Notes From Sorting Algorithms :-

① Insertion sort / selection sort. } → concept

② Quick sort / Merge sort

Insertion sort

steps : Move left to right

① If you are at index i, if A[i] < A[i-1] swap.

② keep moving down till everything from left is always sorted.

③ Move to next index.

Code :-

```
for i: 1 to length (A)-1 :
        j=1
        while j > 0 and A[j-1] > A[j]
              swap (A[j], A[j-1])
              j=j-1
```

Example:-

sort

1 3 5 2 6 4

A[i]=1 :- All sorted :→ 1 3 5 2 6 4

A[i]=3 :- 3>2, all sorted ⇒ 1 3 5 2 6 4

A[i]=5 : 5>3, all sorted ⇒ 1 3 5 2 6 4

A[i]= 2 :

2<5 swap → 1 3 2 5 6 4

2<3 swap → 1 2 3 5 6 4

2>1 → OK

A[i] = 6, 6>5, all ok :→ 1 2 3 5 6 4

A[i] = 4,

4<6 swap → 1 2 3 5 4 6

4<5 swap → 1 2 3 4 5 6

4>3 → OK

Sorted Array : 1 2 3 4 5 6

② Selection Sort :-

Idea :- Find minimum → put to left side
→ continue.

```
for j = 0 , j < n-1 , j++ ,
    int iMin = j ;
    for (i = j+1 , i < n , i++)
        if a[i] < a[iMin]
            iMin = i ;

        if (iMin != j)
            swap (a[j] , a[iMin])
```

For example → Go To Next page

1 3 5 2 6 4

↑
*

Step1 Find min starting @ 1 ——————, replace with 1

New array : 1 3 5 2 6 4

Step2 Find min starting @ 1 3 5 2 6 4

New array = 1 2 5 3 6 4

Step3 Find minimum start @ 1 2 5 3 6 4

New array = 1 2 3 5 6 4

Step4 Find minimum start @ 1 2 3 5 6 4

New array : 1 2 3 4 5 6

Step5 Find min starting @ 1 2 3 4 5 6

New Array : 1 2 3 4 5 6

Step6 : Repeat for final :-

Ans 1, 2, 3, 4, 5, 6

Quicksort :-

3 1 5 2 6 4

Correct: 1 2 3 4 5 6

No Element in its correct position

Qs: Can we get 1 element in its correct position ?

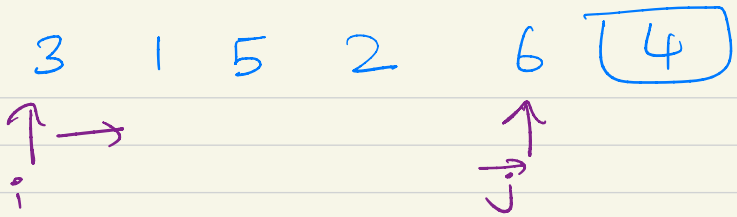
eg: 3 1 5 2 6 [4] → want this in correct posn

How can I arrange this array so that 4 is in correct position ?

3 1 5 2 6 [4]

Goal: [Elements smaller than 4] [4] [Elements larger than 4]

How to do this ? ⇒ Partition Function



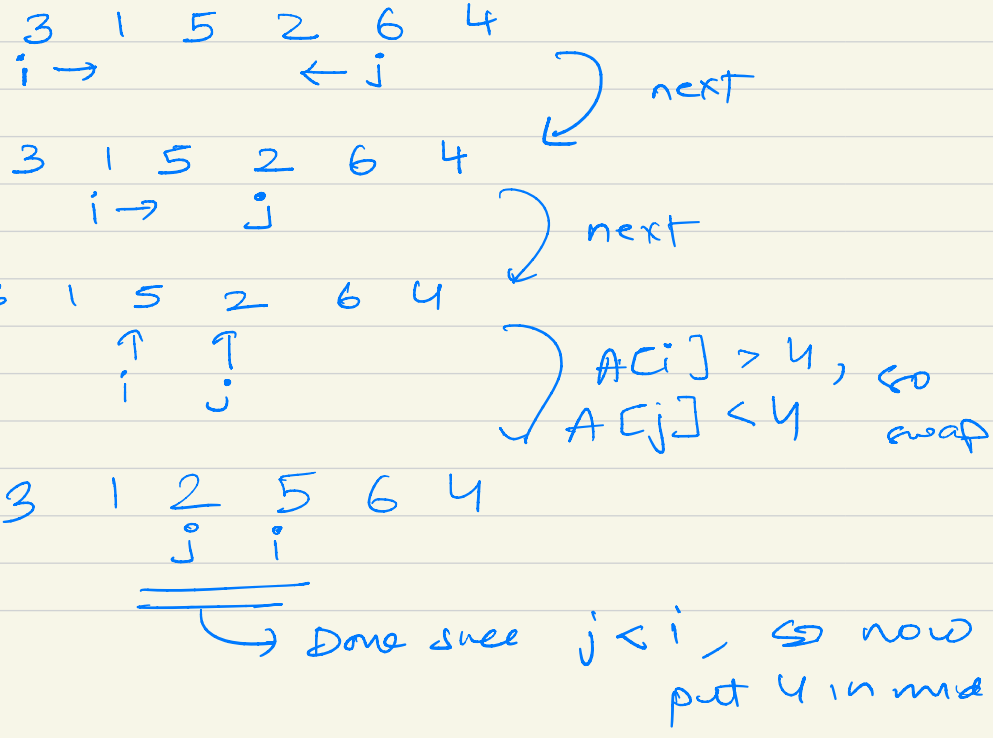
① Move $i \rightarrow$, $j \leftarrow$

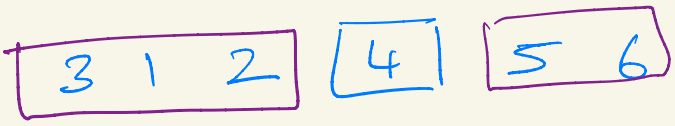
② If $A[i] < 4$, move right
If $A[j] > 4$, move left.

③ whenever you have $A[i] > 4$ and $A[j] < 4$, swap the 2 places

⑤ Keep doing this.

eg:





∴ this array has been partioned on the element [4], ∴ 4 is in correct position.

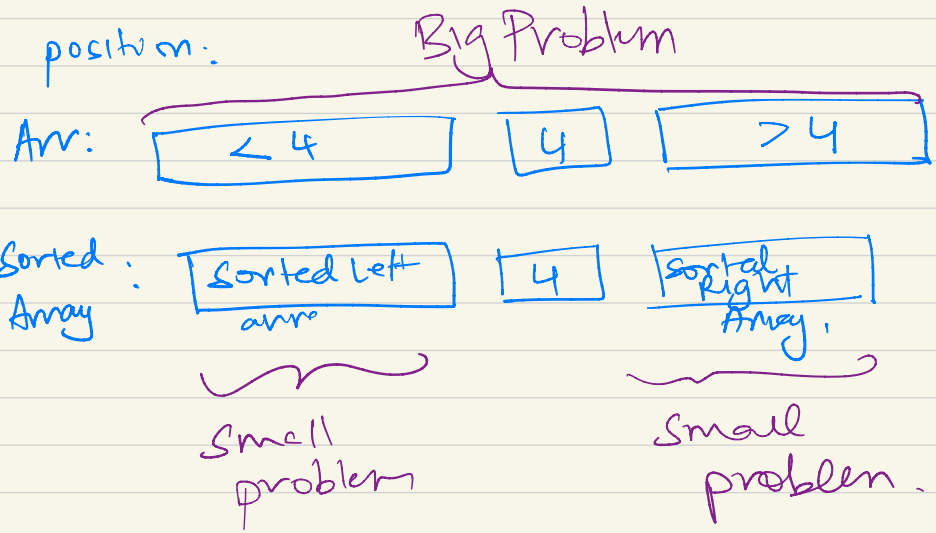
(partition the array based on the last element)

```
Partition ( Arr , low , high ) :{
    pivot = Arr[high]
    i = low - 1
    j = high - 1
while  j > i :-
    while  Arr[i] ≤ pivot :
            i++ ;
    while  Arr[j] ≥ pivot :
            j-- ;
    if  i < j :
            swap( Arr[i] , Arr[j] )                    }
    Arr[high] = Arr[i]
    Arr[i] = pivot
    return  i , Arr[ : i] , Arr[j : ]
```

This function can now place the Rightmost element into the correct location.

How can we use this to sort?

If we know an array has a pivot in correct position.




```
QuickSort [Arr] := {

    pivot, leftArray, RightArray = Partition [Arr]

    sortedLeftArray = QuickSort [leftArr]
    sortedRightArray = QuickSort [RighArray]

    return [sortedleftArray, pivot, sortedRightArray]
}
```

Look @ uploaded code!

Mergesort :-

Qs: can you sort an array which looks like this :

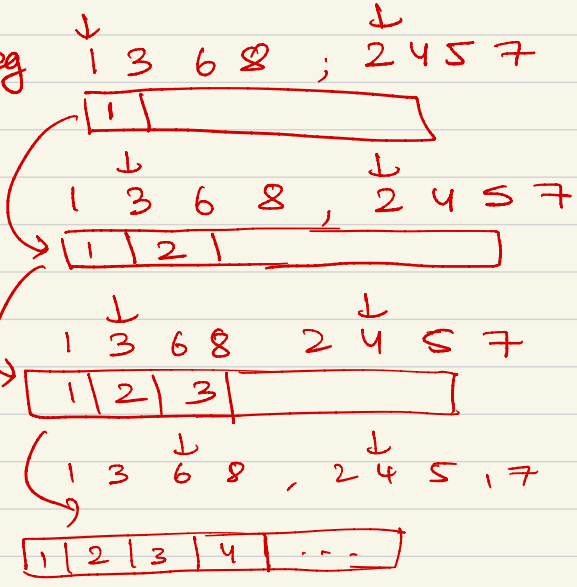
1 3 6 8 | 2 4 5 7

Sorted Left Array | Sorted Right Array

Easy?

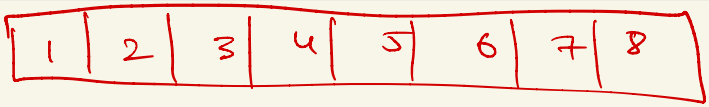
Go left to right on both arrays, add to this empty array whichever is smaller.

eg 1 3 6 8 ; 2 4 5 7

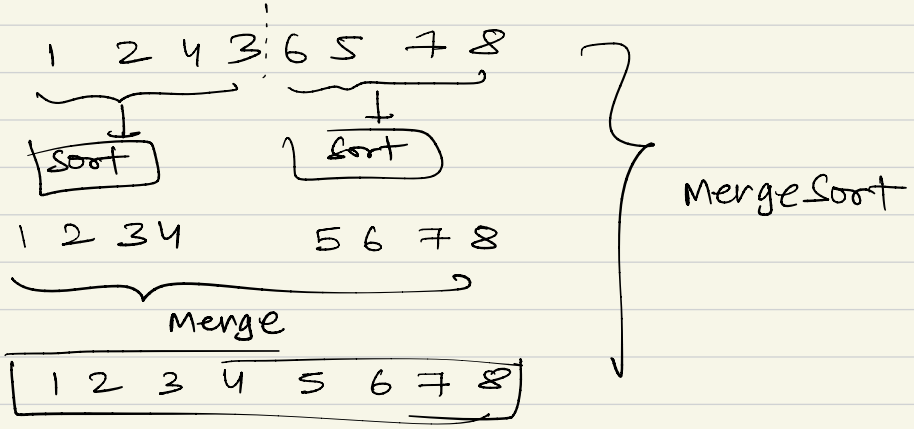


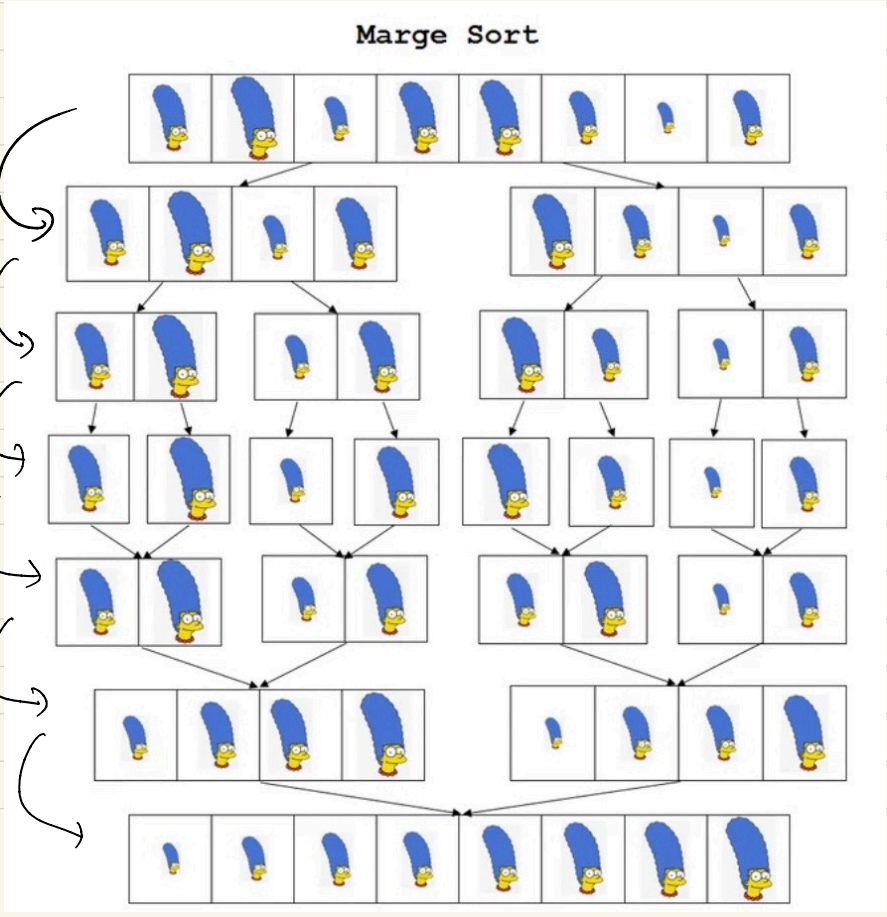
:

so on till you get



Merging 2 sorted arr is easy ⟶ this is GREAT





Code :-

① Define Merge function.

② Define

```
mergesort [Arr] :
    Array1 = Arr [0 : len(Arr)/2]
    Array 2 = Arr [len(Arr)/2 + 1 : ]
    return merge (Mergesort(Array1),
                  Mergesort(Array2))
```

Look @ code