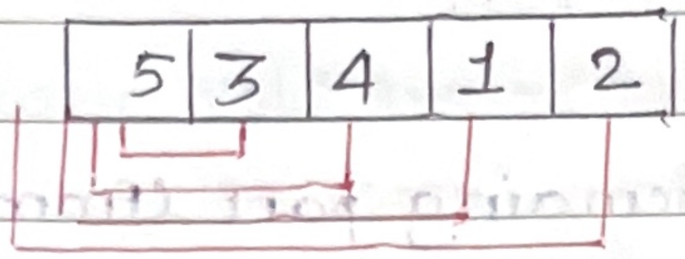
INSERTION SORT

- Insertion sort involves taking one element at a time from an unsorted list and placing it in its proper position within a sorted list, gradually building the sorted list
- Partially sorting the array.



Working -

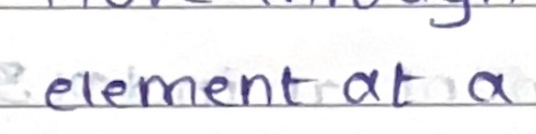
1. Initial State : The first element of the list is considered as a sorted portion and remaing elements are part of unsorted portion
2. Iterate. : Move through unsorted portion taking one element at a time
3. Compare and Shift : Compare the current element with those in sorted portion. Shift larger elements to right.
4. Insertion : Insert the current element in its sorted place among the sorted elements.
5. Repeat : Keep iterating through the unsorted portion untill all elements are sorted

(until $i < n-2$)

first pass:

$i = 0$ $j = i+1$

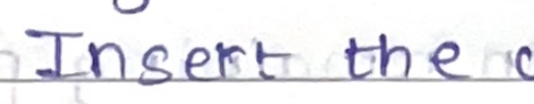
$j = 1$



(is $j - 1 < j$)

$1 < 7 \longrightarrow$ swap

$j--$ ($\because j != 0$ out of loop) $i++$



Second pass

0	1	2	3
1	7	23	4

i=1 j=i+1

j= 2

23 < 7 — No — break (As all element before it will also be smaller)

No swap in this iteration. i++

Third pass

0	1	2	3
1	7	23	4

i=2 j= 2+1

j=3

(j < j-1) — True — Swap

j = j-1 = 2

1	7	4	23

ij

(j < j--) — True — Swap

j = j-- = 1

1	4	7	23

(j < j-1) — False — No Swap

Break; i++

Out of loop as i value become > n-2

final Array :

1	4	7	23

Sorted Array

- Time Complexity-
(runtime efficency)
- Best Case. ⟶ O(N)

This occurs when the input is already sorted

- Worst case ⟶ $O(N^2)$

This occurs when the input is in reverse order leading to maximum comparisons and shifts

Insertion Sort is stable sorting algorithm

Example:

```
class Main {
    public static void main (String[] args) {
        int[] arr = { 7, 1, 23, 4 };
        insertion (arr);
        System.out.println (Arrays.toString (arr));
    }
    static void swap (int[] arr, int first, int second) {
        int temp = arr [first];
        arr [first] = arr [second];
        arr [second] = temp;
    }
    static void insertion (int[] arr) {
        for (int i=0; i<arr.length-1; i++) {
            for (int j=i+1; j>0; j--) {
                if (arr [j] < arr [j-1]) {
                    swap (arr, j, j-1);
                else
                    break;
                }
            }
        }
    }
```

O/P- [1, 4, 7, 23]