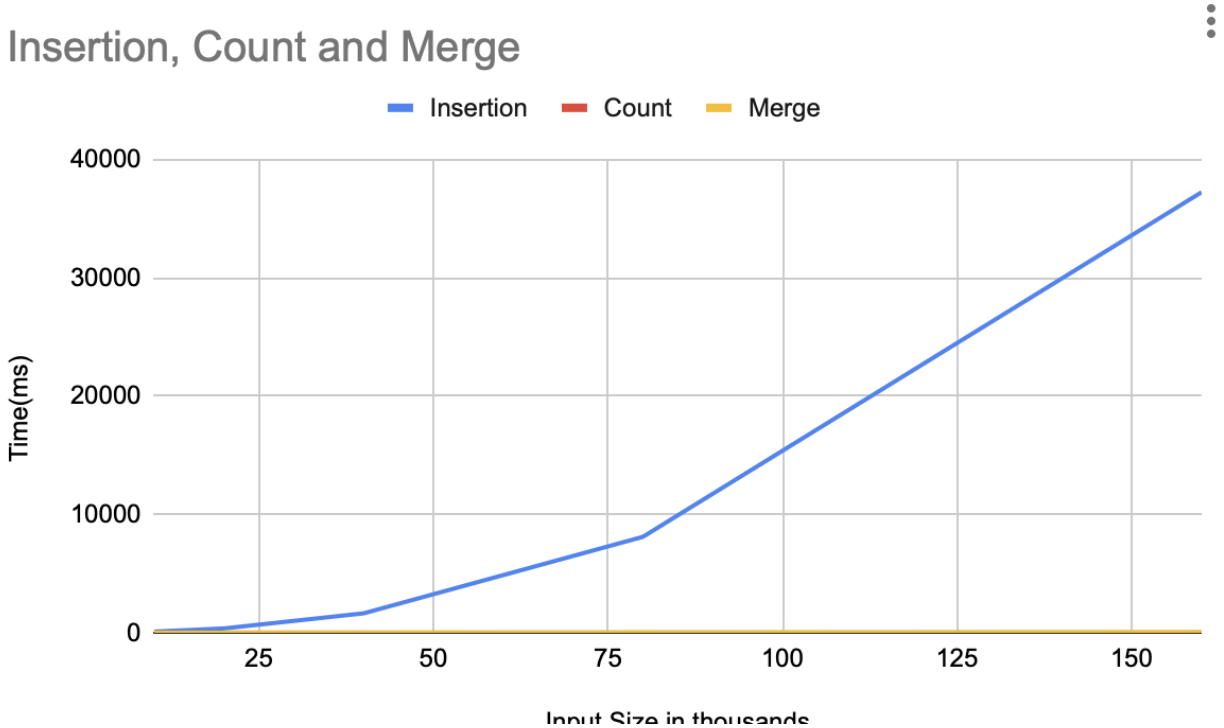
PA3 Part 2.2
Christian Guerra
A17660168
DSC 30

Insertion Sort vs Merge Sort vs Count Sort

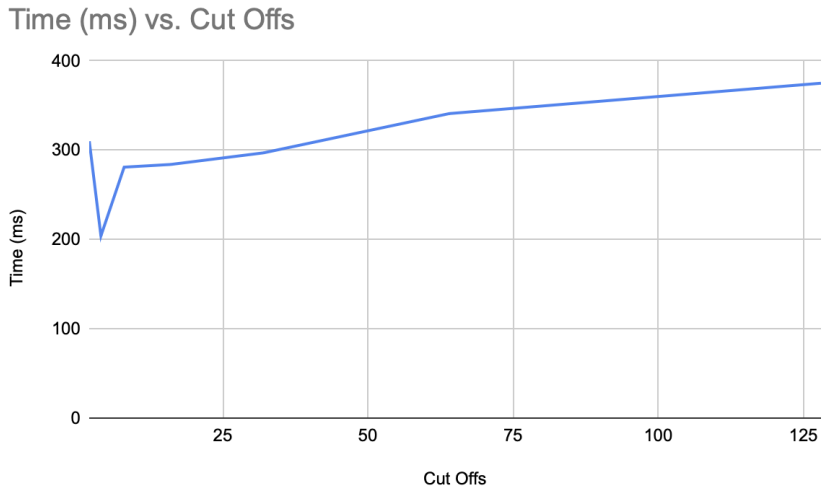


	Insertion Sort	Count Sort	Merge Sort
10	94	2	7
20	361	2	6
40	1636	1	13
80	8102	2	27
160	37229	2	56

1. *Insertion Sort:* This algorithm has a time complexity of approximately O(n^2) which means that the number of operations increases exponentially. We can see from the plot that the Insertion Sort line is the steepest, takes the most time out of the three.
2. *Counting Sort: T*his algorithm has a time complexity of approximately O(n), since the range is constant in this case. We can see from the plot that the Counting Sort line is almost flat, proving that in this case is more efficient than insertion sort.

3. *Merge Sort:* This algorithm has a time complexity of approximately O(n log n) therefore being the fastest of the three algorithms.

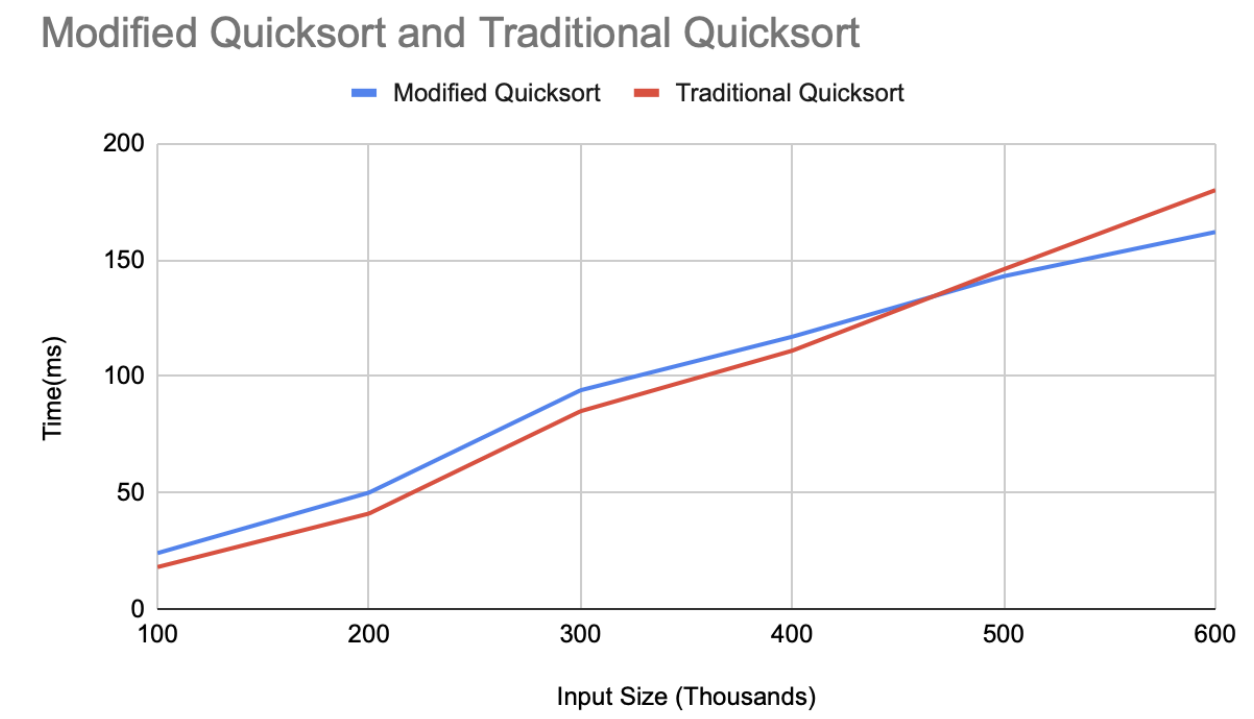
Modified Quicksort and different Cut-offs:



Cut Offs	Time (ms)
2	310
4	204
8	281
16	284
32	297
64	341
128	375

Analysis: As seen in the graph, this sorting algorithm takes approximately exponential time complexity O(n^2) to sort large data, however this depends on the different cut off values. By looking at the graph we are able to see that a cut off value of 8, is able to sort a Data Size of 1,000,000 relatively faster than others (281 milliseconds to be exact).

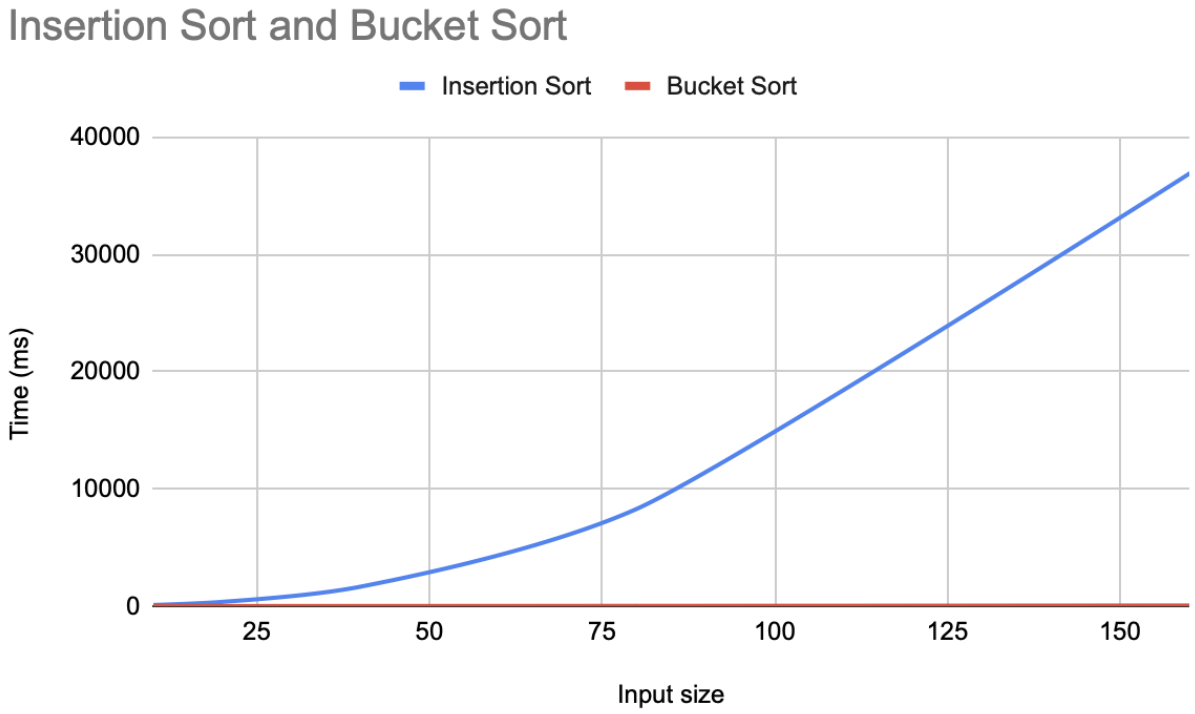
Modified Quicksort vs Traditional Quicksort:



	Modified Quicksc	Traditional Quicksort
100	24	18
200	50	41
300	94	85
400	117	111
500	143	146
600	162	180

Analysis: As seen in the graph both methods follow a time complexity of O(n^2) on which the traditional quicksort outperforms the Modified quicksort for relatively smaller datasets, however this seemed to change as the size of the data continue to increase on which they both performed about the same and at a data size of 500k, the modified quicksort started to outperform the traditional quicksort.

Insertion Sort vs Bucket Sort



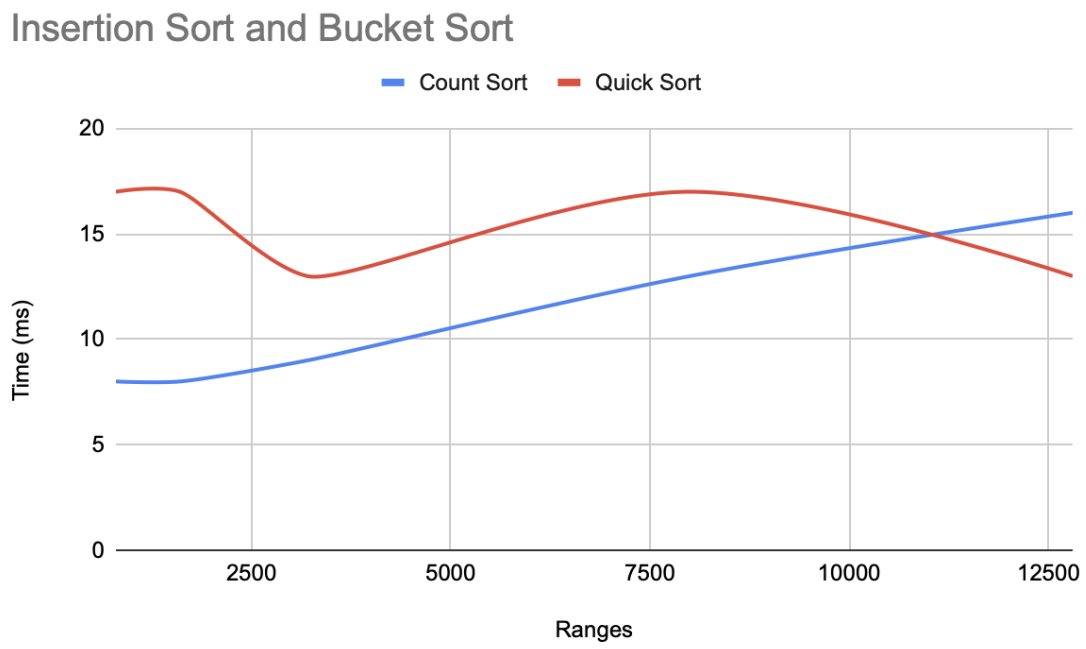
	Insertion Sort	Bucket Sort
10000	93 ms	6 ms
20000	360 ms	2 ms
40000	1655 ms	6 ms
80000	8280 ms	14 ms
160000	36880 ms	44 ms

Analysis: As we can see, Insertion sort time complexity is approximately O(n^2) while Bucket Sort time complexity is approximately O(n+K). This proves that Bucket Sort outperformed Insertion Sort in both small and large Data Sizes from 10 thousand to 160 thousand.

Counting Sort vs Quick Sort In Various ranges.

Data size 50,000:

(Data on table is milliseconds)

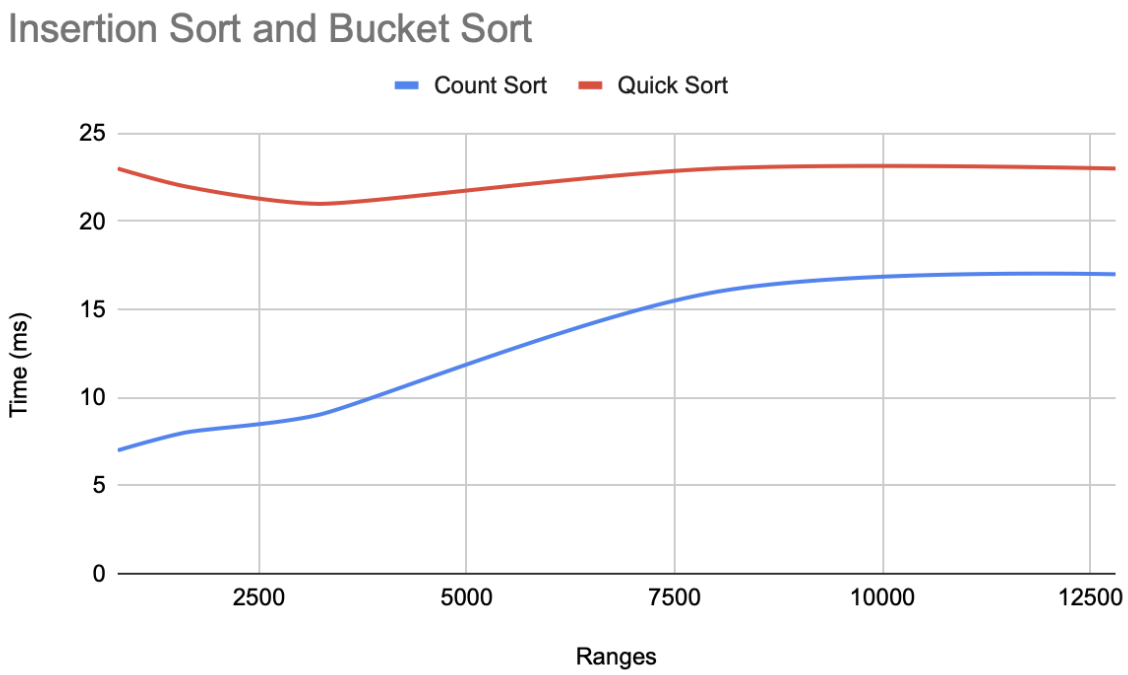


	Count Sort	Quick Sort
800	8	17
1600	8	17
3200	9	13
8000	13	17
12800	16	13

Analysis: Count Sort seems to have a O(n^2) time complexity and increases as the range of Min and Max increases, On the other hand Quick Sorts seems to have a non constant time complexity however as range of Min and Max increases it seems to have a time complexity of approximately O(n*logn) outperforming Count sort in data sets that go over 1.1 million.

Data Size 80,000

(Data on table is milliseconds)



	Count Sort	Quick Sort
800	7	23
1600	8	22
3200	9	21
8000	16	23
12800	17	23

Analysis: Given the definite ranges of Min and Max we can see that Count Sort seems to have a O(n^2) time complexity with the relatively small data and ends up adapting into a O(n) time complexity. On the other hand, Quick Sort seems to Have O(n) Time complexity, however it takes more time throughout to process relatively large data sets.