Day-7 DSA

* Find out the missing number from 1 to N-1 size of array.

[2, 4, 1, 5] N-1

⇓

Ans :- 3

element start from 1 to N

⇓

One of the ele is missing

⇒ [2, 4, 1, 5] $\xrightarrow{\text{sort}}$ [1, 2, 4, 5]

⇓

for loop ⇒ i = 1 to N

⇓

Somewhere

$i \neq arr[i]$

Brute Force

↳ complex logic

↳ Time Comp ↑

→ sort → $O(n^2)$ / $O(n\log n)$

⇓

```
for(int i=1; i<=N; i++)
{
    if (i!=arr[i])
        return i;
}
```

→ $O(n)$

i

1 = arr[1] = 1 ✓

2 = arr[2] = 2 ✓

3 = arr[3] = 4 ✗ ⇒ i = 3 missing term

Approach - 2) :-

sum of first N numbers = 1+2+3+4+5 = A.P. = $\frac{n(n+1)}{2}$ = $\frac{5(5+1)}{2}$ = $\frac{5\times6}{2}$

1 to 5 N = 5 = 15

total_sum = 15

N = 5

$[\cancel{2}, \cancel{4}, \cancel{1}, \cancel{5}]$ ⇒

arr

i = 0; s_array = 0+2 = 2

i = 1; s_array = 2+4 = 6

i = 2; s_array = 6+1 = 7

i = 3; s_array = 7+5 = $\boxed{12}$

```
int sum_array = 0 ⇒
for( int i = 0; i< arr.length; i++)
{
    sum_array += arr[i];
}
```

→ $\boxed{\text{sum\_array} = 12}$

Time Complexity

n times

O(n)

return total_sum - sum_array;

$\underset{15}{\text{total\_sum}} - \underset{12}{\text{sum\_array}}$

$\boxed{3}$

N = 10 array.size = N-1 = 9

[6, 7, 2, 3, 5, 1, 8, 9, 10] ⇒ Total_Sum = sum of first 10 numbers

$= \frac{n \times (n+1)}{2} = \frac{10 \times 11}{2} = \frac{110}{2} = 55$

⇓ int sum_array = 0

```
for( ____ )
{
    ____
}
```

51

return $\frac{55 - 51}{4}$;

* Sorting :- Sort the elements of array either in ascending or descending order.

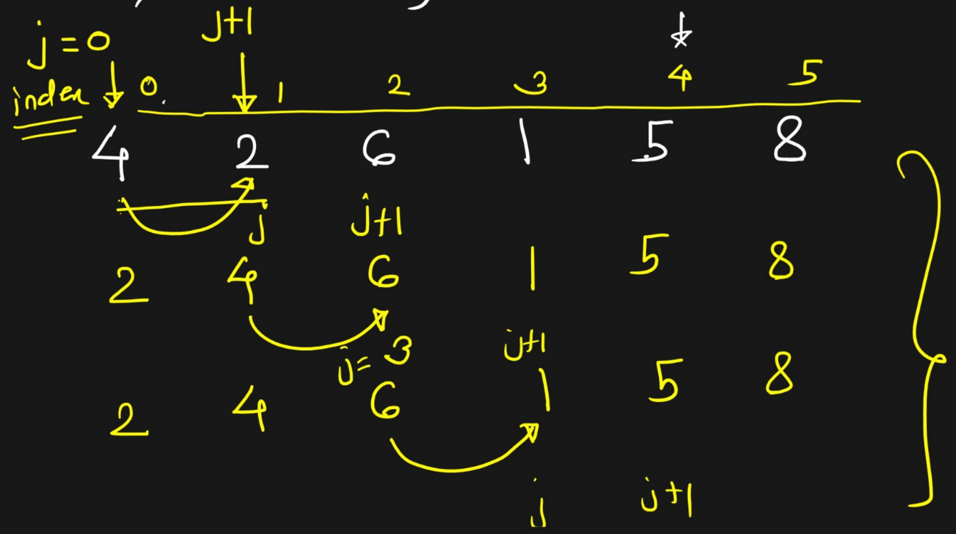
Algorithm :-
1) Bubble Sort
2) Insertion Sort
3) Selection Sort
4) Merge Sort
5) Quick Sort

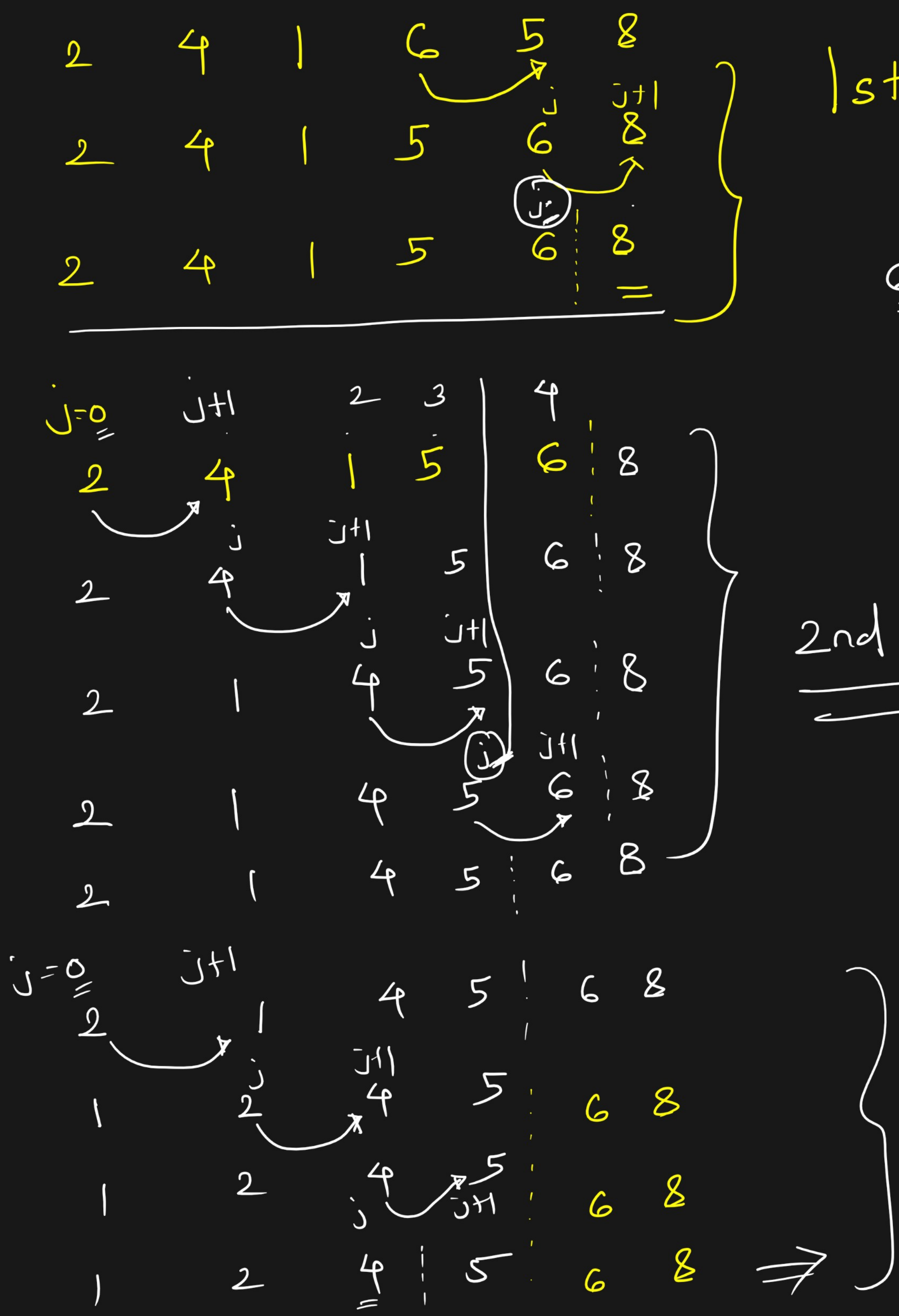
① Bubble Sort :-

⇒ Comparison based algorithm

⇒ adjancent pair comparison

⇒ If they are not in proper seqⁿ then we need swap.





1st Pass = 1st largest ele at the end

$6 \Rightarrow j = 0 \text{ to } \cancel{5}4 \quad = 0 \text{ to } N-1-$

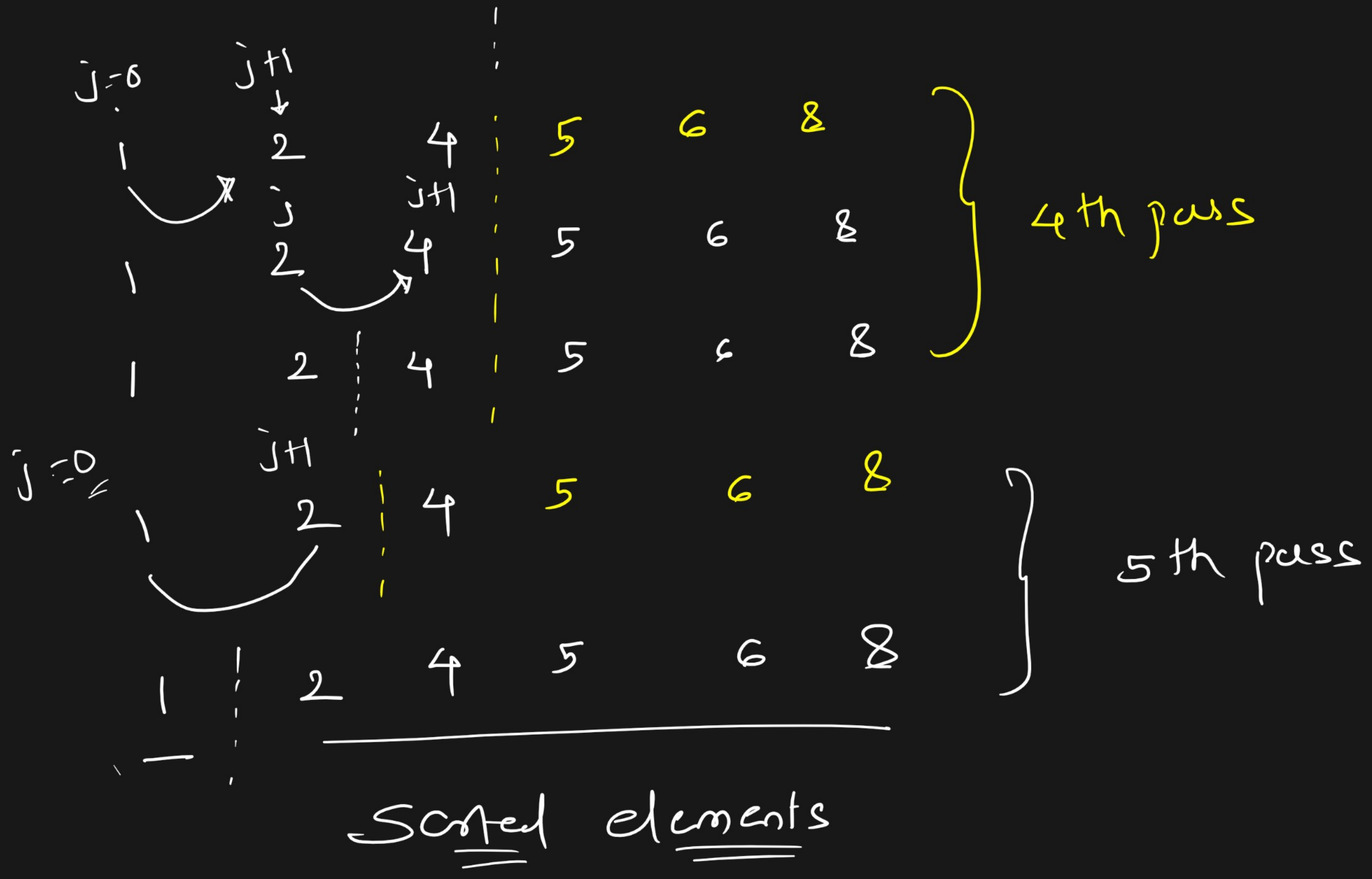
$0 \text{ to } < N-1$

$0 \text{ to } < \overset{6-1}{5}$

$0 \text{ to } 4$

2nd pass ⇒ 2nd largest ele

3rd pass



6-1=5

for (int i = 0; i < N-1 ; i++) → 0 to 4 = 5 times

{

N-1-~~0~~ 1 = 6-1-~~0~~ 1 = ~~5~~ 4   6 to 8

< ~~5~~ 4 = 4

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

1 ✓ 0

2 ✓ 1

3 ✓ 2

No of passes

4 ✓ 3

5 ✓ 4

n=6

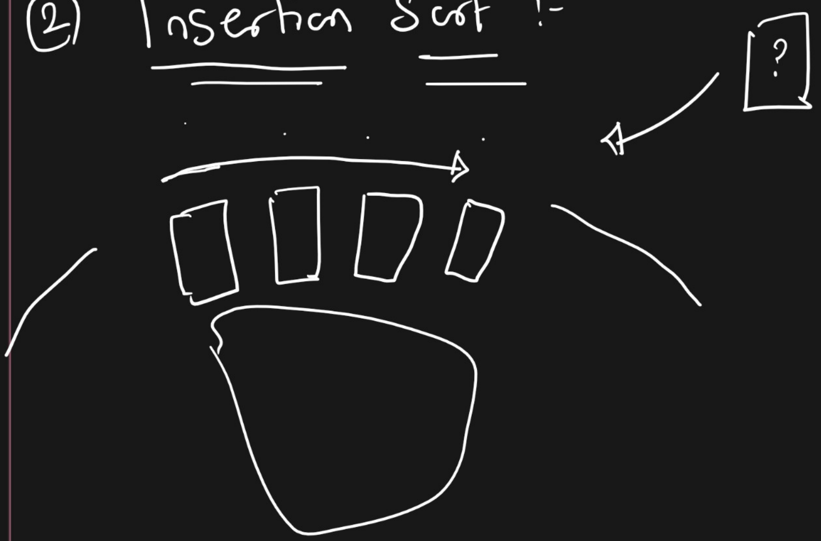
$< n-1 = < 6-1 = < 5$

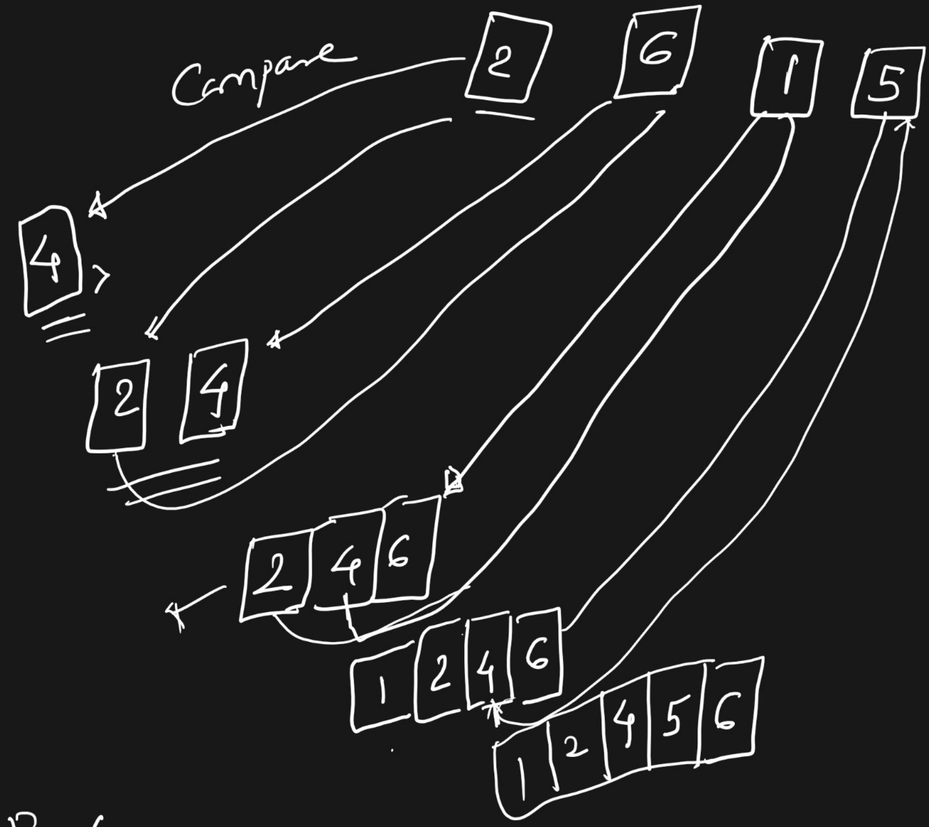
i =	0	1	2	3	4
j =	0 to n-1-i < 6-1-0 < 5 5 times	0 to n-1-i 0 to 6-1-1 ~~5~~4 times	3	2	1

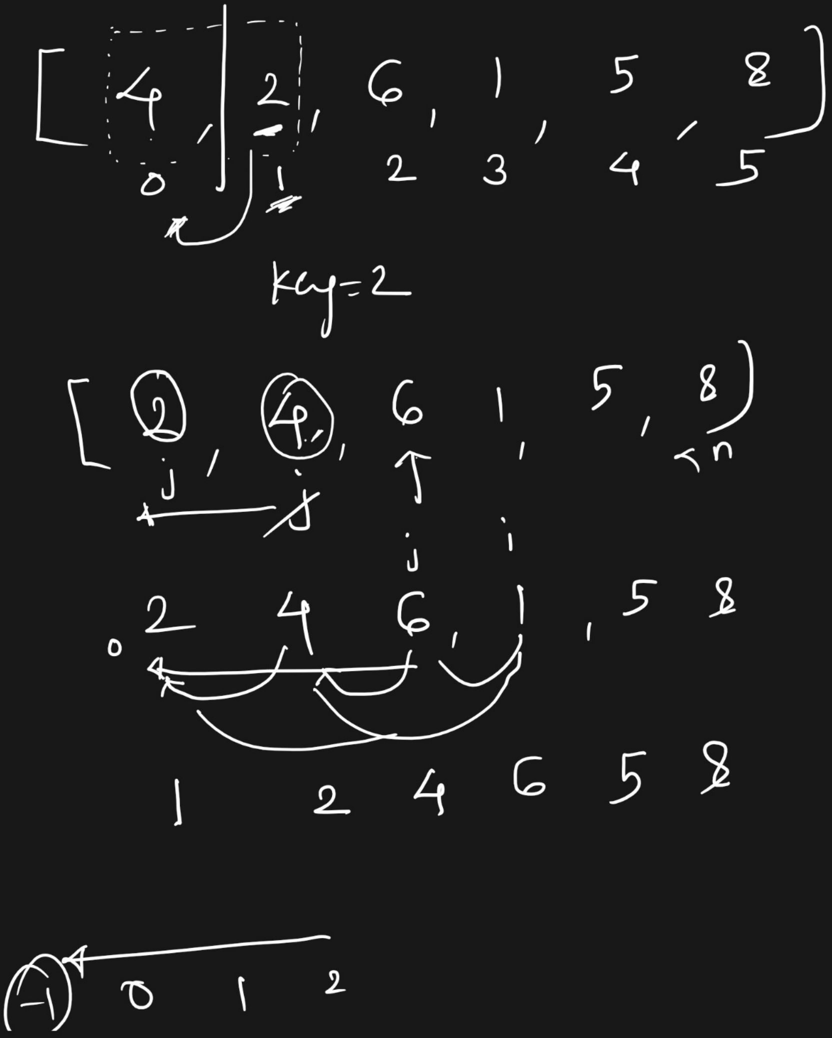
$$5 + 4 + 3 + 2 + 1 = \text{sumofn} = \boxed{\frac{n(n+1)}{2}} = \frac{n^2}{2} + \frac{n}{2}$$

$O(n^2)$

② Insertion Sort :-

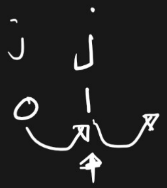






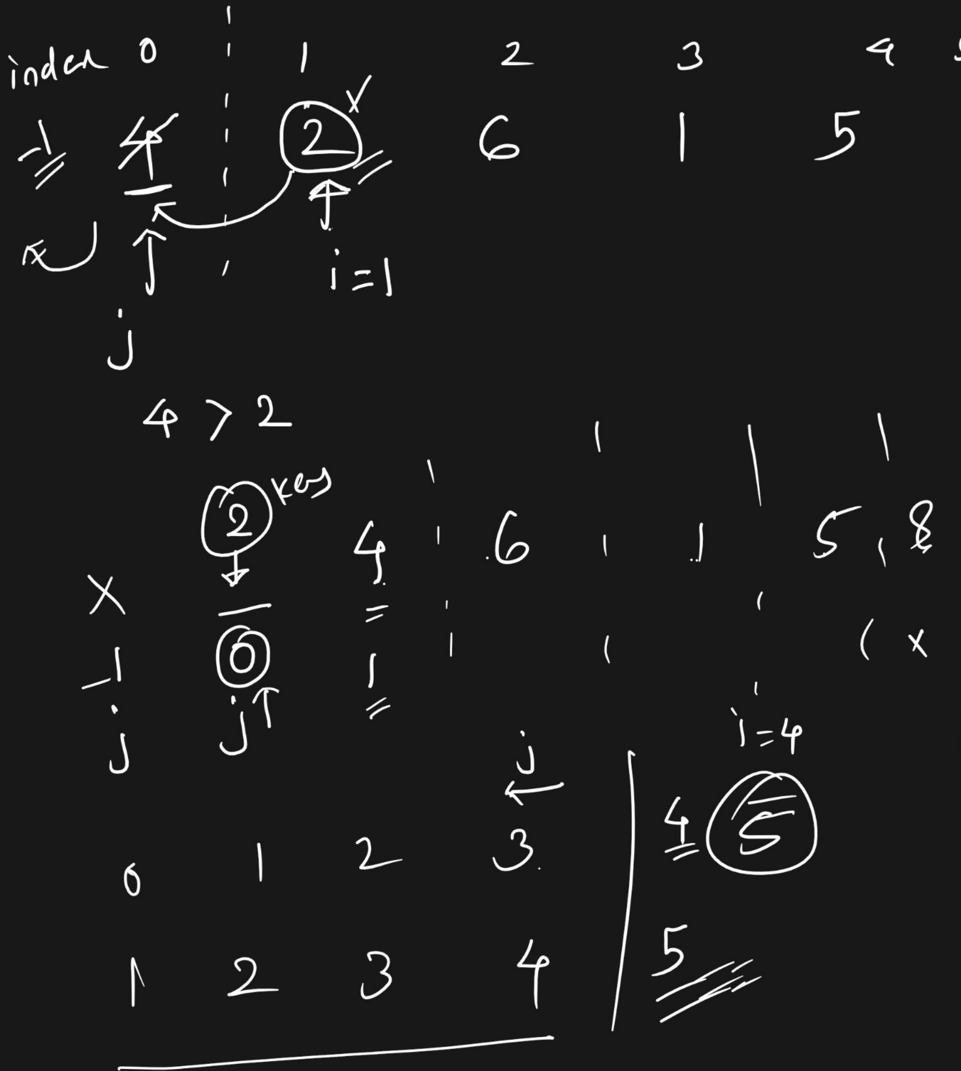
```
for(int i=1; i<n; i++)
{
    int key = arr[i];
    int j = i-1;

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



=

```
}   arr[j+1] = key;
```



```
for(int i=1 ; i<n; i++)
{
    int key = arr[i];
    int j = i-1;

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

    arr[j+1] = key;
}
```

n times

n time

1 + 2+ 3+ 4 + 5 = n

Time Complexity = $O(n^2)$

H.W. Write a code to print following patterns

1)

```
*
* *
* * *
* * * *
```

2)

```
   *
  * *
 * * *
* * * *
```

3)

```
      *
    * *
  * * *
* * * *
```

4)

```
* * * *
 * * *
  * *
   *
```

5)

```
* * * *
* * *
* *
*
```