✳ Sorting ✳

1. Bubble - Sort
2. Insertion - Sort
3. Selection - Sort.

1. **Bubble Sort:-** The Sorting algo. find the largest value and place it into the their right position.
- In each time only one value place it into there right place.
- Explantion:-

```
     0   1   2   3   4
   ┌───┬───┬───┬───┬───┐
   │ 5 │ 3 │ 1 │ 2 │ 4 │
   └───┴───┴───┴───┴───┘
```

unsorted
↓

Bubble sort (run the loop for total-1s array size)

Code:-

```
public void Bubble_Sort (int [] arr)
{
    for(int counter = 0; counter < arr.len -1; i++)
    {
        for (int i=0; i<arr.length-1; i++)
        {
            int temp;
            if (arr[i] > arr[i+1])
            {
                temp = arr[i];
                arr[i] = arr[i+1];
                arr[i+1] = temp;
            }
        }
    }
}
```

Expla

mi

Selection Sort :-

Explanation :- Put the Smallest value into their

p correct position.

```
(; itt)
```

Explanation :-

	0	1	2	3		4
	5	3	2	1		2

min = aor[0];

if (min > aor[j]) {

{ Swap the value's;

}

```
public void Selection Sort ($ o int[]
                                    arr)
{
    int min = arr[0];

    for(int i = 0; i < arr.length-1; i++)
    {
        int min = arr[i];

        if(arr

        for(int j = i+1; j < arr.length; j++)
        {
            int temps
            if( min > arr[j])
            {
                temp = arr[i];

                arr[i] = arr[j];

                arr[j] = min;

                min = temp;
```

```
    {    {    }
```

Insertion Sort :-

In insertion Sort array divided into the two parts first one Sorted and other are unsorted.

Code :-

```
public void Insertion_Sort (int []arr){
    for(int counter =1; counter <= arr.length()-1;
                                    counter ++)
    {
        int val = arr[counter];
        int j = counter -1;

        while (j >=0 && val < arr[j])
        {
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = val;
    }
}
```

* Searching *

1. Linear Search :-
2. Binary Search :-

* Linear Search :- It very simple and easy algo.

```
for(i=0; i<=arr.length-1; i++)
{
    if(arr[i] == item)
}
```

* binary Search :- In Binary Search we used the three different pointer to point the array.

low - arrays [low] ;

low :- It used for loower bound of the array.

high : It used for upper bound of the array.

mid = $\underline{low + high/2}$ — This cal^n the mid of the array.

Note :- for binary Search array must be sorted !!

Codes:-

```
public void int a Binary - Search (int[]
{   int low-bound = 0;
    int high-bound = am.length - 1;
    int mid = (low + high)/2

    while ( low <= high)
    {
        if ( am[mid] > Item)
        {
            high = mid - 1;
        }

        else if (am [mid] < Item)
        {
            low = mid + 1;
        }
```

```
    else
        return mid;

    } mid= low+high/2 ;

}

return -1;

}
```

MAY 2022
M T W T F S S
5 Thursday
MAY
STAR
Personal & Caring

* The list of problem using binary tree Search

1. lower and upper bound find the given item.

2. Book allocation problem.

3. Painter partition problem.

4. Aggressive cows.

And many more on youtube !!!

✳ Time Complexity ✳

Cases	BubbleSort	Modified BubbleSort
BestCase	$O(n^2)$	$O(n)$
Avg.Case	$O(n^2)$	$O(n^2)$
WorstCase	$O(n^2)$	$O(n^2)$

Cases	SelectionSort	Insertion
Best Case	$O(n^2)$	$O(n)$
Avg.Case	$O(n^2)$	$O(n^2)$
WorstCase	$O(n^2)$	$O(n^2)$