K. J. Somaiya College of Engineering

(A Constituent College of Somaiya Vidyavihar University)

Department of Computer Engineering

Batch: A1	**Roll No.: 16010120015**
Experiment No. 1	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Title: Implementation of selection sort/ Insertion sort

Objective: To analyse performance of sorting methods

CO to be achieved:

CO 1 Analyze the asymptotic running time and space complexity of algorithms.

Books/ Journals/ Websites referred:

1. **Ellis horowitz, Sarataj Sahni, S.Rajsekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algortihtms",2nd Edition ,MIT press/McGraw Hill,2001**
3. **http://en.wikipedia.org/wiki/Insertion_sort**
4. **http://www.sorting-algorithms.com/insertion-sort**
5. **http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Insertion_sort.html**
6. **http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/insertionSort.htm**
7. **http://en.wikipedia.org/wiki/Selection_sort**
8. **http://www.sorting-algorithms.com/selection-sort**
9. **http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/selectionSort.htm**
10. **http://courses.cs.vt.edu/~csonline/Algorithms/Lessons/SelectionCardSort/selectioncardsort.html**

Pre Lab/ Prior Concepts:
Data structures, sorting techniques

Historical Profile:
There are various methods to sort the given list. As the size of input changes, the performance of these strategies tends to differ from each other. In such a case, the priori analysis can help the engineer to choose the best algorithm.

New Concepts to be learned:
Space complexity, time complexity, size of input, order of growth.

Algorithm Insertion Sort

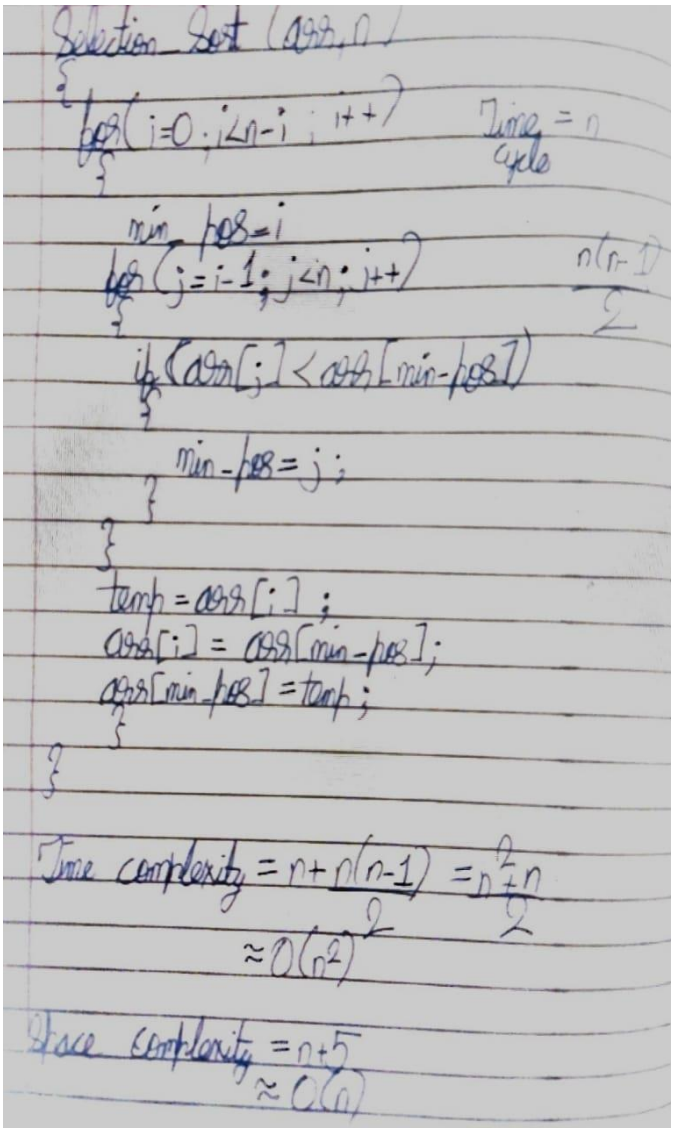
```
INSERTION_SORT (A,n)
//The algorithm takes as parameters an array A[1.. n] and the length n of the array.
//The array A is sorted in place: the numbers are rearranged within the array
// A[1..n] of eletype, n: integer

  FOR j ← 2 TO length[A]
      DO  key ← A[j]
          {Put A[j] into the sorted sequence A[1 . . j − 1]}
              i ← j − 1
              WHILE i > 0 and A[i] > key
                  DO A[i +1] ← A[i]
                        i ← i − 1
               A[i + 1] ← key
```

Algorithm Selection Sort

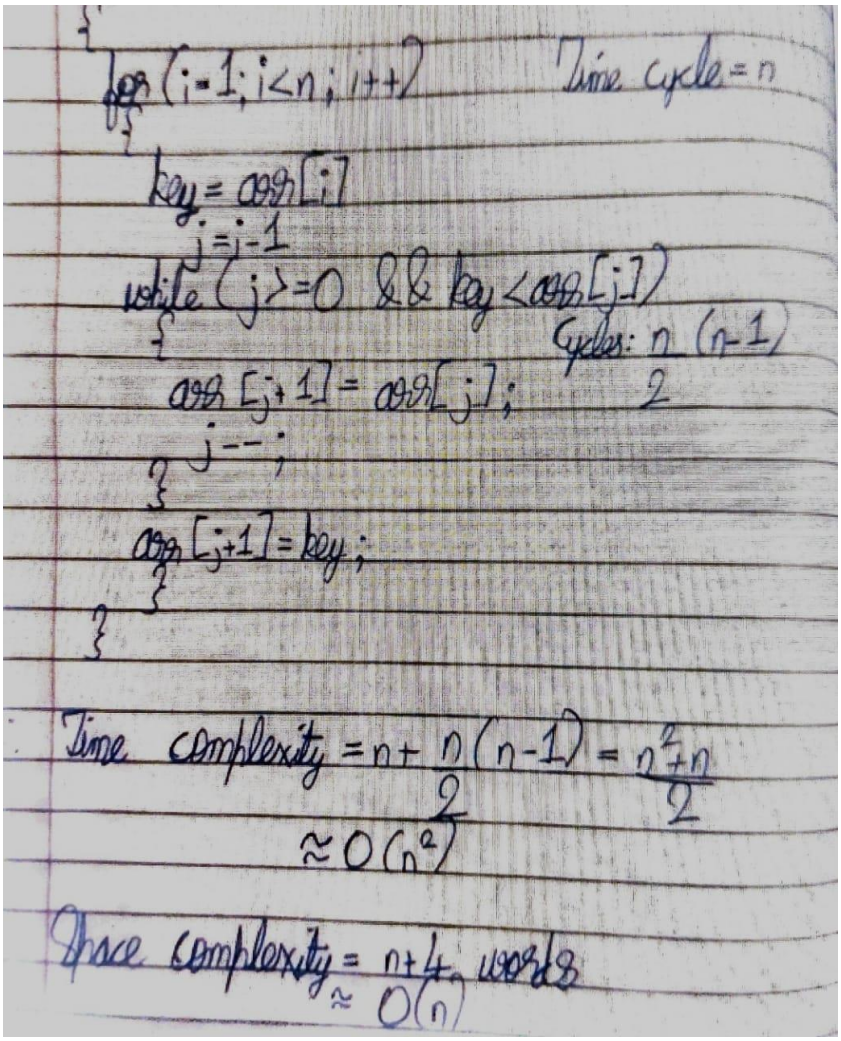
```
SELECTION_SORT (A,n)
//The algorithm takes as parameters an array A[1.. n] and the length n of the array.
//The array A is sorted in place: the numbers are rearranged within the array
// A[1..n] of eletype, n: integer

          FOR i ← 1 TO n-1 DO
             min j ← i;
             min x ← A[i]
            FOR j ← i + 1 to n do
               IF A[j] < min x then
                  min j ← j
                  min x ← A[j]
```

A[min j] ← A [i]
A[i] ← min x

Time and space complexity for selection sort



Time and space complexity for insertion sort


for(i=1; i<n; i++) Time cycle = n
key = arr[i]
j = j-1
while(j>=0 && key < arr[j]) Cycles: n(n-1)/2
arr[j+1] = arr[j];
j--;
arr[j+1] = key;
Time complexity = n + n(n-1)/2 = (n^2+n)/2
≈ O(n^2)
Space complexity = n+4 words
≈ O(n)

IMPLEMENTATION DETAILS

1. Selection Sort

```
import java.util.*;
import java.util.Random;
public class Main
{
    static void SelectionSort(int arr[]){
        int min_index;
        for(int i=0;i<arr.length - 1;i++){
            min_index = i;
            for(int j=i+1;j<arr.length;j++){
                if(arr[i]>arr[j]){
                    int temp = arr[i];
                    arr[i]=arr[j];
                    arr[j]=temp;
                }
            }
        }
    }

   public static void main(String[] args) {

       System.out.println("************************************************");


System.out.println(" -- 1.SELECTION SORT --");
```

```
        Random random = new Random();
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the size of the array : ");

        int n = s.nextInt();
        int arr[] = new int[n];
        for(int i=0;i<arr.length;i++){
            arr[i]= random.nextInt(1000);
        }

        long t1 = System.nanoTime();
                    SelectionSort(arr);
        long t2 = System.nanoTime();

        System.out.printf("\ntime taken : %d nano seconds",(t2-t1));

        System.out.println("\n**********************************************");
    }
}
```

OUTPUT

TEST CASE 1

```
************************************************
 -- 1.SELECTION SORT --
Enter the size of the array : 5

time taken : 9700 nano seconds
************************************************
```

TEST CASE 2

```
************************************************
 -- 1.SELECTION SORT --
Enter the size of the array : 10

time taken : 10200 nano seconds
************************************************
```

TEST CASE 3

```
************************************************
 -- 1.SELECTION SORT --
Enter the size of the array : 100000

time taken : 10000737300 nano seconds
************************************************
```

2. Insertion Sort

```
import java.util.*;
import java.util.Random;
public class Main
{
   static void insertionSort(int arr[]) {
```

```
    int key;
    int j;
    for (int i = 1; i < arr.length; i++) {
        key = arr[i];
        j = i- 1;
        while (j >= 0 && key < arr[j]) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

public static void main(String[] args) {

    System.out.println("************************************************");
    System.out.println(" -- 2.INSERTION SORT --");



    Random random = new Random();
    Scanner s = new Scanner(System.in);
    System.out.print("Enter the size of the array : ");
    int n = s.nextInt();
    int arr[] = new int[n];
    for(int i=0;i<arr.length;i++){
        arr[i]= random.nextInt(1000);
    }
    long t1 = System.nanoTime();
```

```
        insertionSort(arr);

        long t2 = System.nanoTime();

        System.out.printf("\ntime taken : %d nano seconds",(t2-t1));

        System.out.println("\n************************************************");
    }
}
```

OUTPUT

TEST CASE 1

```
************************************************
 -- 2.INSERTION SORT --
Enter the size of the array : 5

time taken : 5000 nano seconds
************************************************
```

TEST CASE 2

```
************************************************
 -- 2.INSERTION SORT --
Enter the size of the array : 10

time taken : 6700 nano seconds
************************************************
```

TEST CASE 3

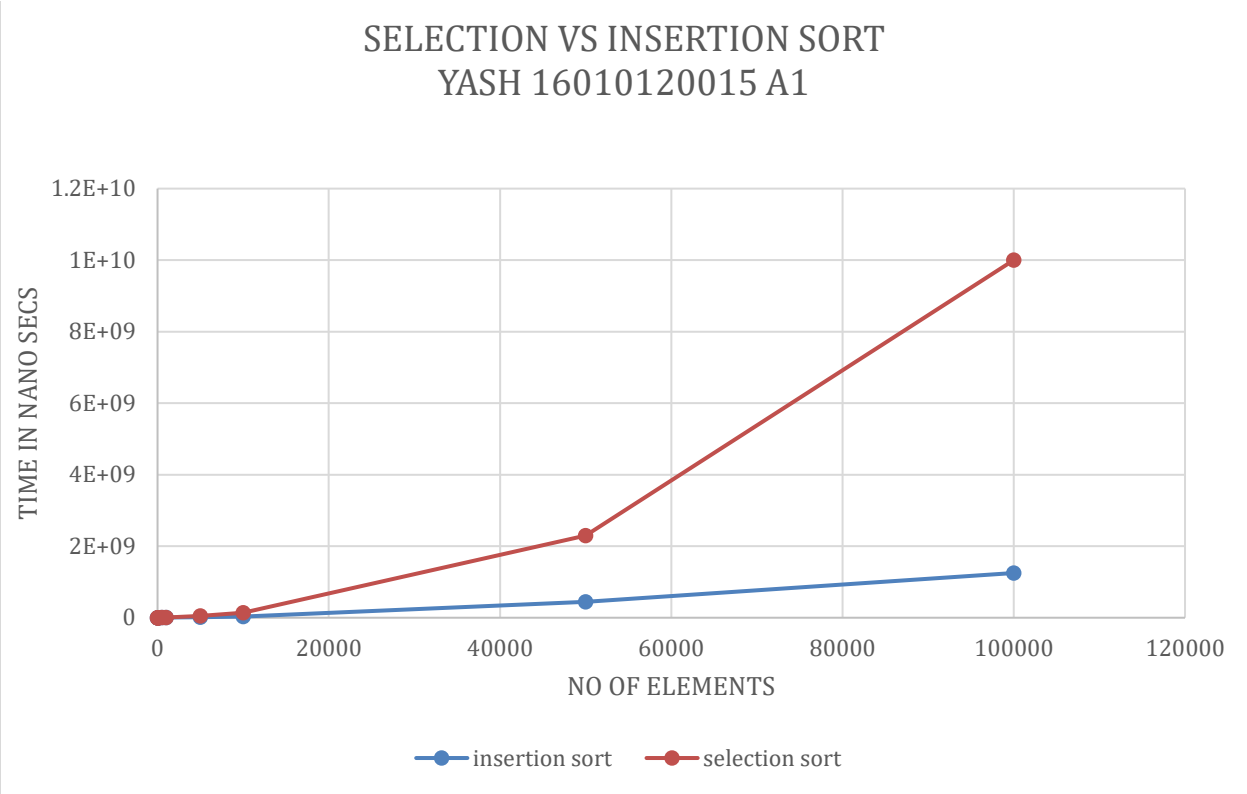
```
************************************************
 -- 2.INSERTION SORT --
Enter the size of the array : 100000

time taken : 1250669700 nano seconds
************************************************
```

GRAPHS FOR VARYING INPUT SIZES: (INSERTION SORT & SELECTION SORT)

DATA

n	insertion sort	selection sort
5	5000	9700
10	6700	10200
50	38200	95300
100	158400	926200
500	2807000	4053900
1000	9224800	8636700
5000	16769700	52250800
10000	29674100	141711900
50000	448082500	2294461100
100000	1250669700	10000737300

GRAPH



Conclusion:

By this experiment we were able to learn, understand and implement the following concepts:

- ✓ **Implementation of Selection and Insertion Sort in Java programming language**
- ✓ **Time and Space complexity of both the sorting algorithms**
- ✓ **Graphs for varying input sizes (insertion sort & selection sort)**