Dr. Ohidujjaman Tuhin
Assistant Professor
Dept. of CSE , UIU

Insertion Sort

A sorting algorithm that inserts each item in the proper location comparing it with each item in the list until it finds the end of the list. Insertion sort is an elementary sorting algorithm that sorts one element at a time. The algorithm takes an element from the list and places it in the correct location of the list. This process is repeated until there are no more unsorted items in the list. It is less efficient than more advanced sorting algorithms, such as quick sort, heap sort, or merge sort, especially for large lists. For instance, most humans, when sorting a deck of cards, will use a strategy similar to insertion sort.

Mechanism of Insertion Sort

Ascending order:

Let A be an array of several elements n, j=location of array elements.

A	10	5	9	6
	1	2	3	4

j=2 t=A[j] i=j-1
t=5 i=1

i>=1 AND A[i]>=t
1>=1 AND A [1]>=5
True
A[i+1]=A[i]
i=i-1

10	10	9	6
1	2	3	4

i=0

i>=1 AND A[i]>=t
0>=1 AND A [0]>=5
False
A[i+1]=t

5	10	9	6
1	2	3	4

j=3 t=A[j] i=j-1
t=9 i=2

i>=1 AND A[i]>=t
2>=1 AND A[2]>=9
True
A[i+1]=A[i]
i=i-1

5	10	10	6
1	2	3	4

i=1

i>=1 AND A[i]>=t
1>=1 AND A[1]>=9
False
A[i+1]=t

5	9	10	6

1 2 3 4

j=4

t=A[j]

t=6

i=j-1

i=3

i>=1 AND A[i]>=t

3>=1 AND A[3]>=6

True

A[i+1]=A[i]

i=i-1

5	9	10	10

1 2 3 4

i=2

i>=1 AND A[i]>=t

2>=1 AND A[2]>=6

True

A[i+1]=A[i]

i=i-1

5	9	9	10

1 2 3 4

i=1

i>=1 AND A[i]>=t

1>=1 AND A[1]>=6

False

A[i+1]=t

5	6	9	10

1 2 3 4

Sorted List

Iterative Algorithm of Insertion Sort

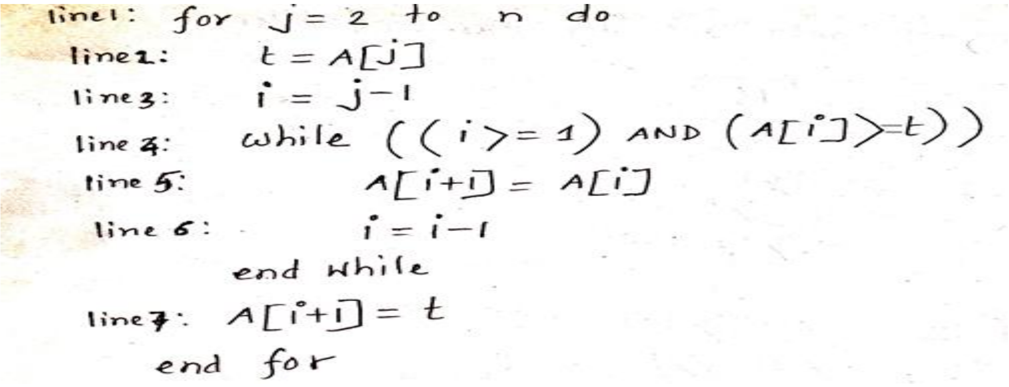
Algorithm ascending order:

```
for j=2   to    n
        t=A[j]
        i=j-1
       while((i>=1) &&(A[i]>=t
                  A[i+1]=A[i]
                    i=i-1
        end while loop
       A[i+1]=t
end for loop
```

Algorithm descending order:

```
for j=2   to    n
        t=A[j]
        i=j-1
       while((i>=1) &&(A[i]<=t
                  A[i+1]=A[i]
                    i=i-1
        end while loop
       A[i+1]=t
end for loop
```

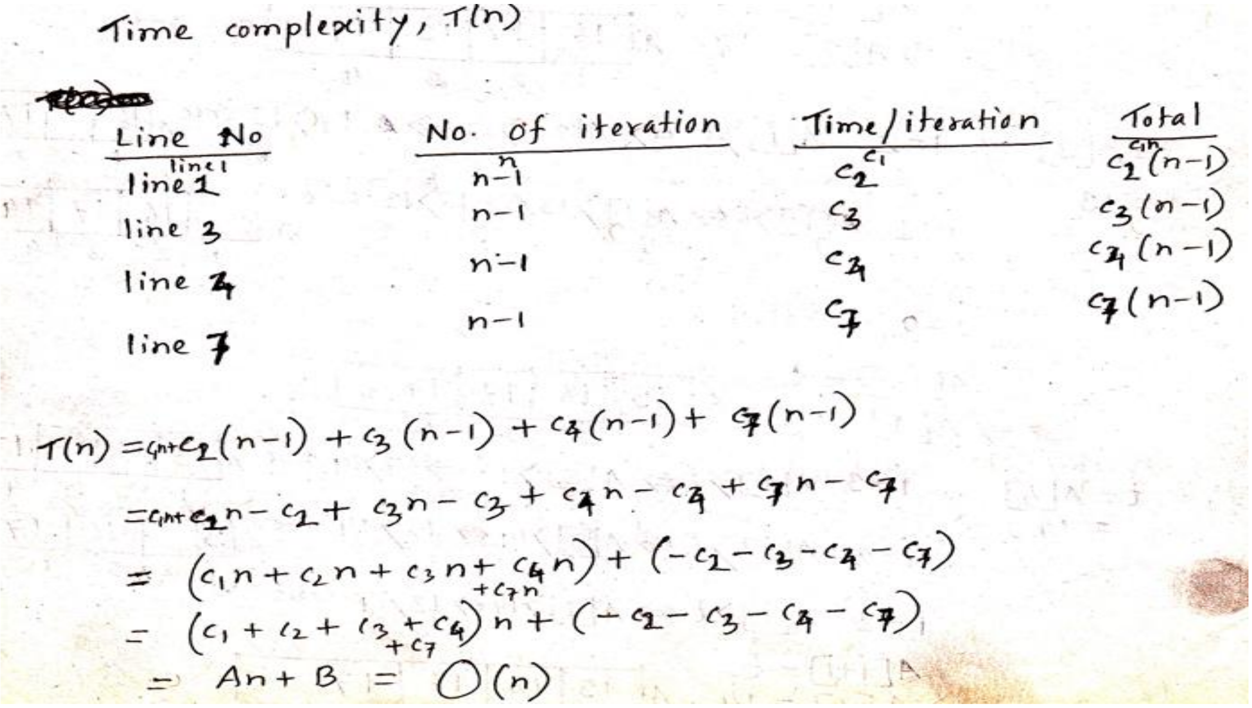
Time complexity analysis:



```
line1:  for j = 2 to n do
line2:      t = A[j]
line3:      i = j-1
line4:      while ((i >= 1) AND (A[i] >= t))
line5:          A[i+1] = A[i]
line6:          i = i-1
            end while
line7:      A[i+1] = t
        end for
```

Best case: when data is sorted A[]

5	6	9	10



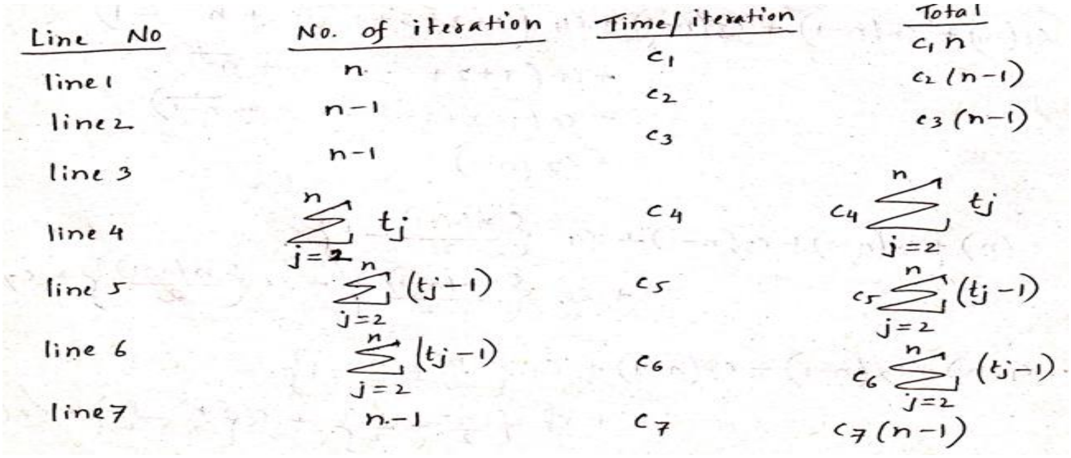
Time complexity, T(n)

Line No	No. of iteration	Time/iteration	Total
$\overset{\text{line1}}{\text{line 2}}$	$\overset{n}{n-1}$	$\overset{c_1}{c_2}$	$\overset{c_1 n}{c_2(n-1)}$
line 3	$n-1$	c_3	$c_3(n-1)$
line 4	$n-1$	c_4	$c_4(n-1)$
line 7	$n-1$	c_7	$c_7(n-1)$

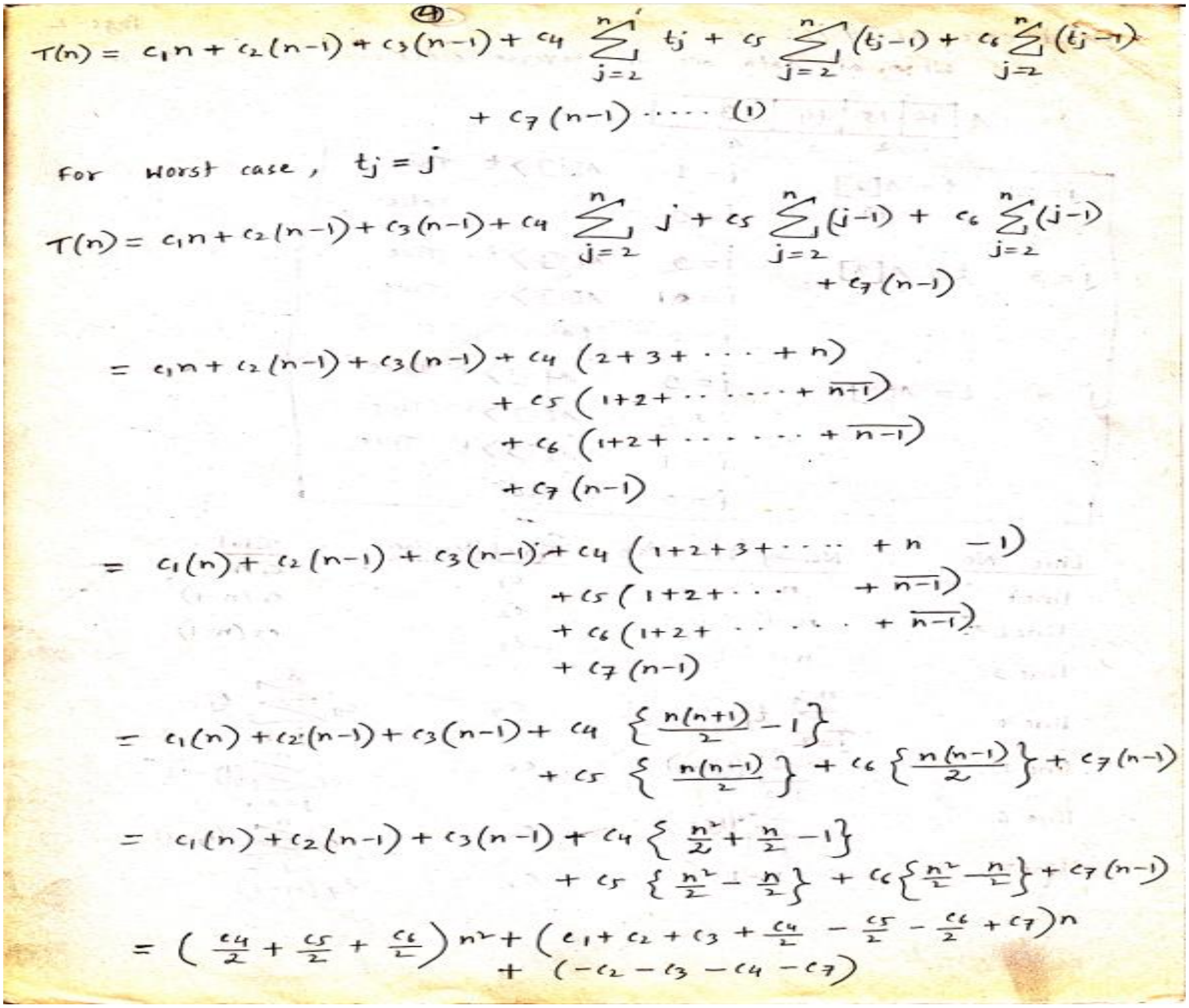
$$
\begin{aligned}
T(n) &= c_1n + c_2(n-1) + c_3(n-1) + c_4(n-1) + c_7(n-1) \\
&= c_1n + c_2n - c_2 + c_3n - c_3 + c_4n - c_4 + c_7n - c_7 \\
&= (c_1n + c_2n + c_3n \underset{+c_7n}{+} c_4n) + (-c_2 - c_3 - c_4 - c_7) \\
&= (c_1 + c_2 + c_3 \underset{+c_7}{+} c_4)n + (-c_2 - c_3 - c_4 - c_7) \\
&= An + B = O(n)
\end{aligned}
$$

Worst case: when all data are in reverse order A[]

10	9	6	5



Line No	No. of iteration	Time/iteration	Total
line1	n	c_1	$c_1 n$
line2	$n-1$	c_2	$c_2(n-1)$
line 3	$n-1$	c_3	$c_3(n-1)$
line 4	$\sum_{j=2}^{n} t_j$	c_4	$c_4 \sum_{j=2}^{n} t_j$
line 5	$\sum_{j=2}^{n} (t_j-1)$	c_5	$c_5 \sum_{j=2}^{n} (t_j-1)$
line 6	$\sum_{j=2}^{n} (t_j-1)$	c_6	$c_6 \sum_{j=2}^{n} (t_j-1)$
line7	$n-1$	c_7	$c_7(n-1)$



④

$$T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=2}^{n} t_j + c_5 \sum_{j=2}^{n} (t_j-1) + c_6 \sum_{j=2}^{n} (t_j-1) + c_7(n-1) \cdots\cdots (1)$$

For Worst case, $t_j = j$

$$T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=2}^{n} j + c_5 \sum_{j=2}^{n} (j-1) + c_6 \sum_{j=2}^{n} (j-1) + c_7(n-1)$$

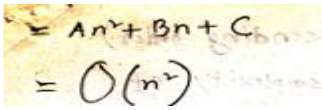
$$= c_1 n + c_2(n-1) + c_3(n-1) + c_4(2+3+\cdots+n) + c_5(1+2+\cdots\cdots+\overline{n-1}) + c_6(1+2+\cdots\cdots+\overline{n-1}) + c_7(n-1)$$

$$= c_1(n) + c_2(n-1) + c_3(n-1) + c_4(1+2+3+\cdots+n \quad -1) + c_5(1+2+\cdots \quad +\overline{n-1}) + c_6(1+2+\cdots\cdots+\overline{n-1}) + c_7(n-1)$$

$$= c_1(n) + c_2(n-1) + c_3(n-1) + c_4\left\{\frac{n(n+1)}{2} - 1\right\} + c_5\left\{\frac{n(n-1)}{2}\right\} + c_6\left\{\frac{n(n-1)}{2}\right\} + c_7(n-1)$$

$$= c_1(n) + c_2(n-1) + c_3(n-1) + c_4\left\{\frac{n^2}{2} + \frac{n}{2} - 1\right\} + c_5\left\{\frac{n^2}{2} - \frac{n}{2}\right\} + c_6\left\{\frac{n^2}{2} - \frac{n}{2}\right\} + c_7(n-1)$$

$$= \left(\frac{c_4}{2} + \frac{c_5}{2} + \frac{c_6}{2}\right)n^2 + \left(c_1 + c_2 + c_3 + \frac{c_4}{2} - \frac{c_5}{2} - \frac{c_6}{2} + c_7\right)n + (-c_2 - c_3 - c_4 - c_7)$$

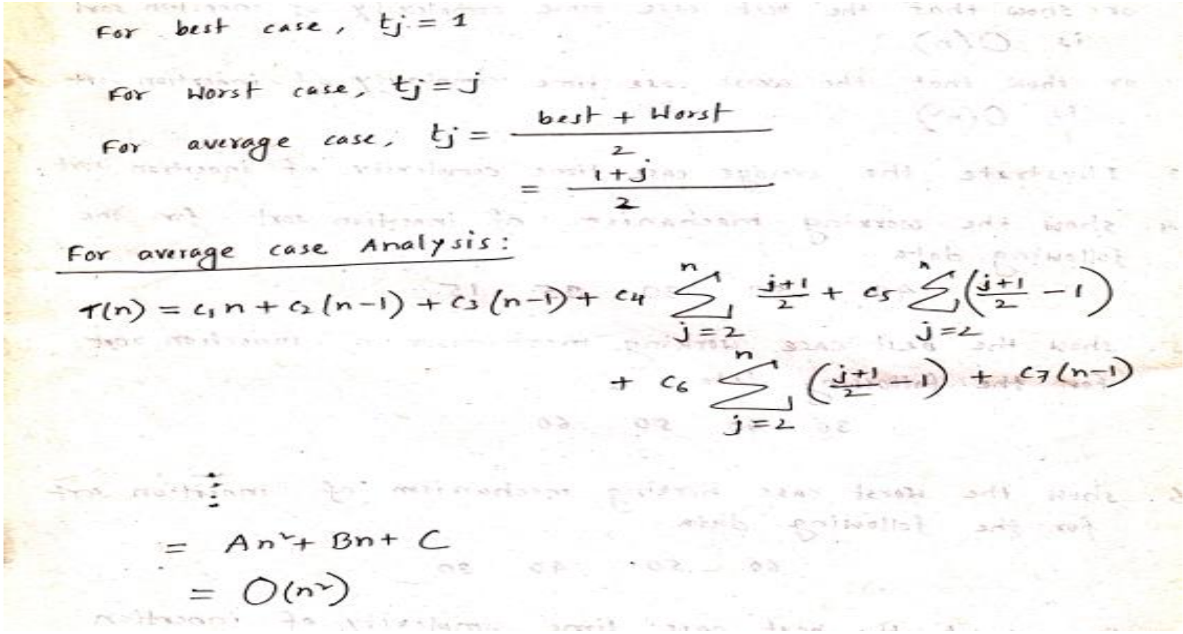


$$= An^2 + Bn + C$$

$$= O(n^2)$$

A quadratic time complexity= O(n^2)

For average case:



For best case, $t_j = 1$

For Worst case, $t_j = j$

For average case, $t_j = \frac{\text{best} + \text{Worst}}{2}$

$= \frac{1+j}{2}$

For average case Analysis:

$$T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=2}^{n} \frac{j+1}{2} + c_5 \sum_{j=2}^{n}\left(\frac{j+1}{2} - 1\right)$$

$$+ c_6 \sum_{j=2}^{n}\left(\frac{j+1}{2} - 1\right) + c_7(n-1)$$

$\vdots$

$= An^2 + Bn + C$

$= O(n^2)$

Write a program in C for iterative insertion sort algorithm (ascending case)

```
#include<stdio.h>
void main()
{
        int A[10],n,i,j,t;
        printf("Enter the number of Array elements n:=");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        {
       printf("Enter %d element of Array[]:=",i);
       scanf("%d",&A[i]);
        }
        for(j=2;j<=n; j++)
        {
                t=A[j];
                i=j-1;
                while(i>=1 && A[i]>=t)
                {
                        A[i+1]=A[i];
                        i=i-1;
                }
                A[i+1]=t;
        }
        printf("After Insertion Sort:=");
        for(i=1;i<=n;i++)
        {
          printf("  %d",A[i]);
        }
        printf("\n");
}
```

Output on monitor:

```
Enter the number of Array elements n:=4
Enter 1 element of Array[]:=10
Enter 2 element of Array[]:=5
Enter 3 element of Array[]:=9
Enter 4 element of Array[]:=6
After Insertion Sort:=  5  6  9  10
```