Assignment 2: Assignment based on some simple coding problems on numbers, graphs, matrices

Name: Adnan Sadar

Roll No: 2

Batch:3

Problem Statement: Assignment based on analysis of quick sort (deterministic and randomized variant)

Code:

import java.util.\*;

class QuickSort {

public static int N = 5;

public static int[] arr = new int[N];

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

Random random = new Random();

System.out.println("Enter n: ");

int n = sc.nextInt();

int arr[] = new int[n];

for (int i = 0; i < n; i++) {

arr[i] = random.nextInt(100000);

}

long[] time = new long[10];

QuickSort ob = new QuickSort();

// Randomised Quick Sort

System.out.println("Randomised Sort:");

for (int i = 0; i < 10; i++) {

long start = System.nanoTime();

ob.randomisedSort(arr, 0, n - 1);

long end = System.nanoTime();

time[i] = end - start;

}

int maxrandomised = 0, min = 0, avgrandomised = 0;

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

System.out.println("Test Case " + (i + 1) + ": " + time[i] + "ns");

if (time[i] < time[min]) {

min = i;

}

if (time[i] > time[maxrandomised]) {

maxrandomised = i;

}

avgrandomised += time[i];

}

avgrandomised = avgrandomised / 10;

System.out.println("\nRandomised Sort:");

System.out.println("\nBest Case: " + time[min] + "ns");

System.out.println("Worst Case: " + time[maxrandomised] + "ns");

System.out.println("Average Case: " + avgrandomised + "ns\n");

// Deterministic Quick Sort

for (int i = 0; i < 10; i++) {

arr[i] = random.nextInt(100000);

long start = System.nanoTime();

ob.deterministicSort(arr, 0, n - 1);

// System.out.println("sorted array");

// printArray(arr);

long end = System.nanoTime();

time[i] = end - start;

}

int maxdeterministic = 0;

min = 0;

int avgdeterministic = (int) time[0];

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

System.out.println("Test Case " + (i + 1) + ": " + time[i] + "ns");

if (time[i] < time[min]) {

min = i;

}

if (time[i] > time[maxdeterministic]) {

maxdeterministic = i;

}

avgdeterministic += time[i];

}

avgdeterministic = avgdeterministic / 10;

System.out.println("\nDeterministic Sort:");

System.out.println("\nBest Case: " + time[min] + "ns");

System.out.println("Worst Case: " + time[maxdeterministic] + "ns");

System.out.println("Average Case: " + avgdeterministic + "ns");

if (avgdeterministic < avgrandomised) {

System.out.println("Conclusion: Deterministic Quick Sort is faster than Randomised Quick Sort");

} else {

System.out.println("Conclusion: Randomised Quick Sort is faster than Deterministic Quick Sort");

}

sc.close();

}

int randomisedPartition(int arr[], int low, int high) {

Random rand = new Random();

int pivot = rand.nextInt(high - low) + low;

int i = (low - 1); // index of smaller element

for (int j = low; j < high; j++) {

if (arr[j] <= pivot) {

i++;

// swap arr[i] and arr[j]

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

int temp = arr[i + 1];

arr[i + 1] = arr[high];

arr[high] = temp;

return i + 1;

}

void randomisedSort(int arr[], int low, int high) {

if (low < high) {

int pi = randomisedPartition(arr, low, high);

randomisedSort(arr, low, pi - 1);

randomisedSort(arr, pi + 1, high);

}

}

int deterministicPartition(int arr[], int low, int high) {

int pivot = arr[high];

int i = (low - 1); // index of smaller element

for (int j = low; j < high; j++) {

// If current element is smaller than the pivot

if (arr[j] < pivot) {

i++;

// swap arr[i] and arr[j]

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

// swap arr[i+1] and arr[high] (or pivot)

int temp = arr[i + 1];

arr[i + 1] = arr[high];

arr[high] = temp;

return i + 1;

}

void deterministicSort(int arr[], int low, int high) {

if (low < high) {

int pi = deterministicPartition(arr, low, high);

deterministicSort(arr, low, pi - 1);

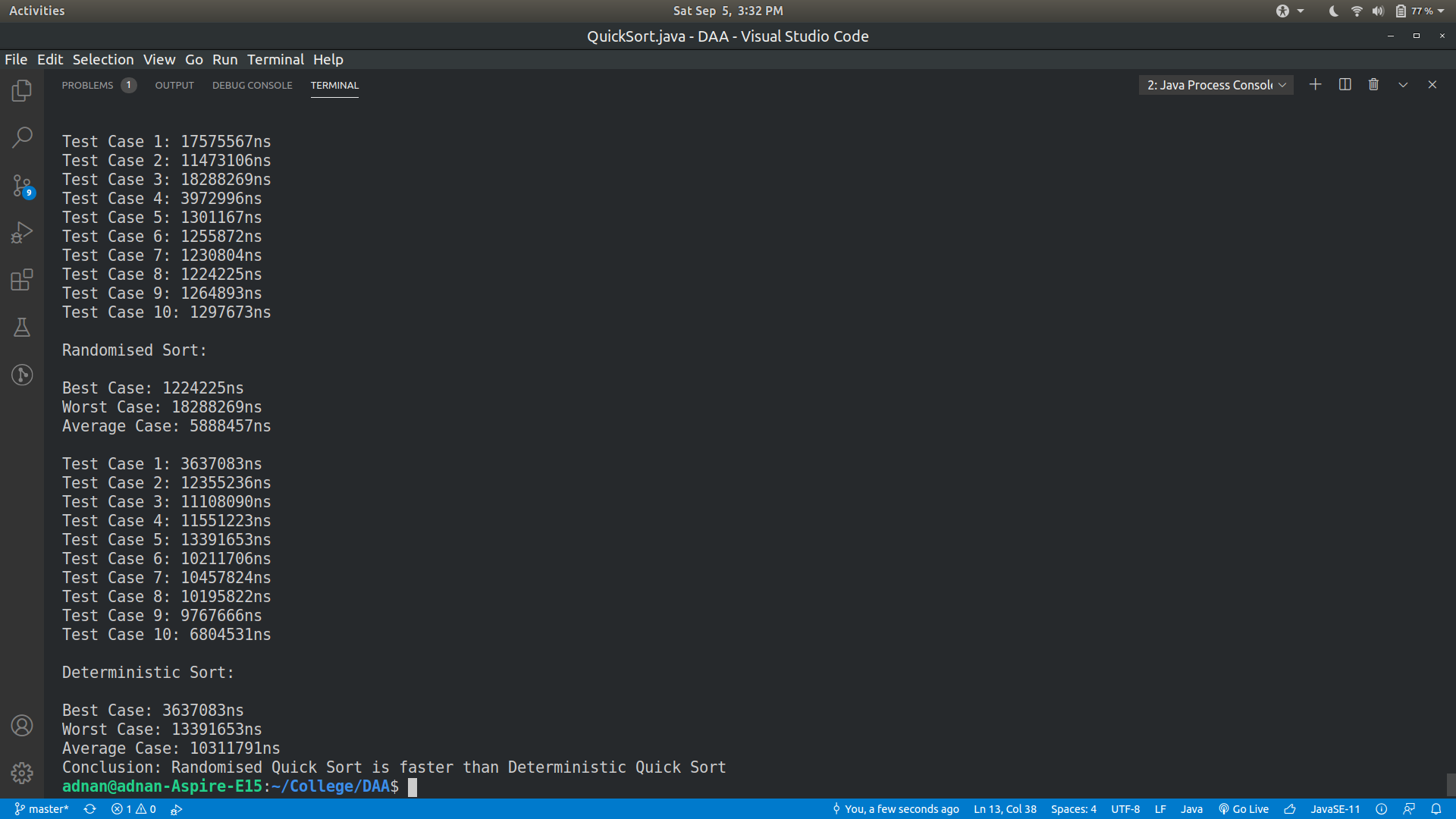
deterministicSort(arr, pi + 1, high);

}

}

}

Output:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N=5 | Deterministic Version | | Randomized Version | |
|  | **Best Case** | 0.003637083 seconds | **Best Case** | 0.001224225 seconds |
|  | **Worst Case** | 0.013391653 seconds | **Worst Case** | 0.018288269seconds |
|  | **Average Case** |  | **Average Case** |  |
| 1 |  | 0.003637083 seconds |  | 0.017575567 seconds |
| 2 |  | 0.012355236 seconds |  | 0.011473106 seconds |
| 3 |  | 0.011108090 seconds |  | 0.018288269 seconds |
| 4 |  | 0.011551223 seconds |  | 0.003972996 seconds |
| 5 |  | 0.013391653 seconds |  | 0.001301167 seconds |
| 6 |  | 0.010211706 seconds |  | 0.001255872 seconds |
| 7 |  | 0.010457824 seconds |  | 0.001230804 seconds |
| 8 |  | 0.010195822 seconds |  | 0.001224225 seconds |
| 9 |  | 0.009767666 seconds |  | 0.001264893 seconds |
| 10 |  | 0.006804531 seconds |  | 0.001297673 seconds |

Conclusion: Randomised Quick Sort performs better as compared to deterministic Quick Sort which is seen when the time complexities of both the approaches are compared side by side.