



# 《누워서 읽는 알고리즘》 정렬 알고리즘

20620 최재혁



## 1. 정렬 알고리즘 소개

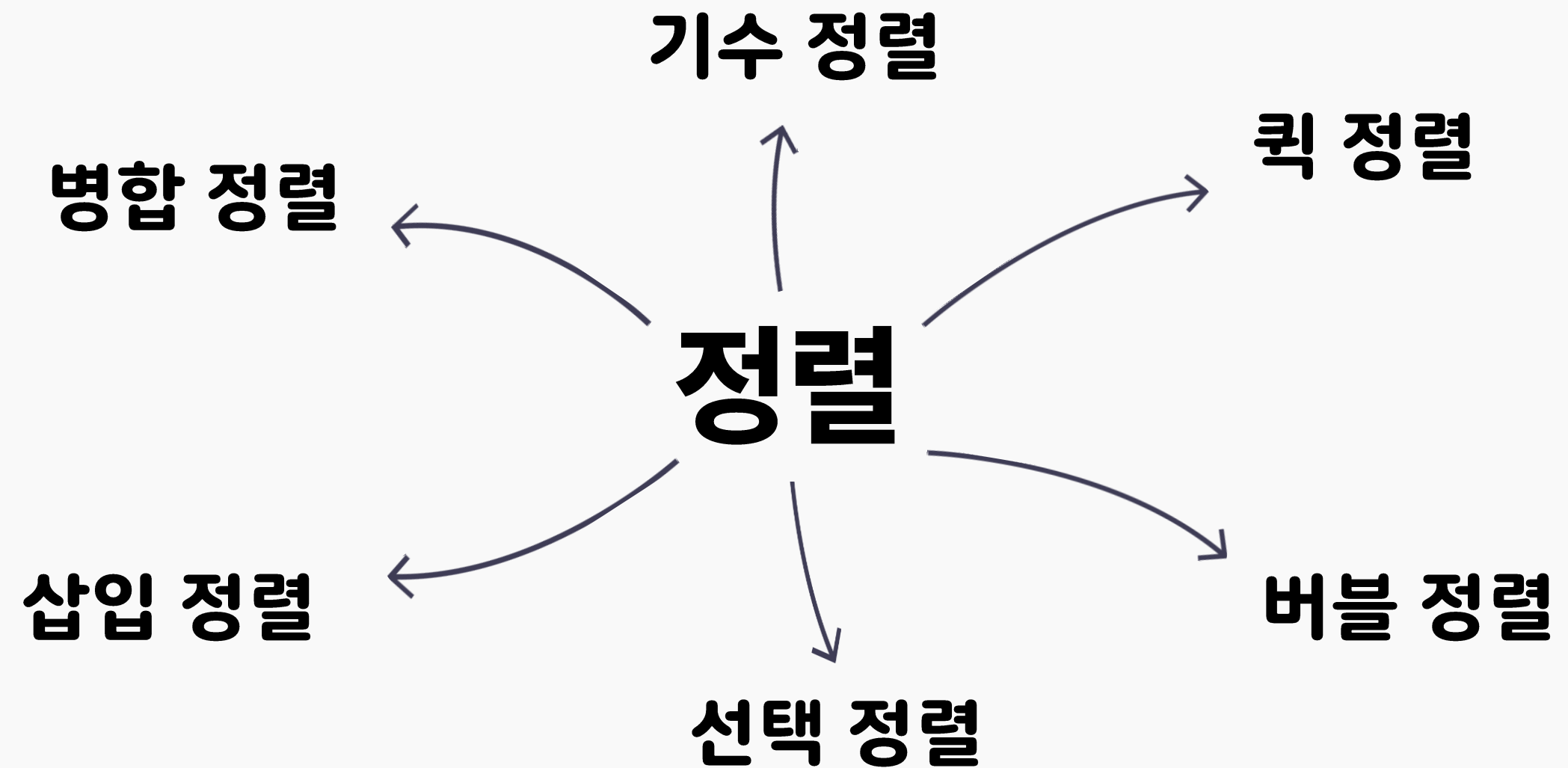
2. 퀵 정렬

3. 버블 정렬

4. 삽입 정렬

5. 93P 문제 해결하기

**정렬** :  $n$ 개의 숫자가 입력으로 주어졌을 때, 사용자의 기준에 맞게 출력하는 알고리즘



## 1. 정렬 알고리즘 소개

### 2. 퀵 정렬


















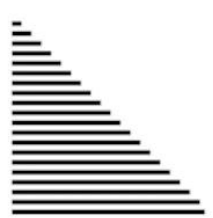



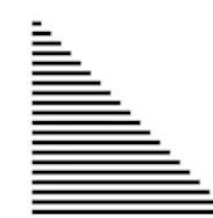




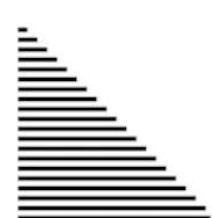
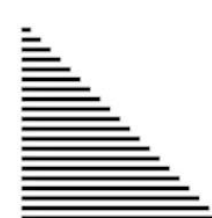

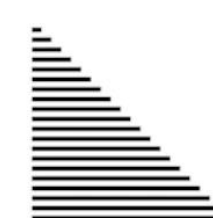










### 3. 버블 정렬

### 4. 삽입 정렬

### 5. 93P 문제 해결하기

**정렬** :  $n$ 개의 숫자가 입력으로 주어졌을 때, 사용자의 기준에 맞게 출력하는 알고리즘

<https://www.toptal.com/developers/sorting-algorithms>

 Play All	 Insertion	 Selection	 Bubble	 Shell	 Merge	 Heap	 Quick
 Random							
 Nearly Sorted							
 Reversed							
 Few Unique							

1. 정렬 알고리즘 소개

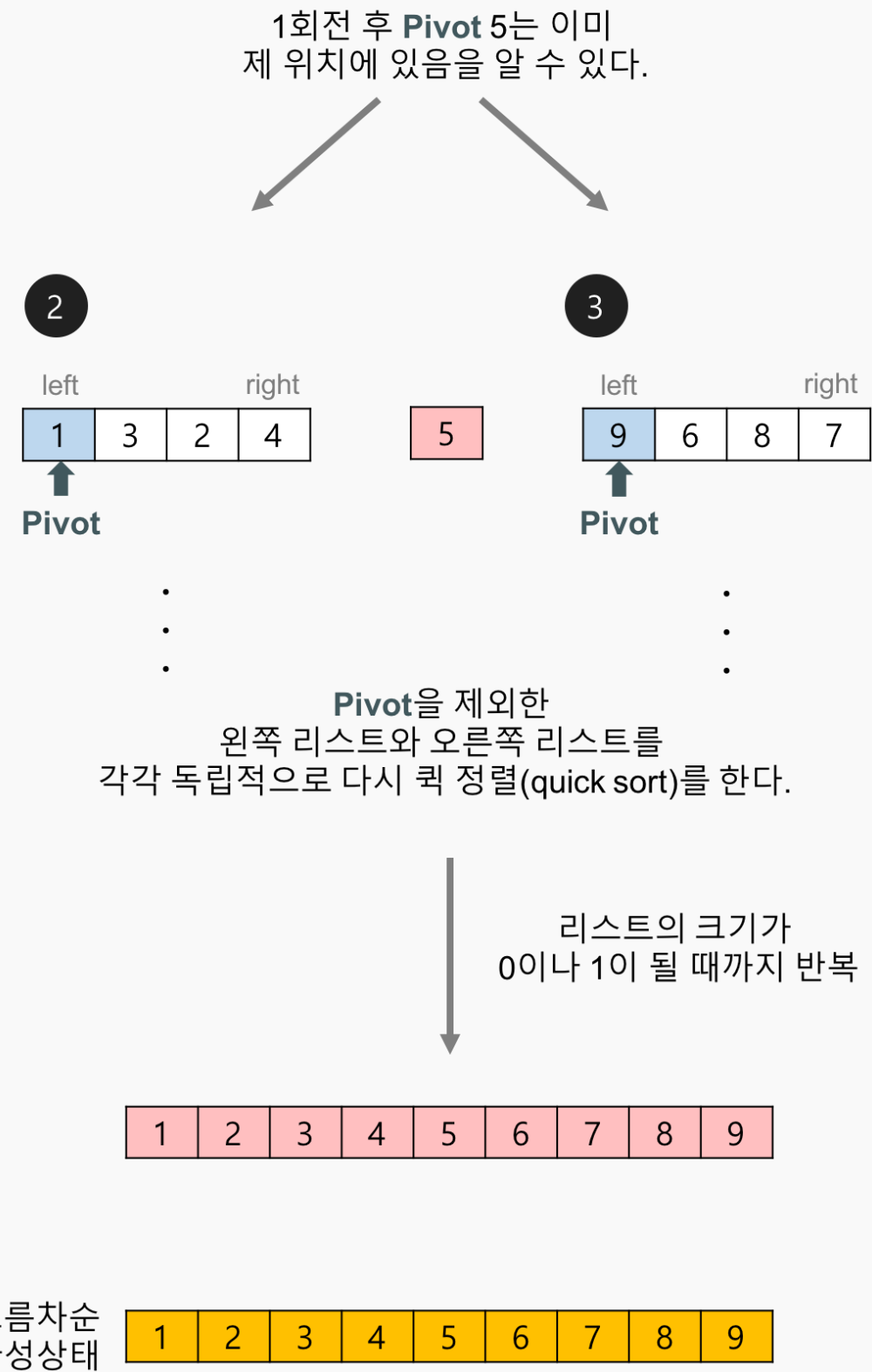
2. 퀵 정렬

3. 버블 정렬

4. 삽입 정렬

5. 93P 문제 해결하기

**퀵 정렬** : 하나의 pivot을 기준으로 두 개의 비균등한 크기로 분할하고 분할된 부분 리스트를 정렬한 다음 정렬된 리스트를 합쳐 전체 정렬하는 방법



## 1. 정렬 알고리즘 소개

### 2. 퀵 정렬

### 3. 버블 정렬

### 4. 삽입 정렬

### 5. 93P 문제 해결하기



Quick



**퀵 정렬** : 하나의 pivot을 기준으로 두 개의 비균등한 크기로 분할하고  
분할된 부분 리스트를 정렬한 다음 정렬된 리스트를 합쳐 전체 정렬하는 방법

#### < 장점 >

- 속도가 매우 빠름
- 추가 메모리 공간을 필요로 하지 않음

#### < 단점 >

- 정렬된 리스트에 대해서는 불균형 분할에 대해  
수행시간이 더 많이 걸림 (pivot 값을 잘못 설정할 경우)

## 1. 정렬 알고리즘 소개

### 2. 퀵 정렬

### 3. 버블 정렬

### 4. 삽입 정렬

### 5. 93P 문제 해결하기

**퀵 정렬** : 하나의 pivot을 기준으로  
두 개의 비균등한 크기로 분할하고  
분할된 부분 리스트를 정렬한 다음  
정렬된 리스트를 합쳐 전체 정렬하는 방법

```
#include <stdio.h>
#include <string.h>
```

```
void quickSort(int arr[], int L, int R) {
    int left = L, right = R;
    int pivot = arr[(L + R) / 2];
    int temp;
    do
    {
        while (arr[left] < pivot){
            left++;
        }

        while (arr[right] > pivot){
            right--;
        }

        if (left <= right) {
            temp = arr[left];
            arr[left] = arr[right];
            arr[right] = temp;

            left++;
            right--;
        }
    } while (left <= right);

    if (L < right)
        quickSort(arr, L, right);

    if (left < R)
        quickSort(arr, left, R);
}
```

→ Pivot을 중앙값으로 설정

→ Pivot보다 Left가 큰 값을 만날 때까지 반복

→ Pivot보다 Right가 작은 값을 만날 때까지 반복

→ Right가 더 크다면, Right가 Left보다 작은 값이 있으므로, 값을 교환해줌

→ Left가 Right보다 오른쪽에 있을 때까지 반복

→ 왼쪽과 오른쪽 배열을 재귀적으로 반복

1. 정렬 알고리즘

2. 퀵 정렬

3. 버블 정렬

4. 삽입 정렬

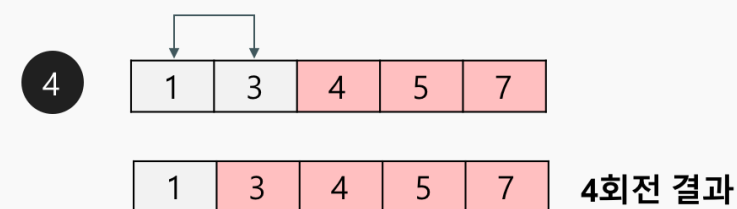
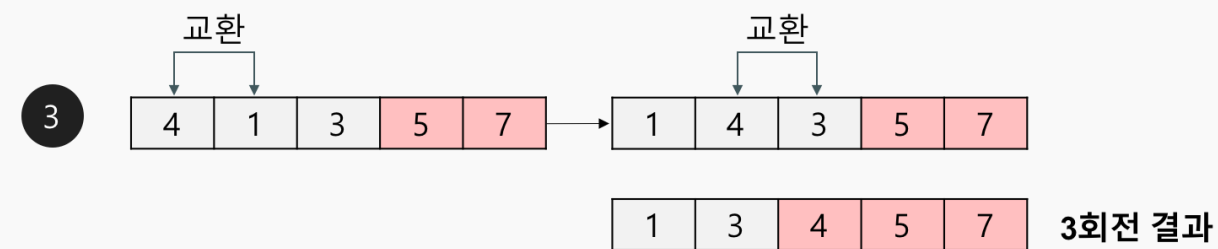
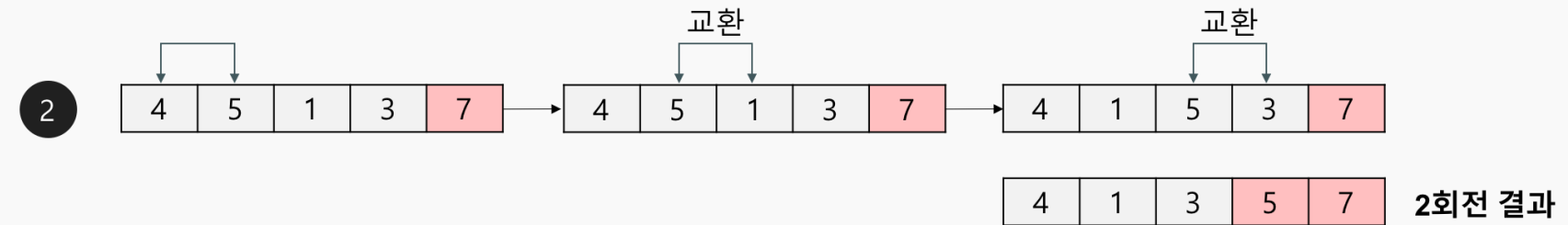
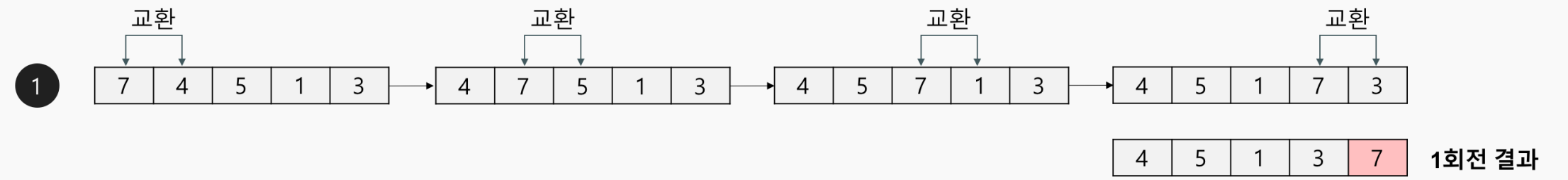
5. 93P 문제 해결하기

## 버블 정렬 : 서로 인접한 두 원소를 검사하여 정렬하는 알고리즘



초기상태 

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



오름차순  
완성상태 

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

1. 정렬 알고리즘

2. 퀵 정렬

3. 버블 정렬

4. 삽입 정렬

5. 93P 문제 해결하기



Bubble



**버블 정렬** : 서로 인접한 두 원소를 검사하여 정렬하는 알고리즘

**< 장점 >**

- 구현이 매우 간단함

**< 단점 >**

- 특정 요소가 이미 위치에 있더라도 교환되는 일이 발생
- 순서에 맞지 않은 요소를 인접한 요소와 교환함
- 가장 왼쪽에서 오른쪽으로 이동할 때는 모든 요소와 교환함



1. 정렬 알고리즘

2. 퀵 정렬

3. 버블 정렬

4. 삽입 정렬

5. 93P 문제 해결하기

```
#include <stdio.h>

int main() {
    for(int i = 0; i<5; i++){
        for(int j = 0; j<5-i; j++){
            if(a[j] > a[j+1]){
                int tmp = a[j];
                a[j] = a[j+1];
                a[j+1] = tmp;
            }
        }
    }
}
```

**버블 정렬** : 서로 인접한 두 원소를  
검사하여 정렬하는 알고리즘

전체를 확인해야 하기 때문에 4번 반복을 해줌  
이미 교환한 요소는 더 교환할 필요 X : 4-i번 반복

왼쪽에 있는 수가 큰 경우,  
값을 변경해주는 코드를 작성함

53241

3 2 4 1 5  
2 3 1 4 5  
2 1 3 4 5  
1 2 3 4 5

12345

정렬 초기 과정



정렬 완료 후 출력결과

1. 정렬 알고리즘

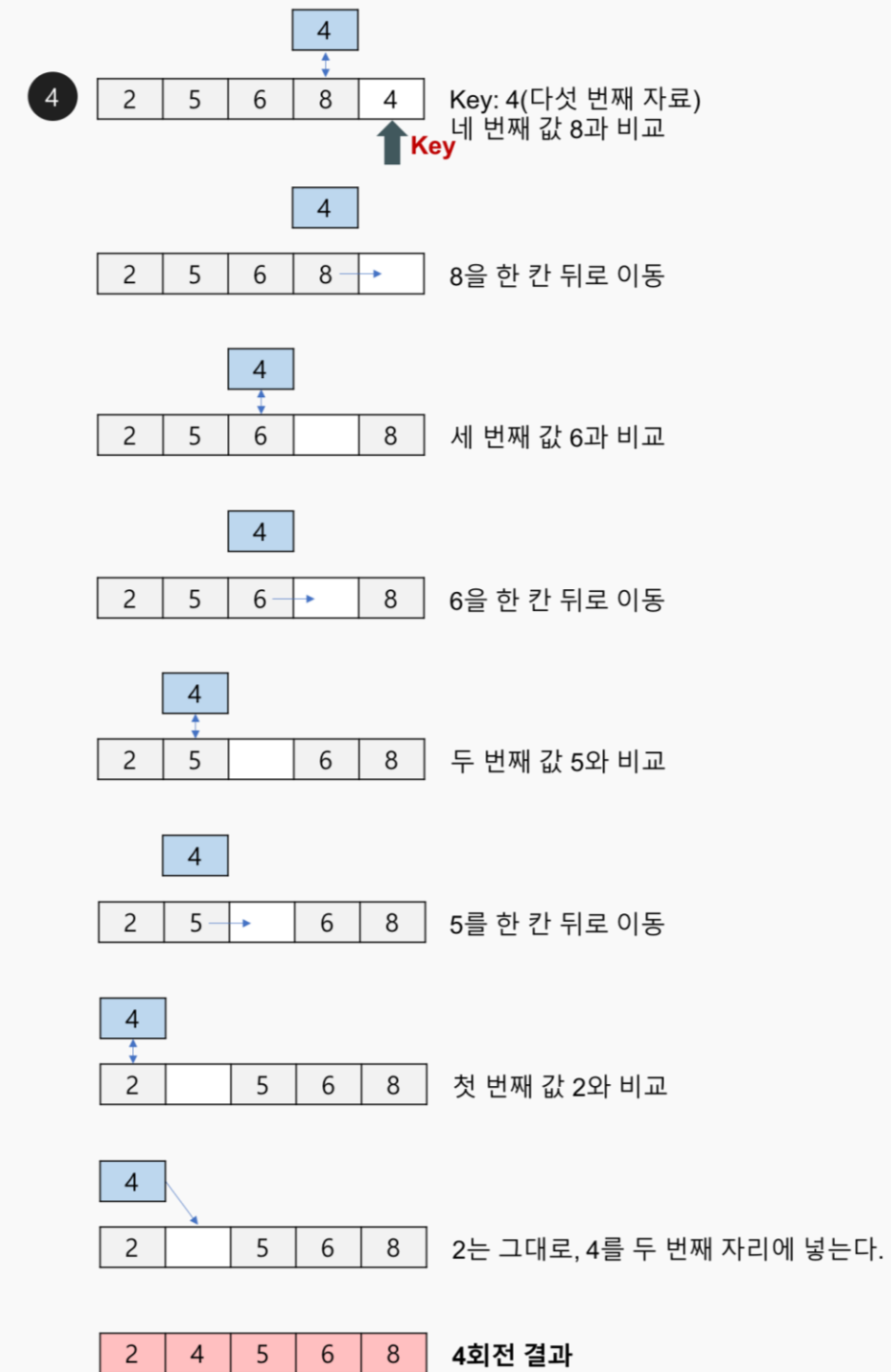
2. 퀵 정렬

3. 버블 정렬

4. 삽입 정렬

5. 93P 문제 해결하기

## 삽입 정렬 : 필요할 때만 각 데이터를 적절한 위치에 삽입하는 정렬 방법



오름차순  
완성상태

2 4 5 6 8

1. 정렬 알고리즘

2. 퀵 정렬

3. 버블 정렬

4. 삽입 정렬

5. 93P 문제 해결하기



**삽입 정렬** : 필요할 때만 각 데이터를  
적절한 위치에 삽입하는 정렬 방법

**< 장점 >**

- 안정된 정렬 방법
- 대부분이 정렬되어 있는 경우에 매우 효율적임

**< 단점 >**

- 비교적 많은 이동을 포함한다
- 정렬해야 할 크기가 많은 경우, 적합하지 않음

1. 정렬 알고리즘

2. 퀵 정렬

3. 버블 정렬

4. 삽입 정렬

5. 93P 문제 해결하기

```
#include <stdio.h>

int main() {
    int a[5] = {5, 3, 2, 4, 1};

    for(int i = 0; i<5; i++){
        int key = a[i];
        int j = 0;

        for(j = i-1; j >= 0 && a[j] > key; j--){
            a[j+1] = a[j];
        }

        a[j+1] = key;
    }
}
```

**삽입 정렬** : 필요할 때만 각 데이터를 적절한 위치에 삽입하는 정렬 방법

정렬해야 할  $a[i]$ 를 key로 받아줌

$a[j]$ 가 key보다 크기 전, 0 인덱스까지 반복  
-> 필요한 위치에 삽입해줌

53241

53241

35241

23541

23451

12345

12345

정렬 초기 과정



정렬 완료 후 출력결과

1. 정렬 알고리즘

2. 퀵 정렬

3. 버블 정렬

4. 삽입 정렬

5. 93P 문제 해결하기

정수를 저장하고 있는  
배열 'array'가 주어졌다.

이 'array'에 저장되어 있는 원소들이  
순서대로 정렬되어 있으면 1,  
아니면 0을 리턴해라

```
#include <stdio.h>

int sortCheck(int index[], int len);

int main() {
    int idx[5] = {1, 2, 3, 4, 5};
    int idx2[5] = {1, 3, 5, 2, 4};

    printf("%d\n", sortCheck(idx, 5));
    printf("%d\n", sortCheck(idx2, 5));
}

int sortCheck(int index[], int len){
    int count = 0;

    for(int i = 0; i<len; i++){
        if(index[i] < index[i+1]){
            count++;
        }
    }

    if(count == len) return 1;
    else return 0;
}
```

1. 정렬 알고리즘

2. 퀵 정렬

3. 버블 정렬

4. 삽입 정렬

5. 93P 문제 해결하기

gghh3017

17

1

Silver IV 373

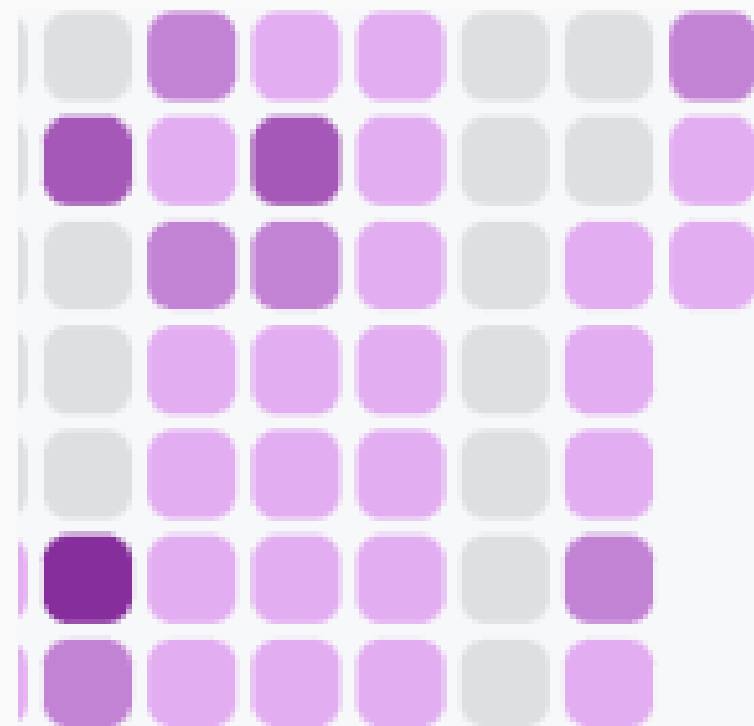
Silver III 승급까지 -27

선린인터넷고등학교

86문제 해결 1명의 라이벌

스트릭

최대 **23일** 연속 문제 해결, 현재 **8일**



4월

5월

5 2693	N번째 큰 수
5 2751	수 정렬하기 2
5 2822	점수 계산
5 10867	중복 빼고 정렬하기
5 10989	수 정렬하기 3 STANDARD
5 16435	스네이크버그
1 2750	수 정렬하기 STANDARD



*Thank you!*

**감사합니다**