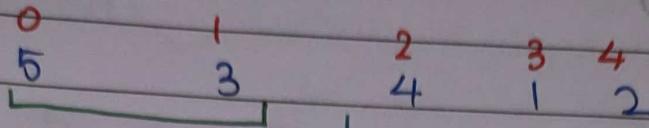


Insertion Sort Algorithm.

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

eg:-



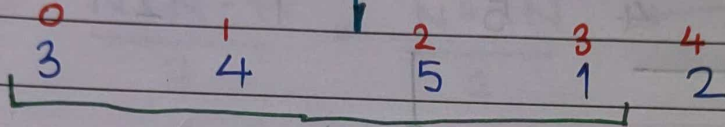
$i=0$

$j=i+1=1$



$i=1$

$j=2$



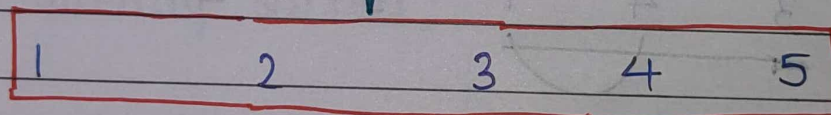
$i=2$

$j=3$



$i=3$

$j=4$



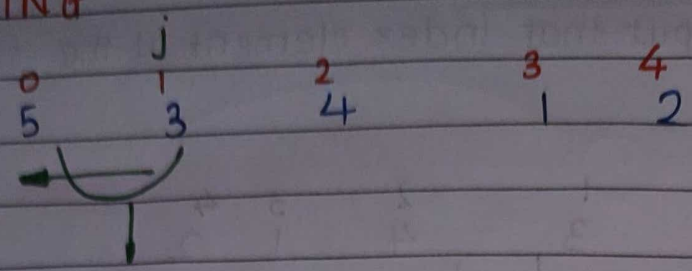
array is sorted!!

$\therefore i$ will run from 0 to $n-2$.

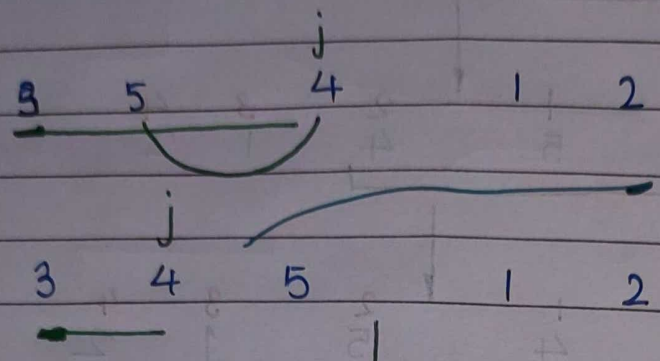
$j > 0$
 $i \leq N-2$

* WORKING

$i=0$
 $j=1$

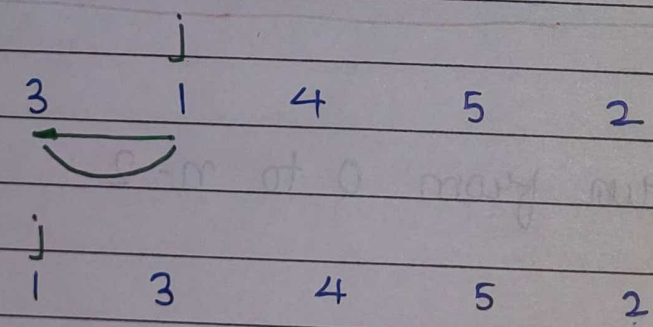
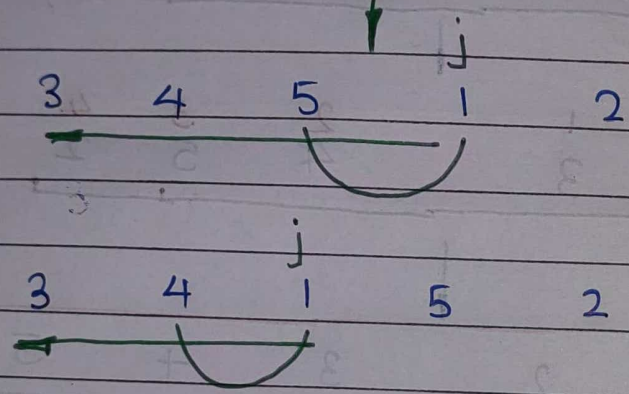


$i=1$
 $j=2$

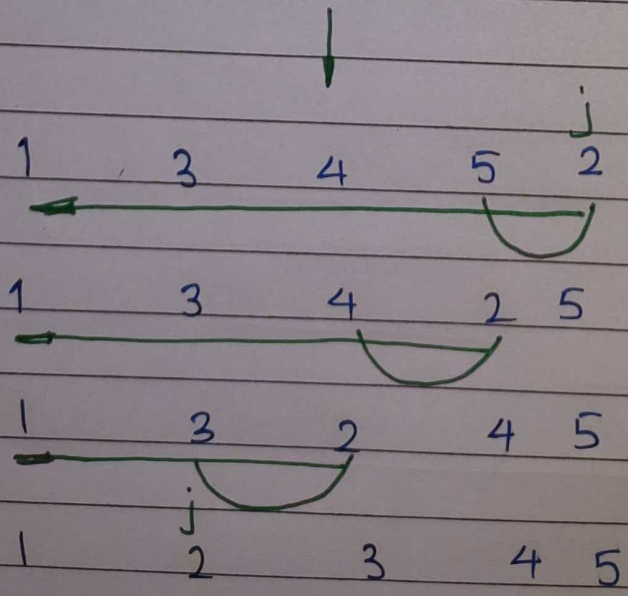


when $ele[j]$ is not smaller than $ele[j-1]$ break.

$i=2$
 $j=3$



$i=3$
 $j=4$



* COMPLEXITY:-

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

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

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

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

$\therefore O(N^2)$

- **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.