K. J. Somaiya College of Engineering

(A Constituent College of Somaiya Vidyavihar University)

Batch: A4 Roll No.: 1911050
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:

Sr. No	Objective
CO 1	Analyze the asymptotic running time and space complexity of algorithms.
CO 2	Describe various algorithm design strategies to solve different problems and analyse Complexity.
CO 3	Develop string matching techniques
CO 4	Describe the classes P, NP, and NP-Complete

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 case, the priori analysis can helps the engineer to choose the best algorithm.

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

Algorithm InsertionSort
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 SelectionSort

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 $i \leftarrow 1$ **TO** n-1 **DO**
min $j \leftarrow i$;
min $x \leftarrow$ A[i]
FOR $j \leftarrow i + 1$ to n do
IF A[j] < min x then
min $j \leftarrow j$
min $x \leftarrow$ A[j]
A[min j] $\leftarrow$ A [i]
A[i] $\leftarrow$ min x

Implementation details

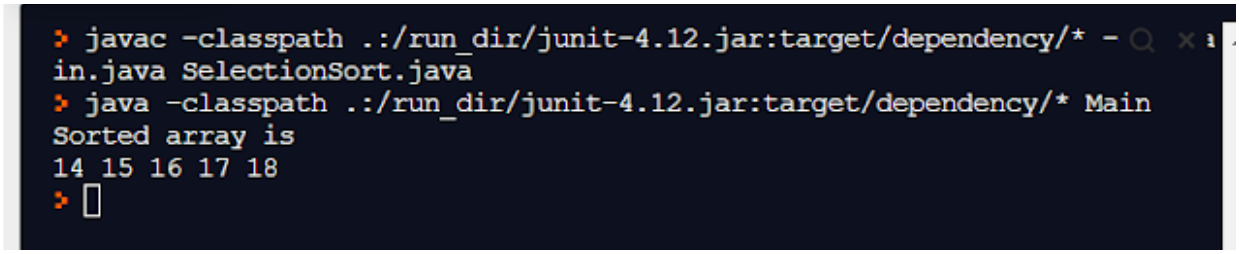
Code:

```
import java.io.*;

class Main

{

   void sort(int arr[])

   {

       int n = arr.length;

       for (int i = 0; i < n-1; i++)

       {
```

```
            int index = i;
            for (int j = i+1; j < n; j++)
                if (arr[j] < arr[index])
                  index = j;
            int temp = arr[index];
            arr[index] = arr[i];
            arr[i] = temp;
        }
    }
    void print(int arr[])
    {
        int n = arr.length;
        for (int i=0; i<n; ++i)
            System.out.print(arr[i]+" ");
        System.out.println();
    }
    public static void main(String args[])
    {
        Main obj = new Main();
        int arr[] = {18,16,15,17,14};
        obj.sort(arr);
        System.out.println("Sorted array is");
```

```
        obj.print(arr);
    }
}
```

OUTPUT



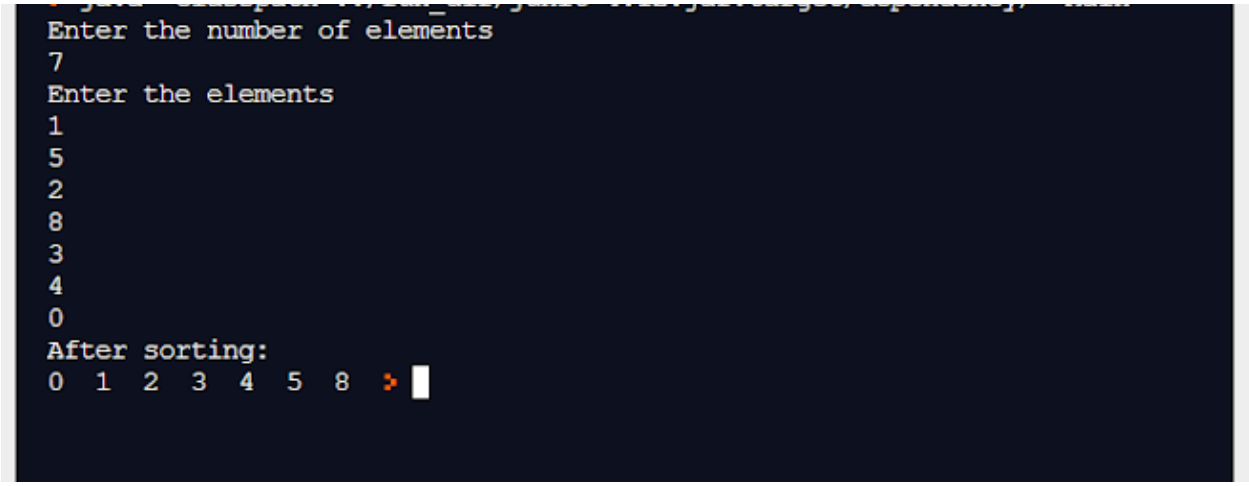
Code:

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

      for (int i = 1; i < size; i++)
      {
         value = arr[i];
```

```
        index = i;


        while (index > 0 && arr[index-1] > value)
        {
           arr[index] = arr[index-1];
           index--;
        }
        arr[index] = value;
    }
}
        public static void main(String[] args)
        {
                Scanner s=new Scanner(System.in);
                System.out.println("Enter the number of elements");
                int size=s.nextInt();
                int[] arr=new int[size];
                System.out.println("Enter the elements");
                for(int i=0;i<size;i++)
                {
                   arr[i]=s.nextInt();
                }
                insertionSort(arr,size);
                System.out.println("After sorting:");
```

```
        for(int i=0;i<size;i++)
        {
           System.out.print(arr[i]+"  ");
        }
    }
}
```

Output:



The space complexity of Insertion sort:

The space complexity of Selection sort:

Time complexity for Insertion sort:

Time complexity for selection sort:

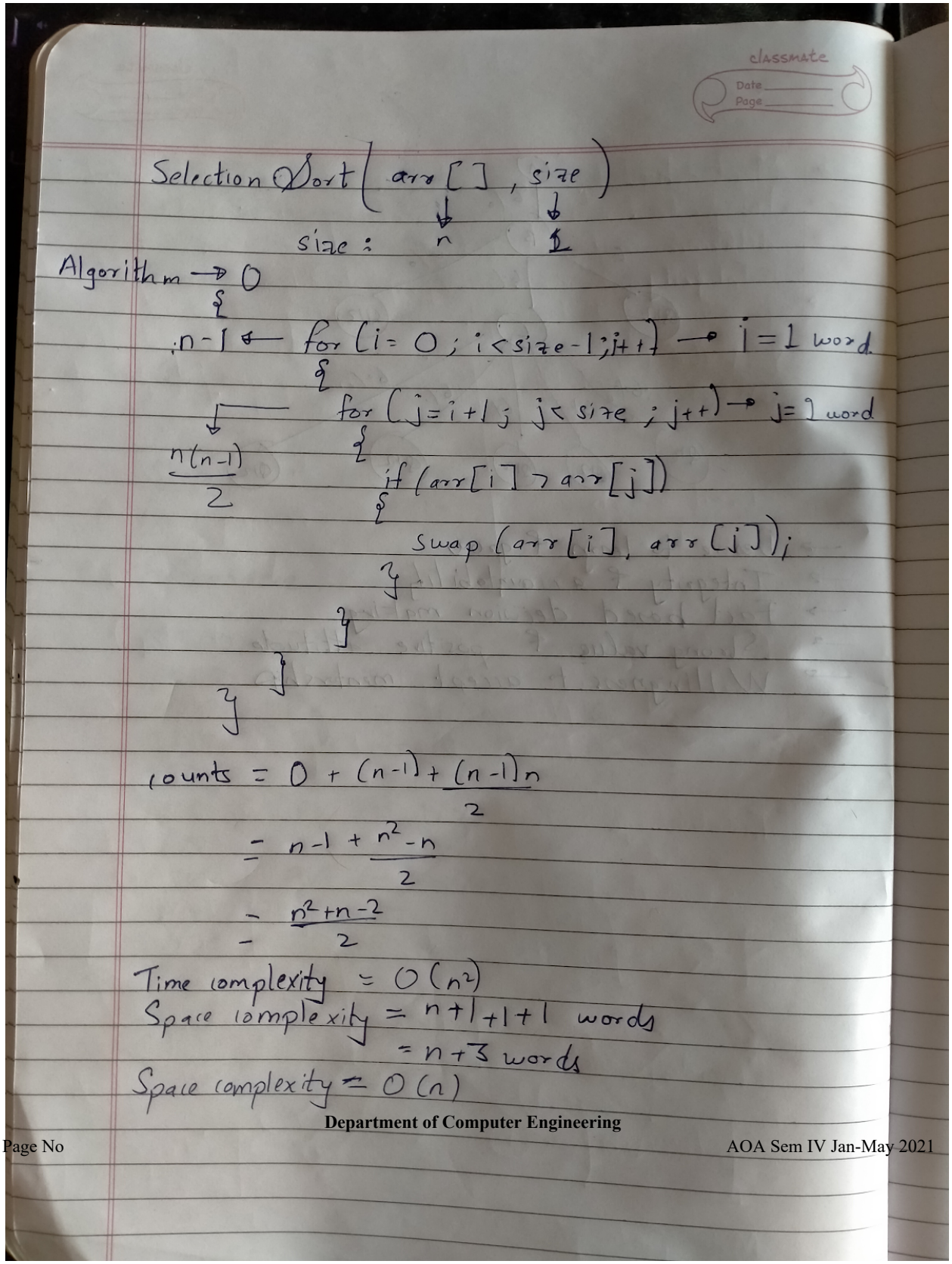
```
insertionSort (int arr[], int size)
{
    int value, index
    for (int i = 1; i < size; i++)          n-1
    {
        value = arr[i];
        index = i;
        while (index > 0 && arr[index-1] > value)
        {
            arr[index] = arr[index-1];
            index--;
        }
        arr[index] = value;
    }
}
```

Time Complexity:

1) Best case:
If size of array is n then if it is already sorted it will require only 1 pass
∴ time complexity = O(n)

2) Worst case:
If the array is arranged in descending order initially Total no of comparisions → $(n-1)+(n-2)+\cdots+3+2+1$
$= \frac{(n-1)n}{2} = \frac{n^2-n}{2} \approx O(n^2)$

3) Space complexity: arr[] → n words
i, size, index, value → 1 word each
$\Rightarrow n+4 \approx O(n)$



```
Selection Sort( arr[], size )
                 ↓     ↓
size :           n     1

Algorithm → 0
{
n-1 ← for(i=0; i<size-1; i++) → i=1 word
{
      for(j=i+1; j<size; j++) → j=1 word
      {
n(n-1)/2    if(arr[i] > arr[j])
        {
          swap(arr[i], arr[j]);
        }
      }
}
```

counts = $0 + (n-1) + \frac{(n-1)n}{2}$

$= n-1 + \frac{n^2-n}{2}$

$= \frac{n^2+n-2}{2}$

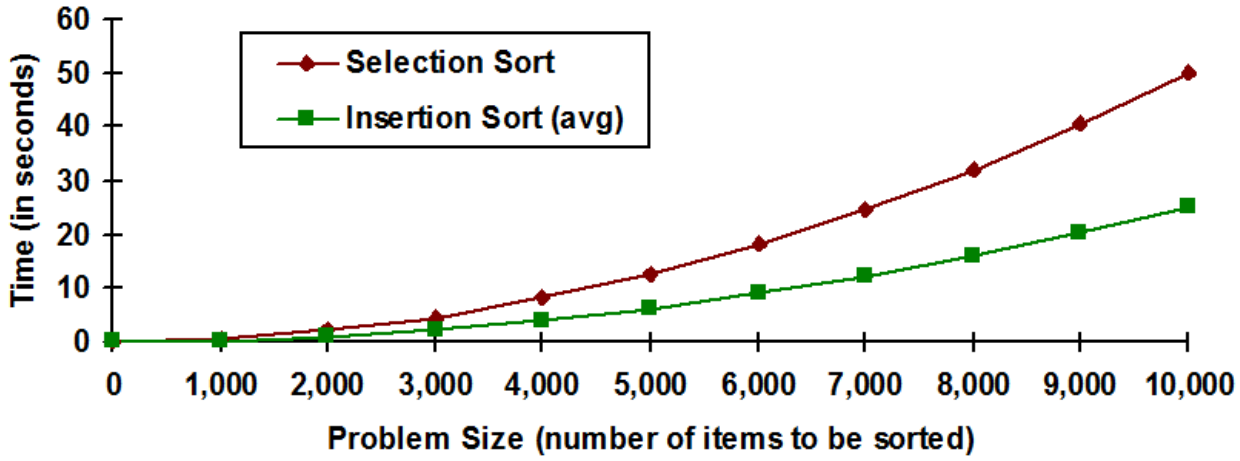
Time complexity = $O(n^2)$

Space complexity = $n+1+1+1$ words

$= n+3$ words

Space complexity = $O(n)$

Graphs for varying input sizes: (Insertion Sort & Selection sort)



CONCLUSION: We have successfully analysed both the sorting algorithms given in the experiment.