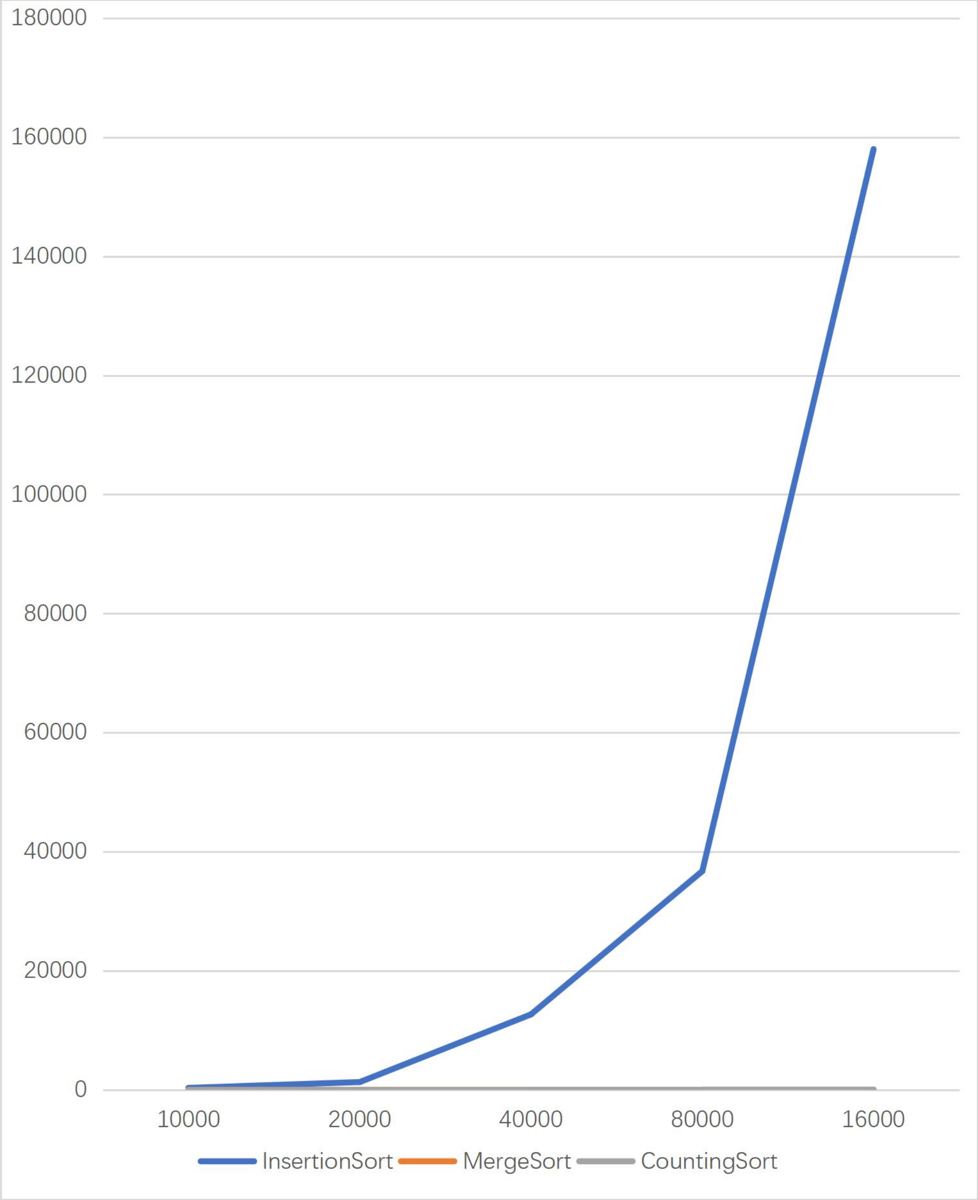
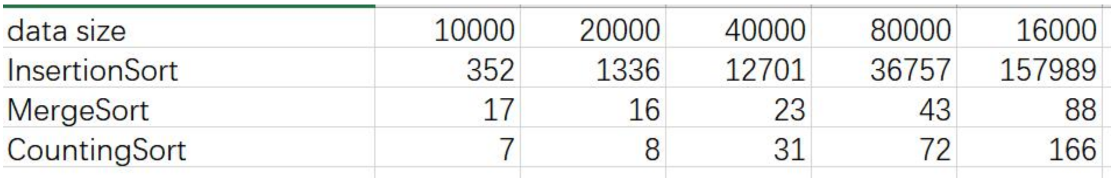
Tasks

1. Counting Sort vs. Insertion Sort vs. Merge Sort
   Plot the runtime for the time taken to sort the arraylist (in milliseconds) and record the values in a table. Briefly analyze the plot in terms of time complexity (or number of operations).
   Sol:



Here the counting sort and merge sort both have really small running time their line overlap in the table. But compare by the original data in the table shown below,



data size	10000	20000	40000	80000	16000
InsertionSort	352	1336	12701	36757	157989
MergeSort	17	16	23	43	88
CountingSort	7	8	31	72	166

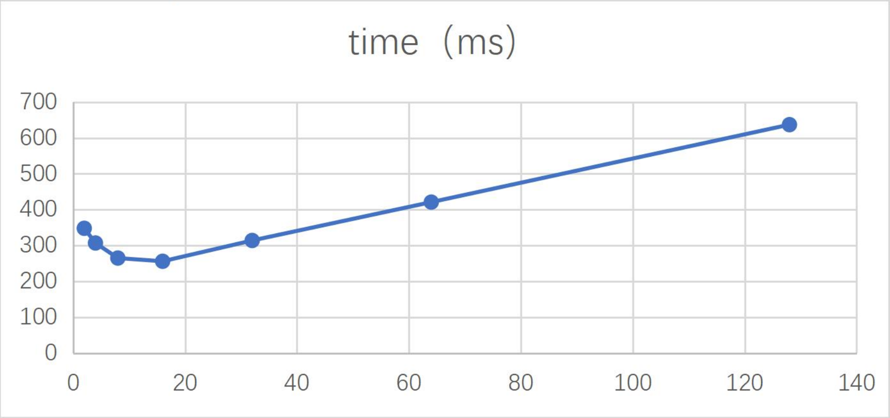
Merge sort is the fastest compared to others depending on time complexity.

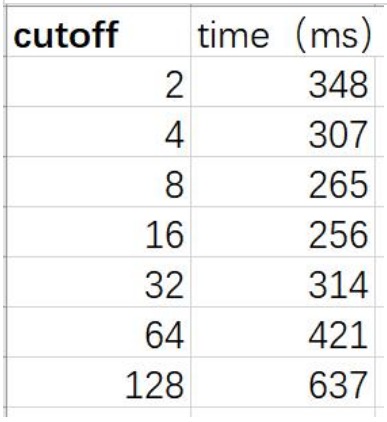
All sorting methods seem to have a longer running time as the data size increases.

2Testing Modified QuickSort Cutoff Values

Plot a line plot of the performance relative to the cutoff and record the values in a table. Which cutoff gave you the fastest performance? (for most implementations of modified quicksort there are 2 cutoffs that have very close runtimes, either of these will be accepted as correct answers)

According to the graph, I think it is the cutoff of 16,that gave me the fastest performance.

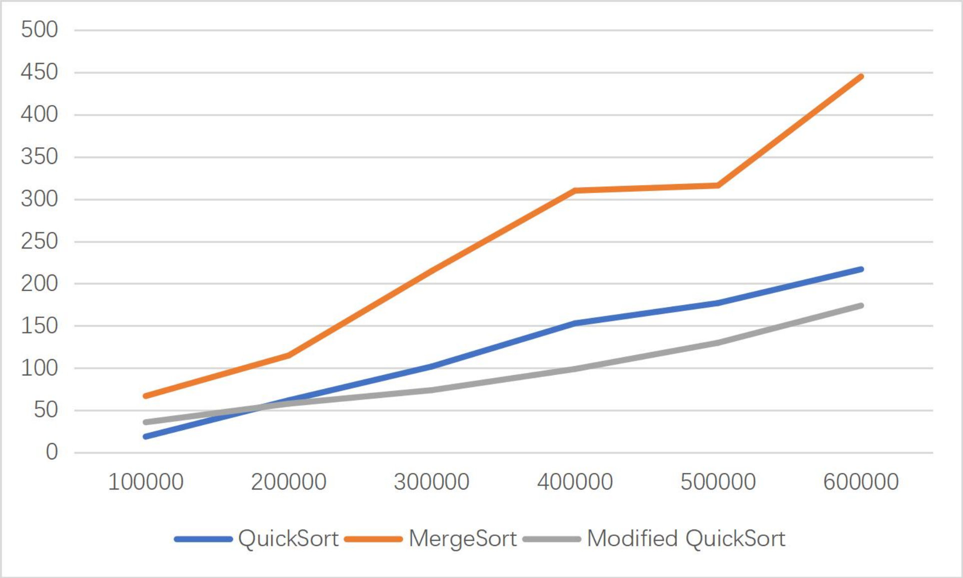


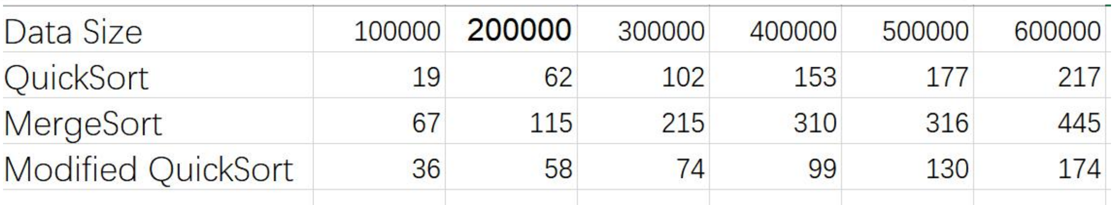


cutoff	time （ms）
2	348
4	307
8	265
16	256
32	314
64	421
128	637

3Testing Traditional QuickSort vs. Modified QuickSort vs Merge Sort

Plot the runtime of the traditional and modified Quicksort relative to the dataset size and record the values in a table.

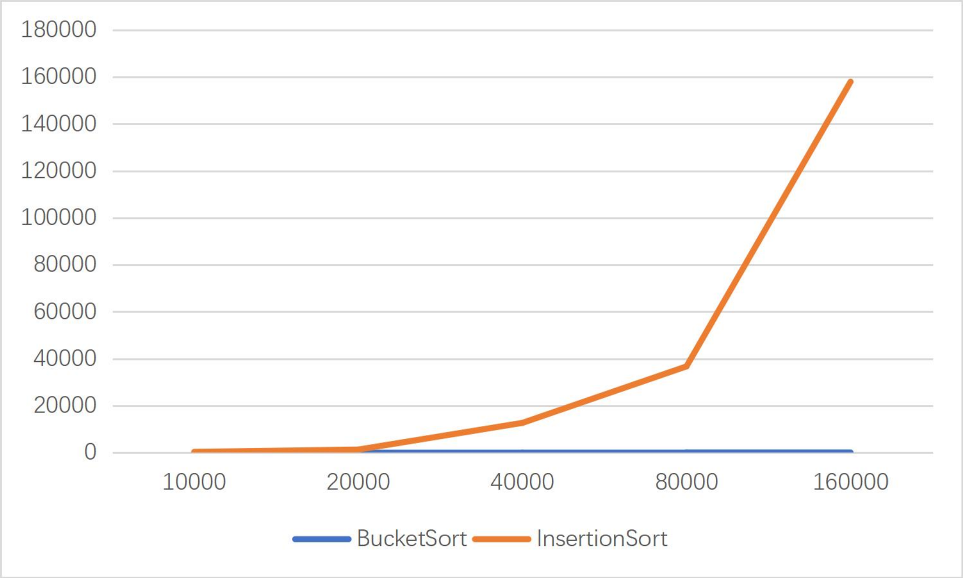


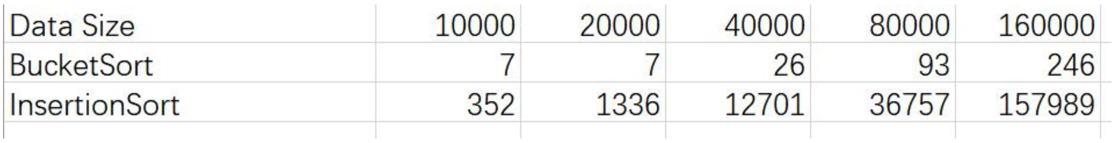


Data Size	100000	200000	300000	400000	500000	600000
QuickSort	19	62	102	153	177	217
MergeSort	67	115	215	310	316	445
Modified QuickSort	36	58	74	99	130	174

4 Testing Insertion Sort vs. Bucket Sort

Plot the runtime for the time taken to sort the arraylist (in milliseconds) and record the values in a table. Briefly analyze the plot in terms of time complexity (or number of operations).





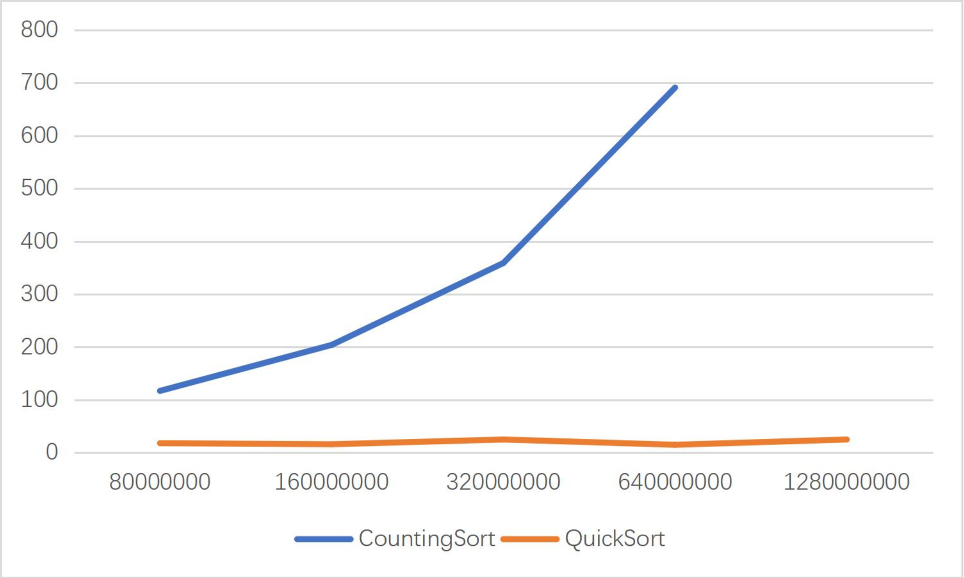
Data Size	10000	20000	40000	80000	160000
BucketSort	7	7	26	93	246
InsertionSort	352	1336	12701	36757	157989

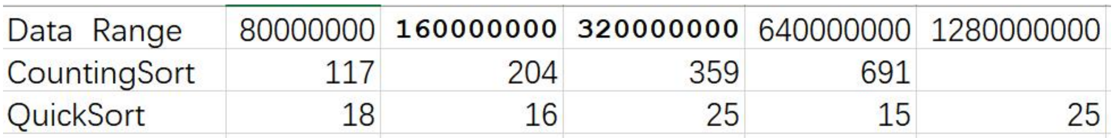
Apparently Bucket Sort has a mush smaller time complexity than insertion Sort, both the sorting methods ’ running time increases as the data size gets larger.

5. Testing Counting Sort vs. QuickSort with various ranges

Plot the runtime for the time taken to sort the arraylist (in milliseconds) of size 50000 for all listed ranges and record the values in a table. Do the same for arraylists of size 200000 (So you should have 2 plots). Briefly analyze the plots in terms of time complexity (or number of operations).

Because when I did the data range designed, the count sort method was working really fast, in order to observe the difference in speed of the two method, I multiplied every data range given by 100, when the data sizes is 50000, count sort method cannot stand data range of 1280000000 because it is out of the storage. This is as shown below:

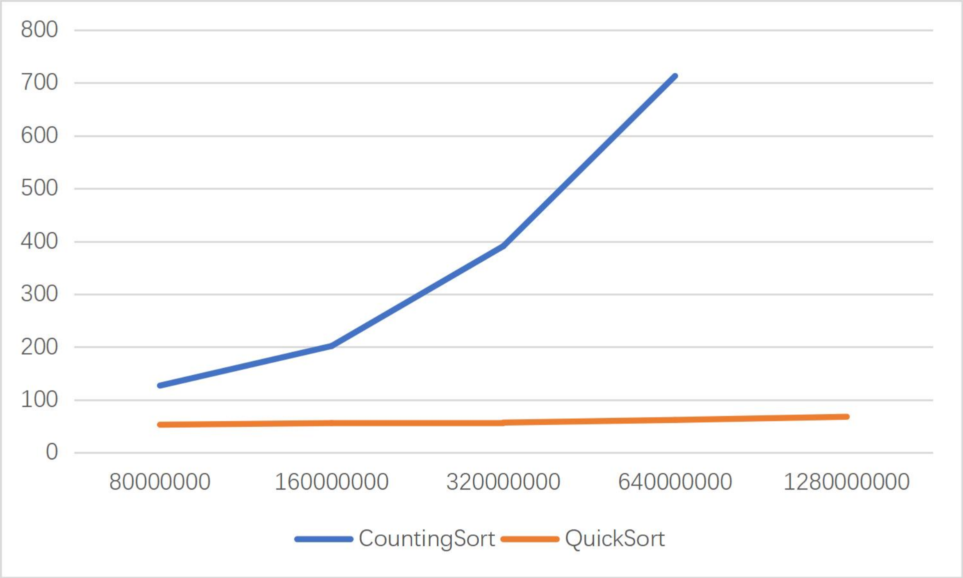


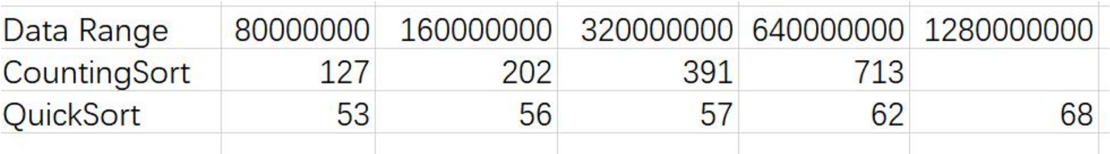


Data Range	80000000	160000000	320000000	640000000	1280000000
CountingSort	117	204	359	691	
QuickSort	18	16	25	15	25

Here We can see that the Counting sort’s time complexity is much larger than the time complexity of Quick sort in terms of the running time, both the sorting methods ’running time increases as the data size gets larger.

When the data size is 200000, and again count sort method cannot stand data range of 1280000000 because it is out of the storage:





Data Range	80000000	160000000	320000000	640000000	1280000000
CountingSort	127	202	391	713	
QuickSort	53	56	57	62	68

Here We can see that the Counting sort’s time complexity is much larger than the time complexity of Quick sort in terms of the running time, both the sorting methods ’running time increases as the data size gets larger.

And when data size increases from 50000 to 200000, the time taken for the sorting methods to run on each level of data range all increased.