Insertion Sort (working)

It inserts each item into the proper place in the final list. The simplest implementation of this requires two list structures - the source list and the list into which sorted items are inserted.

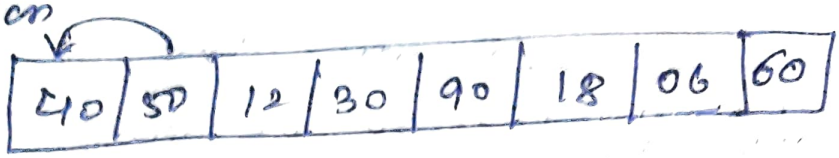
To save memory most implementation use an inplace sort that works by moving the current item past the already sorted items and repeatedly swapping it with the preceding item until it is in place.

During the first iteration, the element position 2nd is compared with the element at the 1st position. During second iteration the element at the position 3rd is compared with the elements at the 2nd & 1st position. This process is repeated for all the elements in the array upto (n-1) iterations

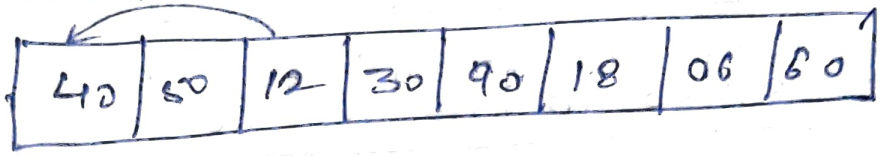
- card players.

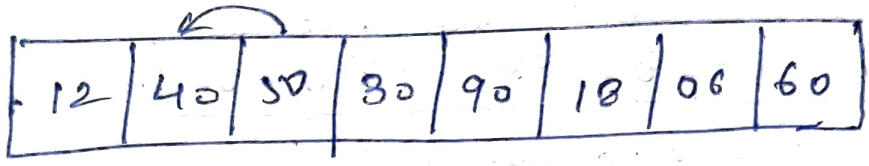
40	50	12	30	90	18	06	60

First iteration

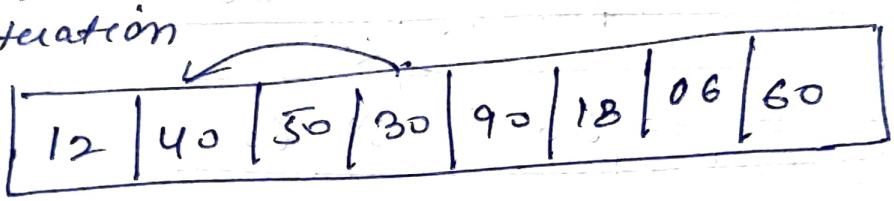


Second iteration

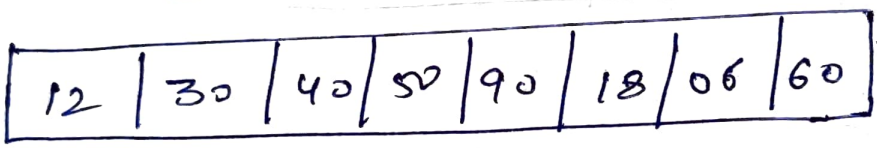




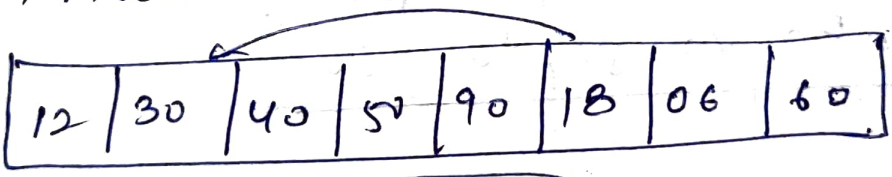
Third iteration

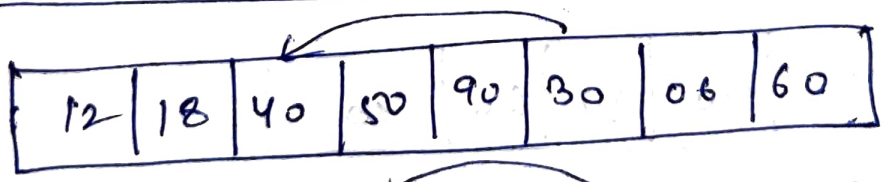


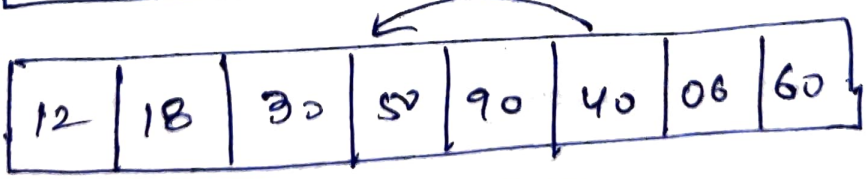
Fourth iteration

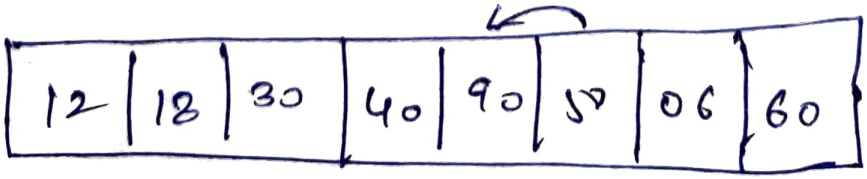


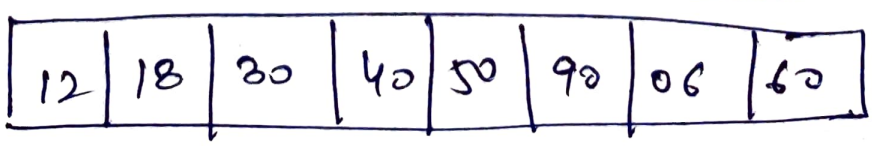
Fifth iteration



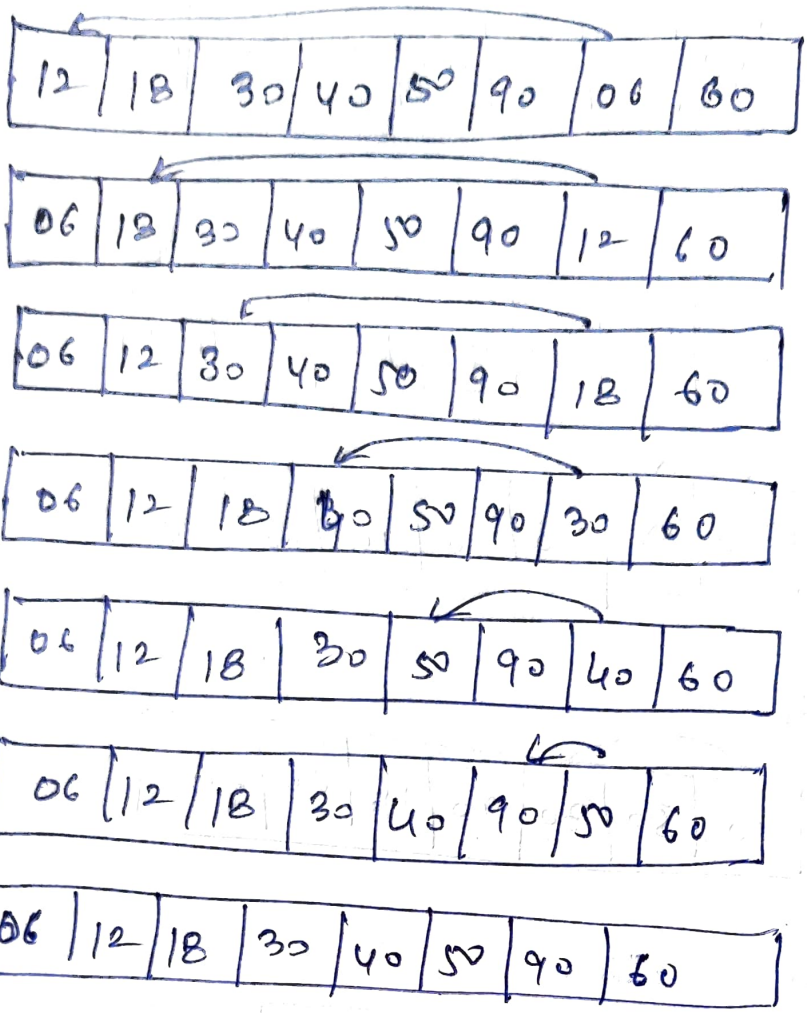




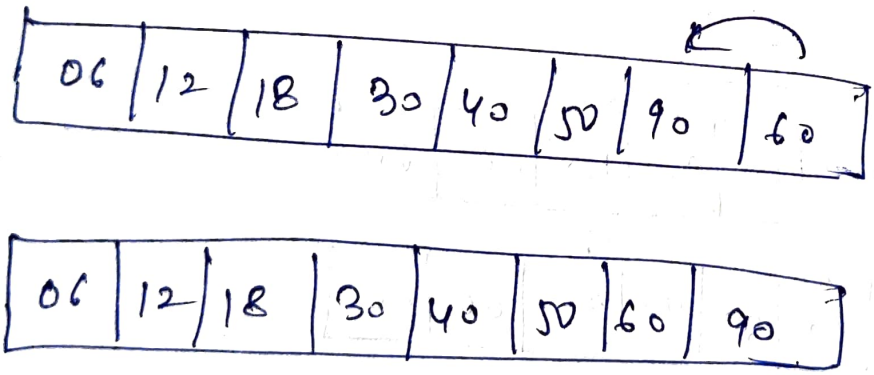




Sixth iteration



Seventh iteration



This is the sorted array.

Insertion_Sort(A)

```
1.  for j ← 2 to length[A]
2.      do key ← A[j]
3.  /* Insert A[j] into the sorted sequence A[1..j-1]
       */
4.      i ← j-1
5.      while i>0 and A[i]>key
6.          do A[i+1] ← A[i]
7.             i ← i-1
8.      A[i+1] ← key
```

Q Illustrate the operation of Insertion Sort on the array A=[2, 13, 5, 18, 14].

Soln

	1	2	3	4	5
A[] =	2	13	5	18	14

length = 5

for j = 2 to 5

Now j=2, key = A[2] ie key=13

i=2-1

ie i=1

while i>0 and A[1]>13

condition false. So no change

Now j=3, key=A[3]=5

$i = 3-1 = 2$

$j = 2$ key $= 5$

while $i > 0$ and $A[2] > key$

condition is true

so $A[2+1] \leftarrow A[2]$

ie $A[3] \leftarrow A[2]$

ie

2	5	18	18	14

and $i = 2-1 = 1$, $i = 1$

while $i > 0$ and $A[1] > key$

Condition false. so no change

then $A[1+1] \leftarrow key$

$A[2] \leftarrow 5$

ie

2	5	13	18	14

for $j = 4$ $\quad i = j-1 = 4-1 = 3$

key $= A[4]$

key $= 18$ $\quad i = 3$

Now while $3 > 0$ and $A[3] > 18$

Condition is false. No change

Similarly $j = 5$

key $= A[5]$

so key $= 14$ $\quad i = 4$

Now, while $i > 0$ and $A[i] > 14$

condition is true

So $A[5] = 18$ and $i = 4-1 = 3$

Now, while $3 > 0$ and $A[3] > 14$

condition is false.

So $A[3+1] = A[4] = 14$

and the sorted array is.

$A[] =$

2	5	13	14	18

Complexity

(1) Best case: If the elements of array is already sorted, no interchange will take place although we have to go through $N-1$ passes.

$$O(N-1) = O(N).$$

(2) Average case :- on average there are $(N-1)/2$ interchanges and we have to go through $N-1$ passes to sort the list.

Therefore $(N-1) \times (N-1)/2 = \left(N^2 - \frac{2N+1}{2}\right)$

$$= O(N^2)$$

(3) Worst case: When the list is in the reverse order then each element inserted will be compared with previous $N-1$ elements in sorted list. We

face & go through N-1 passes to sort the

so $(N-1) \times (N-1) = N^2 - 2N + 1 = O(N^2)$