Experiment No.1
Insertion Sort
Date of Performance:
Date of Submission:

Title: Insertion Sort

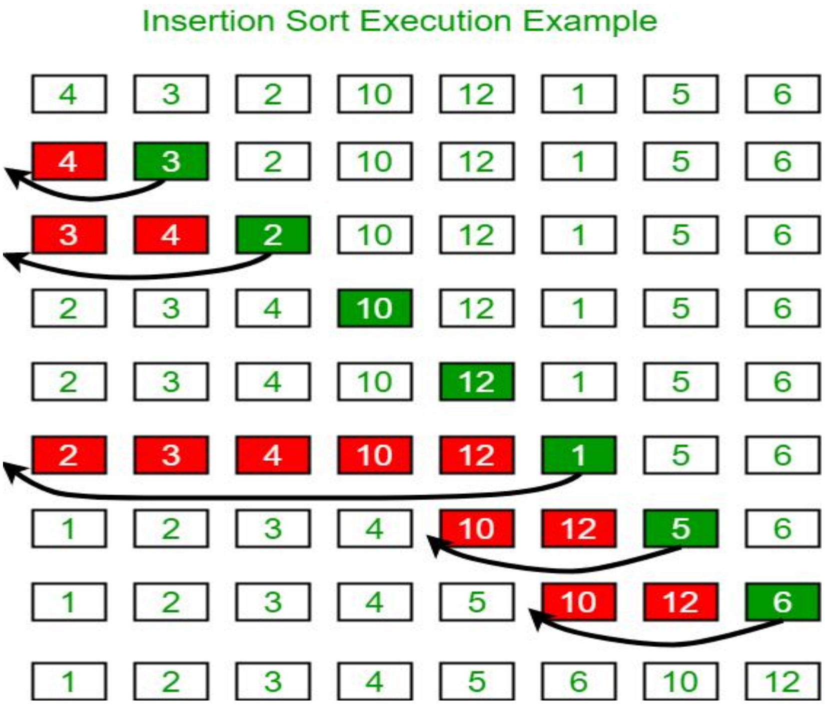
Aim: To implement Selection Comparative analysis for large values of 'n'

Objective: To introduce the methods of designing and analysing algorithms

Theory:

Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

Example:



4 3 2 10 12 1 5 6

4 3 2 10 12 1 5 6

3 4 2 10 12 1 5 6

2 3 4 10 12 1 5 6

2 3 4 10 12 1 5 6

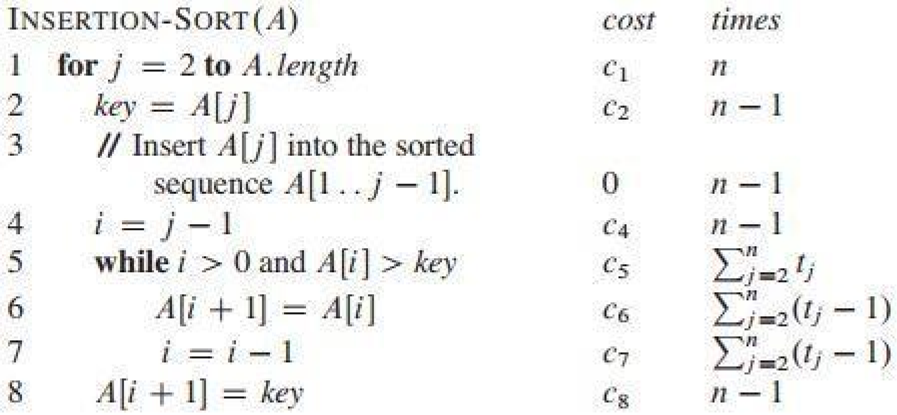
2 3 4 10 12 1 5 6

1 2 3 4 10 12 5 6

1 2 3 4 5 10 12 6

1 2 3 4 5 6 10 12

Algorithm and Complexity:



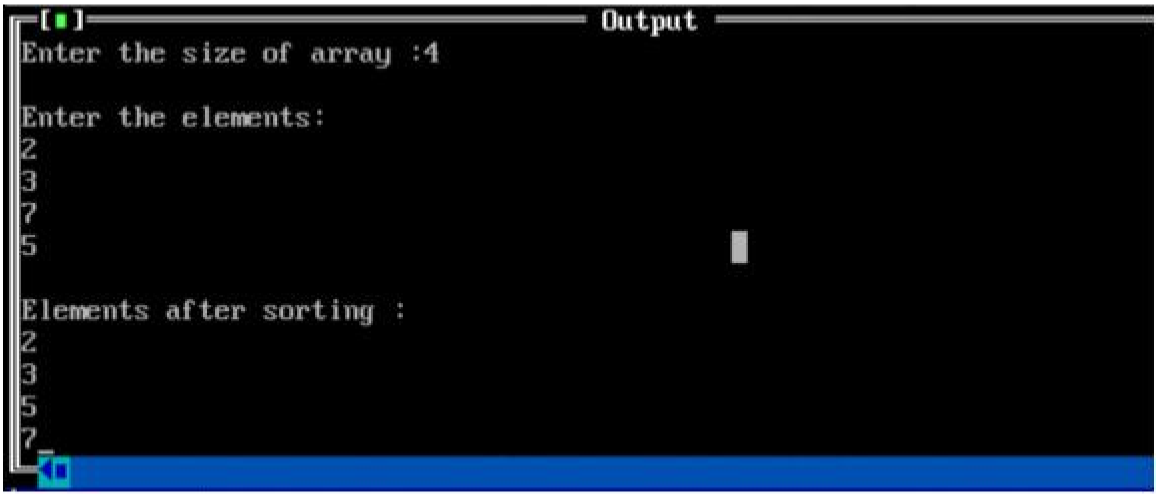
```
INSERTION-SORT(A)                                   cost    times
for $j = 2$ to $A.length$                           $c_1$   $n$
    $key = A[j]$                                    $c_2$   $n-1$
    // Insert $A[j]$ into the sorted
         sequence $A[1..j-1]$.                      0       $n-1$
    $i = j-1$                                       $c_4$   $n-1$
    while $i > 0$ and $A[i] > key$                  $c_5$   $\sum_{j=2}^{n} t_j$
        $A[i+1] = A[i]$                             $c_6$   $\sum_{j=2}^{n} (t_j - 1)$
        $i = i-1$                                   $c_7$   $\sum_{j=2}^{n} (t_j - 1)$
    $A[i+1] = key$                                  $c_8$   $n-1$
```

Implementation:

```
#include<stdio.h>

#include<conio.h>

int main(){

int i,j,key,n;

int A[100];

clrscr();

printf("***INSERTION SORT***");

printf("\nEnter the size of array :");

scanf("%d",&n);

printf("\nEnter the elements: \n");

for(i=0;i<n;i++){

scanf("%d",&A[i]);

}


for(j=1;j<=n;j++){

  key=A[j];
```

```
 i=j-1;

 while(i>0 && A[i]>key){

  A[i+1]=A[i];

  i=i-1;

  }

 A[i+1]=key;

 }



printf("\nElements after sorting :");

for(i=0;i<n;i++){

printf("\n%d",A[i]);

}


return 0;


}
```

Output:



```
Enter the size of array :4

Enter the elements:
2
3
7
5

Elements after sorting :
2
3
5
7_
```

Conclusion: The implementation of the insertion sort algorithm demonstrated its effectiveness in sorting small to moderate-sized datasets. While its simplicity and efficiency are notable, scalability limitations highlight the need for alternative algorithms for larger datasets. Nonetheless, insertion sort remains a valuable foundational concept in computer science education.