

**Statement Purpose:**

Purpose of this Lab is to familiarize the students with the use of N-Squared sorting algorithms namely Selection Sort, Insertion Sort and Bubble Sort. Another aim is to teach the students how we can improve sorting time using Merge Sort algorithm. The students are given small tasks related to aforementioned sorting techniques under the supervision of the lab instructor. This helps them understand the concepts well which they learn in their lectures.

**Activity Outcomes:**

The students will learn how various sorting techniques e.g. Selection Sort, Insertion Sort, Bubble Sort and Merge Sort work on given data set. This will help them understand the difference among these sorting techniques.

**Theory Review (10 Minutes):****Sorting**

Sorting is a process to put elements of a list in certain order, to make searching easier. Various sorting algorithms are used in computer science to arrange the data. Some of them are efficient in terms of running time e.g. Merge Sort and Quick Sort whereas some take more running time e.g. Selection Sort, Insertion Sort and Bubble Sort. But the latter are studied for their simplicity of code. Following is the brief description of Selection Sort, Insertion Sort and Bubble Sort which take  $O(n^2)$  to run.

**1. Selection Sort:**

Given an array of „n“ unsorted elements, Selection sort works in the following manner:

**Step 1:**

Find the minimum value in the list of n elements.

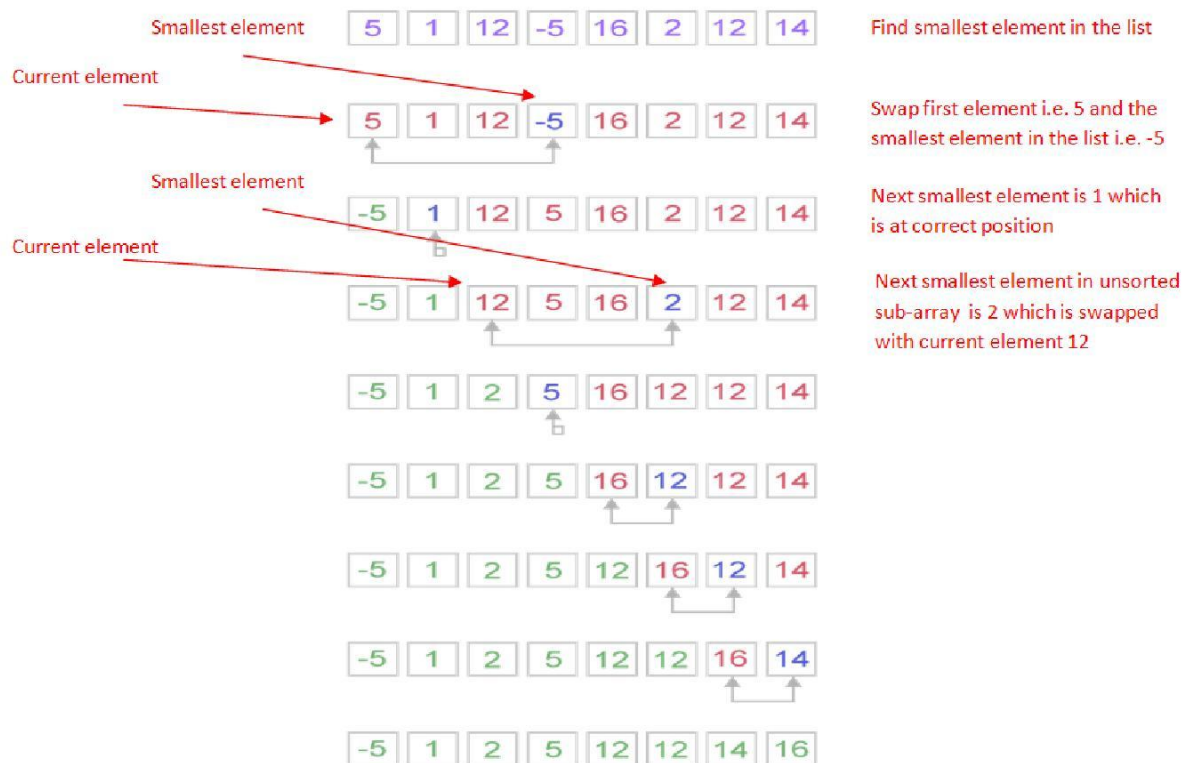
**Step 2:**

Swap this minimum value with the value in the first position i.e. at index 0. It gives smallest value at the first position.

**Step 3:**

Repeat steps 1 and 2, for all the remaining elements, till we get sorted array.



**Example:****2. Insertion Sort:**

Given an array of „n“ unsorted elements, Insertion sort works in the following manner:

**Step 1:**

Pick the element at index 1, i.e. skip element at index 0, and insert it into its correct position. This is done by comparing this element with the previous element. As long as the current element is smaller than the previous element, it is swapped with it, until it gets its correct position.

**Step 2:**

Step 1 is repeated for all the remaining elements, till the given array is sorted.



**Example:**

13	10	-15	15	12
----	----	-----	----	----

Skipping first element i.e. 13, we pick second element 10 and insert it into its correct position

10	13	-15	15	12
----	----	-----	----	----

Now we pick third element -15 and insert it into its correct position

-15	10	13	15	12
-----	----	----	----	----

Now we pick fourth element 15 and insert it into its correct position. It is in fact in its correct position.

-15	10	13	15	12
-----	----	----	----	----

Now we pick last element 12 and insert it into its correct position.

-15	10	12	13	15
-----	----	----	----	----

All elements are sorted.

**3. Bubble Sort:**

Given an array of „n“ unsorted elements, Bubble sort works in the following manner:

**Step 1:**

Compare two consecutive elements. If they are out of place, swap them.

**Step 2:**

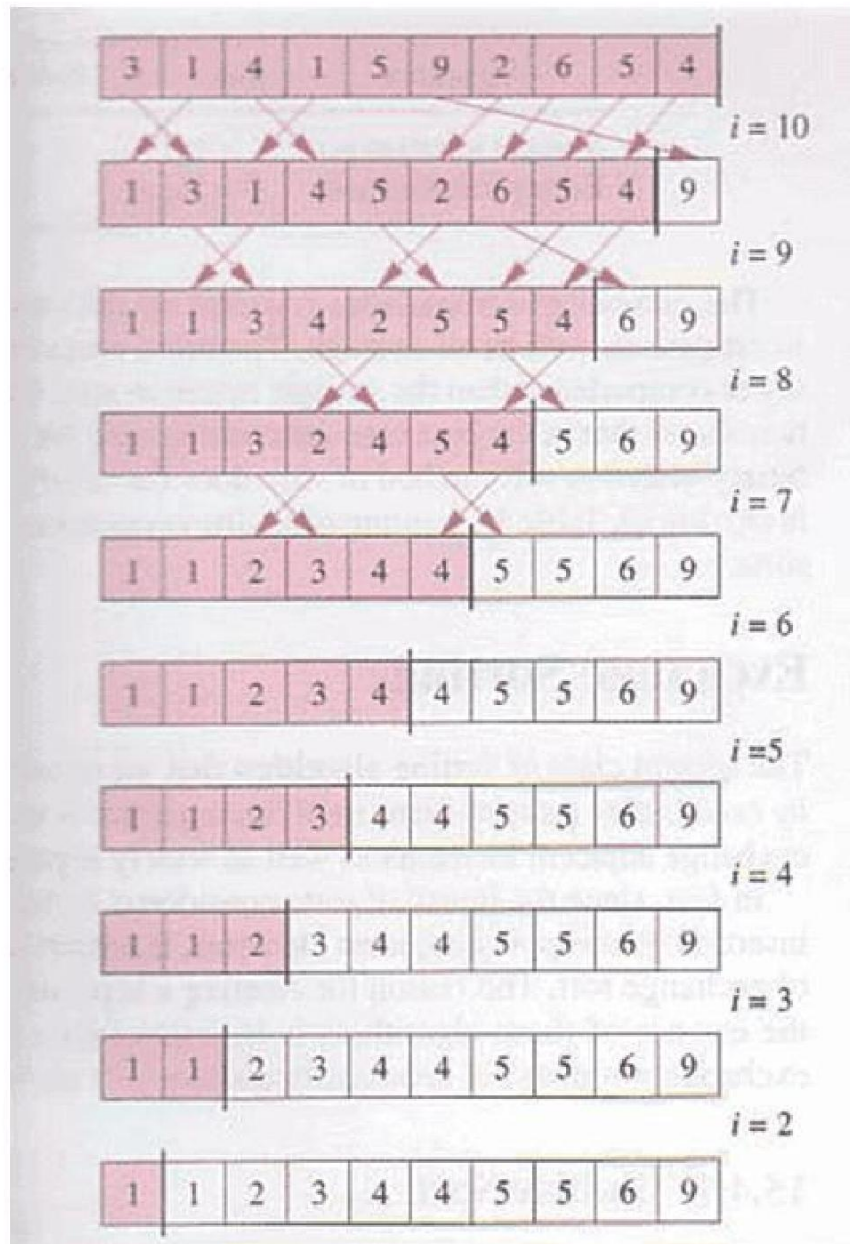
Repeat step 1 for all the elements till you compare last two elements. At the end of first iteration, largest element will be at the last position.

**Step 3:**

Repeat steps 1 and 2 for all the remaining elements till you get the sorted array.



Example:



### Practice Activity with Lab Instructor: (10 Minutes)

1. Create a project with the name "Lab12".
2. Create a main class with the name "InsertionSort" within this project.
3. Type the following code in this main class.

```
1 import java.util.Scanner;
2 public class InsertionSort {
3
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6         System.out.println("How many elements? ");
7         int n = sc.nextInt();
8         int[] a = new int[n];
9         int i, j, temp;
10        System.out.println("Enter the elements of array");
11        for(i=0; i<a.length; i++) {
12            System.out.print("Enter element "+(i+1)+"----> ");
13            a[i] = sc.nextInt();
14            System.out.println();
15        }
16        System.out.println();
17        System.out.println("Array elements before sorting are");
18        for(i=0; i<a.length; i++) {
19            System.out.print(a[i]+" ");
20        }
21
22        Insertion_Sort(a);
23
24        System.out.println();
25        System.out.println("Array elements after sorting are");
26        for(i=0; i<a.length; i++) {
27            System.out.print(a[i]+" ");
28        }
29
30        System.out.println();
31    }
32
33
34
35    public static void Insertion_Sort(int[] x){
36        int key, j;
37        for(int i = 1; i<x.length; i++) {
38            key = x[i];
39            j = i-1;
40            while(j >= 0 && x[j] > key) {
41                x[j+1] = x[j];
42                j = j-1;
43            }
44            x[j+1] = key;
45        }
46    }
47 }
```



4. When you run this program, it displays the following output.

```
Output - Project1 (run) ✖
run:
How many elements?
10
Enter the elements of array
Enter element 1---> 5
Enter element 2---> 6
Enter element 3---> 9
Enter element 4---> 7
Enter element 5---> 2
Enter element 6---> 4
Enter element 7---> 3
Enter element 8---> 8
Enter element 9---> 10
Enter element 10---> 13

Array elements before sorting are
5 6 9 7 2 4 3 8 10 13
Array elements after Insertion sorting are
2 3 4 5 6 7 8 9 10 13
BUILD SUCCESSFUL (total time: 31 seconds)
```

### Practice Activity:

Your job is to make changes in the above code so that we get the elements in descending order.





**LAB EXERCISES: (60 Minutes)**

- 1)** Show the contents of the array below being sorted using Insertion Sort at the end of each loop iteration.

Initial	2	8	3	6	5	1	4	7
Sorted	1	2	3	4	5	6	7	8

- 2)** Show the contents of the array below being sorted using Selection Sort at the end of each loop iteration. As shown in the class, please run the algorithm by placing the smallest item in first place.

Initial	6	2	8	1	3	7	5	4
Sorted	1	2	3	4	5	6	7	8

- 3)** Show the contents of the array below being sorted using Bubble Sort at the end of each loop iteration. As shown in the class, please run the algorithm by placing the largest item in first place.

Initial	4	2	6	5	7	1	8	3
Sorted	1	2	3	4	5	6	7	8



- 4) When Merge Sort is run on an array of size 8, the merge function gets called 7 times. Consider running Merge Sort on the array below. What would the contents of the array be right before the 7<sup>th</sup> call to the Merge function?

Initial	7	2	1	5	8	3	4	6
Before 7 <sup>th</sup> Merge								

- 5) Show the result of running Partition on the array below using the leftmost element as the pivot element. Show what the array looks like after each swap.

Initial	5	2	1	7	8	3	4	6
After Partition								

- 6) Show the contents of the array below after each merge occurs in the process of Merge-Sorting the array below.

Initial	3	6	8	1	7	4	5	2
Last	1	2	3	4	5	6	7	8

- 7) The following MergeSort() java code and Merge() algorithm are used to sort elements in the array in ascending order. Your job is to write java code for Merge() method and make changes in it so that it arranges the elements of the array in **descending order**.





**Code for MergeSort() Method:**

```
public static void MergeSort(int[] values, int start, int end) {
    int mid;

    // Check if our sorting range is more than one element.
    if (start < end) {
        mid = (start+end)/2;

        // Sort the first half of the values.
        MergeSort(values, start, mid);

        // Sort the last half of the values.
        MergeSort(values, mid+1, end);

        // Put it all together.
        Merge(values, start, mid+1, end);
    }
}
```

**Code for Merge() Method:**

```
public static void Merge(int[] values, int p, int q, int r) {
    int[] B = new int[values.length];
    int i, j, k;
    i = k = p;
    j = q + 1;
    while ((i <= q) && (j <= r)) {
        if (values[i] <= values[j]) {
            B[k++] = values[i++];
        }
        else {
            B[k++] = values[j++];
        }
    } //End of while
    while (i <= q) { //If left subarray has more elements than right
        //subarray
        B[k++] = values[i++];
    }
}
```



```
while ( j <= r){ //If right subarray has more elements than
                left // subarray
    B[k++] = values[j++];
}
for (int n = p ; n < r ; n++) {
    values[n] = B[n]; //Storing all elements of temporary array B
                    //in original array A
}
}
```

