Chapter 28: Sorting

Parameter	Description
Stability	A sorting algorithm is **stable** if it preserves the relative order of equal elements after sorting.
In place	A sorting algorithm is **in-place** if it sorts using only `O(1)` auxiliary memory (not counting the array that needs to be sorted).
Best case complexity	A sorting algorithm has a best case time complexity of `O(T(n))` if its running time is **at least** `T(n)` for all possible inputs.
Average case complexity	A sorting algorithm has an average case time complexity of `O(T(n))` if its running time, **averaged over all possible inputs**, is `T(n)`.
Worst case complexity	A sorting algorithm has a worst case time complexity of `O(T(n))` if its running time is **at most** `T(n)`.

Section 28.1: Stability in Sorting

Stability in sorting means whether a sort algorithm maintains the relative order of the equals keys of the original input in the result output.

So a sorting algorithm is said to be stable if two objects with equal keys appear in the same order in sorted output as they appear in the input unsorted array.

Consider a list of pairs:

```
(1, 2) (9, 7) (3, 4) (8, 6) (9, 3)
```

Now we will sort the list using the first element of each pair.

A **stable sorting** of this list will output the below list:

```
(1, 2) (3, 4) (8, 6) (9, 7) (9, 3)
```

Because `(9, 3)` appears after `(9, 7)` in the original list as well.

An **unstable sorting** will output the below list:

```
(1, 2) (3, 4) (8, 6) (9, 3) (9, 7)
```

Unstable sort may generate the same output as the stable sort but not always.

Well-known stable sorts:

- Merge sort
- Insertion sort
- Radix sort
- Tim sort
- Bubble Sort

Well-known unstable sorts:

- Heap sort
- Quick sort