

자료구조 (3가지 정렬의 종류)

정렬이란?

-수많은 자료를 특정 목적에 맞게 순서있게 재배치하는 것이고,
컴퓨터에서는, 레코드들을 키 값의 순서로 재배치하는 것이다.

우리가 자료구조에서 정렬 기법을 사용해야하는 목적에 대해 알아보자.

정렬은 컴퓨터에서, 가장 많이 이용되는연산 중 하나이고,
자료검색의효율성 제고, 실용성, 이론 설명 등을 위해서도 정렬기법이 필요하다

1.삽입정렬(insertion sort)



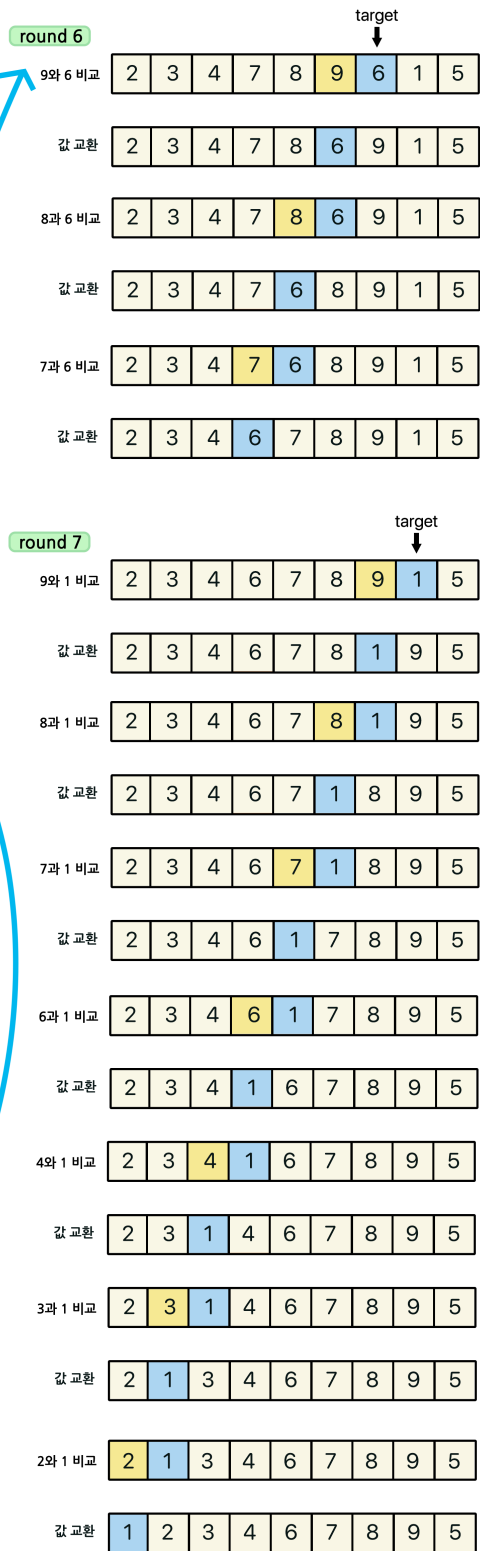
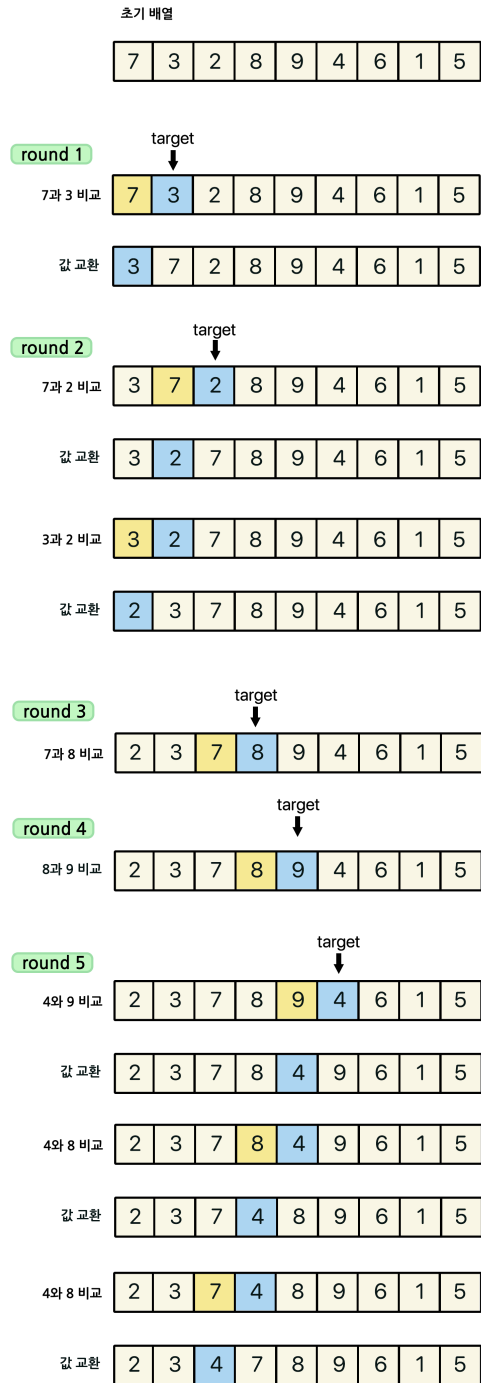
삽입 정렬은 정렬 대상을 두 부분으로, 정렬 안된 부분에 있는 데이터를 정렬 된 부분의 특정 위치에 "삽입"해 가면서 정렬을 진행하는 알고리즘이다.

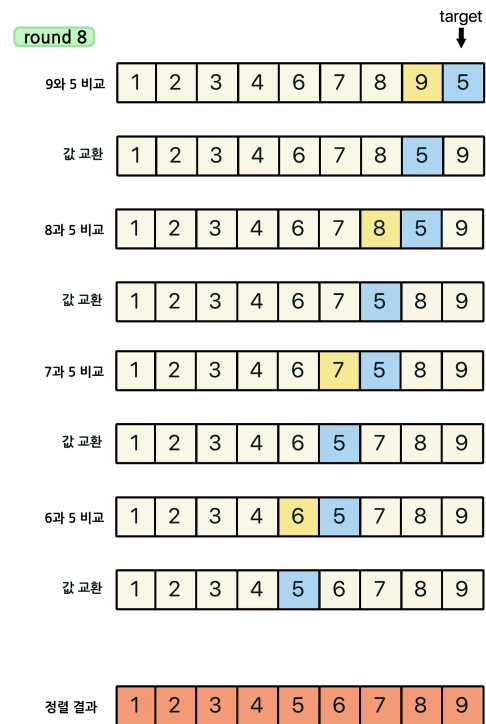
삽입정렬의 방법



1. 현재 타겟이 되는 숫자와 이전 위치에 있는 원소들을 비교한다. (첫 번째 타겟은 두 번째 원소부터 시작한다.)
2. 타겟이 되는 숫자가 이전 위치에 있던 원소보다 작다면 위치를 서로 교환한다.
3. 그 다음 타겟을 찾아 위와 같은 방법으로 반복한다.

예시 이미지





예시 자료

6 5 3 1 8 7 2 4

https://en.wikipedia.org/wiki/Insertion_sort

간단한 삽입정렬의 알고리즘 코드의 예시를 보자

```
package dd;

import java.util.Arrays;

public class bubblesort {
```

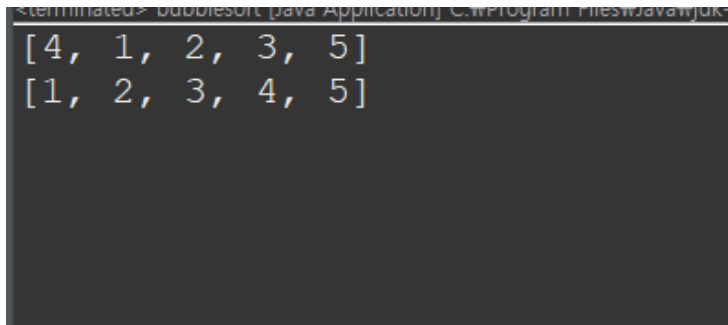
```

public static void main(String[] args) {
    // TODO Auto-generated method stub

    int array[] = {4, 1, 2, 3, 5}; //배열안의 숫자들 선언
    int i, j, temp; //3가지 변수 선언
    System.out.println(Arrays.toString(array));
    for (i = 1; i < array.length; i++){
        for(j = i; j > 0; j--){
            if(array[j] < array[j-1]){ //두 수를 비교함
                temp = array[j-1]; //빈 방에 수를 집어넣음
                array[j-1] = array[j]; //수를 집어넣어 생긴 공간에 다른 수를 집어넣음
                array[j] = temp; //다른 공간에 수를 넣어 생긴 빈공간에 빈방에 있던 수를 다시 집어넣음
            }
        }
    }

    System.out.println(Arrays.toString(array));
}
}

```



```

<terminated> BubbleSort Java Application [C:\Program Files\Java\jdk-
[4, 1, 2, 3, 5]
[1, 2, 3, 4, 5]

```

2.버블 정렬(Bubble sort)



Bubble Sort는 Selection Sort와 유사한 알고리즘으로 서로 인접한 두 원소의 대소를 비교하고, 조건에 맞지 않다면 자리를 교환하며 정렬하는 알고리즘이다.

버블 정렬의 방법

과정



과정 자료

6 5 3 1 8 7 2 4

<https://upload.wikimedia.org/wikipedia/commons/c/c8/Bubble-sort-example-300px.gif>

자바 알고리즘 구현

```
package dd;
```

```
import java.util.Arrays;

public class bubblesort {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        int array[] = {3,5,1,4,2}; //배열안의 숫자들 선언
        int i, j, temp; //3가지 변수 선언
        System.out.println(Arrays.toString(array));
        for (i = 0; i < 4; i++){
            for(j = 0; j < 4; j++){
                if(array[j] > array[j+1]){ //두 수를 비교함
                    temp = array[j]; //빈 방에 수를 집어넣음
                    array[j] = array[j+1]; //수를 집어넣어 생긴 공간에 다른 수를 집어넣음
                    array[j+1] = temp; //다른 공간에 수를 넣어 생긴 빈공간에 빈방에 있던 수를 다시 집어넣음
                    System.out.println(Arrays.toString(array));
                }
            }
        }
    }
}
```

3. 퀵 정렬(quick sort)



하나의 리스트를 피벗을 기준으로 두 개의 비균등한 크기로 분할하고 분할된 부분 리스트를 정렬한 다음, 두 개의 정렬된 부분 리스트를 합하여 전체가 정렬된 리스트가 되게 하는 방법이다.

퀵 정렬의 3단계 방법



1단계. 분할(Divide) : 입력 배열을 피벗을 기준으로 비균등하게 2개의 부분 배열(피벗을 중심으로 왼쪽 = 피벗보다 작은 요소들, 오른쪽 = 피벗보다 큰 요소들)로 분할한다.

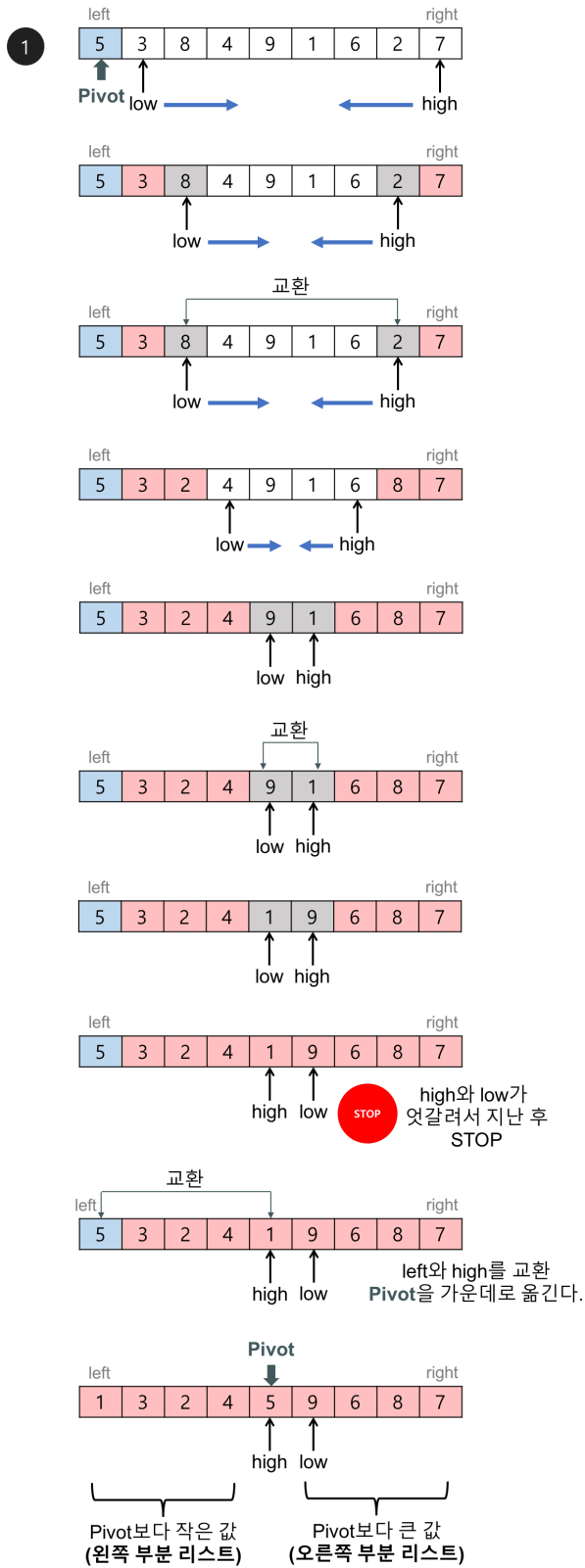
2단계. 정복(Conquer) : 부분 배열을 정렬한다. 부분 배열의 크기가 충분히 작지 않으면 순환 호출을 이용하여 다시 분할 정복 방법을 적용한다.

3단계. 결합(Combine) : 정렬된 부분 배열들을 하나의 배열에 합병한다.

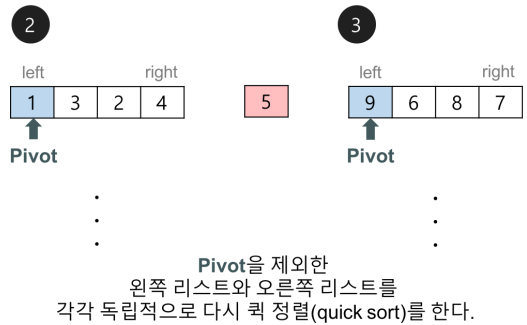
예시자료

초기상태

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



1회전 후 Pivot 5는 이미 제 위치에 있음을 알 수 있다.



리스트의 크기가 0이나 1이 될 때까지 반복

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

오름차순 완성상태

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

자바로 구현한 간단한 알고리즘

```
public class QuickSorter {
    public static List<Integer> quickSort(List<Integer> list) {
        if (list.size() <= 1) return list;
        int pivot = list.get(list.size() / 2);

        List<Integer> lesserArr = new LinkedList<>();
        List<Integer> equalArr = new LinkedList<>();
        List<Integer> greaterArr = new LinkedList<>();

        for (int num : list) {
            if (num < pivot) lesserArr.add(num);
            else if (num > pivot) greaterArr.add(num);
            else equalArr.add(num);
        }

        return Stream.of(quickSort(lesserArr), equalArr, quickSort(greaterArr))
            .flatMap(Collection::stream)
            .collect(Collectors.toList());
    }
}
```