**K. J. Somaiya College of Engineering, Mumbai-77**

**(Autonomous College Affiliated to University of Mumbai)**



|  |
| --- |
| **Title: Implementation of selection sort/ Insertion sort** |



**Objective:** To analyse performance of sorting methods



**CO to be achieved:**

|  |  |
| --- | --- |
| Sr. No | Objective |
| CO 1 | Compare and demonstrate the efficiency of algorithms using asymptotic complexity notations. |
| CO 2 | Analyze and solve problems for divide and conquer strategy, greedy method, dynamic programming approach and backtracking and branch & bound policies. |
| CO 3 | Analyze and solve problems for   different string matching algorithms. |



**Books/ Journals/ Websites referred:**

1. **Ellis horowitz, SaratajSahni, 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**](http://en.wikipedia.org/wiki/Insertion_sort)
4. [**http://www.sorting-algorithms.com/insertion-sort**](http://www.sorting-algorithms.com/insertion-sort)
5. [**http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Insertion\_sort.html**](http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Insertion_sort.html)
6. [**http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/insertionSort.htm**](http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/insertionSort.htm)
7. [**http://en.wikipedia.org/wiki/Selection\_sort**](http://en.wikipedia.org/wiki/Selection_sort)
8. [**http://www.sorting-algorithms.com/selection-sort**](http://www.sorting-algorithms.com/selection-sort)
9. [**http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/selectionSort.htm**](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 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*

**The space complexity of Insertion sort:**

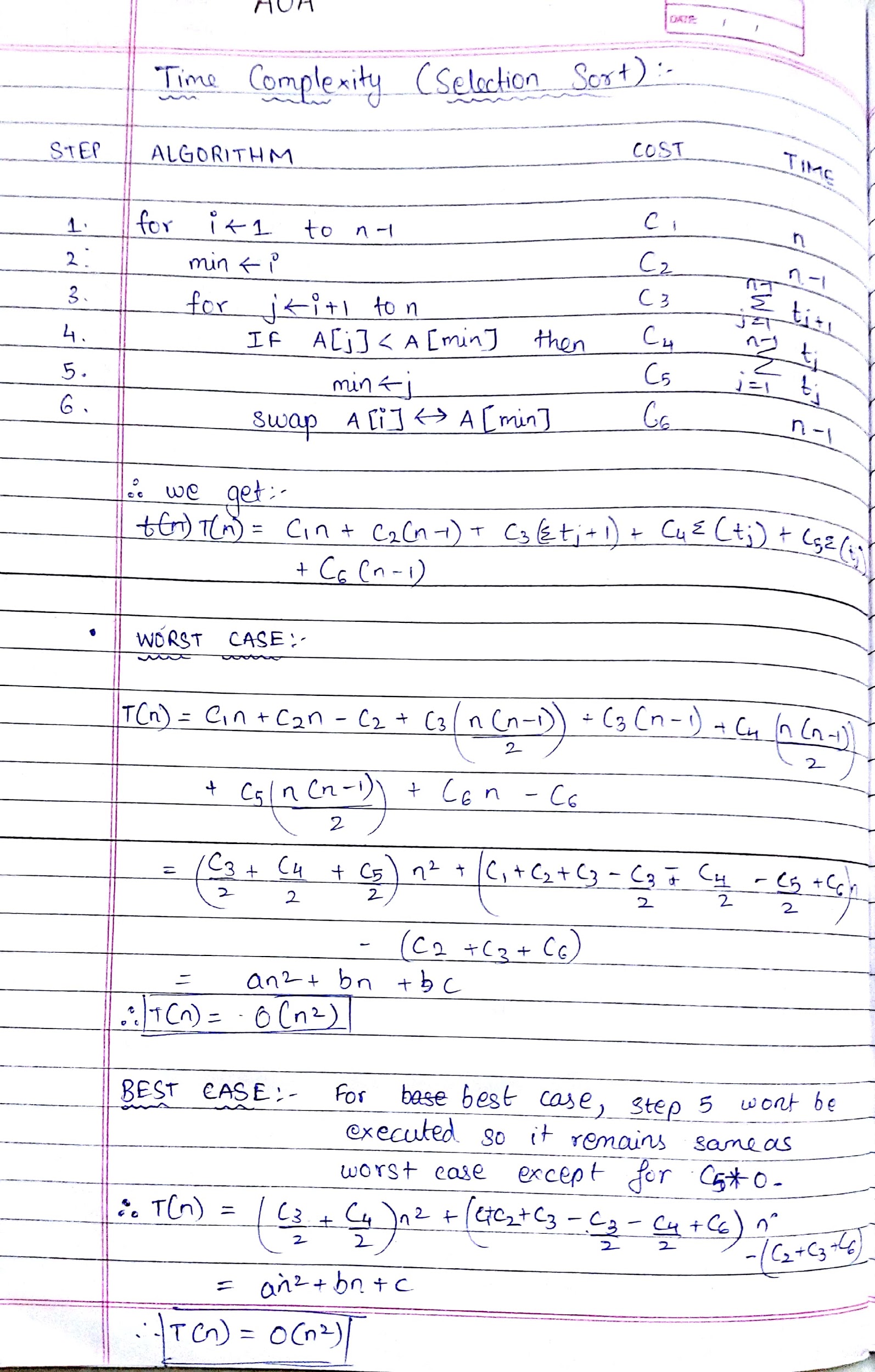
The space complexity of Insertion Sort is O(1).

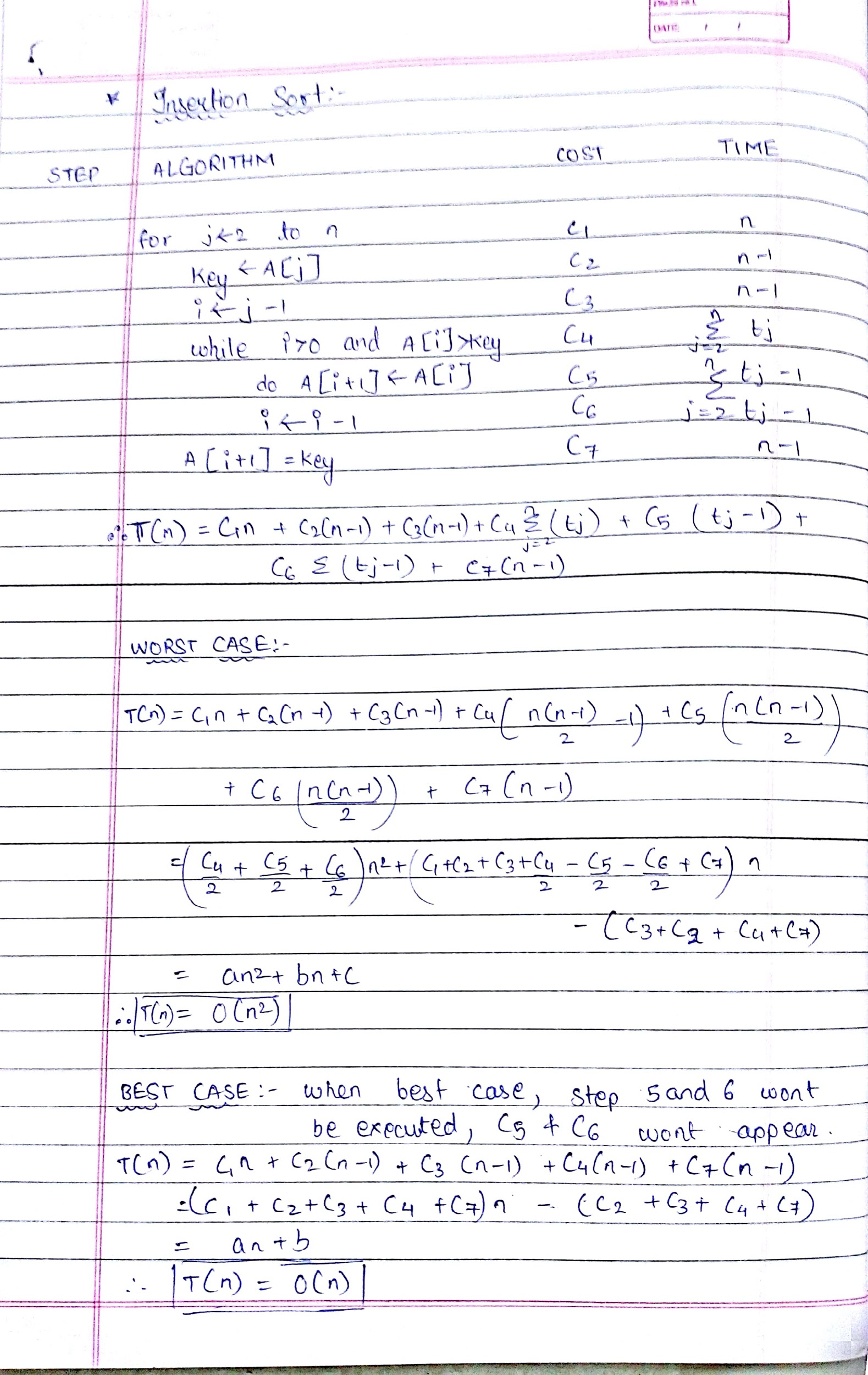
**The space complexity of Selection sort:**

The space complexity of Selection Sort is O(1).

**Time complexity for Insertion sort and Selection Sort:**

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Data Structure** | **Time Complexity** |
|  |  | **Best** | **Average** | **Worst** |
| [Insertion Sort](http://scanftree.com/Data_Structure/Insertion-sort) | Array | O(n) | O(n^2) | O(n^2) |
| [Select Sort](http://en.wikipedia.org/wiki/Selection_sort) | Array | O(n^2) | O(n^2) | O(n^2) |

****

****

**Implementation Code:**

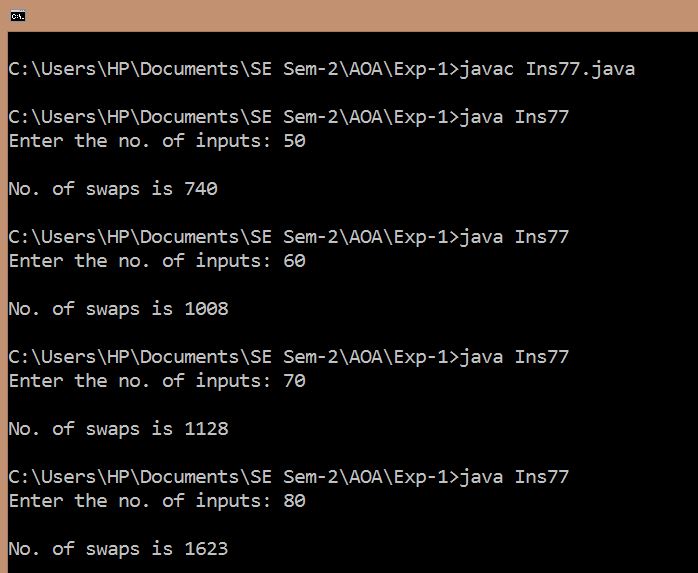
Insertion Sort

import java.util.\*;  
  
class Ins77  
{  
 public static void main(String[] args)  
 {  
 Scanner sc=new Scanner(System.in);  
 Random rand=new Random();  
 int i,j,temp,count=0;  
 System.out.print("Enter the no. of inputs: ");  
 int n=sc.nextInt();  
 int a[]=new int[n];  
 for(i=0;i<n;i++)  
 {  
 a[i]=rand.nextInt(100);  
 }  
 for(i=1;i<n;i++)   
 {   
 temp=a[i];  
 j=i-1;  
 while (j>=0 && a[j]>temp)  
 {  
 a[j+1]=a[j];  
 count++;  
 j=j-1;  
 }  
 a[j+1]=temp;  
 }  
 System.out.println(" ");  
 for(i=0;i<n;i++)  
 {  
 System.out.print(a[i]+" ");  
 }  
 System.out.println("\n No. of swaps is "+count);  
 }  
}

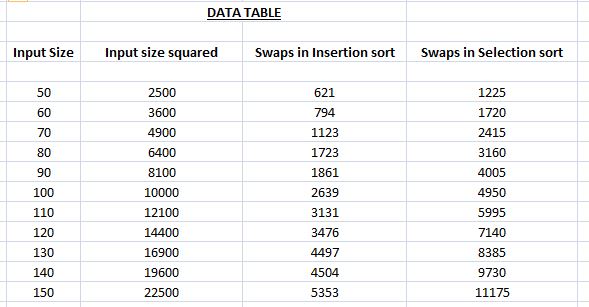
Selection Sort

import java.util.\*;  
  
class Sel77  
{  
 public static void main(String[] args)  
 {  
 Scanner sc=new Scanner(System.in);  
 Random rand=new Random();  
 int i,j,temp,s,count=0;  
 System.out.print("Enter the no. of inputs: ");  
 int n=sc.nextInt();  
 int a[]=new int[n];  
 for(i=0;i<n;i++)  
 {  
 a[i]=rand.nextInt(100);  
 }  
 for(i=0;i<n;i++)   
 {   
 s=i;  
 for(j=i+1;j<n;j++)  
 {  
 if(a[j]<a[s])  
 {  
 s=j;  
 }  
 count++;  
 }  
 temp=a[s];  
 a[s]=a[i];  
 a[i]=temp;  
 }  
 System.out.println(" ");  
 for(i=0;i<n;i++)  
 {  
 System.out.print(a[i]+" ");  
 }  
 System.out.println("\n No. of swaps is "+count);  
 }  
}

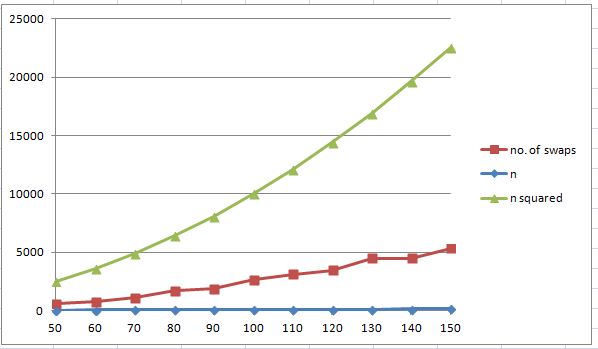
**Sample Output :**

****

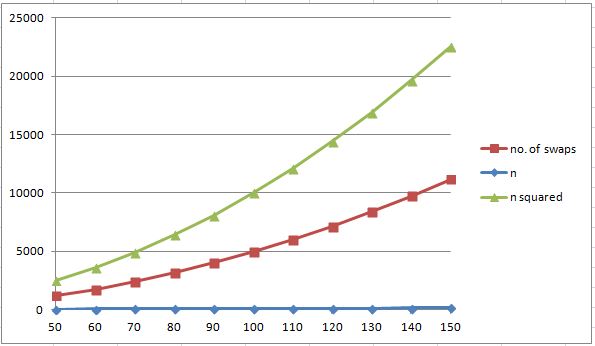
**Data Table:**

****

**Graph of Time Complexity for Insertion Sort:**

****

**Graph of Time Complexity for Selection Sort:**

****

**CONCLUSION:**

Therefore, the time Complexities of Selection Sort and Insertion Sort algorithm are successfully estimated by calculating number of comparisons or swaps made by the algorithm for various test cases each having different number of input sizes, by implementing the algorithms in Java.

In general, Insertion Sort shows better performance. Further, the number of swaps in both cases is less than n squared, where n is the input size, as seen from the graph.