Insertion Sort

In-place and Stable sorting.

Case	Run Time
Best	$O(n)$
Worst	$O(n^2)$
Average	$O(n^2)$

If the number of inversions is d, the run time is $\Theta(n+d)$.

Benefits: Even in the worst case, the algorithm is fast for small problems.

Bubble Sort

In-place and Stable sorting.

Case	Run Time
Best	$O(n)$
Worst	$O(n^2)$
Average	$O(n^2)$

It is significantly worse than insertion sort.

Improvement process:

Basic Bubble Sort

Flagged Bubble Sort

Range-limiting Bubble Sort

Alternating Bubble Sort

Merge Sort

Not in-place and Stable sorting.

Case	Run Time
Best	$O(n\ln(n))$
Worst	$O(n\ln(n))$
Average	$O(n\ln(n))$

Use to count inversions.

Run-time Analysis of Merge Sort

The time required to sort an array of size $n > 1$ is:

- the time required to sort the first half,
- the time required to sort the second half, and
- the time required to merge the two lists

That is: $$T(n) = \begin{cases} \Theta(1) & n = 1 \\ 2\,T\left(\frac{n}{2}\right) + \Theta(n) & n > 1 \end{cases}$$

Solution: $T(n) = \Theta(n \ln(n))$

Proof by Recursion Tree

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ \underbrace{2T(n/2)}_{\text{sorting both halves}} + \underbrace{n}_{\text{merging}} & \text{otherwise} \end{cases}$$