CS263 ASSIGNMENT 1

NAME:

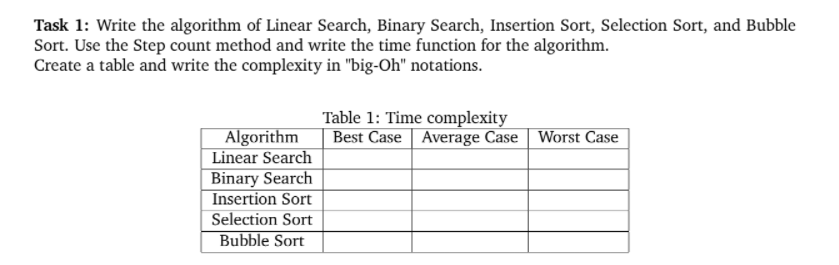
ARCHIT AGRAWAL

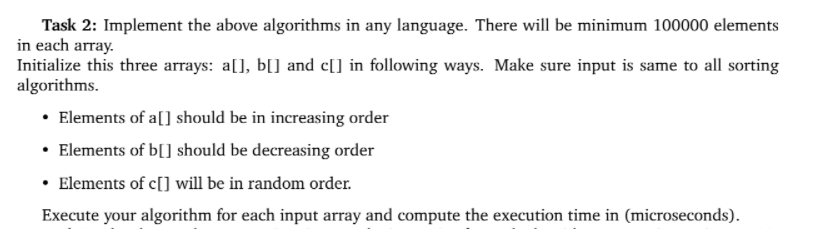
ROLL NO. :

202052307

SECTION:

A





***CODE***

import java.util.\*;

public class Search{

    public static int linearSearch(int[] arr, int num){

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

            if(arr[i] == num) return i;

        }

        return -1;

    }

    public static int binarySearch(int[] nums, int target) {

        int ans = -1;

        int i = 0;

        int j = nums.length - 1;

        while(j >= i){

            int mid = (i + j) / 2;

            if(nums[mid] == target){

                ans = mid;

                break;

            } else if(nums[mid] < target){

                i = mid + 1;

            } else {

                j = mid - 1;

            }

        }

        return ans;

    }

    public static void insertionSort(int[] arr){

        //Insertion Sort Algorithm

        int key;

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

            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;

        }

    }

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the size of array: ");

        int n = sc.nextInt();

        Random rand = new Random();

        int[] arr = new int[n];

        System.out.println("Creating the array using Random class: ");

        for(int i = 0; i < n; i++){

            arr[i] = rand.nextInt(10000);

        }

        System.out.println();

        int x = arr[rand.nextInt(n)];

        double start = System.nanoTime();

        linearSearch(arr, 0);

        double end = System.nanoTime();

        System.out.println("Time taken in Linear Search (best case) in milliseconds is : "+ (end - start)/1000000.0);

        start = System.nanoTime();

        linearSearch(arr, arr.length - 1);

        end = System.nanoTime();

        System.out.println("Time taken in Linear Search (worst case) in milliseconds is : "+ (end - start)/1000000.0);

        start = System.nanoTime();

        linearSearch(arr, x);

        end = System.nanoTime();

        System.out.println("Time taken in Linear Search (average case) in milliseconds is : "+ (end - start)/1000000.0);

        System.out.println();

        System.out.println("Sorting the array..");

        insertionSort(arr);

        System.out.println();

        x = arr[rand.nextInt(n)];

        start = System.nanoTime();

        binarySearch(arr, arr.length/2);

        end = System.nanoTime();

        System.out.println("Time taken in Binary Search (best case) in milliseconds is : "+ (end - start)/1000000.0);

        start = System.nanoTime();

        binarySearch(arr, arr.length - 1);

        end = System.nanoTime();

        System.out.println("Time taken in Binary Search (worst case) in milliseconds is : "+ (end - start)/1000000.0);

        start = System.nanoTime();

        binarySearch(arr, x);

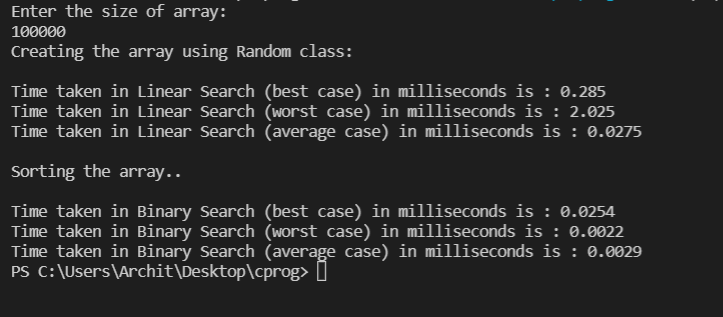
        end = System.nanoTime();

        System.out.println("Time taken in Binary Search (average case) in milliseconds is : "+ (end - start)/1000000.0);

    }

}

***Output***



***CODE***

import java.util.\*;

public class Sort{

    public static void bubbleSort(Integer []arr){

        //Bubble Sort Algorithm

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

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

                if(arr[j] > arr[j + 1]){

                    int temp = arr[j];

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

                    arr[j + 1] = temp ;

                }

            }

        }

    }

    public static void selectionSort(Integer []arr){

        //Selection Sort Algorithm

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

            int greatest = arr[0];

            int tempJ = 0;

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

                if(arr[j] > greatest ){

                    greatest = arr[j] ;

                    tempJ = j ;

                }

            }

            int tempA = arr[arr.length - i - 1] ;

            arr[arr.length - i - 1] = greatest;

            arr[tempJ] = tempA;

        }

    }

    public static void insertionSort(Integer []arr){

        //Insertion Sort Algorithm

        int key;

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

            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;

        }

    }

    public static void main(String []args){

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the size of array");

        int n = sc.nextInt();

        System.out.println("1. If you want the input array to be randomly arranged, enter 1");

        System.out.println("2. If you want the input array to be arranged in increasing order, enter 2");

        System.out.println("3. If you want the input array to be arranged in descending order, enter 3");

        int order = sc.nextInt();

        Integer[] a = new Integer[n];

        Integer[] b = new Integer[n]; //a copy of a[]

        Integer[] c = new Integer[n]; //another copy of a[]

        Random rand = new Random();

        for (int i = 0; i < n; i++) {

            a[i] = rand.nextInt(9000) + 1000;

            //System.out.print(a[i] + " ");

        }

        System.out.println();

        for(int i = 0; i < n; i++){

            b[i] = a[i];

            c[i] = a[i];

        }

        //a[] will be bubble sorted

        //b[] will be selection sorted

        //c[] will be insertion sorted

        // as random will give a new number everytime that is why

        //a copy of a[] is created to ensure that all the sorting

        //methods gets the same array

        if (order == 2) {

            Arrays.sort(a);

            Arrays.sort(b);

            Arrays.sort(c);

        } else if (order == 3) {

            Arrays.sort(a, Collections.reverseOrder());

            Arrays.sort(b, Collections.reverseOrder());

            Arrays.sort(c, Collections.reverseOrder());

        }

        double start = System.nanoTime();

        bubbleSort(a);

        double end = System.nanoTime();

        double time\_in\_bubble = (end - start)/1000000.0;

        start = System.nanoTime();

        selectionSort(b);

        end = System.nanoTime();

        double time\_in\_selection = (end - start)/1000000.0;

        start = System.nanoTime();

        insertionSort(c);

        end = System.nanoTime();

        double time\_in\_insertion = (end - start)/1000000.0;

        System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

        System.out.printf("Time taken in bubble sort in milliseconds : %.3f ",time\_in\_bubble);

        System.out.println();

        System.out.printf("Time taken in selection sort milliseconds : %.3f ",time\_in\_selection);

         System.out.println();

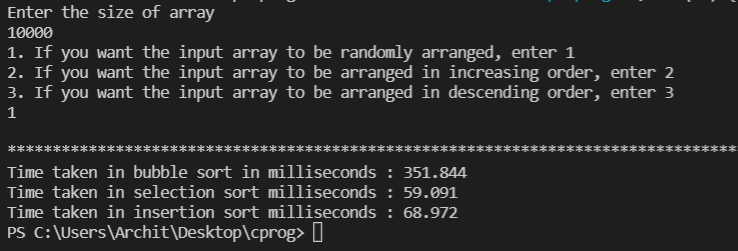
        System.out.printf("Time taken in insertion sort milliseconds : %.3f ",time\_in\_insertion);

    }

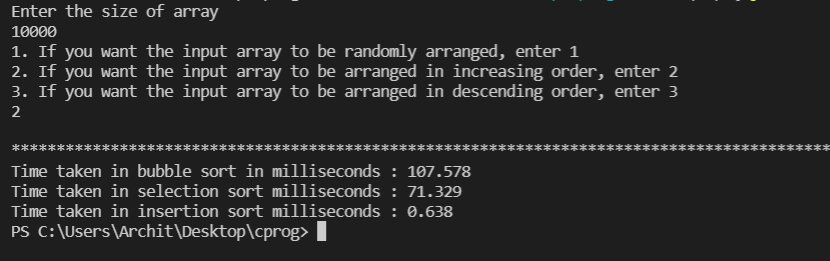
}

***Output***

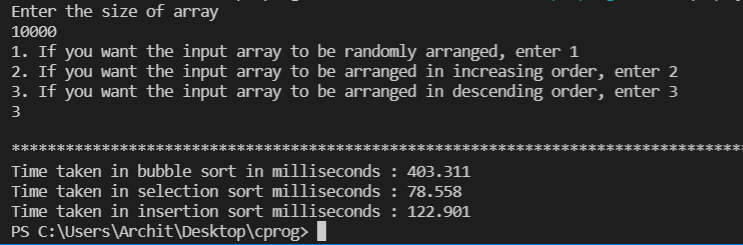
* ***Input is randomly arranged***

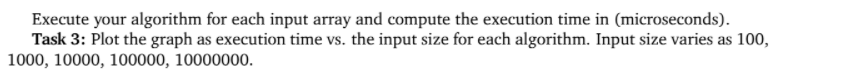


* ***Input is non-decreasingly arranged***



* ***Input is non-incsreasingly arranged***





*Bubble Sort*

|  |  |
| --- | --- |
| **Size** | **Time(in milliseconds)** |
| 100 | 0.554 |
| 500 | 8.001 |
| 1000 | 19.526 |
| 5000 | 130.355 |
| 10000 | 365.168 |

*Selection Sort*

|  |  |
| --- | --- |
| **Size** | **Time(in milliseconds)** |
| 100 | 0.139 |
| 500 | 3.748 |
| 1000 | 5.276 |
| 5000 | 19.791 |
| 10000 | 58.391 |

*Insertion Sort*

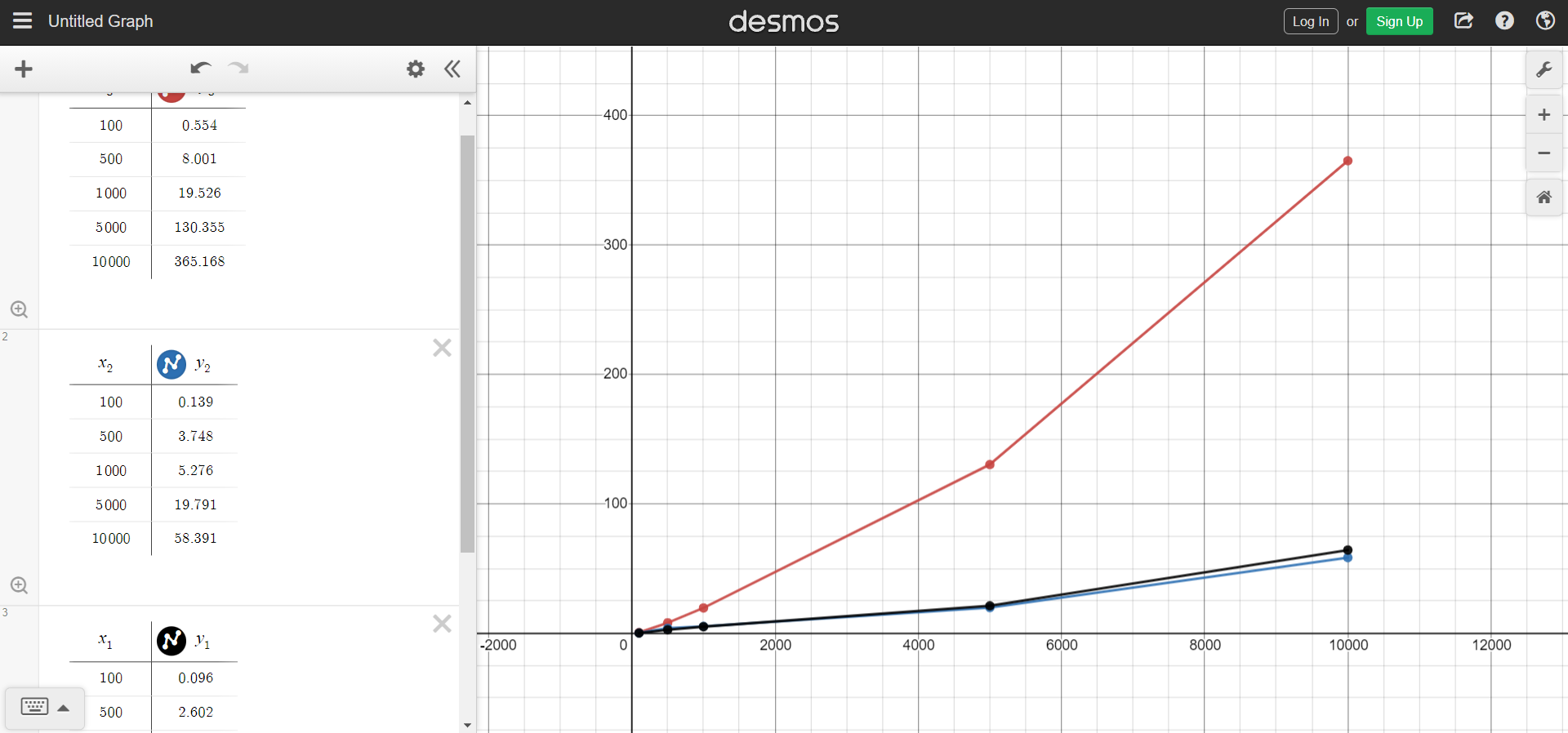
|  |  |
| --- | --- |
| **Size** | **Time(in milliseconds)** |
| 100 | 0.096 |
| 500 | 2.602 |
| 1000 | 5.028 |
| 5000 | 21.168 |
| 10000 | 64.239 |

***Graph***

Bubble Sort (in Red)

Selection Sort (in Blue)

Insertion Sort (in Black)



The codes weren’t executing for the input sizes mentioned in the assignment, hence different array sizes are used.