Analysis Of Algorithms

Name: Wajahat Ali Khan **Sap I'd:** 55431

Section: BSCS 4-1 **Instructor:** Usman Sharif

Assignment: 04

1. Implementation

Bubble Sorting in Java for arr1[5], arr2[10], arr3[50] and arr4[100]

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package bubblesorting;
import java.util.Arrays;
public class BubbleSorting {
    public static void bubbleSort(int[] arr) {
        int n = arr.length;
        for (int i = 0; i < n-1; i++) {
            for (int j = 0; j < n-i-1; j++) {
                if (arr[j] > arr[j+1]) {
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
```

```
        }
    }
}
public static long measureTime(int[] array) {
    int[] copy = Arrays.copyOf(array, array.length);
    long start = System.nanoTime();
    bubbleSort(copy);
    long end = System.nanoTime();
    return end - start;
}
public static void main(String[] args) {
    int[] arr1 = {1, 2, 3, 4, 5};
    int[] arr2 = {1,2,3,4,5,6,7,8,9,10};
    int[] arr3 = new int[50];
    int[] arr4 = new int[100];
    for (int i = 0; i < 50; i++) arr3[i] = i+1;
    for (int i = 0; i < 100; i++) arr4[i] = i+1;

    int[][] arrays = {arr1, arr2, arr3, arr4};
    String[] names = {"Arr1 (5)", "Arr2 (10)", "Arr3 (50)", "Arr4 (100)"};

    System.out.printf("%-15s%-20s%n", "Array Size", "Avg Time (ms)");
    System.out.println("-----------------------------------");

    for (int i = 0; i < arrays.length; i++) {
        long total = 0;
```

```
        for (int j = 0; j < 5; j++) {
            total += measureTime(arrays[i]);
        }
        double avg = total / 5.0 / 1_000_000;
        System.out.printf("%-15s%-20.5f%n", names[i], avg);
    }
  }
}
```

Out Put

```
run:
Array Size      Avg Time (ms)
---------------------------------------
Arr1 (5)        0.00140
Arr2 (10)       0.00412
Arr3 (50)       0.04002
Arr4 (100)      0.15584
BUILD SUCCESSFUL (total time: 0 seconds)
```

Selection Sorting in Java for arr1[5], arr2[10], arr3[50] and arr4[100]

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package selectionsortings;
import java.util.Arrays;
public class SelectionSortings {
    public static void selectionSort(int[] arr) {
        int n = arr.length;
```

```
    for (int i = 0; i < n-1; i++) {
        int minIdx = i;
        for (int j = i+1; j < n; j++) {
            if (arr[j] < arr[minIdx]) {
                minIdx = j;
            }
        }
        int temp = arr[minIdx];
        arr[minIdx] = arr[i];
        arr[i] = temp;
    }
}
public static long measureTime(int[] array) {
    int[] copy = Arrays.copyOf(array, array.length);
    long start = System.nanoTime();
    selectionSort(copy);
    long end = System.nanoTime();
    return end - start;
}
public static void main(String[] args) {
    int[] arr1 = {1, 2, 3, 4, 5};
    int[] arr2 = {1,2,3,4,5,6,7,8,9,10};
    int[] arr3 = new int[50];
    int[] arr4 = new int[100];
    for (int i = 0; i < 50; i++) arr3[i] = i+1;
    for (int i = 0; i < 100; i++) arr4[i] = i+1;
```

```
        int[][] arrays = {arr1, arr2, arr3, arr4};
        String[] names = {"Arr1 (5)", "Arr2 (10)", "Arr3 (50)", "Arr4 (100)"};

        System.out.printf("%-15s%-20s%n", "Array Size", "Avg Time (ms)");
        System.out.println("-------------------------------------");

        for (int i = 0; i < arrays.length; i++) {
            long total = 0;
            for (int j = 0; j < 5; j++) {
                total += measureTime(arrays[i]);
            }
            double avg = total / 5.0 / 1_000_000;
            System.out.printf("%-15s%-20.5f%n", names[i], avg);
        }
    }
}
```

Out Put

```
run:
Array Size     Avg Time (ms)
-------------------------------------
Arr1 (5)       0.00110
Arr2 (10)      0.00256
Arr3 (50)      0.04842
Arr4 (100)     0.18048
BUILD SUCCESSFUL (total time: 0 seconds)
```

Insertion Sorting in Java for arr1[5], arr2[10], arr3[50] and arr4[100]

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package insertionsorting;
import java.util.Arrays;
public class InsertionSorting {
    public static void insertionSort(int[] arr) {
        int n = arr.length;
        for (int i = 1; i < n; i++) {
            int 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 long measureTime(int[] array) {
        int[] copy = Arrays.copyOf(array, array.length);
        long start = System.nanoTime();
        insertionSort(copy);
        long end = System.nanoTime();
```

```
        return end - start;
    }
    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3, 4, 5};
        int[] arr2 = {1,2,3,4,5,6,7,8,9,10};
        int[] arr3 = new int[50];
        int[] arr4 = new int[100];
        for (int i = 0; i < 50; i++) arr3[i] = i+1;
        for (int i = 0; i < 100; i++) arr4[i] = i+1;

        int[][] arrays = {arr1, arr2, arr3, arr4};
        String[] names = {"Arr1 (5)", "Arr2 (10)", "Arr3 (50)", "Arr4 (100)"};

        System.out.printf("%-15s%-20s%n", "Array Size", "Avg Time (ms)");
        System.out.println("-------------------------------------");

        for (int i = 0; i < arrays.length; i++) {
            long total = 0;
            for (int j = 0; j < 5; j++) {
                total += measureTime(arrays[i]);
            }
            double avg = total / 5.0 / 1_000_000;
            System.out.printf("%-15s%-20.5f%n", names[i], avg);
        }
    }
}
```

Out Put

```
run:
Array Size      Avg Time (ms)
-------------------------------------
Arr1 (5)        0.00124
Arr2 (10)       0.00274
Arr3 (50)       0.00570
Arr4 (100)      0.01426
BUILD SUCCESSFUL (total time: 0 seconds)
```

Merge Sorting in Java for arr1[5], arr2[10], arr3[50] and arr4[100]

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package mergesorting;
import java.util.Arrays;
public class MergeSorting {
    public static void mergeSort(int[] arr, int left, int right) {
        if (left < right) {
            int mid = left + (right - left)/2;
            mergeSort(arr, left, mid);
            mergeSort(arr, mid+1, right);
            merge(arr, left, mid, right);
        }
    }
    private static void merge(int[] arr, int left, int mid, int right) {
        int n1 = mid - left + 1;
        int n2 = right - mid;
```

```
int[] L = new int[n1];
int[] R = new int[n2];

for (int i=0; i<n1; ++i)
    L[i] = arr[left + i];
for (int j=0; j<n2; ++j)
    R[j] = arr[mid + 1 + j];

int i = 0, j = 0;
int k = left;
while (i < n1 && j < n2) {
    if (L[i] <= R[j]) {
        arr[k] = L[i];
        i++;
    } else {
        arr[k] = R[j];
        j++;
    }
    k++;
}
while (i < n1) {
    arr[k] = L[i];
    i++;
    k++;
}
```

```
        while (j < n2) {
            arr[k] = R[j];
            j++;
            k++;
        }
    }
    public static long measureTime(int[] array) {
        int[] copy = Arrays.copyOf(array, array.length);
        long start = System.nanoTime();
        mergeSort(copy, 0, copy.length - 1);
        long end = System.nanoTime();
        return end - start;
    }
    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3, 4, 5};
        int[] arr2 = {1,2,3,4,5,6,7,8,9,10};
        int[] arr3 = new int[50];
        int[] arr4 = new int[100];
        for (int i = 0; i < 50; i++) arr3[i] = i+1;
        for (int i = 0; i < 100; i++) arr4[i] = i+1;

        int[][] arrays = {arr1, arr2, arr3, arr4};
        String[] names = {"Arr1 (5)", "Arr2 (10)", "Arr3 (50)", "Arr4 (100)"};

        System.out.printf("%-15s%-20s%n", "Array Size", "Avg Time (ms)");
        System.out.println("-----------------------------------");
```

```
        for (int i = 0; i < arrays.length; i++) {
            long total = 0;
            for (int j = 0; j < 5; j++) {
                total += measureTime(arrays[i]);
            }
            double avg = total / 5.0 / 1_000_000;
            System.out.printf("%-15s%-20.5f%n", names[i], avg);
        }
    }
}
```

Out Put

```
run:
Array Size      Avg Time (ms)
---------------------------------------
Arr1 (5)        0.00334
Arr2 (10)       0.00588
Arr3 (50)       0.05044
Arr4 (100)      0.07732
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Data Collection and Analysis

Sorting-type	Input Size	Average Execution Time (m s)
Bubble	5	0.00140
Bubble	10	0.00412
Bubble	50	0.04002
Bubble	100	0.15584
Selection	5	0.00110
Selection	10	0.00256
Selection	50	0.04842
Selection	100	0.18048
Insertion	5	0.00124

Insertion	10	0.00274
Insertion	50	0.00570
Insertion	100	0.01426
Merge	5	0.00334
Merge	10	0.00588
Merge	50	0.05044
Merge	100	0.07732

Graph