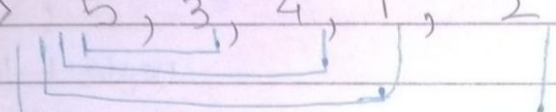


* Insertion Sort *

Array \Rightarrow 5, 3, 4, 1, 2 [Partially sorting the array]



For every index, put that index element at the correct index of LHS

i.e. 1st pass $\rightarrow i = 0$ this will be sorted

[5, 3], 4, 1, 2

\downarrow

3, 5, 4, 1, 2

2nd pass $\rightarrow i = 1$ this will be sorted

[3, 5, 4], 1, 2

\downarrow

3, 4, 5, 1, 2

3rd pass $\rightarrow i = 2$ this will be sorted

[3, 4, 5, 1], 2

\downarrow

1, 3, 4, 5, 2

4th pass $\rightarrow i = 3$ this will be sorted

[1, 3, 4, 5, 2]

[1, 2, 3, 4, 5]

array is sorted !!

* To understand what every 'i' is doing :

Outer loop :

5, 3, 4, 1, 2

$[i]$ j

sort arry till
index 1

pass ① $\rightarrow 0$

— 11 —

pass ② $\rightarrow 1$

index 2

— 11 —

pass ③ $\rightarrow 2$

3

— 11 —

pass ④ $\rightarrow 3$

" 4

i.e. i will run from 0 to $(n-2)$

★ Working —

5, 3, 4, 1, 2

$i < (n-2)$

$j > 0$

$3 < 5 \rightarrow \text{swap} \downarrow$

3, 5, 4, 1, 2

3, 5, 4, 1, 2

$4 < 5 \rightarrow \text{swap} \downarrow$

3, 4, 5, 1, 2

Now since $3 < 4$ already sorted
when element j is not smaller
than element $(j-1)$ break the
loop because the previous (LHS)
side array is already sorted.

3, 4, 5, 1, 2

1 < 5 swap ↓

3, 4, 1, 5, 2

1 < 4 swap ↓

3, 1, 4, 5, 2

1 < 3 swap ↓

1, 3, 4, 5, 2

1, 3, 4, 5, 2

2 < 5 swap ↓

1, 3, 4, 2, 5

2 < 4 swap ↓

1, 3, 2, 4, 5

2 < 3 swap ↓

already sorted → 1, 2, 3, 4, 5

so break.

$i < (n-2)$

$j > 0$

→ 2

3

→ 3

4

Now if we take $i = 4$ then $j = 5$ which is index out of bound.

∴ we take $i < (n-2)$ where n is length of array

Time Complexity :

1) Worst case $\Rightarrow O(n^2)$
(descending sorted)

2) Best case $\Rightarrow O(n)$
(already sorted)

Why to use insertion sort?

★ **Adaptive** : steps get reduced if array is sorted
[i.e. no of swaps are reduced as compared to bubble sort]

★ **Stable sorting Algorithm**

★ **Used for smaller values of n** : works good when
array is partially sorted,

↓
it takes part in hybrid sorting
algorithm

Ex.

```
static void insertionSort(int[] arr) {  
    for (int i = 0; i < arr.length - 1; i++) {  
        for (int j = i + 1; j > 0; j--) {  
            if (arr[j] < arr[j - 1]) {  
                swap(arr, j, j - 1);  
            }  
            else {  
                break; }  
        }  
    }  
}
```

see swap funcⁿ in selection sort.