Alexander Rodriguez
Data Structures
Sorting Analysis Commentary

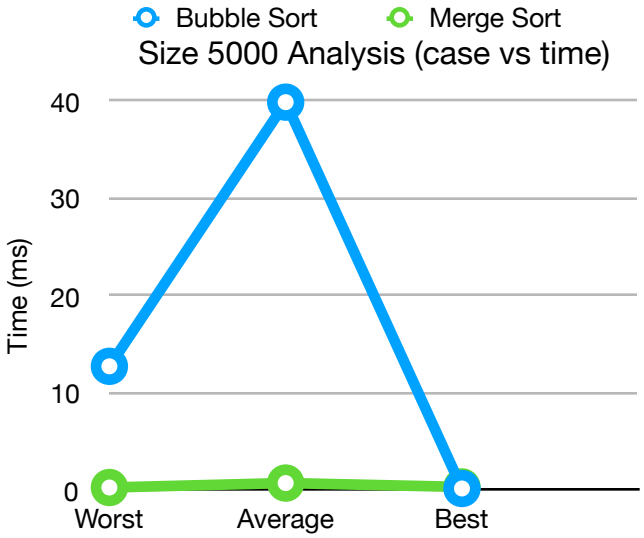
Analysis of Data

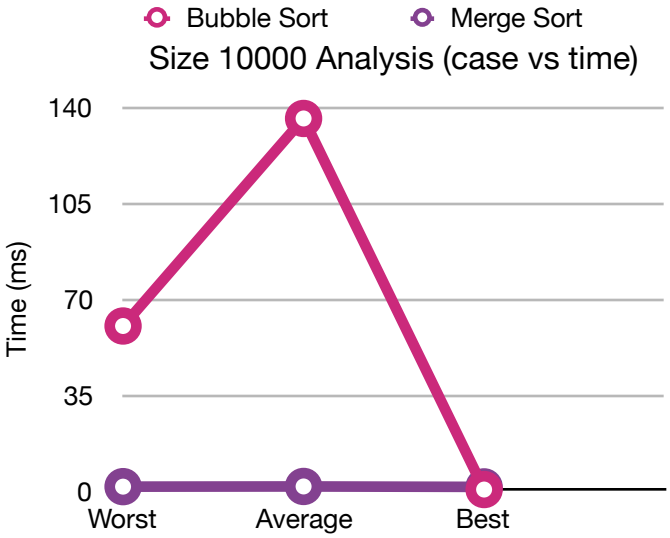
From the data I received from by testing the performance of a Bubble Sort algorithm vs a Merge Sort algorithm gave some surprising and interesting results. To no surprise, the Merge Sort algorithm did extremely better in comparison to the Bubble Sort algorithm in terms of the time complexity. Merge Sort has a worst case time complexity of O(n log n), but this is also the result for both the average and best cases. Meanwhile Bubble Sort has a time complexity of Ω(n) in just the best case, Θ(n^2) for the average case, and O(n^2) for the worst case.

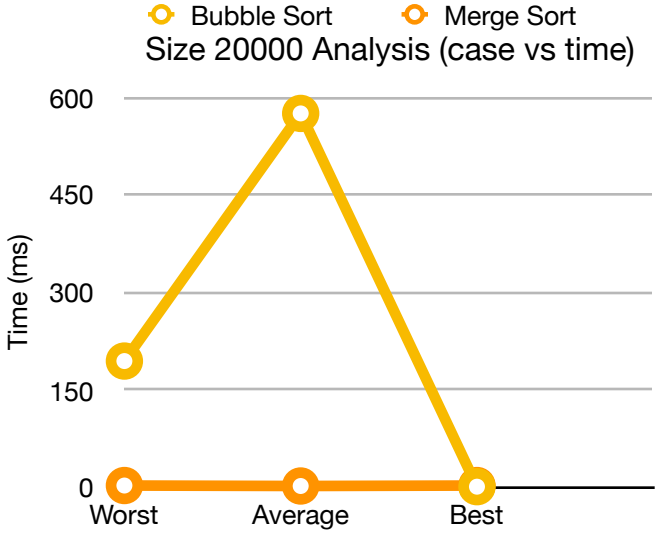
File Size	Merge Sort(in milliseconds)			Bubble Sort(in milliseconds)		
	Worst	Average	Best	Worst	Average	Best
5000	0.320624	0.777791	0.380998	12.75599	39.858489	0.219941
10000	1.458932	1.529143	1.37088	60.211722	136.116982	0.40178
20000	2.618828	1.887292	2.587077	194.181167	575.289839	0.846774
50000	4.745755	3.361802	3.261522	1237.050011	3836.45186	1.495021

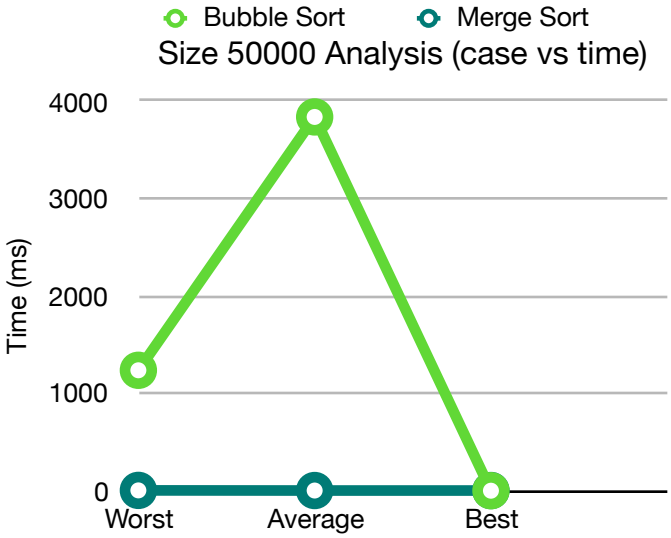
As can be seen from the data above, the difference in time time of both of these algorithms is considerably large. Although they both complete the same task, one algorithm does it more efficiently than the other. One thing that is surprising is that in each file size, the average case seems to have done worse in comparison to worst case. I believe this is due to branch prediction for the best and worst cases, making it so the program predicts whether or not the next value will be larger or smaller using previous tests used while it is using pipelining. Additionally, I believe that the order at which the programs were meant to run may have also had an effect of run time due to pipelining where one part of the program may have to stall in order to complete a previous part. This can be seen by the graphs below.

Graphs

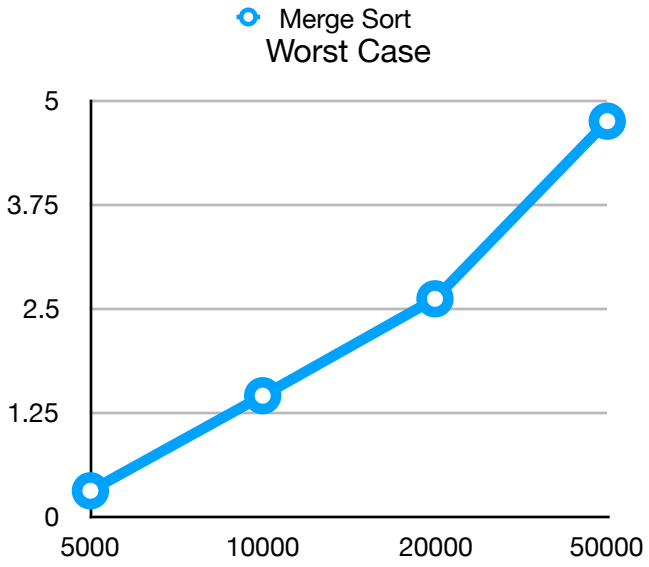


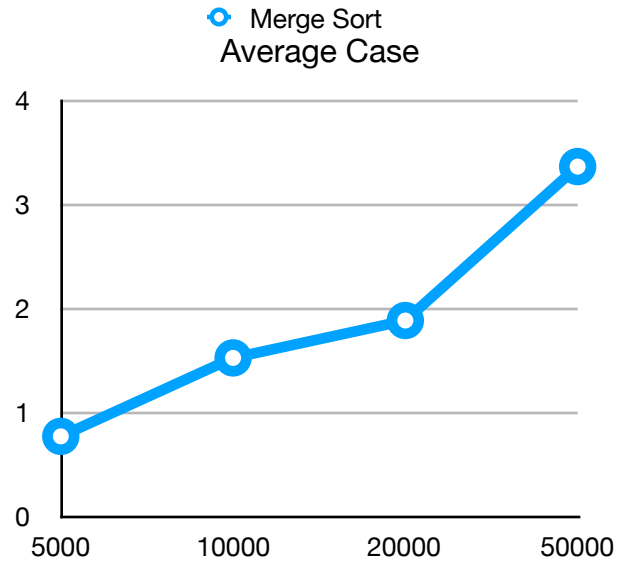


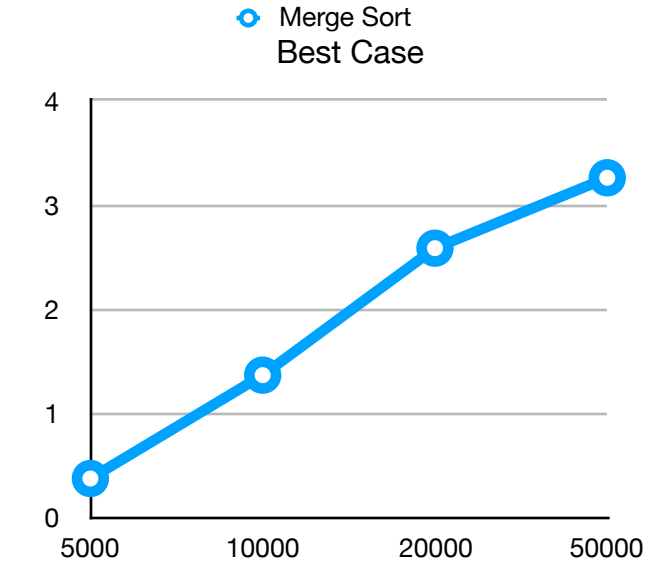


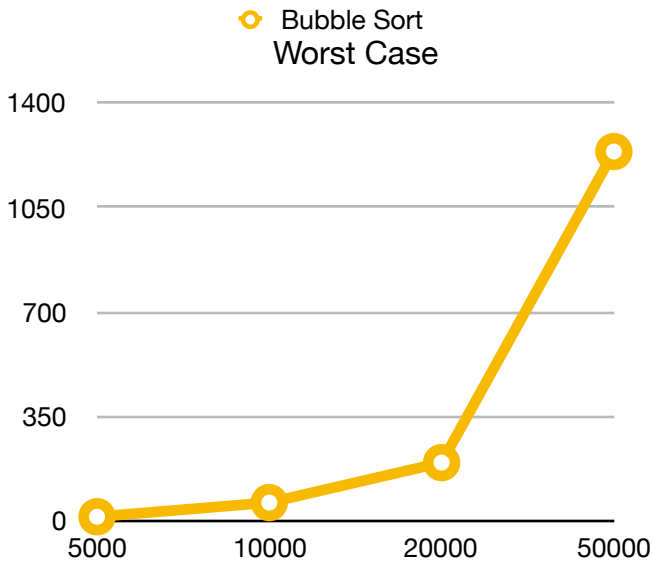


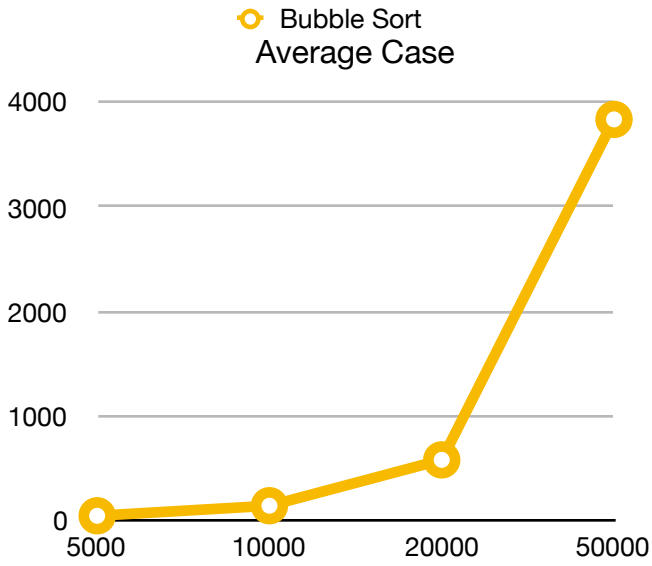
Multiple conclusions can be taken from the graphs above. As you can see, there is a similar trend throughout the data where the average, random cases, fails in comparison to the worst case due to branch prediction assisting the program in the worst case. Although no as clear as it is for the Bubble Sort, the average case for Merge Sort is also effected by the use of branch prediction. However, it is on a much smaller scale than Bubble Sort. Another conclusion that can be taken from the graphs above is the time difference between each points of the sorting algorithms. While Merge Sort is nearly consistent at about 0 ms in each graph, Bubble Sort is increasing immensely as can be seen from the highest reference points for the time in milliseconds for each graph. Showing the effect of file sizes on the run time. Additional graph examples are provided below.

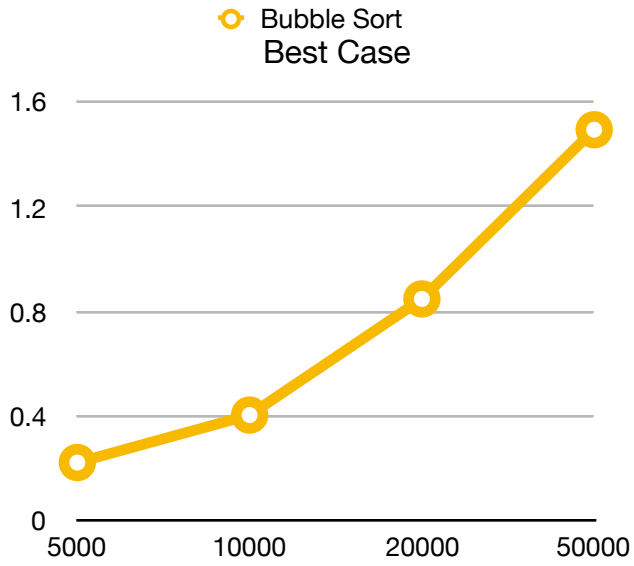

Merge Sort
Worst Case
5
3.75
2.5
1.25
0
5000
10000
20000
50000


Merge Sort
Average Case
4
3
2
1
0
5000
10000
20000
50000


Merge Sort
Best Case
4
3
2
1
0
5000
10000
20000
50000


Bubble Sort
Worst Case
1400
1050
700
350
0
5000
10000
20000
50000


Bubble Sort
Average Case
4000
3000
2000
1000
0
5000
10000
20000
50000


Bubble Sort
Best Case
1.6
1.2
0.8
0.4
0
5000
10000
20000
50000

As can be seen from the final graphs above, the graphs for the Merge Sort are consistent for all three graphs. However the Bubble sort is quadratic for any case other than the best case. In the best case, the graph closely resembles a linear function more than it does a quadratic function.

Conclusion

From the program results and data, it is obvious that Merge Sort outperforms Bubble Sort. This has provided me a good demonstration of the effects of pipelining and branch prediction. It has also provided a better understanding of how significant and important time complexity is for programming efficiently. Since I had searched the algorithms online to use as a reference, I have realized how uncommonly used Bubble Sort is since every example I had found online only had the code that caused the best case time complexity to be $\Omega(n^2)$. In which I had to edit in order for the sorting algorithm to function as intended.

References

[1] “Merge Sort.” *GeeksforGeeks*, 8 Mar. 2018, www.geeksforgeeks.org/merge-sort/.

[2] “Bubble Sort.” *GeeksforGeeks*, 20 Mar. 2018, www.geeksforgeeks.org/bubble-sort/.