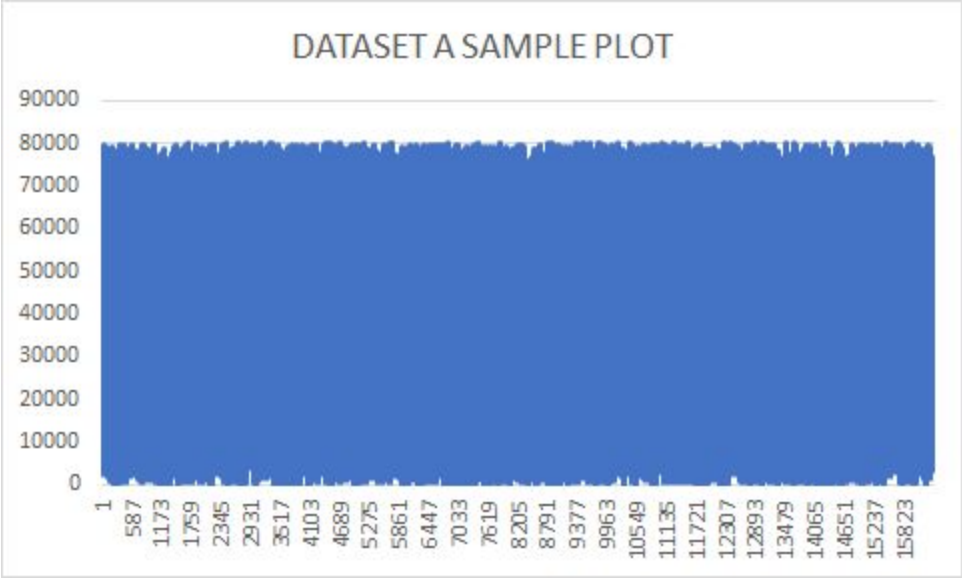
CSCI 2270 Final Project Report

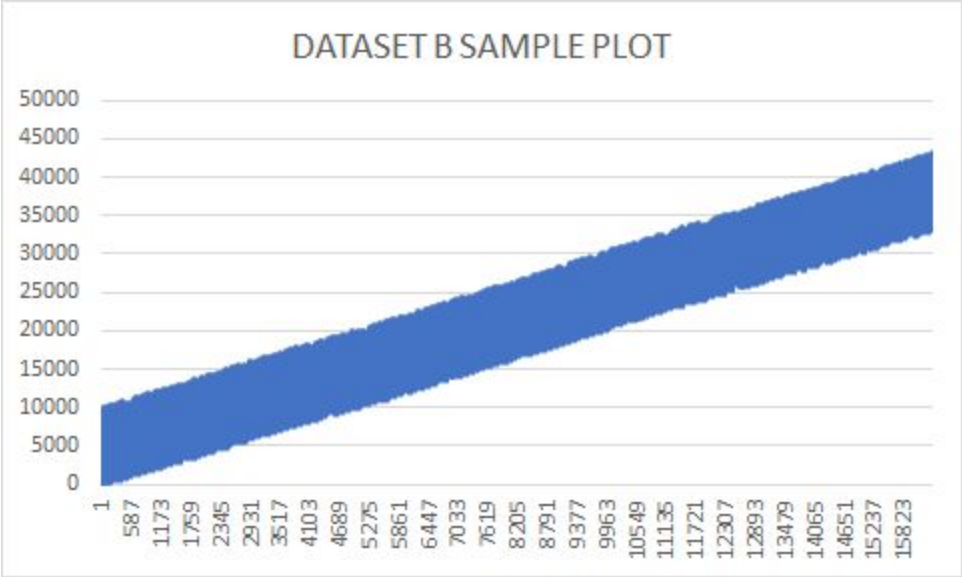
Kent Li and Weige Wong

04/25/2020

1. **Preliminary Analysis:**

The two datasets have differing qualities which might then subsequently affect how efficient our data structure will operate. These graphs show how row versus shipment id correlation in dataset A are randomized while dataset B's shipment data have some sequential ordering in terms of its shipment labels.

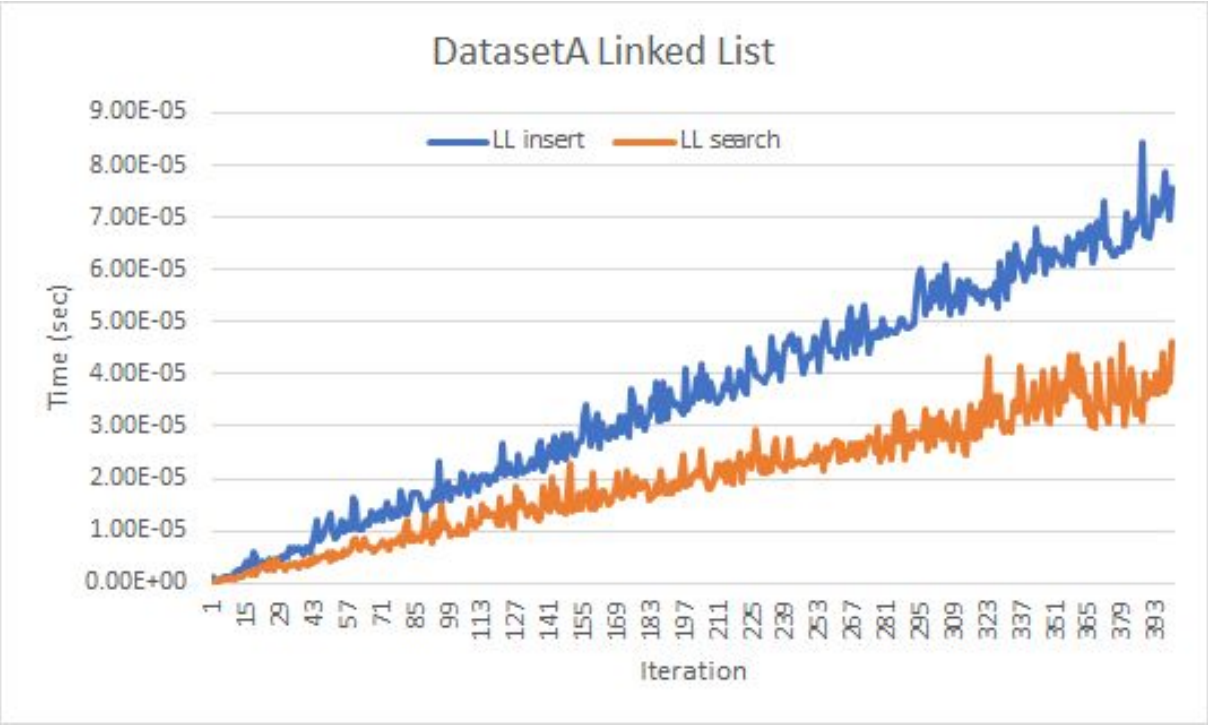


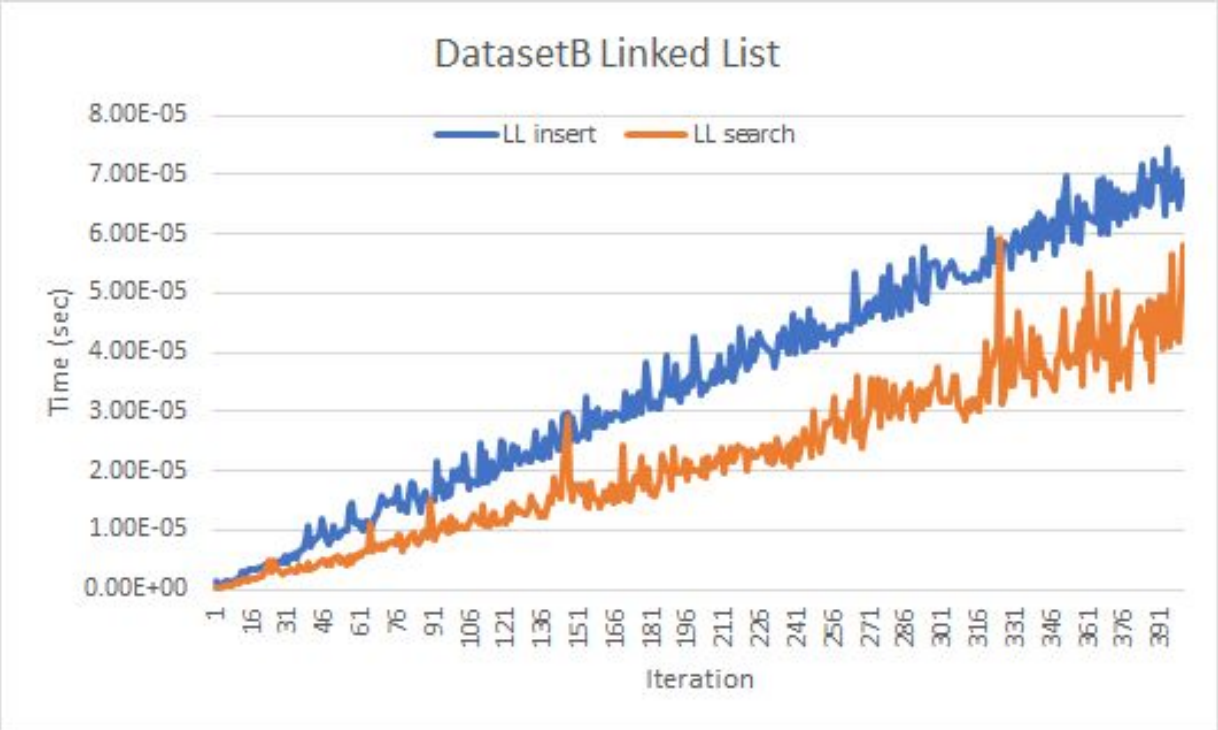


2. Findings and Results

2.1. Linked List:

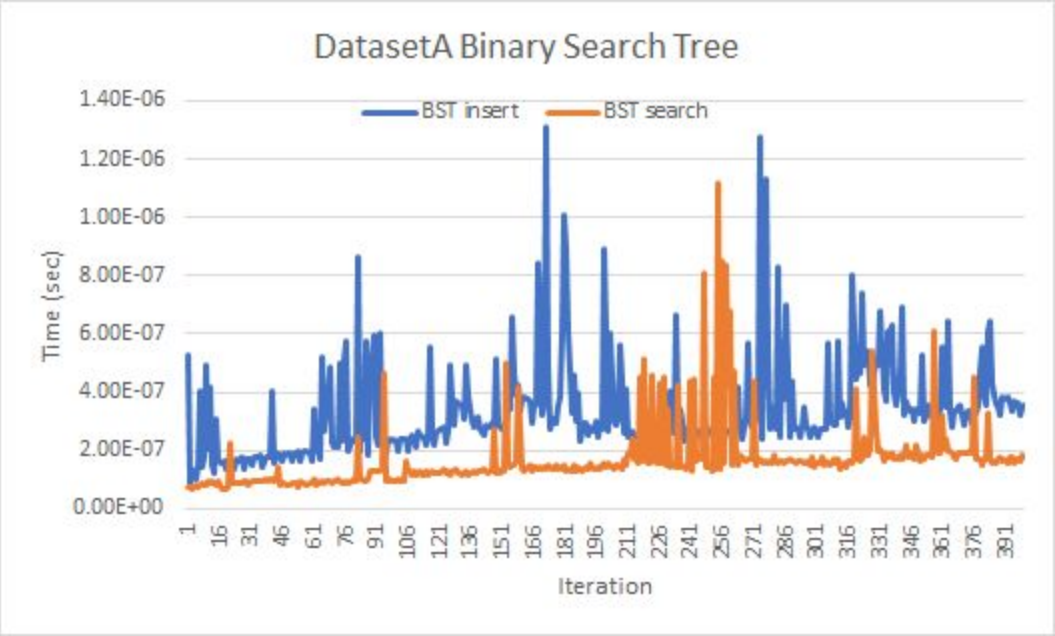
2.1.1. The linked list appears to be slow for both datasets compared to the other data structures, as it's time for insertion and searching grow in a linear fashion for both datasets.

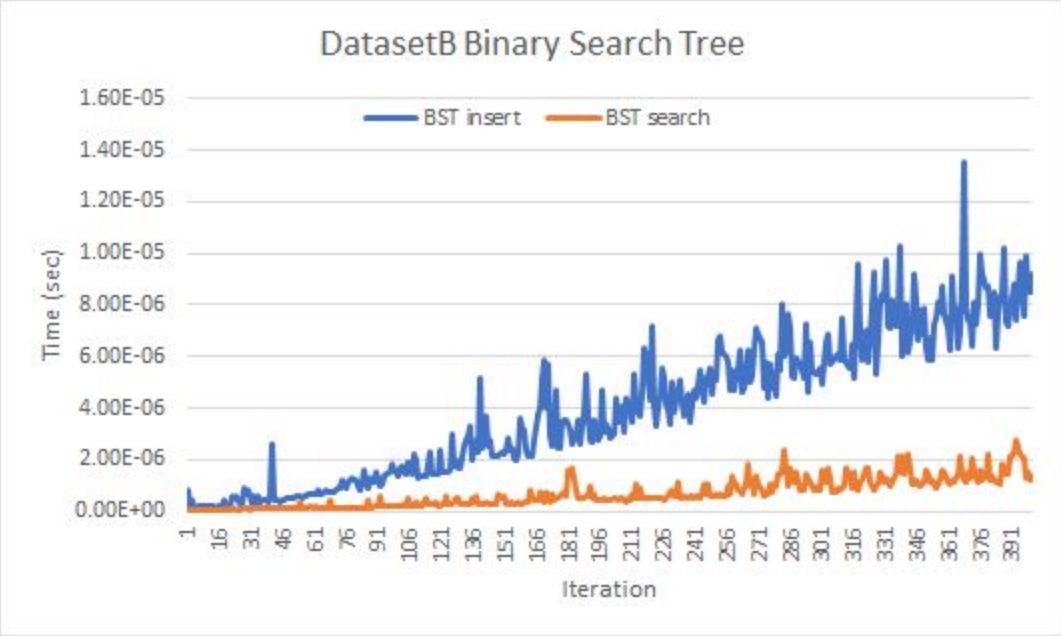




2.2. Binary Search Tree (BST)

2.2.1. The binary search tree is superior to the linked list. The graphs show a massive time improvement. When used on datasetA a logarithmic growth in time can be seen. However, a flaw is that when used on datasets which have ordering (datasetB), the BST reverts to slower, more linear implementation time.

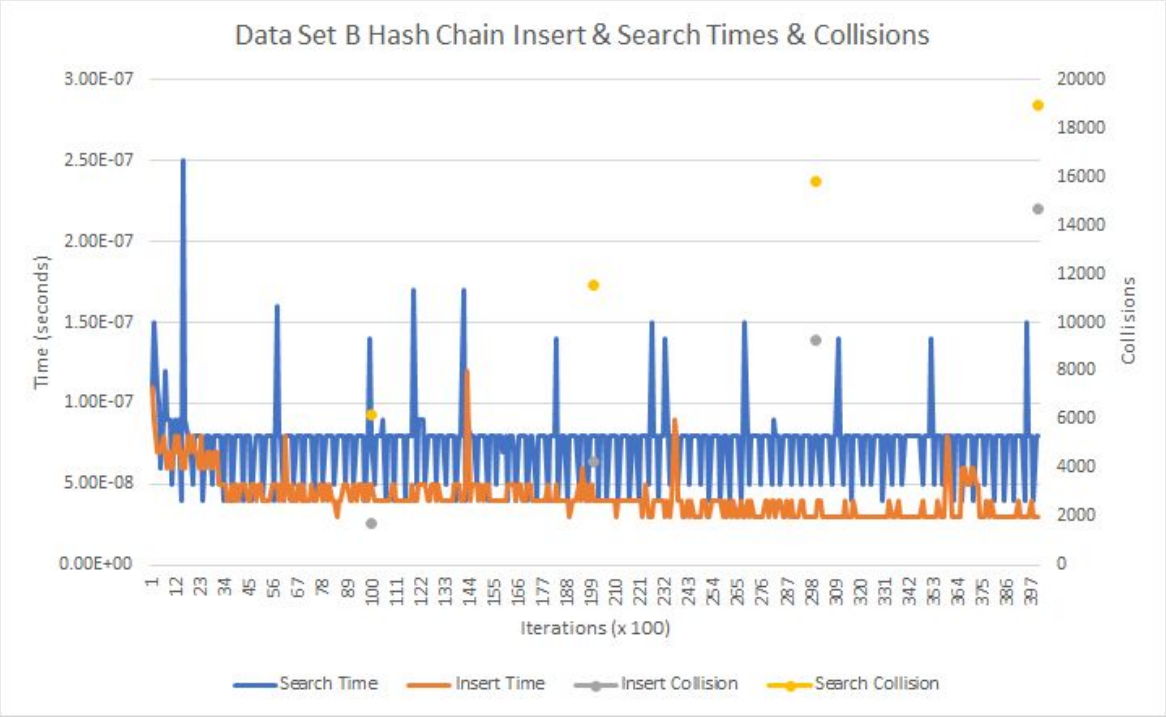


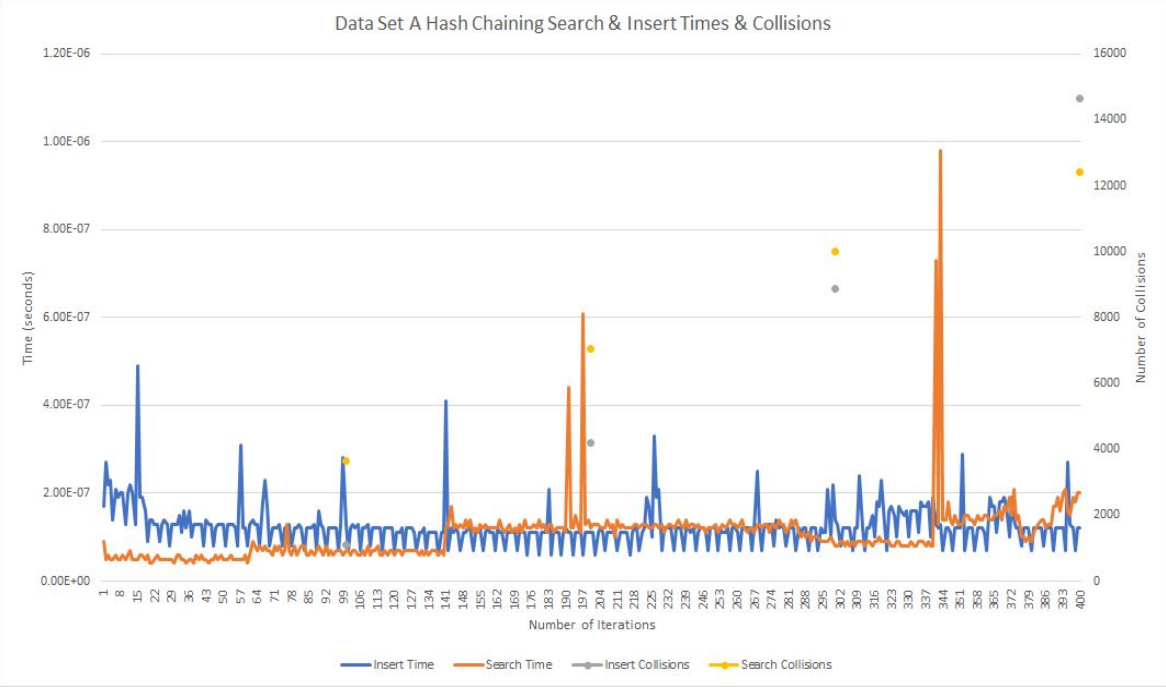


2.3. Hash Table

2.3.1. Chaining

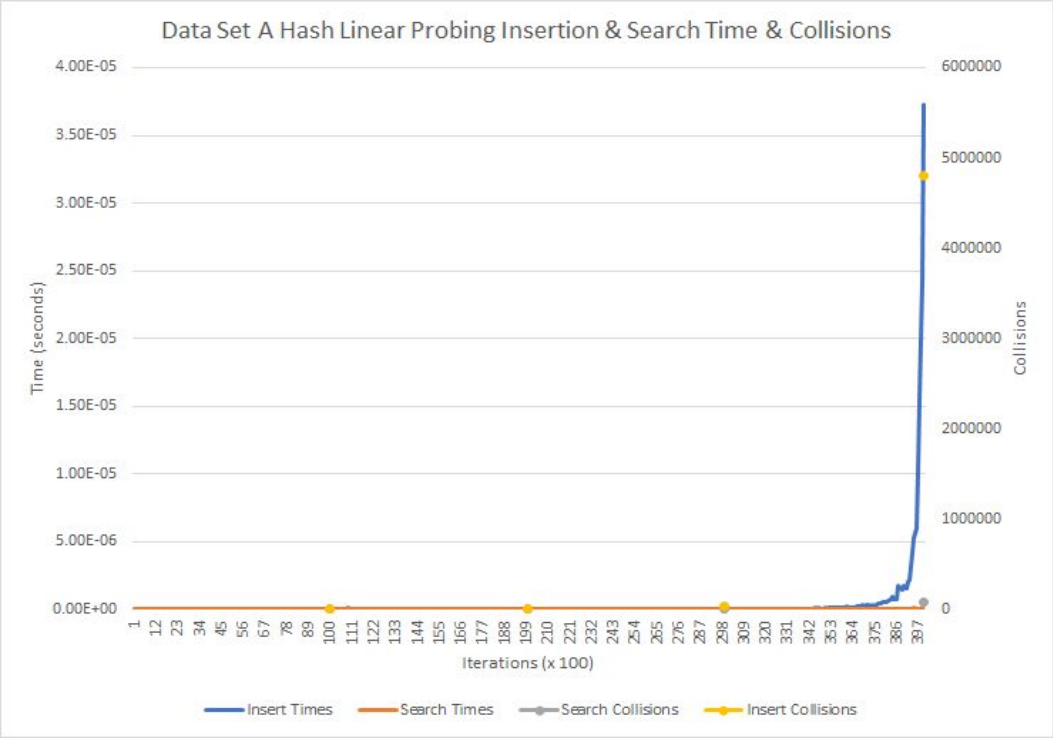
2.3.1.1. The hash table with chaining was amongst the quickest insertion methods for the hash table for insertion and searching. There were seemingly random spikes in time in search and insertion but those could be attributed to the program having to add to a linked list or having to search through an entire linked list.

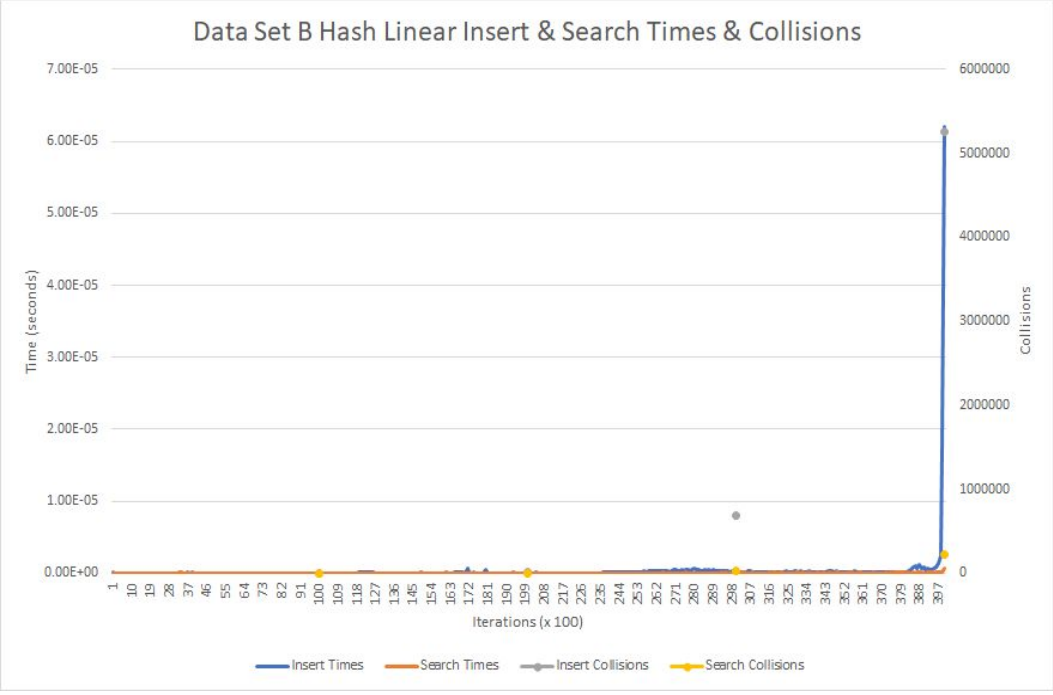




2.3.2. Linear Probing

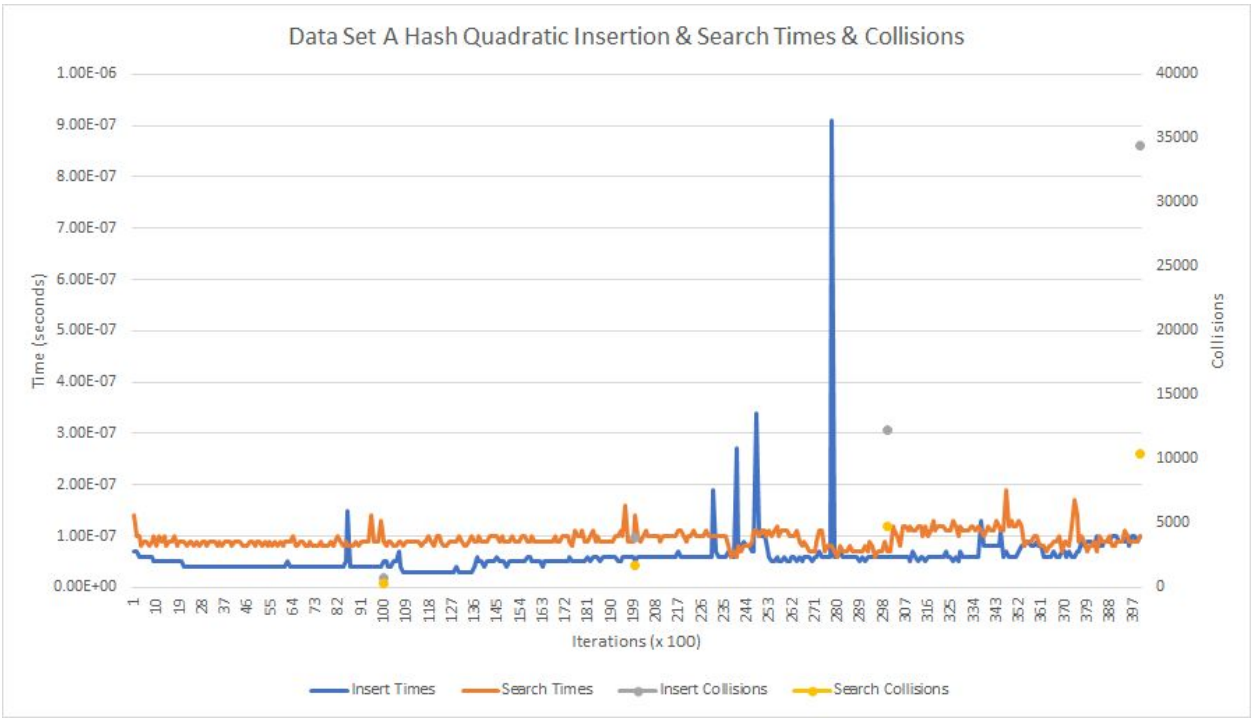
2.3.2.1. Linear probing was the “slowest” of the hashing collision resolutions and even then, it was only because as the table filled up, the program had to search longer and longer for a new space which took more time. The search time was shorter since it’s a random key as opposed to a series of them. This is also reflected in the fact that there are comparatively few insert collisions until the last few inserts.

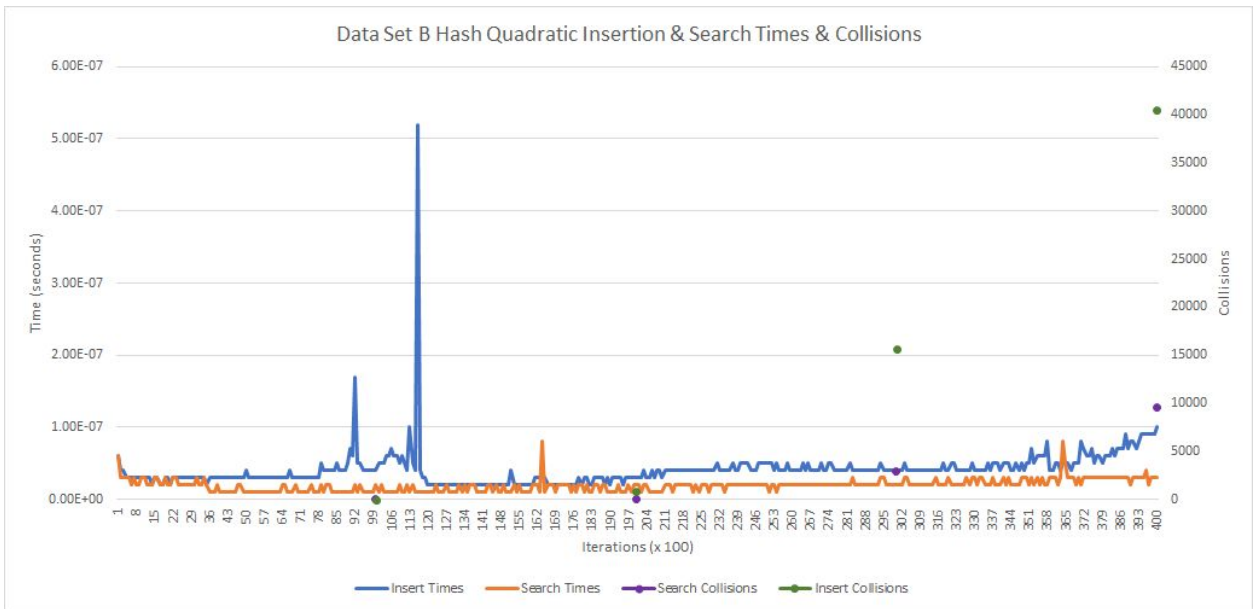




2.3.3. Quadratic Probing

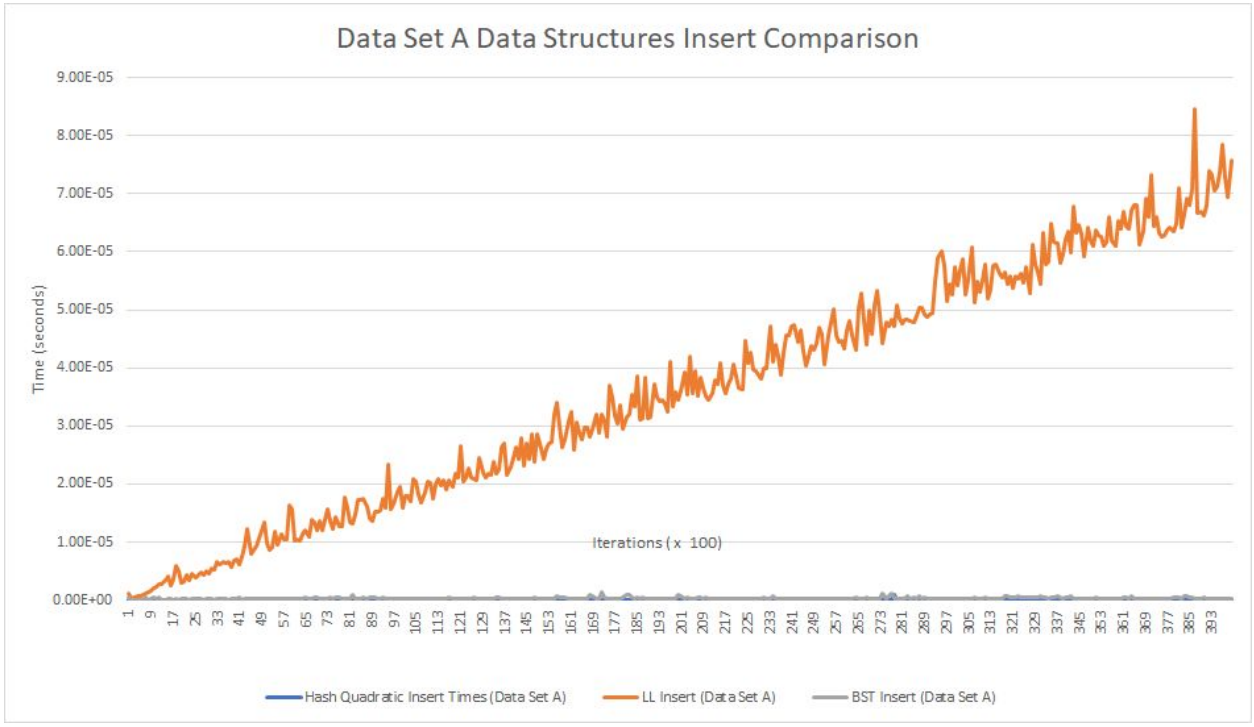
2.3.3.1. The quadratic probing was also similarly quick. The insert and search timings were very close to each other throughout all insertions and searches. There were spikes in time but those may have come from when the quadratic equation had to increment multiple times to find an open position.

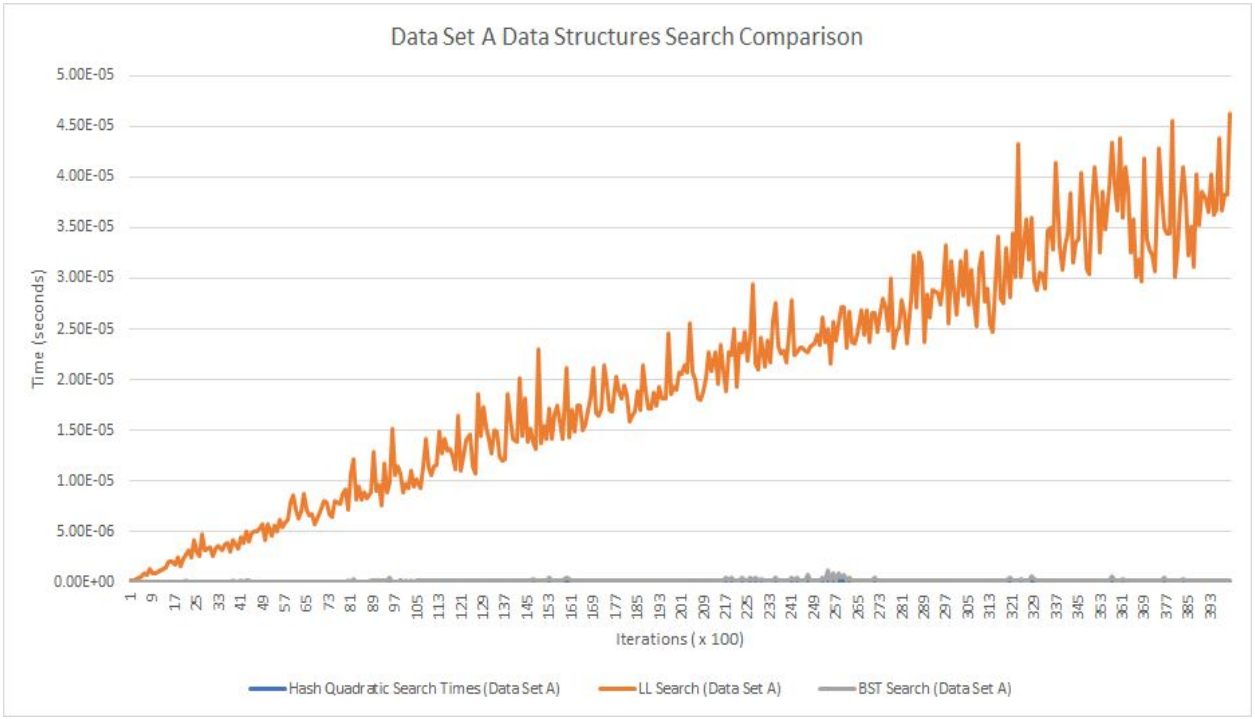




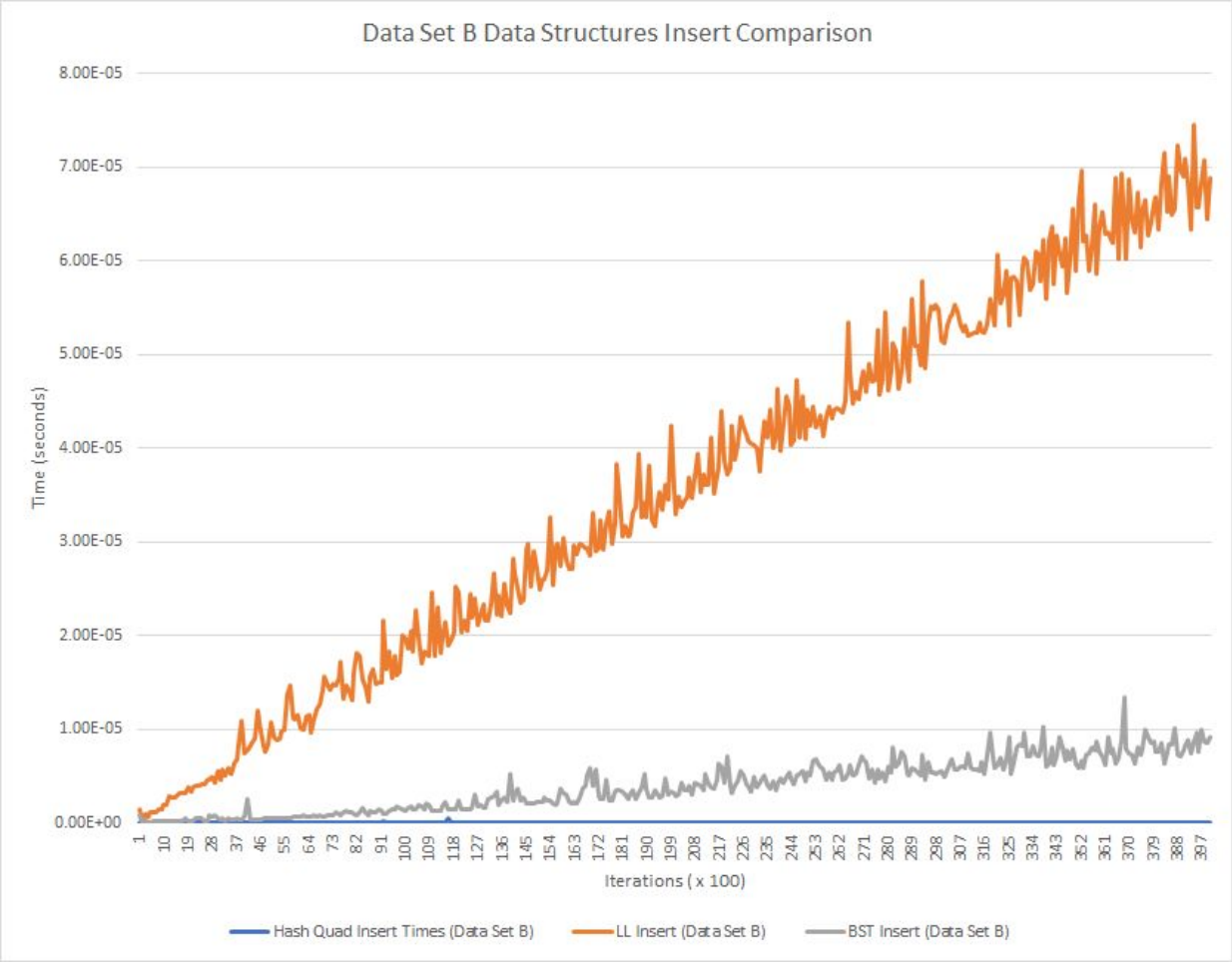
2.4. Summary Figures:

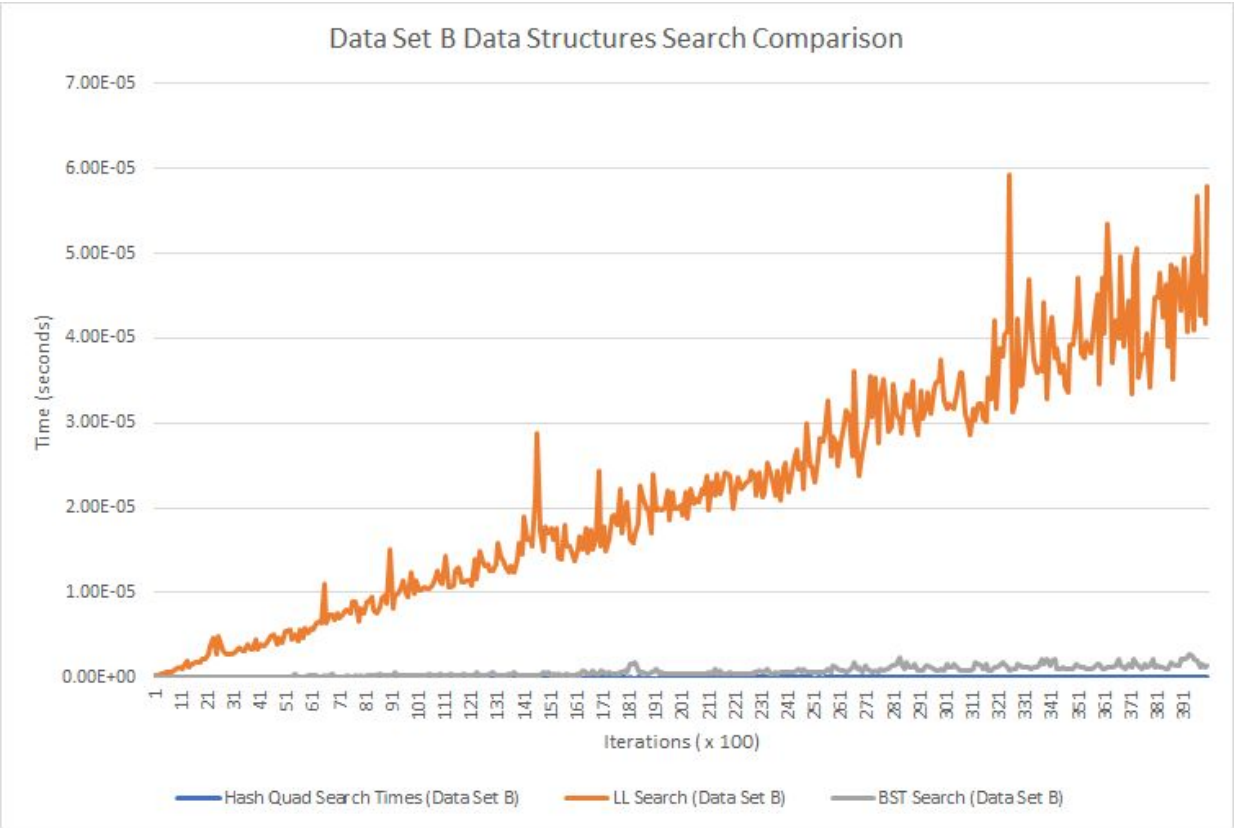
2.4.1. Data Set A





2.4.2. Data Set B





3. Proposed Solution

In forming our proposed solution we primarily looked at the time efficiency for insertion and search, but also the scalability of each specific data structure. We recommend that the USPS adopt the Hash Table utilizing a quadratic probing mechanism for their new algorithm to store their shipment data, as it is the most superior data structure in both regards.

The linked list is far too slow, and the time needed for insertion and searching grows linearly with size.

Although the binary search tree is very fast, in scenarios where it is not balanced, such as dataset B, the data structure would adopt a growth rate that is much more linear than logarithmic. However, our recommendation might be changed if USPS wanted to try implementing a self balancing tree.

The other hash table collisions are quite superior in time efficiency as they all appear to have an average search/insertion time of 1; however, we believed there were problems in scalability. The linear collision would not be as scalable in light of USPS massive data, as there will be a high amount of collisions near the end of the dataset due to clustering. Chain collision could also face problems in spacing and efficiency as we are left with gaps in the table and the fear that given enough data a node could have a very long linked list—leading to similar timing as our linked list.

With quadratic probing, although its time is also affected by collisions, is still fast and allows better scalability for USPS. In addition, all the available spots for storage would be filled eventually. Therefore, we believe the best data structure implementation for USPS would be a hash table enforcing quadratic probing.

Running the program instructions:

Binary Search Tree and Linked List

- Open driver.cpp, linkedlist.cpp, linkedlist.hpp, BST.cpp, and BST.hpp
- Run in terminal using *driver.cpp*
    - C++ driver.cpp linkedlist.cpp BST.cpp

Hash Table Chaining

- Open hashChaining.cpp, hashChaining.hpp, and hashCDriver.cpp
- Run in terminal using *hashCDriver.cpp*
    - C++ hashCDriver.cpp hashChaining.hpp

Hash Table Linear

- hashLinear.hpp, hashLinear.cpp, hashLDriver.cpp
- Run hashLinear.cpp and hashLDriver.cpp in terminal

Hash Table Quadratic

- hashQuad.hpp, hashQuad.cpp, hashQDriver.cpp
- Run hashQuad.cpp, hasQDriver.cpp