|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithms | Best Case Space Complexity | Best Case Time Complexity | Avg Case Time Complexity | Reasons |
| 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 | Bubble is best when values are arranged and worst for when values are not arranged and we’ve to sort them. |
|  |  |  |  | When the given array list is already sorted the total count of comparisons of each interval is equal to the size of the given array. then it is best case else avg or worst case. |
|  |  |  |  | There ae two nested loops.  One loop to select an element of Array one by one = O(N). Another loop to compare that element with every other Array element = O(N)  Therefore overall complexity = O(N) O(N) = O(N\*N) = O(N2) |
|  |  |  |  | Merge sort always divides the array into two halves and takes linear time to merge two halves. |
|  |  |  |  | N is the number of elements in the input array and K is the range of input. 1)-sort count 2)-set position  3)-output character array |
|  |  |  |  | **Reasons** |