Hacettepe University

Computer Engineering Department

BM204 Software Practicum II - 2022 Spring

Programming Assignment 1

March 8, 2022

Student name: Alihan
Name Surname : Sağöz

Student Number: 21993035
b21993035

1 Problem Definition

Briefly state the problem that you are trying to solve. Add any background information if necessary.

2 Solution Implementation

Your answers, explanations, code go into this section.

2.1 Sorting Algorithm 1

Example how to add Java code:

```java
public static void insertionSort(int array[]) {
        int n = array.length;
        for (int j = 1; j < n; j++) {
            int key = array[j];
            int i = j-1;
            while ( (i > -1) && ( array [i] > key ) ) {
                array [i+1] = array [i];
                i--;
            }
            array[i+1] = key;
        }
    }
```

And you can reference line ?? in the code like this.

2.2 Sorting Algorithm 2

Example how to add Java code:

```java
    public static void mergeSort(int[] a, int n) {
        if (n < 2) {
            return;
        }
        int mid = n / 2;
        int[] l = new int[mid];
        int[] r = new int[n - mid];

        for (int i = 0; i < mid; i++) {
            l[i] = a[i];
        }
        for (int i = mid; i < n; i++) {
            r[i - mid] = a[i];
        }
        mergeSort(l, mid);
        mergeSort(r, n - mid);

        merge(a, l, r, mid, n - mid);
    }

    public static void merge(
            int[] a, int[] l, int[] r, int left, int right) {

        int i = 0, j = 0, k = 0;
        while (i < left && j < right) {
            if (l[i] <= r[j]) {
                a[k++] = l[i++];
            }
            else {
                a[k++] = r[j++];
            }
        }
        while (i < left) {
            a[k++] = l[i++];
        }
        while (j < right) {
            a[k++] = r[j++];
        }
    }
```

And you can reference line ?? in the code like this.

2.3 Sorting Algorithm 3

Example how to add Java code:

```
public static void pigeonhole_sort(int arr[],
                                       int n)
    {
        int min = arr[0];
        int max = arr[0];
        int range, i, j, index;

        for (int a = 0; a < n; a++) {
            if (arr[a] > max)
                max = arr[a];
            if (arr[a] < min)
                min = arr[a];
        }

        range = max - min + 1;
        int[] phole = new int[range];
        Arrays.fill(phole, 0);

        for (i = 0; i < n; i++)
```

```java
            phole[arr[i] - min]++;

        index = 0;

        for (j = 0; j < range; j++)
            while (phole[j]-- > 0)
                arr[index++] = j + min;
    }
```

2.4 Sorting Algorithm 4

Example how to add Java code:

```java
public static void countSort(int[] arr)
    {
        int max = Arrays.stream(arr).max().getAsInt();
        int min = Arrays.stream(arr).min().getAsInt();
        int range = max - min + 1;
        int count[] = new int[range];
        int output[] = new int[arr.length];
        for (int i = 0; i < arr.length; i++) {
            count[arr[i] - min]++;
        }

        for (int i = 1; i < count.length; i++) {
            count[i] += count[i - 1];
        }

        for (int i = arr.length - 1; i >= 0; i--) {
            output[count[arr[i] - min] - 1] = arr[i];
            count[arr[i] - min]--;
        }

        for (int i = 0; i < arr.length; i++) {
            arr[i] = output[i];
        }
    }
```

3 Results, Analysis, Discussion

Your explanations, results, plots go in this section...

Running time test results are given in Table 3. Copy-paste the table to add two more for the results on sorted and reversely sorted data test. Don't forget to change the captions.

Table 1: Results of the running time tests performed on the random data of varying sizes (in ms).

Algorithm	Input Size									
	512	1024	2048	4096	8192	16384	32768	65536	131072	251281
Insertion sort	0.9	0.1	7.8	1.3	4.8	27	83	354	1410	6091
Merge sort	0.7	1.1	2	4.1	10	25	30	28.7	59	124
Pigeonhole sort	0.02	0.05	0.12	0.23	0.41	0.43	0.12	0.61	3.30	181.90
Counting sort	0.21	0.12	0.23	0.28	0.55	0.47	0.70	1.06	3.24	126.25

Results of the running time tests performed on the sorted data of varying sizes (in ms).

Algorithm	Input Size									
	512	1024	2048	4096	8192	16384	32768	65536	131072	251281
Insertion sort	0.001	0.002	0.005	0.007	0.012	0.042	0.090	0.120	0.384	0.452
Merge sort	0.020	0.040	0.078	0.291	0.718	1.500	1.914	4.632	7.691	11.886
Pigeonhole sort	0.001	0.003	0.005	0.011	0.024	0.047	0.093	0.178	1.095	184.838
Counting sort	0.025	0.022	0.028	0.055	0.066	0.11	0.17	0.49	1.08	152.20

Results of the running time tests performed on the reversed data of varying sizes (in ms).

Algorithm	Input Size									
	512	1024	2048	4096	8192	16384	32768	65536	131072	251281
Insertion sort	0.007	0.028	0.124	0.50	1.84	7.67	24.40	71.27	362.94	1142.37
Merge sort	0.012	0.026	0.050	0.109	0.231	0.475	0.942	2.139	4.373	9.557
Pigeonhole sort	0.001	0.002	0.005	0.011	0.022	0.044	0.089	0.179	0.978	183.82
Counting sort	0.004	0.006	0.012	0.024	0.047	0.094	0.176	0.327	1.34	133.955

Complexity analysis tables to complete:

Table 2: Computational complexity comparison of the given algorithms.

Algorithm	**Best Case**	**Average Case**	**Worst Case**
Insertion Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$
Merge Sort	$\Omega(n \log n)$	$\Theta(n \log n)$	$O(n \log n)$
Pigeonhole Sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n+k)$
Counting Sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n+k)$

Table 3: Auxiliary space complexity of the given algorithms.

Algorithm	**Auxiliary Space Complexity**
Insertion Sort	$O(1)$
Merge Sort	$O(n)$
Pigeonhole Sort	$O(n+r)$ r = range
Counting Sort	$O(n+k)$ k= max

Example how to add a formula: $F(A, B, C, D) = \sum(0, 1, 2, 3, 10)$
Example how to include and reference a figure: Fig. 3.

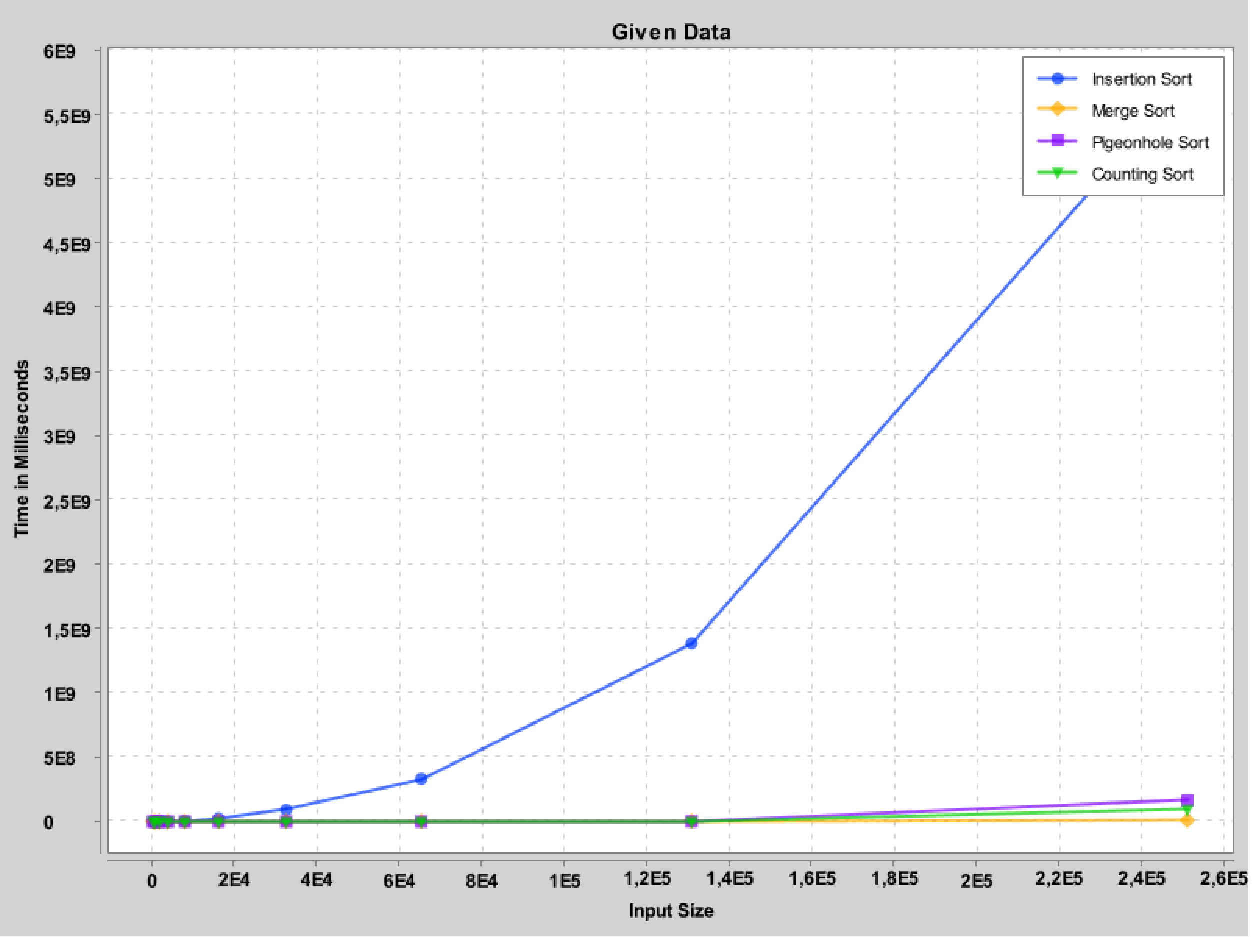

Figure 1: Plot of the functions.

Results analysis, explanations...

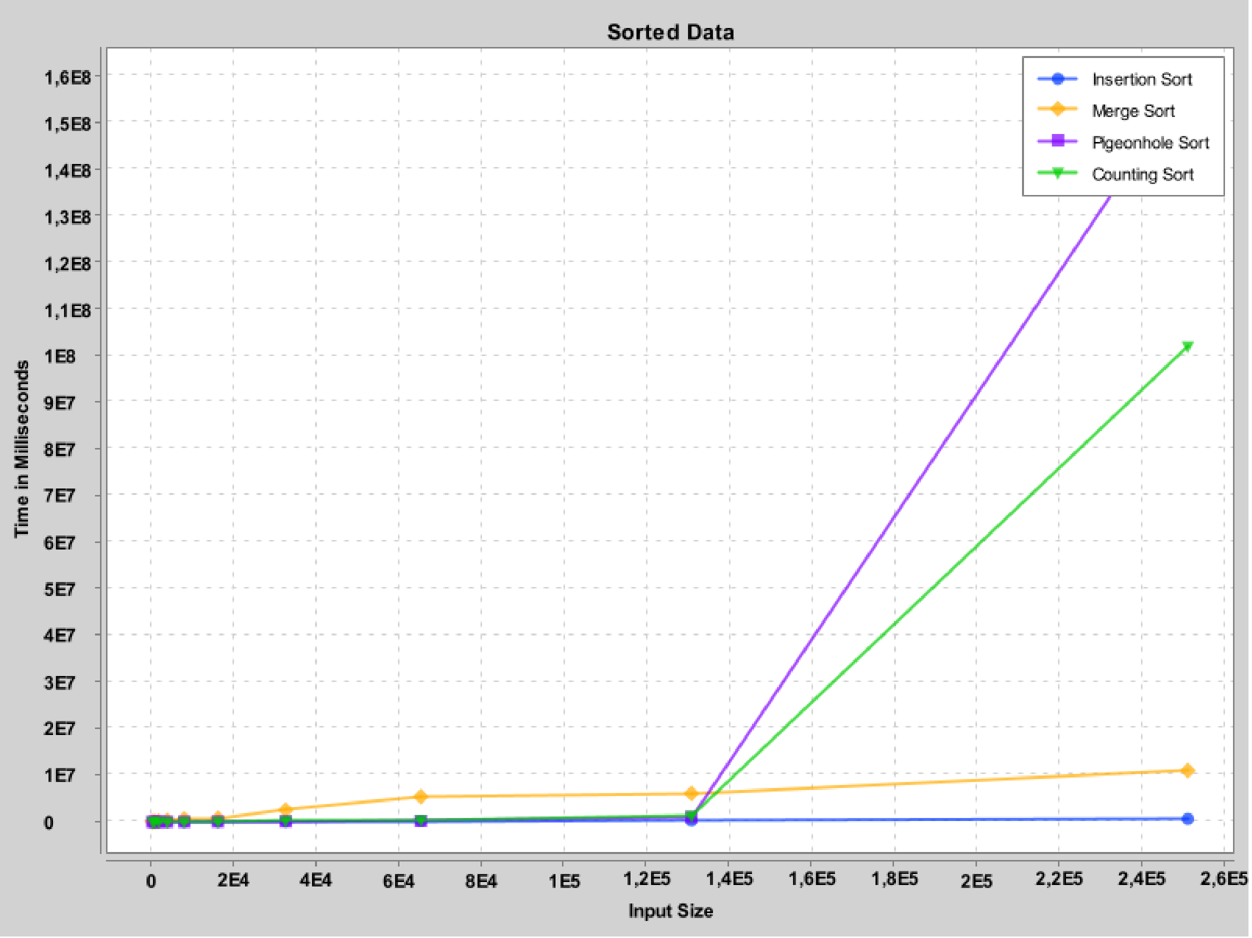

Figure 2: Plot of the functions.

4 Notes

Here you can add your notes if any.

References

- Reference 1...
- Reference 2...

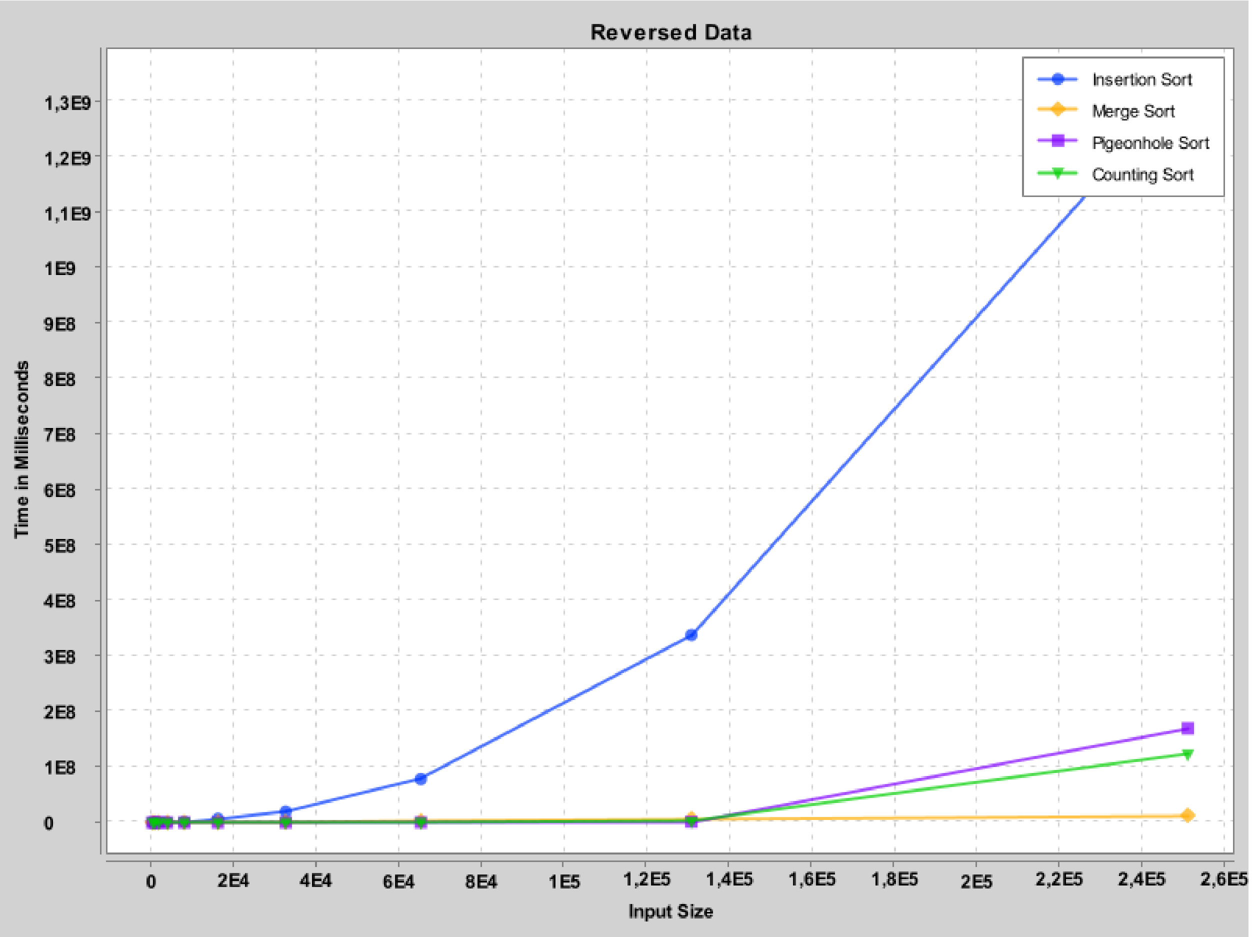

Figure 3: Plot of the functions.