Insertion Sort (working)

It inserts each item its proper place in the fi. list. The simplest implementation of this require f 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 item preceding item until ~~it~~ is in pl 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

Eg card players.

40	50	12	30	90	18	06	60

rst itegation

40	50	12	30	90	18	06	60

Second iteration

40	50	12	30	90	18	06	60

12	40	50	30	90	18	06	60

Third iteration

12	40	50	30	90	18	06	60

fourth itegation

12	30	40	50	90	18	06	60

fifth iteration

12	30	40	50	90	18	06	60

12	18	40	50	90	30	06	60

12	18	30	50	90	40	06	60

12	18	30	40	90	50	06	60

12	18	30	40	50	90	06	60

Sixith iteration

12	18	30	40	50	90	06	60

06	18	30	40	50	90	12	60

06	12	30	40	50	90	18	60

06	12	18	40	50	90	30	60

06	12	18	30	50	90	40	60

06	12	18	30	40	90	50	60

06	12	18	30	40	50	90	60

Seventh iteration

06	12	18	30	40	50	90	60

06	12	18	30	40	50	60	90

This is the sorted array.

```
Insertion_Sort(A)

for j ← 2 to length[A]
        do key ← A[j]
/* Insert A[j] into the sorted sequence A[1..j-1]
                                                   */

        i ← j-1
        while i>0 and A[i] > key
                do A[i+1] ← A[i]
                        i ← i-1
        A[i+1] ← key
```

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

Sol^n

A[] =

1	2	3	4	5
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 $A[2+1] \leftarrow A[2]$

iè $A[3] \leftarrow A[2]$

iè

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

$A[2] \leftarrow 5$

iè

2	5	13	18	14

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

key $= A[4]$

key $= 18$ $i = 3$

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

Condition is false. No change

Similarly $j = 5$

key $= A[5]$

so key $= 14$ $i = 4$

Now, while 4 > 0 and A[4] > 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 takeplace 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 $(N-1) \times (N-1)/2 = \frac{(N^2 - 2N + 1)}{2}$

$= O(N^2)$

(3) Worst case! When the list is in the reverse order then each element inserted will be composed with previous N-1 elements in sorted list, we

have to go through N-1 passes to sort the

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