|  |  |
| --- | --- |
| Name | Sandeep |
| RollNo: | 21sw015 |
| Section | One |
| Lab no: | 4 |

**QUESTION NO 01**

Sort the given 05 students in ascending order of their heights

i.Bubble sort //Create method BubbleSort1D(int[] A)

ii.Selection sort //Create method SelectionSort1D(int[] A)

iii.Insertion sort//Create method InsertionSort1D(int[] A)

iv.Quick sort //Create method QuickSort1D(int[] A)

v.Display the execution time of sorting algos and examine which one is the fastestand explain why?

**CODE:**

package DSA\_practical;

public class lab\_4\_t1 {

    // sorting with help of different algorithim

    public static void selectionsort(int[] arr) {

        int minindex = 0;

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

            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;

        }

    }

    public static void bubbleSort(int[] arr) {

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

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

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

                    int temp = arr[j + 1];

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

                    arr[j] = temp;

                }

            }

        }

    }

    public static void insertionSort(int[] arr) {

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

            int currentElement = arr[i];

            int j = i - 1;

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

                arr[j + 1] = arr[j];

                j--;

            }

            arr[j + 1] = currentElement;

        }

    }

    public static int partition(int[] arr, int beg, int last) {

        int pivot = arr[beg];

        int lm = beg + 1;

        int rm = last, temp;

        do {

            while (arr[rm] > pivot)

                rm--;

            while (arr[lm] <= pivot && lm < last)

                lm++;

            if (lm < rm) {

                temp = arr[rm];

                arr[rm] = arr[lm];

                arr[lm] = temp;

            }

        } while (lm < rm);

        arr[beg] = arr[rm];

        arr[rm] = pivot;

        return rm;

    }

    public static void quickSort(int[] arr, int beg, int last) {

        int p;

        if (beg < last) {

            p = partition(arr, beg, last);

            quickSort(arr, beg, p - 1);

            quickSort(arr, p + 1, last);

        }

    }

    public static void displayArray(int[] arr) {

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

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

        }

    }

    public static void main(String[] args) {

        int[] newArr = { 180, 165, 150, 170, 145, -34, 8, -2, 0, -23, 8, 234 };

        System.out.println("\nSelection sort");

        long before = System.nanoTime();

        selectionsort(newArr);

        long after = System.nanoTime();

        long total\_time\_ss = after - before;

        System.out.println("Total execution time of selection sort  is : " + total\_time\_ss + " nano second");

        displayArray(newArr);

        System.out.println("\n\nBubble sort ");

        before = System.nanoTime();

        bubbleSort(newArr);

        after = System.nanoTime();

        long total\_time\_bs = after - before;

        System.out.println("Total execution time of bubble sort is : " + total\_time\_bs + " nano second");

        displayArray(newArr);

        System.out.println("\n\nInsertion sort ");

        before = System.nanoTime();

        insertionSort(newArr);

        after = System.nanoTime();

        long total\_time\_is = after - before;

        System.out.println("Total execution time of Insertion sort is : " + total\_time\_is + " nano second");

        displayArray(newArr);

        System.out.println("\n\nQuick sort ");

        before = System.nanoTime();

        quickSort(newArr, 0, newArr.length - 1);

        after = System.nanoTime();

        long total\_time\_qs = after - before;

        System.out.println("Total execution time of Quick sort is : " + total\_time\_qs + " nano second");

        displayArray(newArr);

        // comparesion of execution time among all four types of sorting that which has

        // took least time :

        long x = Math.min(total\_time\_ss, total\_time\_bs);

        long y = Math.min(total\_time\_is, total\_time\_qs);

        long least\_time\_taken = 0L;

        if (x < y)

            least\_time\_taken = x;

        else

            least\_time\_taken = y;

        System.out.println("\n\n");

        if (least\_time\_taken == total\_time\_ss)

            System.out.println("selection sort take the least execution time to sort an array " + least\_time\_taken);

        else if (least\_time\_taken == total\_time\_bs)

            System.out.println("bubble sort take the least execution time to sort an array " + least\_time\_taken);

        else if (least\_time\_taken == total\_time\_is)

            System.out.println("Insertion sort take the least execution time to sort an array " + least\_time\_taken);

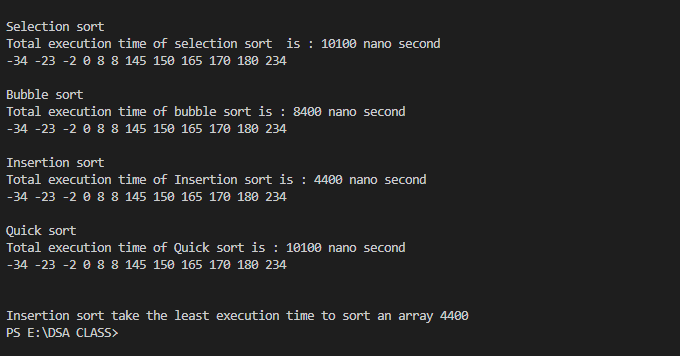
        else if (least\_time\_taken == total\_time\_qs)

            System.out.println("Quick sort take the least execution time to sort an array " + least\_time\_taken);

    }

}

**OUTPUT:**

****

**Q NO: 02**

Sort the following 2D array using (bubble, selection, insertion, and quick sort)

2 10 15

5 1 3

6 9 4

**Code:**

package DSA\_practical;

import org.w3c.dom.ls.LSException;

public class Lab\_04\_t2 {

    public static void selectionsort(int[] arr) {

        int minindex = 0;

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

            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;

        }

    }

    public static void bubbleSort(int[] arr) {

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

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

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

                    int temp = arr[j + 1];

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

                    arr[j] = temp;

                }

            }

        }

    }

    public static void insertionSort(int[] arr) {

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

            int currentElement = arr[i];

            int j = i - 1;

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

                arr[j + 1] = arr[j];

                j--;

            }

            arr[j + 1] = currentElement;

        }

    }

    public static int partition(int[] arr, int beg, int last) {

        int pivot = arr[beg];

        int lm = beg + 1;

        int rm = last, temp;

        do {

            while (arr[rm] > pivot)

                rm--;

            while (arr[lm] <= pivot && lm < last)

                lm++;

            if (lm < rm) {

                temp = arr[rm];

                arr[rm] = arr[lm];

                arr[lm] = temp;

            }

        } while (lm < rm);

        arr[beg] = arr[rm];

        arr[rm] = pivot;

        return rm;

    }

    public static void quickSort(int[] arr, int beg, int last) {

        int p;

        if (beg < last) {

            p = partition(arr, beg, last);

            quickSort(arr, beg, p - 1);

            quickSort(arr, p + 1, last);

        }

    }

    public static void displayArray(int[] arr) {

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

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

        }

    }

    public static void main(String[] args) {

        int[][] arr = { { 2, 10, 15 }, { 5, 1, 3 }, { 6, 9, 4 } };

        int size = (arr.length) \* (arr[0].length);

        int[] newArr = new int[size];

        int k = 0;

        System.out.println("  2D array before sorting ");

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

            for (int j = 0; j < arr[0].length; j++) {

                newArr[k] = arr[i][j];

                System.out.print(arr[i][j] + " ");

                k++;

            }

            System.out.println();

        }

        System.out.println();

        System.out.println("\nSelection sort");

        long before = System.nanoTime();

        selectionsort(newArr);

        long after = System.nanoTime();

        long total\_time\_ss = after - before;

        System.out.println("Total execution time of selection sort  is : " + total\_time\_ss + " nano second");

        displayArray(newArr);

        System.out.println("\n\nBubble sort ");

        before = System.nanoTime();

        bubbleSort(newArr);

        after = System.nanoTime();

        long total\_time\_bs = after - before;

        System.out.println("Total execution time of bubble sort is : " + total\_time\_bs + " nano second");

        displayArray(newArr);

        System.out.println("\n\nInsertion sort ");

        before = System.nanoTime();

        insertionSort(newArr);

        after = System.nanoTime();

        long total\_time\_is = after - before;

        System.out.println("\n\nTotal execution time of Insertion sort is : " + total\_time\_is + " nano second");

        displayArray(newArr);

        System.out.println("\n\nQuick sort ");

        before = System.nanoTime();

        quickSort(newArr, 0, newArr.length - 1);

        after = System.nanoTime();

        long total\_time\_qs = after - before;

        System.out.println("Total execution time of Quick sort is : " + total\_time\_qs + " nano second");

        displayArray(newArr);

        // comparesion of execution time among all four types of sorting that which has

        // took least time :

        long x = Math.min(total\_time\_ss, total\_time\_bs);

        long y = Math.min(total\_time\_is, total\_time\_qs);

        long least\_time\_taken = 0L;

        if (x < y)

            least\_time\_taken = x;

        else

            least\_time\_taken = y;

        System.out.println("\n\n");

        if (least\_time\_taken == total\_time\_ss)

            System.out.println("selection sort take the least execution time to sort an array " + least\_time\_taken);

        else if (least\_time\_taken == total\_time\_bs)

            System.out.println("bubble sort take the least execution time to sort an array " + least\_time\_taken);

        else if (least\_time\_taken == total\_time\_is)

            System.out.println("Insertion sort take the least execution time to sort an array " + least\_time\_taken);

        else if (least\_time\_taken == total\_time\_qs)

            System.out.println("Quick sort take the least execution time to sort an array " + least\_time\_taken);

    }

}

**OUTPUT:**

**Text

Description automatically generated**