*** Insertion Sort ***

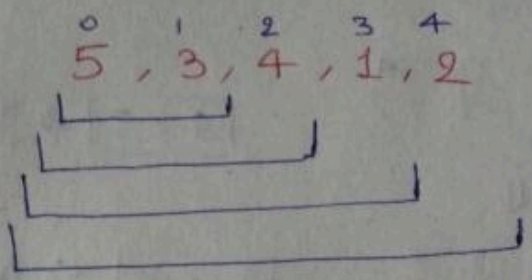
* Insertion Sort

⇒ A sorting algorithm in which the elements are transferred one at a time to its correct position.

⇒ In this method, we are partially sorting the array. Means, sort the array parts-by-parts.

Example:

idea



- 1st sort till index 1.
- Then sort till index 2.
- Then sort till index 3.
- Then sort till index 4.

*NOTE: • For Every index:
Put that index element at the correct index of Left-Hand-Side (LHS).

- With every pass, Left-hand array is getting sorted

i.e

For, i=0 ⇒ Pass No. 1 ⇒ Sort array till index 1

For, i = 1 ⇒ Pass No. 2 ⇒ Sort array till index 2.

⋮

and so on....

NOTE i will run from 0 till n-2. Becoz after that J+1 give index out of bound.
where, n = length of array.

Example

0 1 2 3 4
5 , 3 , 4 , 1 , 2

* Pass 1

i=0, j=1

i j
5 , 3 , 4 , 1 , 2

3 < 5 ⇒ Yes ⇒ Swap

⇓

j
3 , 5 , 4 , 1 , 2

Now, j=0
So, we don't check
J > 0 → always

* Pass 2

i=1, j=2

i j
3 , 5 , 4 , 1 , 2

4 < 5 ⇒ Yes ⇒ Swap

⇓

j
3 , 4 , 5 , 1 , 2

Now, j = 1

4 < 3 ⇒ No

↓

Break the loop

* NOTE * when element j not smaller than element j-1, break the loop.

* Reason *
This LHS array is already Sorted

* Pass 3

i=2, j=3

i j
3 , 4 , 5 , 1 , 2

1 < 5 ⇒ Yes ⇒ Swap

⇓

j
3 , 4 , 1 , 5 , 2

Now, j = 2

1 < 4 ⇒ Yes ⇒ Swap

⇓

j
3 , 1 , 4 , 5 , 2

Now, j = 1

1 < 3 ⇒ Yes ⇒ Swap

⇓

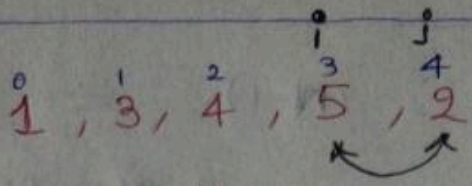
j
1 , 3 , 4 , 5 , 2

Now, j = 0

J has to be > 0
Hence, Break

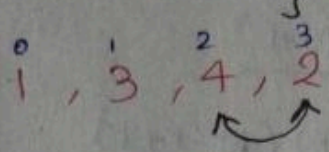
* **Pass 4 :**

$i = 3$, $j = 4$,



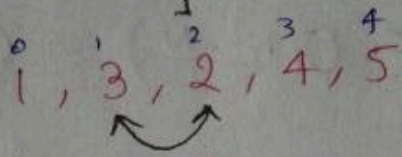
$2 < 5$ ⇒ Yes ⇒ Swap

Now, $j = 3$



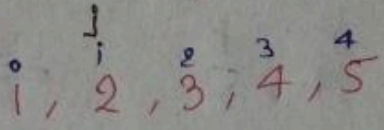
$2 < 4$ ⇒ Yes ⇒ Swap

Now, $j = 2$



$2 < 3$ ⇒ Yes ⇒ Swap

Now, $j = 1$



$2 < 1$ ⇒ No

↓

Break the loop

Now, if we take $i = 4$, then j will be index out of bound.

Hence, $\boxed{i < n-2}$

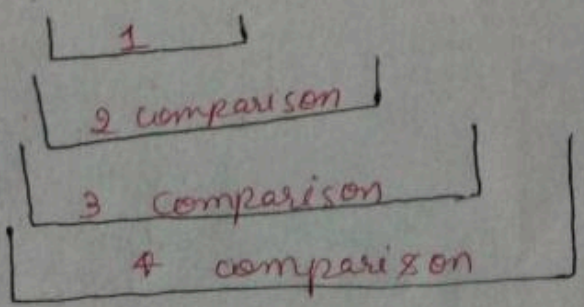
* **ANS** ⇒ 1, 2, 3, 4, 5

* **Complexity of Insertion Sort :**

* **Worst Case** $= O(n^2)$, n = no. of elements

Example : descending sorted

5, 4, 3, 2, 1



1, 2, 3, ------ (N-1) ← comparisons

Total comparisons = Sum of N nos.

$$= \frac{N(N+1)}{2}$$

$$= \frac{(N-1)(N-1+1)}{2} = \frac{N(N-1)}{2}$$

$$= O\left(\frac{N^2-N}{2}\right) = O(N^2)$$

* Best case : when the array is already sorted

i j i j
1, 2, 3, 4, 5 ⇒ (N-1) comparison

$= O(N)$

* Why Use Insertion Sort ?

→ Adaptive : Steps get reduced if array is sorted.
i.e, No. of swaps reduced as compared to bubble sort.

→ It is Stable.

→ Used for smaller values of N.
works good when array is partially sorted.
That's why, it takes part in Hybrid Sorting algorithms