Stable Sort

↳ A sorting algorithm is stable if it preserves the relative order of equal elements.

Why important → Sorting objects
Multiple level sorting (eg. sort by age, then by name)

Example:

Input (value, original index)

(5, A), (3, B), (5, C) —stable sort→ (3, B), (5, A), (5, C)

(5, A), (3, B), (5, C) —unstable sort→ (3, B), (5, C), (5, A)

Stable
- Bubble sort
- Insertion sort
- Merge sort
- Counting sort
- Radix sort

unstable
- Selection sort
- Quick sort
- Heap sort

Adaptive Sort

↳ Takes advantage of existing order of the input → reduce runtime

Key Idea:
- Faster on nearly sorted arrays

Example:

[1, 2, 3, 5, 4] → Only one inversion → adaptive algorithm runs fast.

Adaptive
- Insertion sort (best O(n))
- Bubble sort (with optimization)
- Tim sort (Python, Java)

Not Adaptive
- Selection sort
- Merge sort
- Heap sort

Summary:

In-place → memory usage
Stable → relative order
Adaptive → Input awareness