Sprint-4 [Day-1]

Sorting Techniques

2ptr

→ phase's (1, 2, 3 . . .)

✓1. Bubble Sort

i/p arr ↳

0	1	2	3	4	5	6	7
7	1	9	2	4	6	5	3

↑

o/p arr ↳

0	1	2	3	4	5	6	7
1	2	3	4	5	6	7	9

✓ Sorted

↳ largest ele of Array should present @ last in the Array

(Array was sorted in the ↑ order)

$i = 0$

→ phase-1 :- → i

$n = 8$

	0	1	2	3	4	5	6	7
i/p arr ↳	7	1	9	2	4	6	5	3
	j	$j+1$						
arr ↳	1	7	9	2	4	6	5	3
arr ↳	1	7	9	2	4	6	5	3
				j	$j+1$			
arr ↳	1	7	2	9	4	6	5	3
arr ↳	1	7	2	4	9	6	5	3
arr ↳	1	7	2	4	6	9	5	3
arr ↳	1	7	2	4	6	5	9	3
							* ✓	
arr ↳	1	7	2	4	6	5	3	$\boxed{9}$
arr ↳								

→ ↑

$< n - 1$

$i = 0 : \quad \leq n - 2$

$j = 0 \; . \; . \; . \; . \; 6$

```
if ( a[j] > a[j+1] )
{
        swap

}
```

i = 1

Phase-2:-

→ o/p of phase-1

← i/p for phase-2

i = 1

j = 0 1 - - 5

	0	1	2	3	4	5	6	7
arr	1	7	2	4	6	5	3	9

arr	1	7	2	4	6	5	3	9

arr	1	2	7	4	6	5	3	9

arr	1	2	4	7	6	5	3	9

arr	1	2	4	6	7	5	3	9

arr	1	2	4	6	5	7	3	9

*

arr	1	2	4	6	5	3	7	9

arr	1	2	4	6	5	3	7	9

o/p

phase-3:-

i = 2

	0	1	2	3	4	5	6	7
arr ↳	1.	2	4	6	5	3	7	9

→ o/p of phase-2

← i/p for phase-3

✓

arr ↳	1	2	4	6	5	3	7	9

✓

i = 2

j = ~~0~~ ~~1~~ - - 4

arr ↳	1	2	4	6	5	3	7	9

✓

arr ↳	1	2	4	6	5	3	7	9

✓

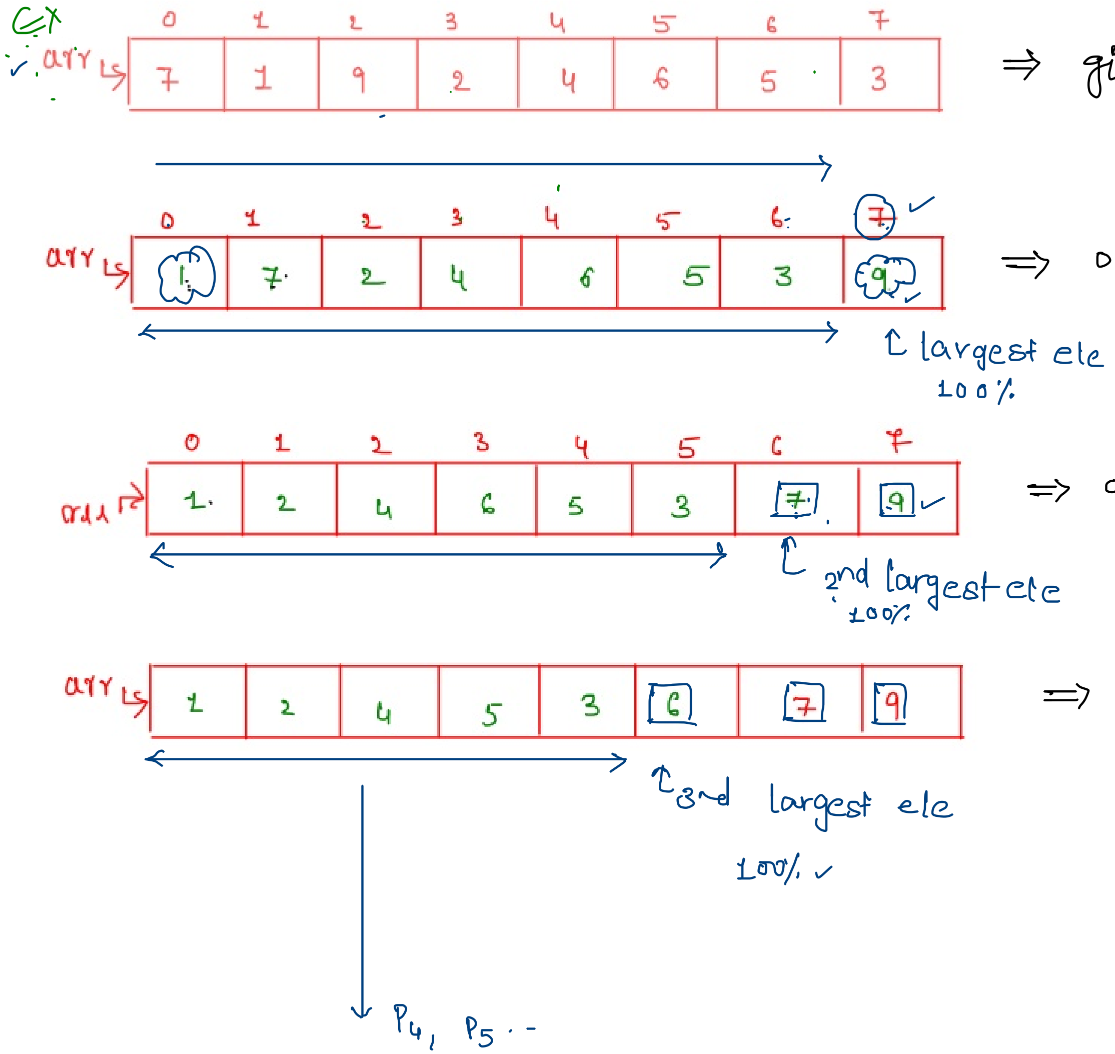
arr ↳	1	2	4	5	6	3	7	9

arr ↳	1	2	4	5	3	6	7	9

o/p

↳ i/p for phase-4

Ex



arr ↳

0	1	2	3	4	5	6	7
7	1	9	2	4	6	5	3

⇒ given i/p

$n=8$ (fixed)

arr ↳

0	1	2	3	4	5	6	7
1	7	2	4	6	5	3	9

↑ largest ele 100%

⇒ o/p of P1

✓ $i=0$ $n=8$

$j = 0\ 1\ 2 \cdots 6$

↑ $8-0-2$

arr ↳

0	1	2	3	4	5	6	7
1	2	4	6	5	3	7	9 ✓

↑ 2nd largest ele 100%

⇒ o/p of P2

✓ $i=1$ $n=8$

$j = 0\ 1 \cdots 5$

↑ $8-1-2$

arr ↳

1	2	4	5	3	6	7	9

↑ 3rd largest ele 100% ✓

⇒ o/p of P3

✓ $i=2$ $n=8$

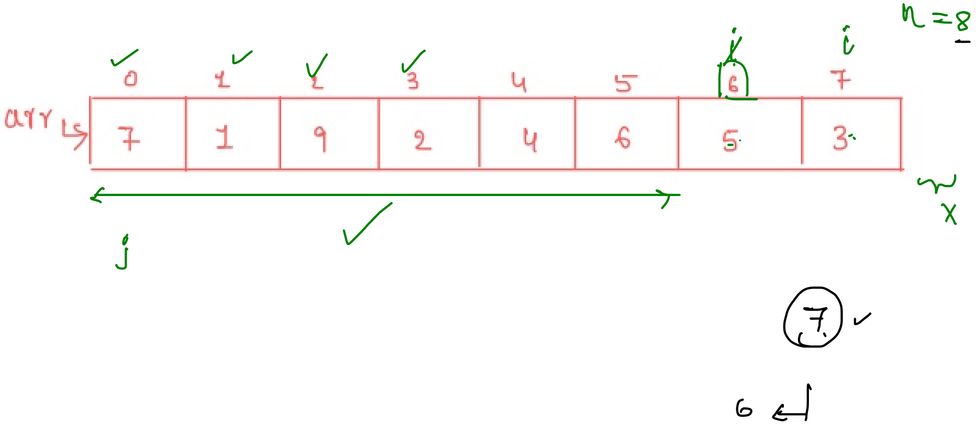
$j = 0 \cdots 4$

↑ $8-2-2$

$j = 0$ to $n-i-2$ including

↓ P4, P5 ..

```
for(j=0; j ≤ n-i-2; j++)
{
}
```



0	1	2	3	4	5	6	7
7	1	9	2	4	6	5	3

1-min

```
function bubbleSort(arr,n)
{
        for(i=0;i<=n-2;i++)        ⇒ n
        {
                for(j=0;j<=n-i-2;j++)        ⇒ n
                {
                        if(arr[j]>arr[j+1])
                        {
                                temp=arr[j];
                                arr[j]=arr[j+1];
                                arr[j+1]=temp
                        }
                }
        }
        return arr;
}
```

⇒ $O(n^2)$ ✓

↑ order

Best case i/p

	0	1	2	3	4	5	6	7
arr ↳	10	20	30	40	50	60	70	80

worst case i/p

	0	1	2	3	4	5	6	7
arr ↳	80	70	60	50	40	30	20	10

$\Rightarrow O(n^2)$

i=0

n=8 → 7 phses ((1, 2 [illegible])

	0	1	2	3	4	5	6	7
arr ↳	10	20	30	40	50	60	70	80

P-1 i=0 ⇒ 7 Comp + 0 swap ✓ arr sorted

i=1 ⇒ 6 " + 0 "

i=2 ⇒ 5 " + 0 "

⋮

1

1+2+ . . . +7

$1+1+ \dots + n-1 \Rightarrow \frac{n(n-1)}{2} \Rightarrow O(n^2)$

```
function bubbleSort(arr,n)
{

    for(i=0;i<=n-2;i++)
    {
        isSwapped=0;
        for(j=0;j<=n-i-2;j++) → n ✓
        {
            if(arr[j]>arr[j+1])
            {
                isSwapped=1;
                temp=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp
            }
        }
        if(isSwapped==0)
        {
            break;
        }
    }
    return arr;
```

Best case i/p : O(n) ✓

worst case i/p : O(n²)

U1

BS

↳ $O(n^2)$

↓ ✓

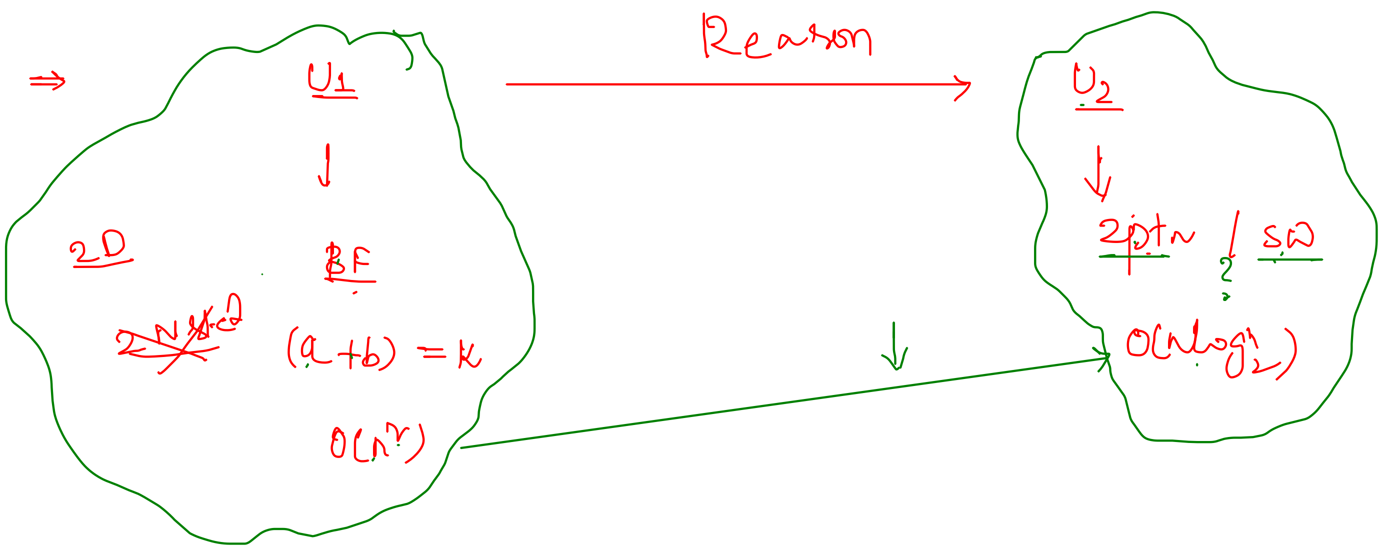
2ptr

arr.sort() ⇒ $O(n\log_2 n)$ ✓

U3

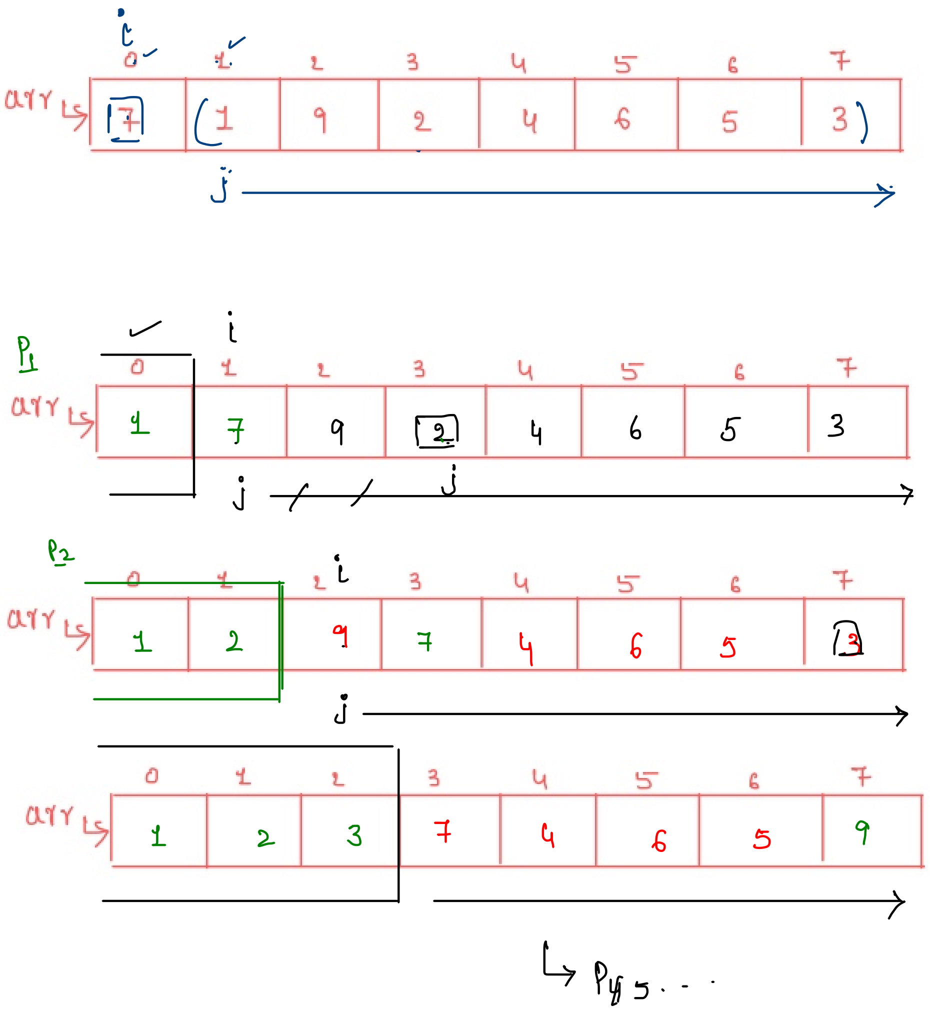
→ Merge_sort

Quick_sort



res = ~~7~~ 1 ✓

2) Selection sort



```
function minElement(arr,n)
{
        let res=arr[0]
        for(let i=1;i<n;i++)
        {
                if(arr[i]<res)
                {
                        res=arr[i]
                }
        }
        return res
}
```