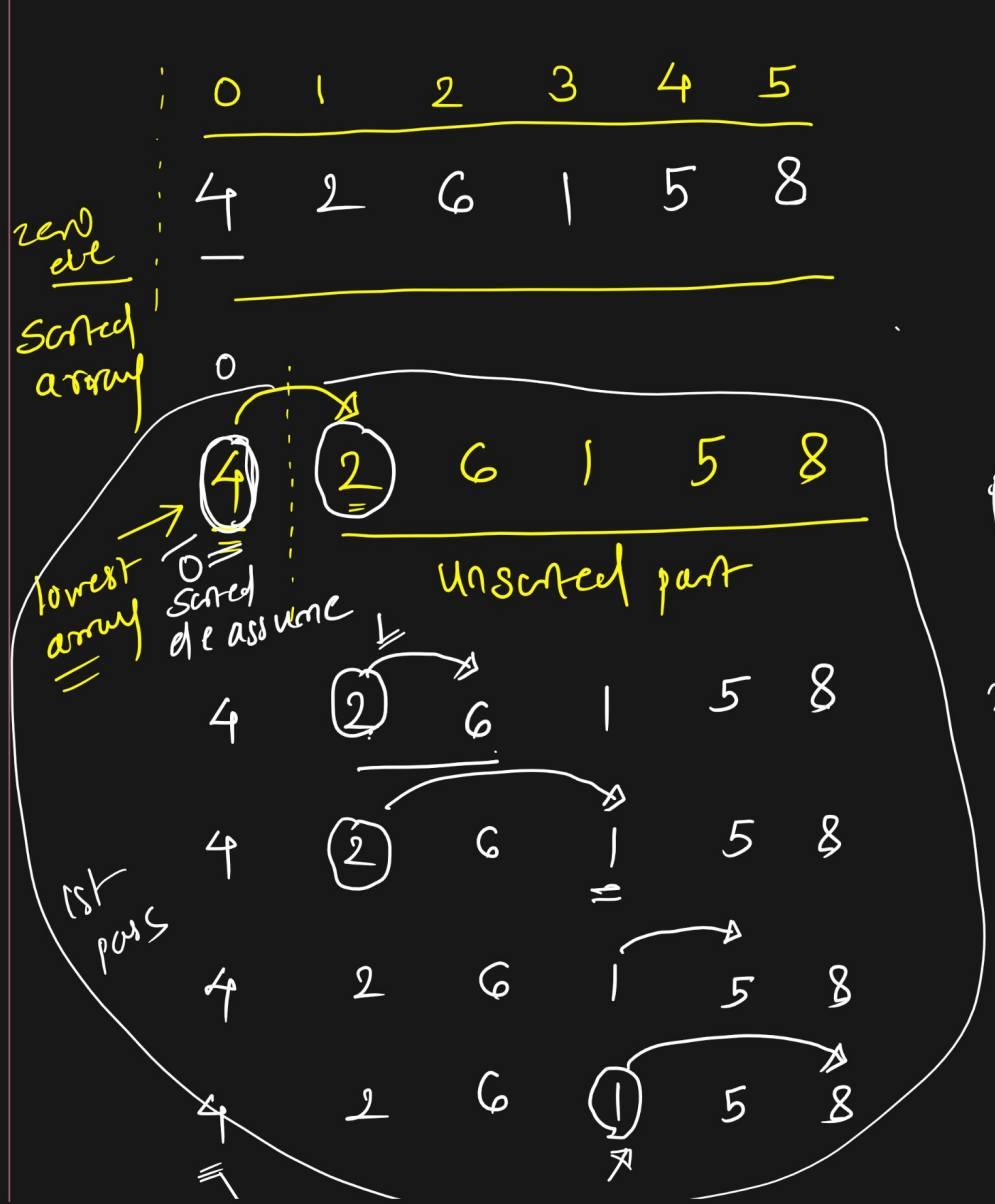
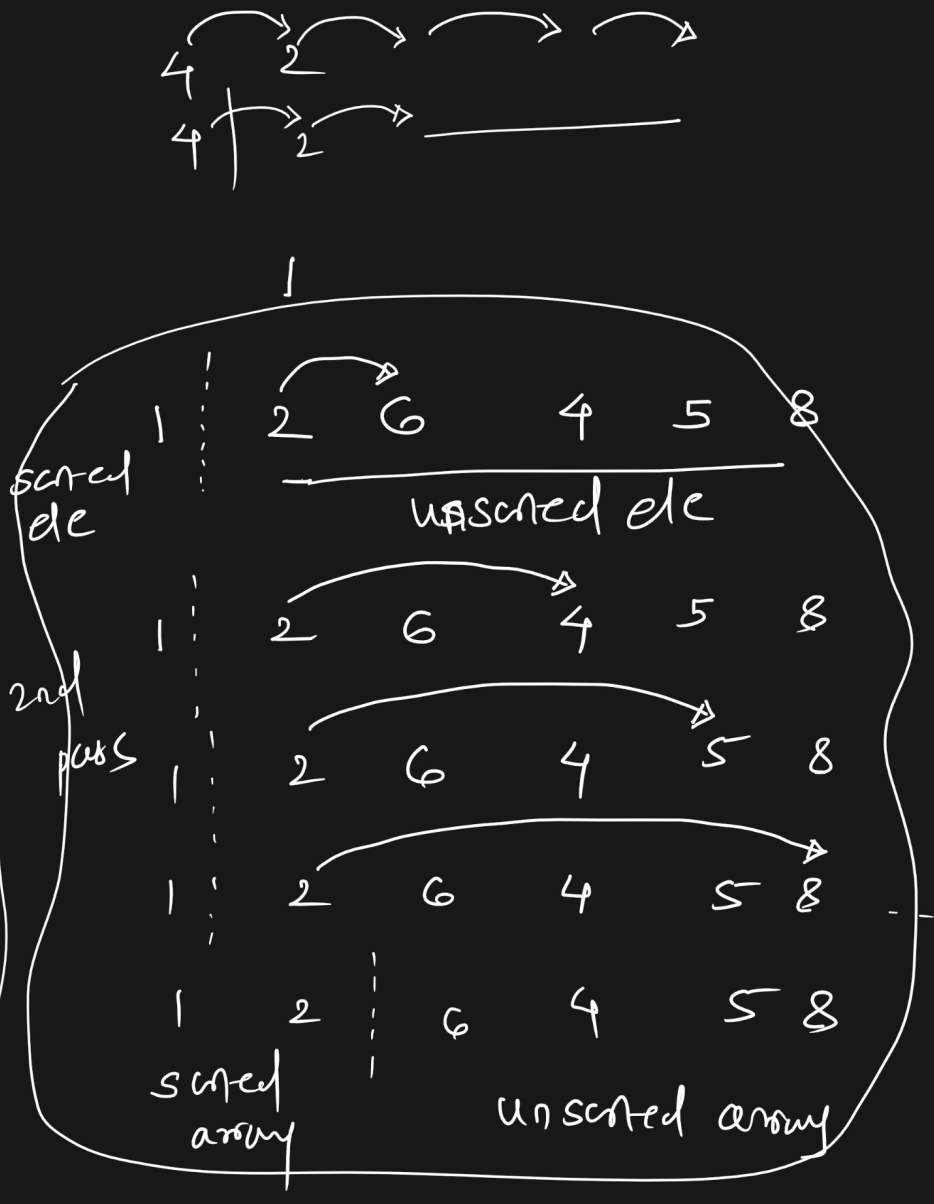
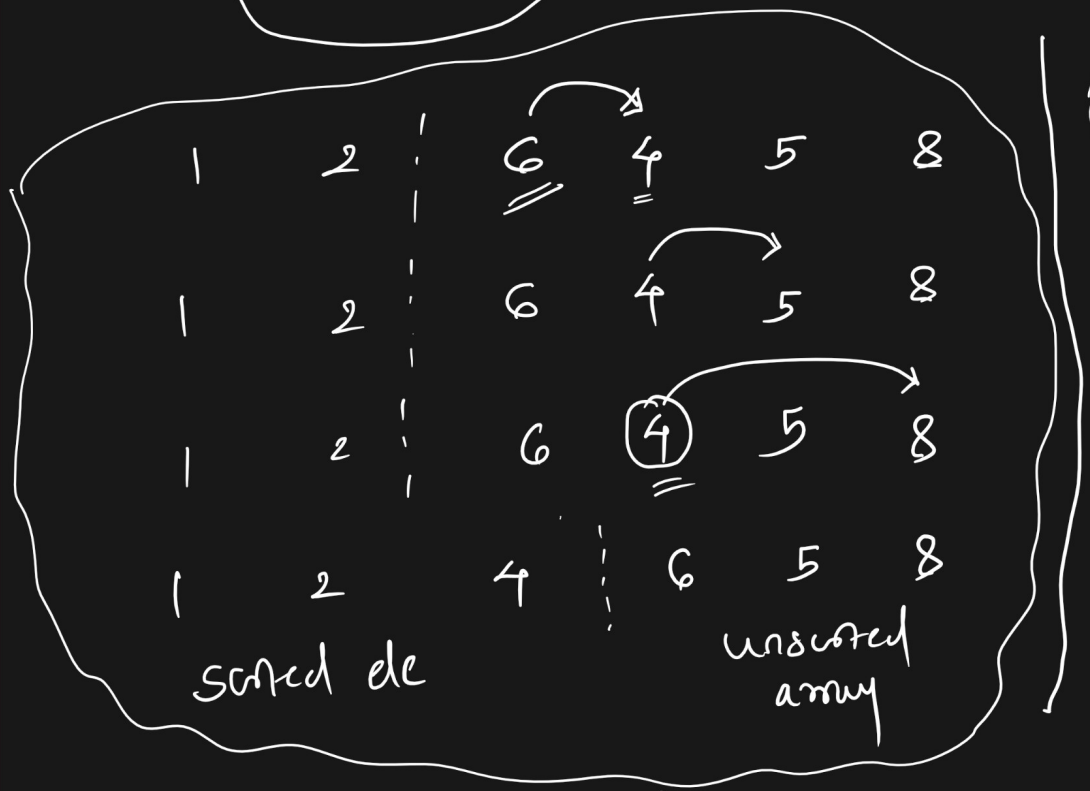
Day 8 :- DSA

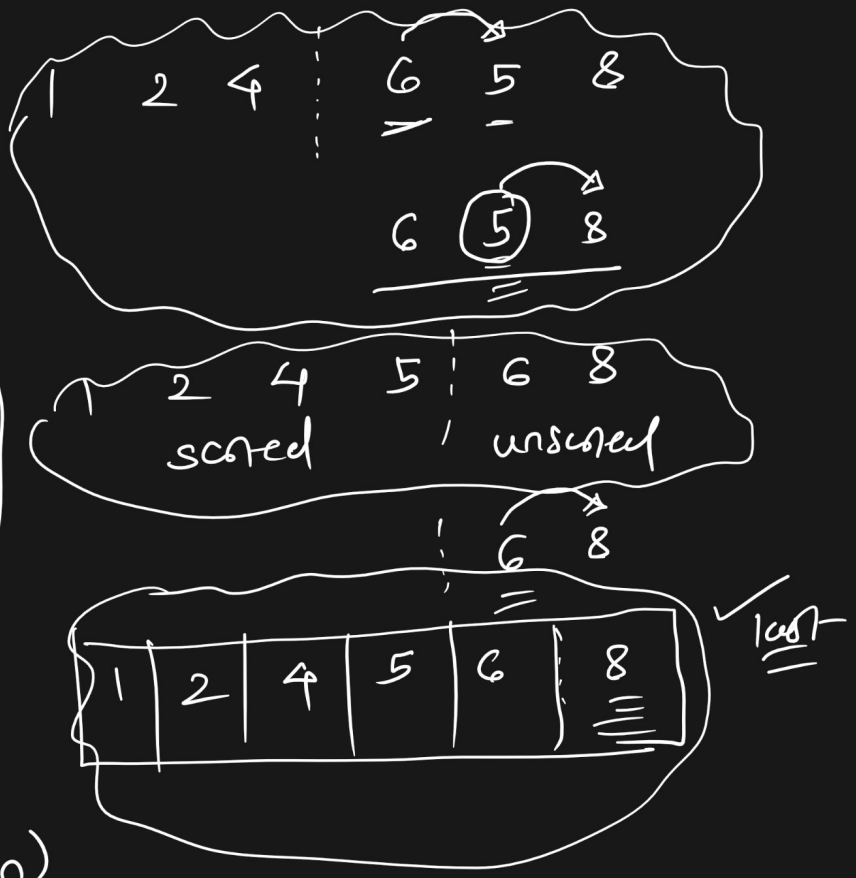
Sorting Algorithm :-

③ Selection Sort :-










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

Find out the ~~min~~ lowest ele index →

i=0 ; min = 0 → lowest min index

→ j = i+1 = 0+1 = 1 → 2 3 4 5

if(x) min = j-1 → lowest ele index

j=2 arr[min] = 2

arr[2] = 6

if(x) min = 1

```
        min = j;
    }
}
swap(arr[i], arr[min]);
```

④ Merge Sort :-

⑤ Quick Sort :-

left = 0, right = 6

$mid = \frac{l+r}{2} = \frac{0+6}{2} = 3$

Merge Sort

0	1	2	3	4	5	6
38	27	43	3	9	82	10

mid+1 for j

4+0 = 4
4+1 = 5
4+2 = 6

$n_1 = mid - left + 1$
$= 3 - 0 + 1 = 4$

left = 0, right = 6, mid = 3

$n_2 = right - mid$
$= 3$

l = 0, r = 3

$mid = \frac{0+3}{2} = 1$

0	1	2	3
38	27	43	3

l = 4 r = 6

$mid = \frac{4+6}{2} = \frac{10}{2} = 5$

4	5	6
9	82	10

l = 0, r = 1

$mid = \frac{0+1}{2} =$

0	1
38	27

38	27

27	38

k

2	3
43	3

$mid = \frac{2+3}{2} = \frac{5}{2}$

43	3

3	43

3	27	38	43

l = 4 mid = 5 r

9	82

$mid = \frac{4+5}{2} = 9/2 = 4$

9

82

9	82

6
10

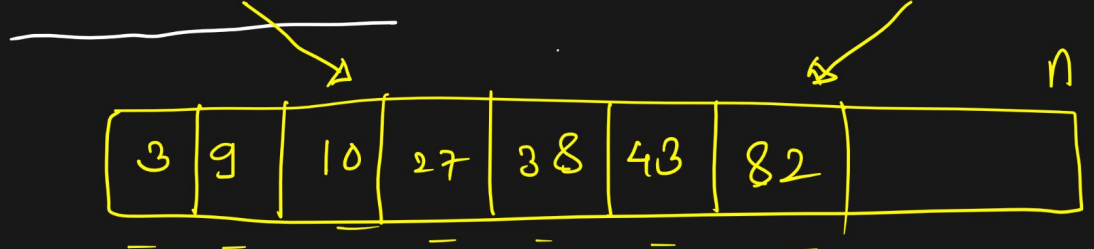
10

9	10	82

Time Complexity

$O(n(\log n))$

$n_1 = 5 - 4 + 1$
$= 1 + 1 = 2$



Merge Sort

sorted

↳ Divide & Conqure Technique

```
void mergeSort (int left, int right)
{
    while( left < right)
    {
        int mid = (left + right)/2;
        mergeSort( left, mid);
        mergeSort(mid+1, right);
        combine (left, mid, right);
    }
}
```

left → recursive Fun^n

divide

single → combine

```
void combine (int left, int mid, int right)
{
    int n1 = mid - left + 1;      // left array size
    int n2 = right - mid;         // right array size

    int L[] = new int[n1];        // left array creation
    int R[] = new int[n2];        // right array creation

    for(int i=0; i<n1; i++)       // initialize left array
    {
        L[i] = arr[left + i]
                   0      0
    }

    for(int j=0; j<n2; j++)       // initialize right array
    {
        R[j] = arr[mid+1+j]
    }
```

to represent index of left array

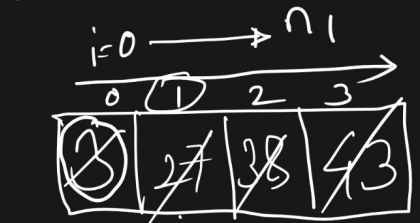
to rep. index of right array

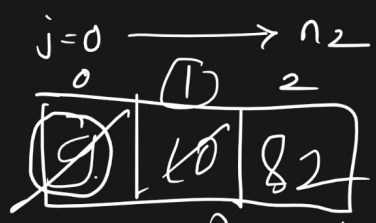
original array

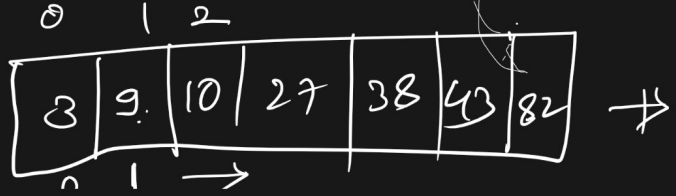
```
    int i=0, j=0, k=left;

    while((i<n1) && (j<n2))
    {
        if(L[i] < R[j])
        {
            arr[k] = L[i];
```

compare the elements







left & right array

```
        i++; k++;
    }
    else
    {
        arr[k] = R[j];
        k++;
        j++;
    }
}
```

```
while (i < n1)
{
    arr[k] = L[i];
    k++;
    i++;
}
```

```
while (j < n2)
{
    arr[k] = R[j];
    k++;
    j++;
}
```

```
}
```