Page No.
Date :

* Insertion Sort *

Array => 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 → i = 0 — this will be sorted

[5, 3], 4, 1, 2

↓

3, 5, 4, 1, 2

2nd pass → i = 1 — this will be sorted

[3, 5, 4], 1, 2

↓

3, 4, 5, 1, 2

3rd pass → i = 2 — this will be sorted

[3, 4, 5, 1], 2

↓

1, 3, 4, 5, 2

4th pass → i = 3 — this will be sorted

[1, 3, 4, 5, 2]

[1, 2, 3, 4, 5] array is sorted !!

Page No.
Date :

* To understand what every 'i' is doing:

Outer loop:

5, 3, 4, 1, 2

	$\boxed{i}$	j
sort arry till index 1 — pass ①	→ 0	
— " — index 2 — pass ②	→ 1	
— " — 3 — pass ③	→ 2	
— " — " 4 — Pass ④	→ 3	

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

* working —

	$i < (n-2)$	$j > 0$
0 1 2 3 4 5, 3, 4, 1, 2 j 3 < 5 → swap ↓ 3, 5, 4, 1, 2	← 0	1
3, 5, 4, 1, 2 j 4 < 5 → swap ↓ 3, 4, 5, 1, 2	← 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.

Page No.
Date :

	$i<(n-2)$	$j>0$
$3, 4, 5, \overset{j}{1}, 2$		
$1<5$ swap ↓		
$3, 4, \underset{j}{1}, 5, 2$	→ 2	3
$1<4$ swap ↓		
$3, \underset{j}{1}, 4, 5, 2$		
$1<3$ swap ↓		
$1, 3, 4, 5, 2.$		
$1, 3, 4, 5, \overset{j}{2}$		
$2<5$ swap ↓		
$1, 3, 4, \underset{j}{2}, 5.$	→ 3	4
$2<4$ swap ↓		
$1, 3, \underset{j}{2}, 4, 5$		
$2<3$ swap ↓		
already sorted so break. → $1, 2, 3, 4, 5$		

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

Page No.
Date :

Time Complexity :

1) worst case => $O(n^2)$
(descending sorted)

2) Best case => $O(n)$
(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 fun in selection sort.