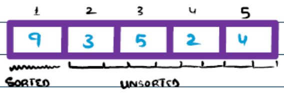
Sorting Algorithm

① Bubble Sort

② Insertion sort

Insertion Sort

* Think of it as cards



Consider Element 2

Is 9 > 3

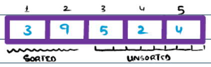
Yes

temp = 3

Shift 9 to right

element 1 will be empty

store 3 in element



temp = 3

Consider element 3

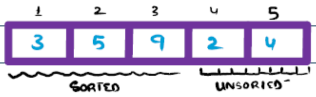
Where 5 should be inserted ?

b/w 3 and

Store 5 to temp variable

Shift 9 to right

Insert 5 b/w 3 and 9



temp = 5

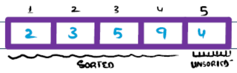
Consider element 4

Where 2 should be inserted

store 2 to temp

and shift element 1 , 2 and 3 values to right

insert 2 in element 1 .



temp = 2

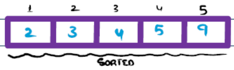
Consider element 5

Where 4 should be inserted ?

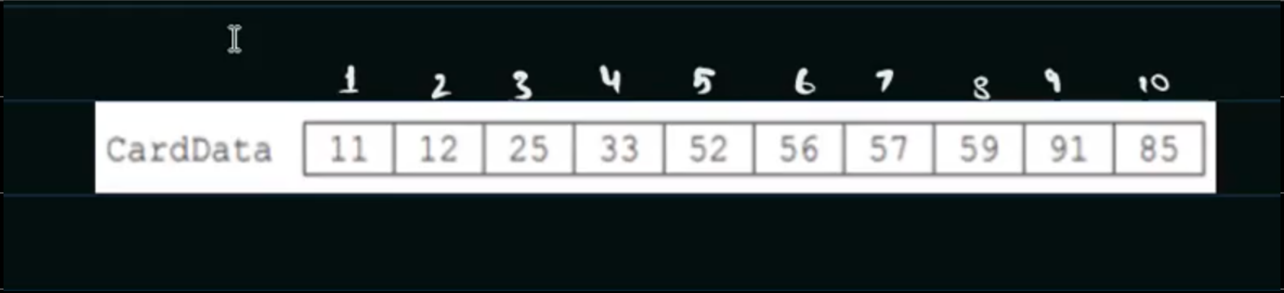
store 4 to temp

shift element $\overset{3}{3}$ and $\overset{5}{4}$ to right.

insert 4 in element 3



Pseudocode



	1	2	3	4	5	6	7	8	9	10
CardData	11	12	25	33	52	56	57	59	91	85

```
ArraySize ← 10

FOR Pointer ← 2 TO ArraySize
    ValueToInsert ← CardData [Pointer]    ] storing
    HolePosition ← Pointer

    WHILE (HolePosition > 1) AND (CardData [HolePosition -1] > ValueToInsert) DO
        CardData [Holeposition] ← CardData [HolePosition -1]
        HolePosition ← HolePosition -1
    END WHILE
```

shift
ing

```
    CardData[HolePosition] ← ValueToInsert      ] Insertion
END FOR
```

Q- Situation and reason when insertion sort is more efficient than a bubble sort.

Situation: When a list is almost sorted

Reason: ... Because it will stop as soon as it is sorted

Situation: When there are large number of data items

Reason: Because it will perform fewer comparisons.

File 1: Q 16, 28(b), $\overbrace{\text{36 (c)}}$, 42, 50

File 2: Q4, 14

· Works well for incremental sorting as elements are added to a list over a period of time

- As the number of elements increase , time taken to sort the data increase.
- As number of elements increase, performance of bubble sort deteriorates faster than insertion sort.