Hacettepe University

Computer Engineering Department

BM204 Software Practicum II - 2022 Spring

Programming Assignment 1

March 5, 2022

Student name:
Osman Faruk Derdiyok

Student Number:
b21946036

1 Problem Definition

In the dataset given to us, we need to put the 8th column into various sorting algorithms. It can be briefly explained as follows: Using 3 algorithms, running the data 10 times in random, sorted and reverse order, taking the average, showing them on the graph and making the necessary analysis of this graph.

2 Solution Implementation

The algorithms used on the program are explained below

2.1 Insertion Sort

```
1  static void insertionSort(int[] array) {
2      int n = array.length;
3      for (int j = 1; j < n; j++) {
4          int key = array[j];
5          int i = j-1;
6          while ( (i > -1) && ( array [i] > key ) ) {
7              array [i+1] = array [i];
8              i--;
9          }
10         array[i+1] = key;
11     }
12 }
```

The results for random order, ordered, reverse order data are explained in the next section.

2.2 Merge Sort

```
15 static void sort(int[] arr, int l, int r){
16     if (l < r) {
17         int m =l+ (r-l)/2;
18         sort(arr, l, m);
19         sort(arr, m + 1, r);
20         merge(arr, l, m, r);
21     }
22 }
23
24 static void merge(int[] arr, int l, int m, int r){
25     // Find sizes of two subarrays to be merged
26     int n1 = m - l + 1;
27     int n2 = r - m;
28
29     int L[] = new int[n1]; //Create temp arrays
30     int R[] = new int[n2]; //Create temp arrays
```

```
    /*Copy data to temp arrays*/
    for (int i = 0; i < n1; ++i)
        L[i] = arr[l + i];
    for (int j = 0; j < n2; ++j)
        R[j] = arr[m + 1 + j];

    /* Merge the temp arrays */

    // Initial indexes of first and second subarrays
    int i = 0, j = 0;

    // Initial index of merged subarray array
    int k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    /* Copy remaining elements of L[] if any */
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy remaining elements of R[] if any */
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}


//Coded with the geeksforgeeks reference.
//Check the references title for details.
```

The results for random order, ordered, reverse order data are explained in the next section.

2.3 Counting Sort

```
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];
    }
}
```

The results for random order, ordered, reverse order data are explained in the next section.

2.4 Pigeonhole Sort

```
static void pigeonhole_sort(int[] arr){
    int n = arr.length;
    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++)
        phole[arr[i] - min]++;

    index = 0;

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

The results for random order, ordered, reverse order data are explained in the next section.

3 Results, Analysis, Discussion

Complexity analysis tables:

Table 1: 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	$\Omega(n + 2^k)$	$\Theta(n + 2^k)$	$O(n + 2^k)$
Counting Sort	$\Omega(n + k)$	$\Theta(n + k)$	$O(n + k)$

Table 2: Auxiliary space complexity of the given algorithms.

Algorithm	**Auxiliary Space Complexity**
Insertion Sort	$O(1)$
Merge Sort	$O(n)$
Pigeonhole Sort	$O(2^k)$
Counting Sort	$O(k)$

Running time test results are given in Table 3, Table 4, Table 5.

Table 3: 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	0	0	1	4	19	69	269	1116	4612
Merge sort	0	0	0	0	0	1	2	5	11	28
Pigeonhole sort	194	178	214	179	240	223	210	180	218	181
Counting sort	200	156	138	115	144	150	171	116	133	132

The graph of the random data set is given below.

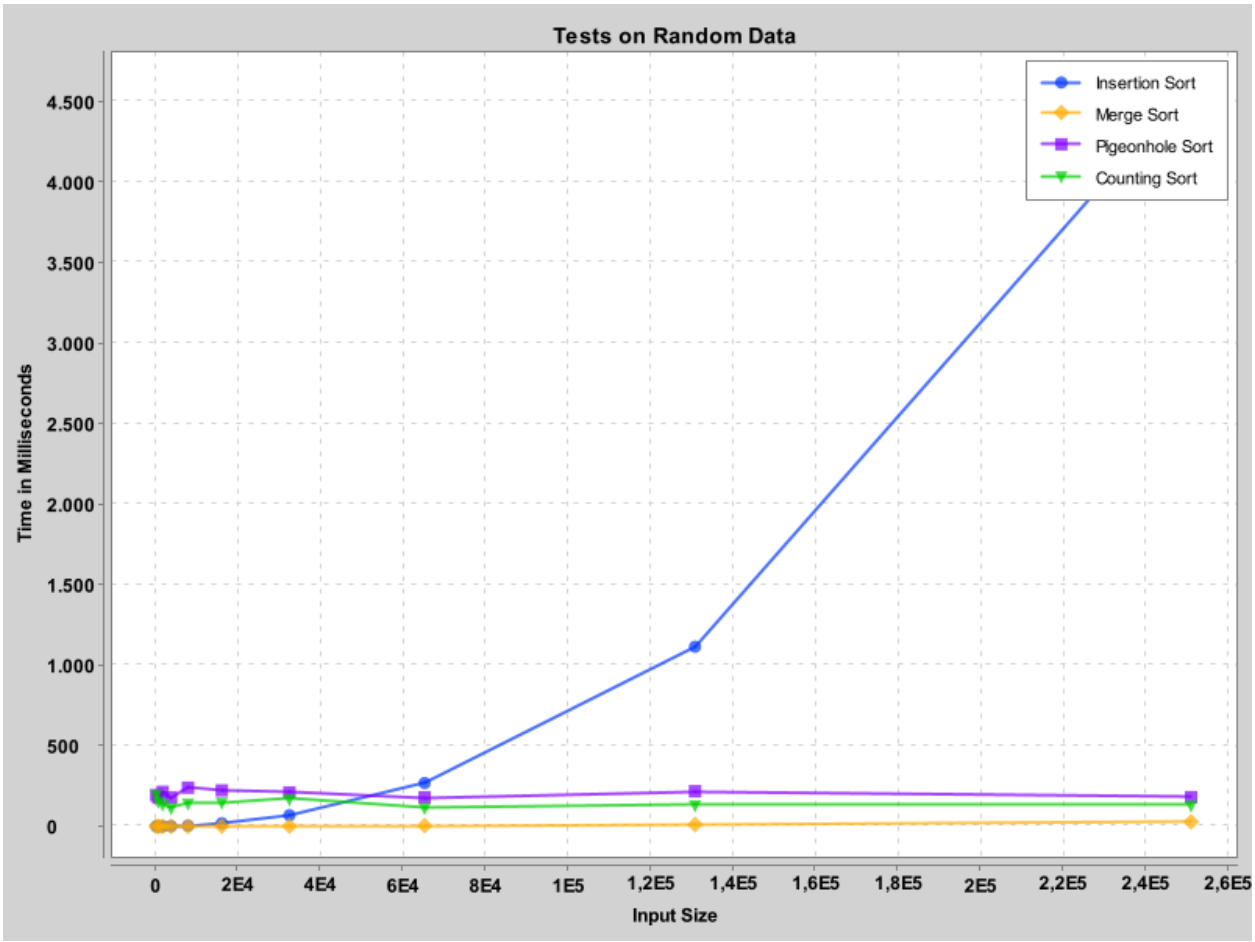

Figure 1: Plot of the functions for random dataset.

Table 4: 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	0	0	0	0	0	0	0	0	0
Merge sort	0	0	0	0	0	0	1	5	4	10
Pigeonhole sort	217	178	181	184	245	222	214	177	228	181
Counting sort	139	140	128	111	139	127	143	114	132	126

The graph about the sorted data set.

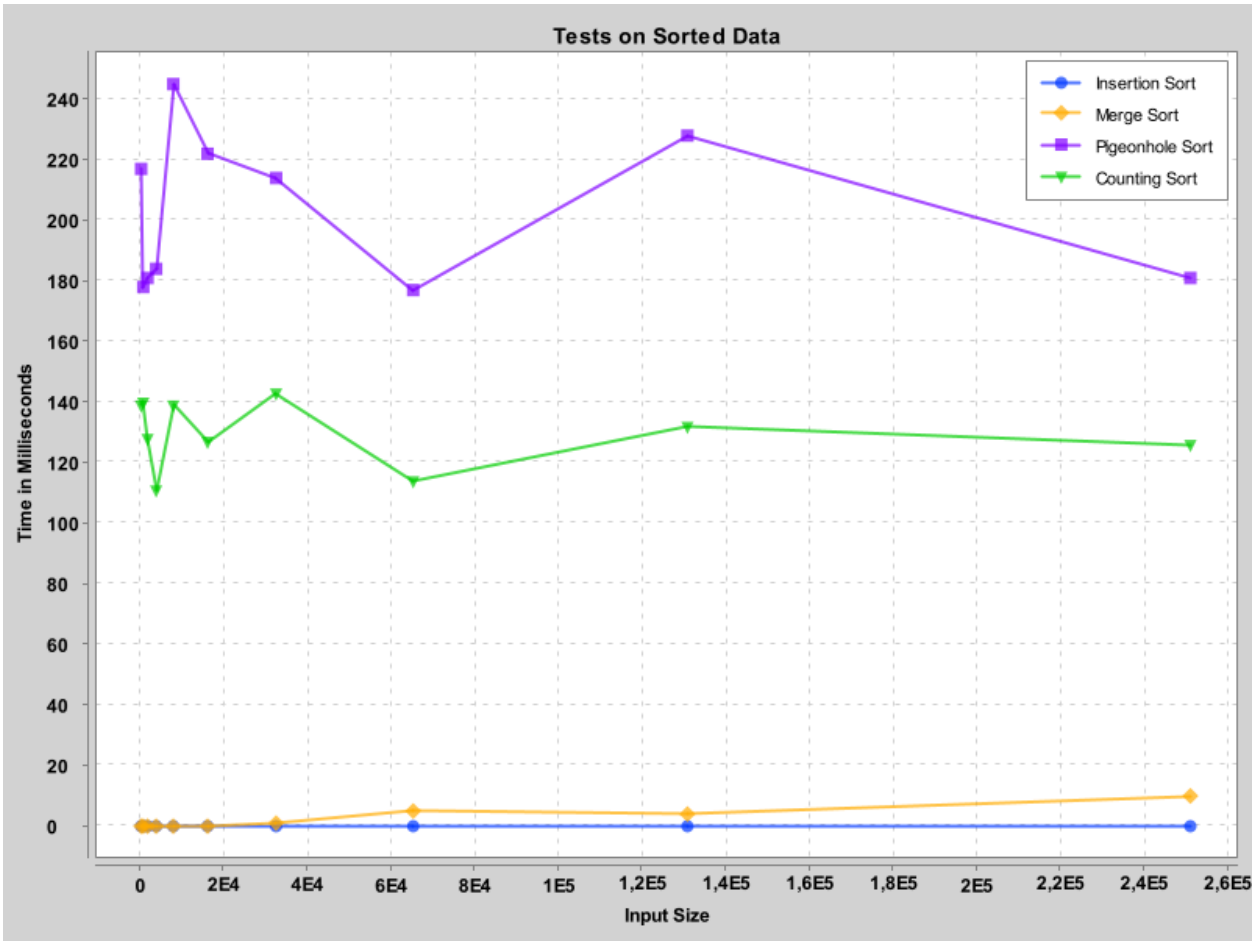

Figure 2: Plot of the functions for random dataset.

Table 5: 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	0	0	2	8	34	136	548	2219	8204
Merge sort	0	0	0	0	0	0	1	5	9	11
Pigeonhole sort	238	191	178	236	259	245	224	192	240	184
Counting sort	133	114	122	112	144	121	123	113	120	120

The graph about the reversed data set.

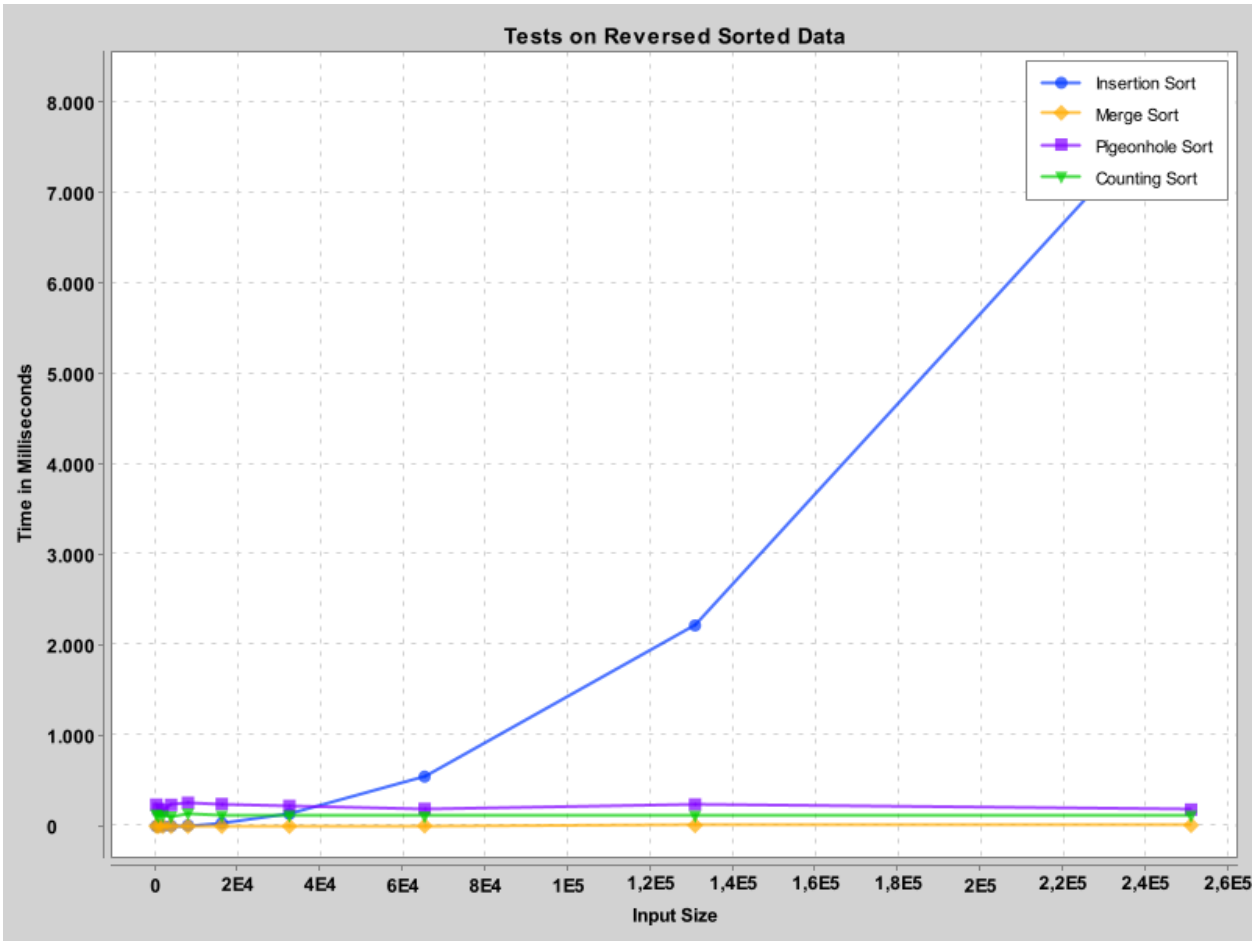

Figure 3: Plot of the functions for reversed dataset.

- What are the best, average, and worst cases for the given algorithms in terms of the given input data to be sorted?
  - For the insertion sort: obviously sorted data best case. worst case is reversed data.But, for other sorting algorithms(merge, pigeonhole, counting), as we see in table 1, this does not affect it. They have been working recently for all their data.

- Do the obtained results (running times of your algorithm implementations) match their theoretical asymptotic complexities?
  - No. The results were not as theoretically as they should be in any output. There are many issues affecting this. Instant status of the computer, temperature, open applications in the background, etc. But in some cases, the results did not increase even though they should have increased :O. (this is said to be caused by 'cache', in detail: `https://piazza.com/class/kyzqhsc5lck76h?cid=25_f13`)

4 Notes

- To reverse the sorted data, reverseArray() algorithm is used.

```
124         static void reverseArray(int[] intArray, int size){
125             int i, k, temp;
126             for (i = 0; i < size / 2; i++) {
127                 temp = intArray[i];
128                 intArray[i] = intArray[size - i - 1];
129                 intArray[size - i - 1] = temp;
130             }
131         }
```

- The charts are commented out because they prevent the code from completing. Also, xchart library throws error in cmd and dev.

```
132         /*
133
134         // X axis data
135         int[] inputAxis = {512, 1024, 2048, 4096, 8192, 16384, 32768,
                65536, 131072, 251282};
136
137         // results on random data
138         double[][] yAxis = new double[4][10];
139         yAxis[0] = new double[]{0, 0, 0, 1, 4, 19, 69, 269, 1116, 4612};
140         yAxis[1] = new double[]{0, 0, 0, 0, 0, 1, 2, 5, 11, 28};
141         yAxis[2] = new double[]{194, 178, 214, 179, 240, 223, 210, 180,
                218, 181};
142         yAxis[3] = new double[]{200, 156, 138, 115, 144, 150, 171, 116,
                133, 132};
143         showAndSaveChart("Tests on Random Data", inputAxis, yAxis);
144
```

```
145         // results on sorted data
146         double[][] yAxisSorted = new double[4][10];
147         yAxisSorted[0] = new double[]{0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
148         yAxisSorted[1] = new double[]{0, 0, 0, 0, 0, 0, 1, 5, 4, 10};
149         yAxisSorted[2] = new double[]{217, 178, 181, 184, 245, 222, 214,
                177, 228, 181};
150         yAxisSorted[3] = new double[]{139, 140, 128, 111, 139, 127, 143,
                114, 132, 126};
151         showAndSaveChart("Tests on Sorted Data", inputAxis, yAxisSorted);
152
153
154         // results on reversed data
155         double[][] yAxisReversed = new double[4][10];
156         yAxisReversed[0] = new double[]{0, 0, 0, 2, 8, 34, 136, 548,
                2219, 8204};
157         yAxisReversed[1] = new double[]{0, 0, 0, 0, 0, 0, 1, 5, 9, 11};
158         yAxisReversed[2] = new double[]{238, 191, 178, 236, 259, 245,
                224, 192, 240, 184};
159         yAxisReversed[3] = new double[]{133, 114, 122, 112, 144, 121,
                123, 113, 120, 120};
160         showAndSaveChart("Tests on Reversed Sorted Data", inputAxis,
                yAxisReversed);
161         */
```

References

- ("Array Sorting Algorithms Cheet Sheet" , "https://towardsdatascience.com/understanding-time-complexity-with-python-examples-2bda6e8158a7")
- ("Pigeonhole sorting algorithm" , "https://en.wikipedia.org/wiki/Pigeonhole_sort" , "Last Accessed: 05/03/2022")
- ("https://stackoverflow.com/questions/2137755/how-do-i-reverse-an-int-array-in-java")
- ("https://www.geeksforgeeks.org/system-arraycopy-in-java/")
- ("https://www.geeksforgeeks.org/merge-sort/")
- ("https://www.geeksforgeeks.org/heap-sort/")
- ("Latex", "https://www.overleaf.com/learn/latex/Positioning_of_Figures")