

**Experiment No. :**

**1**

**Title**

**:**

Introduction to Analysis of Algorithm

**Batch: B2 Roll No.: 1914078 Experiment No.:1**

**Aim:** To implement and analyze time complexity of insertion sort & selection sort.

**Algorithm of insertion sort & selection sort:**

**Algorithm for Selection sort:**

1. Start
2. Initialize array a[],n, i, j ,min, temp
3. Accept Number of users for n elements.
4. For i=0 to n-1 min=i for j=i+1 to n-1

if(a[j]<a[min]) min=j swap(a[i]<->a[min])

1. Print sorted array a[]
2. Stop

**Algorithm for Insertion sort:**

1. Start
2. Initialize array a[],n, i, j ,key
3. Accept Number of users for n elements. 4. for i = 1 to n key =a [i] j =i – 1

while j > = 0 and a[j] > key a[j+1] =a[j]

j = j – 1

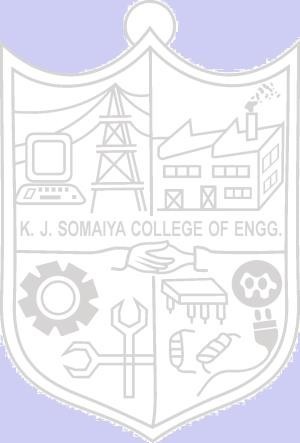
End while a[j+1] = key End for

1. Print sorted array a[]
2. Stop

**Derivation of Analysis insertion sort & selection sort:**

**Selection Sort**

**Worst Case Analysis**

 In case of selection sort the number of values to be processed are reduced in each step

No. of comparisons in step 1=n-1

No. of comparisons in step 2=n-2

.

.

.

No. of comparisons in last step=1

Hence

F(n)=n(n-1)/2

=(n^2-n)/2

=(n^2)/2-n/2

Hence Big O notation is given by

F(n)= O(n^2)

**Best Case Analysis**

In case of selection sort the number of values to be processed are reduced in each step

No. of comparisons in step 1=n-1

No. of comparisons in step 2=n-2

.

.

.

No. of comparisons in last step=1

Hence

F(n)=n(n-1)/2

=(n^2-n)/2

=(n^2)/2-n/2

Hence Big O notation is given by

F(n)= O(n^2)

**Insertion Sort**

**Worst Case Analysis**

In case of Insertion sort the number of values to be processed are increased in each step

No. of comparisons in step 1= 1

No. of comparisons in step 2= 2

.

.

.

No. of comparisons in last step= n-1

Hence

F(n)=n(n-1)/2

=(n^2-n)/2

=(n^2)/2-n/2

Hence Big O notation is given by

F(n)= O(n^2)

**Best Case Analysis**

In case of Insertion sort the number of values to be processed are same in each step

No. of comparisons in step 1= 1

No. of comparisons in step 2= 1

.

.

.

No. of comparisons in last step=1

Hence

F(n) =1+1+1…. n-1 times

= n-1

Hence Big O notation is given by

F(n)= O(n)

**Program of Selection sort:**

import java.io.\*;

import java.util.Scanner;

class ABC {

    static int a[];

    static int innerCount = 0, outerCount = 0;

    public static void insertion() {

        int key, i;

        for (int j = 1; j < a.length; j++) {

            key = a[j];

            i = j - 1;

            while (i >= 0 && a[i] > key) {

                a[i + 1] = a[i];

                i = i - 1;

                innerCount++;

            }

            a[i + 1] = key;

            outerCount++;

        }

    }

    public static void selection() {

        int min;

        for (int i = 0; i < a.length; i++) {

            min = i;

            for (int j = i + 1; j < a.length; j++) {

                if (a[j] < a[min]) {

                    min = j;

                }

                innerCount++;

            }

            if (min != i) {

                int temp = a[min];

                a[min] = a[i];

                a[i] = temp;

            }

            outerCount++;

        }

    }

    // Driver code

    public static void main(String[] args) {

        Scanner ob = new Scanner(System.in);

        System.out.println("Enter number of elements: ");

        int n = ob.nextInt();

        System.out.println("\n1.Best case\n2.Worst case\nEnter choice from 1-2: ");

        int chnum = ob.nextInt();

        a = new int[n];

        if (chnum == 1) {

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

                a[i] = i + 1;

            }

        } else {

            int j = n;

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

                a[i] = j;

                j--;

            }

        }

        System.out.print("\n1.Insertion\n2.Selection\nEnter the coice from 1-2: ");

        int ch = ob.nextInt();

        switch (ch) {

            case 1:

                insertion();

                System.out.println("The output via insertion sort");

                break;

            case 2:

                selection();

                System.out.println("The output via selection sort");

                break;

            default:

                System.out.println("Incorrect choice");

                System.exit(0);

        }

        for (int i = 0; i < n; i++) {

            System.out.print(a[i] + " ");

        }

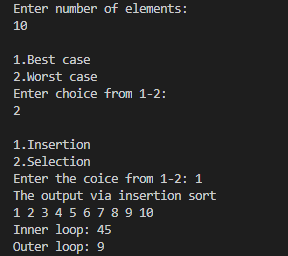
        System.out.println("\nInner loop: " + innerCount);

        System.out.println("Outer loop: " + outerCount);

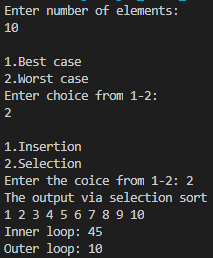
    }

}

**Insertion:**



**Selection:**



**Results:**

**Time Complexity of Selection sort:**

**Worst Case Analysis:**

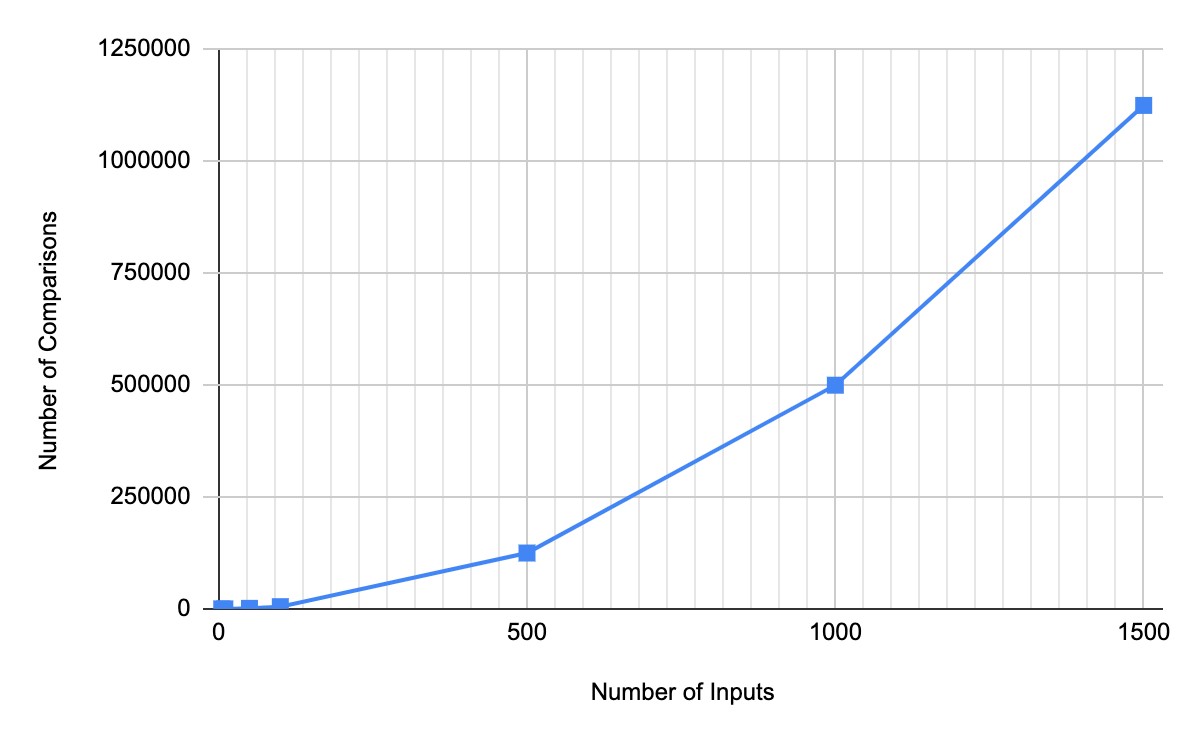
|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Input size** | **No: of steps from Algorithm analysis** | **No: of steps from Theoretical analysis** |
| 1 | 5 | 10 | 10 |
| 2 | 10 | 45 | 45 |
| 3 | 50 | 1225 | 1225 |
| 4 | 100 | 4950 | 4950 |
| 5 | 500 | 124750 | 124750 |
| 6 | 1000 | 499500 | 499500 |
| 7 | 1500 | 1124250 | 1124250 |

**Best Case Analysis:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Input size** | **No: of steps from Algorithm analysis** | **No: of steps from Theoretical analysis** |
| 1 | 5 | 10 | 10 |
| 2 | 10 | 45 | 45 |
| 3 | 50 | 1225 | 1225 |
| 4 | 100 | 4950 | 4950 |
| 5 | 500 | 124750 | 124750 |
| 6 | 1000 | 499500 | 499500 |
| 7 | 1500 | 1124250 | 1124250 |

**GRAPH:**

Since the Best Case and the Worst-Case Complexity of Selection Sort is similar, the number of comparisons is also similar and so will be the graph.

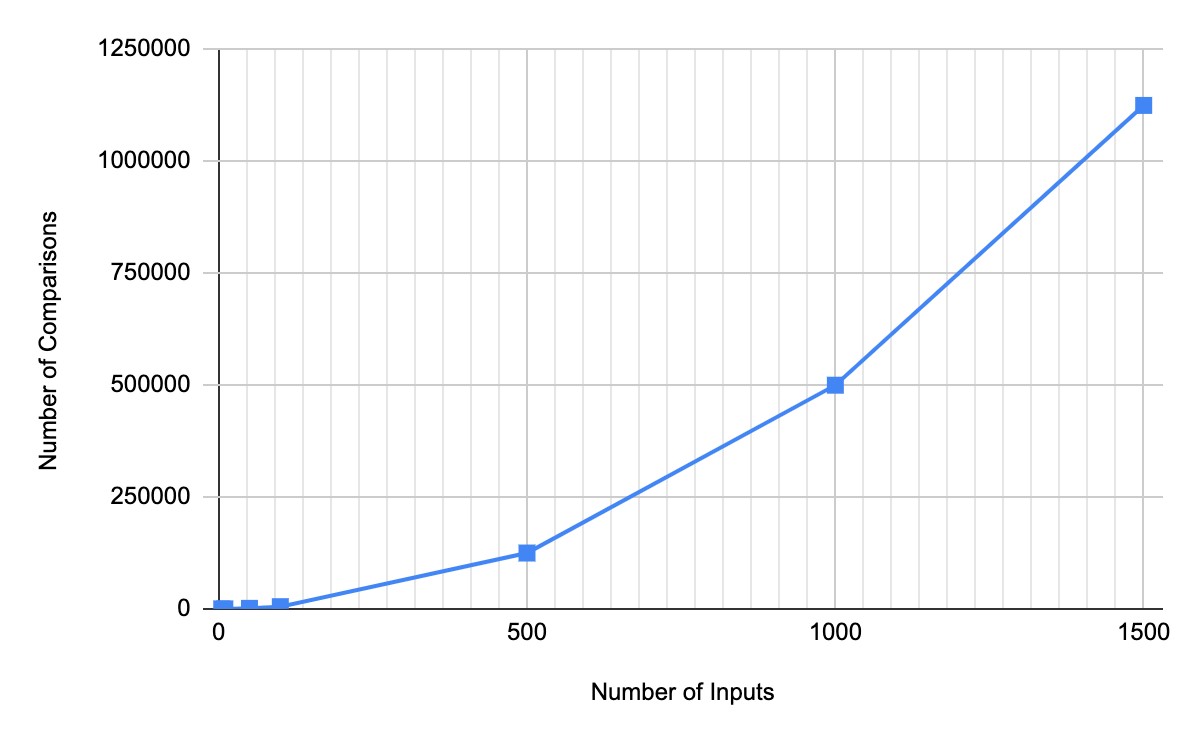


**Time Complexity of Insertion sort:**

**Worst Case Analysis:**

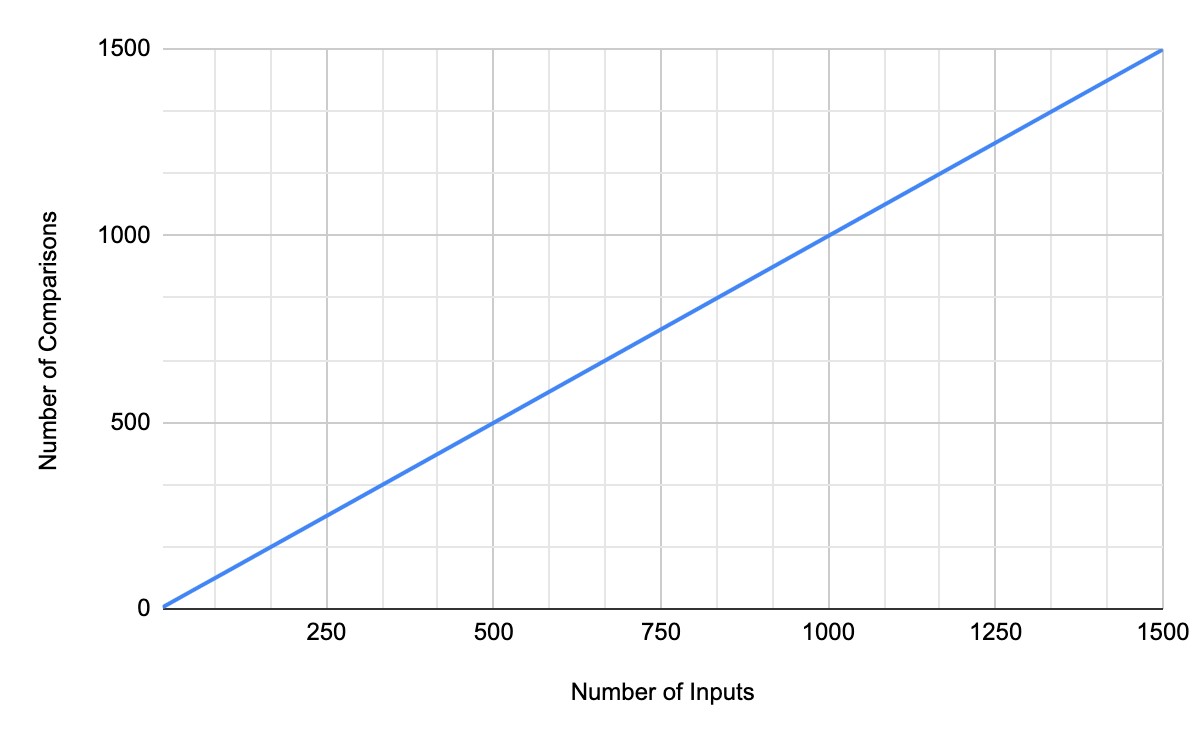
|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Input size** | **No: of steps from Algorithm analysis** | **No: of steps from Theoretical analysis** |
| 1 | 5 | 10 | 10 |
| 2 | 10 | 45 | 45 |
| 3 | 50 | 1225 | 1225 |
| 4 | 100 | 4950 | 4950 |
| 5 | 500 | 124750 | 124750 |
| 6 | 1000 | 499500 | 499500 |
| 7 | 1500 | 1124250 | 1124250 |

**GRAPH:**



**Best Case Analysis:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Input size** | **No: of steps from Algorithm analysis** | **No: of steps from Theoretical analysis** |
| 1 | 5 | 4 | 4 |
| 2 | 10 | 9 | 9 |
| 3 | 50 | 49 | 49 |
| 4 | 100 | 99 | 99 |
| 5 | 500 | 499 | 499 |
| 6 | 1000 | 999 | 999 |
| 7 | 1500 | 1499 | 1499 |



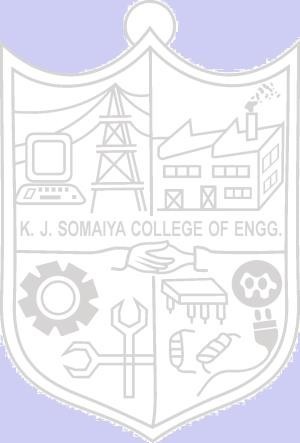
**Conclusion: (Based on the observations):**

We understand the implementation of Selection Sort and Insertion Sort and learn about the Time Complexity of Best Case and Worst Case and also, the nature of the Time Complexity.

**Outcome:**

**CO1:** Analyze time and space complexity of algorithms**.**

**References:**

1. Richard E. Neapolitan, " Foundation of Algorithms ", 5th Edition 2016, Jones & Bartlett Students Edition
2. T.H. Coreman ,C.E. Leiserson,R.L. Rivest, and C. Stein, " Introduction to algorithms", 3rd Edition 2009, Prentice Hall India Publication