|  |  |
| --- | --- |
| Student Name | Uzair Hussain |
| Roll Number | 21SW085 |
| Section # | 3rd or III |
| Lab # | 4th – Sorting in Arrays |

**Task#01**

**Code:**

import java.util.Arrays;

class Sorts{

    // Bubble sort of 1D Array

    public int[] Bubble\_Sort(int [] arr){

        for (int i=0; i< arr.length; i++){

            for (int j=0; j< arr.length-1; j++){

                if (arr[j]>arr[j+1]){

                    int temp = arr[j];

                    arr[j] = arr[j+1];

                    arr[j+1] = temp;

                }

            }

        }

        return arr;

    }

    // Insertion Sort of 1D Array

    public int[] Insertion\_Sort(int [] arr){

        for (int i=1; i< arr.length; i++){

            int temp = arr[i];

            int j = i-1;

            while (j>=0 && arr[j]>temp){

                arr[j+1] = arr[j];

                j--;

            }

            arr[j+1] = temp;

        }

        return arr;

    }

    //Selction SOrt of 1D Array

    public int[] Selection\_Sort(int [] arr){

        for (int i=0; i< arr.length-1; i++){

            int minIndex = i;

            for (int j=i+1; j< arr.length; j++){

                if (arr[j]<arr[minIndex]){

                    minIndex = j;

                }

            }

                int temp = arr[i];

                arr[i] = arr[minIndex];

                arr[minIndex] = temp;

        }

        return arr;

    }

    // Quick Sort

    public void quickSort(int[] array, int low, int high) {

        if (array == null || array.length == 0){

            return;

        }

        if (low >= high){

            return;

        }

        // pick the pivot

        int middle = low + (high - low) / 2;

        int pivot = array[middle];

        // make left < pivot and right > pivot

        int i = low, j = high;

        while (i <= j) {

            while (array[i] < pivot) {

                i++;

            }

            while (array[j] > pivot) {

                j--;

            }

            if (i <= j) {

                int temp = array[i];

                array[i] = array[j];

                array[j] = temp;

                i++;

                j--;

            }

        }

        // recursively sort two sub parts

        if (low < j)

            quickSort(array, low, j);

        if (high > i)

            quickSort(array, i, high);

    }

    public long TimeCalculator(long b, long a){

        return (a-b);

    }

}

public class Task1\_1D\_Sorting {

    public static void main(String[] args) {

        int [] array = {4, -6, 5, 7, 8, 9, 1, 10, -9};

        Sorts s = new Sorts();

        long before1 = System.nanoTime();

        System.out.println("Bubble Sort()  -->  "+Arrays.toString(s.Bubble\_Sort(array)));

        long after1 = System.nanoTime();

        long before2 = System.nanoTime();

        System.out.println("Insertion Sort()  -->   "+Arrays.toString(s.Insertion\_Sort(array)));

        long after2 = System.nanoTime();

        long before3 = System.nanoTime();

        System.out.println("Selection Sort()  -->   "+Arrays.toString(s.Selection\_Sort(array)));

        long after3 = System.nanoTime();

        long before4 = System.nanoTime();

        s.quickSort(array, 0, array.length-1);

        System.out.println("Quick Sort()  -->   "+Arrays.toString(array));

        long after4 = System.nanoTime();

        System.out.println("\n\nExecution time of Bubble Sort() method is -->   "+s.TimeCalculator(before1, after1));

        System.out.println("Execution time of Insertion Sort() method is -->   "+s.TimeCalculator(before2, after2));

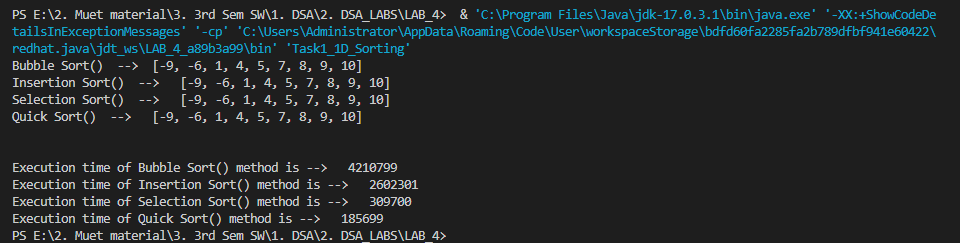
        System.out.println("Execution time of Selection Sort() method is -->   "+s.TimeCalculator(before3, after3));

        System.out.println("Execution time of Quick Sort() method is -->   "+s.TimeCalculator(before4, after4));

    }

}

**Output 1:**

****

**Code 2:**

import java.util.Arrays;

import java.util.Scanner;

class Test {

    void binary\_search\_2D(int[][] arr, int target) {

        int row = 0;

        int col = arr[row].length - 1;

        while (row < arr.length && col >= 0) {

            if (arr[row][col] == target) {

                System.out.println("The row index is  " + row + "\n" + "The column index is " + col);

            }

            // Target lies in further row

            if (arr[row][col] < target) {

                row++;

            }

            // Target lies in previous column

            else {

                col--;

            }

        }

    }

}

class Task2\_BinarySearch\_2D {

    public static void main(String[] args) {

        Test t = new Test();

        Scanner sc = new Scanner(System.in);

        int[][] arr2 = { { 1, 2, 3, 4, 5 }, { 6, 7, 8, 9, 10 } };

        System.out.print("Enter number you want to search in 2D search: ");

        int key2 = sc.nextInt();

        long before2 = System.nanoTime();

        t.binary\_search\_2D(arr2, key2);

        long after2 = System.nanoTime();

        System.out.println("2D linear search method consuming :" + (after2 - before2)

                + " milliseconds");

    }

}

**Output 2:**

**Text

Description automatically generated**

**The End!**