Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department &
Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** DAA

Name	**Pratik Pujari**		
UID no.	**2020300054**	**Class:**	**Comps C Batch**
Experiment No.	1		

AIM:	To sort arrays using selection and insertion sort
EXPERIMENT 1	
THEORY:	**Sorting** A Sorting Algorithm is used to rearrange a given array or list elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of element in the respective data structure. In computer science, a sorting algorithm is an algorithm that puts elements of a list into an order. The most frequently used orders are numerical order and lexicographical order, and either ascending or descending. Efficient sorting is important for optimizing the efficiency of other algorithms (such as search and merge algorithms) that require input data to be in sorted lists. Sorting is also often useful for canonicalizing data and for producing human-readable output. Formally, the output of any sorting algorithm must satisfy two conditions: 1. The output is in monotonic order (each element is no smaller/larger than the previous element, according to the required order). 2. The output is a permutation (a reordering, yet retaining all of the original elements) of the input. **Selection sort**

Bharatiya Vidya Bhavan’s

Sardar Patel Institute of Technology

Bhavan’s Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department &
Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** DAA

Selection sort is an in-place comparison sort. It has O(n^2) complexity, making it inefficient on large lists, and generally performs worse than the similar insertion sort. Selection sort is noted for its simplicity, and also has performance advantages over more complicated algorithms in certain situations.
The algorithm finds the minimum value, swaps it with the value in the first position, and repeats these steps for the remainder of the list. It does no more than *n* swaps, and thus is useful where swapping is very expensive.

Time Complexity
In computer science, selection sort is an in-place comparison sorting algorithm. It has an O(n^2) time complexity, which makes it inefficient on large lists, and generally performs worse than the similar insertion sort.

- Worst case time complexity: Θ(N^2) comparisons and Θ(N) swaps
- Average case time complexity: Θ(N^2) comparisons and Θ(N) swaps
- Best case time complexity: Θ(N^2) comparisons and Θ(N) swaps
- Space complexity: Θ(1) auxillary space

Worst Case Time Complexity
The worst case is the case when the array is already sorted (with one swap) but the smallest element is the last element. For example, if the sorted number as a1, a2, ..., aN, then: a2, a3, ..., aN, a1 will be the worst case for our particular implementation of Selection Sort.
Worst Case: a2, a3, ..., aN, a1
The cost in this case is that at each step, a swap is done. This is because the smallest element will always be the last element and the swapped element which is kept at the end will be the second smallest element that is the smallest element of the new unsorted sub-array. Hence, the worst case has:

- N * (N+1) / 2 comparisons
- N swaps

Hence, the time complexity is O(N^2).

Best Case Time Complexity
The best case is the case when the array is already sorted.
For example, if the sorted number as a1, a2, ..., aN, then:
a1, a2, a3, ..., aN will be the best case for our particular implementation of Selection Sort.
This is the best case as we can avoid the swap at each step but the time spend to find the smallest element is still O(N).
Hence, the best case has:

- N * (N+1) / 2 comparisons
- 0 swaps

Average Case Time Complexity
Based on the worst case and best case, we know that the number of comparisons will be the same for every case and hence, for average case as well, the number of comparisons will be constant.
Number of comparisons = N * (N+1) / 2
Therefore, the time complexity will be O(N^2).
To find the number of swaps,

- There are N! different combination of N elements
- Only for one combination (sorted order) there is 0 swaps.
- In the worst case, a combination will have N swaps. There are several such combinations.
- Number of ways to select 2 elements to swap = nC2 = N * (N-1) / 2
- From sorted array, this will result in O(N^2) combinations which need 1 swap.

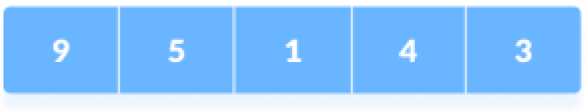
ALGO:

1. Call insert to insert the element that starts at index 1 into the sorted subarray in index 0.
2. Call insert to insert the element that starts at index 2 into the sorted subarray in indices 0 through 1.

3. Call insert to insert the element that starts at index 3 into the sorted subarray in indices 0 through 2.
4. …
5. Finally, call insert to insert the element that starts at index n-1$n-1$n, minus, 1 into the sorted subarray in indices 0 through n-2$n-2$n, minus, 2.

Working of Insertion Sort

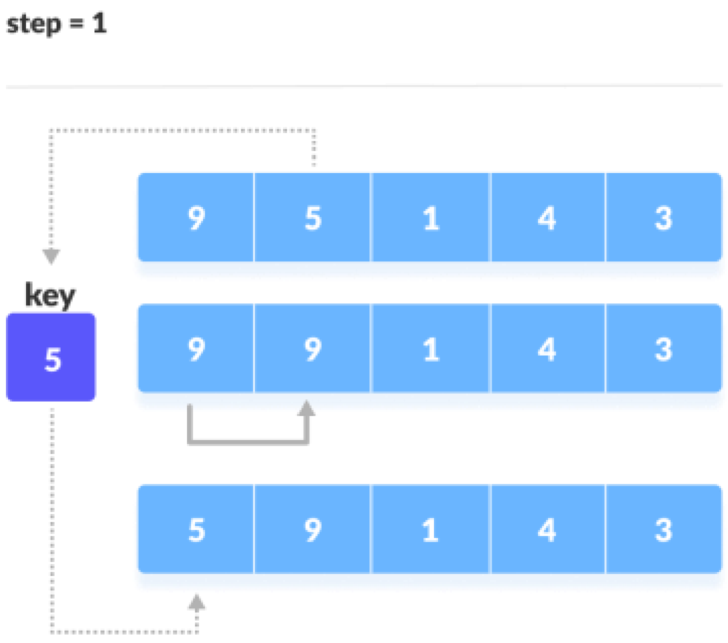
Suppose we need to sort the following array.

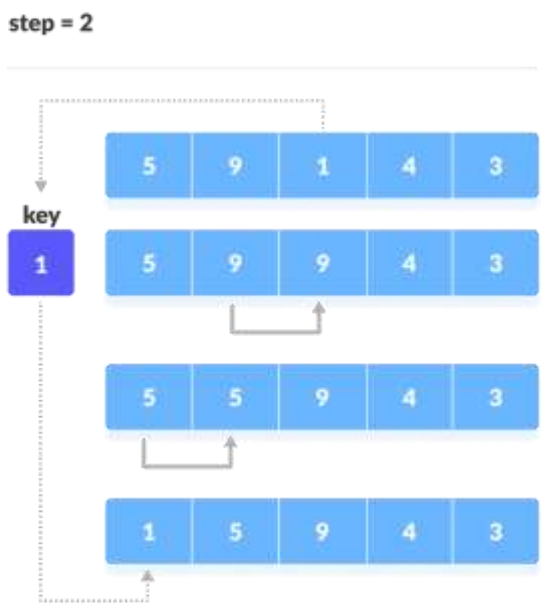
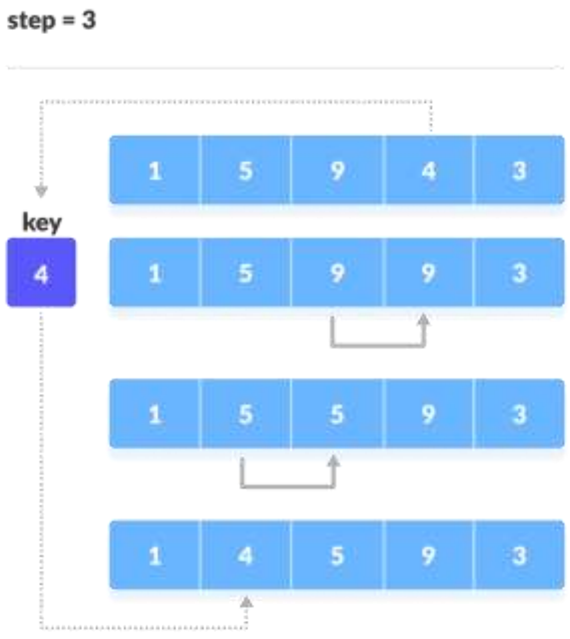


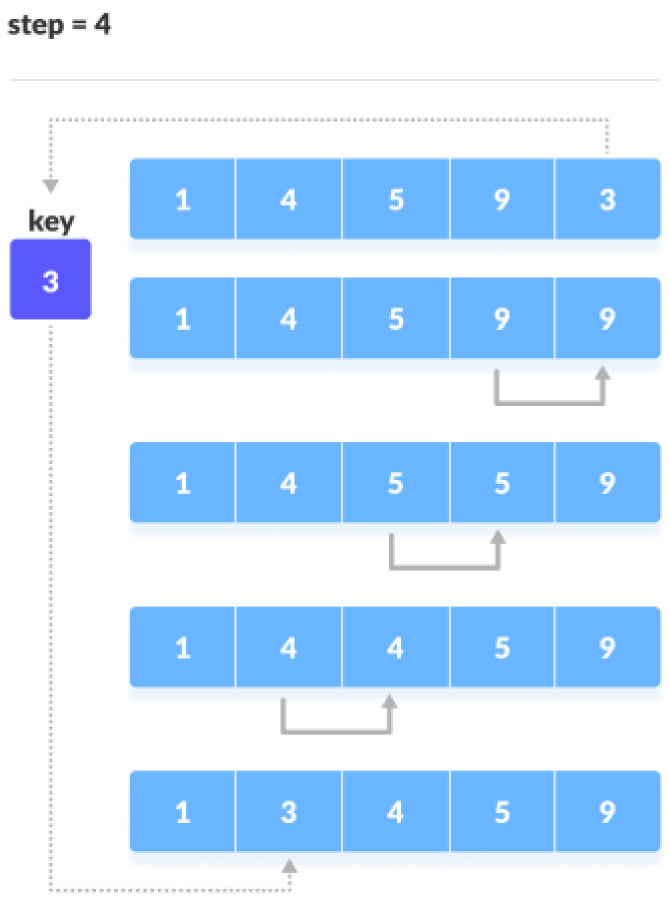
Initial array

1. The first element in the array is assumed to be sorted. Take the second element and store it separately in key.

   Compare key with the first element. If the first element is greater than key, then key is placed in front of the first

	element.  If the first element is greater than key, then key is placed in front of the first element. 2. Now, the first two elements are sorted. Take the third element and compare it with the elements on the left of it. Placed it just behind the element smaller than it. If there is no element smaller than it, then place it at the beginning of the array.

	 step = 2 key 1 5 9 1 4 3 5 9 9 4 3 5 5 9 4 3 1 5 9 4 3 Place 1 at the beginning 3. Similarly, place every unsorted element at its correct position.  step = 3 key 4 1 5 9 4 3 1 5 9 9 3 1 5 5 9 3 1 4 5 9 3 Place 4 behind 1

	 step = 4 key 3 1 4 5 9 3 1 4 5 9 9 1 4 5 5 9 1 4 4 5 9 1 3 4 5 9 Place 3 behind 1 and the array is sorted **Insertion Sort** *Insertion sort* is a simple sorting algorithm that is relatively efficient for small lists and mostly sorted lists, and is often used as part of more sophisticated algorithms. It works by taking elements from the list one by one and inserting them in their correct position into a new sorted list similar to how we put money in our wallet. In arrays, the new list and the remaining elements can share the array's space, but insertion is expensive, requiring shifting all following elements over by

one. Shellsort (see below) is a variant of insertion sort that is more efficient for larger lists.

Time Complexity
The worst case time complexity of Insertion sort is O(N^2)
The average case time complexity of Insertion sort is O(N^2)
The time complexity of the best case is O(N).
The space complexity is O(1)

Working Principle
- Compare the element with its adjacent element.
- If at every comparison, we could find a position in sorted array where the element can be inserted, then create space by shifting the elements to right and insert the element at the appropriate position.
- Repeat the above steps until you place the last element of unsorted array to its correct position.

Best Case Analysis
In Best Case i.e., when the array is already sorted, $t_j = 1$
Therefore,$T(n) = C_1 * n + (C_2 + C_3) * (n - 1) + C_4 * (n - 1) + (C_5 + C_6) * (n - 2) + C_8 * (n - 1)$
which when further simplified has dominating factor of n and gives T(n) = C * (n) or O(n)

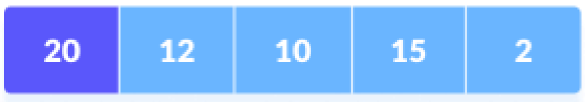
Worst Case Analysis
In Worst Case i.e., when the array is reversly sorted (in descending order), $t_j = j$
Therefore,$T(n) = C_1 * n + (C_2 + C_3) * (n - 1) + C_4 * (n - 1) (n) / 2 + (C_5 + C_6) * ((n - 1) (n) / 2 - 1) + C_8 * (n - 1)$
which when further simplified has dominating factor of n^2 and gives $T(n) = C * (n^2)$ or $O(n^2)$

Average Case Analysis
Let's assume that $t_j = (j-1)/2$ to calculate the average case
Therefore,$T(n) = C_1 * n + (C_2 + C_3) * (n - 1) + C_4/2 * (n - 1) (n) / 2 + (C_5 + C_6)/2 * ((n - 1) (n) / 2 - 1) + C_8 * ($

n - 1)
which when further simplified has dominating factor of n^2 and gives $T(n) = C * (n^2)$ or $O(n^2)$

Working of Selection Sort:

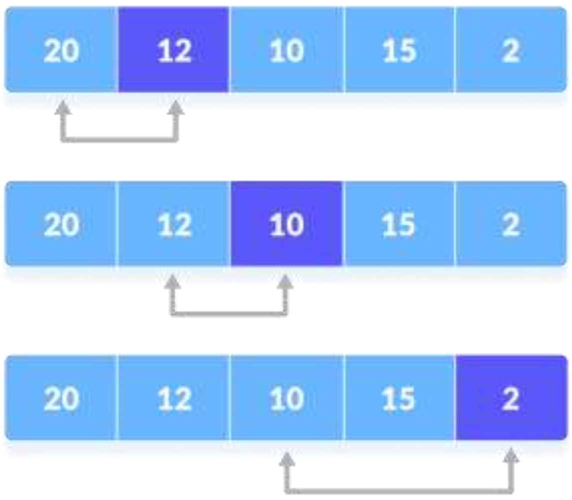
1. Set the first element as minimum.



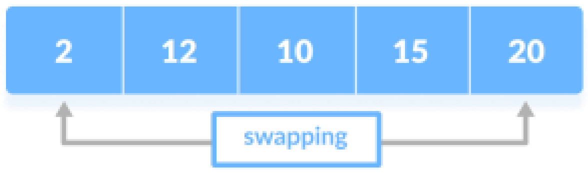
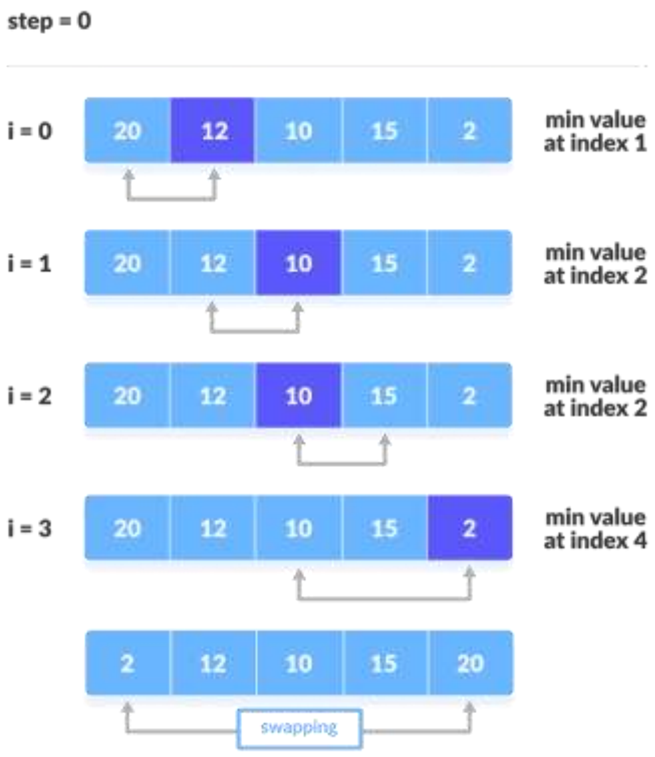
Select first element as minimum

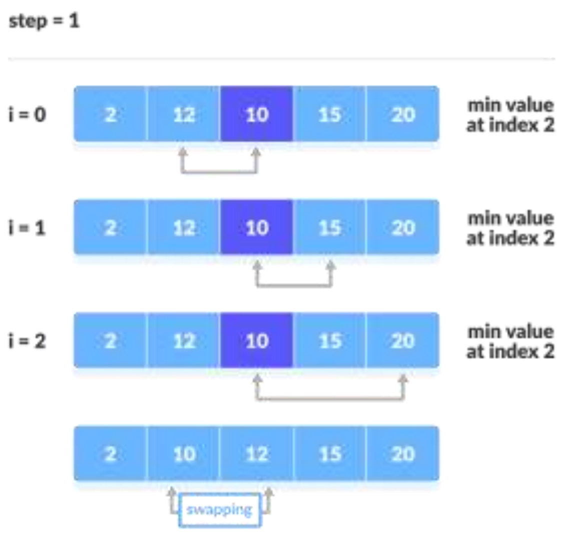
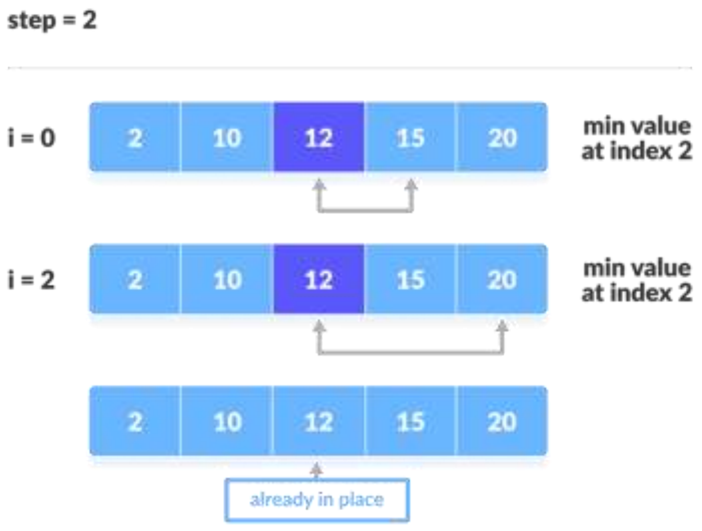
2. Compare minimum with the second element. If the second element is smaller than minimum, assign the second element as minimum.

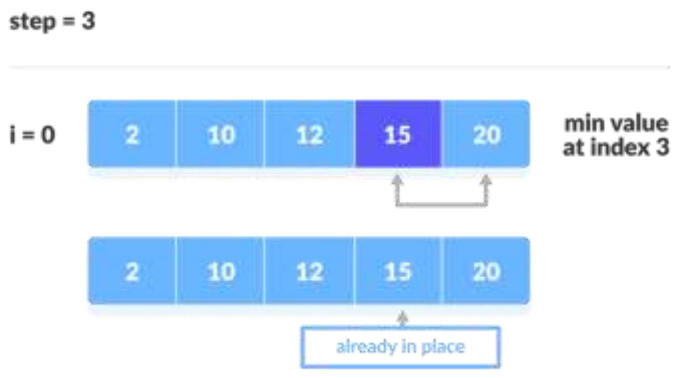
   Compare minimum with the third element. Again, if the third element is smaller, then assign minimum to the third element otherwise do nothing. The process goes on until the last element.



Compare minimum with the remaining elements

	3. After each iteration, minimum is placed in the front of the unsorted list.  Swap the first with minimum 4. For each iteration, indexing starts from the first unsorted element. Step 1 to 3 are repeated until all the elements are placed at their correct positions.  step = 0 i = 0 20 12 10 15 2 min value at index 1 i = 1 20 12 10 15 2 min value at index 2 i = 2 20 12 10 15 2 min value at index 2 i = 3 20 12 10 15 2 min value at index 4 2 12 10 15 20 swapping The first iteration

	 The second iteration  The third iteration

	 step = 3 i = 0 2 10 12 15 20 min value at index 3 2 10 12 15 20 already in place The fourth iteration **ALGO:** 1. Find the smallest card. Swap it with the first card. 2. Find the second-smallest card. Swap it with the second card. 3. Find the third-smallest card. Swap it with the third card. 4. Repeat finding the next-smallest card, and swapping it into the correct position until the array is sorted.

CODE:	import java.io.BufferedReader;
	import java.io.IOException;
	import java.util.*;
	
	public class Sort {
	    public static void insertionSort(int array[]) {
	        int arrayLen = array.length;
	        for (int i = 1; i < arrayLen; ++i) {
	            System.out.println("\n****************
	****************");
	            int currentPos = array[i];
	            System.out.print("\nIteration on Element " +
	currentPos);
	            int j = i - 1;
	            while (j >= 0 && array[j] > currentPos) {

```
            System.out.print("\nSwapping " + array[j] + " with
" + array[j + 1]);
            array[j + 1] = array[j];
            j--;
         }
         array[j + 1] = currentPos;
         System.out.print("\nArray : " + printArray(array));
      }
   }

   public static void selectionSort(int array[]) {
      int arrayLen = array.length;

      for (int i = 0; i < arrayLen - 1; i++) {
         System.out.print("\n*************
*********************");
         int minElement = i;
         for (int j = i + 1; j < arrayLen; j++) {
            if (array[j] < array[minElement]) {
               minElement = j;
            }
         }
         System.out.print("\nThe minimum Element is " +
array[minElement]);
         System.out.print("\nSwapping " + array[i] + " with "
+ array[minElement]);
         int temp = array[minElement];
         array[minElement] = array[i];
         array[i] = temp;
         System.out.print("\nArray : " + printArray(array));
      }
   }

   public static String printArray(int array[]) {
      String result = "[ ";
      for (int i : array) {
         result += i + " ";
      }
```

```
        result += "]";
        return result;
    }

    public static void main(String[] args) throws IOException {

        ArrayList<Integer> arrayList = new
ArrayList<Integer>();

        int sortingArray[];
        BufferedReader br = new BufferedReader(new
java.io.InputStreamReader(System.in));
        System.out.print("\n1.Manual Input\n2.Random
Input\n3.Roll no Input : ");
        int choice = Integer.parseInt(br.readLine());

        if (choice == 1) {
            System.out.print("\nEnter the elements of the
array(with space)\n-> ");
            String numbers[] = br.readLine().split(" ");
            for (String number : numbers)
                arrayList.add(Integer.parseInt(number));
        } else if (choice == 2) {
            System.out.print("\nEnter the size of the array : ");
            int size = Integer.parseInt(br.readLine());
            for (int i = 0; i < size; i++) {
                arrayList.add((int) (Math.random() * 100));
            }
        } else {
            System.out.print("\nEnter the roll no : ");
            int rollNo = Integer.parseInt(br.readLine());
            for (int i = 0; i < 10; i++)
                arrayList.add(rollNo + (rollNo + 1) * i);
            Collections.shuffle(arrayList);
        }

        sortingArray = new int[arrayList.size()];
        for (int i = 0; i < arrayList.size(); i++) {
            sortingArray[i] = arrayList.get(i);
        }

        System.out.print("\nArray : " +
printArray(sortingArray));
        System.out.print("\n\n===================
=================");
        System.out.print("\nWhich sorting algorithm do you
want to use?\n1. Insertion Sort\n2. Selection Sort\n-> ");
        int decision = Integer.parseInt(br.readLine());
        System.out.print("\n===================
=================");
        System.out.print("\nBefore Sort: " +
printArray(sortingArray));
        switch (decision) {
            case 1:
                System.out.print("\nInsertion Sort");
                insertionSort(sortingArray);
                break;
            case 2:
                System.out.print("\nSelection Sort");
                selectionSort(sortingArray);
                break;
            default:
                System.out.print("\nInvalid Choice");
        }

        System.out.print("\n-------------------
---------------------------------");
        System.out.print("\nFinal After Sort: " +
printArray(sortingArray));

        System.out.print("\n\n1.Exit\n2.Continue\n-> ");
        int exit = Integer.parseInt(br.readLine());
        if (exit == 1)
            System.exit(0);
        else
            main(null);}}
```

Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

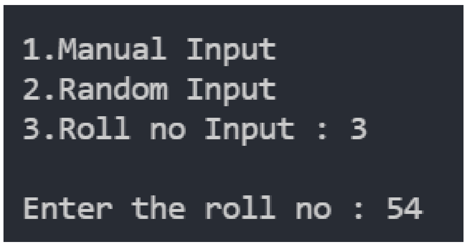
Computer Engineering Department &
Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** DAA

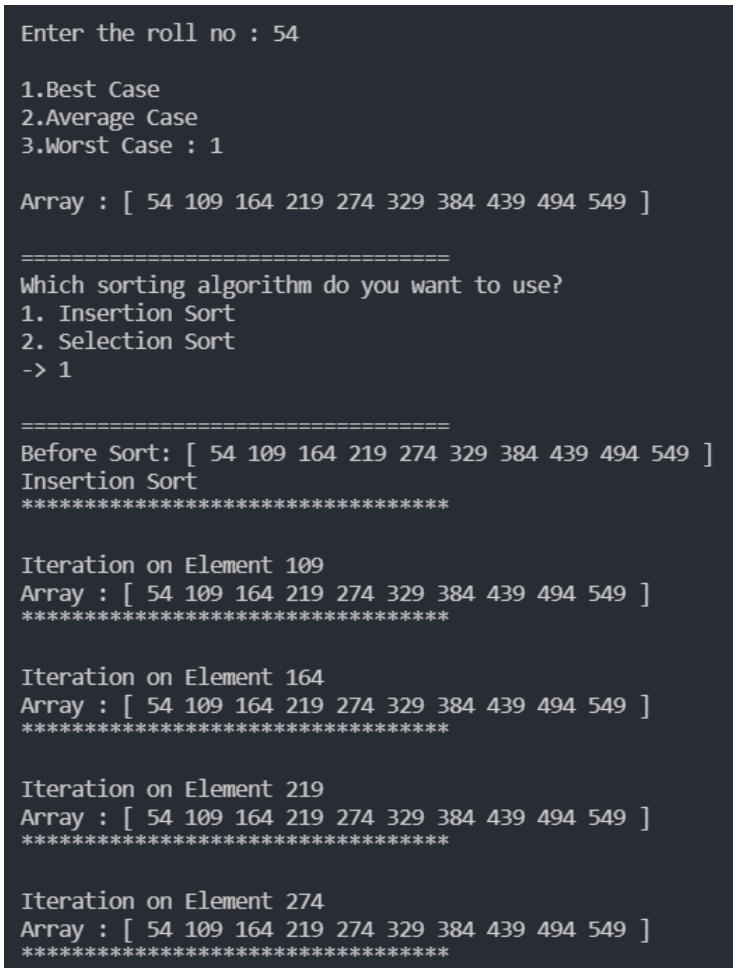
OUTPUT:

Insertion Sort:

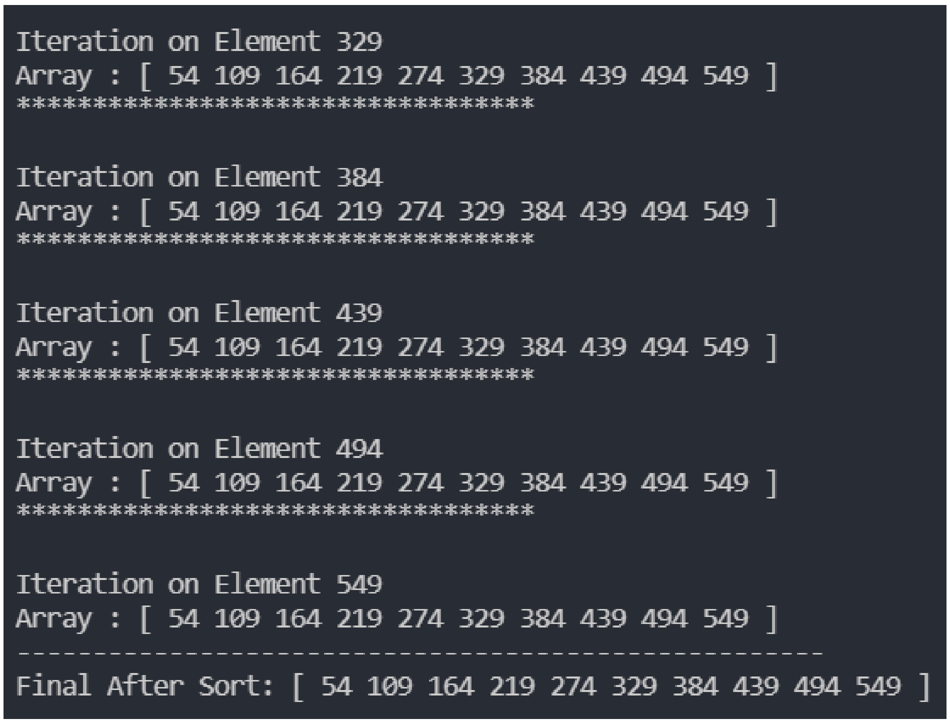


```
1.Manual Input
2.Random Input
3.Roll no Input : 3

Enter the roll no : 54
```

Best Case:

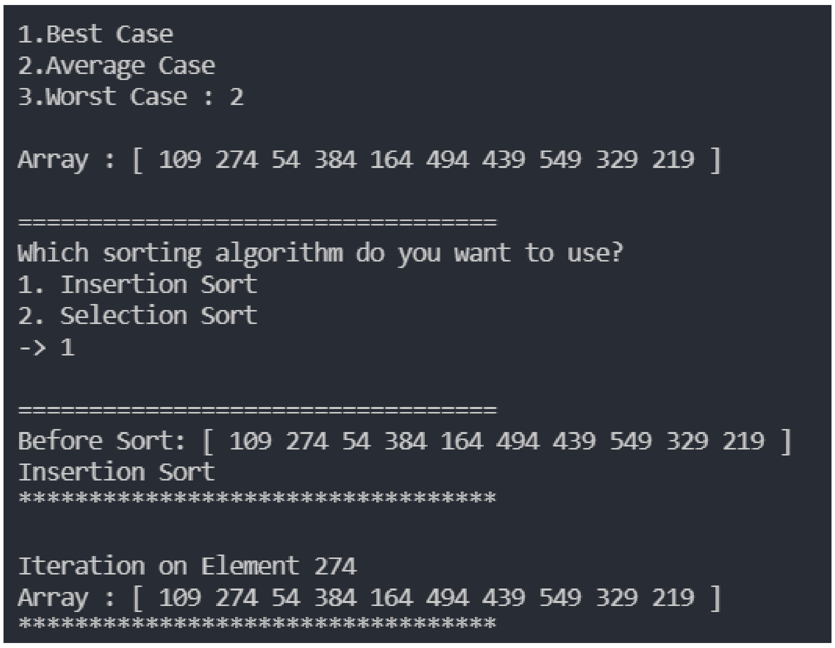


```
Enter the roll no : 54

1.Best Case
2.Average Case
3.Worst Case : 1

Array : [ 54 109 164 219 274 329 384 439 494 549 ]

==================================
Which sorting algorithm do you want to use?
1. Insertion Sort
2. Selection Sort
-> 1

==================================
Before Sort: [ 54 109 164 219 274 329 384 439 494 549 ]
Insertion Sort
**********************************

Iteration on Element 109
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
**********************************

Iteration on Element 164
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
**********************************

Iteration on Element 219
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
**********************************

Iteration on Element 274
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
**********************************
```



```
Iteration on Element 329
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
************************************

Iteration on Element 384
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
************************************

Iteration on Element 439
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
************************************

Iteration on Element 494
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
************************************

Iteration on Element 549
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
--------------------------------------------------------
Final After Sort: [ 54 109 164 219 274 329 384 439 494 549 ]
```

Average Case:



```
1.Best Case
2.Average Case
3.Worst Case : 2

Array : [ 109 274 54 384 164 494 439 549 329 219 ]

==================================
Which sorting algorithm do you want to use?
1. Insertion Sort
2. Selection Sort
-> 1

==================================
Before Sort: [ 109 274 54 384 164 494 439 549 329 219 ]
Insertion Sort
**********************************

Iteration on Element 274
Array : [ 109 274 54 384 164 494 439 549 329 219 ]
**********************************
```

Bharatiya Vidya Bhavan's

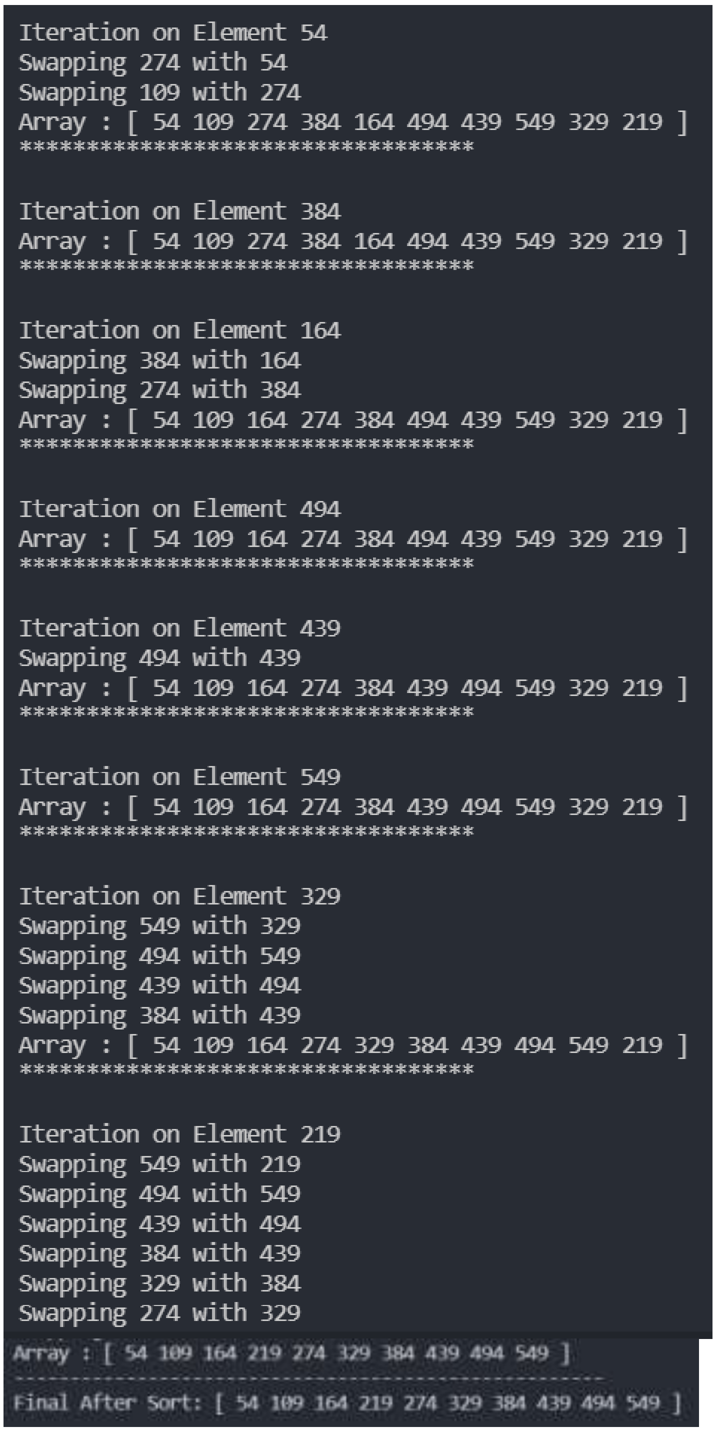
Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department &
Information Technology Engineering Department

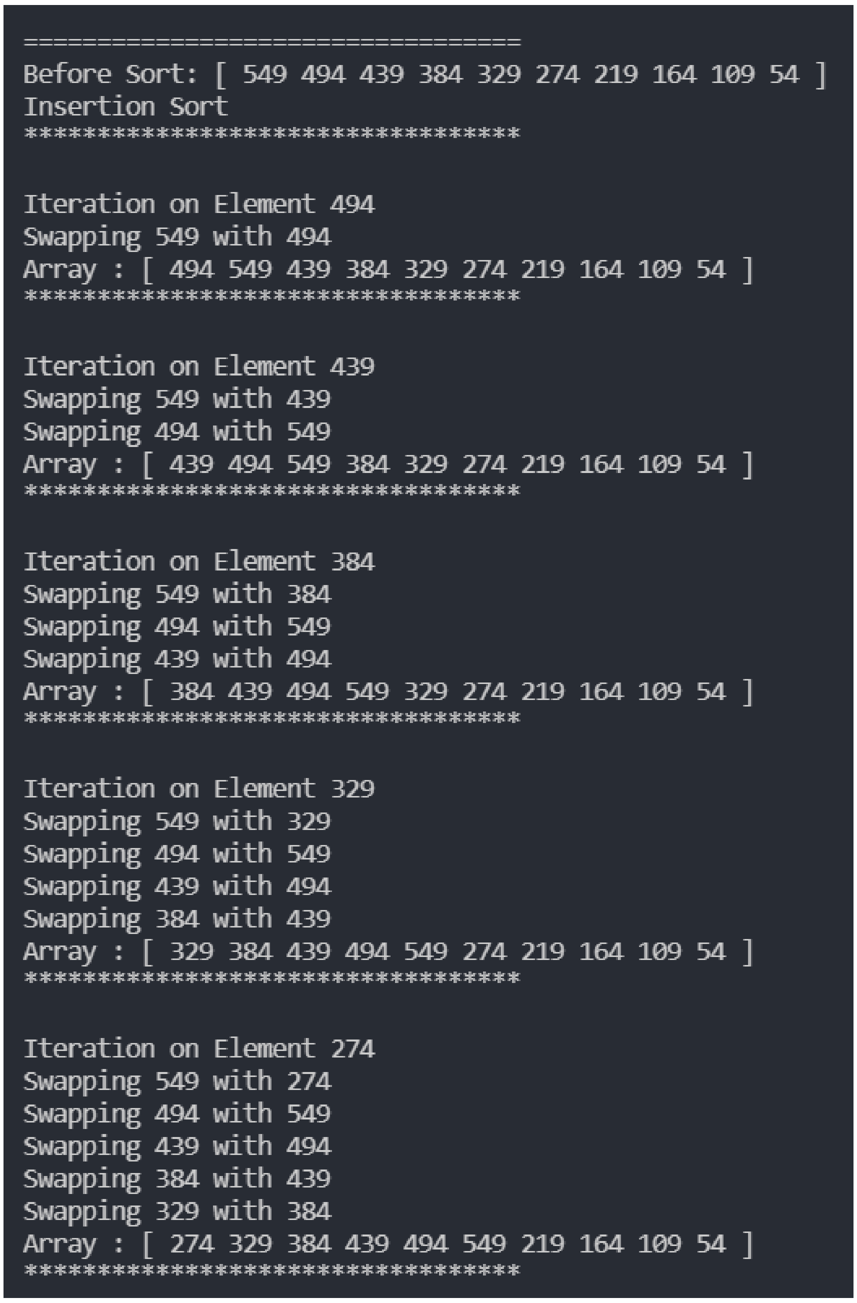
Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** DAA

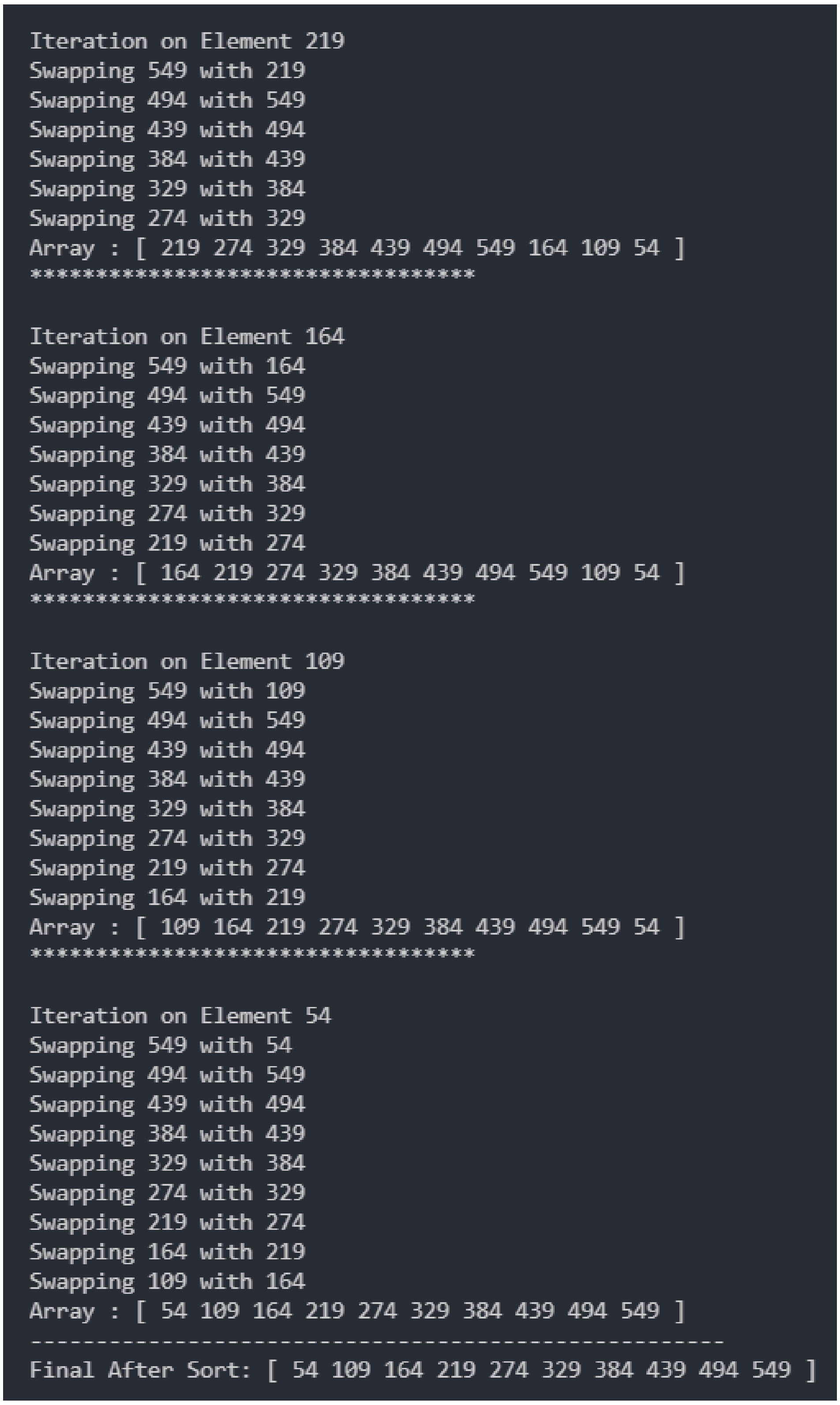
	

```
Iteration on Element 54
Swapping 274 with 54
Swapping 109 with 274
Array : [ 54 109 274 384 164 494 439 549 329 219 ]
**********************************

Iteration on Element 384
Array : [ 54 109 274 384 164 494 439 549 329 219 ]
**********************************

Iteration on Element 164
Swapping 384 with 164
Swapping 274 with 384
Array : [ 54 109 164 274 384 494 439 549 329 219 ]
**********************************

Iteration on Element 494
Array : [ 54 109 164 274 384 494 439 549 329 219 ]
**********************************

Iteration on Element 439
Swapping 494 with 439
Array : [ 54 109 164 274 384 439 494 549 329 219 ]
**********************************

Iteration on Element 549
Array : [ 54 109 164 274 384 439 494 549 329 219 ]
**********************************

Iteration on Element 329
Swapping 549 with 329
Swapping 494 with 549
Swapping 439 with 494
Swapping 384 with 439
Array : [ 54 109 164 274 329 384 439 494 549 219 ]
**********************************

Iteration on Element 219
Swapping 549 with 219
Swapping 494 with 549
Swapping 439 with 494
Swapping 384 with 439
Swapping 329 with 384
Swapping 274 with 329
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
------------------------------------------------
Final After Sort: [ 54 109 164 219 274 329 384 439 494 549 ]
```

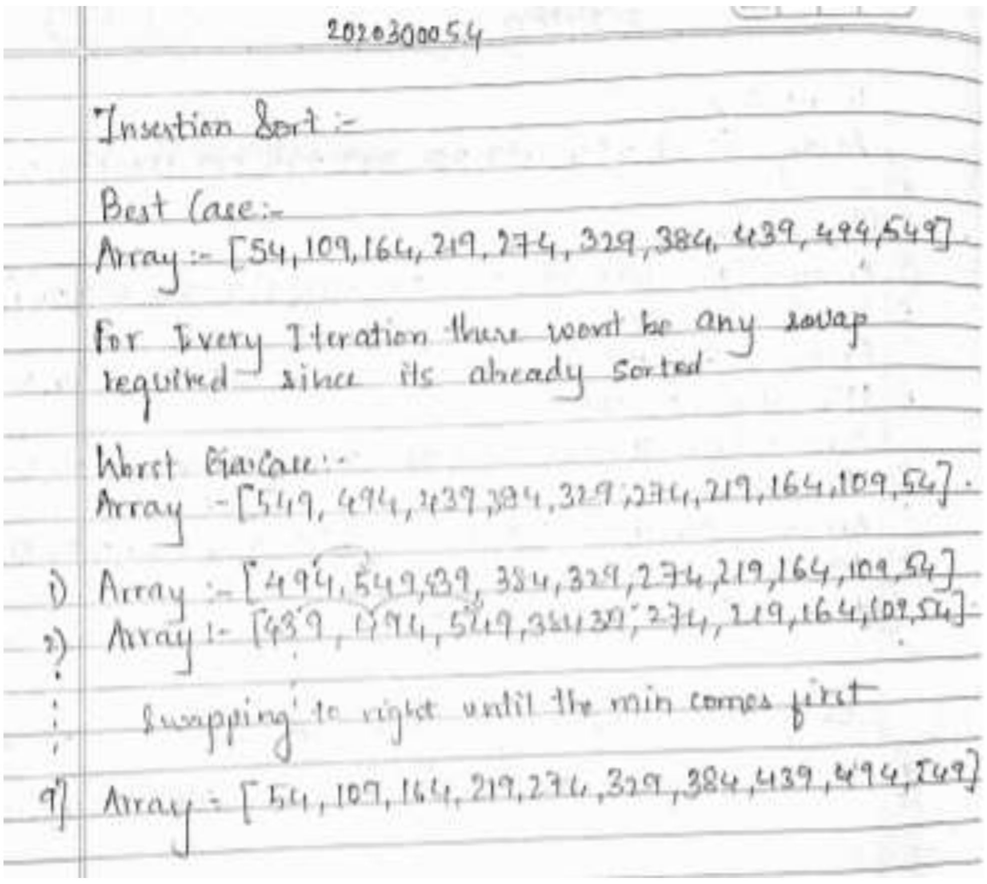
Worth Case:



```
==================================
Before Sort: [ 549 494 439 384 329 274 219 164 109 54 ]
Insertion Sort
**********************************

Iteration on Element 494
Swapping 549 with 494
Array : [ 494 549 439 384 329 274 219 164 109 54 ]
**********************************

Iteration on Element 439
Swapping 549 with 439
Swapping 494 with 549
Array : [ 439 494 549 384 329 274 219 164 109 54 ]
**********************************

Iteration on Element 384
Swapping 549 with 384
Swapping 494 with 549
Swapping 439 with 494
Array : [ 384 439 494 549 329 274 219 164 109 54 ]
**********************************

Iteration on Element 329
Swapping 549 with 329
Swapping 494 with 549
Swapping 439 with 494
Swapping 384 with 439
Array : [ 329 384 439 494 549 274 219 164 109 54 ]
**********************************

Iteration on Element 274
Swapping 549 with 274
Swapping 494 with 549
Swapping 439 with 494
Swapping 384 with 439
Swapping 329 with 384
Array : [ 274 329 384 439 494 549 219 164 109 54 ]
**********************************
```



```
Iteration on Element 219
Swapping 549 with 219
Swapping 494 with 549
Swapping 439 with 494
Swapping 384 with 439
Swapping 329 with 384
Swapping 274 with 329
Array : [ 219 274 329 384 439 494 549 164 109 54 ]
**********************************

Iteration on Element 164
Swapping 549 with 164
Swapping 494 with 549
Swapping 439 with 494
Swapping 384 with 439
Swapping 329 with 384
Swapping 274 with 329
Swapping 219 with 274
Array : [ 164 219 274 329 384 439 494 549 109 54 ]
**********************************

Iteration on Element 109
Swapping 549 with 109
Swapping 494 with 549
Swapping 439 with 494
Swapping 384 with 439
Swapping 329 with 384
Swapping 274 with 329
Swapping 219 with 274
Swapping 164 with 219
Array : [ 109 164 219 274 329 384 439 494 549 54 ]
**********************************

Iteration on Element 54
Swapping 549 with 54
Swapping 494 with 549
Swapping 439 with 494
Swapping 384 with 439
Swapping 329 with 384
Swapping 274 with 329
Swapping 219 with 274
Swapping 164 with 219
Swapping 109 with 164
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
-----------------------------------------------------
Final After Sort: [ 54 109 164 219 274 329 384 439 494 549 ]
```



2020300054

Insertion Sort :-

Best Case:-
Array :- [54, 109, 164, 219, 274, 329, 384, 439, 494, 549]

For Every Iteration there won't be any swap required since its already sorted

Worst ~~Ca~~Case:-
Array -[549, 494, 439, 384, 329, 274, 219, 164, 109, 54].

1) Array :- [494, 549, 439, 384, 329, 274, 219, 164, 109, 54]
2) Array :- [439, 494, 549, 384, 329, 274, 219, 164, 109, 54]
⋮
Swapping to right until the min comes first
⋮
9) Array := [54, 109, 164, 219, 274, 329, 384, 439, 494, 549]

For the Average Case:
Every Iteration, the array element will float up using swaps

Complexity of Insertion Sort: The best case complexity of insertion sort is O(n) times, i.e. when the array is previously sorted. In the same way, when the array is sorted in reverse order, the first element of the unsorted array is to be compared with each element in the sorted set. So, in the worst case,running time of Insertion sort is quadratic, i.e., O(n 2).

Bharatiya Vidya Bhavan's

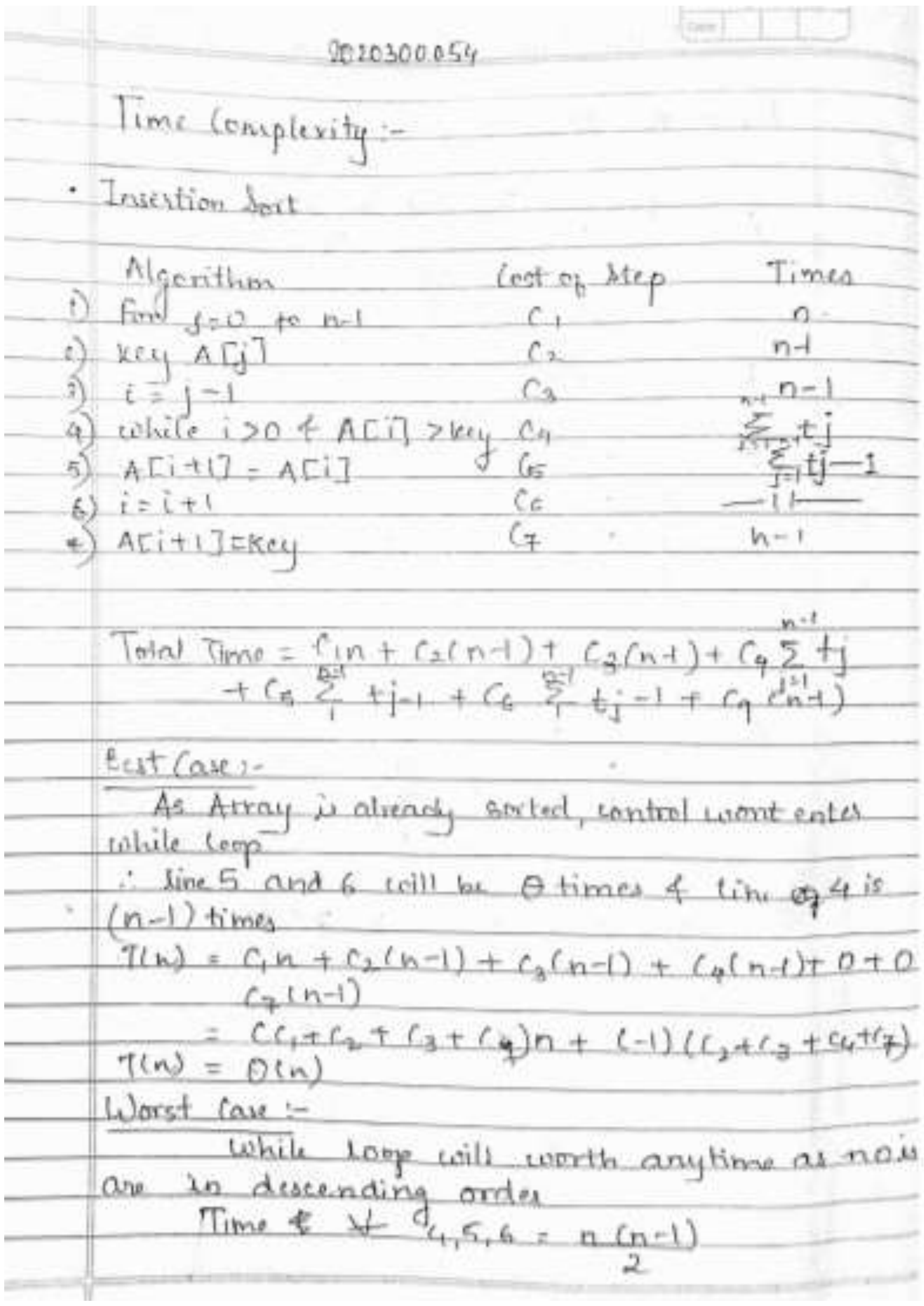
Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department &
Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** DAA



2020300054

Time Complexity :-

• Insertion Sort

	Algorithm	Cost of Step	Times
1)	For $j=0$ to $n-1$	C_1	n
2)	key $A[j]$	C_2	$n-1$
3)	$i=j-1$	C_3	$n-1$
4)	while $i>0$ & $A[i]>key$	C_4	$\sum_{j=1}^{n-1} t_j$
5)	$A[i+1]=A[i]$	C_5	$\sum_{j=1} t_j - 1$
6)	$i=i+1$	C_6	— 11 —
7)	$A[i+1]=Key$	C_7	$n-1$

Total Time $= C_1 n + C_2(n-1) + C_3(n-1) + C_4 \sum_{j=1}^{n-1} t_j$
$+ C_5 \sum_{1}^{n-1} t_{j-1} + C_6 \sum_{1}^{n-1} t_j - 1 + C_7(n-1)$

Best Case :-

As Array is already sorted, control wont enter while loop

∴ line 5 and 6 will be 0 times & line of 4 is (n-1) times

$T(n) = C_1 n + C_2(n-1) + C_3(n-1) + C_4(n-1) + 0 + 0$
$C_7(n-1)$

$= (C_1 + C_2 + C_3 + C_4)n + (-1)(C_2 + C_3 + C_4 + C_7)$

$T(n) = \Theta(n)$

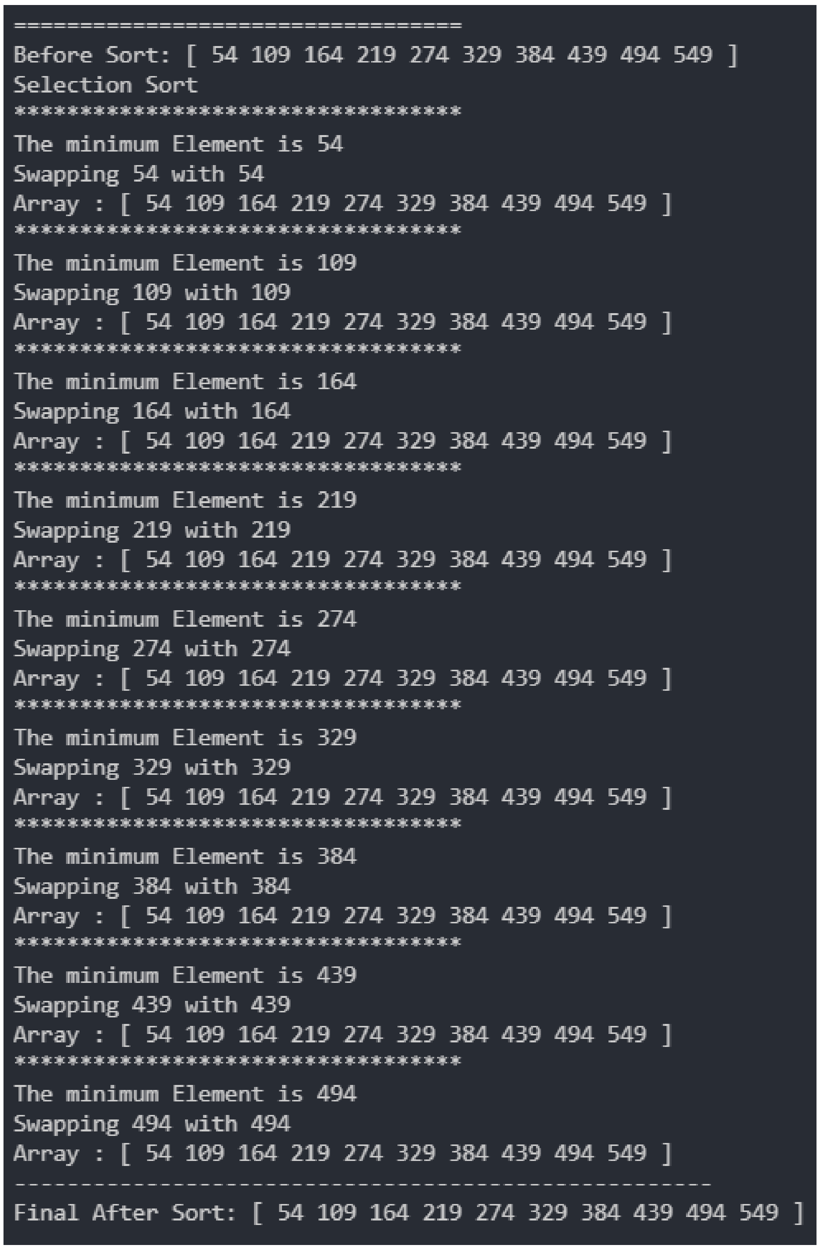
Worst Case :-

While loop will worth anytime as no is are in descending order

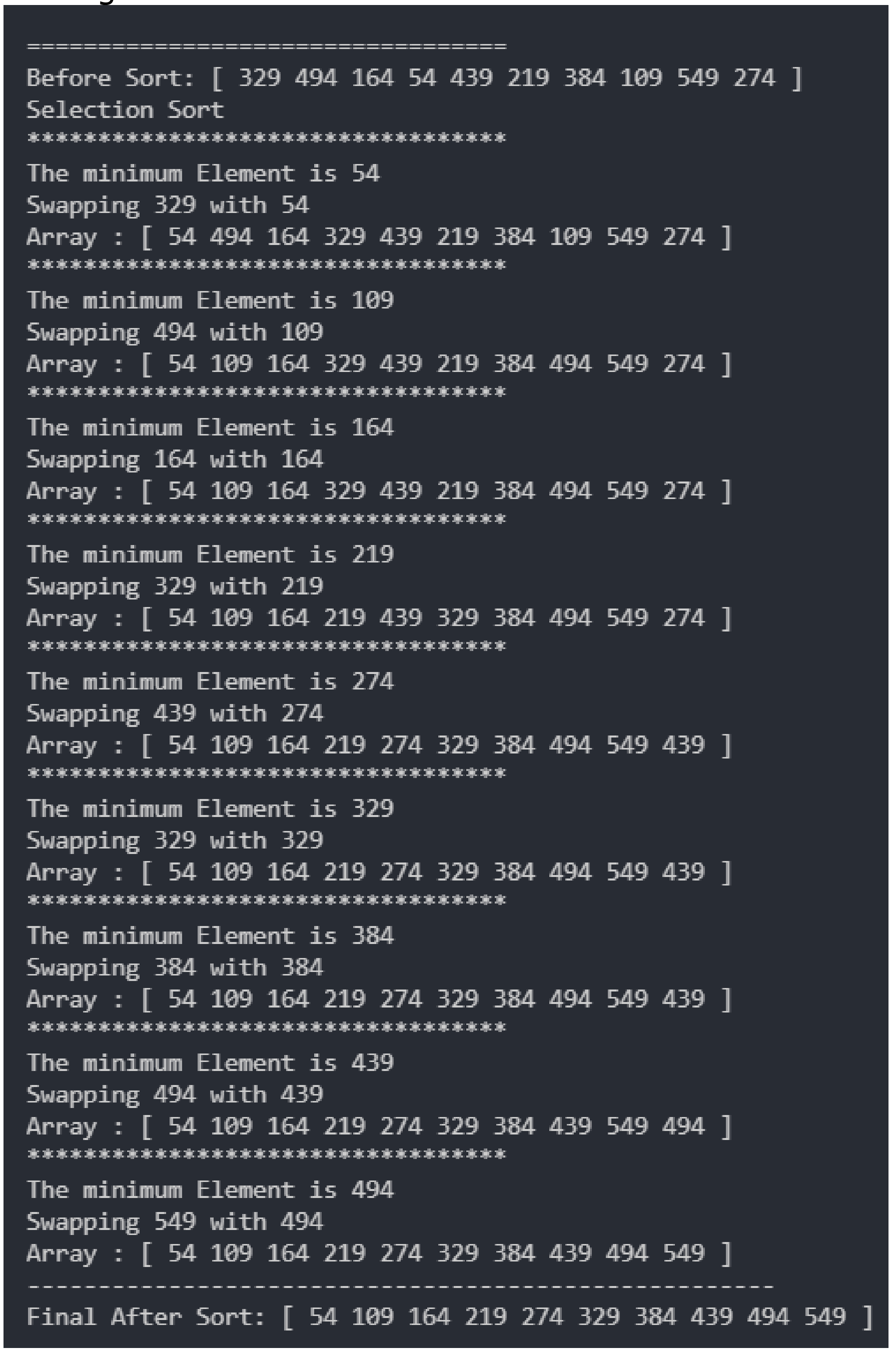
Time ∀ $C_{4,5,6} = \frac{n(n-1)}{2}$

Selection Sort:

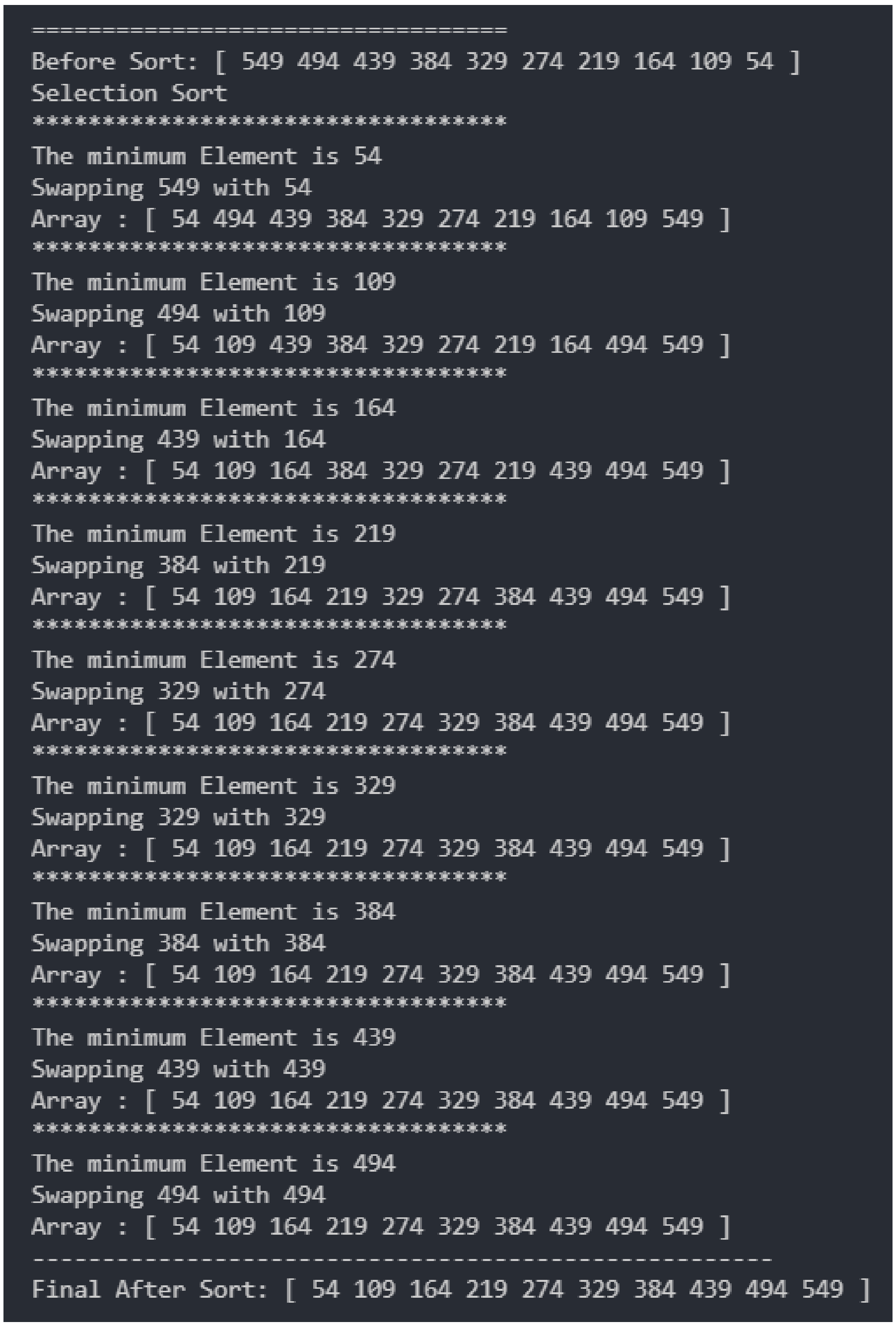
Best Case:



```
==================================
Before Sort: [ 54 109 164 219 274 329 384 439 494 549 ]
Selection Sort
**********************************
The minimum Element is 54
Swapping 54 with 54
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
**********************************
The minimum Element is 109
Swapping 109 with 109
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
**********************************
The minimum Element is 164
Swapping 164 with 164
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
**********************************
The minimum Element is 219
Swapping 219 with 219
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
**********************************
The minimum Element is 274
Swapping 274 with 274
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
**********************************
The minimum Element is 329
Swapping 329 with 329
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
**********************************
The minimum Element is 384
Swapping 384 with 384
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
**********************************
The minimum Element is 439
Swapping 439 with 439
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
**********************************
The minimum Element is 494
Swapping 494 with 494
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
-----------------------------------------------------
Final After Sort: [ 54 109 164 219 274 329 384 439 494 549 ]
```

	Average Case: 

```
==================================
Before Sort: [ 329 494 164 54 439 219 384 109 549 274 ]
Selection Sort
**********************************
The minimum Element is 54
Swapping 329 with 54
Array : [ 54 494 164 329 439 219 384 109 549 274 ]
**********************************
The minimum Element is 109
Swapping 494 with 109
Array : [ 54 109 164 329 439 219 384 494 549 274 ]
**********************************
The minimum Element is 164
Swapping 164 with 164
Array : [ 54 109 164 329 439 219 384 494 549 274 ]
**********************************
The minimum Element is 219
Swapping 329 with 219
Array : [ 54 109 164 219 439 329 384 494 549 274 ]
**********************************
The minimum Element is 274
Swapping 439 with 274
Array : [ 54 109 164 219 274 329 384 494 549 439 ]
**********************************
The minimum Element is 329
Swapping 329 with 329
Array : [ 54 109 164 219 274 329 384 494 549 439 ]
**********************************
The minimum Element is 384
Swapping 384 with 384
Array : [ 54 109 164 219 274 329 384 494 549 439 ]
**********************************
The minimum Element is 439
Swapping 494 with 439
Array : [ 54 109 164 219 274 329 384 439 549 494 ]
**********************************
The minimum Element is 494
Swapping 549 with 494
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
-----------------------------------------------------
Final After Sort: [ 54 109 164 219 274 329 384 439 494 549 ]
```

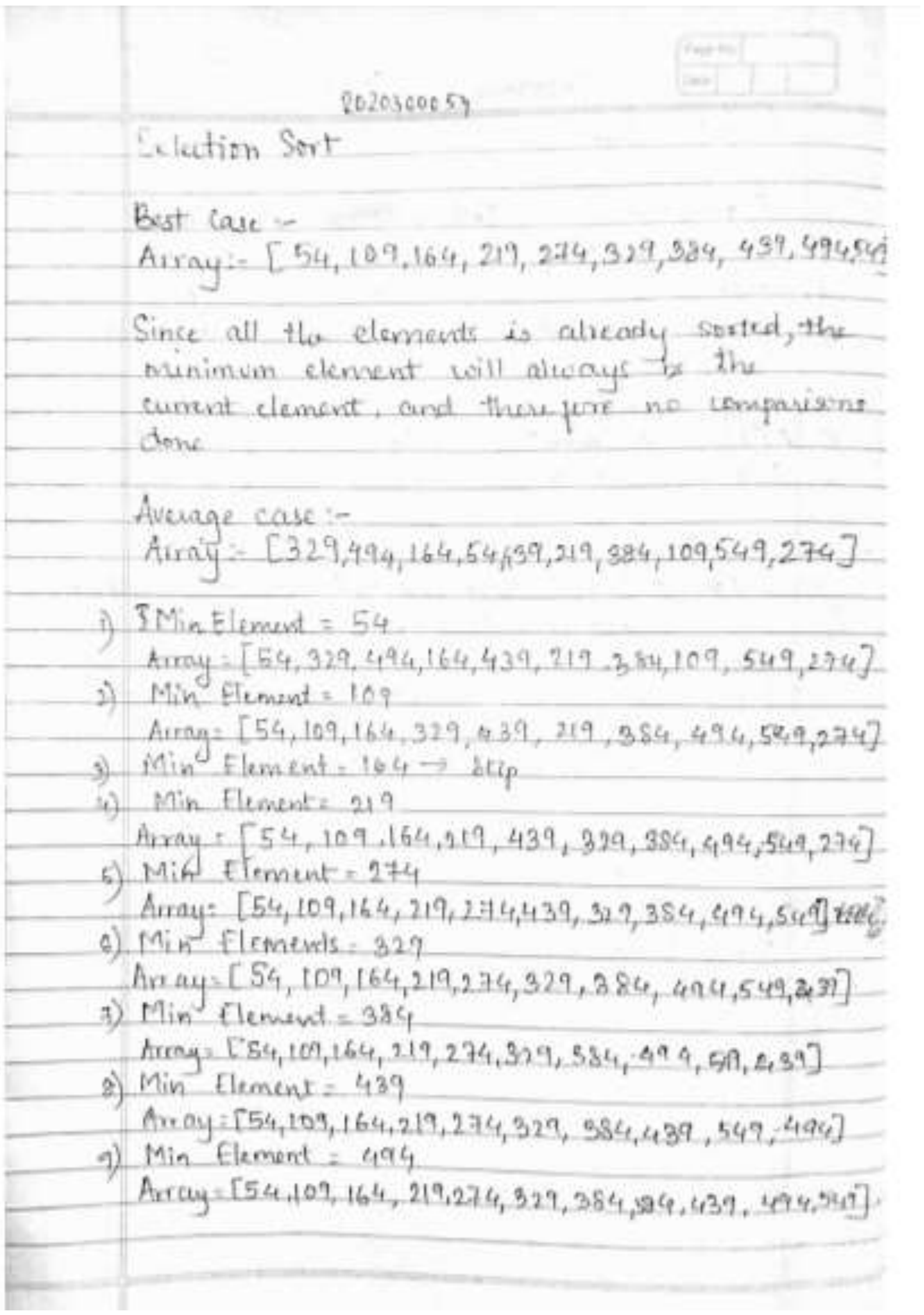
	Worst Case: 

```
==================================
Before Sort: [ 549 494 439 384 329 274 219 164 109 54 ]
Selection Sort
**********************************
The minimum Element is 54
Swapping 549 with 54
Array : [ 54 494 439 384 329 274 219 164 109 549 ]
**********************************
The minimum Element is 109
Swapping 494 with 109
Array : [ 54 109 439 384 329 274 219 164 494 549 ]
**********************************
The minimum Element is 164
Swapping 439 with 164
Array : [ 54 109 164 384 329 274 219 439 494 549 ]
**********************************
The minimum Element is 219
Swapping 384 with 219
Array : [ 54 109 164 219 329 274 384 439 494 549 ]
**********************************
The minimum Element is 274
Swapping 329 with 274
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
**********************************
The minimum Element is 329
Swapping 329 with 329
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
**********************************
The minimum Element is 384
Swapping 384 with 384
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
**********************************
The minimum Element is 439
Swapping 439 with 439
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
**********************************
The minimum Element is 494
Swapping 494 with 494
Array : [ 54 109 164 219 274 329 384 439 494 549 ]
----------------------------------------------------
Final After Sort: [ 54 109 164 219 274 329 384 439 494 549 ]
```

<u>Computer Engineering Department &</u>
<u>Information Technology Engineering Department</u>

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** DAA

2020300057

Selection Sort

Best case :-
Array :- [54, 109, 164, 219, 274, 329, 384, 439, 494, 549

Since all the elements is already sorted, the minimum element will always be the current element, and therefore no comparisons done

Average case :-
Array :- [329, 494, 164, 54, 439, 219, 384, 109, 549, 274]

1) Min Element = 54
Array : [54, 329, 494, 164, 439, 219, 384, 109, 549, 274]
2) Min Element = 109
Array = [54, 109, 164, 329, 439, 219, 384, 494, 549, 274]
3) Min Element = 164 → skip
4) Min Element = 219
Array : [54, 109, 164, 219, 439, 329, 384, 494, 549, 274]
5) Min Element = 274
Array: [54, 109, 164, 219, 274, 439, 329, 384, 494, 549]
6) Min Elements = 329
Array = [54, 109, 164, 219, 274, 329, 384, 494, 549, 439]
7) Min Element = 384
Array = [54, 109, 164, 219, 274, 329, 384, 494, 549, 439]
8) Min Element = 439
Array = [54, 109, 164, 219, 274, 329, 384, 439, 549, 494]
9) Min Element = 494
Array = [54, 109, 164, 219, 274, 329, 384, 384, 439, 494, 549].

Bharatiya Vidya Bhavan's

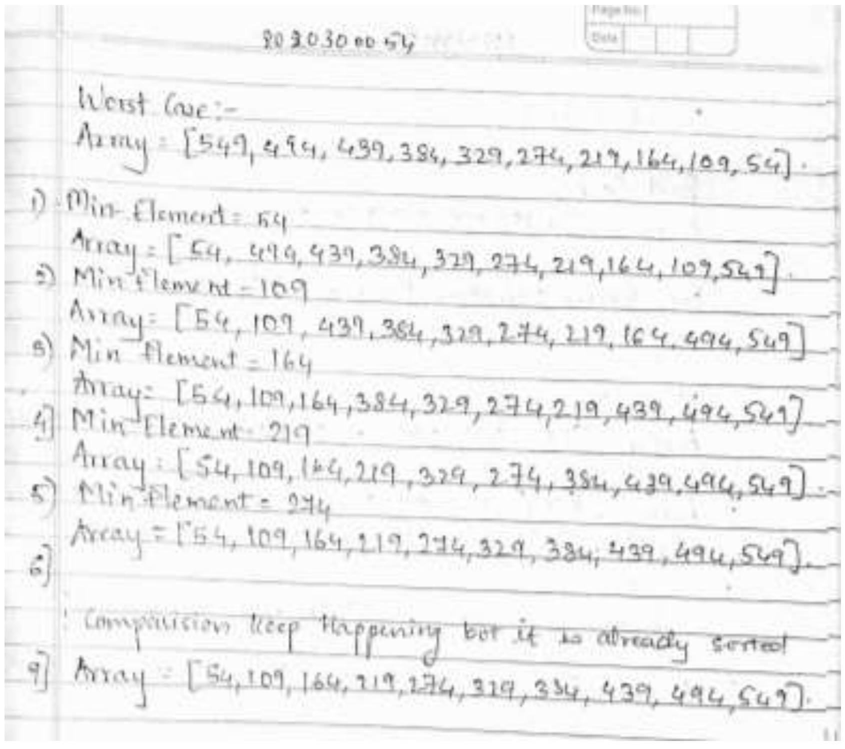
Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department &
Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** DAA

20 3030 00 54

Worst Case:-
Array = [549, 494, 439, 384, 329, 274, 219, 164, 109, 54].

1) Min Element = 54
Array = [54, 494, 439, 384, 329, 274, 219, 164, 109, 549].
2) Min Element = 109
Array = [54, 109, 439, 384, 329, 274, 219, 164, 494, 549]
3) Min Element = 164
Array = [54, 109, 164, 384, 329, 274, 219, 439, 494, 549]
4] Min Element = 219
Array = [54, 109, 164, 219, 329, 274, 384, 439, 494, 549].
5) Min Element = 274
Array = [54, 109, 164, 219, 274, 329, 384, 439, 494, 549].
6]

: Comparision keep happening but it is already sorted
9] Array = [54, 109, 164, 219, 274, 329, 384, 439, 494, 549].

Bharatiya Vidya Bhavan's

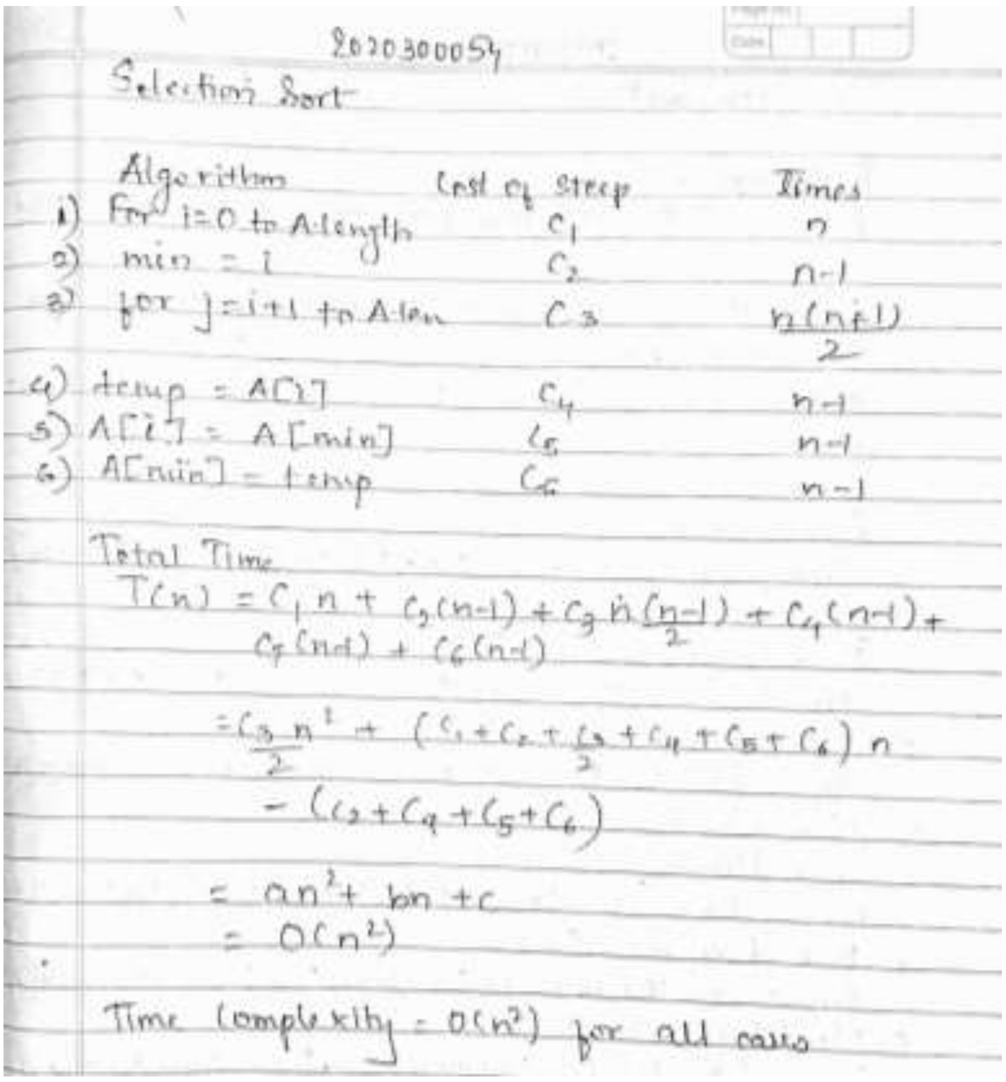
Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department &
Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** DAA



2020300054

Selection Sort

Algorithm	Cost of step	Times
1) For i=0 to A.length	c_1	n
2) min = i	c_2	$n-1$
3) for j=i+1 to A.len	c_3	$\frac{n(n+1)}{2}$
4) temp = A[i]	c_4	$n-1$
5) A[i] = A[min]	c_5	$n-1$
6) A[min] = temp	c_6	$n-1$

Total Time

$$T(n) = c_1 n + c_2(n-1) + c_3 n\frac{(n-1)}{2} + c_4(n-1) + c_5(n-1) + c_6(n-1)$$

$$= \frac{c_3}{2} n^2 + \left(c_1 + c_2 + \frac{c_3}{2} + c_4 + c_5 + c_6\right) n - (c_2 + c_4 + c_5 + c_6)$$

$$= an^2 + bn + c$$

$$= O(n^2)$$

Time Complexity = $O(n^2)$ for all cases

Complexity of Selection Sort: As the working of selection,sort does not depend on the original order of the elements in the array,so there is not much difference between best case and worst case complexity of selection sort. Therefore, in both the cases, the complexity is O(n 2).

CONCLUSION: Concepts learnt during procedural programming of the problem

Bharatiya Vidya Bhavan's

Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department &
Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** DAA

- Learnt about the time complexity and proved and perform complexity cases on selection and insertion sort
- Learnt how to write Insertion and Selection sort using the algorithm
- Learnt how to different cases differ in computation power as the number of elements increase in array
- I learnt about the advantages and disadvantages about selection and insertion sort and how to improve the overall time complexity
- Learnt how to derive and compute the time complexity of selection sort and insertion sort