Insertion Sort Algorithm.

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

eg:-

0	1	2	3	4
5	3	4	1	2

i=0
j=i+1=1

0	1	2	3	4
3	5	4	1	2

i=1
j=2

0	1	2	3	4
3	4	5	1	2

i=2
j=3

0	1	2	3	4
1	3	4	5	2

i=3
j=4

1	2	3	4	5

array is sorted!!

∴ i will run from 0 to n-2.

j > 0
i ≤ N-2

* WORKING

```
            j
        0   1   2   3   4
i=0     5   3   4   1   2
j=1

                j
i=1     3   5   4   1   2
j=2
            j
        3   4   5   1   2

                    j
i=2     3   4   5   1   2
j=3
                j
        3   4   1   5   2
            j
        3   1   4   5   2
        j
        1   3   4   5   2

                        j
i=3     1   3   4   5   2
j=4
        1   3   4   2   5
        1   3   2   4   5
            j
        1   2   3   4   5
```

when ele[j] is not smaller than ele[j-1] break.

Page

* COMPLEXITY :-

- Worst Case - $O(N^2)$
(desc sorted)

eg :- 5, 4, 3, 2, 1

$\therefore$ 1 comp, 2 comp, 3 comp . . . N-1 comparisons

$\therefore$ Sum = $\frac{N(N-1)}{2} = \frac{N^2 - N}{2}$ ← Total No of comparison.

$\therefore$ $O(N^2)$

- Best Case - $O(N)$
array is already sorted.

eg:- 1, 2, 3, 4, 5

N-1 comparisons

* why use insertion sort?

* Adaptive :- steps get reduced if array is sorted.
[i.e no. of swaps are reduced]

* Stable Algorithm

* used for smaller values of n : works good when array is partially sorted

takes place in hybrid algo.