**STUDENT ID: 10663255**

**CSCD 304**

**Summary of Insertion Sort Algorithm**

Insertion sort algorithm is a type of sorting algorithm that search a list or an array with an unsorted element and place it at the suitable place after each iteration.

Consider having ten cards out of a deck of cards in your hand. And they are sorted, or arranged in ascending order of their numbers.

If I give out another card, and ask you to insert the card in just the right position, so that the cards in your hand are still sorted. What will you do?

Well, you will have to go through each card from the starting or the back and find the right position for the new card, comparing it value with each card. Once you find the right position, you will insert the card there.

Similarly if more new cards are provided to you, you can easily repeat the same process and insert the new cards and keep the cards sorted to.

This is exactly how insertion sort works. It starts from the index one not zero, and each index starting from index one is like a new card, that you have place at the right position in the sorted subarray on the left.

Insertion sort works like this;

We start by making the second element of the given array the key. The key element here is the new card that we need to add to our existing sorted set of cards.

Then we compare the key element with the elements before it, in this case, element at index zero. If the key element is less than the first element, we insert the key element before the first element. If the key element is greater than the first element, then we insert it after the first element.

Then, we make the third element of the array as a key and will compare it with element to its left and insert it at the right position. And we go on repeating this, until the array is sorted.

Insertion sort is an efficient sorting algorithm, as it does not run on preset conditions using for loops, but instead it uses one while loop, which avoids extra steps once the array get sorted.

Even insertion sort is efficient, still, if we provide an already sorted array to the insertion sort algorithm, it will still execute the outer for loop, thereby requiring n steps to sort an already sorted array of n elements, which makes its best case time complexity a linear function of n.

Worst Case time complexity [Big-O]: O (n2)

Best case time complexity [Big-omega]: O (n)

Average time complexity [Big-theta]: O (n2)