정렬 알고리즘

주차	24주차
스터디 일자	@2024/04/11

삽입 정렬 (Insertion Sort)

삽입 정렬은 현재 비교하고자 하는 **target(타겟)**과 그 이전의 원소들과 비교하며 자리를 **교환(swap)**하는 정렬 방법

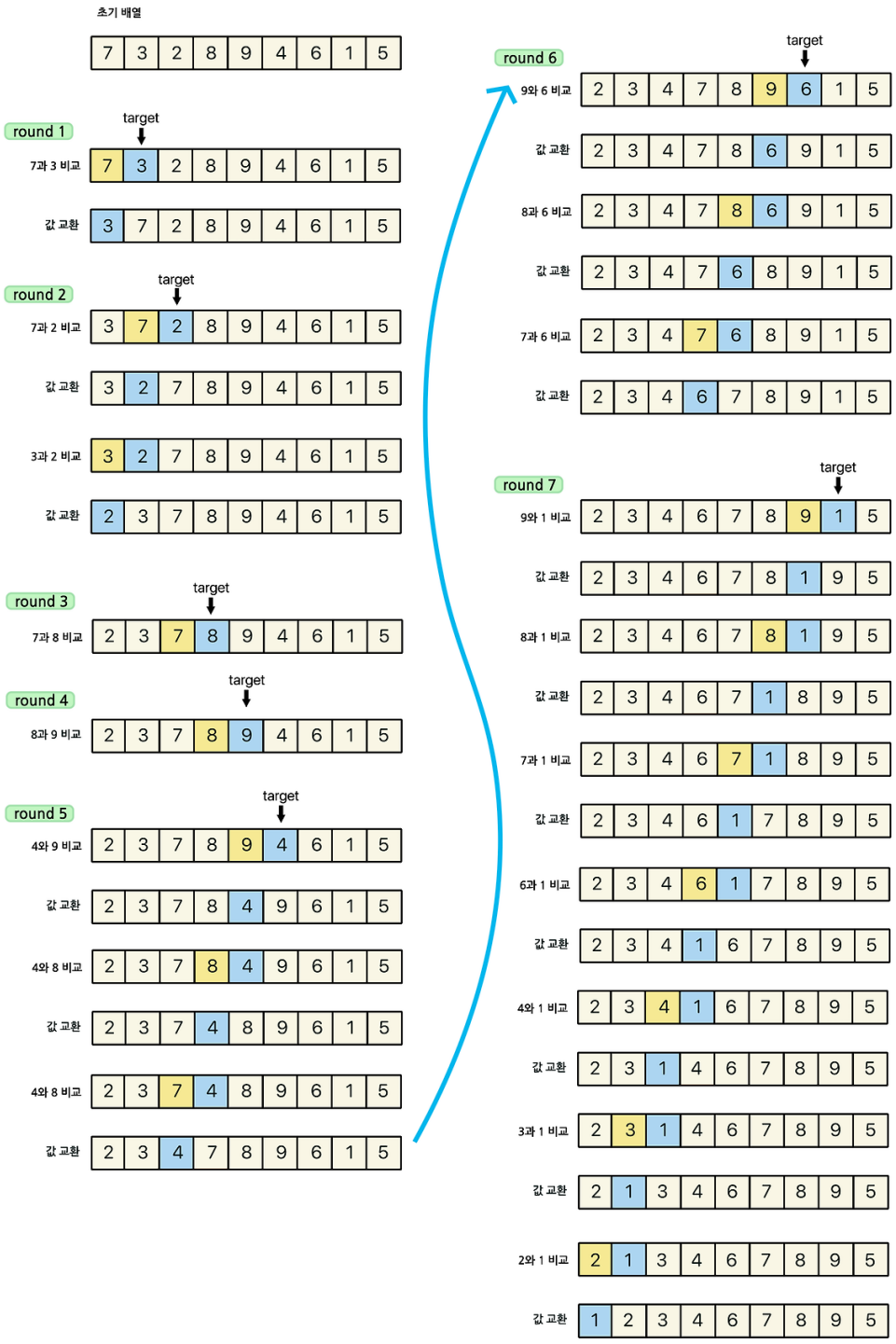
정렬 방식

앞에서부터 해당 원소가 위치 할 곳을 탐색하고 해당 위치에 삽입하는 원리를 사용한다.

삽입 정렬의 전체적인 과정은 이렇다. (오름차순 기준)

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.**



- 첫 번째 원소는 타겟이 되어도 비교 할 원소가 없기 때문에 처음 원소부터 타겟이 될 필요가 없고 두 번째 원소부터 타겟이 되면 된다.

구현 방법 (JAVA)

```
public class Insertion_Sort {

    public static void insertion_sort(int[] a) {
        insertion_sort(a, a.length);
    }

    private static void insertion_sort(int[] a, int size) {
        for(int i = 1; i < size; i++) {
            // 타겟 넘버
            int target = a[i];

            int j = i - 1;

            // 타겟이 이전 원소보다 크기 전 까지 반복
            while(j >= 0 && target < a[j]) {
                a[j + 1] = a[j];    // 이전 원소를 한 칸씩 뒤로 미룬다
                j--;
            }

            /*
             * 위 반복문에서 탈출 하는 경우 앞의 원소가 타겟보다 작다는 의미
             * 타겟 원소는 j번째 원소 뒤에 와야한다.
             * 그러므로 타겟은 j + 1 에 위치하게 된다.
             */
            a[j + 1] = target;
        }
    }
}
```

💡 결과적으로 타겟 이전 원소가 타겟 숫자보다 크기 직전까지 모든 수를 뒤로 한 칸씩 밀어내는 것이다.

삽입 정렬의 장단점!

[장점]

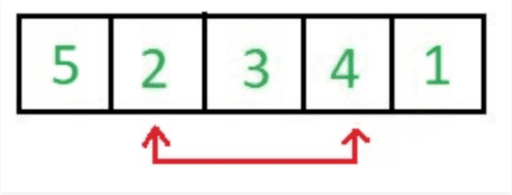
- 거의 정렬 된 경우 매우 효율적이다.

  즉, 최선의 경우 O(N)의 시간복잡도를 갖는다.
- 추가적인 메모리 소비가 작다. (제자리 정렬)
- 안정 정렬이 가능하다.

> 여기서 제자리 정렬, 안정 정렬이란?

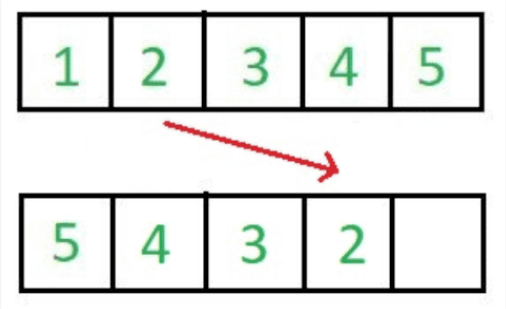
🤔 **추가적인 메모리 공간**을 많이 필요로 하지 않거나 전혀 필요하지 않은 정렬을 제자리 정렬이라고 한다.

제자리 정렬 O



추가적인 공간 없이 현재 배열 안에서 정렬이 다 이뤄진다.

제자리 정렬 X



추가적인 공간이 필요하다.

왼쪽처럼 추가적인 메모리 공간 없이 정렬이 이뤄진다면 제자리 정렬 알고리즘이다!

오른쪽처럼 추가적인 공간이 필요한 경우, **공간복잡도가 높다.**

정렬 전

동일한 키 값의 요소 순서가 정렬 후 **유지**가 되는 정렬을 **안정 정렬(Stable Sort)**이라고 한다.

[예시]

[3, 2, 1, **5**, 7, **5**] 배열이 있다고 가정한다.

이를 오름차순으로 정렬한다고 할 때,

[1, 2, 3, **5**, **5**, 7]이 된다면 안정 정렬

[1, 2, 3, **5**, **5**, 7]이 될 수 있다면 불안정 정렬이다.

안정 정렬에 해당하는 정렬은 버블 정렬, 삽입 정렬, 병합 정렬이 있다.

[단점]

- 역순에 가까울 수록 매우 비효율적이다.

  즉, 최악의 경우 O(n^2)의 시간 복잡도를 갖는다.
- 데이터의 상태에 따라서 성능 편차가 매우 크다.

삽입정렬은 버블정렬이나 선택 정렬과 이론상 같은 시간복잡도를 갖음에도 평균 비교 횟수에 대한 기대값이 상대적으로 적기 때문에 **평균 시간복잡도가 O(N2) 인 정렬 알고리즘 중에서는 빠른편에 속하는 알고리즘**이다.

번외 - 자바 내장 정렬

Arrays.sort

사용 알고리즘 : **dual pivot quicksort**

평균 : O(n log n) / 최악 : O(n^2)

안정 정렬 X

근데 얘는 자료형에 따라 내부에서 다른 정렬을 선택한다는데,

참조타입 정렬은 TimSort 쓴다고 함.

Collections.sort

사용 알고리즘 : TimSort (삽입 정렬 + 합병 정렬)

평균, 최악 : O(n log n)

안정 정렬 O

Type	Available method	Sorting Algorithm	Time Complexity
primitive (기본형)	Arrays.sort()	DualPivotQuicksort	Bset : O(NlogN)
			Average : O(NlogN)
			Worst : O(N^2)
reference (참조형)	Arrays.sort() Collections.sort()	TimSort	Bset : O(N)
			Average : O(NlogN)
			Worst : O(NlogN)

> 제 생각에는 모든 면에서 훨 나은 Collections.sort를 쓰는게 낫지 않을까 싶은걸요 ? ㅎ..ㅎ

https://st-lab.tistory.com/179

https://codingnojam.tistory.com/38