+ Sorting Algorithms:

Algorithm & Implementation

1. "Selection Sort":

- in this algorithm, in first iteration the first position gets selected and element which is at selected position gets compared with remaining all its next position elements, if in any comparison element which is at selected position is found greater than element at any other position then they gets swapped, and by this way in first iteration the smallest element in an array gets settled at the first position
- in second iteration second position gets selected and element which is at selected position gets compared with all its next position elements and as per condition mentioned above and second smallest element gets settled at second position and so on, and hence in max (n-1) no. of iterations all elements in an array gets arranged in a sorted manner.

Best case : O(n^2)
Worst case : O(n^2)
Average case: O(n^2)

2. "Bubble Sort":

- in this algorithm elements which are at two consecutive positions gets compared, if they are not in order (i.e. if prev position element is greater than next position element) then swapping takes place otherwise no need of swapping.
- by applying above logic, in the first iteration largest element gets settled at last position, in second iteration second largest element gets settled at second last position and so on, in max (n-1) no. of iterations all array elements gets arranged in a sorted manner in an ascending order.

10 20 30 40 50 60

Best Case : Big Omega(n) -> it occures only if all array ele's are already sorted
Worst Case : Big Oh(n^2)
Average Case: Big Theta(n^2)

- this algo is also called as "sinking sort".
- selection sort & bubble sort algo's are simple but these algo's are not efficient for larger input size array.

3. Insertion Sort:

```
for( i = 1 ; i < size ; i++ )
{
        j = i-1;
        key = arr[i];

        while( j >= 0 && arr[j] > key )
```

Sunbeam

```
        {
                arr[j+1] = arr[j];
                j--;
        }

        arr[j+1] = key;
}
```

Best Case : Big Omega(n) -> it occures only if all array ele's are already sorted
Worst Case : Big Oh(n^2)
Average Case: Big Theta(n^2)

- insertion sort algorithm is an efficient for already sorted input sequence by design, and hence insertion is an efficient sorting algo for smaller input size array (if size of an array is smaller then insertion sort is efficient than quick sort as well).

Only Algorithm:
4. Quick sort:

partitioning:
- we need to select pivot element
- we need to shift ele's which are smaller than pivot towards its left side as possible, and ele's which are greater than pivot need to shift towards as right as possible
- ele's which are at left side of pivot will be referred as "left partition", whereas ele's which are at right side of pivot will be referred as "right partition", and pivot ele gets settled at its appropriate position

- apply partitioning further on left partition as well on right partition, till size of subarray is greater than 1.

Best Case : Big Omega(n (log n))
Worst Case : Big Oh(n^2)
Average Case : Big Theta(n (log n))

```
pivot = arr[left];

while( i < j )
{
        while( i <= right && arr[i] <= pivot )
                i++;

        while( arr[j] > pivot )
                j--;

        if( i <= j )
                SWAP(arr[i], arr[j]);
```

SimBeam

```
}

SWAP(arr[j], arr[left]);
```

5. Merge Sort

Best Case : Big Omega(n (log n))
Worst Case : Big Oh(n (log n))
Average Case : Big Theta(n (log n))

O(1) < O(log n) < O(n) < O(n log n) < O(n^2)

SunBeam