**Day 5: 22 February 2025**

Sorting user defined object property with help to sort function in ascending or descending order.

Pre defined sorting algorithms.

**Time complexity: (Best, Worst and Average) :** In Soring algorithms time complexity describe how the run the of an algorithms increase as the input of values or data grow.

Best (Omega notation): **Ω**

Average (Theta notation): Θ-Notation

Worst (Big O Notation): Notation

**Bubble sort** Bubble sort is a simple sorting technique which also known as comparison – based sorting algorithms that repeatedly steps through a list or array, it compared adjacent element and swap them if are in the wrong order. This process continues util the entire list or array is sorted with each pass “bubble “ the largest unsorted element to its correct position. This technique is good if list hold very less data.

Best case : if data already sorted 🡪 only one pass needed.

Average case : random input or data 🡪 multiple passed required.

Worst case : reverse sorted 🡪 maximum swap is required.

**Selection sort :** selection sort is another simplest sorting technique it works repeatedly find the smallest or largest elements from the unsorted portion of the array and swapping it with the first unsorted elements. This process divides the array into a sorted and unsorted part, gradually growing the sorted section.

Best case : if list sorted, the selection sort still goes through the whole list and check each elements. So the time complexity is till O(n2)

Average case : the time complexity is till O(n2)

Worst case : the time complexity is till O(n2)

**Selection sort** perform fewer swap than Bubble sort. It only makes one swap per iteration (when it select the smallest or largest elements). This sort always does same number of comparison no matter the elements are sorted, random or unsorted.

In Bubble sort you could make multiple swap in each iteration an element when bubble up or down. It depends upon the initial order it may worst case or best case.

3. insertion sort : insertion sort is another simplest sorting technique. In this technique of we build the sorted section of the array one element at a time by comparing each new element with the element in sorted part and inserting it in the correct position.

Best case : if element already sorted : only one comparison per elements.

Average case : if elements are random : many swaps needed.

Worst case : if elements reverse order or descending order : many shift is required.

4. **Merge sort :** Merge sort is a type of sorting technique which use divide-and-conquer rules. In merge sort it divided list into 2 parts that is know as halves, this process perform recursively and them merges those sorted halves back together.

If array or list hold 100 data or element

1st we divided 1 to 50 and 51 to 100

2nd we divided 1 to 25 and 25 to 50

3rd we divide 1 to 12 and 13 to 25

4th we divided 1 to 6 and 7 to 12

5th we divided 1 to 3 and 4 to 6

6th we 1 to 2 and 3

7th we 1 and 2

Best case : O(n log n): even if list or array already sorted. Merge sort still divides and merge the list.

Average case : O(n log n): on average merge sort the list and sort the smallest part O(log n) steps, with each step taking O(n) times.

Worst case : O(n log n) times.

Merge sort always O(n log n) which is much faster than simple algorithms like Bubble sort or Selection sort or Insertion sort.

**Quick sort** Quick sort is also one the efficient sorting technique which use divide and conquer rules it use pivot element from the array and base upon pivot value we divided and array into two parts.

Pivot element can be first, last, middle or random number.

**Searching algorithms**

Linear search : it is also known as sequential search. In this search technique we search each element from array one by one. If element present we break the look.

Best base if element is present at the beginning no need to search till end.

Worst case if element present at last or not present we need to check all element from array.