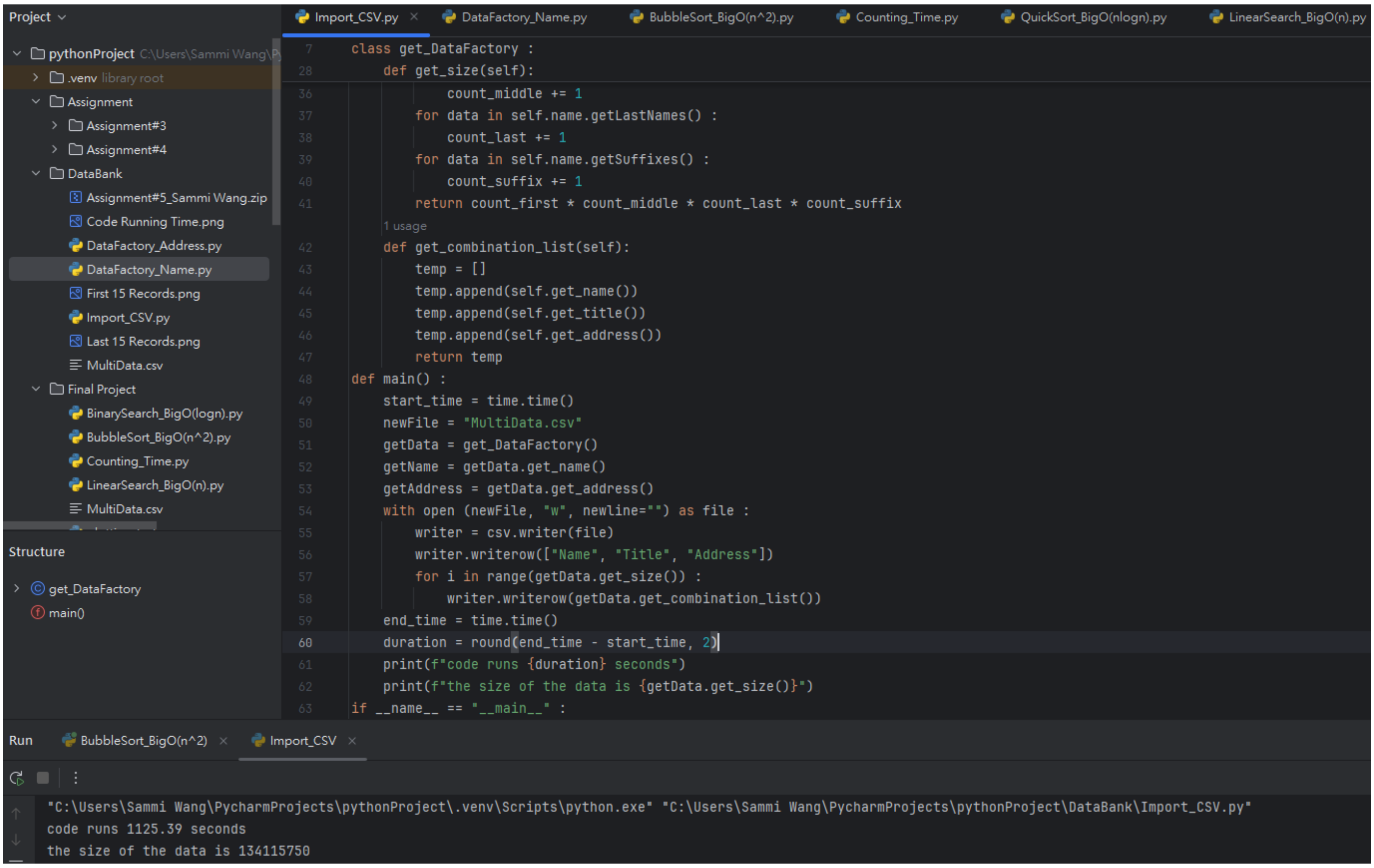
Final Project

a. Asymptotic datasets used for this final project:

Since we already have an assignment to create our own large datasets, I increased the size of the dataset from 2 million to 100 million so that the time taken by the four algorithms may vary more significantly.
From the snapshot below, I took over 1000 seconds to create a bigger datasets. (the csv file size is over 9GB for storing such big datasets…)
However, I encountered some time issue during this final project, so I eventually reduce the greatest size of datasets to 20 million.



b. Algorithms chose for the final project:

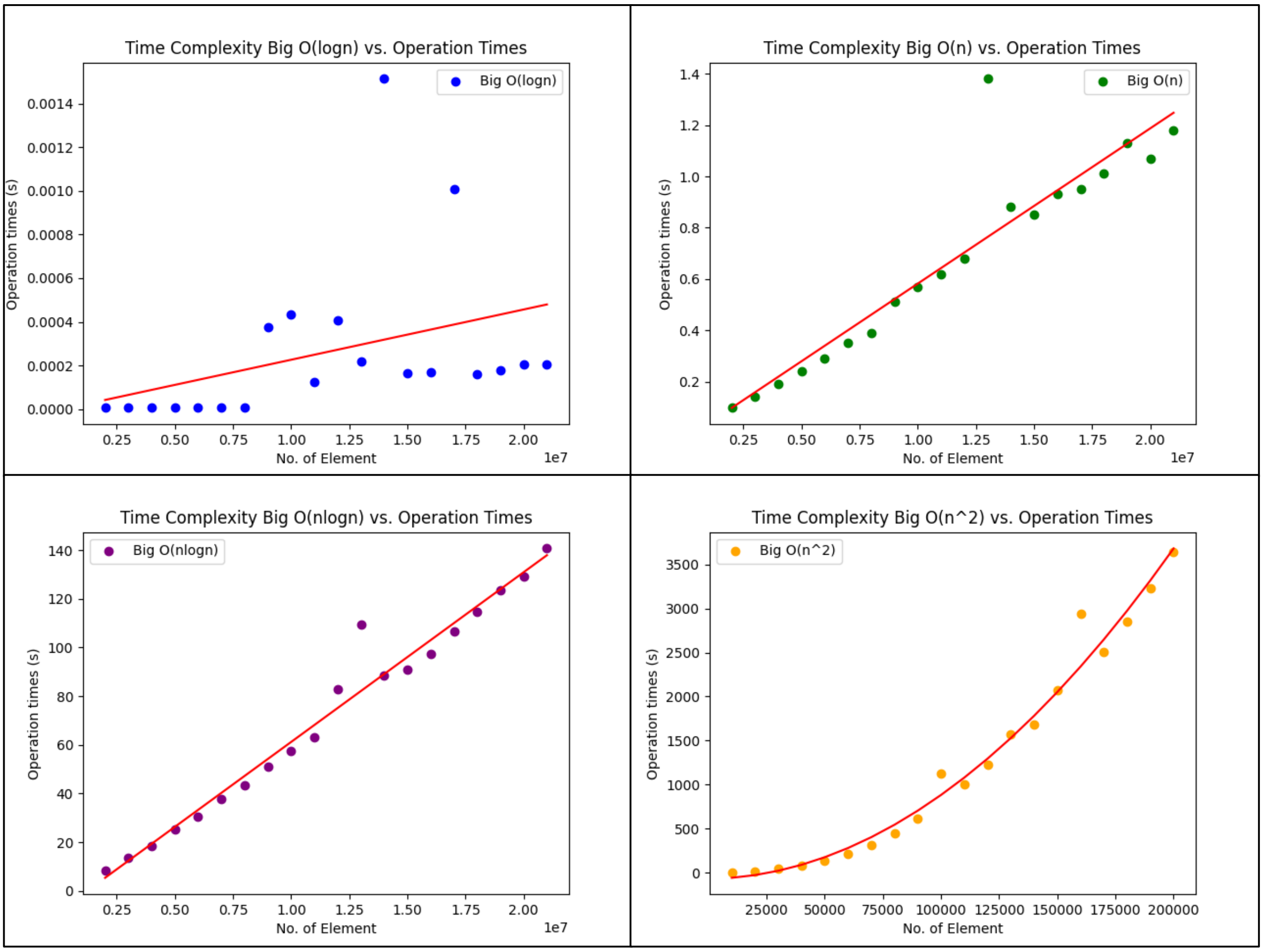
My datasets are essentially composed of three components: name, job title, and address. I couldn't think of a better way to organize or structure such a 'somewhat complex dataset,' so I opted for a simple and basic data structure, which is a list. As for the algorithms, I didn't want to complicate things, so I chose the simplest ones: bubble sort, quicksort, linear search, and binary search. My main goal is to produce the graph first, and if time allows, I will work on implementing more complex algorithms.

1. Big O(n^2): Bubble Sort
2. Big O(nlogn): Quick Sort
3. Big O(n): Linear Search
4. Big O(logn): Binary Search

c. Results (Output)

The graph below shows the operation times versus the same size datasets using different algorithms with various time complexities (except for Big O(n^2)). I used datasets from assignment #5 and increased the size from 2 million to 20 million to magnify the time differences between the algorithms. From the results, I found that Big O(log n) is much faster than the other three algorithms. As for Big O(n), it is quite fast compared to

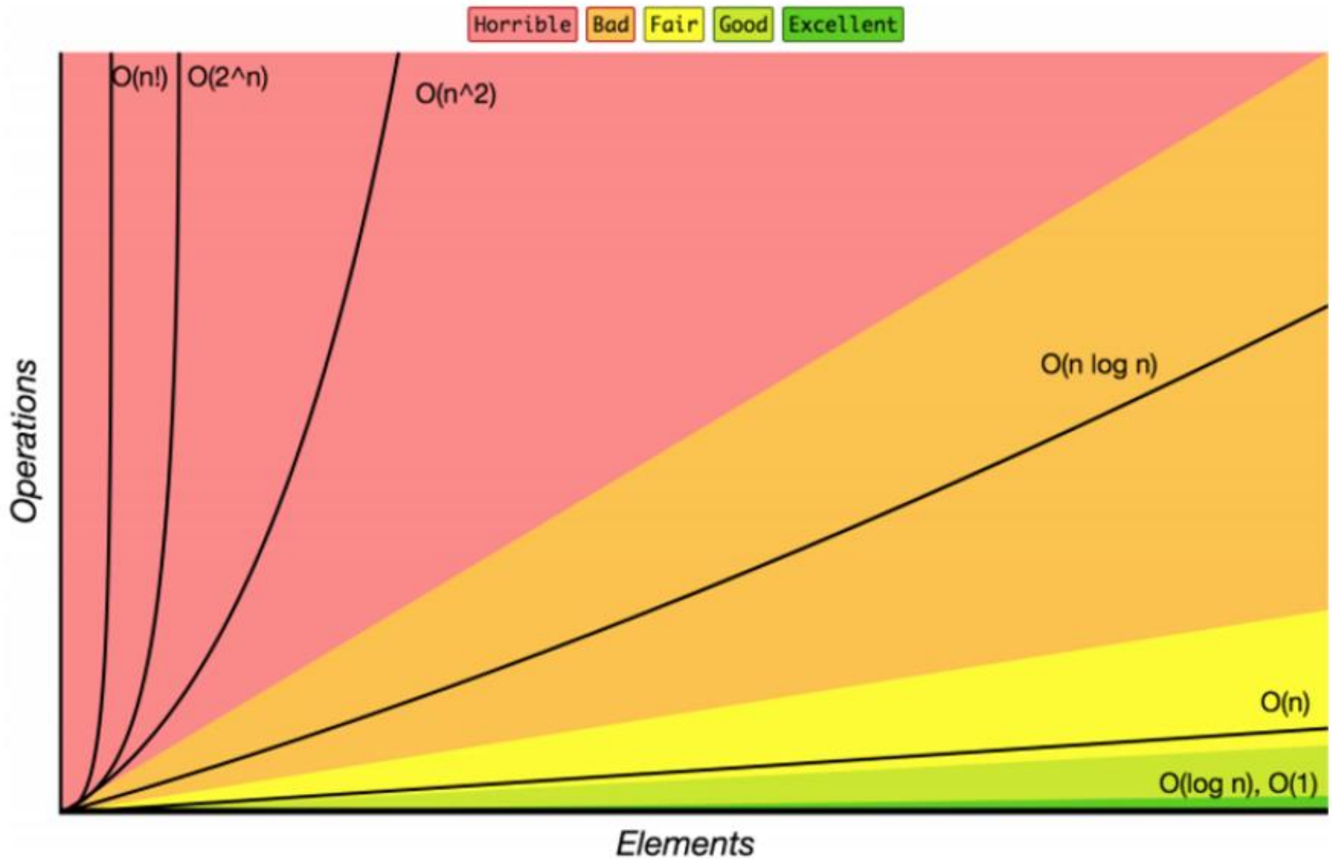
O(n log n). Additionally, I observed that Big O(n log n) fits the linear pattern the most. Lastly, processing Big O(n^2) took a significant amount of time (almost an entire night). Initially, I used the same dataset size for O(n^2), but it was impractical since it took forever and the code never completed. Therefore, I reduced the size from 20 million to 200,000, and my code finally finished.

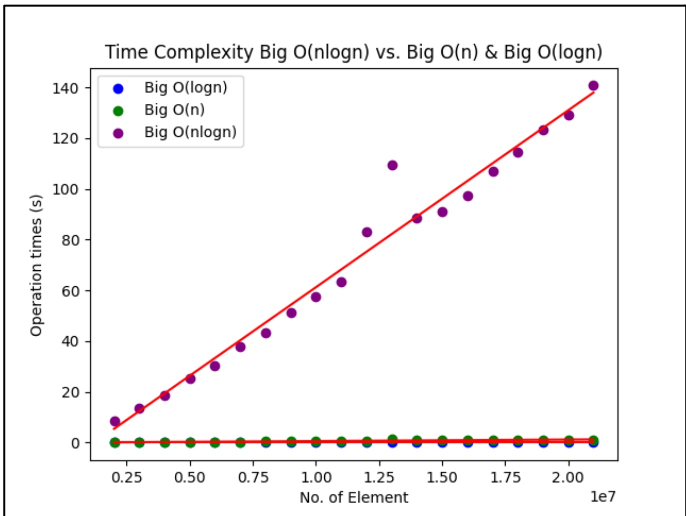


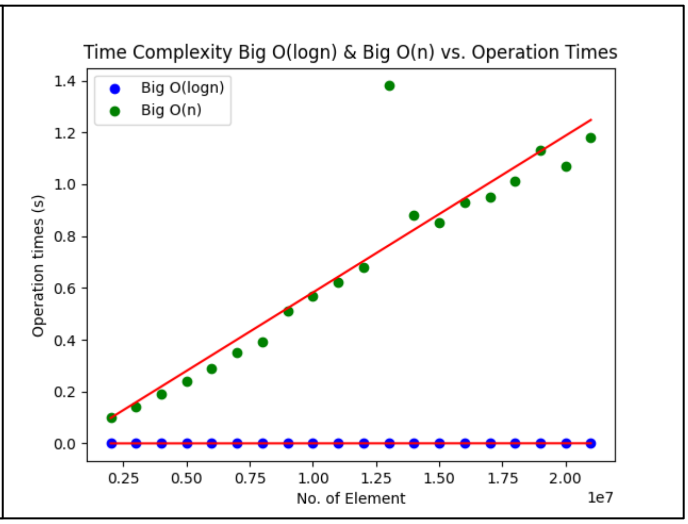
d. Additional Results (Comparison)

Since the behavior of O(log n), O(n), and O(n log n) nearly fits a linear pattern, I wanted to examine the time differences between them. From the graph on the right, I found that it's quite clear the code almost completed instantly for O(log n). However, it still took a couple of seconds for O(n) to process. The time difference became even more dramatic when I added the regression line and scatter points for O(n log n) to the graph. The behavior of O(log n) and O(n) almost forms a horizontal line, while the time for O(n log n) is nearly 100 times greater than O(n).

Compared to the sample graph provided by the professor, I still cannot distinguish much difference between O(n) and O(log n) in my graph, as they are very close. On the other hand, the behavior of O(n log n) and O(n^2) seems more reasonable to me. I was wondering if there is an error in my code, but after checking it several times, I couldn't find any issues.


Horrible
Bad
Fair
Good
Excellent
O(n!)
O(2^n)
O(n^2)
O(n log n)
O(n)
O(log n), O(1)
Operations
Elements


Time Complexity Big O(nlogn) vs. Big O(n) & Big O(logn)
Big O(logn)
Big O(n)
Big O(nlogn)
Operation times (s)
No. of Element
1e7


Time Complexity Big O(logn) & Big O(n) vs. Operation Times
Big O(logn)
Big O(n)
Operation times (s)
No. of Element
1e7