Insertion Sort

$$\text{Array} = \begin{bmatrix} 5 & 3 & 4 & 1 & 2 \end{bmatrix}$$
$$j: \; 0 \quad 1 \quad 2 \quad 3 \quad 4$$

1st pass ($i = 0$)

$$\begin{bmatrix} 5 & 3 & 4 & 1 & 2 \end{bmatrix}$$
$$j: \; \underbrace{0 \quad 1} \quad 2 \quad 3 \quad 4$$

Sort elements at $j = 0, 1$

Result $\begin{bmatrix} 3 & 5 & 4 & 1 & 2 \end{bmatrix}$

2nd pass ($i = 1$)

$$\begin{bmatrix} 3 & 5 & 4 & 1 & 2 \end{bmatrix}$$
$$j: \; \underbrace{0 \quad 1 \quad 2} \quad 3 \quad 4$$

Sort element at $j = 0, 1, 2$

Result :- $\begin{bmatrix} 3 & 4 & 5 & 1 & 2 \end{bmatrix}$

3rd pass ($i = 2$)

$$\begin{bmatrix} 3 & 4 & 5 & 1 & 2 \end{bmatrix}$$
$$j: \; \underbrace{0 \quad 1 \quad 2 \quad 3} \quad 4$$

sort elements $j = 0, 1, 2, 3$

Result :- $\begin{bmatrix} 1 & 3 & 4 & 5 & 2 \end{bmatrix}$

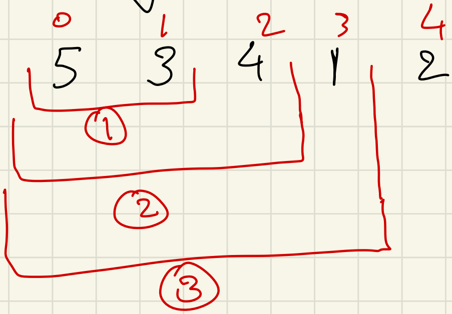
4th pass $i = 3$

$$\begin{bmatrix} 1 & 3 & 4 & 5 & 2 \end{bmatrix}$$
$$j: \; \underbrace{0 \quad 1 \quad 2 \quad 3 \quad 4}$$

sort element $j = 0, 1, 2, 3, 4$

Result :- $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$

Internal Algorithm



Range 0 → (n−1)

(j > 0)

		i	j
Sort array till index 1	⇐ Pass1 ⇐	0	1
Sort array till index 2	⇐ Pass1 ⇐	1	2
Sort array till index 3	⇐ Pass1 ⇐	2	3
Sort array till index 4	⇐ Pass1 ⇐	3	4

i will run from 0 to n − 2
(n = length of the array)

Whenever Array[j] is not smaller than Array[j − 1] ⇒ Break the inner loop

And also $\boxed{j > 0}$

Example Array = [5 3 4 1 2]

For i = 2, 3rd pass

$$\begin{array}{rccccccc} & & & & & j & & \\ \text{Initial} = & [& 3 & 4 & 5 & 1 & 2 &] \end{array}$$

[At this point j = 3]

$$\begin{array}{rccccccc} & & & & j & & & \\ = & [& 3 & 4 & 1 & 5 & 2 &] \end{array}$$

$$\begin{array}{rccccccc} & & & j & & & & \\ = & [& 3 & 1 & 4 & 5 & 2 &] \end{array}$$

$$\begin{array}{rccccccc} & & j & & & & & \\ = & [& 1 & 3 & 4 & 5 & 2 &] \\ & & 0 & 1 & 2 & 3 & 4 & \end{array}$$

for i = 2, till index 3 it is sorted

For i = 3, 4th pass

$$\begin{array}{rccccccc} & & & & & & j & \\ \text{Initial} = & [& 1 & 3 & 4 & 5 & 2 &] \end{array}$$

[j = 4]

$$\begin{array}{rccccccc} = & [& 1 & 3 & 4 & 2 & 5 &] \\ & & & & & j & & \end{array}$$

$$\begin{array}{rccccccc} = & [& 1 & 3 & 2 & 4 & 5 &] \\ & & & & j & & & \end{array}$$

$$\begin{array}{rccccccc} = & [& 1 & 2 & 3 & 4 & 5 &] \\ & & & j & & & & \end{array}$$

Since $j > 0$ and $A[j] < A[j-1]$

[Break the loop]

For $i = 4$, $j < n-2 \Rightarrow 4 < 3$ [Break out loop]

Complexity

Worst Case : $O(N^2)$ [descending array]

N = no of element

Ex 5, 4, 3, 2 1

First pass no of Comparisons = 1 (5, 4)

2nd pass Comparisons = 2 (5, 4, 3)

$\hookrightarrow$ (4, 3), (5, 3)

3rd pass Comparisons = 3 (5, 4, 3, 2)

(5, 2) (4, 2) (3, 2)

4th pass : Comparison = 4

So total Comparison = 1 + 2 + 3 + 4

$$= 1 + 2 + \cdots (N-1)$$

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

$$= N(N-1)/2$$

$$= (N^2 - N)/2$$

$$= O(N^2)$$

Best Case : Array already sorted

Ex 1, 2, 3, 4, 5

First pass : no of Comparison = 1 [1, 2]

2nd pass : no of Comparison = 1 [1, 2, 3]
only [3, 2)

3rd pass : no of Comparison = 1 [1, 2, 3, 4)
only (3, 4)

4th pass : no of Comparison = 1 [1, 2, 3, 4, 5]
only (4, 5)

Total number of Comparison $= 1 + 1 + 1 + 1$

$= 4$

$= (N - 1)$

$= O(N)$

Why use insertion sort

① Adaptive, steps get reduced if array is sorted
(no of swaps are reduced as compared to bubble sort)

② Stable = Yes [refer to bubble sort]

③ used for smaller values of N
⇒ works good when part of the array is partially sorted

This is the reason, it takes part in hybrid sorting algorithms

[bubble sort
insertion sort
Selection sort ⇒ do not work well with large arrays]

[merge sort
quick sort ⇒ work well with large arrays/datasets]

hybrid sorting algorithms

Ex:- Combination of merge sort + insertion sort