Experiment No. : 1

Title: Basic Sorting algorithm and its analysis

Batch: B1 **Roll No.: 16010420133** **Experiment No.: 1**

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

Algorithm of insertion sort & Heap sort:

1) **Insertion Sort : Pseudocode -**

```
1: for j = 2 to A.length do
2:     key = A[j]
3:     i = j − 1
4:     while i > 0 and A[i] > key do
5:         A[i + 1] = A[i]
6:         i = i − 1
7:     end while
8:     A[i + 1] = key
9: end for
```

2) **Heap sort : Pseudocode –**

Heap Sort:

Here is the pseudocode for Heap Sort, modified to include a counter:

```
count ← 0
HeapSort(A)
1     Build_Max_Heap(A)
2     for i ← length[A] downto 2 do
3           exchange A[1] ↔ A[i]
4           heap-size[A] ← heap-size[A] - 1
5           Max_Heapify(A, 1)
```

And here is the algorithm for the Max_Heapify function used by Heap Sort:

```
Max_Heapify(A, i)
1     l ← LEFT(i)
2     r ← RIGHT(i)
3     if l ≤ heap-size[A] and A[l] > A[i]
4           then largest ← l
5           else largest ← i
6     if r ≤ heap-size[A] and A[r] > A[largest]
7           then largest ← r
8     if largest ≠ i
9           then  exchange A[i] ↔ A[largest]
9.5               count ← count + 1
10                Max_Heapify (A, largest)
```

And here is the algorithm for the Build_Max_Heap function used by Heap Sort:

```
Build_Max_Heap(A)
1     heap-size[A] ← length[A]
2     for i ← floor(length[A]/2) downto 1 do
3           Max_Heapify (A, i)
```

Derivation of Analysis insertion sort & Heap sort:

Heap Sort:

Worst Case Analysis

$T(n) = O(n(\log(n))$.

Best Case Analysis

$T(n) = O(n)$.

Insertion Sort:

Worst Case Analysis

$T(n) = C * (n^2)$ or $O(n^2)$

Best Case Analysis

$T(n) = C * (n)$ or $O(n)$

Program(s) of insertion sort & Heap sort:

Insertion Sort:

```
#include<iostream>

using namespace std;

int main()
{
    int i,j,n,key,a[30],c=0;
    cout<<"Enter the number of elements:";
    cin>>n;
    cout<<"\nEnter the elements\n";

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

```
        cin>>a[i];
    }

    for(i=1;i<=n-1;i++)
    {
        key=a[i];
        j=i-1;

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

        a[j+1]=key;
        c++;
    }

    cout<<"\nSorted list is as follows\n";
    for(i=0;i<n;i++)
    {
        cout<<a[i]<<" ";
    }
    cout<<"\nNo of passes : "<<c;

    return 0;
}
```

Heap Sort:

```
#include <iostream>
using namespace std;

int counter = 0;

void Max_Heapify(int array[], int i, int size)
{
    int temp = array[i];
    int j = 2 * i;

    while (j <= size)
    {
        if (j < size && array[j + 1] > array[j])
            j = j + 1;
```

```
        if (temp >= array[j])
        {
            break;
        }

        else if (temp < array[j])

        {
            counter++;
            array[j / 2] = array[j];
            j = 2 * j;
        }
    }

    array[j / 2] = temp;
}
void HeapSort(int array[], int size)
{
    int temp;

    for (int i = size; i >= 2; i--)
    {

        temp = array[i];
        array[i] = array[1];
        array[1] = temp;

        counter++;

        Max_Heapify(array, 1, i - 1);

    }

}

void Build_Max_Heap(int array[], int size)
{

    for (int i = size / 2; i >= 1; i--)
        Max_Heapify(array, i, size);

}

int main()
{
    int size = 10;
    size++;
```

```
    int case1[size];

    // int case1[size] = {0 ,23, 75, 11, 46, 1, 73, 62, 56, 6, 9};
    // int case1[size] = {0 ,1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    // int case1[size] = {0, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1};
    // int case1[size] = {0 ,75,56,73,46,9,11,62,23,6,1};
    // int case1[size] = {0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3};

/*  for (int k = 1; k < size; k++)
    {
        case1[k] = k ;
    } */



    Build_Max_Heap(case1, size - 1);
    HeapSort(case1, size - 1);

    // cout << size << endl;

    // for (int i = 1; i < size; i++)
    //  cout << " " << case1[i];
    cout << "\n" << counter << endl;
    return 0;
}
```

Output(o) of insertion sort & Heap sort:

Insertion Sort:

```
Enter the number of elements:5

Enter the elements
5
4
3
2
1

Sorted list is as follows
1 2 3 4 5
No of passes : 14
```

Heap Sort:

25

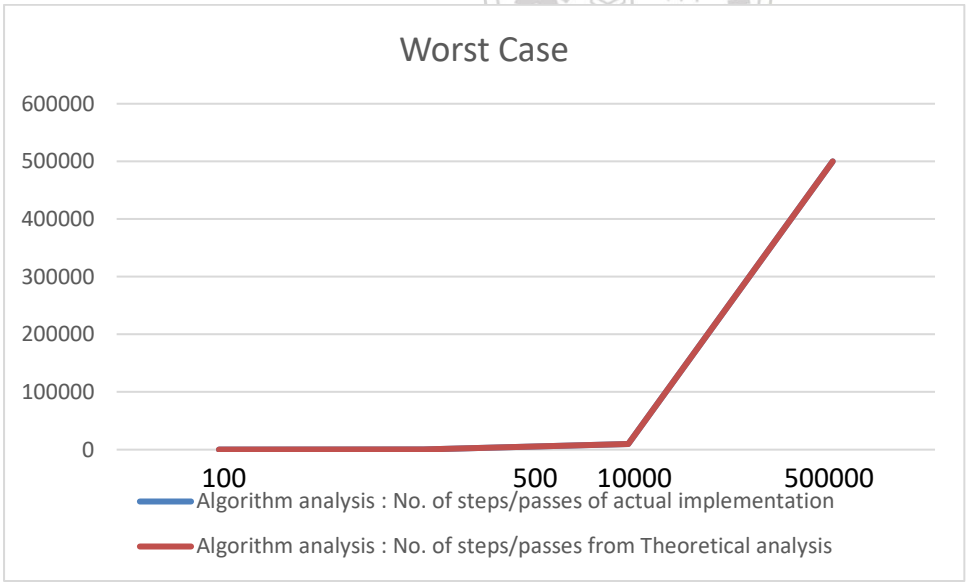
Results:

Time Complexity of Insertion sort:

Worst Case Analysis:

Sr. No.	Input size	Algorithm analysis : No. of steps/passes of actual implementation	Algorithm analysis : No. of steps/passes from Theoretical analysis
	100	4950	10000
	500	124750	250000
	10000	49995000	100000000
	500000	2,49,99,95,00,000	1,24,99,97,50,000

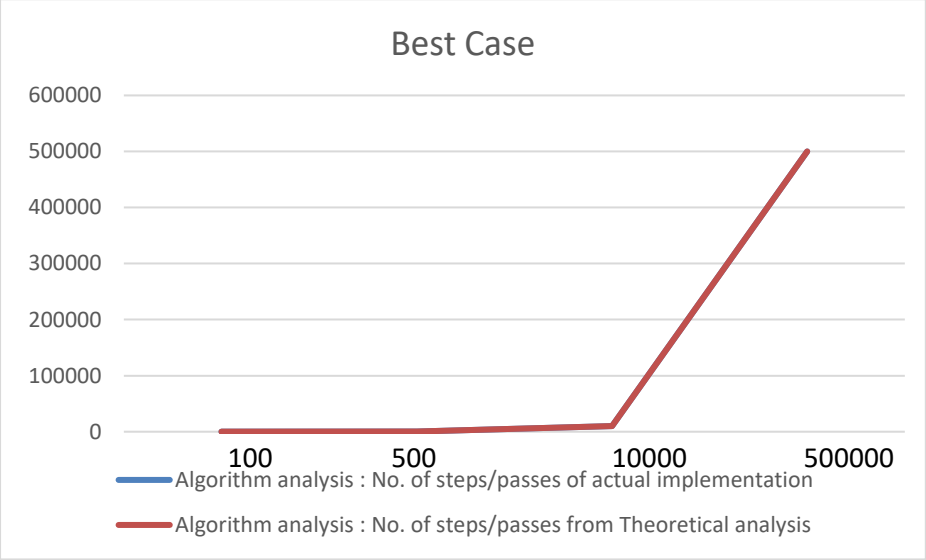
GRAPH:



Best Case Analysis:

Sr. No.	Input size	Algorithm analysis : No. of steps/passes of actual implementation	Algorithm analysis : No. of steps/passes from Theoretical analysis
	100	99	100
	500	499	500
	10000	9999	10000
	500000	499999	500000

GRAPH

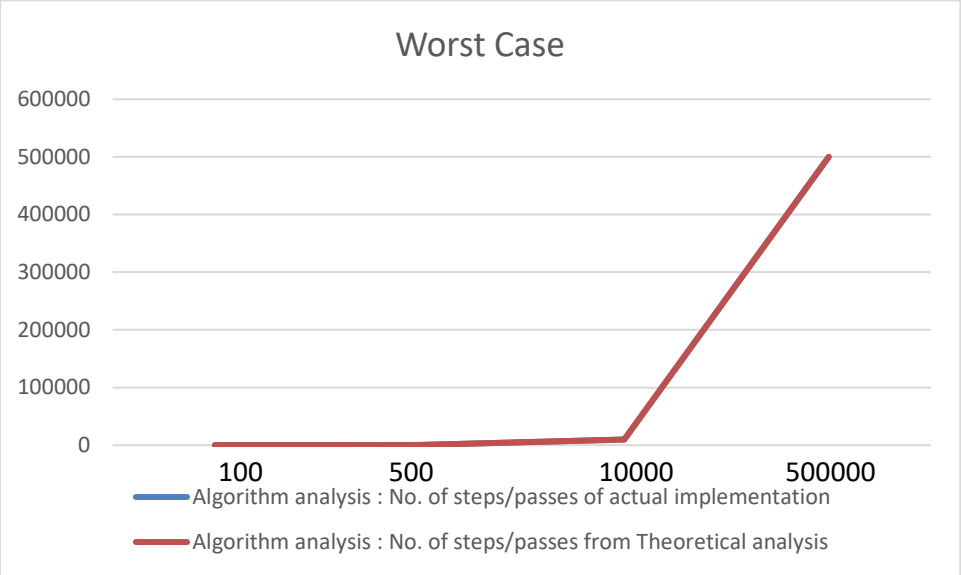


Time Complexity of Heap sort:

Worst Case Analysis:

Sr. No.	Input size	Algorithm analysis : No. of steps/passes of actual implementation	Algorithm analysis : No. of steps/passes from Theoretical analysis
	100	640	664
	500	4354	4482
	10000	131956	132877
	500000	9355700	9465784

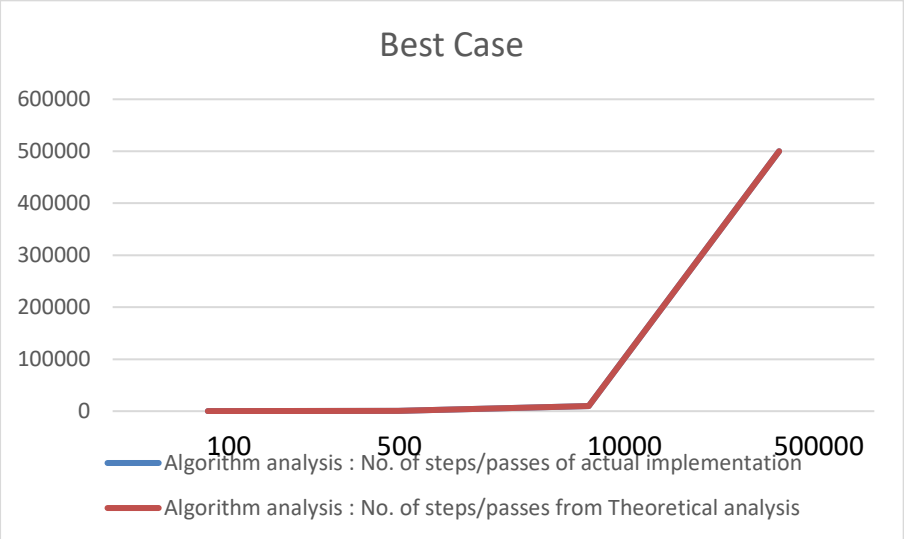
GRAPH:



Best Case Analysis:

Sr. No.	Input size	Algorithm analysis : No. of steps/passes of actual implementation	Algorithm analysis : No. of steps/passes from Theoretical analysis
	100	99	100
	500	499	500
	10000	9999	10000
	500000	499999	500000

GRAPH



Conclusion: From this experiment we learnt about the derivation of insertion and heap sort, performed the program and saw the graph. We also filled the complexity tables for best case and worst case of both the algorithms.

Outcome: CO1: Analyze time and space complexity of basic algorithms

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.