Experiment No. : 1

Title: Basic Sorting algorithm and its analysis

Batch:SY-IT(B3) **Roll No.: 16010423076** **Experiment No.: 1**

Aim: To implement and analyse time complexity of insertion sort.

Explanation and Working of insertion sort:

Explanation :
Insertion Sort builds the sorted array one element at a time.
It takes elements from an unsorted array and places them in the correct position in the sorted portion.

Working :
Start from the second element(index 1).
Compare it with the elements before it and shift larger elements one position to the right.
Insert the current element in its correct position.
Repeat for all elements in the array.

Algorithm of insertion sort:
1. Start from the second element (index 1).
2. Store the current element in a temporary variable (we name it key here).
3. Compare key with previous elements:
   Shift elements larger than key to the right.
4. Insert key in its correct position.
5. Repeat for all elements in the array.

K. J. SOMAIYA COLLEGE OF ENGG.

Derivation of Analysis insertion sort:

Worst Case Analysis

Happens when the array is sorted in reverse order.
Every element is compared with all previous elements.
Formula / comparisons : n(n-1)/2 = O(n^2)
Shifts : O(n^2)

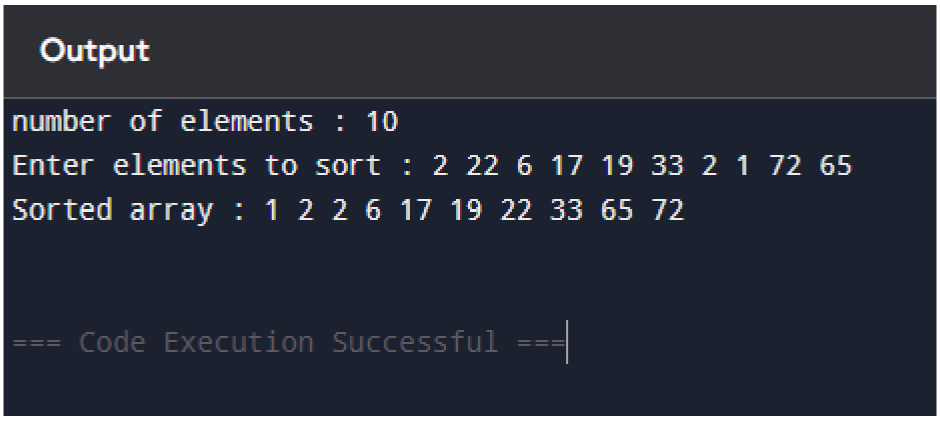
Best Case Analysis
Happens when the array is already sorted.
Only one comparison per element.
Comparisons: O(n)
Shifts: O(1)

Average Case Analysis
Array is partially sorted.
Comparisons and shifts depend on how unsorted the array is.
Time complexity: O(n^2)

Program(s) of insertion sort:

```
#include <stdio.h>

//insertion sort algorithm
void insertionSort(int *arr, int n) {
    for (int i=1; i<n; i++) {
        int key = arr[i];
        int j = i-1;

        while (j>=0 && arr[j]>key)
        {
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = key;
    }
}

//to print sorted array
void printArray(int *arr, int n) {
    for (int i=0; i<n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

//main function to take input and call function
int main() {
    int n;
    printf("number of elements : ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter elements to sort : ");
    for (int i=0; i<n; i++)
        scanf("%d", &arr[i]);

    //function call to sort the array
    insertionSort(arr, n);

    printf("Sorted array : ");
    printArray(arr, n); //function call to print the array

    return 0;
}
```

Output(o) of insertion sort:



Output

```
number of elements : 10
Enter elements to sort : 2 22 6 17 19 33 2 1 72 65
Sorted array : 1 2 2 6 17 19 22 33 65 72


=== Code Execution Successful ===
```

Results:

Time Complexity of Insertion sort:

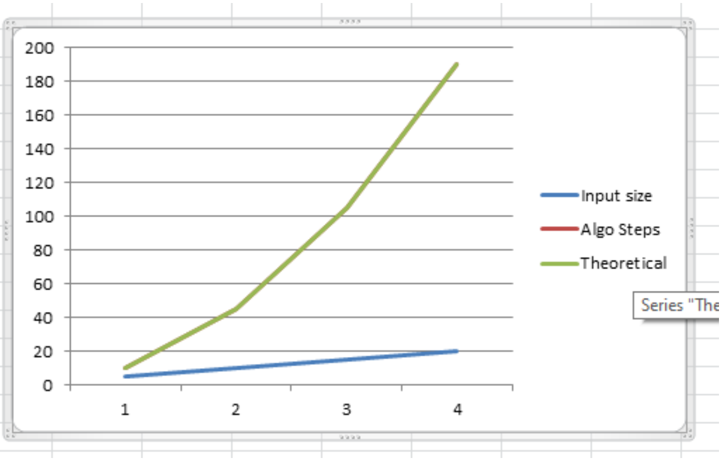
- **Best case: O(n)**
- **Average case: O(n^2)**
- **Worst case: O(n^2)**

Worst Case Analysis:

Sr. No.	Input size	No: of steps from Algorithm analysis	No: of steps from Theoretical analysis
1	5	10	n(n-1)/2 = 5(5-1)/2 = 10
2	10	45	n(n-1)/2 = 10(10-1)/2 = 45
3	15	105	n(n-1)/2 = 15(15-1)/2 = 105
4	20	190	n(n-1)/2 = 20(20-1)/2 = 190

GRAPH:

K. J. SOMAIYA COLLEGE OF ENGG.

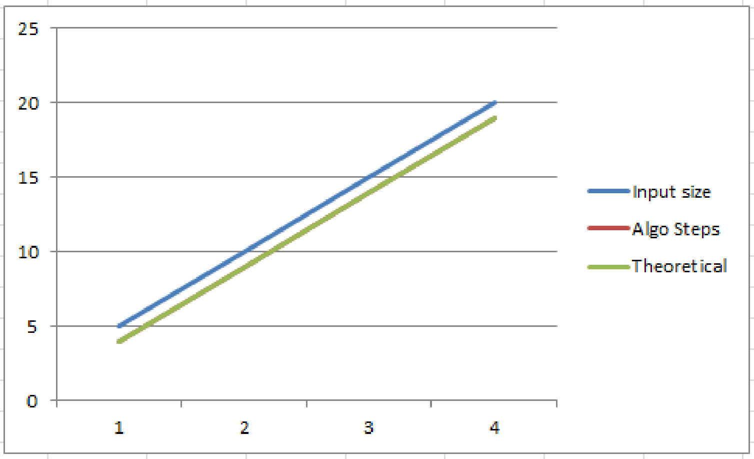


Best Case Analysis:

Sr. No.	Input size(n)	No: of steps from Algorithm analysis	No: of steps from Theoretical analysis
1	5	4	n-1=4
2	10	9	n-1=9
3	15	14	n-1=14
4	20	19	n-1=19

K. J. SOMAIYA COLLEGE OF ENGG.

GRAPH



Conclusion: (Based on the observations):

From this experiment, I learned how insertion sort works, its step-by-step algorithm and its behavior in different scenarios like the worst, best and average cases. I understood that the worst case happens when the array is sorted in reverse order, requiring maximum comparisons and shifts, while the best case occurs when the array is already sorted, requiring minimal steps.
Additionally, I learned how to represent this data visually using graphs, which provided a clearer understanding of the differences between algorithmic and theoretical analysis.

Outcome:
CO1. Analyse basic algorithms with its time and space complexity.

References:

1. Richard E. Neapolitan, " Foundation of Algorithms ", 5th Edition 2016, Jones & Bartlett Students Edition
2. Harsh Bhasin , " Algorithms : Design & Analysis", 1st Edition 2013, Oxford Higher education, India
3. T.H. Coreman ,C.E. Leiserson,R.L. Rivest, and C. Stein, " Introduction to algorithms", 3rd Edition 2009, Prentice Hall India Publication
4. Jon Kleinberg, Eva Tardos, " Algorithm Design", 10th Edition 2013, Pearson India Education Services Pvt. Ltd.

K. J. SOMAIYA COLLEGE OF ENGG.