
Algorithm

1주차: 정렬 알고리즘

1주차: 정렬 알고리즘

INDEX

1. 개요
2. 시간 복잡도
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
6. Divide and Conquer
7. Merge Sort
8. Quick Sort
9. 요약
10. 과제
11. References

1. 개요

정렬 알고리즘

Bubble Sort

Selection Sort

Recursive Bubble Sort

Insertion Sort

Recursive Insertion Sort

Merge Sort

Iterative Merge Sort

Quick Sort

Iterative Quick Sort

Heap Sort

Counting Sort

Radix Sort

Bucket Sort

ShellSort

TimSort

Comb Sort

Pigeonhole Sort

Cycle Sort

Cocktail Sort

Bitonic Sort

Pancake sorting

Strand Sort

Binary Insertion Sort

BogoSort or Permutation Sort

Gnome Sort

Sleep Sort – The King of Laziness / Sorting while Sleeping

Structure Sorting (By Multiple Rules) in C++

Stooge Sort

Tag Sort (To get both sorted and original)

Tree Sort

Cartesian Tree Sorting

Odd-Even Sort / Brick Sort

QuickSort on Singly Linked List

QuickSort on Doubly Linked List

3-Way QuickSort (Dutch National Flag)

Merge Sort for Linked Lists

Merge Sort for Doubly Linked List

3-way Merge Sort

1. 개요

왜 배워야 하는가?

1. 알고리즘 설계 / 디자인 하는 법을 배우
2. 실제 정렬을 해야 하는 데이터에 따라 유리한 정렬 알고리즘이 다름
3. 코딩테스트의 문제를 풀 때 가장 기초가 되는 것이 정렬임

2. 시간복잡도

정렬 알고리즘의 성능 측정

$O(1)$

```
printf("Hello");
```

$O(n)$

```
for (i=0; i<n; i++)  
{  
    printf("Hello");  
}
```

$O(n^2)$

```
for (i=0; i<n; i++)  
{  
    for (j=0; j<n; j++)  
    {  
        printf("Hello");  
    }  
}
```

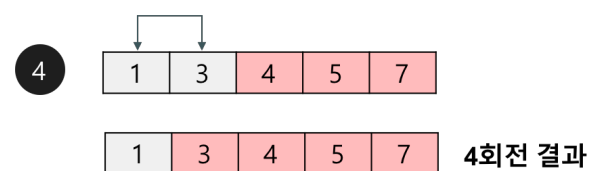
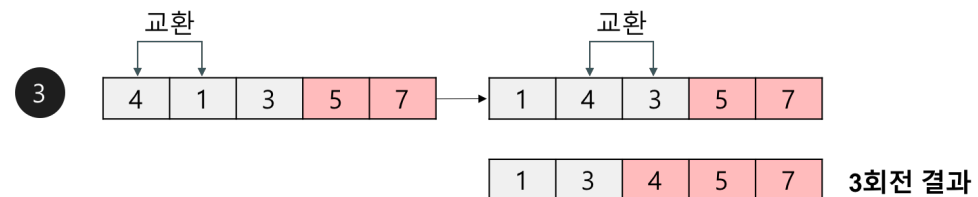
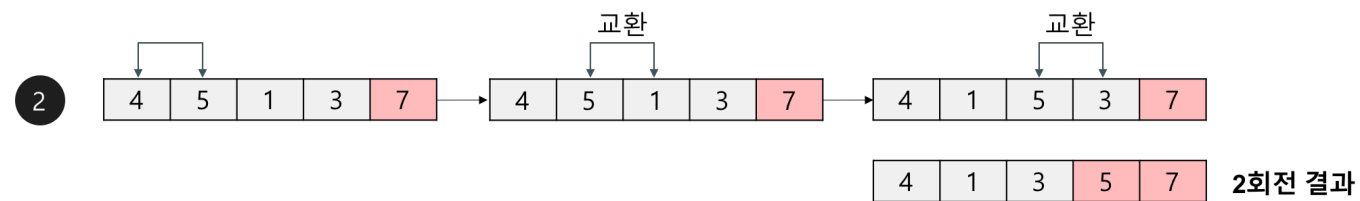
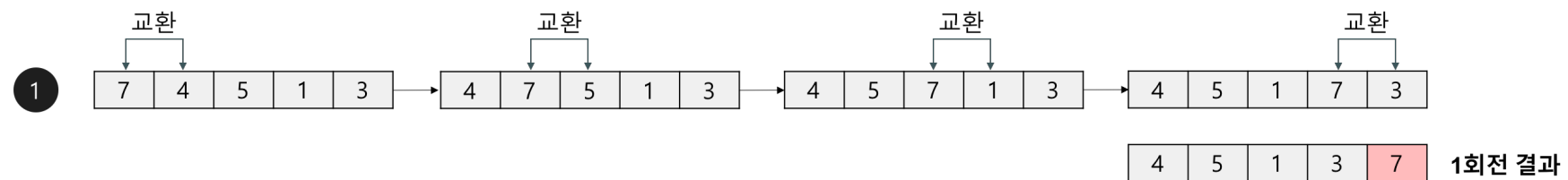
3. Bubble Sort

가장 간단한 정렬 방법

핵심 전략: Swap

초기상태

7	4	5	1	3
---	---	---	---	---



오름차순
완성상태

1	3	4	5	7
---	---	---	---	---

3. Bubble Sort

최적화한 Bubble Sort

최적화 전

```
for (i=0; i<n; i++)
{
    for (j=0; j<n-1; j++)
    {
        if (arr[j] > arr[j+1])
        {
            swap(&arr[j], &arr[j+1]);
        }
    }
}
```

최적화 후

```
for (i=0; i<n; i++)
{
    for (j=0; j<n-i; j++)
    {
        if (arr[j] > arr[j+1])
        {
            swap(&arr[j], &arr[j+1]);
        }
    }
}
```

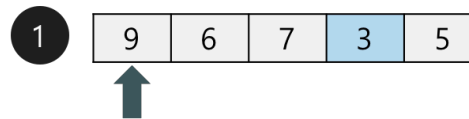
4. Selection Sort

최대 / 최소값을 Select

핵심 전략: Select

초기상태

9	6	7	3	5
---	---	---	---	---



최솟값 탐색: 3
첫 번째 값 9와 최솟값 3을 교환

3	6	7	9	5
---	---	---	---	---

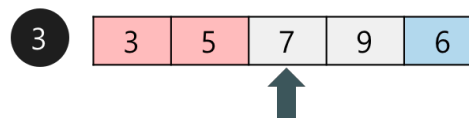
1회전 결과



최솟값 탐색: 5
두 번째 값 6과 최솟값 5를 교환

3	5	7	9	6
---	---	---	---	---

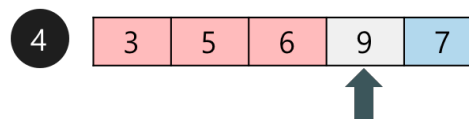
2회전 결과



최솟값 탐색: 6
세 번째 값 7과 최솟값 6을 교환

3	5	6	9	7
---	---	---	---	---

3회전 결과



최솟값 탐색: 7
네 번째 값 9와 최솟값 7을 교환

3	5	6	7	9
---	---	---	---	---

4회전 결과

오름차순
완성상태

9	6	7	3	5
---	---	---	---	---

5. Insertion Sort

자기 위치를 찾아서 Insert

핵심 전략: Insert

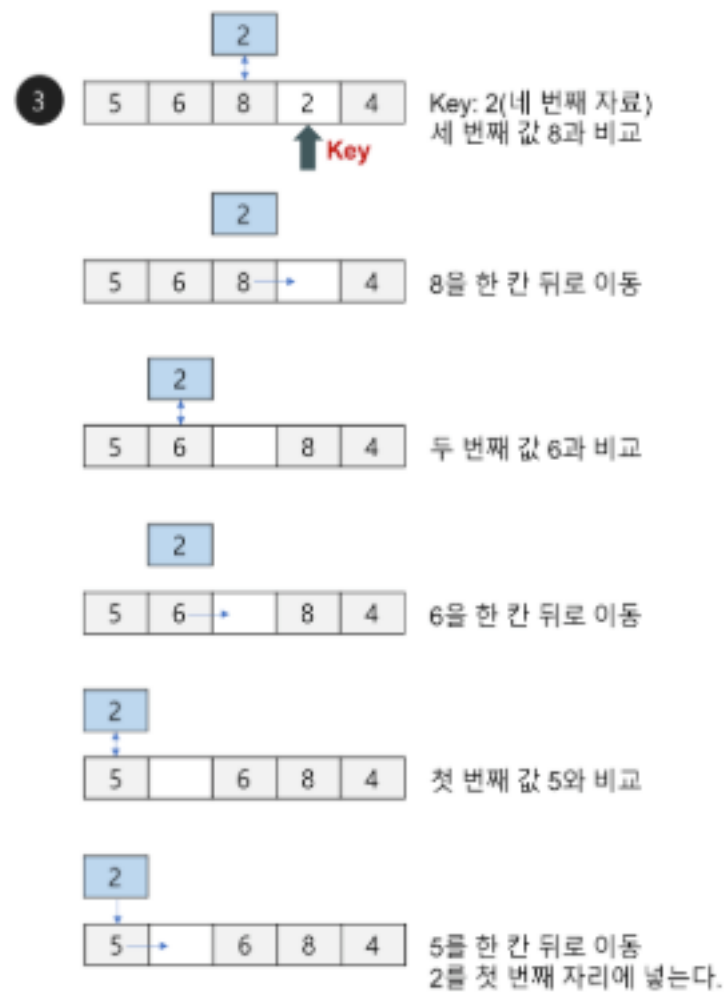
초기상태 8 5 6 2 4



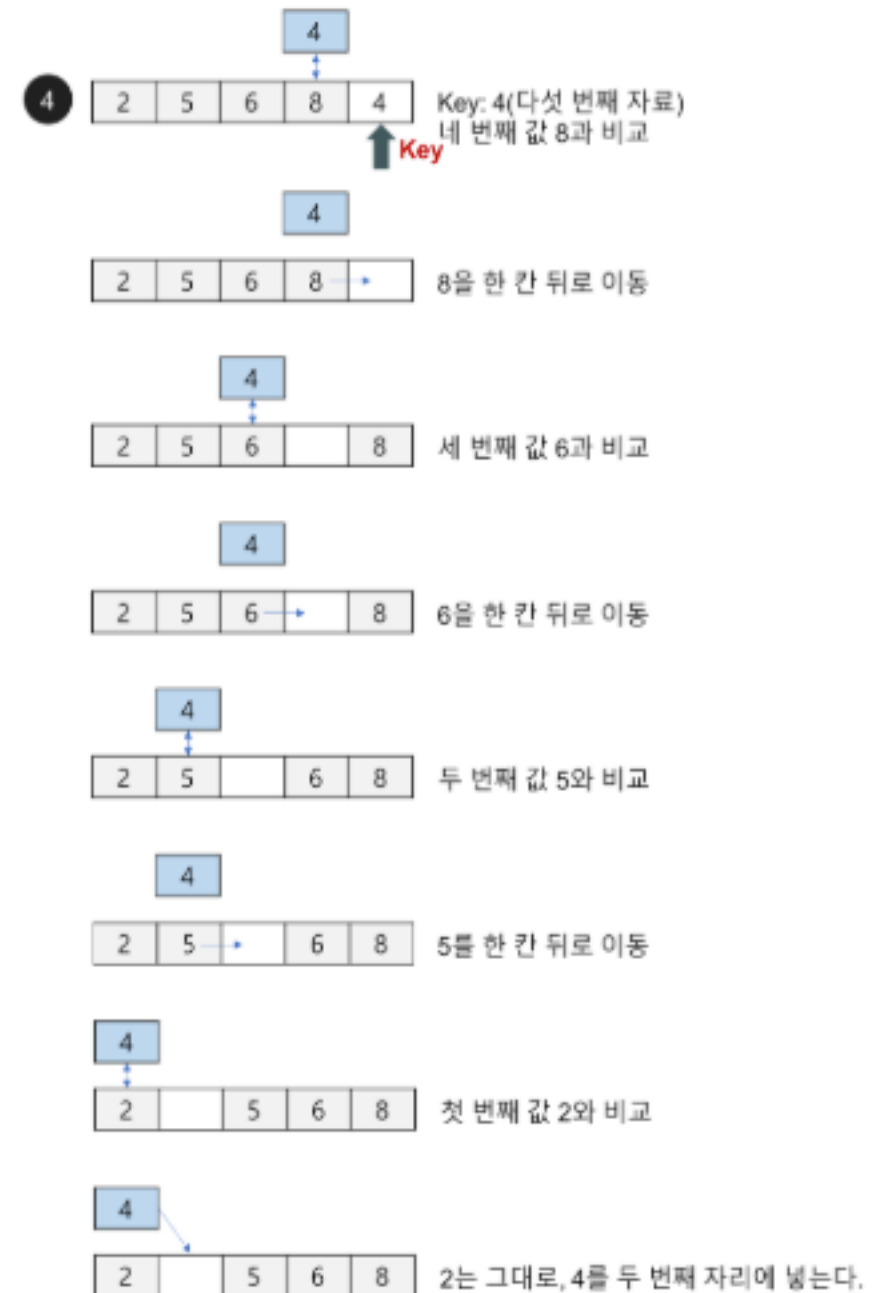
5 8 6 2 4 1회전 결과



5 6 8 2 4 2회전 결과



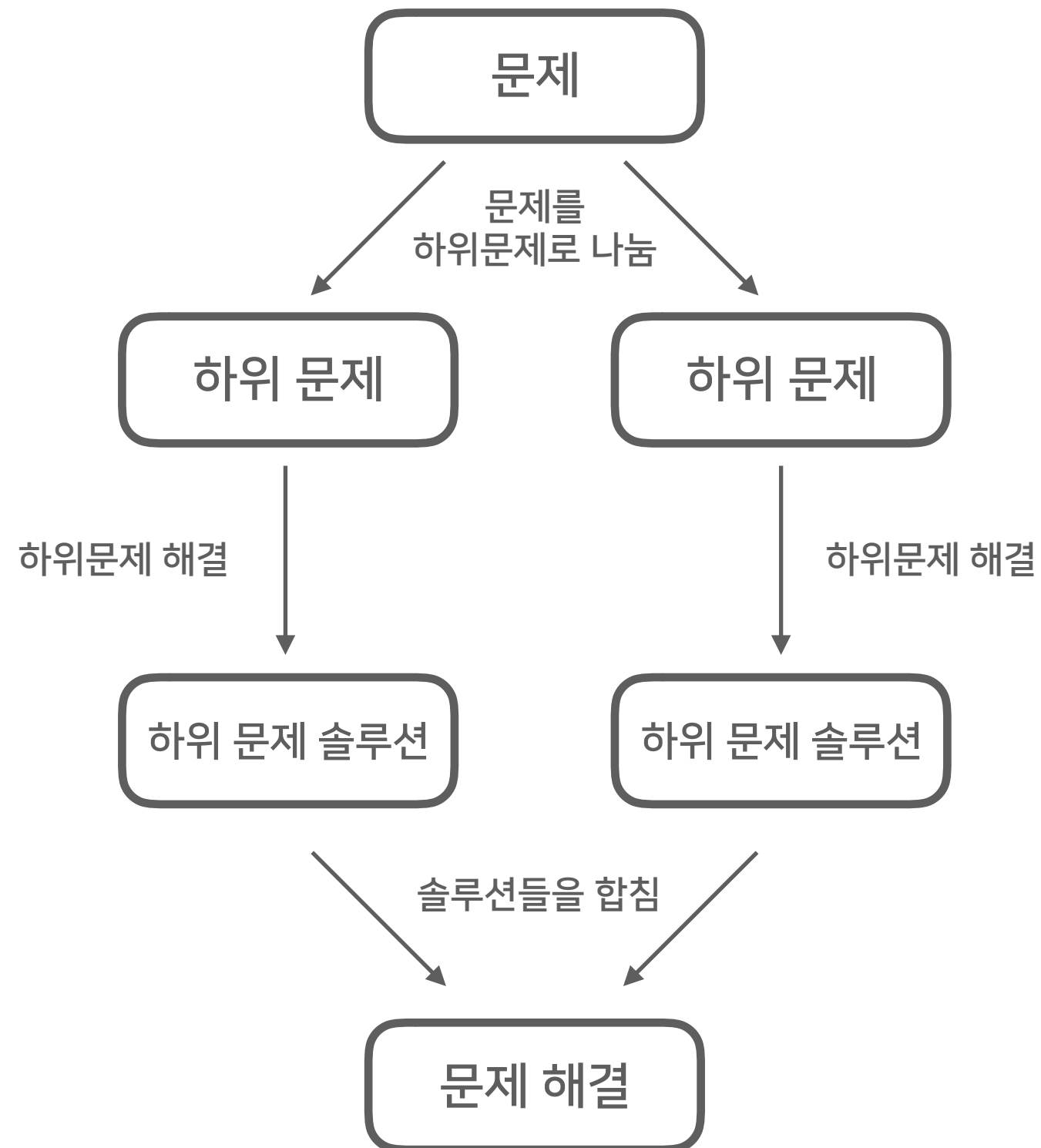
2 5 6 8 4 3회전 결과



2 4 5 6 8 4회전 결과

6. Divide and Conquer

재귀적으로 문제를 나눔



6. Divide and Conquer

문제: A의 B승 계산

정수 A와 B를 입력받고 A의 B승을 계산하여 출력 하시오.

입력 예시1:	출력 예시1:
2 11	2048

입력 예시2:	출력 예시2:
5 10	9765625

6. Divide and Conquer

해답: A의 B승 계산

```
for (i=0; i<B; i++)  
{  
    answer *= A;  
}
```

$O(n)$

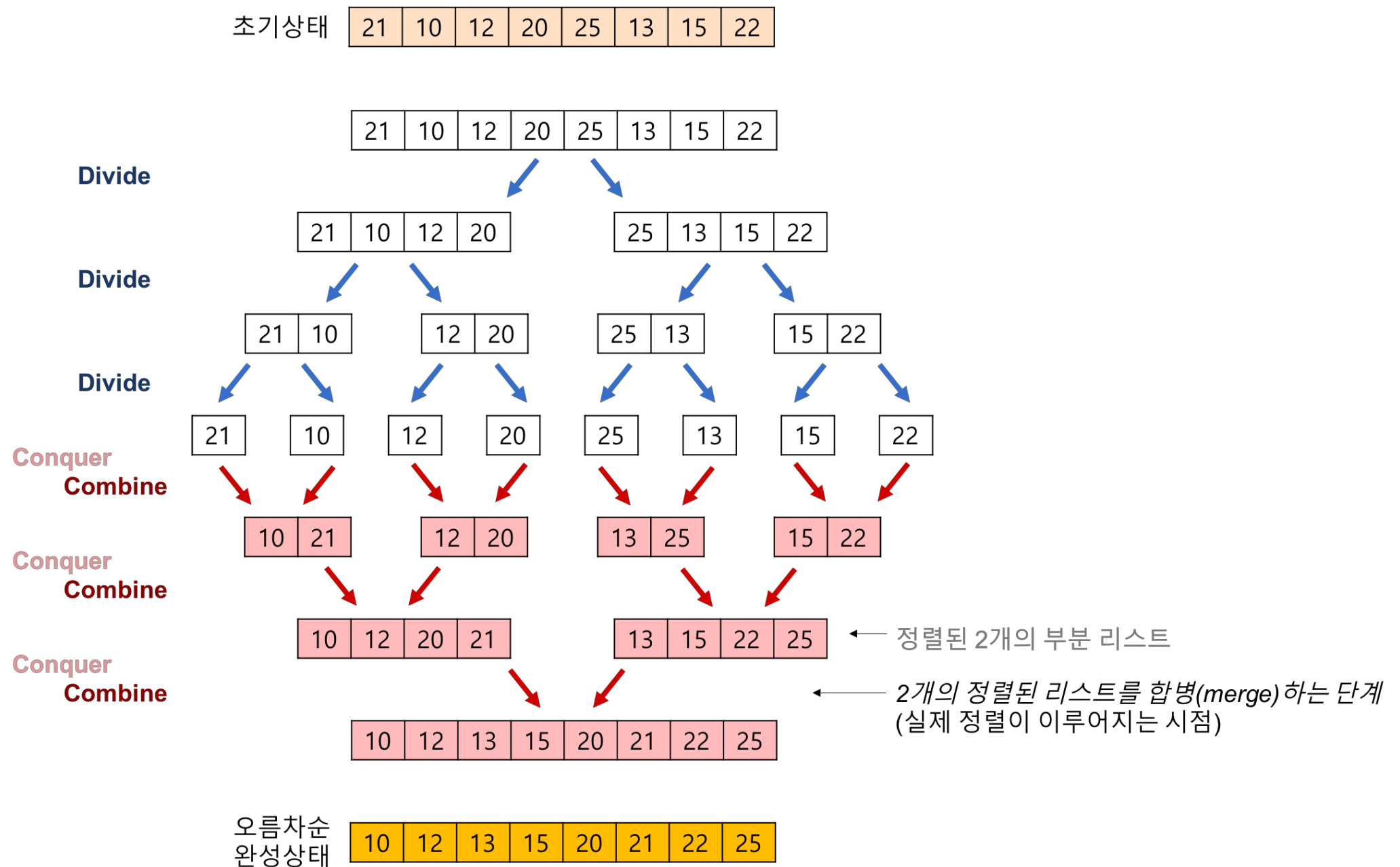
```
int power(int A, int B)  
{  
    int tmp;  
    if (B == 0) return 1;  
    tmp = power(A, B/2);  
    if (B % 2 == 0) return tmp * tmp;  
    else return A * tmp * tmp;  
}
```

$O(\log n)$

7. Merge Sort

잘게 나눴다가 합치면서 정렬

핵심 전략: Divide and Conquer



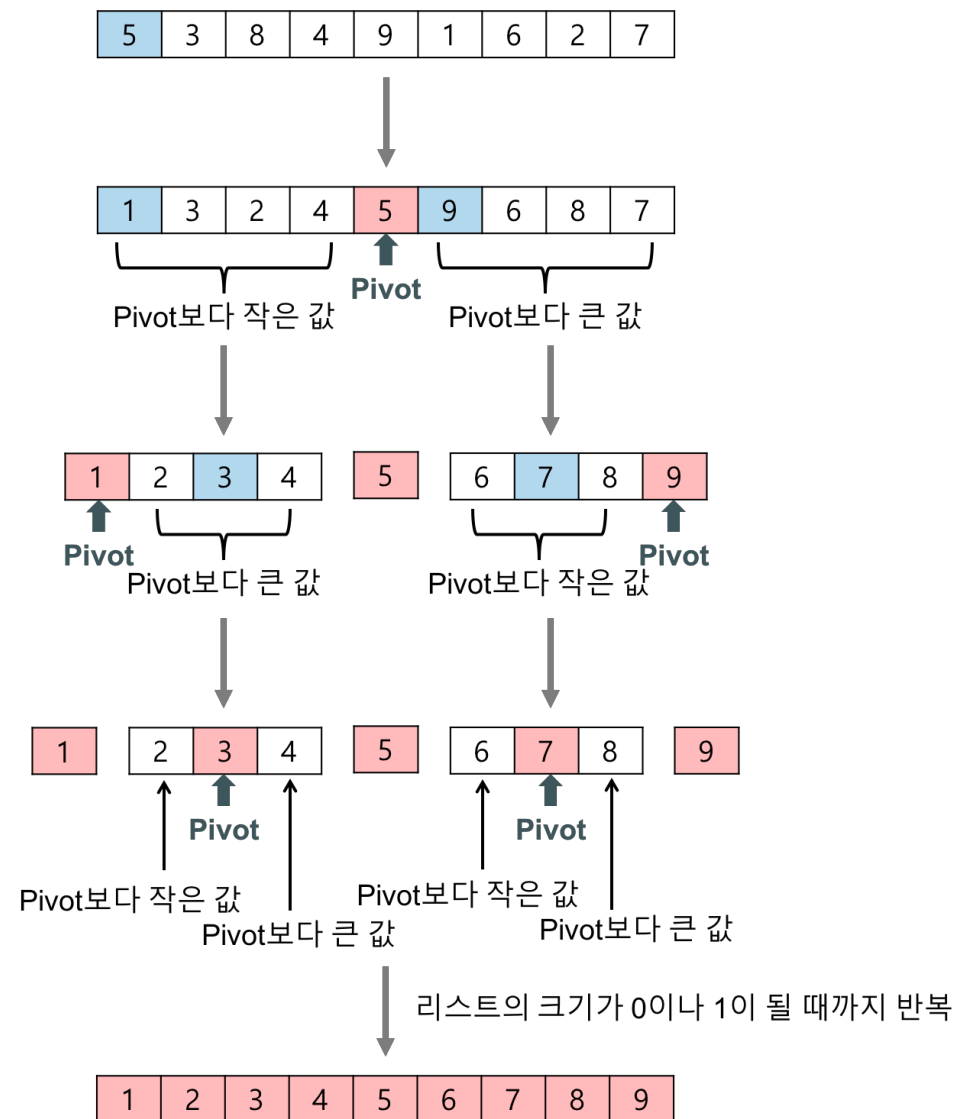
8. Quick Sort

Pivot 값을 기준으로 나눔

핵심 전략: Divide and Conquer

초기상태

5	3	8	4	9	1	6	2	7
---	---	---	---	---	---	---	---	---



오름차순
완성상태

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

최선, 최악, 평균 비교

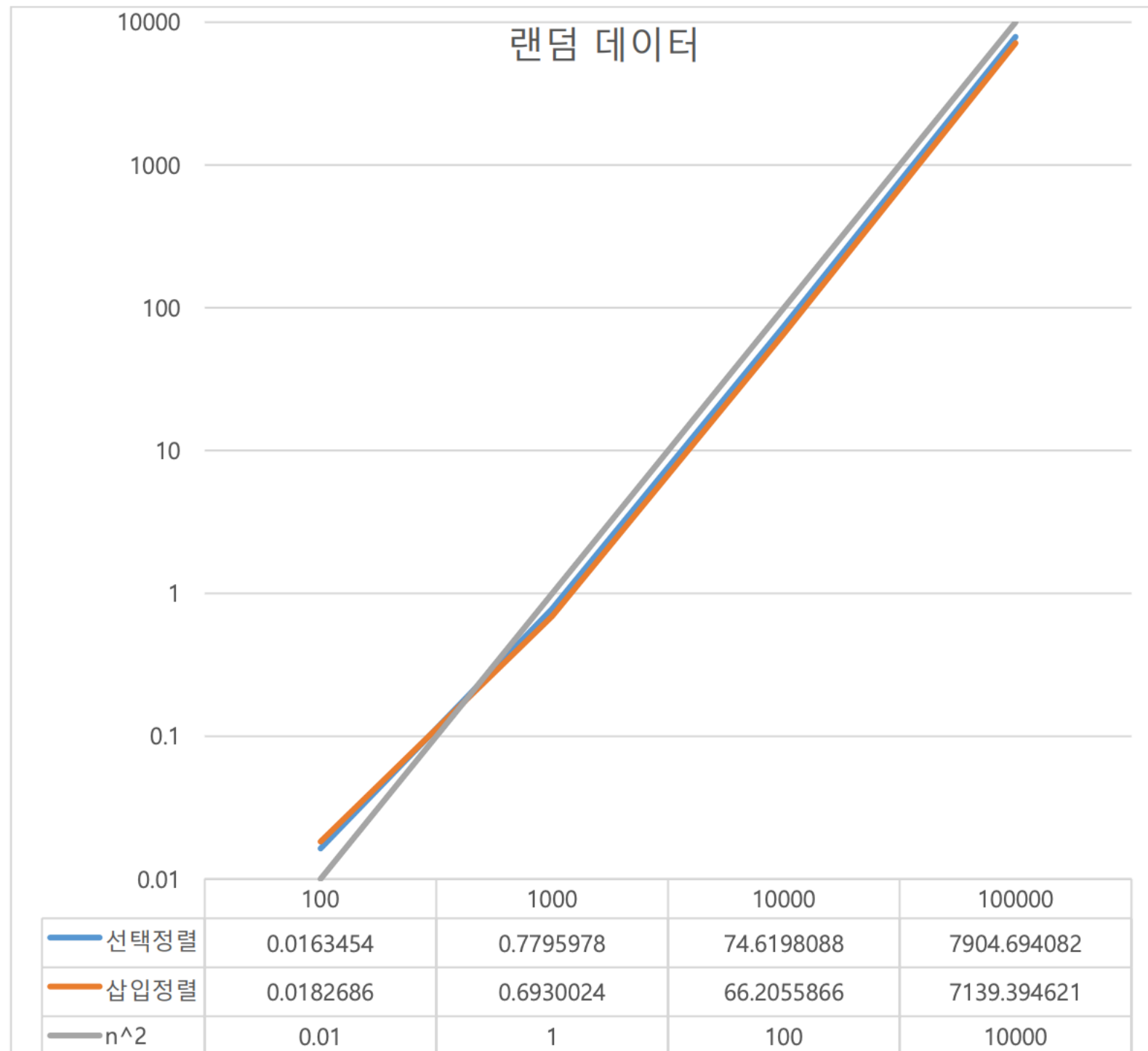
	Best	Worst	Average
Bubble Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Merge Sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Quick Sort	$O(n \log n)$	$O(n^2)$	$O(n \log n)$

정렬 알고리즘의 수행시간 비교

- 지금까지 배운 정렬 알고리즘을 직접 구현하고, 수행시간을 측정해서 비교한다.
- 데이터의 갯수를 증가시키면서 시간복잡도와 비례하는지 확인한다.
- 데이터는 랜덤 데이터, 정렬 데이터, 역정렬 데이터를 사용하여 최선, 평균, 최악을 비교한다.
- 구현한 알고리즘과 수행시간을 측정하는 코드를 Github에 Pull Request를 보내서 제출한다.
- 수행시간을 기록하고, 해당 정렬방법의 시간복잡도와 비교 / 분석한 보고서를 작성한다.
- 보고서는 이메일 (jamesj0107@naver.com)로 제출한다.
- 기한은 2019년 7월 14일 23시 59분 까지.
- 기한내 제출 못할 시 벌금있음.

10. 과제

과제 예시



11. References

<https://gmlwjd9405.github.io/2018/05/06/algorithm-bubble-sort.html>

<https://www.geeksforgeeks.org/analysis-of-different-sorting-techniques/>

<https://www.geeksforgeeks.org/divide-and-conquer/>