Stable Sorting algorithm:
A sorting algorithm is said to be stable if two objects with equal values appear in the same order in sorted output as they appear in the input unsorted array.

In place algorithm:
That does not need an extra space and produces an output in the same memory that contains the data by transforming the input 'in-place'. However, a small constant extra space used for variables is allowed.
O(1) implies that the changes are done in place, but vice versa may not be true.

Comparison of time complexities of different sorting algorithms

Algorithm Name	**Best Case**	**Average Case**	**Worst Case**	**Space Compl.**	**In-Place?**	**Stable ?**
Selection sort	O(n^2)	O(n^2)	O(n^2)	O(1)	Yes	No
Bubble Sort	O(n) (when already sorted)	O(n^2)	O(n^2)	O(1)	Yes	Yes
Insertion Sort	O(n)	O(n^2)	O(n^2)	O(1)	Yes	Yes
Merge Sort	O(nlogn)	O(nlogn)	O(nlogn)	O(n)	No	Yes
Heap Sort	O(nlogn)	O(nlogn)	O(nlogn)	O(n)	Yes	No
Quick Sort	O(nlogn)	O(nlogn) avg & pract.	O(n^2)	O(logn)	Yes	No
Count sort	O(n+range)	O(n+range)	O(n+range)	O(n+range)	No	Yes

1.Selection Sort:

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from the unsorted part and putting it at the beginning.
Unsorted part:[0,n-1]-->[1,n-1]-->....--> [n-,n-1]

https://ideone.com/jtj4pQ → Code

The good thing about selection sort is it never makes more than O(n) swaps and can be useful when memory write is a costly operation.

In place:It doesn’t require extra space.

2. Bubble sort:

It works by repeatedly swapping the adjacent elements if they are in the wrong order.In each iteration,the largest element gets placed at its right position.
The algorithm needs one whole pass without any swap to know it is sorted.

https://ideone.com/8U8nWh → code iterative as well as recursive.

3. Insertion Sort:

The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

It is just like we sort the cards.

In-place:

https://ideone.com/0bqXh0 → code

4. Merge Sort:

https://ideone.com/zcABJG → code
Not in place.
In the linked list ,it can be implemented in O(1) space.
Can be implemented recursively as well as iteratively.

5.Quick Sort

Partition function returns index of pivot i.e. pIndex and do changes in array such that elements left to pivot are less than pivot and elements right to pivot are greater than pivot. Initially,last element of subarray is chosen as pivot

Using divide and conquer ,we keep doing partition from (low,pIndex-1),(pIndex+1,high)

The worst T.C. will happen when our array will be sorted and we select smallest or largest indexed element as pivot

https://ideone.com/hUtE4s
Divide and conquer algorithm.

5.2 Randomised Quick Sort:
In Quick Sort,the last element of the subarray is chosen as pivot but in Randomised Q.S.,any random element can be chosen as pivot.
//Below two lines are different in ‘partition’,the rest are the same.
pIndex=random(low,high);
swap(arr[pIndex],arr[high]);

Randomized Quick sort has the worst T.C. : O(nlogn)

6.Count sort:

It is used when there are many data points but range is less like marks of students in a competitive exam.

Given nums array,we have to sort.
Step1: Find min and max of array and calculate range.
Step2: Make a frequency array of size ‘range’ where freq[i] denotes freq. of ‘i+min’ no.
Step3:

```
nums.clear();
for(int i=0;i<=range;i++){
        if(res[i]!=0){
            while(res[i]--!=0)
            nums.push_back(i+mn);
        }
 }
      return nums;
```

Many data points but range is less.
https://leetcode.com/submissions/detail/650980790/