Insertion Sort

arr[] → | 10 | 1 | 7 | 4 | 8 | 2 | 11 |

↑

Round 1:

i = 1

1 < 10 → left-side

10 → 1 place

0th index → 1 copy karke

1 10 | 7 4 8 2 11

Round 2:

i = 2

7 < 10 → left

7 > 1 → right

1 7 10 | 4 8 2 11

Round 3:

i = 3

4 < 10 → left ∴ 10, 7 → 1 place

4 < 7 → left

4 > 1 → right

4

1 4 7 10 | 8 2 11

Round 4:

8 < 10 → right

Round 5:

1 4 7 8 10 | 2 11

↓

right shift

Round 6:

1 2 4 7 8 10 | 11

11 > 10 → no shift.

→ Sorted.

```
int i, key, j;
 for(i=1; i<n; i++){
       key = arr[i]
       j = i-1;
     while(j>=0 && arr[j]>key){
          arr[j+1] =arr[j]
           j = j-1;
     }
      arr[j+1]=key;

 }
```

why? ⇒ Adaptable method.
→ stable Algorithm.

* **T.C** ⇒ $O(n^2)$, **best case**: $O(n)$ **worst case**: $O(n^2)$

$$\left.\begin{array}{l}1\\2\\3\\\vdots\\n-1\end{array}\right\}\frac{n(n-1)}{2}$$

Space complexity:- constant.