**Insertion Sort**

Simple and efficient comparison sort. In this algorithm, each iteration removes an element from the input data and inserts it into the correct position in the list being sorted. The choice of the element being removed from the input is random and the process is repeated until all input elements have gone through.

**Algorithm:**

Step 1 − If it is the first element, it is already sorted. return

Step 2 − Pick next element

Step 3 − Compare with all elements in the sorted sub-list

Step 4 − Shift all the elements in the sorted sub-list that is greater than the value to be sorted

Step 5 − Insert the value

Step 6 − Repeat until list is sorted

**Analysis:**

Worst case analysis: Worst case occurs when for every i the inner loop has to move all elements arr[1].......a[i-1]

T(n) = O() = O()

Average case analysis: For average case, inner loop will insert arr[i] in the middle of arr[1]……arr[i-1].

T(n) = O()

Best case analysis: T(n) = O(n)

Worst case space complexity: O() total, O(1) auxiliary