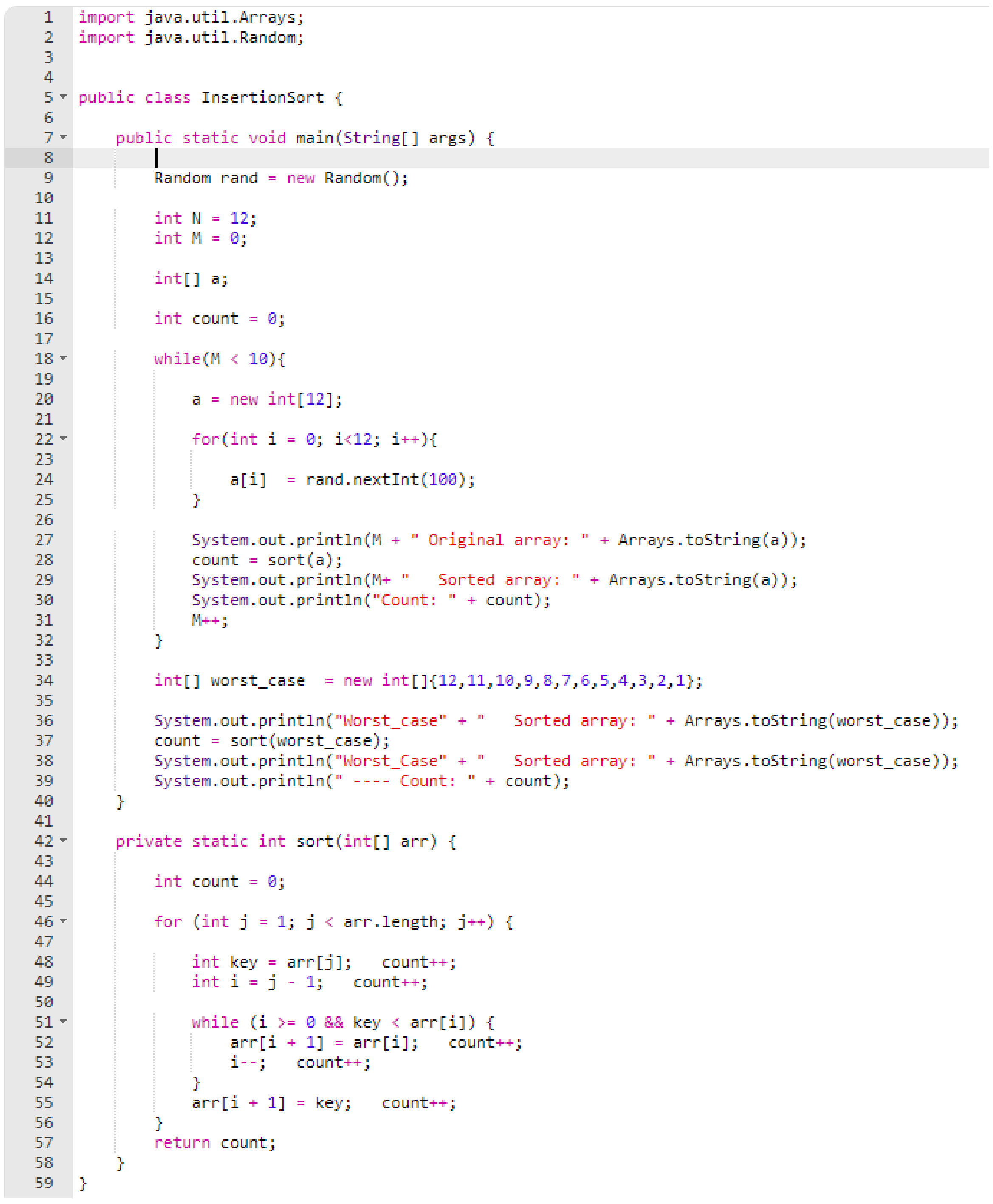
PROJECT 1 （Zebo Xiong | A04907051）

INSERTION SORT, MERGE SORT and ALGORITHM ANALYSIS

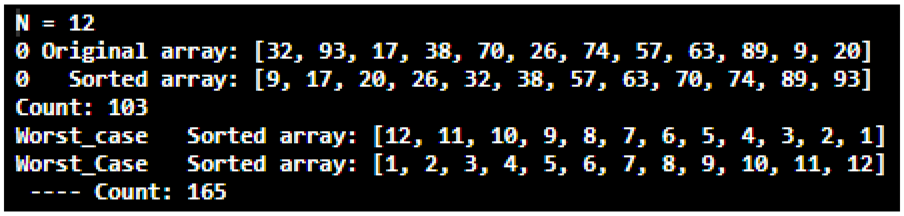
a) Using Insertion Sort algorithm Implement a Program in any language you desire (preferably C++ , java, or Python) to sort an array of Real Numbers of size N.

   1. Input at least 10 or more sets of randomized unsorted data with at least N elements in each set. For example, N= 12, 29, 30, 35, 49, 52, 65, and 76

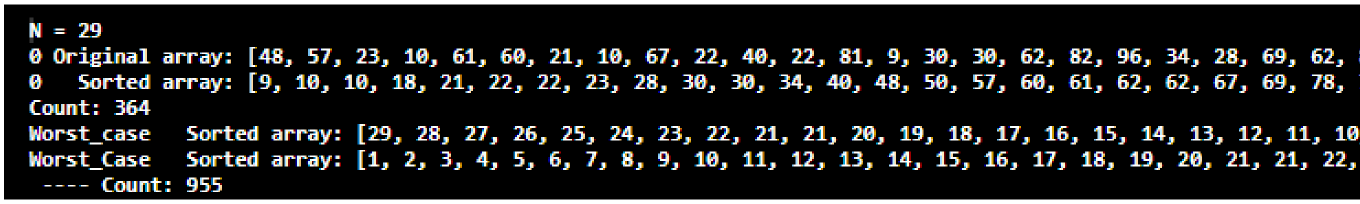


```java
import java.util.Arrays;
import java.util.Random;


public class InsertionSort {

    public static void main(String[] args) {

        Random rand = new Random();

        int N = 12;
        int M = 0;

        int[] a;

        int count = 0;

        while(M < 10){

            a = new int[12];

            for(int i = 0; i<12; i++){

                a[i]  = rand.nextInt(100);
            }

            System.out.println(M + " Original array: " + Arrays.toString(a));
            count = sort(a);
            System.out.println(M+ "   Sorted array: " + Arrays.toString(a));
            System.out.println("Count: " + count);
            M++;
        }

        int[] worst_case  = new int[]{12,11,10,9,8,7,6,5,4,3,2,1};

        System.out.println("Worst_case" + "   Sorted array: " + Arrays.toString(worst_case));
        count = sort(worst_case);
        System.out.println("Worst_Case" + "   Sorted array: " + Arrays.toString(worst_case));
        System.out.println(" ---- Count: " + count);
    }

    private static int sort(int[] arr) {

        int count = 0;

        for (int j = 1; j < arr.length; j++) {

            int key = arr[j];   count++;
            int i = j - 1;   count++;

            while (i >= 0 && key < arr[i]) {
                arr[i + 1] = arr[i];   count++;
                i--;   count++;
            }
            arr[i + 1] = key;   count++;
        }
        return count;
    }
}
```

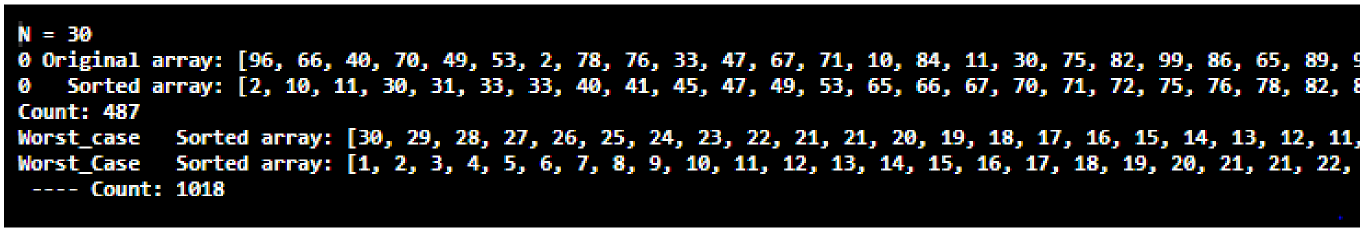
2. Display the Unsorted data input and sorted data output for each array of N elements. Plot a graph to compare the worst case (average case) algorithm and the actual count putting counters in strategic points of your programs by assuming the cost ci = 1 for all statements.



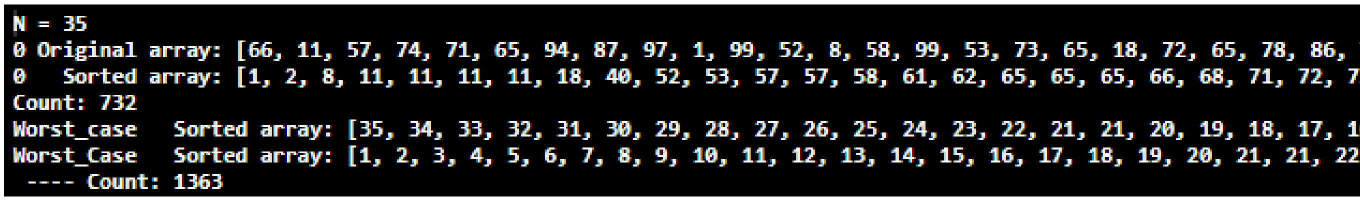
```
N = 12
0 Original array: [32, 93, 17, 38, 70, 26, 74, 57, 63, 89, 9, 20]
0   Sorted array: [9, 17, 20, 26, 32, 38, 57, 63, 70, 74, 89, 93]
Count: 103
Worst_case   Sorted array: [12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
 ---- Count: 165
```



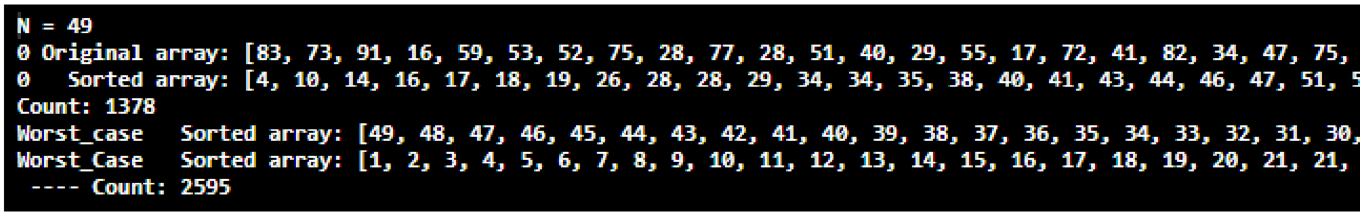
```
N = 29
0 Original array: [48, 57, 23, 10, 61, 60, 21, 10, 67, 22, 40, 22, 81, 9, 30, 30, 62, 82, 96, 34, 28, 69, 62,
0   Sorted array: [9, 10, 10, 18, 21, 22, 22, 23, 28, 30, 30, 34, 40, 48, 50, 57, 60, 61, 62, 62, 67, 69, 78,
Count: 364
Worst_case   Sorted array: [29, 28, 27, 26, 25, 24, 23, 22, 21, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 21, 22,
 ---- Count: 955
```



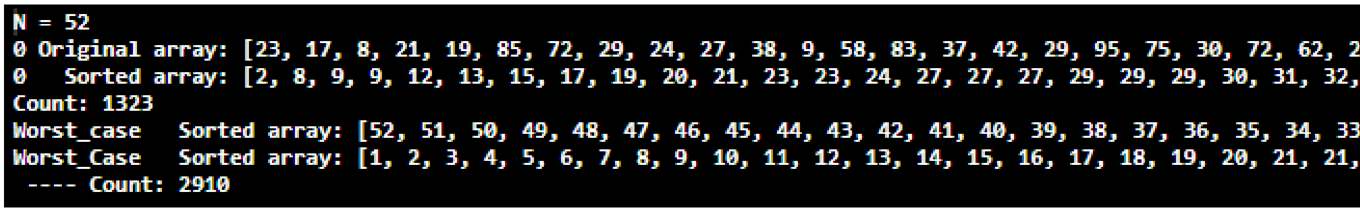
```
N = 30
0 Original array: [96, 66, 40, 70, 49, 53, 2, 78, 76, 33, 47, 67, 71, 10, 84, 11, 30, 75, 82, 99, 86, 65, 89,
0   Sorted array: [2, 10, 11, 30, 31, 33, 33, 40, 41, 45, 47, 49, 53, 65, 66, 67, 70, 71, 72, 75, 76, 78, 82,
Count: 487
Worst_case   Sorted array: [30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11,
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 21, 22,
 ---- Count: 1018
```



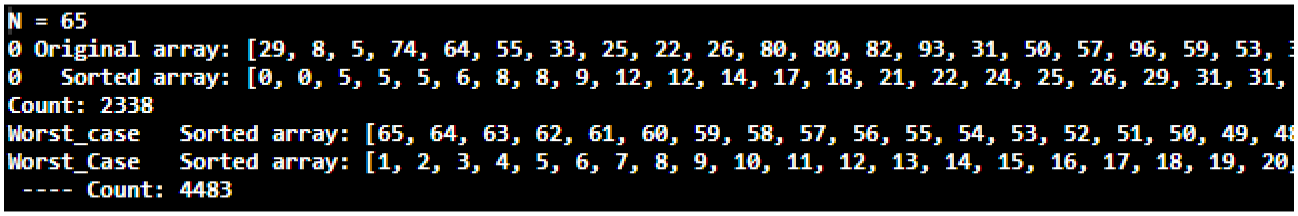
```
N = 35
0 Original array: [66, 11, 57, 74, 71, 65, 94, 87, 97, 1, 99, 52, 8, 58, 99, 53, 73, 65, 18, 72, 65, 78, 86,
0   Sorted array: [1, 2, 8, 11, 11, 11, 11, 18, 40, 52, 53, 57, 57, 58, 61, 62, 65, 65, 65, 66, 68, 71, 72,
Count: 732
Worst_case   Sorted array: [35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 21, 20, 19, 18, 17,
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 21, 22
 ---- Count: 1363
```



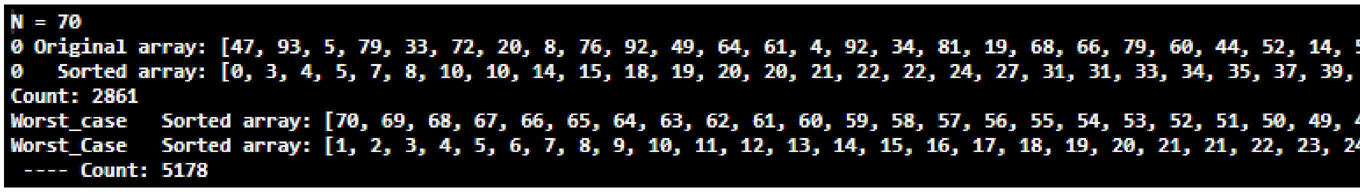
```
N = 49
0 Original array: [83, 73, 91, 16, 59, 53, 52, 75, 28, 77, 28, 51, 40, 29, 55, 17, 72, 41, 82, 34, 47, 75,
0   Sorted array: [4, 10, 14, 16, 17, 18, 19, 26, 28, 28, 29, 34, 34, 35, 38, 40, 41, 43, 44, 46, 47, 51,
Count: 1378
Worst_case   Sorted array: [49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30,
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 21,
 ---- Count: 2595
```



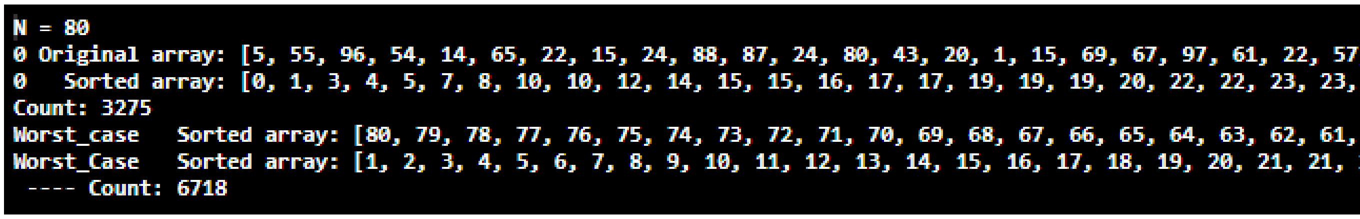
```
N = 52
0 Original array: [23, 17, 8, 21, 19, 85, 72, 29, 24, 27, 38, 9, 58, 83, 37, 42, 29, 95, 75, 30, 72, 62,
0   Sorted array: [2, 8, 9, 9, 12, 13, 15, 17, 19, 20, 21, 23, 23, 24, 27, 27, 27, 29, 29, 29, 30, 31, 32,
Count: 1323
Worst_case   Sorted array: [52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 21,
 ---- Count: 2910
```



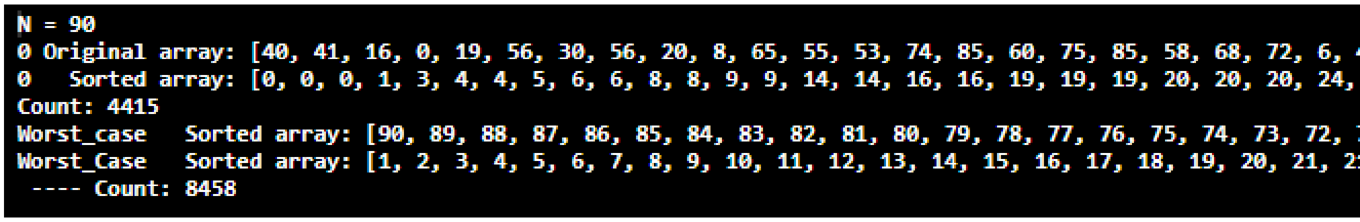
```
N = 65
0 Original array: [29, 8, 5, 74, 64, 55, 33, 25, 22, 26, 80, 80, 82, 93, 31, 50, 57, 96, 59, 53,
0   Sorted array: [0, 0, 5, 5, 5, 6, 8, 8, 9, 12, 12, 14, 17, 18, 21, 22, 24, 25, 26, 29, 31, 31,
Count: 2338
Worst_case   Sorted array: [65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49,
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
 ---- Count: 4483
```



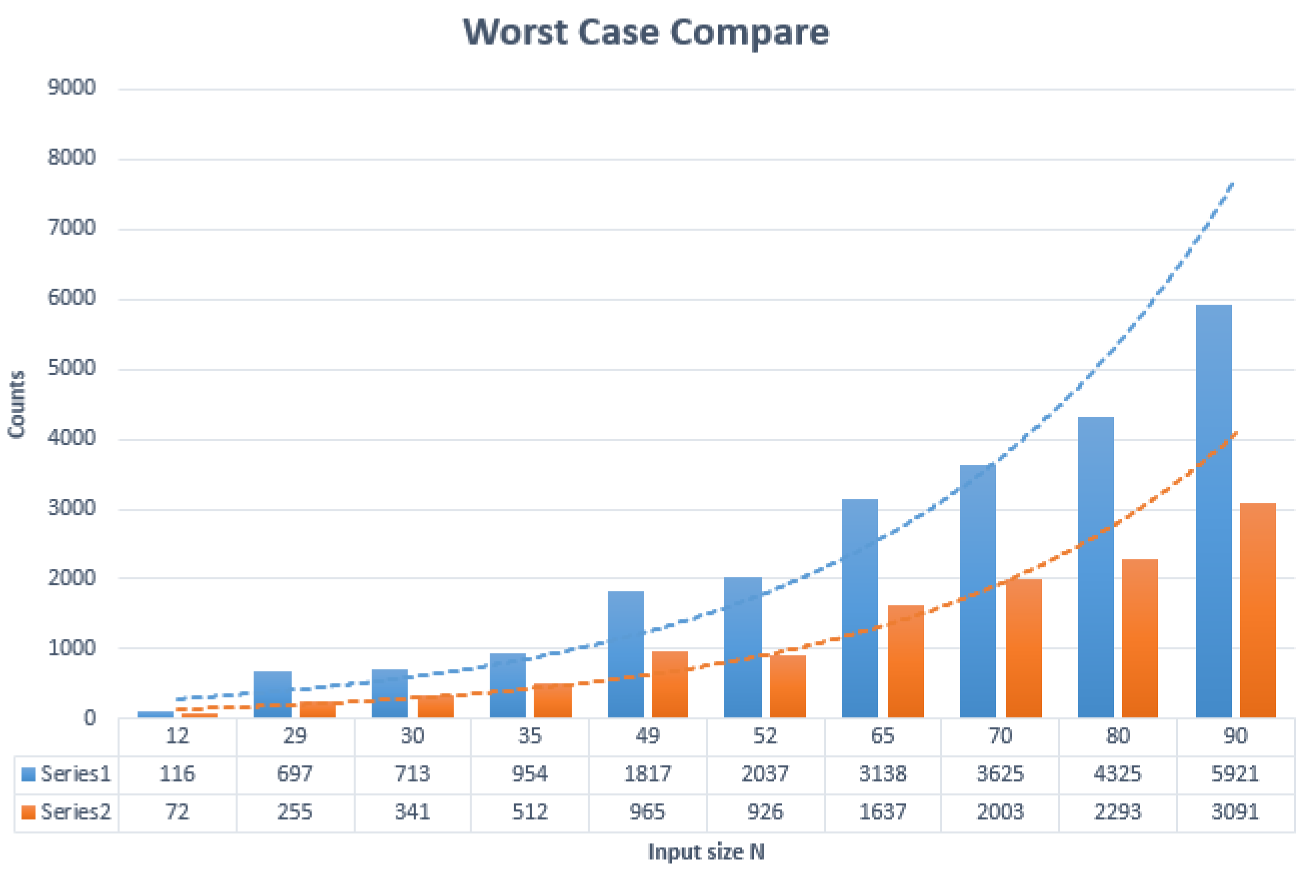
```
N = 70
0 Original array: [47, 93, 5, 79, 33, 72, 20, 8, 76, 92, 49, 64, 61, 4, 92, 34, 81, 19, 68, 66, 79, 60, 44, 52, 14, 5
0   Sorted array: [0, 3, 4, 5, 7, 8, 10, 10, 14, 15, 18, 19, 20, 20, 21, 22, 22, 24, 27, 31, 31, 33, 34, 35, 37, 39,
Count: 2861
Worst_case   Sorted array: [70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 4
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 21, 22, 23, 24
 ---- Count: 5178
```



```
N = 80
0 Original array: [5, 55, 96, 54, 14, 65, 22, 15, 24, 88, 87, 24, 80, 43, 20, 1, 15, 69, 67, 97, 61, 22, 57
0   Sorted array: [0, 1, 3, 4, 5, 7, 8, 10, 10, 12, 14, 15, 15, 16, 17, 17, 19, 19, 19, 20, 22, 22, 23, 23,
Count: 3275
Worst_case   Sorted array: [80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61,
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 21,
 ---- Count: 6718
```



```
N = 90
0 Original array: [40, 41, 16, 0, 19, 56, 30, 56, 20, 8, 65, 55, 53, 74, 85, 60, 75, 85, 58, 68, 72, 6, 4
0   Sorted array: [0, 0, 0, 1, 3, 4, 4, 5, 6, 6, 8, 8, 9, 9, 14, 14, 16, 16, 19, 19, 19, 20, 20, 20, 24,
Count: 4415
Worst_case   Sorted array: [90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 7
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 2
 ---- Count: 8458
```



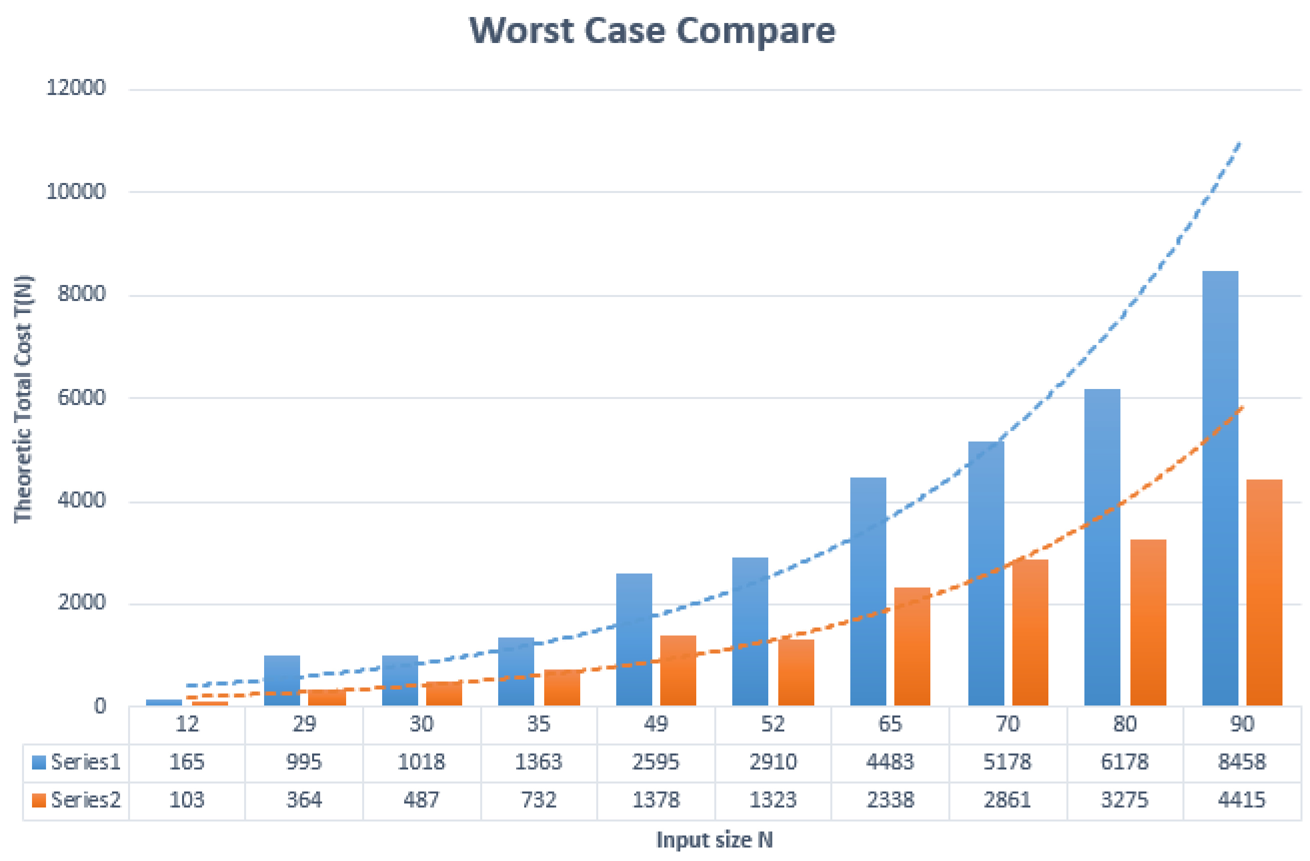
	12	29	30	35	49	52	65	70	80	90
Series1	116	697	713	954	1817	2037	3138	3625	4325	5921
Series2	72	255	341	512	965	926	1637	2003	2293	3091

3. Display a table of N, Actual count and the worst case T(N).

N	Worst Case T(N)	Actual Count
12	165	103

29	995	364
30	1018	487
35	1363	732
49	2595	1378
52	2910	1323
65	4483	2338
70	5178	2861
80	6178	3275
90	8458	4415

4. Plot a graph to compare the Worst Case Complexity of the algorithm and actual count putting counters in strategic points of your programs. Input data must be good for Worst Case Insertion Sort or average case insertion sort.



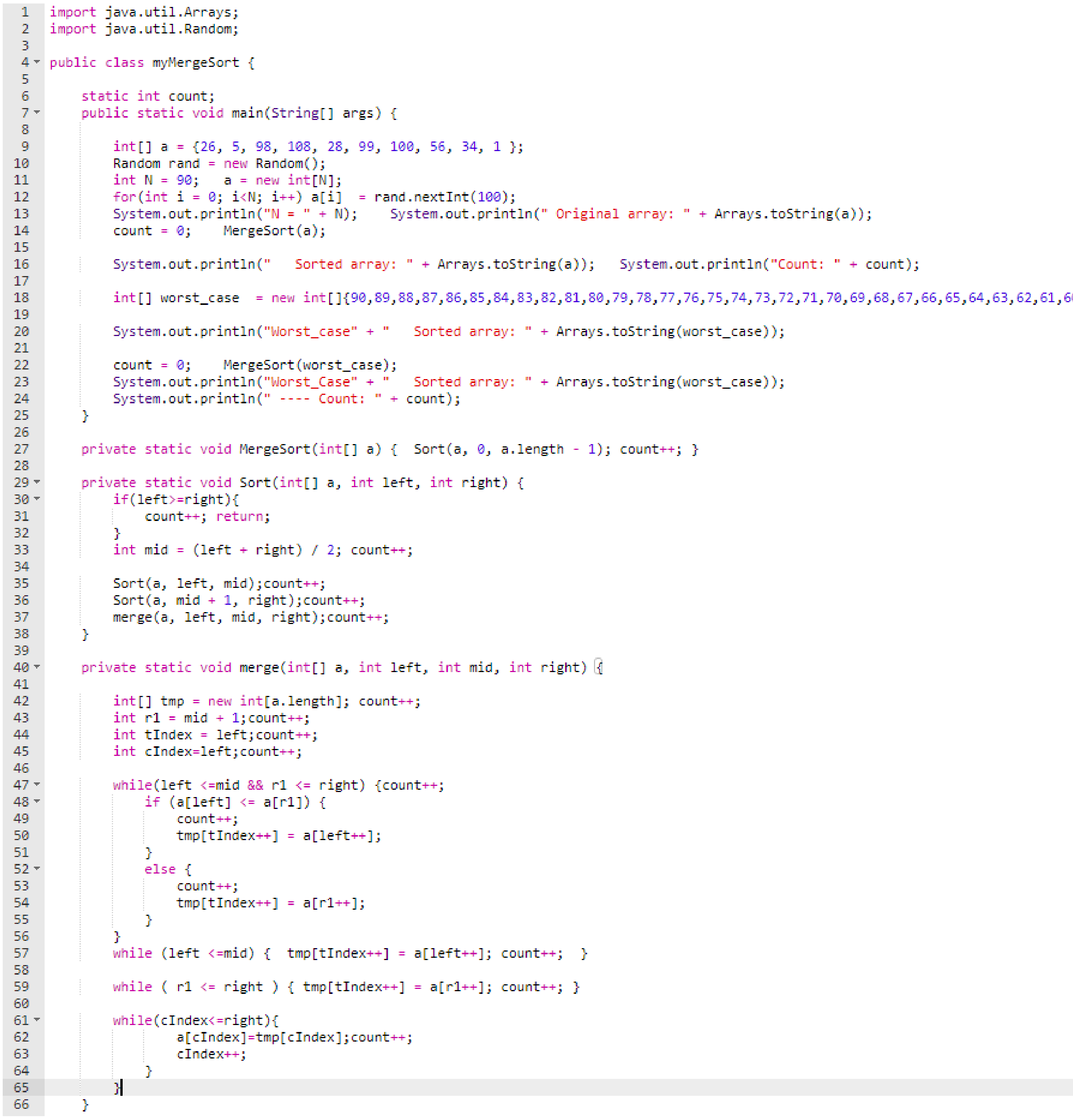
	12	29	30	35	49	52	65	70	80	90
🟦 Series1	165	995	1018	1363	2595	2910	4483	5178	6178	8458
🟧 Series2	103	364	487	732	1378	1323	2338	2861	3275	4415

Notes: The axis of the graph should be the theoretic total cost T(N) or the actual count vs. the N value.

b) Using Merge sort algorithm Implement a Program in any language you desire (preferably C++ or java) to sort an array of Real numbers of size N.

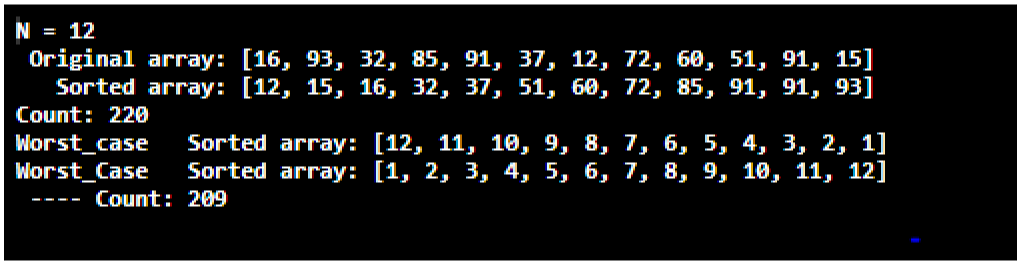
Do (1), (2), (3) and (4) same as in problem (a) using general case for Merge Sort.

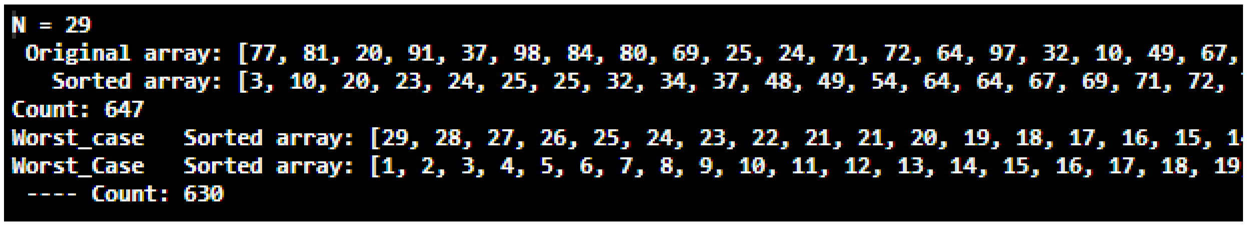
(1)

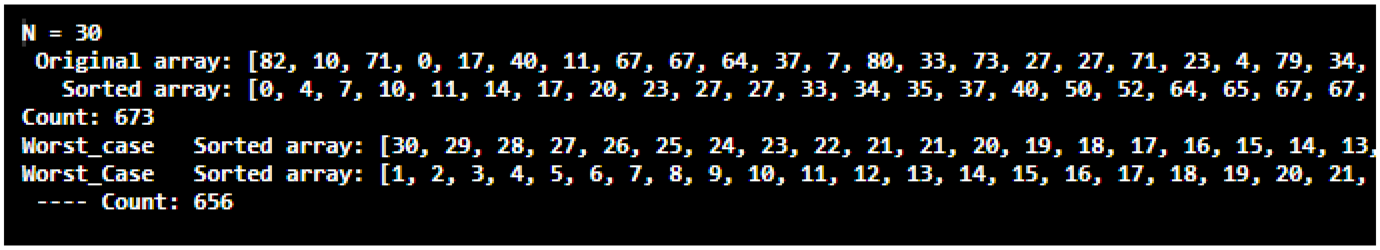


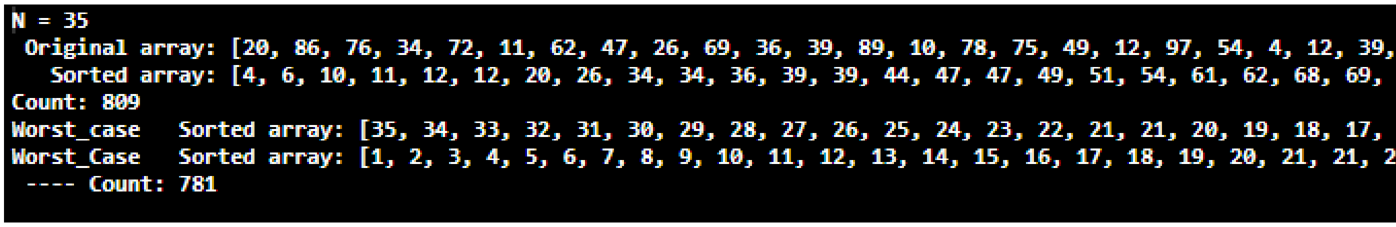
```java
import java.util.Arrays;
import java.util.Random;

public class myMergeSort {

    static int count;
    public static void main(String[] args) {

        int[] a = {26, 5, 98, 108, 28, 99, 100, 56, 34, 1 };
        Random rand = new Random();
        int N = 90;   a = new int[N];
        for(int i = 0; i<N; i++) a[i]  = rand.nextInt(100);
        System.out.println("N = " + N);    System.out.println(" Original array: " + Arrays.toString(a));
        count = 0;    MergeSort(a);

        System.out.println("   Sorted array: " + Arrays.toString(a));   System.out.println("Count: " + count);

        int[] worst_case  = new int[]{90,89,88,87,86,85,84,83,82,81,80,79,78,77,76,75,74,73,72,71,70,69,68,67,66,65,64,63,62,61,6

        System.out.println("Worst_case" + "   Sorted array: " + Arrays.toString(worst_case));

        count = 0;    MergeSort(worst_case);
        System.out.println("Worst_Case" + "   Sorted array: " + Arrays.toString(worst_case));
        System.out.println(" ---- Count: " + count);
    }

    private static void MergeSort(int[] a) {  Sort(a, 0, a.length - 1); count++; }

    private static void Sort(int[] a, int left, int right) {
        if(left>=right){
            count++; return;
        }
        int mid = (left + right) / 2; count++;

        Sort(a, left, mid);count++;
        Sort(a, mid + 1, right);count++;
        merge(a, left, mid, right);count++;
    }

    private static void merge(int[] a, int left, int mid, int right) {

        int[] tmp = new int[a.length]; count++;
        int r1 = mid + 1;count++;
        int tIndex = left;count++;
        int cIndex=left;count++;

        while(left <=mid && r1 <= right) {count++;
            if (a[left] <= a[r1]) {
                count++;
                tmp[tIndex++] = a[left++];
            }
            else {
                count++;
                tmp[tIndex++] = a[r1++];
            }
        }
        while (left <=mid) {  tmp[tIndex++] = a[left++]; count++;  }

        while ( r1 <= right ) { tmp[tIndex++] = a[r1++]; count++; }

        while(cIndex<=right){
                a[cIndex]=tmp[cIndex];count++;
                cIndex++;
            }
        }
    }
```

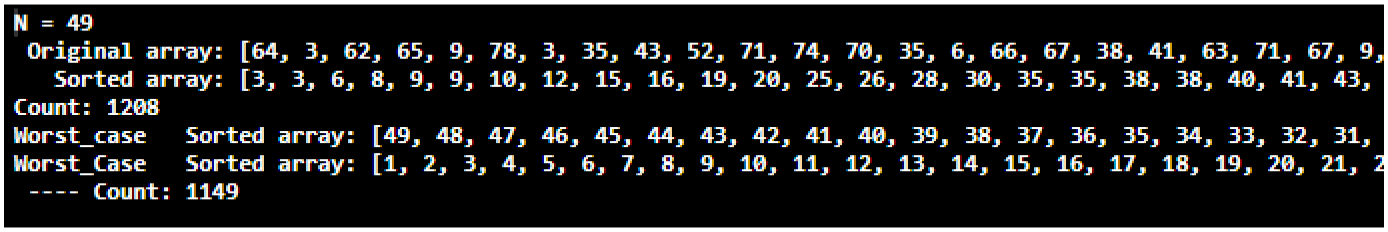
(2)

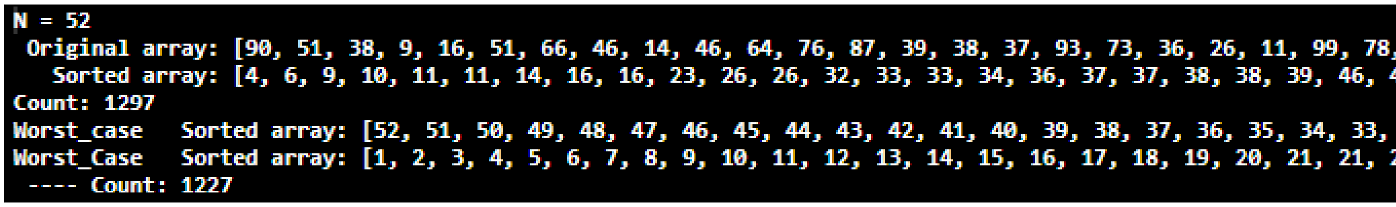

```
N = 12
 Original array: [16, 93, 32, 85, 91, 37, 12, 72, 60, 51, 91, 15]
   Sorted array: [12, 15, 16, 32, 37, 51, 60, 72, 85, 91, 91, 93]
Count: 220
Worst_case   Sorted array: [12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
 ---- Count: 209
```

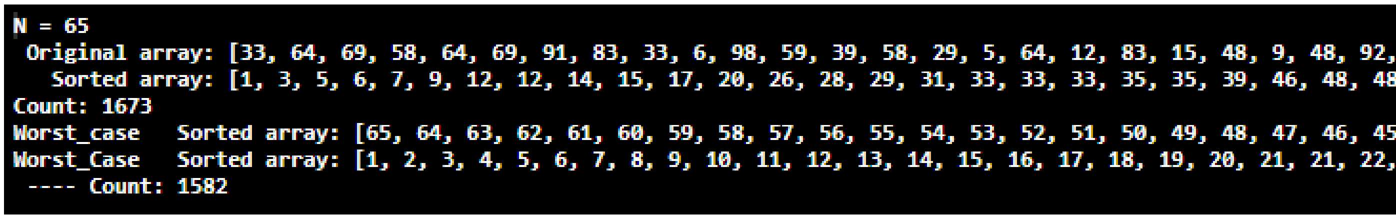

```
N = 29
 Original array: [77, 81, 20, 91, 37, 98, 84, 80, 69, 25, 24, 71, 72, 64, 97, 32, 10, 49, 67,
   Sorted array: [3, 10, 20, 23, 24, 25, 25, 32, 34, 37, 48, 49, 54, 64, 64, 67, 69, 71, 72,
Count: 647
Worst_case   Sorted array: [29, 28, 27, 26, 25, 24, 23, 22, 21, 21, 20, 19, 18, 17, 16, 15, 1
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19
 ---- Count: 630
```

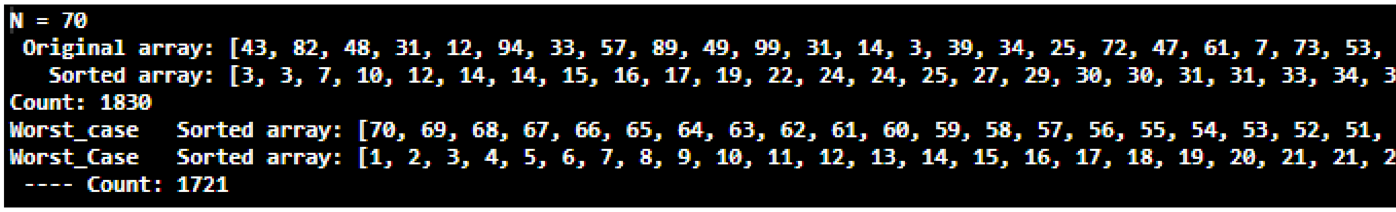

```
N = 30
 Original array: [82, 10, 71, 0, 17, 40, 11, 67, 67, 64, 37, 7, 80, 33, 73, 27, 27, 71, 23, 4, 79, 34,
   Sorted array: [0, 4, 7, 10, 11, 14, 17, 20, 23, 27, 27, 33, 34, 35, 37, 40, 50, 52, 64, 65, 67, 67,
Count: 673
Worst_case   Sorted array: [30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 21, 20, 19, 18, 17, 16, 15, 14, 13,
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
 ---- Count: 656
```


```
N = 35
 Original array: [20, 86, 76, 34, 72, 11, 62, 47, 26, 69, 36, 39, 89, 10, 78, 75, 49, 12, 97, 54, 4, 12, 39,
   Sorted array: [4, 6, 10, 11, 12, 12, 20, 26, 34, 34, 36, 39, 39, 44, 47, 47, 49, 51, 54, 61, 62, 68, 69,
Count: 809
Worst_case   Sorted array: [35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 21, 20, 19, 18, 17,
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 21, 2
 ---- Count: 781
```

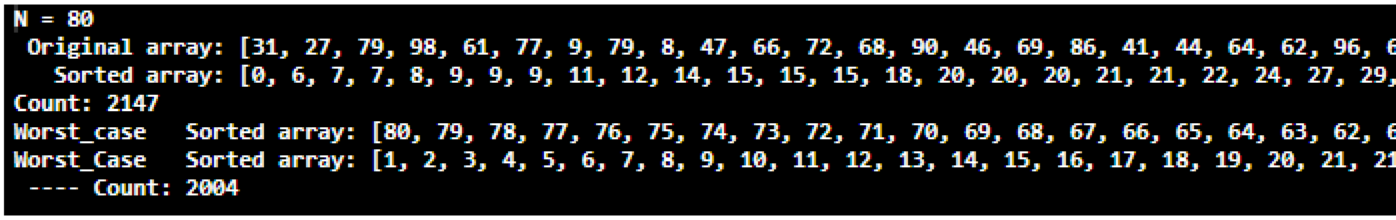

```
N = 49
 Original array: [64, 3, 62, 65, 9, 78, 3, 35, 43, 52, 71, 74, 70, 35, 6, 66, 67, 38, 41, 63, 71, 67, 9,
   Sorted array: [3, 3, 6, 8, 9, 9, 10, 12, 15, 16, 19, 20, 25, 26, 28, 30, 35, 35, 38, 38, 40, 41, 43,
Count: 1208
Worst_case   Sorted array: [49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31,
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 2
 ---- Count: 1149
```


```
N = 52
 Original array: [90, 51, 38, 9, 16, 51, 66, 46, 14, 46, 64, 76, 87, 39, 38, 37, 93, 73, 36, 26, 11, 99, 78,
   Sorted array: [4, 6, 9, 10, 11, 11, 14, 16, 16, 23, 26, 26, 32, 33, 33, 34, 36, 37, 37, 38, 38, 39, 46, 4
Count: 1297
Worst_case   Sorted array: [52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33,
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 21, 2
 ---- Count: 1227
```

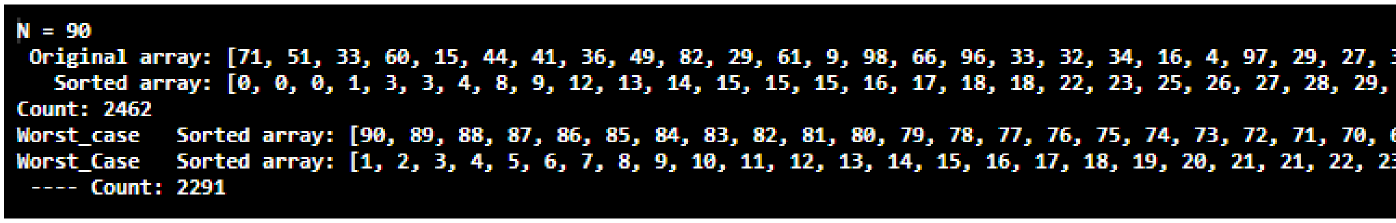

```
N = 65
 Original array: [33, 64, 69, 58, 64, 69, 91, 83, 33, 6, 98, 59, 39, 58, 29, 5, 64, 12, 83, 15, 48, 9, 48, 92,
   Sorted array: [1, 3, 5, 6, 7, 9, 12, 12, 14, 15, 17, 20, 26, 28, 29, 31, 33, 33, 33, 35, 35, 39, 46, 48, 48
Count: 1673
Worst_case   Sorted array: [65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 21, 22,
 ---- Count: 1582
```



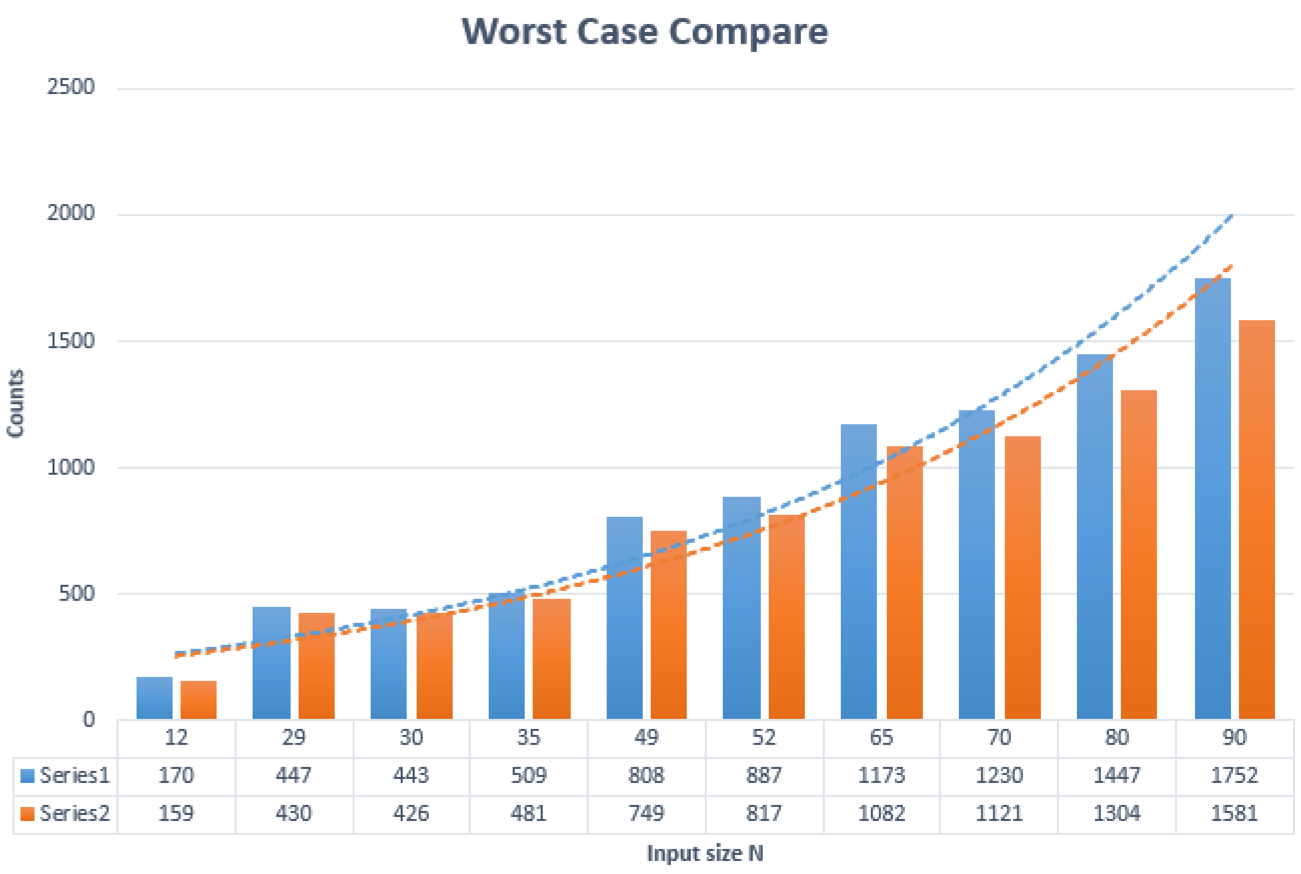
```
N = 70
 Original array: [43, 82, 48, 31, 12, 94, 33, 57, 89, 49, 99, 31, 14, 3, 39, 34, 25, 72, 47, 61, 7, 73, 53,
   Sorted array: [3, 3, 7, 10, 12, 14, 14, 15, 16, 17, 19, 22, 24, 24, 25, 27, 29, 30, 30, 31, 31, 33, 34, 3
Count: 1830
Worst_case   Sorted array: [70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51,
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 21, 2
 ---- Count: 1721
```



```
N = 80
 Original array: [31, 27, 79, 98, 61, 77, 9, 79, 8, 47, 66, 72, 68, 90, 46, 69, 86, 41, 44, 64, 62, 96, 6
   Sorted array: [0, 6, 7, 7, 8, 9, 9, 9, 11, 12, 14, 15, 15, 15, 18, 20, 20, 20, 21, 21, 22, 24, 27, 29,
Count: 2147
Worst_case   Sorted array: [80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 6
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 21
 ---- Count: 2004
```



```
N = 90
 Original array: [71, 51, 33, 60, 15, 44, 41, 36, 49, 82, 29, 61, 9, 98, 66, 96, 33, 32, 34, 16, 4, 97, 29, 27, 3
   Sorted array: [0, 0, 0, 1, 3, 3, 4, 8, 9, 12, 13, 14, 15, 15, 15, 16, 17, 18, 18, 22, 23, 25, 26, 27, 28, 29,
Count: 2462
Worst_case   Sorted array: [90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 6
Worst_Case   Sorted array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 21, 22, 23
 ---- Count: 2291
```



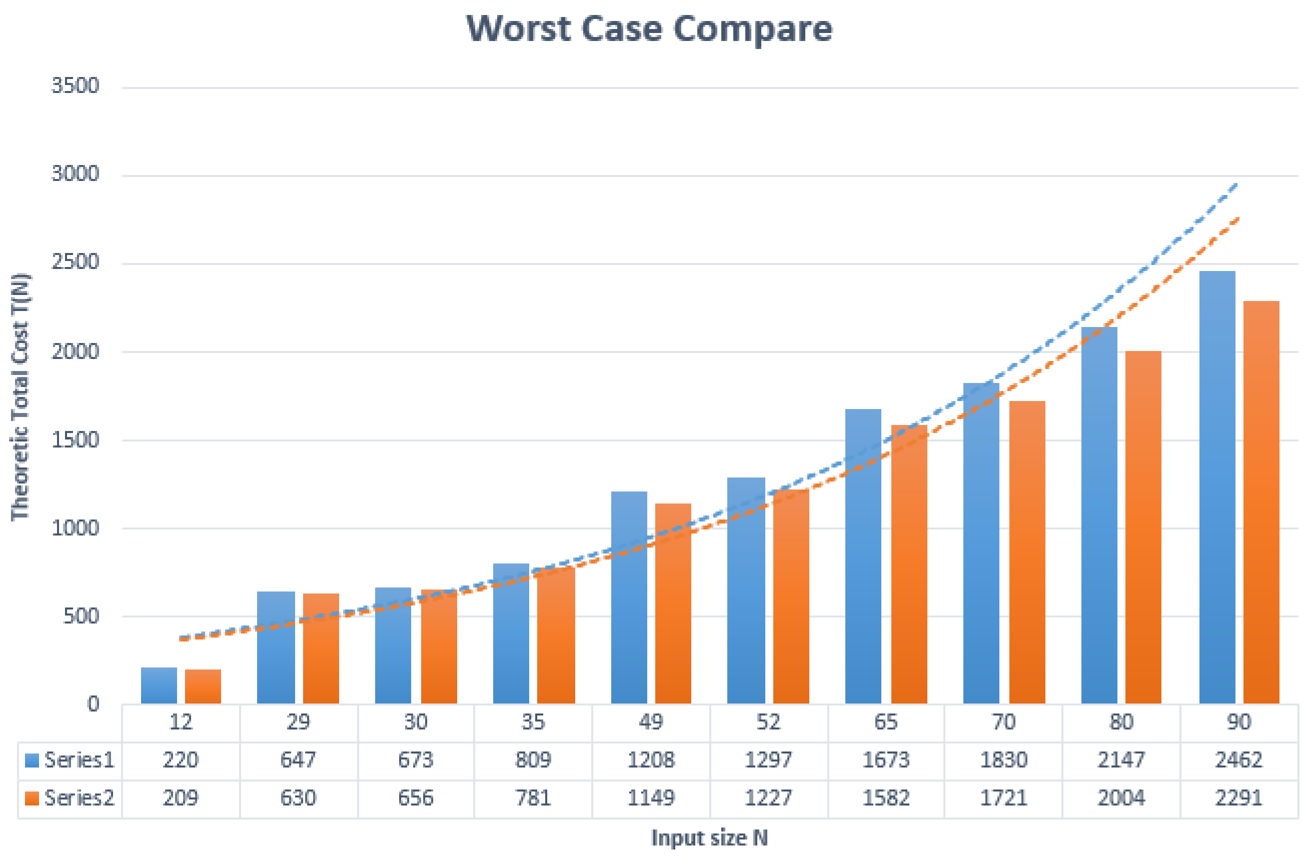
	12	29	30	35	49	52	65	70	80	90
🟦 Series1	170	447	443	509	808	887	1173	1230	1447	1752
🟧 Series2	159	430	426	481	749	817	1082	1121	1304	1581

(3)

N	Worst Case T(N)	Actual Count

12	220	209
29	647	630
30	673	656
35	809	781
49	1208	1149
52	1297	1227
65	1673	1582
70	1830	1721
80	2147	2004
90	2462	2291

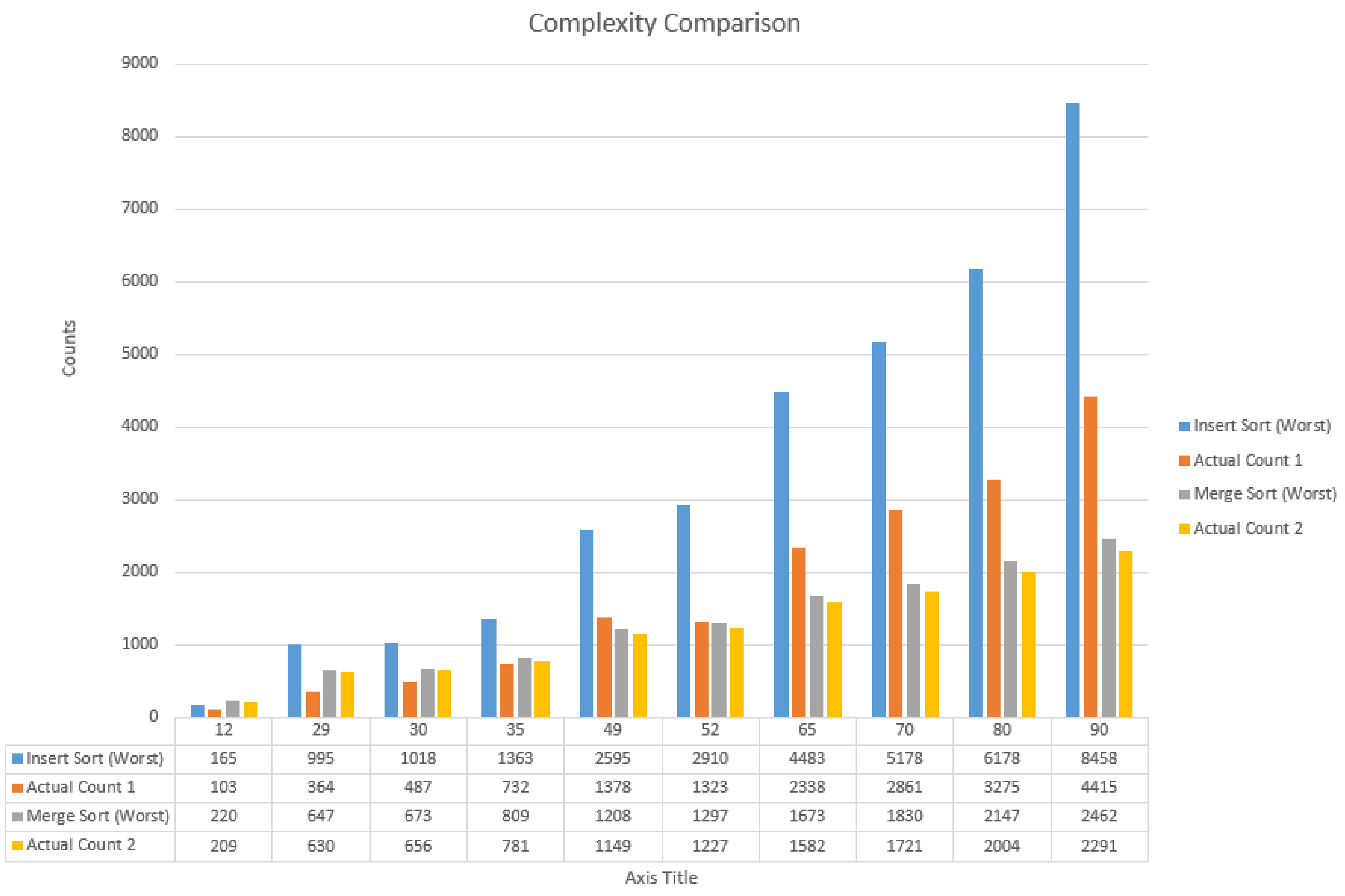
(4)



	12	29	30	35	49	52	65	70	80	90
🟦 Series1	220	647	673	809	1208	1297	1673	1830	2147	2462
🟧 Series2	209	630	656	781	1149	1227	1582	1721	2004	2291

c) Draw a third graph combining the results of (a) and (b)

The graphs of both algorithms should be combined into one.



	12	29	30	35	49	52	65	70	80	90
🟦 Insert Sort (Worst)	165	995	1018	1363	2595	2910	4483	5178	6178	8458
🟧 Actual Count 1	103	364	487	732	1378	1323	2338	2861	3275	4415
⬜ Merge Sort (Worst)	220	647	673	809	1208	1297	1673	1830	2147	2462
🟨 Actual Count 2	209	630	656	781	1149	1227	1582	1721	2004	2291

Notes: **Turn in with hard copy and must be professional and the turn in should include the source Program, and the displayed outputs as described above. Proper message in the output is required to indicate the execution and different outputs. Turn in all assignments in class. It will have late penalty 10% if your turn in after 7:00 pm and before next class.**