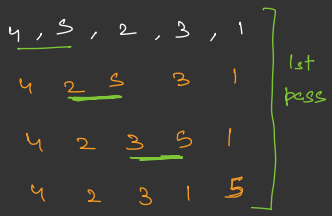
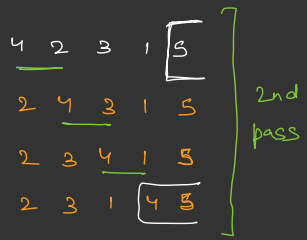
Sorting

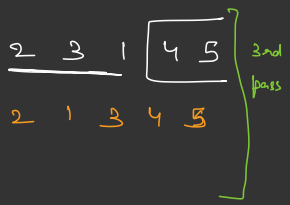
4, 1, 5, 2, 3 ⇒ 1, 2, 3, 4, 5

Sorting

Bubble Sort



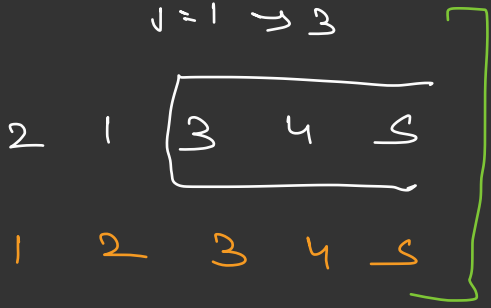




j = 1 → 4

j = 1 → 3

j = 1 → 2



n sized array

n - 1 passes

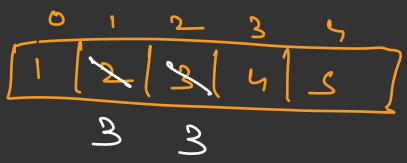
complete array is sorted

i = 1 j = 2

int a = arr[i]

arr[i] = arr[j]; ←

arr[j] = a ←



a = 2

```
for (int pass = 1; pass <= n-1; pass++)
{
    for (int j = 1; j <= arr.length - pass; j++)
    {
        if(arr[j] < arr[j-1]) {
            swap(j, j-1)
        }
    }
}
```

n-1 + n-2 + n-3 ... 3,2,1

1 → n-1

$\frac{n \times (n-1)}{2}$

$\frac{n^2}{2} - \frac{n}{2}$

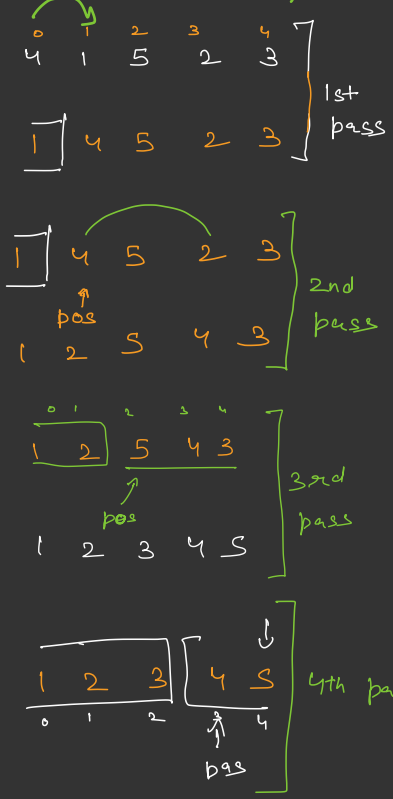
$\Rightarrow O(n^2)$

Input → O(n)

auxilary → O(1)

best case → Ω(n)

Selection Sort



min idx = ~~0~~ ~~1~~ ~~1~~ ~~3~~ ~~2~~ ~~3~~ ~~4~~ 3

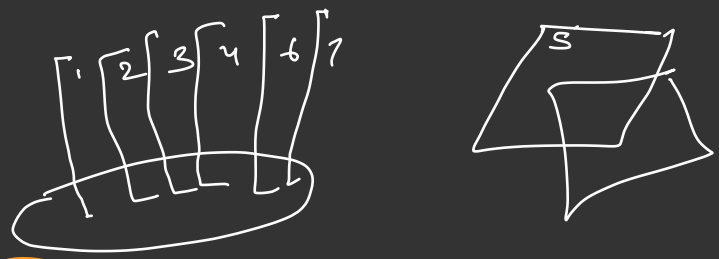
swap (pas, minIdx)

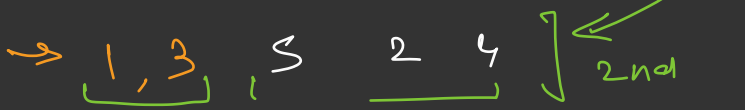
Ls → n

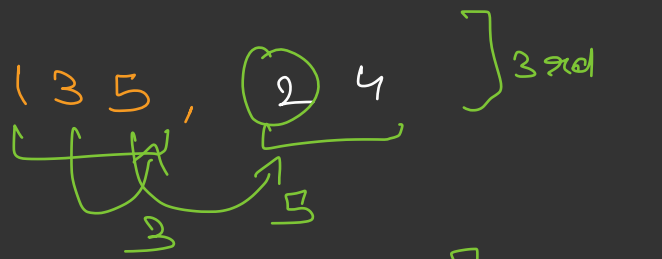
passes = n − 1 ⇒ TC = O(n²)

Auxilary space ⇒ O(1)

Insertion Sort





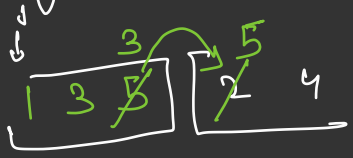


1 2 3 4 5

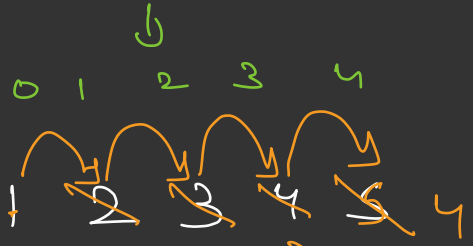
key = 2

key = 5 | 1 3 | 5 2 4

arr[j+1] = key;



key = 2



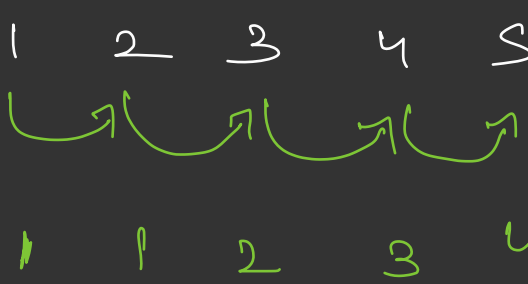
k = arr[e-i]

k = 5

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

→ presume last

shifting → end → j>=0

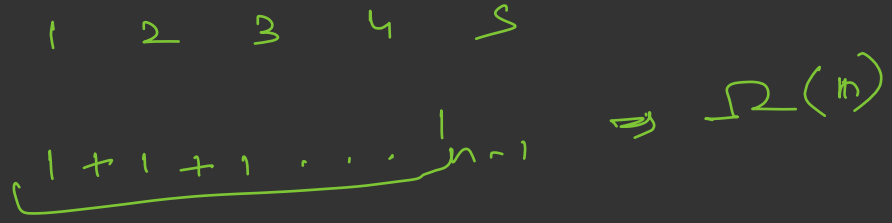


Time Complexity

4 5 3 2 1

$1 + 2 + 3 + 4 \ldots\ldots n-1$

$\rightarrow \frac{n \times (n-1)}{2} \Rightarrow \frac{n^2}{2} - \frac{n}{2}$ $O(n^2)$



inplace Sorting

→ Array is sorted in input array

Space → Auxiliary space → O(1)

Ques Merge 2 sorted arrays

res

(1)

Create new array (n1 + n2) $(n_1 + n_2)$

Copy both the arrays

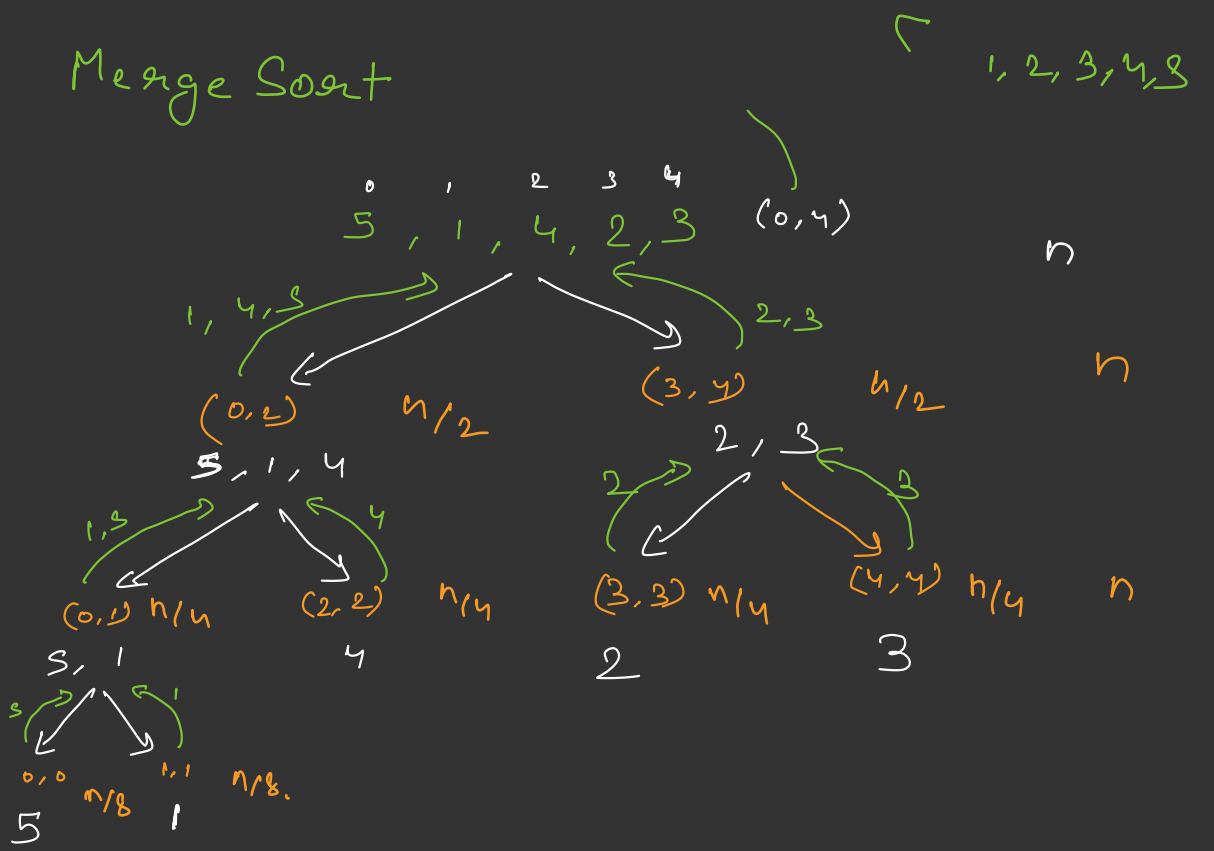
Sort the resutant array n log n

$n = n1 + n2$

Time ⇒ $n + n\log n$ ⇒ $O(n \log n)$

(2)

Merge Sort



$$\frac{n}{2^k} = 1$$

$$\Rightarrow k = \log_2(n)$$

$k\,n$ $\qquad$ $\underbrace{n + n + n \ldots -}_{k \text{ step}}$

Time $\Rightarrow O(n \log n)$

Space $\rightarrow$ Auxiliary Space $\Rightarrow O(n)$ $\rightarrow$ not inplace

Ques Partition the array

5	4	2	6	(3)	1

pivot = 3

⇒ 2, 1, 3, 5, 4, 6

2, 3, 1, 6, 4, 5

i

j

<= 3 0 to j-1

> 3 j to i-1

Unknow i to n-1

> 3 i++

<= 3

swap(i, j),

i++

j++.

j = 3 4

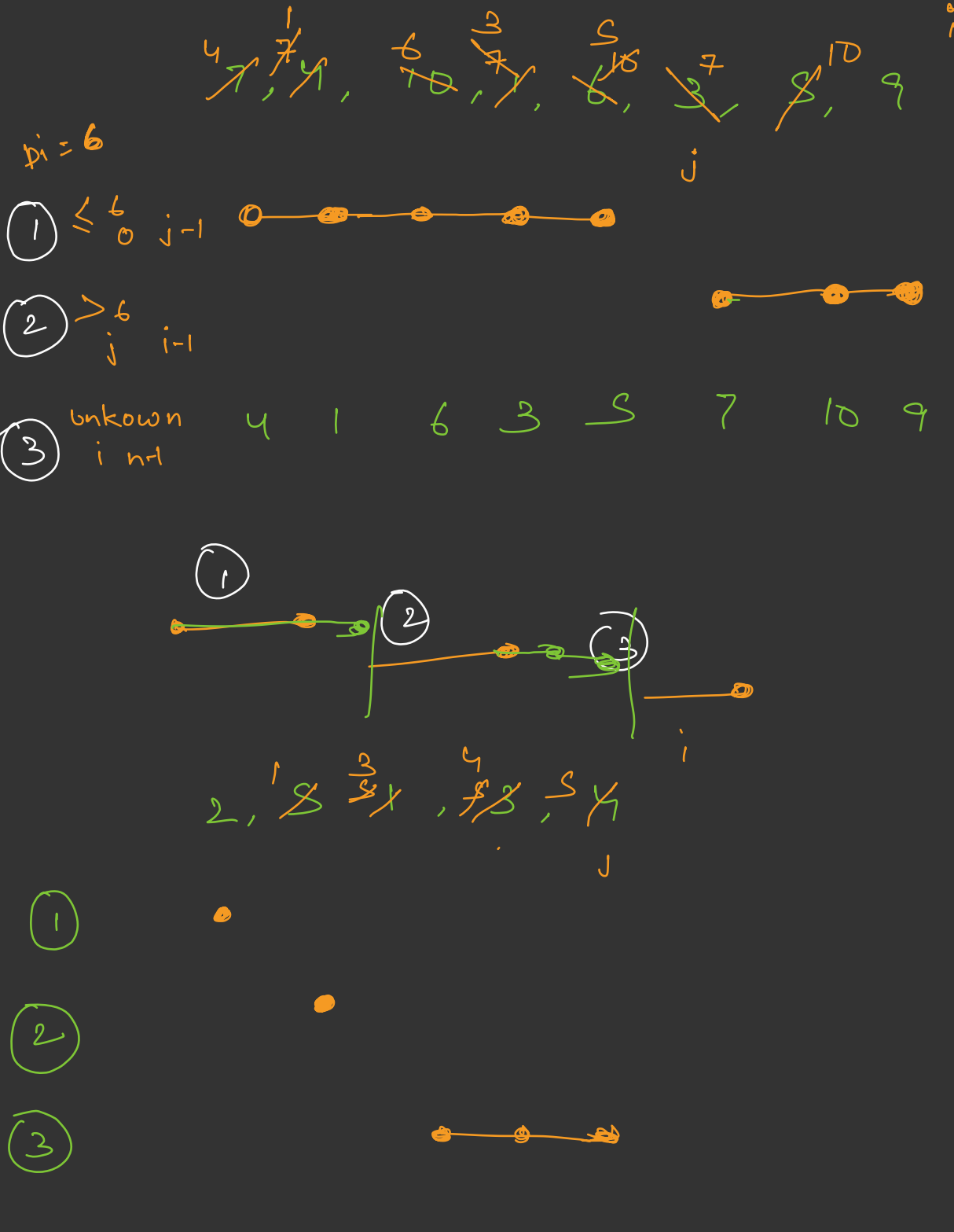
i = 6 7

3 - 5

2

4 - 6

2


4 7 1 6 10 5 6 3 8 9
pi = 6
j
1
≤ 6
0 j-1
2
> 6
j i-1
3
unkown
i n-1
4 1 6 3 5 7 10 9
1
2
3
i
2 8 3 1 4 3 5 4
j
1
2
3

Quick Sort

0 1 2 3 4 5 6 7

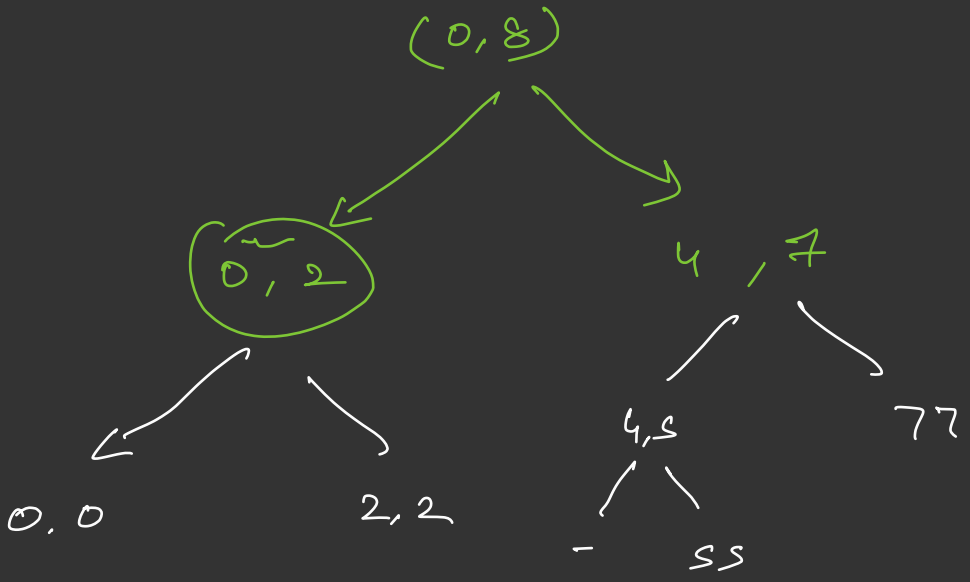
5, 8, 3, 1, 6, 2, 7, 4

0 1 2 3 4 5 6 7

~~3, 1, 2~~ 4 ~~5, 8, 6, 7~~

1, 2, 3 4 5, 6, 7, 8

1, 2, 3



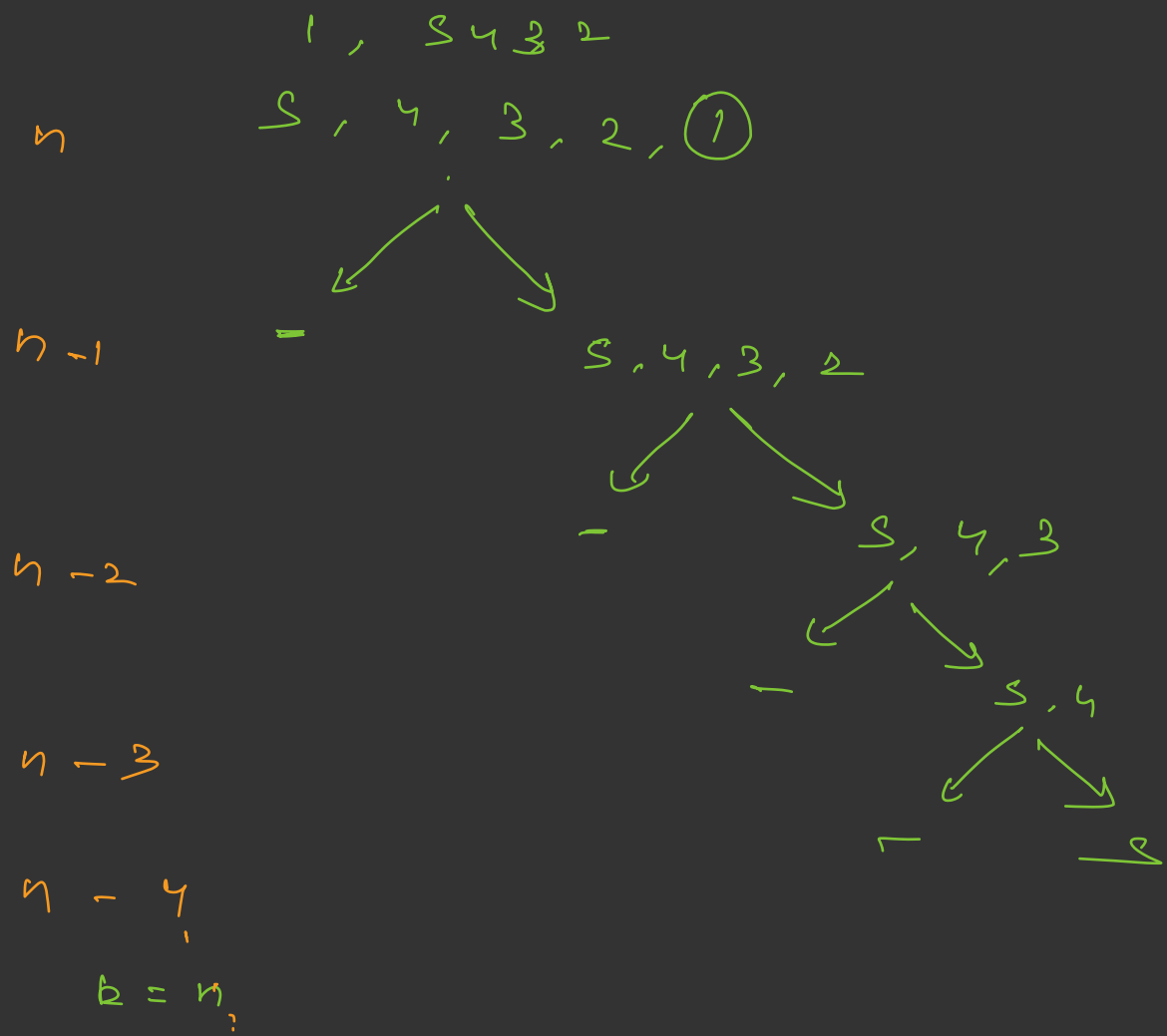
$n_0 = \frac{n}{2^k} = 1$

$n + n + n \ldots n$ (k times)

$\Rightarrow n \times k$

$k = \log n$

$O(n \log n)$


1, 5 4 3 2
n
5, 4, 3, 2, 1
n-1
5, 4, 3, 2
n-2
5, 4, 3
n-3
5, 4
n-4
k = n;
1
1 + 2 + 3 + ... n-1
⇒ n x (n-1) / 2 ⇒ O(n²)

3 2 1 5 4

→ Choose pivot randomly

Time → $O(n \log n)$

Auxilary space → $O(1)$

Inplace sorting

Arrays.sort(arr);