__/__/____

Sorting :-

16 september

Selection sort :-

{13, 46, 24, 52, 20, 9}
 0 1 2 3 4 5

9 {46 24 52 20 13} step 1
9 13 {24 52 20 46} 2
9 13 20 {52 24 46} 3
9 13 20 24 {52 46} 4
9 13 20 24 46 52 5

swap at index 0, & minimum index {0, n-1}
swap at index 1, & minimum index {1, n-1}
swap at index 2, & minimum index {2, n-1}
⋮
n-2

$n + n-1 + n-2 + n-3 - - - 2 = \frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2}$

$O(n^2)$.

Pseudo code :-

```
For(int i=0; i<=n-2; i++){
    int mini=i;
    for(int j=i; j<=n-1; j++){
        if(arr[j]<arr[mini]){
            mini=j;
        }
    }
```

swap(arr[i], arr[mini]);
}

$$O(n^2)\begin{cases}\text{Best}\\ \text{worst}\\ \text{Average}\end{cases}$$

<u>Bubble sort :-</u>

{push the max^m to the Last by adjacent swaps}

13 , 46, 24, 52, 20, 9

13, 46, 24, 52, 20, 9
✓ 52 ✓
✗ 46 ✗ 46 ✗

13 24 46 20 9 $\boxed{52}$ step 1

13 24 20 9 $\boxed{46 \quad 52}$ step 2

13 20 9 $\boxed{24 \quad 46 \quad 52}$ step 3

13 9 $\boxed{20 \quad 24 \quad 46 \quad 52}$ step4

$\boxed{9 \quad 13 \quad 20 \quad 24 \quad 46 \quad 52}$ step5

0 - -. n-1   i = n-1 ; i >= 1
0 - - - n-2
0 - - - n-3
0 - - - n- 4
0 - - n- 5
0 - - 1

__/__/__

Pseudo code :-

```
for(int i = n-1 ; i >= 0 ; i--){
    for(j = 0 ; j <= i-1 ; j++){
        if (arr [j] > arr[j+1]){
            swap(arr[j], arr[j+1]);
        }
    }
}
```

$n + n-1 + n-2 + n-3 + n-4 \dots 2+1 = \frac{n(n+1)}{2}$

$O(n^2)$ → worst complexity
└→ Average complexity

{Already sorted} $O(n)$ → Best complexity

Insertion sort :- {Takes an element and place it in correct order}

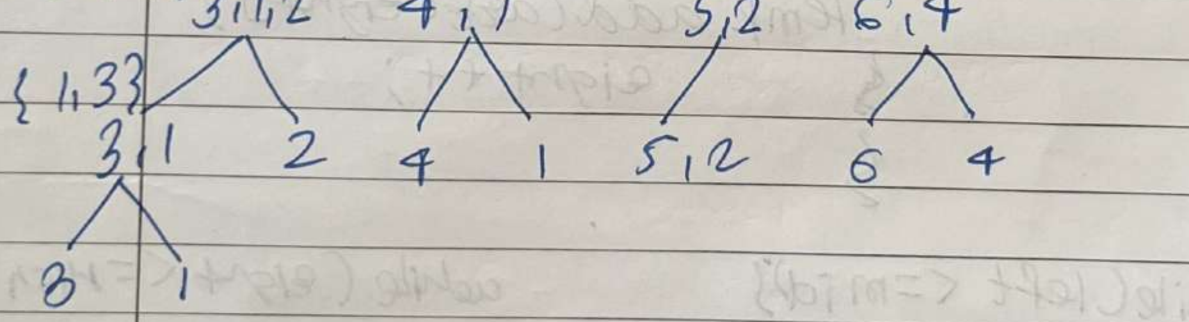
14, $\textcircled{9}$, 15, 12, 6, 8, 13
$\textcircled{9}$, 14, 15, $\textcircled{12}$, 6, 8, 13
9, $\textcircled{12}$, 14, 15, $\textcircled{6}$, 8, 13
$\textcircled{6}$, 9, 12, 14, 15, $\textcircled{8}$, 13
6, $\textcircled{8}$, 9, 12, 14, 15, $\textcircled{13}$
6, 8, 9, 12, $\textcircled{13}$, 14, 15

Pseudo code :-

```
for (int i=0 ; i<=n-1 ; i++){
        j = i;
    while ( j>0 && a[j-1] > a[j]){
            swap - - -
            j--;
    }
}
```

$O(n^2)$ worst case
Avg case
$O(n)$ -Best case {sorted}

Merge sort :- (Divide & Merge)



T.C O(nlogn) S.C O(n)

_/_/_

Pseudo Code :-

```
merge-sort(arr, low, high){
    if (low >= High) return;
     mid = low + High / 2;
    merge-sort(arr, low mid);
    merge-sort(arr, mid+1, High);
   merge (arr, low, mid, High);
   }

merge (arr, low, mid, High){   temp = [];
      left = low
      right = ~~Hi~~ mid + 1;
     while(left <= mid && right <= High){
           if (arr[left] < arr[right]){
                 temp.add(arr[left]);
                    left++;
           }
          else {
             temp.add(arr[right]);
                 right++;
           }
          }

 while(left <= mid){                        while (right <= High) {
     temp.add(arr[left])}                      temp.add(arr[High]);
 }         left++;                             right++;
                                          }

}
```

T.C O(nlogn) S.C O(1) __/__/____

Quick Sort :- ① Pick a Pivot and place it in its correct Place. In sorted array
② Smaller on the left larger on the right

Pivot
↓
→ 4 , 6 , 2 , 5 , 7 , 9 , 1 , 3
2 , 1 , 3 $\boxed{4}$ 6 , 5 , 7 , 9
← smaller   larger →

→ 2 , 1 , 3   Pivot ↓ { 6 , 5 , 7 , 9 }
Pivot
{ 1 2 3 } { 4 }
4 { 5 6 7 9 }

~~Arrays~~ ~~essay~~ easy
- find largest element
- second largest element
- check array is sorted
- Remove duplicate from sorted array