|  |  |
| --- | --- |
| Student Name | Asadullah |
| Roll Number | 21SW036 |
| Section # | 03 |
| Lab # | 04 |

**Task#01**

Question statement

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)

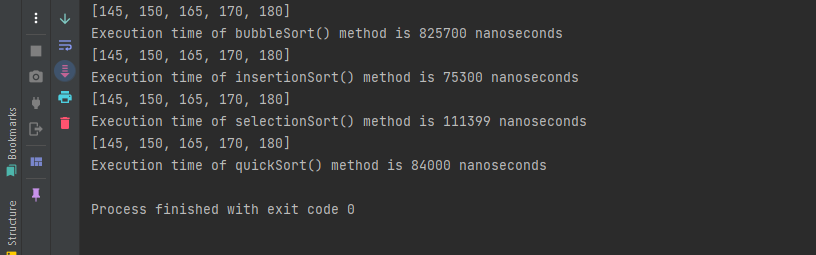
v.Display the execution time of sorting algos and examine which one is the fastest and explain why?

# SortingAlgosFor1DArray.Java

**Code:**

import java.util.Arrays;  
public class SortingAlgosFor1DArray {  
  
 public static int[] bubbleSort1D(int [] arr){  
 for (int i=0; i< arr.length; i++){  
 for (int j=0; j< arr.length; j++){  
 if (arr[j]>arr[i]){  
 int temp = arr[i];  
 arr[i] = arr[j];  
 arr[j] = temp;  
 }  
 }  
 }  
 return arr;  
 } // end of bubbleSort()  
  
 public static int[] insertionSort1D(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;  
 } // end of insertionSort() method  
  
 public static int[] selectionSort1D(int [] arr){  
 for (int i=0; i< arr.length-1; i++){  
 int min = i;  
 for (int j=i+1; j< arr.length; j++){  
 if (arr[j]<arr[min]){  
 min = j;  
 }  
 }  
 if (i!=min){  
 int temp = arr[i];  
 arr[i] = arr[min];  
 arr[min] = temp;  
 }  
 }  
 return arr;  
 } // end of selectionSort() method  
  
 public static int partition1D(int [] a, int lowerbound, int upperbound){  
 int pivot = a[upperbound];  
 int i = lowerbound-1;  
 for (int j=lowerbound; j<upperbound; j++){  
 if(a[j]<=pivot){  
 i++;  
 int temp = a[i];  
 a[i] = a[j];  
 a[j] = temp;  
 }  
 }  
 int temp = a[i+1];  
 a[i+1] = a[upperbound];  
 a[upperbound] = temp;  
 return (i+1);  
 }  
  
 public static void quickSort1D(int [] a, int lowerbound, int upperbound){  
 if(lowerbound<upperbound){  
 int location = *partition1D*(a, lowerbound, upperbound);  
 *quickSort1D*(a, lowerbound, location-1);  
 *quickSort1D*(a, location+1, upperbound);  
 }  
 }  
  
 public static void main(String[] args) {  
  
 int [] array = {180, 165, 150, 170, 145};  
  
 long before1 = System.*nanoTime*();  
 System.*out*.println(Arrays.*toString*(*bubbleSort1D*(array)));  
 long after1 = System.*nanoTime*();  
 System.*out*.println("Execution time of bubbleSort() method is "+(after1-before1)+" nanoseconds");  
  
 long before2 = System.*nanoTime*();  
 System.*out*.println(Arrays.*toString*(*insertionSort1D*(array)));  
 long after2 = System.*nanoTime*();  
 System.*out*.println("Execution time of insertionSort() method is "+(after2-before2)+" nanoseconds");  
  
 long before3 = System.*nanoTime*();  
 System.*out*.println(Arrays.*toString*(*selectionSort1D*(array)));  
 long after3 = System.*nanoTime*();  
 System.*out*.println("Execution time of selectionSort() method is "+(after3-before3)+" nanoseconds");  
  
 long before4 = System.*nanoTime*();  
 *quickSort1D*(array, 0, array.length-1);  
 System.*out*.println(Arrays.*toString*(array));  
 long after4 = System.*nanoTime*();  
 System.*out*.println("Execution time of quickSort() method is "+(after4-before4)+" nanoseconds");  
 }  
}

**Output:**

****

**Task#02**

Question statement

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

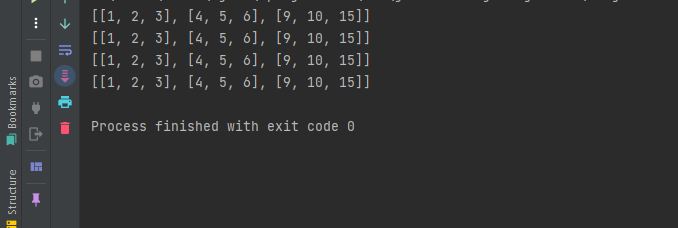
|  |  |  |
| --- | --- | --- |
| **2** | **10** | **15** |
| **5** | **1** | **3** |
| **6** | **9** | **4** |

# SortingAlgosFor2DArray.Java

**Code:**

import java.util.Arrays;  
import java.util.ArrayList;  
  
public class SortingAlgosFor2DArray {  
  
 public static int[][] insertionSort2D(int [][] matrix){  
 int [] arr = *conversionIn1D*(matrix);  
 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 *conversionIn2D*(arr, matrix.length, matrix[0].length);  
 } // end of insertionSort() method  
  
 public static int[][] selectionSort2D(int [][] matrix){  
 int [] arr = *conversionIn1D*(matrix);  
 for (int i=0; i< arr.length-1; i++){  
 int min = i;  
 for (int j=i+1; j< arr.length; j++){  
 if (arr[j]<arr[min]){  
 min = j;  
 }  
 }  
 if (i!=min){  
 int temp = arr[i];  
 arr[i] = arr[min];  
 arr[min] = temp;  
 }  
 }  
 return *conversionIn2D*(arr, matrix.length, matrix[0].length);  
 } // end of selectionSort() method  
  
 public static int[] conversionIn1D(int [][] matrix){  
 ArrayList<Integer> arrayList = new ArrayList<>();  
 int k=0;  
 for (int i=0; i< matrix.length; i++) {  
 for (int j=0; j< matrix[i].length; j++) {  
 arrayList.add(k, matrix[i][j]);  
 k++;  
 }  
 }  
 int [] array = new int[arrayList.size()];  
 for (int i=0; i< array.length; i++) {  
 array[i] = arrayList.get(i);  
 }  
 return array;  
 } // end of conversionIn1D() method  
  
 public static int[][] conversionIn2D(int [] array, int rows, int cols){  
 int[][] matrix = new int[rows][cols];  
 int k = 0;  
 for (int i=0; i< rows; i++){  
 for (int j=0; j< cols; j++){  
 matrix[i][j] = array[k];  
 k++;  
 }  
 }  
 return matrix;  
 } // end of conversionIn2D() method  
 public static int[][] bubbleSort2D(int [][] matrix){  
 int [] array = *conversionIn1D*(matrix);  
 for (int i=0; i<array.length; i++){  
 for (int j=i+1; j<array.length; j++){  
 if (array[j]<array[i]){  
 int temp = array[i];  
 array[i] = array[j];  
 array[j] = temp;  
 }  
 }  
 }  
 return *conversionIn2D*(array, matrix.length, matrix[0].length);  
 }  
  
 public static int partition2D(int [] a, int lowerbound, int upperbound){  
 int pivot = a[upperbound];  
 int i = lowerbound-1;  
 for (int j=lowerbound; j<upperbound; j++){  
 if(a[j]<=pivot){  
 i++;  
 int temp = a[i];  
 a[i] = a[j];  
 a[j] = temp;  
 }  
 }  
 int temp = a[i+1];  
 a[i+1] = a[upperbound];  
 a[upperbound] = temp;  
 return (i+1);  
 }  
  
 public static void quickSort2D(int [] a, int lowerbound, int upperbound){  
 if(lowerbound<upperbound){  
 int location = *partition2D*(a, lowerbound, upperbound);  
 *quickSort2D*(a, lowerbound, location-1);  
 *quickSort2D*(a, location+1, upperbound);  
 }  
 }  
  
  
 public static void main(String[] args) {  
  
 int [][] matrix = {{2, 10, 15}, {5, 1, 3}, {6, 9, 4}};  
 System.*out*.println(Arrays.*deepToString*(*bubbleSort2D*(matrix)));  
 System.*out*.println(Arrays.*deepToString*(*selectionSort2D*(matrix)));  
 System.*out*.println(Arrays.*deepToString*(*insertionSort2D*(matrix)));  
  
 int [] array = *conversionIn1D*(matrix);  
 *quickSort2D*(array, 0, array.length-1);  
 matrix = *conversionIn2D*(array, matrix.length, matrix[0].length);  
 System.*out*.println(Arrays.*deepToString*(matrix));  
  
 } // end of main() method  
} // end of program

**Output:**

****