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

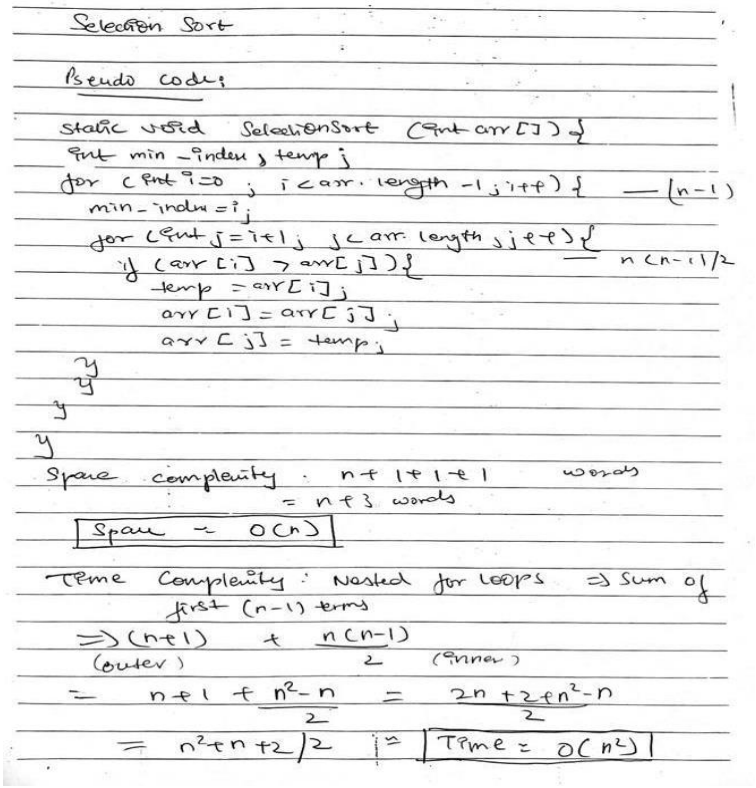
```
INSERTION_SORT (A,n)
//The algorithm takes as parameters an array A[1.. n] and the length n of the array.
//The array A is sorted in place: the numbers are rearranged within the array
// A[1..n] of eletype, n: integer

  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

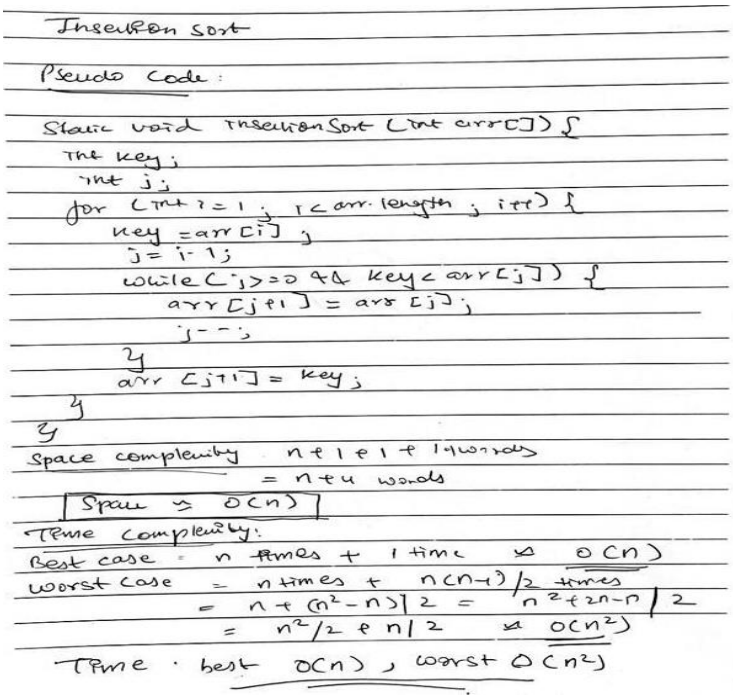
```
SELECTION_SORT (A,n)
//The algorithm takes as parameters an array A[1.. n] and the length n of the array.
//The array A is sorted in place: the numbers are rearranged within the array
// A[1..n] of eletype, n: integer

          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



IMPLEMENTATION DETAILS

For random array:

```
import random
import time
def insertion(arr):

    for i in range(1,len(arr)):
        key=arr[i]
        j=i-1
        while j>=0 and key<arr[j]:
            arr[j+1]=arr[j]
            j-=1
        arr[j+1]=key

def selection(arr):

    for i in range(len(arr)):
        min=i
        for j in range(i+1,len(arr)):
            if arr[min]>arr[j]:
                min=j


        arr[i],arr[min] = arr[min],arr[i]
```

```
array=[]
a = int(input("Enter number of elements in array: "))
for n in range(a):
    x= random.randint(0,a)
    array.append(x)
t1=time.time()
#selection(array)
#insertion(array)
t2=time.time()
print("Time taken is: ",1000*(t2-t1))
```

For selection:

```
Enter number of elements in array: 4000
Time taken is:  515.1159763336182

Process finished with exit code 0
```

For insertion:

```
Enter number of elements in array: 4000
Time taken is:  599.2879867553711

Process finished with exit code 0
```

SORTED ARRAY:

Code:

```
import random
import time
def insertion(arr):

    for i in range(1,len(arr)):
        key=arr[i]
        j=i-1
        while j>=0 and key<arr[j]:
            arr[j+1]=arr[j]
            j-=1
```

```
        arr[j+1]=key


def selection(arr):

    for i in range(len(arr)):
        min=i
        for j in range(i+1,len(arr)):
            if arr[min]>arr[j]:
                min=j


        arr[i],arr[min] = arr[min],arr[i]




array=[]
a = int(input("Enter number of elements in array: "))
fac=100
for n in range(a):
    x= random.randint(0,a) + fac*n
    array.append(x)
t1=time.time()
#selection(array)
#insertion(array)
t2=time.time()
print("Time taken is: ", 1000 * (t2 - t1))
```

Selection Sort:

```
Enter number of elements in array: 4000
Time taken is:  507.1120262145996

Process finished with exit code 0
```

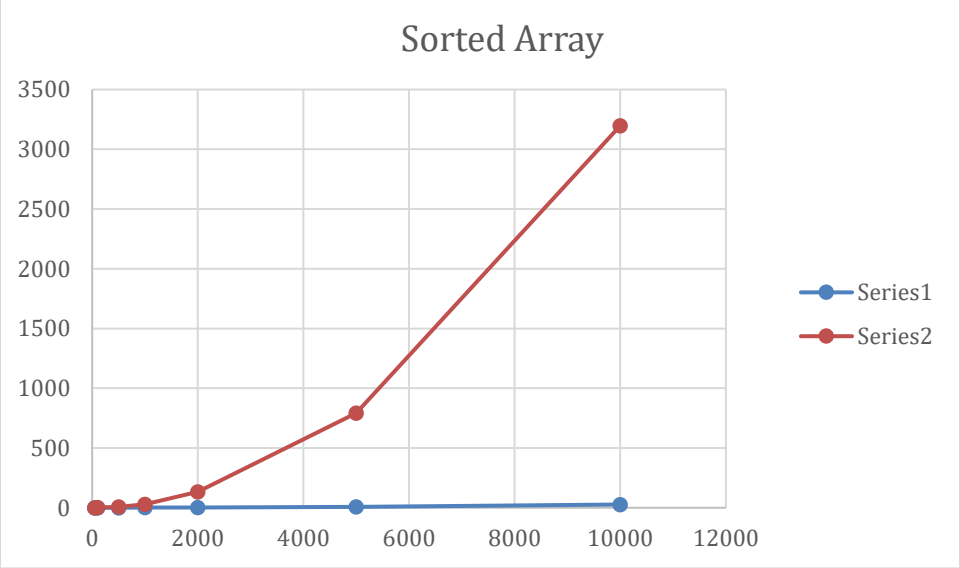
Insertion Sort:

```
Enter number of elements in array: 4000
Time taken is:  5.575895309448242

Process finished with exit code 0
```

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

No. of Elements	Time Taken in Nanoseconds Insertion Sort	Selection Sort
50	0	0
100	0	1.002
500	8.006	4.999
1000	28.005	20.004
2000	92.018	87.895
5000	635.144	541.52
10000	2349.621	2142.039

No. of elements	Time taken in nanoseconds For Sorted Insertion Sort	Selection Sort
50	0	0
100	0	1
500	0	7.016
1000	1.01	30.012
2000	2.005	133.837
5000	7.522	792.966
10000	26.986	3197.286



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)**