Algorithms	Best Case Space Complexity	Best Case Time Complexity	Avg Case Time Complexity	Worst Case Time Complexity
Insertion Sort	***O (1)*** it necessitates some extra memory space for a key variable to perform swaps.	***O(n)*** already sorted array	***O(n²)*** when existing elements are in jumbled order, i.e., neither ascending nor scending	***O(n²)*** It occurs when we sort ascending array into descending order.
Bubble Sort	***O (1)*** an extra variable is required for swapping	***O(n)*** When array is already sorted	***O(n²)*** when existing elements are in jumbled order, i.e., neither ascending nor scending	***O(n²)*** when array elements are required to be sorted in reverse order
Shell Sort	***O(1)*** Algorithm has constant timing	***O(n*logn)*** when there is no sorting require	***O(n*log(n)²)*** When array elements are in jumbled order that is not properly ascending and not properly descending	***O(n²)*** It occurs when the array elements are required to be sorted in reverse order. That means suppose you have to sort the array elements in ascending order, but its elements are in descending order.
Selection Sort	***O(1)*** An extra variable is required for swapping.	***O(n²)*** when there is no sorting required	***O(n²)*** It occurs when the array elements are in jumbled order that is not properly ascending and not properly descending	***O(n²)*** It occurs when the array elements are required to be sorted in reverse order.
Merge Sort	***O(n)*** An extra variable is required for swapping.	***O(n log n)*** It occurs when there is no sorting required, i.e. the array is already sorted.	***O(n log n)*** It occurs when the array elements are in jumbled order that is not properly ascending and not properly descending.	***O(n log n)*** It occurs when the array elements are required to be sorted in reverse order. That means suppose you have to sort the array elements in ascending order, but its elements are in descending order.
Count Sort	O(max) The larger the range of elements, the larger the space complexity.	O(N + K) It occurs when there is no sorting required, i.e. the array is already sorted	***O(N + K)*** It occurs when the array elements are in jumbled order that is not properly ascending and not properly descending.	***O(N + K)*** It occurs when the array elements are required to be sorted in reverse order.
Quick Sort	***O(log(n))*** Since quicksort calls itself on the order of	***O(n*logn)*** the best-case occurs when the pivot element	***O(n*logn)*** It occurs when the array elements are in jumbled	***O(n²)*** worst case occurs when the pivot element is either greatest

	log(n) times at each recursive call a new stack frame of constant size must be allocated. Hence the O(log(n)) space complexity.	is the middle element or near to the middle element.	order that is not properly ascending and not properly descending.	or smallest element. Suppose, if the pivot element is always the last element of the array, the worst case would occur when the given array is sorted already in ascending or descending order
Radix Sort	***O(n+k)*** Because Radix sort employs Counting sort, which uses auxiliary arrays of sizes n and k, where n is the number of elements in the input array and k is the largest element among the dth place elements (ones, tens, hundreds, and so on) of the input array.	***Ω (n+k)*** It occurs when there is no sorting required, i.e. the array is already sorted.	***ϑ(nk)*** It occurs when the array elements are in jumbled order that is not properly ascending and not properly descending.	***O(nk)*** It occurs when the array elements are required to be sorted in reverse order. That means suppose you have to sort the array elements in ascending order, but its elements are in descending order.