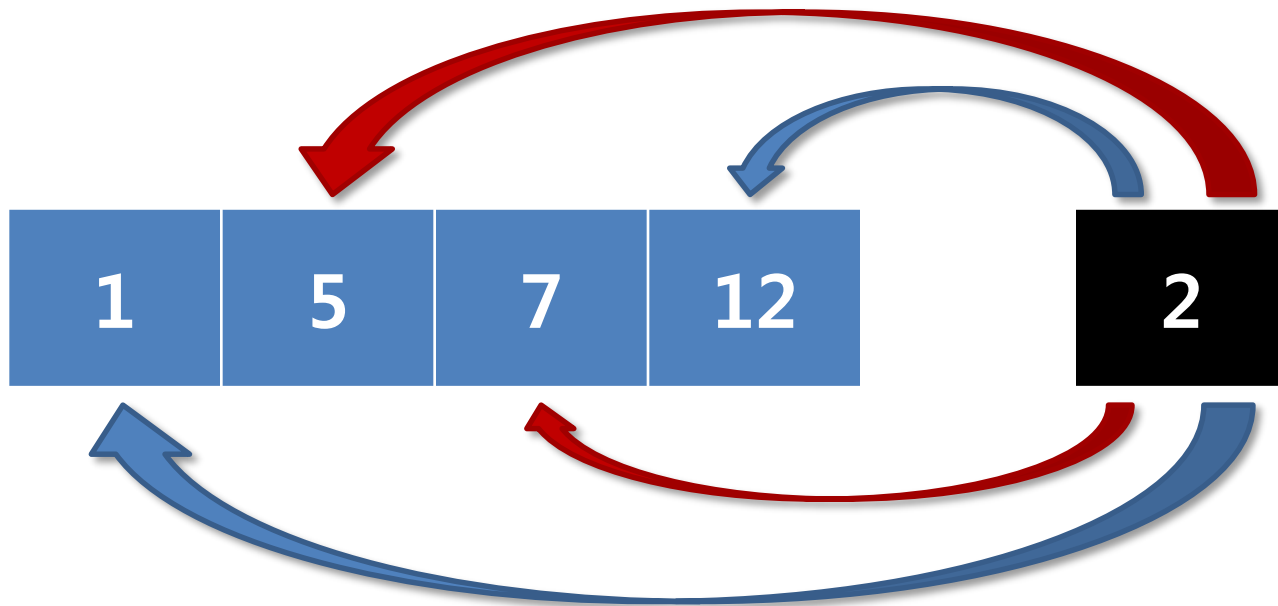
정렬되지 않은 배열로부터 **데이터를 하나씩 꺼내어**  
정렬되어 있는 배열의 알맞은 위치에 삽입하는 정렬 방법



# Insertion Sort

## ▶ 프로그램으로 구현 시 달라지는 점

알맞은 위치에 데이터를 삽입하기 위해  
배열에 저장된 값들을 하나씩 **순서대로 비교**해 보아야 함



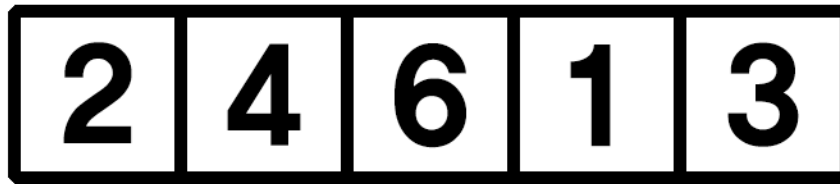
# Insertion Sort

## ▶ 정렬 방법 (1/7)

배열의 첫 번째 데이터를 정렬된 배열,  
나머지 데이터를 정렬되지 않은 배열로 나누어 생각한다



A[0]



A[1]

A[2]

A[3]

A[4]

A[5]

# Insertion Sort

## ▶ 정렬 방법 (2/7)

우측 배열의 첫 번째 데이터 값을 변수 **Key**에 복사한다



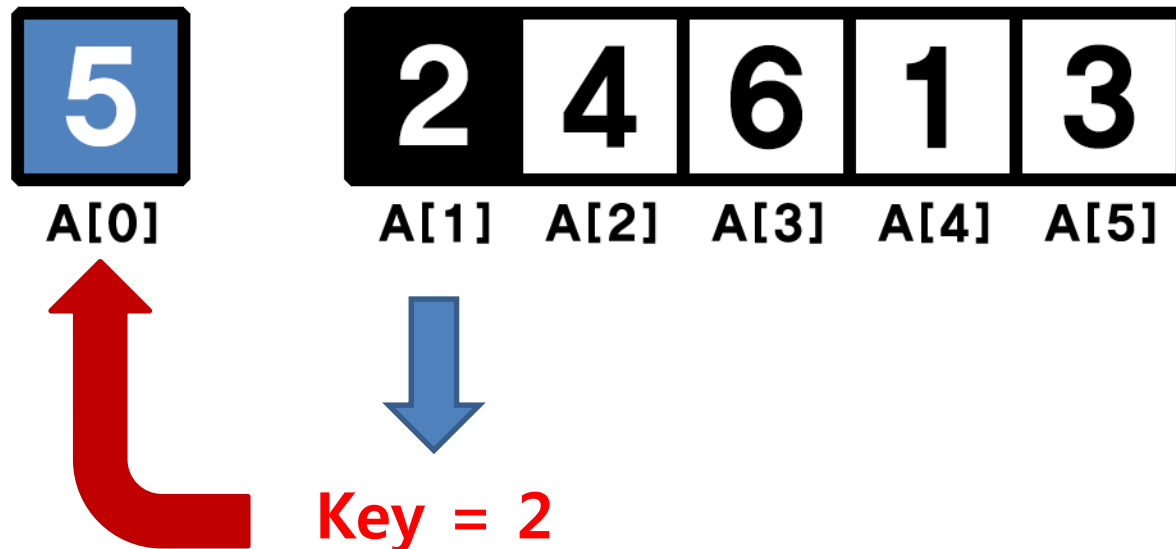
**Key = 2**



# Insertion Sort

## ▶ 정렬 방법 (3/7)

복사한 Key 값을 좌측 배열에 저장된 수 중  
가장 마지막(오른쪽) 배열에 저장된 값과 비교한다

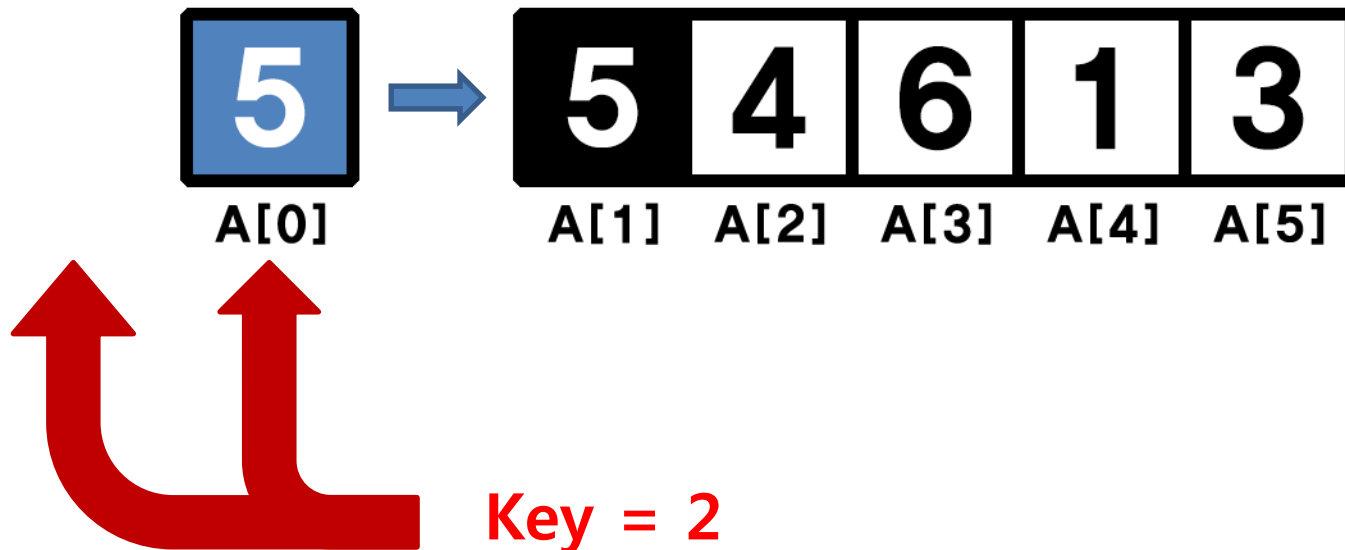


# Insertion Sort

## ▶ 정렬 방법 (4/7)

만약 비교한 값이 Key보다 크면

**해당 값을 오른쪽 인덱스에 복사**하고 그 왼쪽 값을 비교한다



# Insertion Sort

## ▶ 정렬 방법 (5/7)

더 비교할 값이 없거나, 비교한 값이 Key보다 작거나 같다면  
비교했던 위치의 바로 오른쪽 인덱스에 Key 값을 복사한다



# Insertion Sort

## ▶ 정렬 방법 (6/7)

1개 데이터에 대한 삽입 정렬이 완료되었다  
남은  $A[2] \sim A[5]$ 의 데이터에도 같은 작업을 반복한다



# Insertion Sort

## ▶ 정렬 방법 (7/7)

완료



# Insertion Sort

## ▶ pseudo-code (의사 코드)

“pseudocode” {

```
INSERTION-SORT ( $A, n$ )    ▷  $A[1 \dots n]$   
  for  $j \leftarrow 2$  to  $n$     ▷  $c_1 * n$   
    do  $key \leftarrow A[j]$     ▷  $c_2 * (n - 1)$   
       $i \leftarrow j - 1$     ▷  $c_3 * (n - 1)$   
        ▷  $c_4 * \sum_{j=2}^n t_j$  while  $i > 0$  and  $A[i] > key$   
          ▷  $c_5 * \sum_{j=2}^n (t_j - 1)$  do  $A[i+1] \leftarrow A[i]$   
            ▷  $c_6 * \sum_{j=2}^n (t_j - 1)$     $i \leftarrow i - 1$   
               $A[i+1] = key$     ▷  $c_7 * (n - 1)$ 
```

※ 반복문의 루프가 종료될 때, 한 번 더 검사를 수행하는 점에 유의한다.  
또한  $j = 2$  to  $n$  일 때의  $t_j$  값은, *best case* = 1, *worst case* =  $j$ 이다.

# Review : Sequence

## ▶ 등차 수열 공식

$$\{a_n\} : a_1, a_2, \dots, a_n \text{ (공차 : } d)$$

$$a_n = a_1 + (n - 1)d \quad (n = 1, 2, 3, \dots)$$

$$S_n = \frac{n(a_1 + a_n)}{2} = \frac{n\{2a_1 + (n-1)d\}}{2}$$

## ▶ 등비 수열 공식

$$\{a_n\} : a_1, a_2, \dots, a_n \text{ (공비 : } r)$$

$$a_n = a_1 r^{n-1} \quad (n = 1, 2, 3, \dots)$$

$$S_n = \frac{a_1(1-r^n)}{1-r} = \frac{a_1(r^n-1)}{r-1} \quad (r \neq 1) \qquad S_n = na \quad (r = 1)$$

# Merge Sort

## ▶ Merge Sort (합병 정렬)

[분할] - [정복] - [결합] 과정을 **재귀적으로 반복**하는 정렬 방법

### (1) 분할 (Divide) → mergeSort()

배열의 크기가 1이 될 때까지 계속하여 **배열을 둘로 나눈다**

### (2) 정복 (Conquer) → merge()

나뉘진 데이터를 **2개 배열씩 비교하여 재귀적으로 정렬**한다

### (3) 결합 (Combine) → merge()

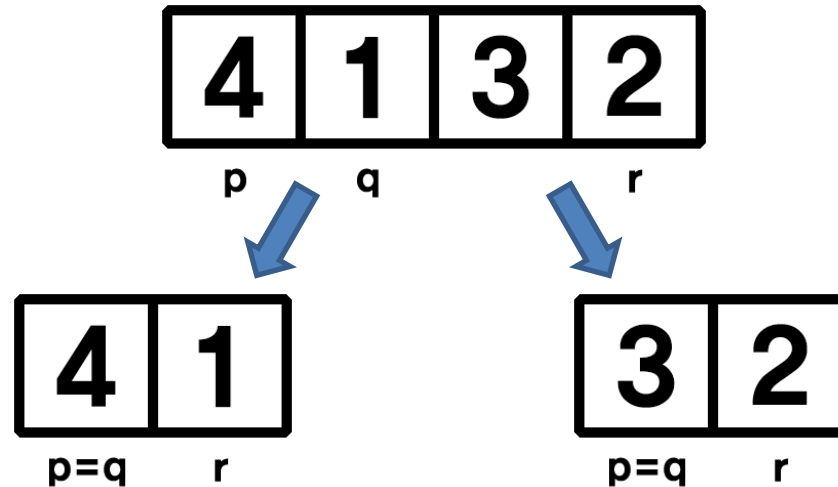
**정렬된 두 개의 배열을 병합**해 하나의 정렬된 배열로 만든다



# Merge Sort

## ▶ 정렬 방법 (1/4) – 분할 (1/2)

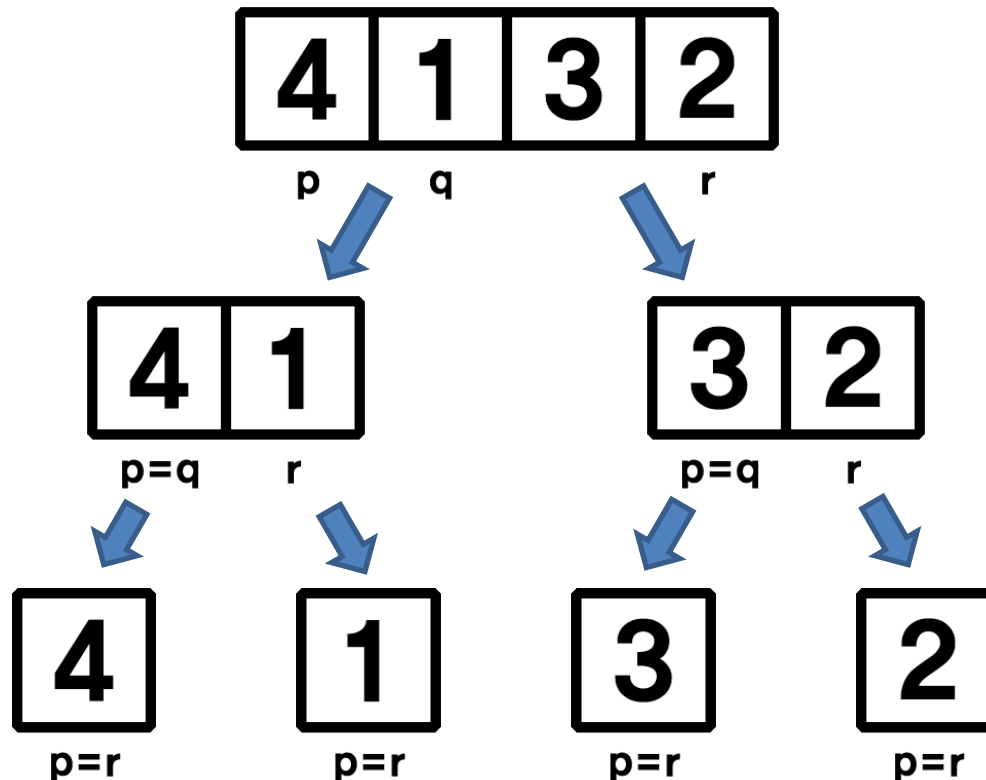
배열의 처음과 마지막 인덱스 번호를  $p$ ,  $r$ 이라 하고  
가운데 인덱스 번호를  $q$ 라 하여, 이를 기준으로 배열을 나눈다



# Merge Sort

## ▶ 정렬 방법 (2/4) – 분할 (2/2)

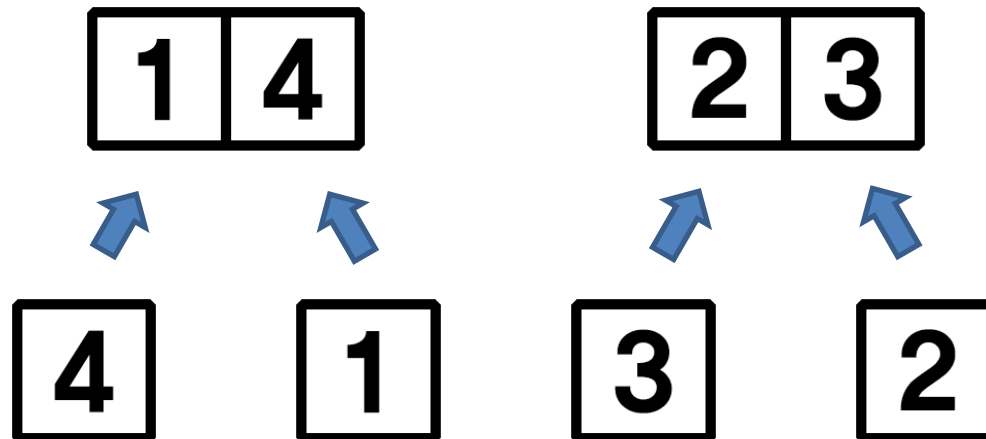
배열의 크기가 1이 될 때까지 계속하여 배열을 둘로 나눈다



# Merge Sort

## ▶ 정렬 방법 (3/4) – 정렬 & 결합 (1/2)

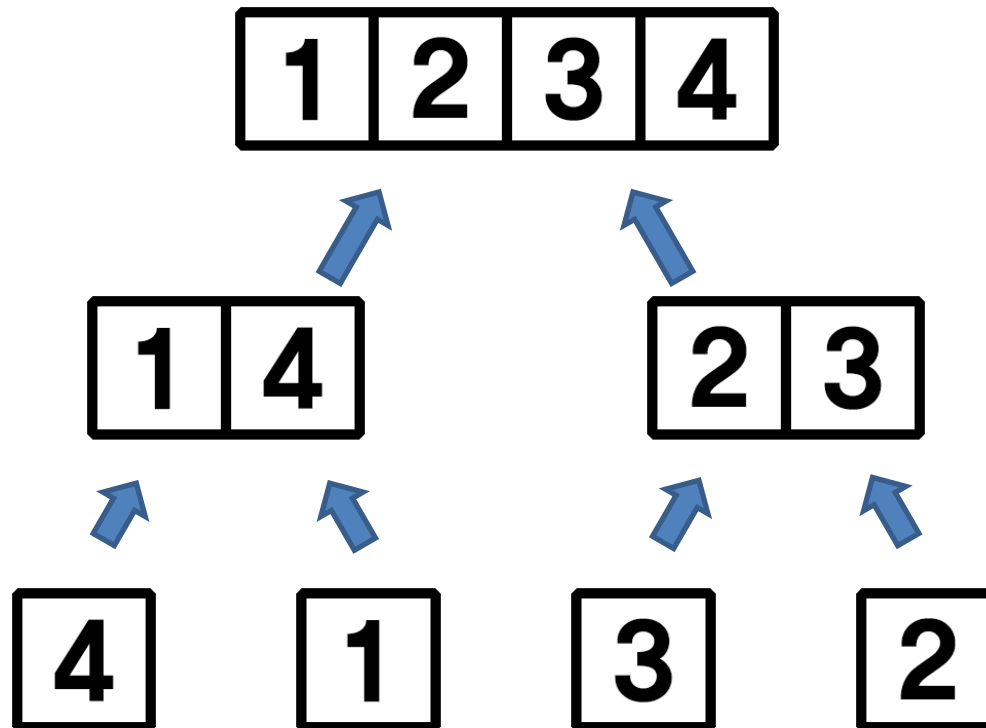
두 배열의 가장 앞 데이터를 비교하여  
더 작은 값부터 차례대로 뽑아내어 정렬 및 결합한다



# Merge Sort

## ▶ 정렬 방법 (4/4) – 정렬 & 결합 (2/2)

원래의 크기가 될 때까지 계속하여 정렬 및 결합하면 완료



# Merge Sort

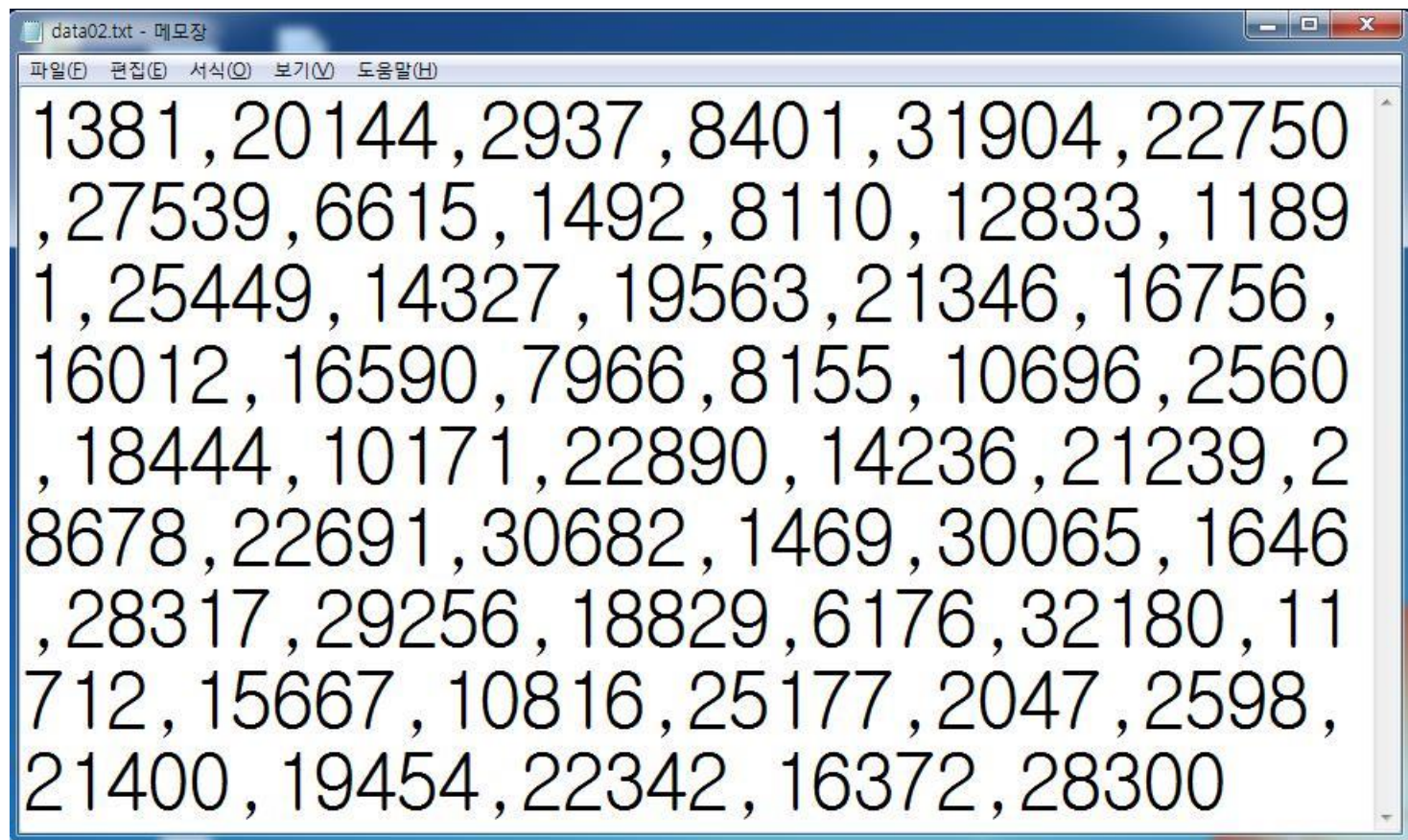
**MERGE-SORT**  $A[1 \dots n]$

1. If  $n = 1$ , done.
2. Recursively sort  $A[1 \dots \lceil n/2 \rceil]$  and  $A[\lceil n/2 \rceil + 1 \dots n]$ .
3. “*Merge*” the 2 sorted lists.

*Key subroutine:* **MERGE**

# Practice / Homework

► Input data : data02.txt



A screenshot of a Windows Notepad window titled "data02.txt - 메모장". The window contains a single line of 30 numbers separated by commas. The numbers are: 1381, 20144, 2937, 8401, 31904, 22750, 27539, 6615, 1492, 8110, 12833, 1189, 1, 25449, 14327, 19563, 21346, 16756, 16012, 16590, 7966, 8155, 10696, 2560, 18444, 10171, 22890, 14236, 21239, 2, 8678, 22691, 30682, 1469, 30065, 1646, 28317, 29256, 18829, 6176, 32180, 11, 712, 15667, 10816, 25177, 2047, 2598, 21400, 19454, 22342, 16372, 28300.

```
1381,20144,2937,8401,31904,22750  
,27539,6615,1492,8110,12833,1189  
1,25449,14327,19563,21346,16756,  
16012,16590,7966,8155,10696,2560  
,18444,10171,22890,14236,21239,2  
8678,22691,30682,1469,30065,1646  
,28317,29256,18829,6176,32180,11  
712,15667,10816,25177,2047,2598,  
21400,19454,22342,16372,28300
```

# Practice / Homework

**삽입 정렬을 구현**하고 data02.txt를 정렬하여 파일로 출력해 보라.  
구현이 어려운 경우, page-24의 프로그래밍 과정을 참고하라.

- a . 정렬 결과를 파일로 출력할 때,  
파일명은 **"hw02\_분반\_학번\_insertion.txt"**로 한다.
  
- b . 정렬 후 출력된 파일은,  
**data02\_insertion.txt와 완전히 같은 파일**이 되어야 한다.  
즉, 파일 쓰기를 할 때 동일한 양식으로 작성하여야 한다.

# Practice / Homework

**삽입 정렬을 구현**하고 data02.txt를 정렬하여 파일로 출력해 보라.  
이 때, 탐색 과정에서 binary search를 이용하여 구현하라. 구현된 알고리즘의 성능을 분석하라.

- a . 정렬 결과를 파일로 출력할 때,  
파일명은 “hw02\_분반\_학번\_binary\_insertion.txt”로 한다.
  
- b . 기존의 탐색 방법을 쓴 삽입 정렬과 성능을 비교하여  
보고서에 첨부한다.



# Practice / Homework

## ▶ 프로그램 구현 과정

#	구현 순서 (예시)
1	파일 입/출력 변수 및 그 외 필요한 변수 선언
2	파일로부터 읽어온 데이터를 저장할 배열 생성 (포인터 변수로 선언하고 동적 공간 할당을 사용)
3	파일을 텍스트 읽기 모드(rt)로 열기
4	파일이 정상적으로 열렸는지 확인 후 예외처리
5	파일을 끝까지 읽어, 각각의 값을 배열에 저장
6	<b>Sorting</b>
7	파일로 출력하기 위해, 새로운 파일을 텍스트 쓰기 모드(wt)로 열기
8	정렬 결과를 파일로 출력
9	파일 닫기 & 동적 공간 할당 해제

# Practice / Homework

**합병 정렬을 구현**하고 data02.txt를 정렬하여 파일로 출력해 보라.  
삽입 정렬과는 별도의 프로젝트를 만들어서 구현해야 한다.

- a . mergeSort( )와 merge( ) 두 개의 함수를 구현하고,  
merge( ) 함수가 사용된 횟수를 기록하여  
정렬 결과를 파일로 출력할 때 마지막에 추가로 기재한다.
- b . mergeSort( )는 **Recursive function**으로 구현되어야 한다.
- c . 출력 파일명은 **"hw02\_분반\_학번\_merge.txt"**로 한다.  
이는 **data02\_merge.txt와 완전히 같은 파일**이 되어야 한다.

# Practice / Homework

**합병 정렬을 구현**하고 data02.txt를 정렬하여 파일로 출력해 보라.  
이 때, sublist의 수를 3개로 늘려서 구현한 뒤, 기존의 합병정렬과의 성능을 비교하라.

- a . 출력 파일명은 “hw02\_분반\_학번\_3way\_merge.txt”로 한다.  
이는 data02\_merge.txt와 완전히 같은 파일이 되어야 한다.
- b. 기존의 합병 정렬과의 성능을 비교하여 보고서에 첨부한다.

# Practice / Homework

## ※ 그 외 실습 과제 수행 중 유의 사항 (C언어)

- a . 과제 제출은 e-mail로, 구현한 소스파일(.c)과 헤더파일(.h)만 zip으로 압축하여 보낼 것. (폴더째로 압축하지 않도록)
- b . 메일 제목과 zip 파일명은 항상 아래의 양식으로 작성.  
>> hw주차\_분반\_학번\_프로그램명
- c . 과제 평가는 별도의 input data를 사용함. (양식은 동일)
- d . 보고서에는 비교한 결과만 포함하면 된다.

# Practice / Homework

## ※ 그 외 실습 과제 수행 중 유의 사항 (JAVA)

- a . 과제 제출은 e-mail로, 구현한 소스파일(.java)만 zip으로 압축하여 보낼 것. (폴더째로 압축하지 않도록)
- b . 메일 제목과 zip 파일명은 항상 아래의 양식으로 작성.  
>> hw주차\_분반\_학번\_프로그램명
- c . 과제 평가는 별도의 input data를 사용함. (양식은 동일)
- d . 보고서에는 비교한 결과만 포함하면 된다.

# Practice / Homework

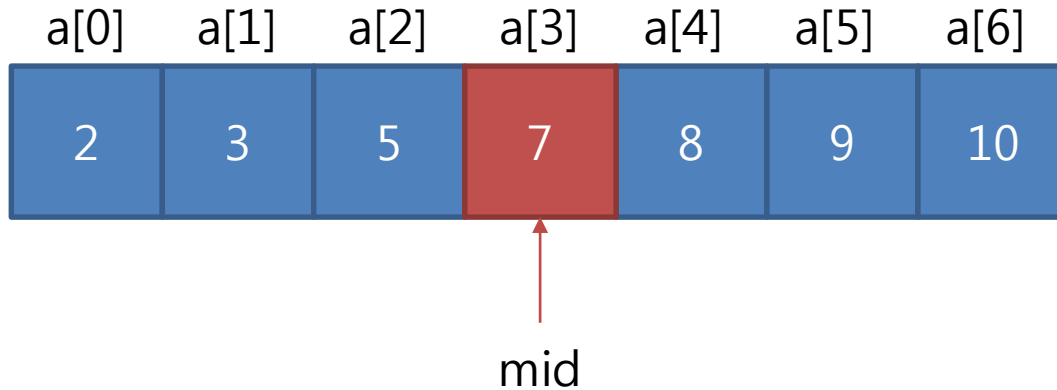
과제 제출 안내	
제출 방법	E-mail ( kimjy-128@cnu.ac.kr )
메일 제목	hw01_분반_학번_Algorithm
File 이름	hw01_분반_학번_insertion/ binary_insertion/merge/3way_merge.zip 보고서 파일
제출 기한	9월 22일 (목) <b>실습 수업 시간 전까지</b>
과제 평가 감점 사항	
제출 지연 (수업 시작부터)	- 50% / 1주
요구 사항 누락 / 결과값 불일치	- 10 ~ 20% / 1개
코드 Error	- 50 ~ 100%
과제 Copy	0점

# hint. binary search

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
2	3	5	7	8	9	10

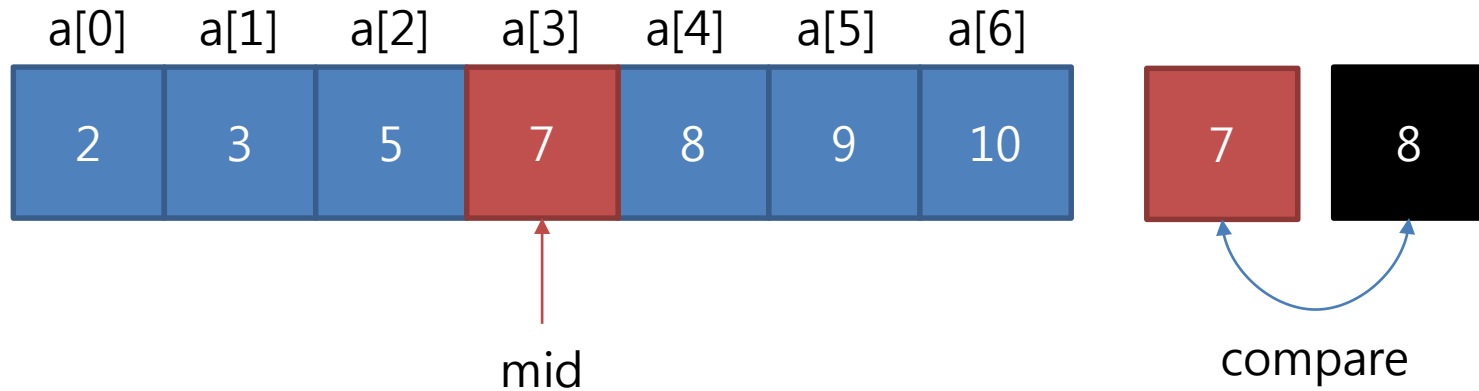
8

# hint. binary search

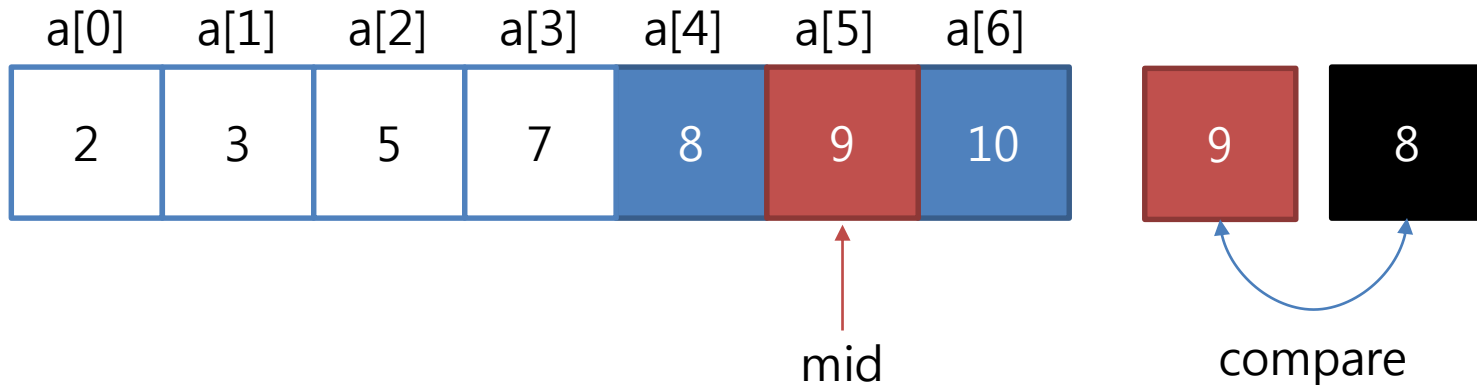




# hint. binary search



# hint. binary search



# hint. binary search

