Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2024-25

Class:	**SE**	**Semester:**	**IV**
Course Code:	**CSL401**	**Course Name:**	**Analysis of Algorithm Lab**

Name of Student:	**Shravani Sandeep Raut**
Roll No. :	**48**
Experiment No.:	**1**
Title of the Experiment:	**Insertion Sort**
Date of Performance:	**09/01/2025**
Date of Submission:	**16/01/2025**

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty : Mrs. Sneha Yadav

Signature :

Date:

Title: Insertion Sort

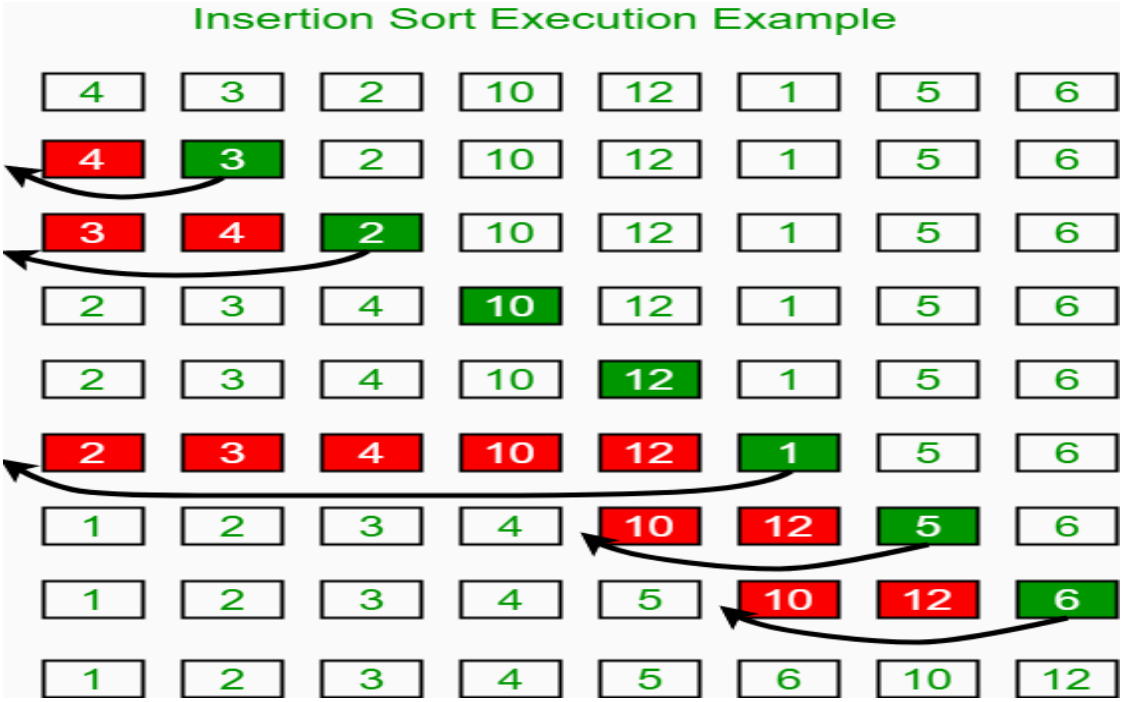
Aim: To implement Selection Comparative analysis for large values of 'n'

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:



Algorithm and Complexity:

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7 $i = i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8 $A[i+1] = key$	c_8	$n-1$

Implementation:

```
#include <stdio.h>

int n, i, j, A[20], key;

void Insertion_Sort(int A[], int n);

void main()
{
   printf("Enter the size of array: ");
   scanf("%d", &n);
   printf("Enter the elements of array: \n");
   for(i=0; i<n; i++)
   {
      printf("Enter value: ");
      scanf("%d", &A[i]);
   }
   printf("The unsorted array is: ");
   for(i=0; i<n; i++)
   {
      printf("%d\t",A[i]);
   }
   Insertion_Sort(A,n);
   printf("\nAfter sorting array is: ");
   for(i=0; i<n; i++)
   {
      printf("%d\t",A[i]);
   }
}
```

```
void Insertion_Sort(int A[], int n)
{
   for(i = 1; i<= n-1; i++)
   {
      key = A[i];
      j = i -1;
      while(j >= 0 && A[j] > key)
      {
         A[j+1] = A[j];
         j = j-1;
      }
      A[j+1] = key;
   }
}
```

Output -

```
Enter the size of array: 5
Enter the elements of array:
Enter value: 23
Enter value: 14
Enter value: 67
Enter value: 45
Enter value: 78
The unsorted array is: 23       14      67      45      78
After sorting array is: 14      23      45      67      78
```

Conclusion:
Insertion Sort is a simple sorting algorithm that builds a sorted array one element at a time.

- Time Complexity
  - Best Case: O(n)
  - Average Case: $O(n^2)$
  -Worst Case: $O(n^2)$

- Space Complexity: O(1) (in-place sorting)

- Advantages:
  - Easy to implement
  - Efficient for small or nearly sorted datasets
  - Stable (preserves the order of equal elements)

Insertion Sort is effective for small lists and is a good introductory algorithm for understanding sorting concepts.