Data Structures - Final Project

Ryan Fisher

For this Final Project, we were tasked to find a data structure algorithm that would speed up the insertion and searching times for the data of shipments in the USPS Tracking System. I implemented several data structures that had quite conclusive results. First and foremost, according to the write-up, the USPS has been utilizing a Linked List sorting algorithm, and that this implementation is inefficient time-wise. After implementing my own Singly Linked List to insert and search for various large number data sets, I discovered that while the insertion time of this algorithm is relatively instantaneous with a complexity of O(1). However, the search times were quite the opposite. With a complexity of O(n), you can observe from *Figure 1*, that as more elements were inserted into the Linked List, the time it took to search for those elements rapidly rises. This is inefficient for the USPS during high demand times. I then implemented a BST, (*see Figure 3 and 4*) to test its performance, and found that it was much more applicable for the USPS than the Linked List was. While the insertion and searching times for the BST, with complexity O(log(n)) were slightly greater than the Linked List Insertion time, overall, this is a great improvement from the Linked List searching times. I then tested out multiple hash tables: Linear Probing(*Figure 5,6*), Quadratic Probing(*Figure 7,8*), and with Chaining(*Figure 9,10*). The Linear and Quadratic Probing timing results were relatively similar throughout most of the data set, however, when the load factor gets closer to 1, the timing on the Linear Probing was almost double that of the Quadratic Probing algorithm. This is due to Linear Probings' problem with clustering, meaning that the elements begin to form groups within the table and it will take longer to search for an empty "bucket". Finally, it came down to the Chained Hash Tables, and it's results look promising. The main difference between open addressing and chaining hash tables, are that the open addressed tables tend to be a bit faster with lower load factors, but slow down substantially with high load factors. With Chaining, the graph shows that even with a high load factor, the search and insert speeds are very optimal. I then plotted the the search and insertion results from the Linked List, BST, and Chained Hash table on a summary plot(*Figures 11-14)* and it was quite clear that the Chained Hash Table out-performed the others in this case. Thus, with these testing results, I have concluded that with USPS being under high volume lately, a Chained Hash Table would be the best option to implement.

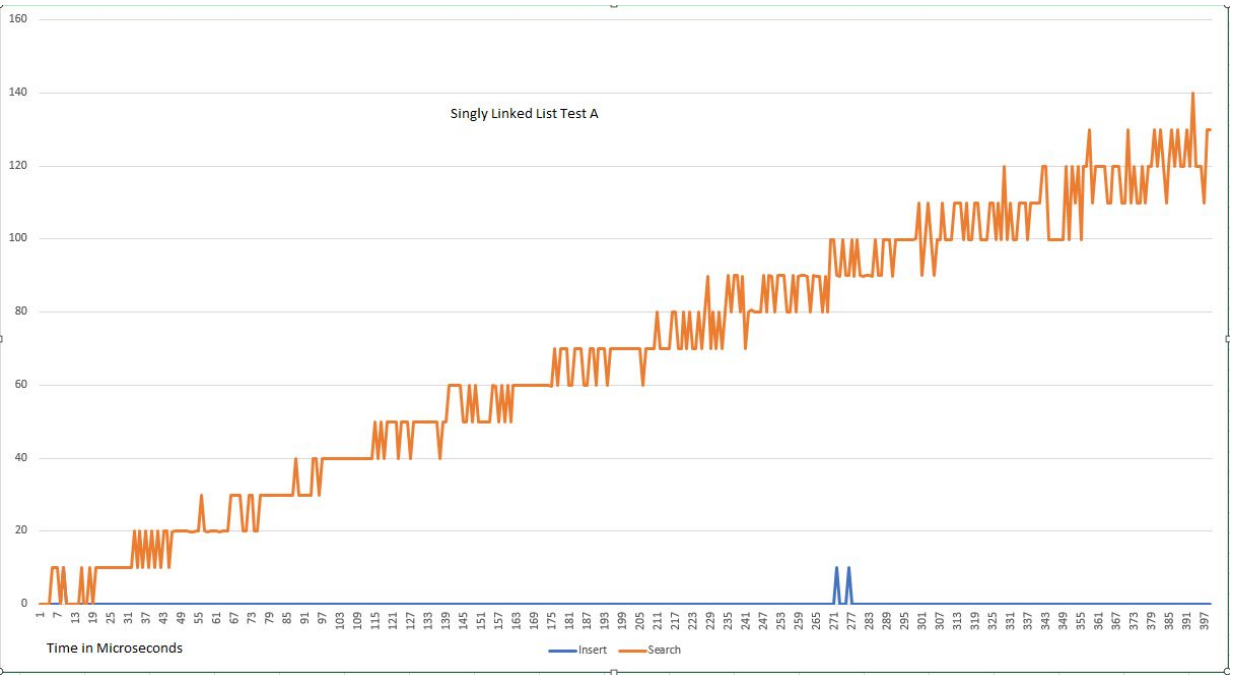


Figure 1: Singly Linked List Test A

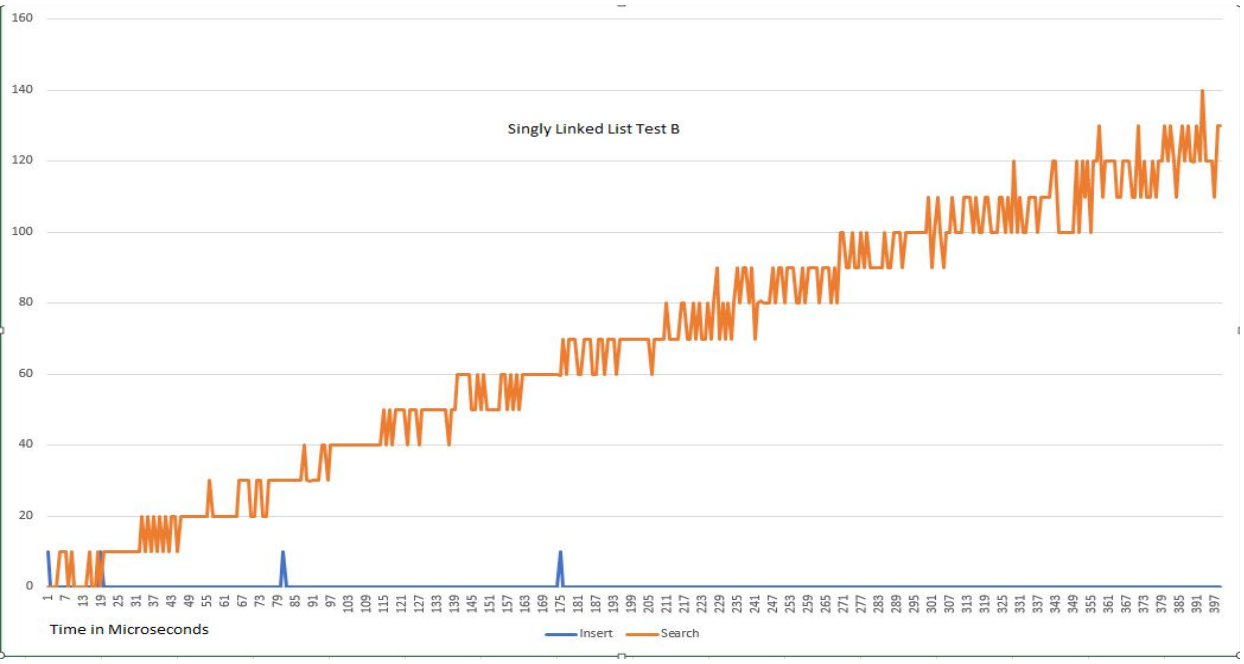


Figure 2: Singly Linked List Test B

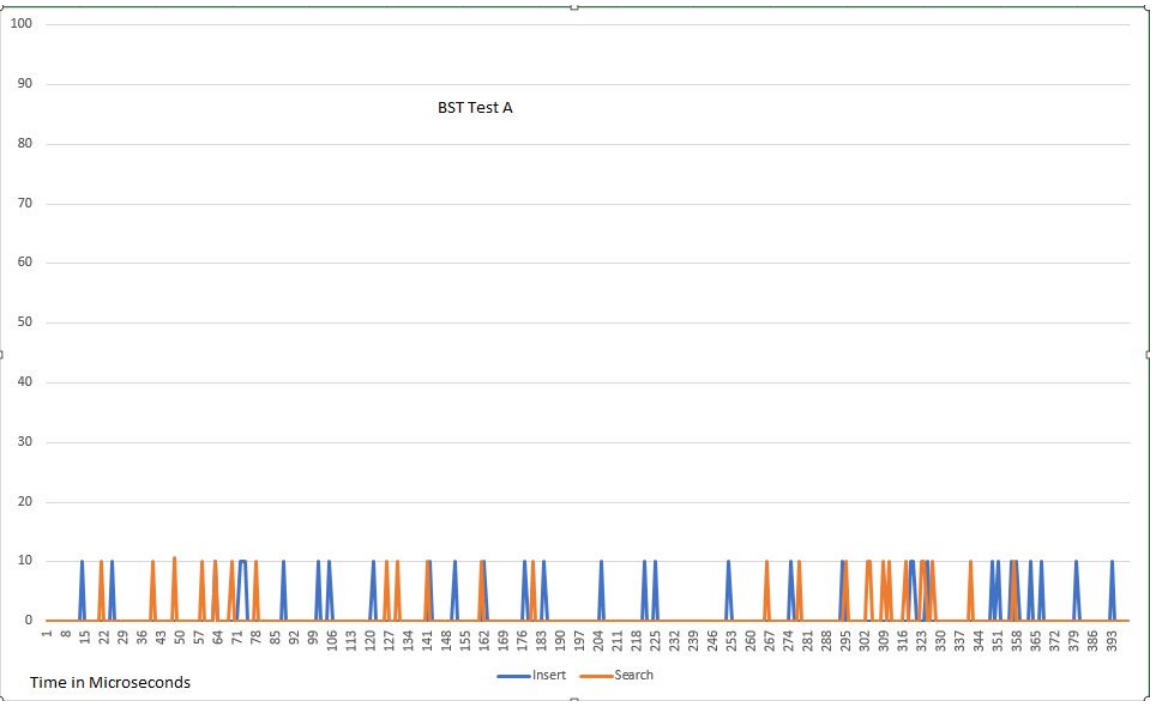


Figure 3: BST Test A

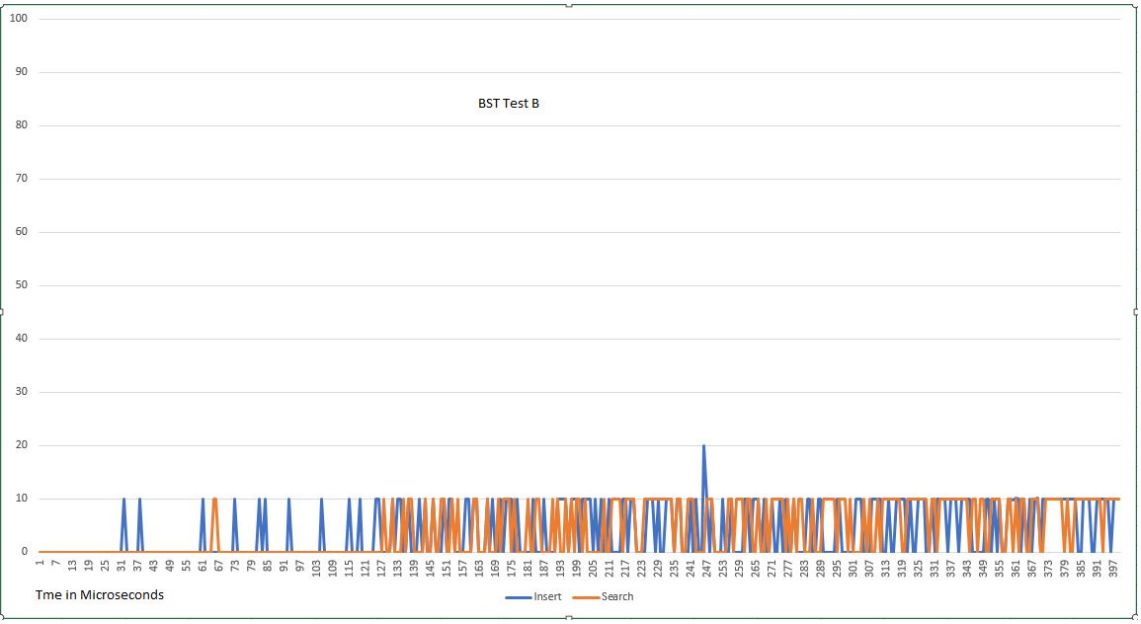


Figure 4: BST Test B

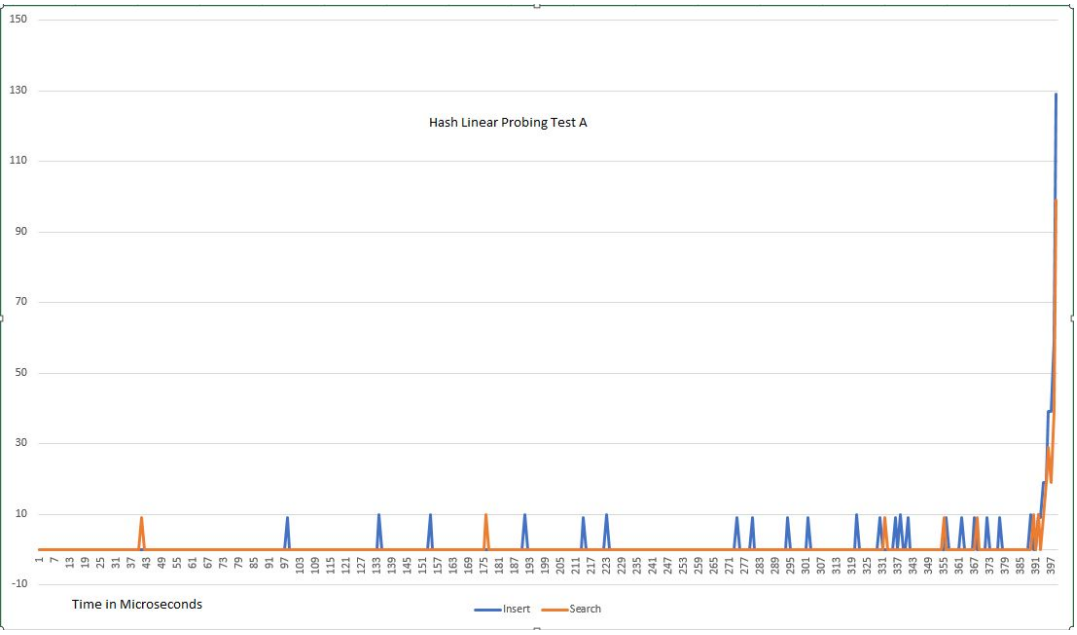


Figure 5: Hash Linear Probing Test A

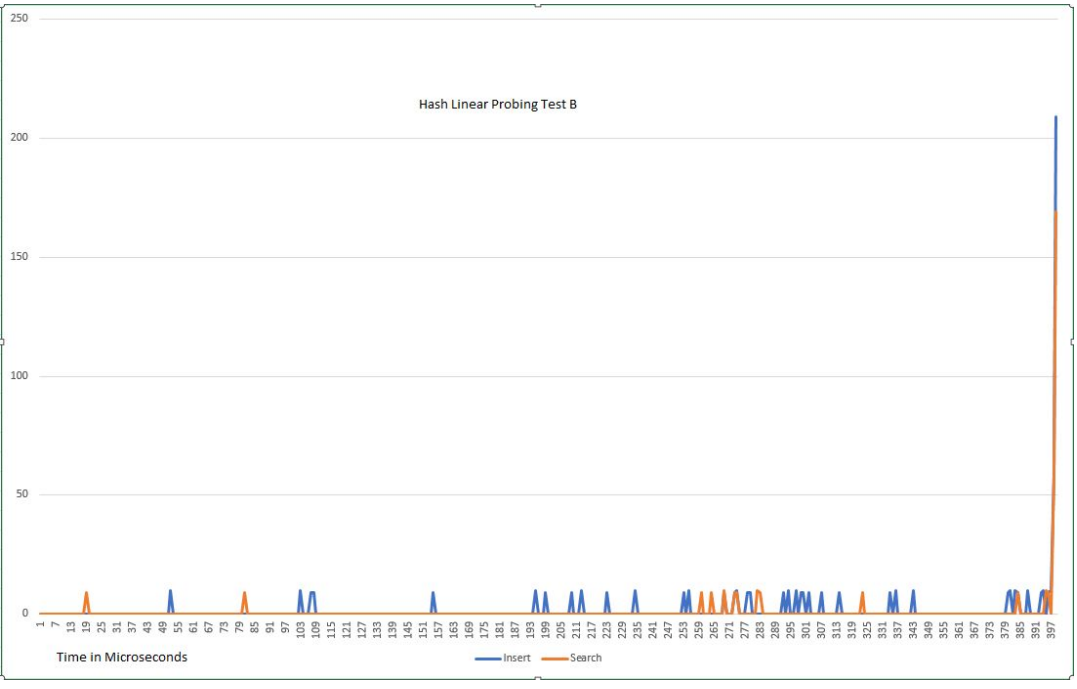


Figure 6: Hash Linear Probing Test B

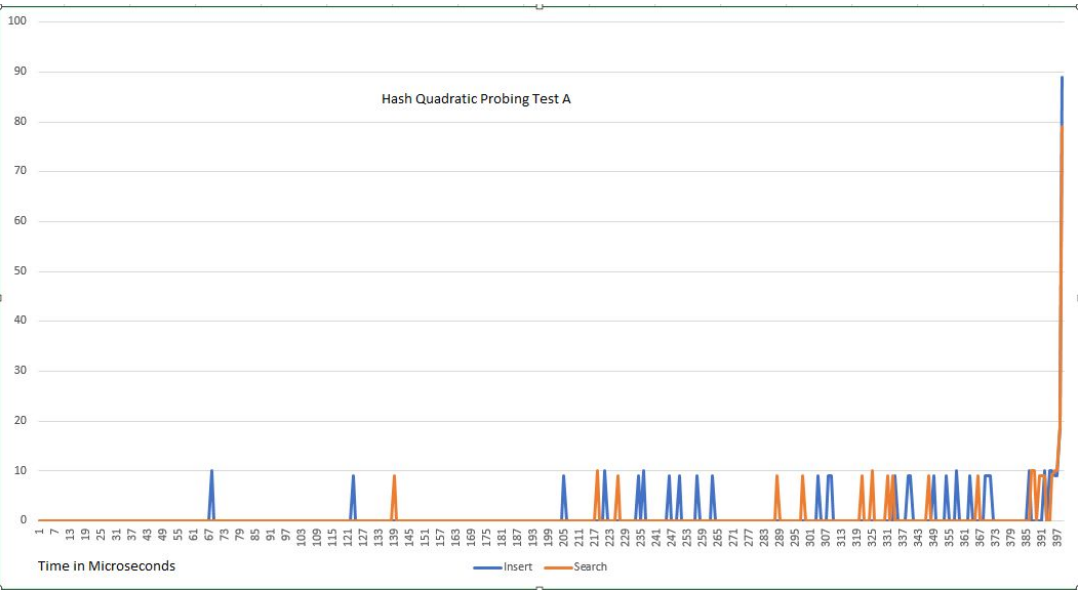


Figure 7: Hash Quadratic Probing Test A

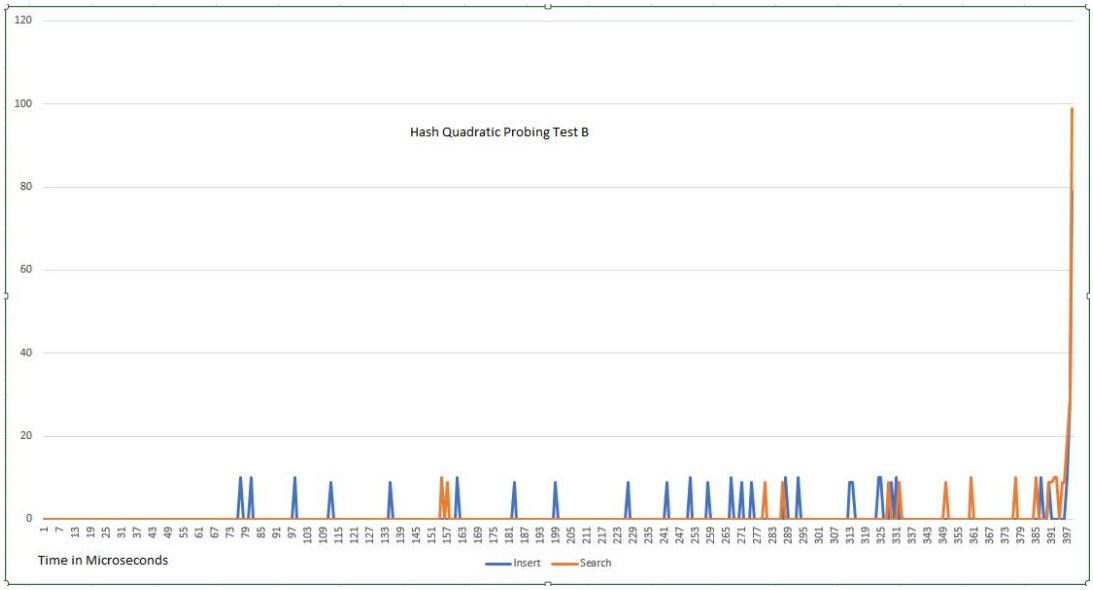


Figure 8: Hash Quadratic Probing Test B

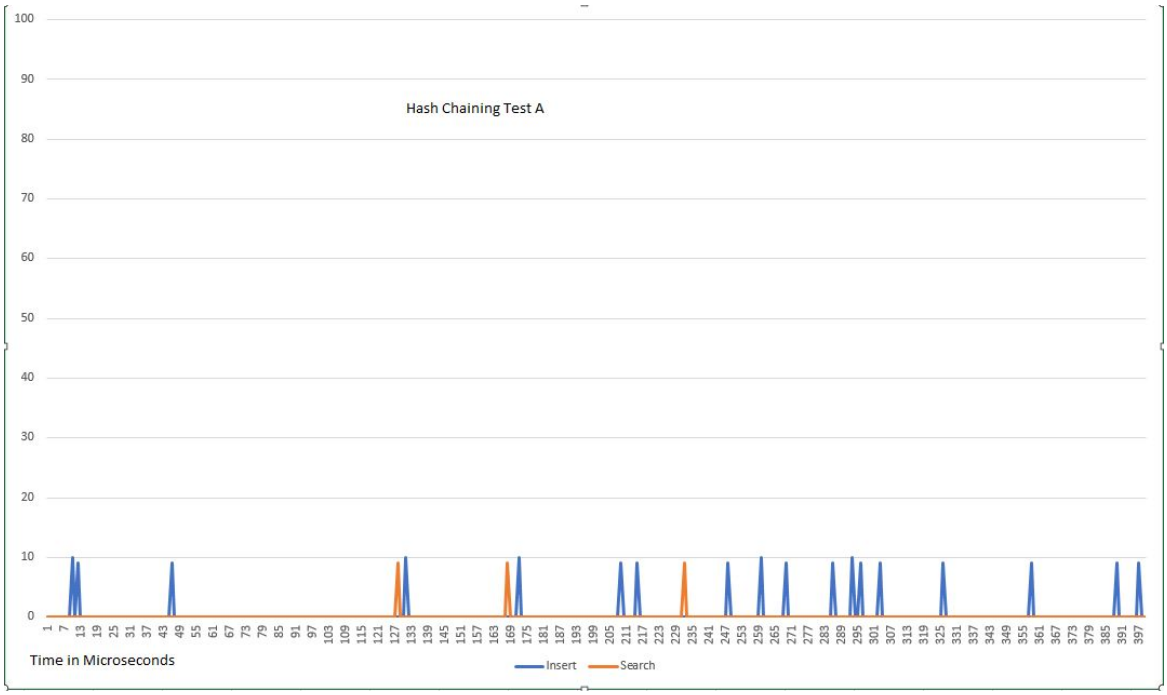


Figure 9: Hash Chaining Test A

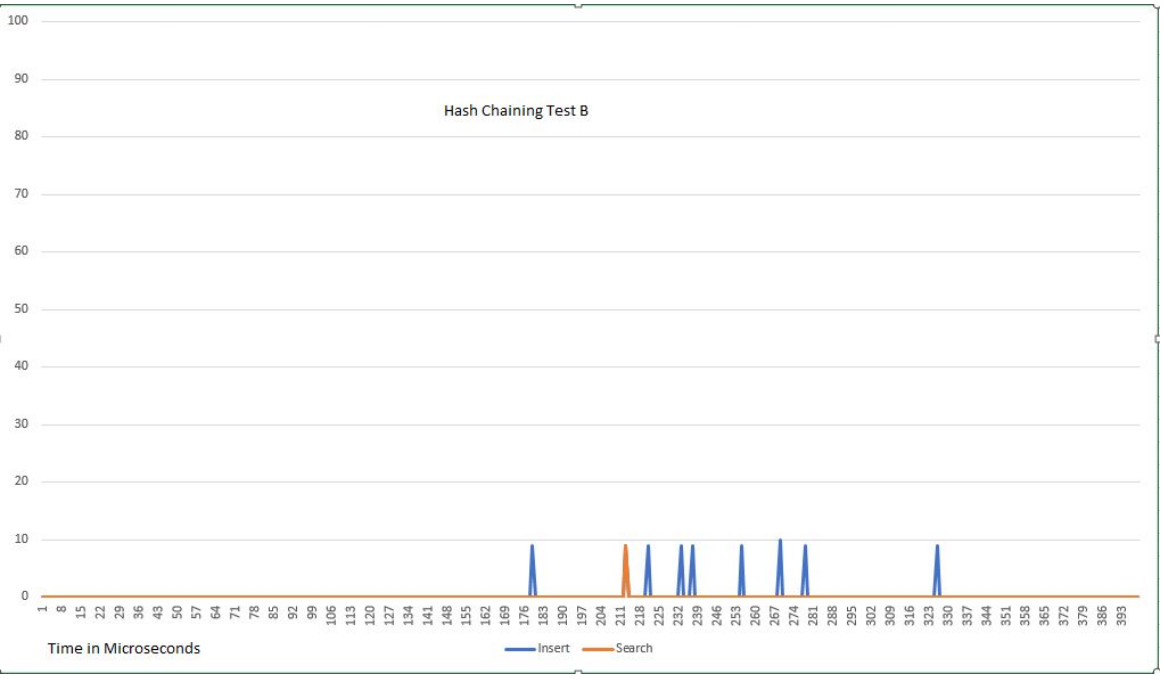


Figure 10 Hash Chaining Test B

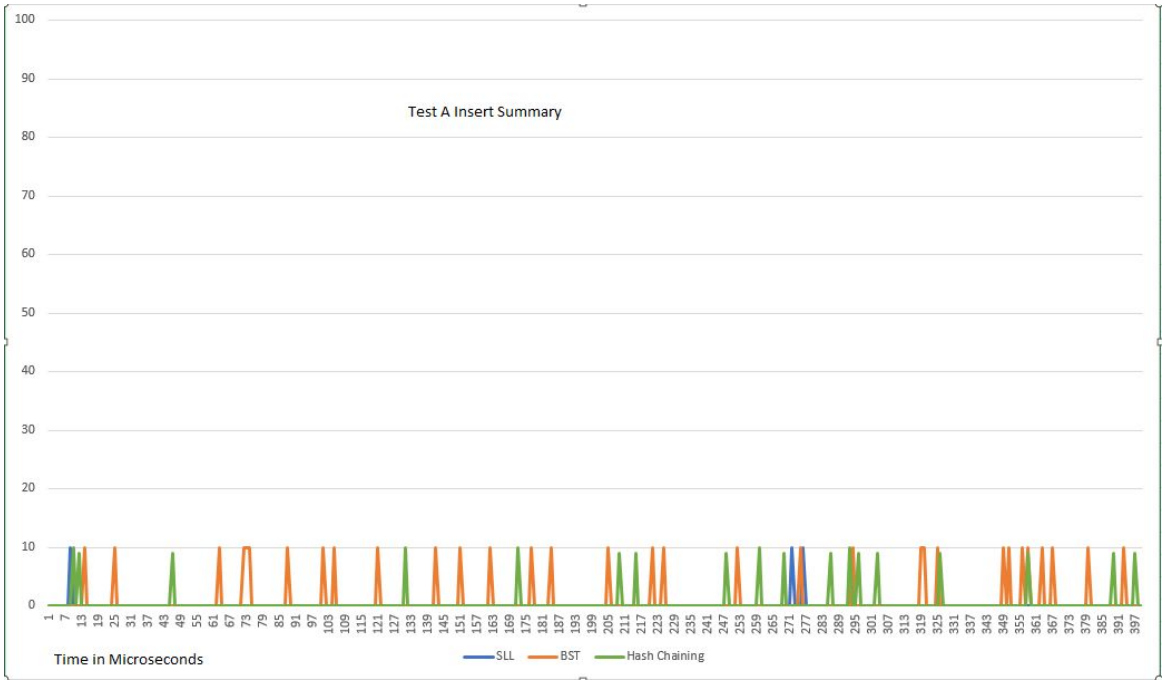


Figure 11: Test A Insert Summary

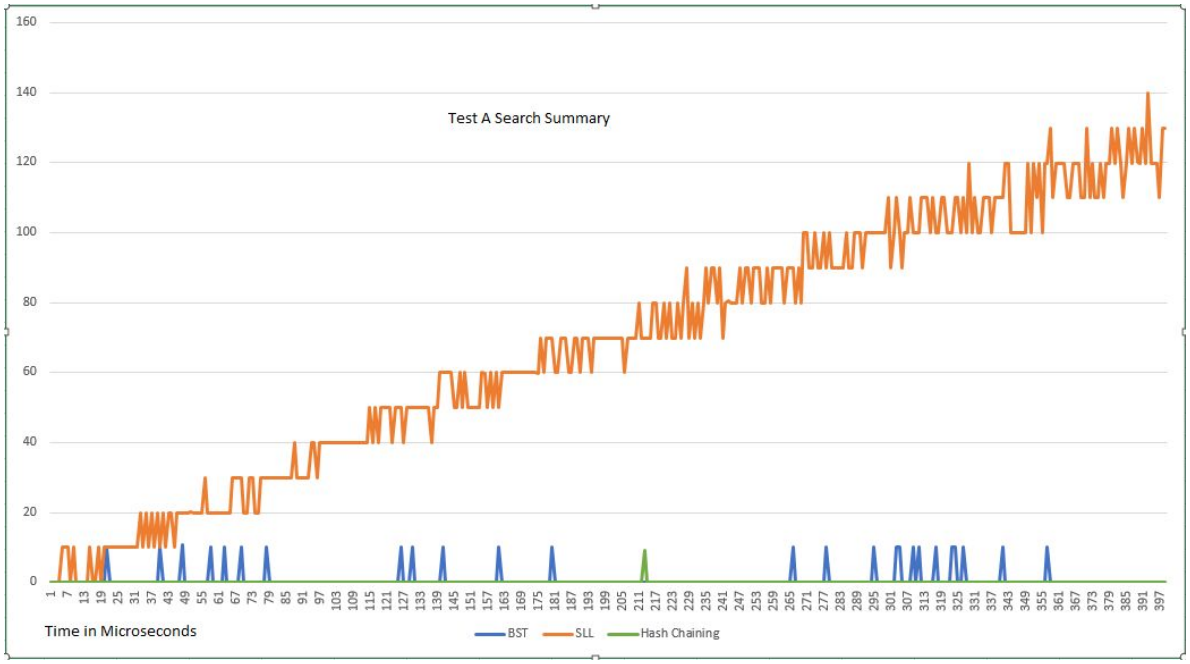


Figure 12: Test A Search Summary

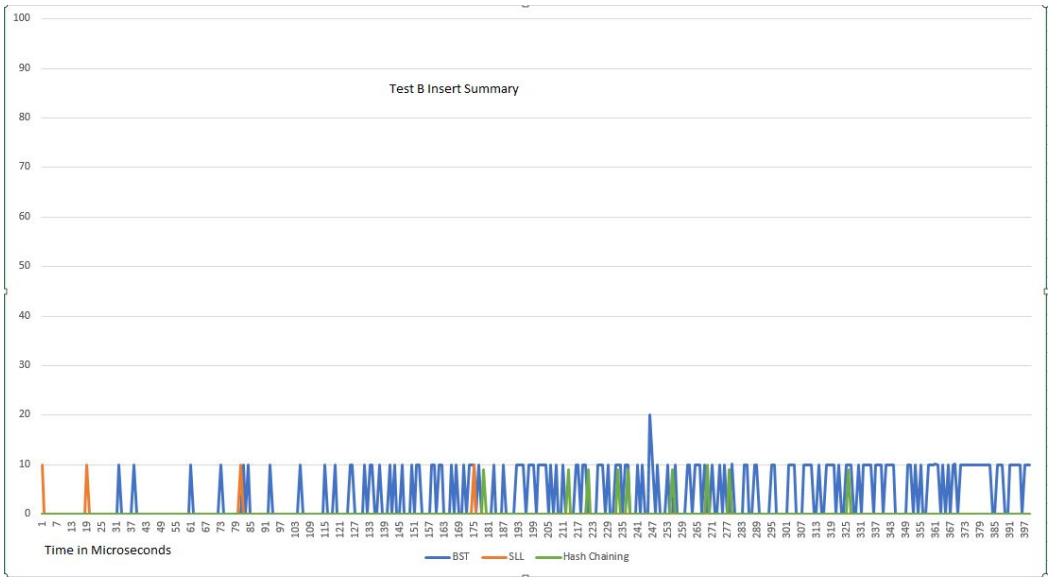


Figure 13: Test B Insert Summary

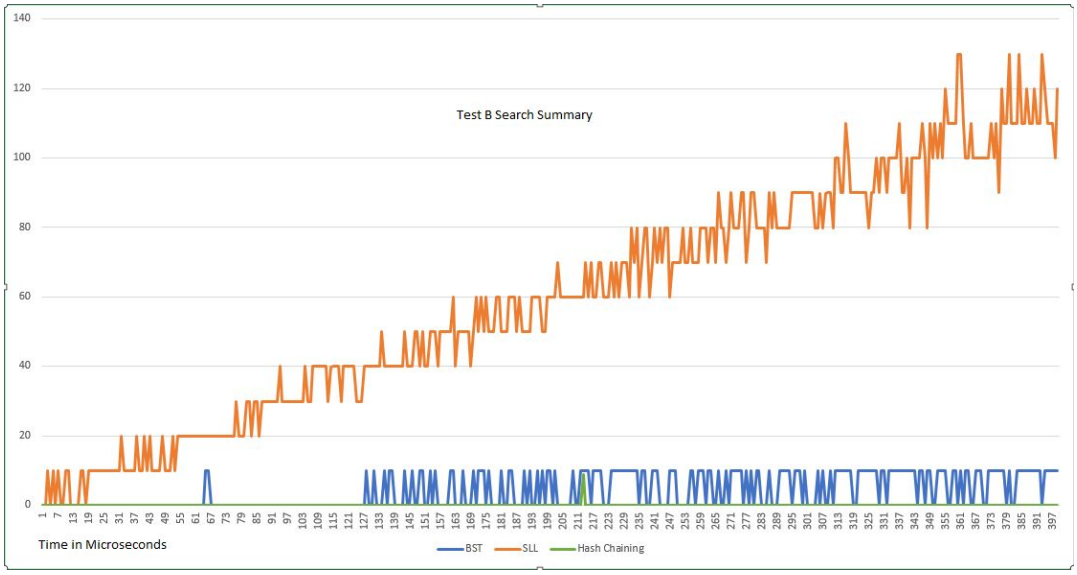
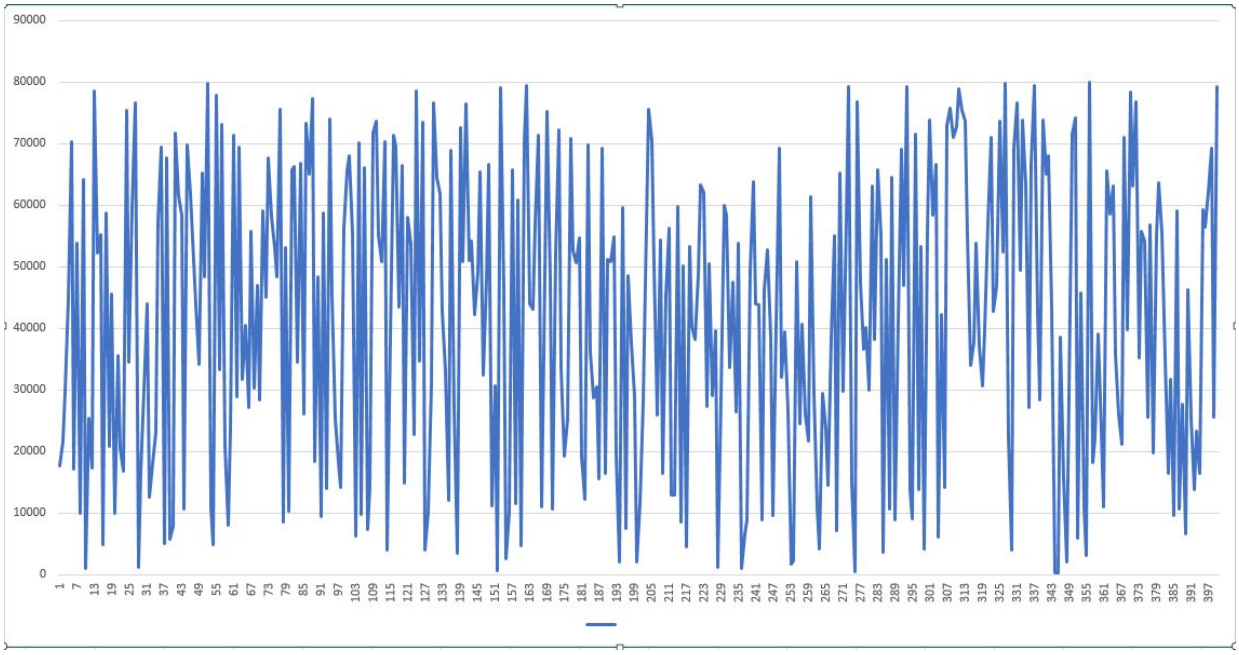
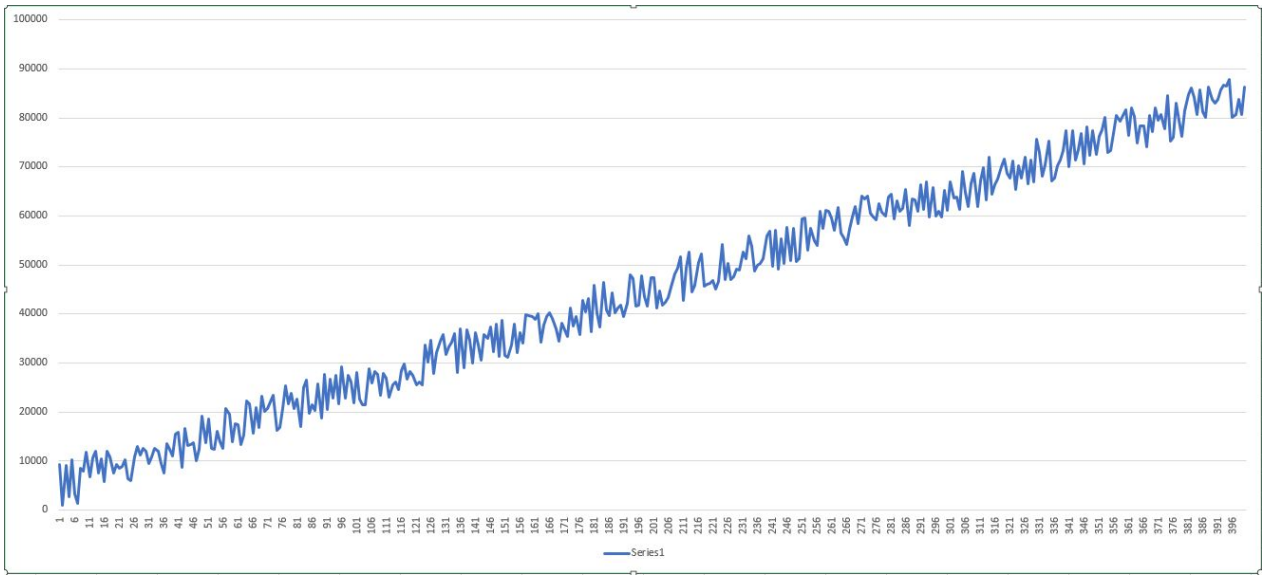


Figure 14: Test B Search Summary



Data A (x-value is # elements in 100’s)



Data B (x-value is # elements in 100’s)