Practical 1: To perform Time analysis of Bubble sort, Selection sort & Insertion sort

The code below here comprises of all three algorithm:

```
import java.util.*;
public class BubbleSort{


   public static void cnt_function_bubble(int n, int arr[])
   {
      System.out.println("Loop starts bubble"); for
      (int i = 0; i < n - 1; i++) {
         for (int j = 0; j < n - i - 1; j++) { if
            (arr[j] > arr[j + 1]) {

               int temp = arr[j];
               arr[j] = arr[j + 1];
               arr[j + 1] = temp;
            }
         }
      }
      System.out.println("Loop ends bubble");
   }
   public static void cnt_function_selection(int n, int arr[]){
      System.out.println("Loop starts selection");
      for (int i = 0; i < n - 1; i++) {

         int min_idx = i;
```

```
        for(int j=i+1;j<n;j++) if
            (arr[j]<arr[min_idx])
                min_idx = j;



        int temp = arr[min_idx];
        arr[min_idx] = arr[i]; arr[i]
        = temp;
    }
    System.out.println("Loop ends selection");
}
    public static void cnt_function_insertion(int n, int arr[])
    {
        System.out.println("Loop starts insertion");

        for(inti=1;i<n;i++){ int
            key = arr[i];
            int j = i - 1;


            while(j>=0&&arr[j]>key){ arr[j
                + 1] = arr[j];
                j = j - 1;
            }
            arr[j + 1] = key;
        }
        System.out.println("Loop ends insertion");
    }

  public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
        int size =sc.nextInt();
        Random rand = new Random();
        int lol_random[] = new int[size]; int
        lol_sorted[]=newint[size];
        int lol_rev_sorted[] = new int[size];
        for(inti=0;i<size;i++)lol_random[i]=rand.nextInt(100000); for (int i =
        0; i < size; i++) lol_sorted[i] = i+1;
        for (int i = size; i >= 1; i--) lol_rev_sorted[i-1] = i;
        //for (int i = size; i >= 1; i--) System.out.print( lol_rev_sorted[i-1] + " ");
        //HERE WE CAN USE THE FUNCTION OF ANY SORTING ALGORITHM WE WANT
        //SO TO REDUCE THE REDUNDANCY I HAVE ONLY CALLED 1 ALGORITHM HERE
        long start = System.currentTimeMillis();
        cnt_function_insertion(size, lol_sorted);
        long end = System.currentTimeMillis();
        System.out.println("Time Taken " + (end - start) + "ms");

    }

}
```

ANALYSIS TABLE :

TYPE OF ALGO	INPUT SIZE	1k	10k	50k	100k	150k	200k
Bubble_random_array		7ms	185ms	4724ms	19055ms	43128ms	76787ms
Bubble_sorted_array		4ms	21ms	351ms	1481ms	3709ms	6810ms
Bubble_rev_array		5ms	35ms	802ms	3080ms	6954ms	13005ms
Selection_random_array		3ms	41ms	976ms	3913ms	8633ms	15347ms
Selection_sorted_array		3ms	22ms	395ms	1569ms	3537ms	6234ms

Selection_rev_array	4ms	20ms	399ms	1573ms	3678ms	6301ms
Insertion_random_array	3ms	24ms	275ms	1022ms	2311ms	3889ms
Insertion_sorted_array	0ms	0ms	1ms	3ms	4ms	5ms
Insertion_rev_array	5ms	27ms	592ms	2001ms	5033ms	7769ms

Practical 2: Implementation & Time Analysis of Linear search and Binary search.

```
import java.util.*;

public class searching {

   public static int LinearSearch(int arr[], int find){ int n
      =arr.length;
      System.out.println("Linear search starts");
      for(int i = 0; i < n; i++)
      {
         if(arr[i]==find)
            return i;
      }
      return -1;

   }
   public static int BinarySearch(int arr[], int find){ int
      left = 0, right = arr.length - 1;
      System.out.println("Binary search starts"); while
      (left <= right) {
         int mid = left + (right - left) / 2;


         if (arr[mid]==find)
            return mid;
```

```
        if (arr[mid] < find)
            left = mid + 1;


        else
            right = mid -1;
    }


    return -1;

}

public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    int n =sc.nextInt();
    Random rand = new Random();
    int find = sc.nextInt();
    int lol_random[] = new int[n];
    int lol_sorted[] = new int[n];
    int lol_rev_sorted[] = new int[n];
    for(int i=0;i<n;i++) lol_random[i]=rand.nextInt(100000); for
    (int i = 0; i < n; i++) lol_sorted[i] = i+1;
    for (int i = n; i >= 1; i--) lol_rev_sorted[i-1] = i;
```

```
        long start = System.currentTimeMillis();
        BinarySearch(lol_sorted,find);
        long end = System.currentTimeMillis();
        System.out.println("Time Taken " + (end - start) + "ms");


    }
}
```

Time Analysis Table :

Algorithm	Best Case	Worst Case	Average Case
Linear Search	0ms	3ms	2ms
Binary Search	0ms	0ms	0ms

Performed for 10lac input.

3150703 Analysis & Design of Algorithms

Practical 3: To perform time analysis of Merge Sort

Code:

```
import
java.util.Random;
import
java.util.Scanner;

public class MergeSort {
   void merge(int arr[], int l, int m, int r)
   {

      int size1 = m - l +
      1; int size2 = r -
      m;



      int left[] = new int[size1];
      int right[] = new
      int[size2];



      for (int i = 0; i < size1; ++i)
         left[i] = arr[l + i];
      for (int j = 0; j < size2; ++j)
         right[j] = arr[m + 1 + j];
```

```
    int i = 0, j = 0;

    int k = l;
    while (i < size1 && j < size2)
       { if (left[i] <= right[j]) {
          arr[k] = left[i];
          i++;
       }
       else {
          arr[k] = right[j];
          j++;
       }
       k++;
    }



    while (i < size1)
       { arr[k] =
       left[i]; i++;
       k++;
    }



    while (j < size2)
       {    arr[k]    =
       right[j]; j++;
       k++;
    }
  }
```

```
    void sort(int arr[], int l, int r)
{
        if (l < r) {

            int m = (l + r) /
            2; sort(arr, l, m);
            sort(arr, m + 1, r);
            merge(arr, l, m, r);
        }
    }

  public static void main(String[] args){
        Scanner sc = new
        Scanner(System.in); int n =
        sc.nextInt();
        Random rand = new Random();

        int lol_random[] = new
        int[n]; int lol_sorted[] = new
        int[n];
        int lol_rev_sorted[] = new int[n];
        for (int i = 0; i < n; i++) lol_random[i] =
        rand.nextInt(100000); for (int i = 0; i < n; i++) lol_sorted[i]
        = i+1;
        for (int i = n; i >= 1; i--) lol_rev_sorted[i-1] =
        i; MergeSort ms = new MergeSort();
        long start =
        System.currentTimeMillis();
        ms.sort(lol_rev_sorted,0,n-1);
```

```
        long end = System.currentTimeMillis();
        System.out.println("Time Taken " + (end - start) +
        "ms");

    }
}
```

TIME ANALYSIS TABLE:

TYPE OF ALGO	INPUT SIZE	1k	10k	50k	100k	150k	200k
Merge_random_array		1ms	2ms	12ms	19ms	28ms	35ms
Merge_sorted_array		0ms	2ms	8ms	16ms	17ms	23ms
Merge_rev_array		1ms	2ms	8ms	13ms	18ms	23ms

3150703 Analysis & Design of Algorithm

Practical 4: Implementation and Time analysis of factorial program using iterative and recursive method.

```
import java.util.Random;
import java.util.Scanner;


public class factorial {
    static int factorial_recursive(int n)
    {
        if (n == 0)
            return 1;


        return n * factorial_recursive(n - 1);
    }


   static int factorial_iterative(int n)
    {
        int res = 1, i;
        for (i = 2; i <= n; i++)
            res *= i;
        return res;
    }
 public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        long start = System.currentTimeMillis();
        factorial_iterative(n);
        long end = System.currentTimeMillis();
        System.out.println("Time Taken " + (end - start) + "ms");
  }
}
```

3150703 Analysis & Design of Algorithm

TIME ANALYSIS:

	INPUT Value					
	5	**10**	**20**	**25**	**50**	**65**
Iterative	0ms	0ms	0ms	0ms	1ms	1ms
Recursive	0ms	0ms	0ms	0ms	0ms	0ms