Final Project: Save The USPS

Omar Chaarawi
CSCI 2270

In this project several data structures are compared with the goal of identifying the structure best suited to increase the efficiency of the USPS. The current algorithms used by the USPS for managing shipments use a linked list data structure to insert and search shipment IDs. Using 40,000 sample ID's,

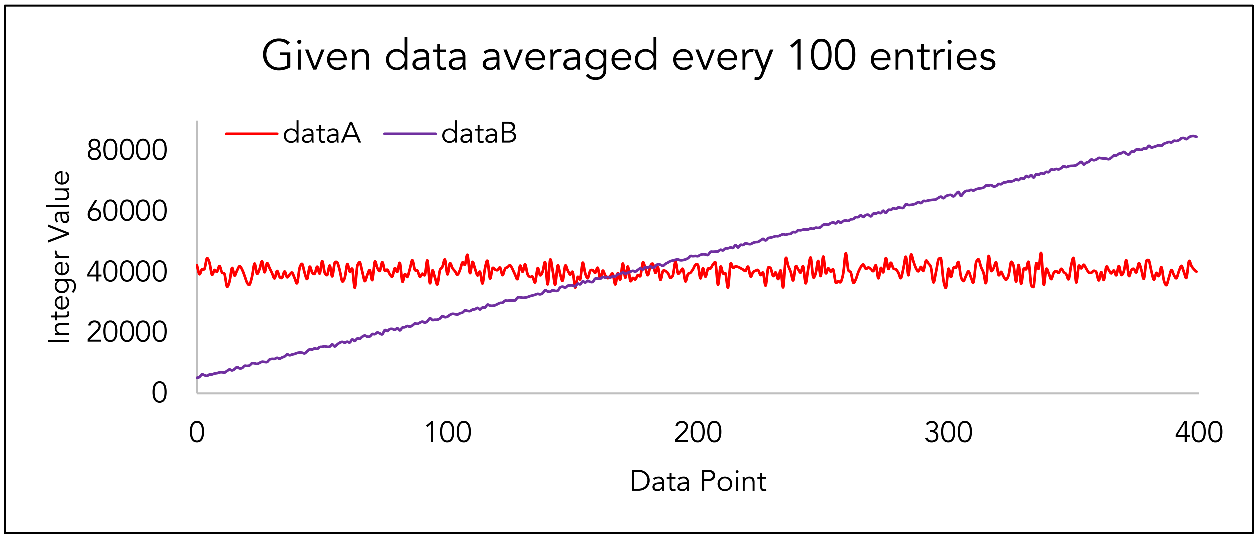


Figure 1. Graph of data provided for testing. Notice that while data in set A is more distributed, data in set B follows a trend of increasing value.

first the linked list data structure was implemented for a basis of comparison. Next various abstract data structures were implemented so as to identify a superior ADT for the USPS.

METHODS

Six data structures were compared by timing their times of insertion and search using the chrono library. Insertion times were recorded every 100 iterations and then divided by 100 to get an average insertion time per shipment ID. The same was done for search. Items were searched using a pseudo random index in the range of the 0^{th} to n^{th} entry.

ANALYSIS

Linked List

In theory a linked list will have linear complexity and that is reflected in the average insertion and search times. While the first couple insertions and searches did not take much time, the linear behavior makes this a poor data structure for working with large datasets.

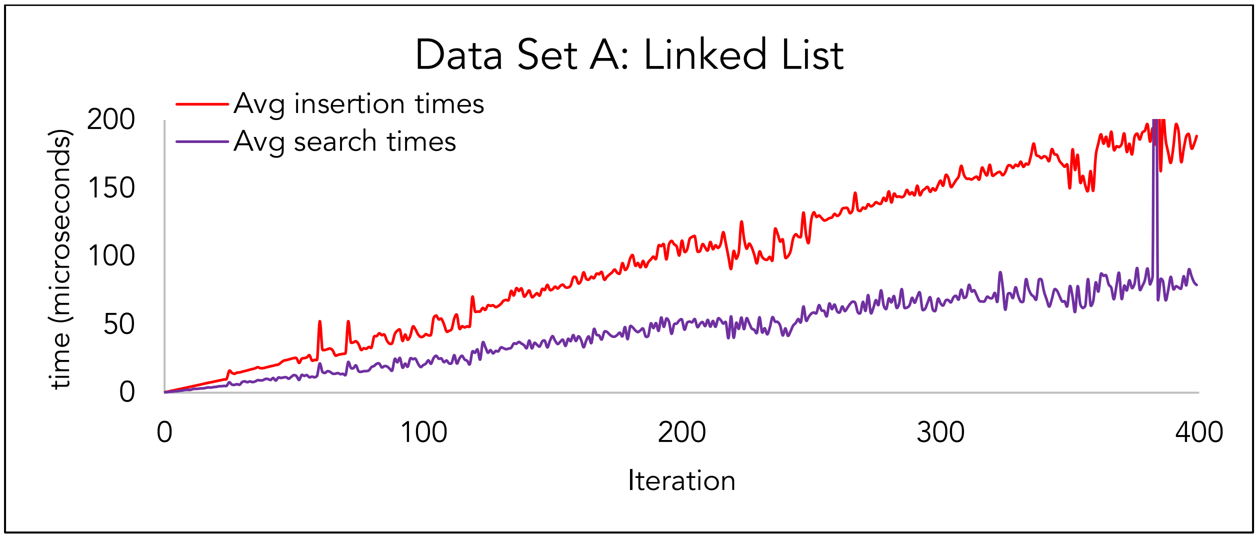


Figure 2. The time of insertion and search is linear for the Linked List ADT.

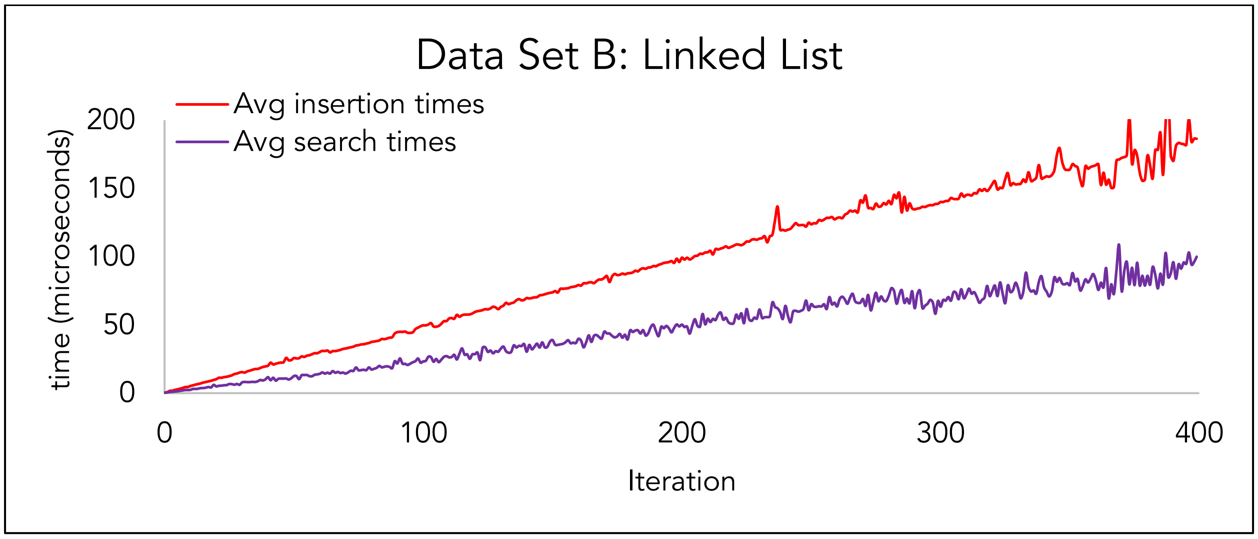


Figure 3. Again the insertion and search of the data in set B showed linear behavior.

Binary Search Tree

A binary search tree offers the ability to insert and delete with a complexity that is logarithmic assuming the tree is well balanced. This structure seems to offer a significant increase in efficiency of well distributed values such as that of data set A. However, when using this structure to assemble data that is not well distributed such as the data from data set B, a lack of balance causes the complexity to take on linear behavior and reduce the efficiency of this structure.

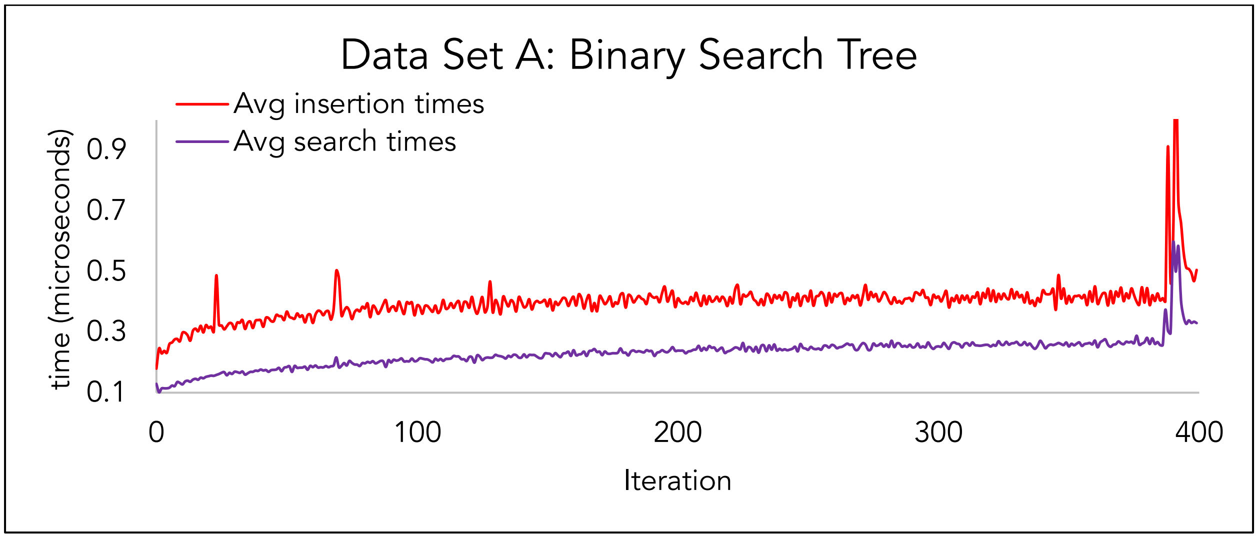


Figure 4. Assembling well distributed data leads to a logarithmic time complexity.

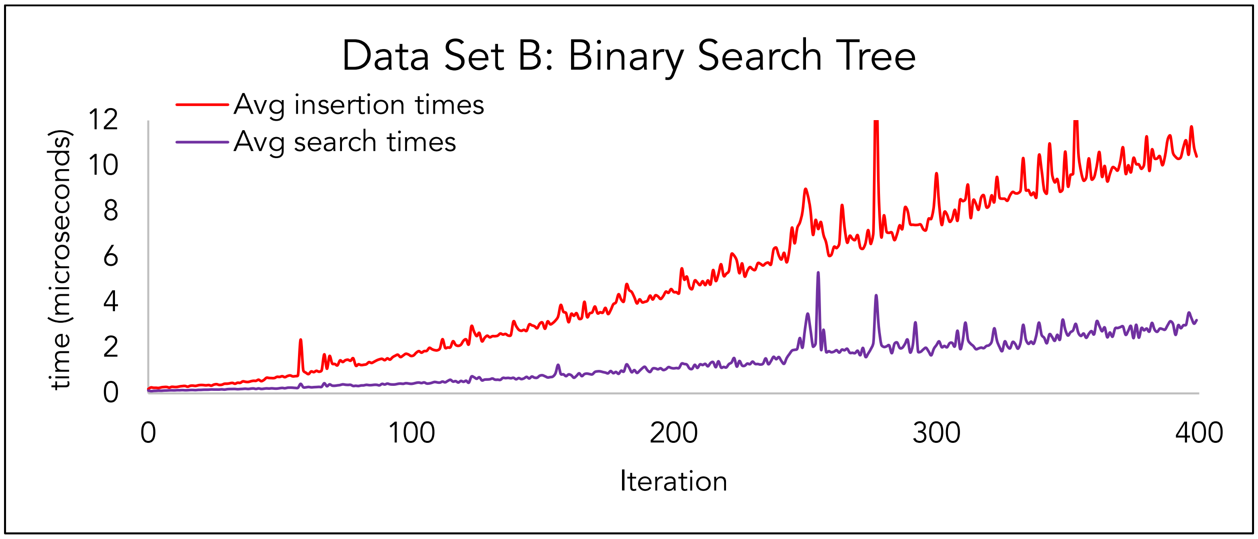


Figure 5. Assembling data that is more ordered may lead to some linear complexity causing inefficiency.

Hash Table: Linear Probing

Using a hash table with linear probing we are able to insert and search in nearly constant time initially. However, as we run out of room in the hash table, the efficiency quickly diminishes as the program struggles to map the remaining shipment IDs to empty spots.

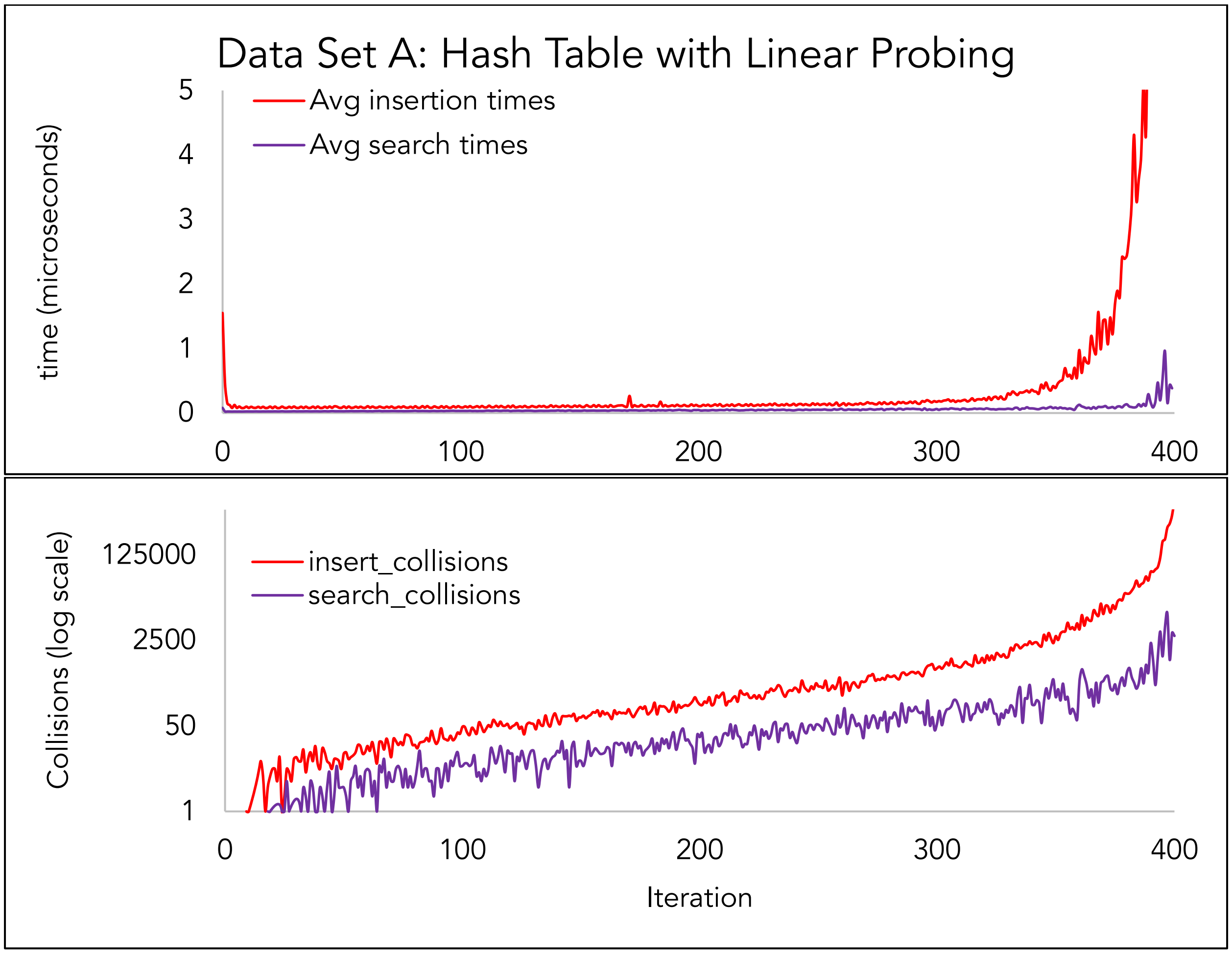
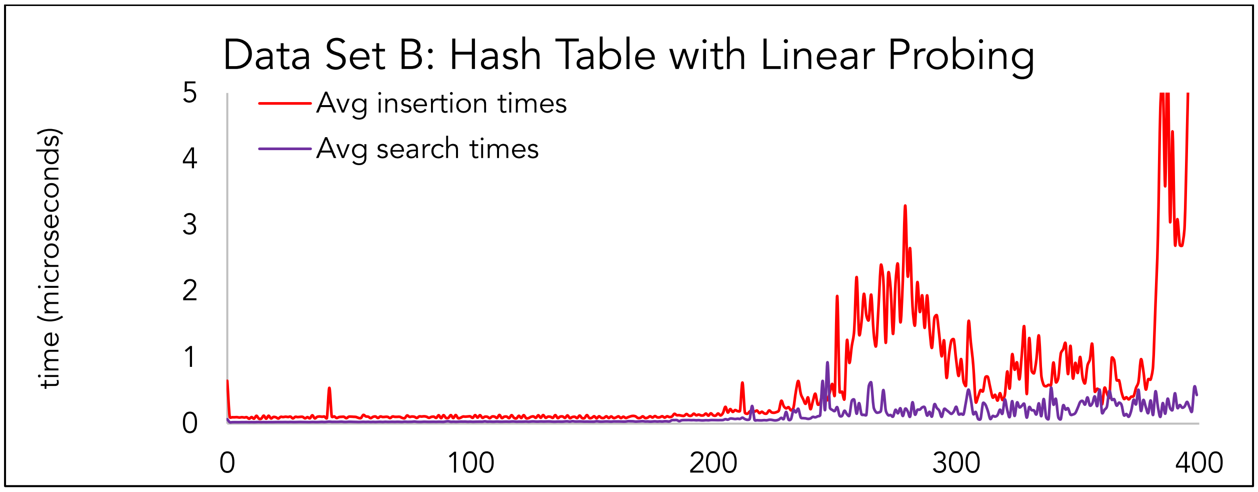


Figure 6. A graph of the insertion and search times paired with the rate of collisions reveals that as space in the table is used the exponential increase in collisions causes a sudden exponential increase in time complexity.



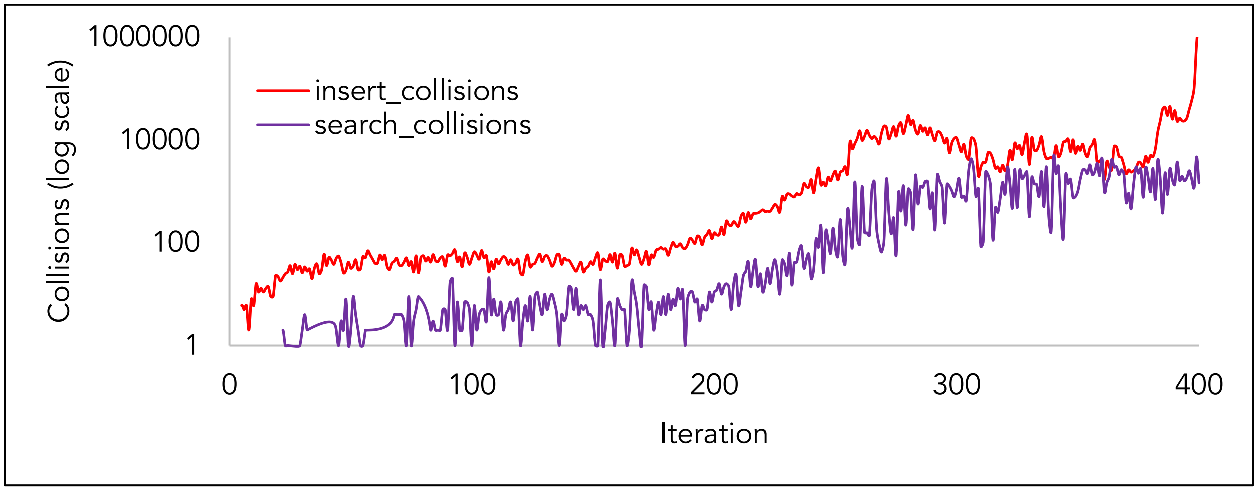


Figure 7. Assembling data in set B shows a similar sudden increase in collisions and time as the capacity is approached.

Hash Table: Quadratic Probing

While there is a decrease in time and collisions using quadratic probing due to less clustering the problem of diminished efficiency when approaching the capacity of the hash table still exists.

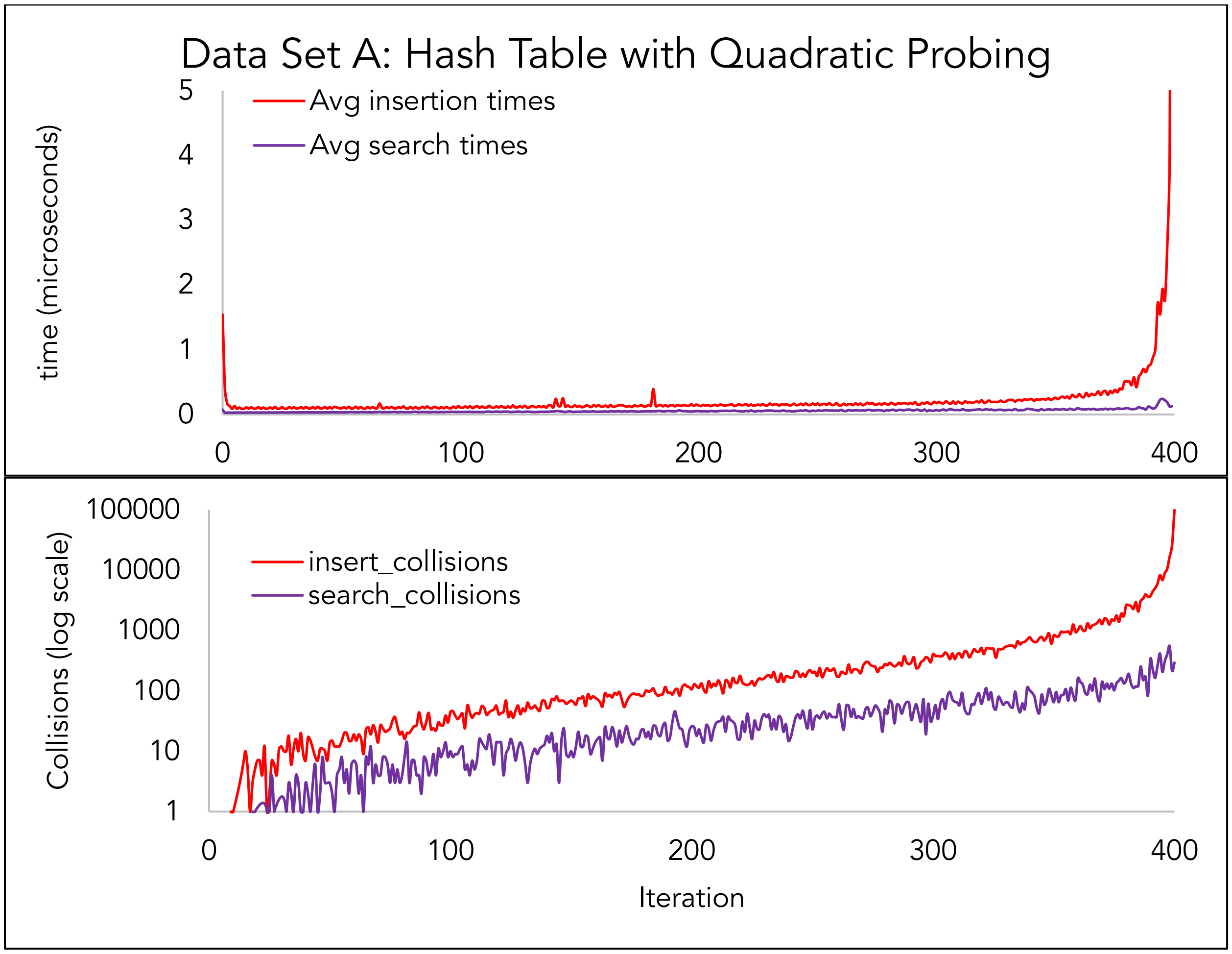


Figure 8. Quadratic probing offers some benefits over linear probing in terms of efficiency due to less clustering.

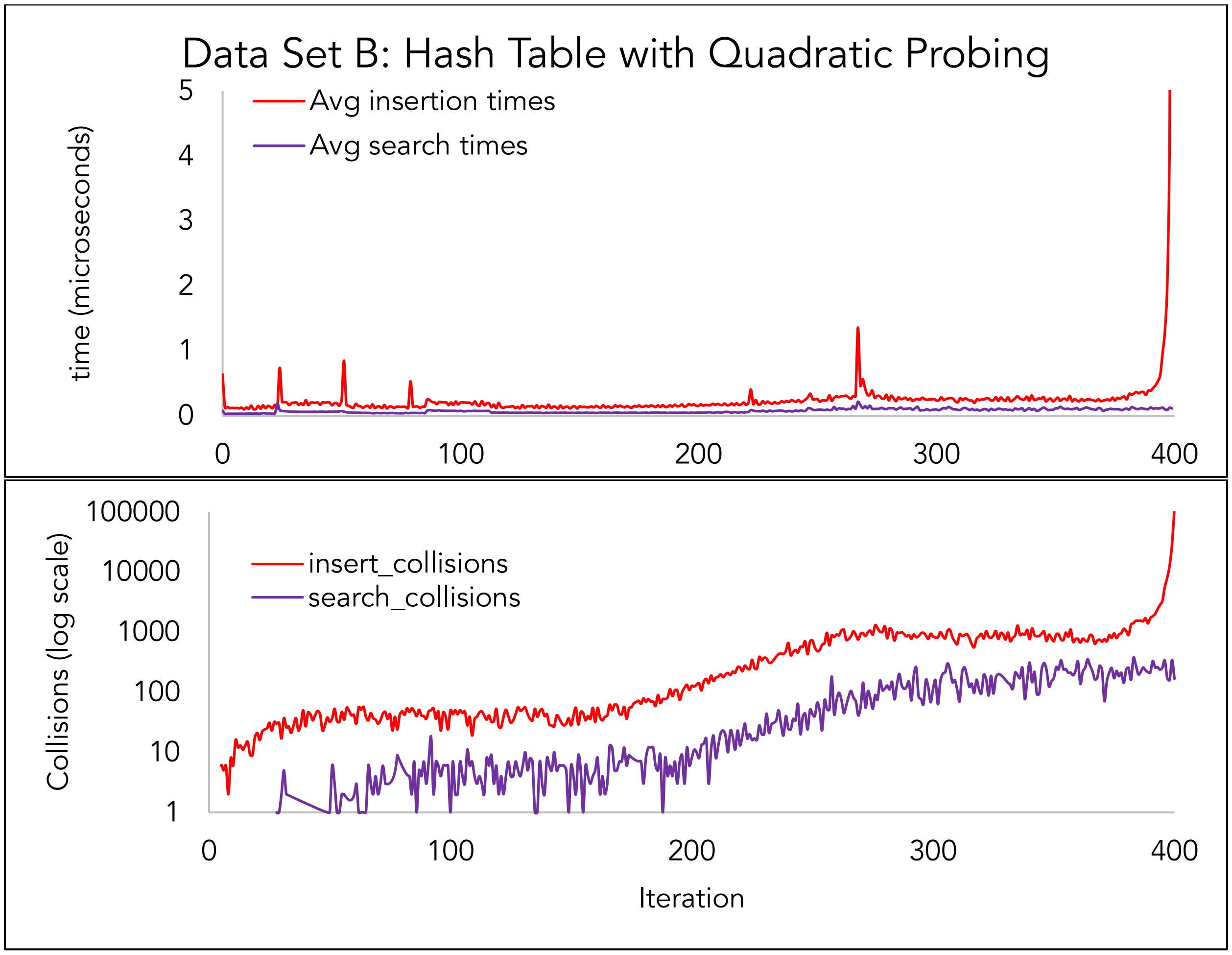


Figure 9. Assembly of data in set B shows similar behavior to assembly of data in set A.

Hash Table: Chaining

Finally, hashing with chaining seems to offer the best efficiency of all structures compared. By creating linked lists to store shipments in the event of a collision we have eliminated the problem associated with approaching the capacity of the hash table. Insertion and search can be performed in nearly constant time.

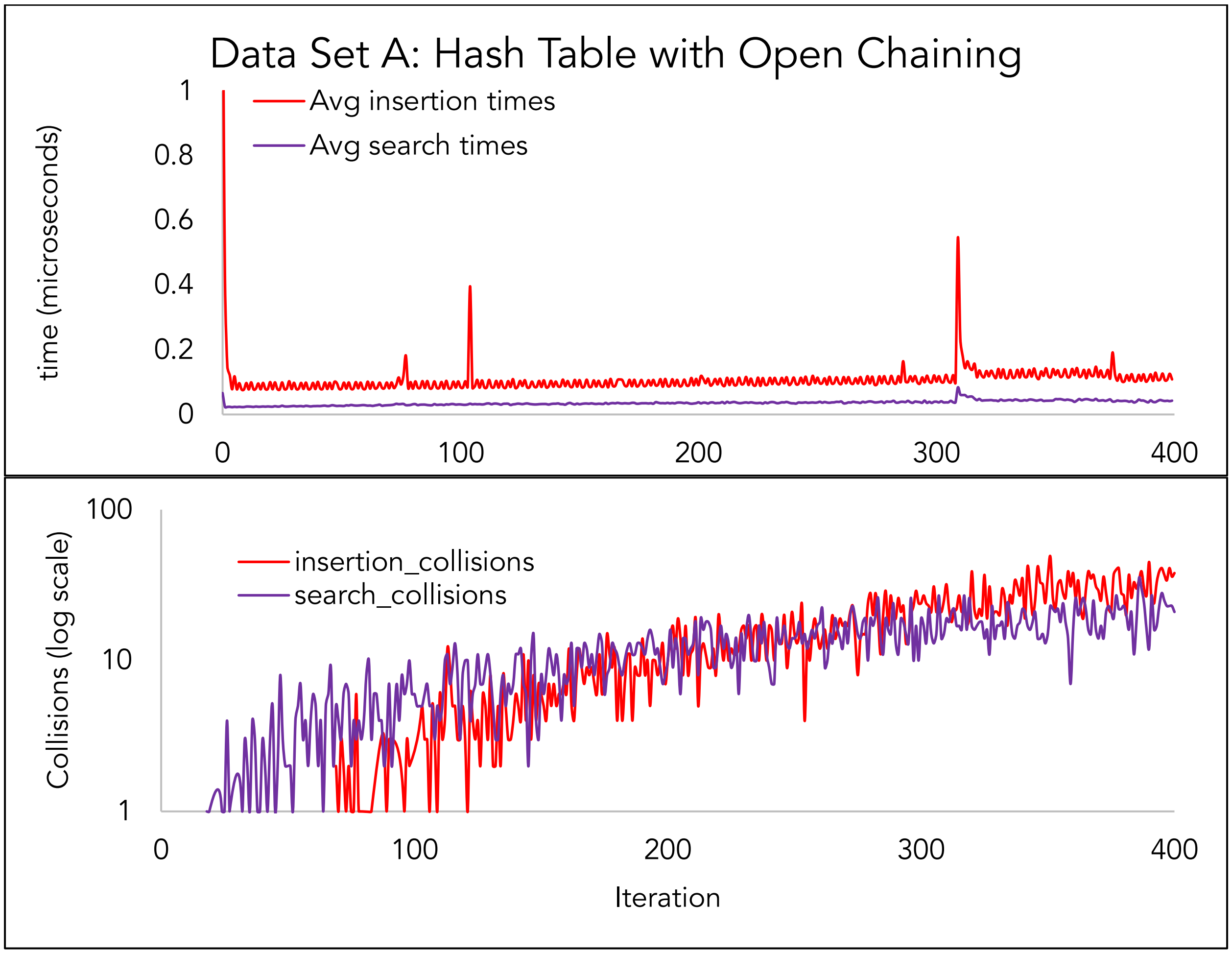
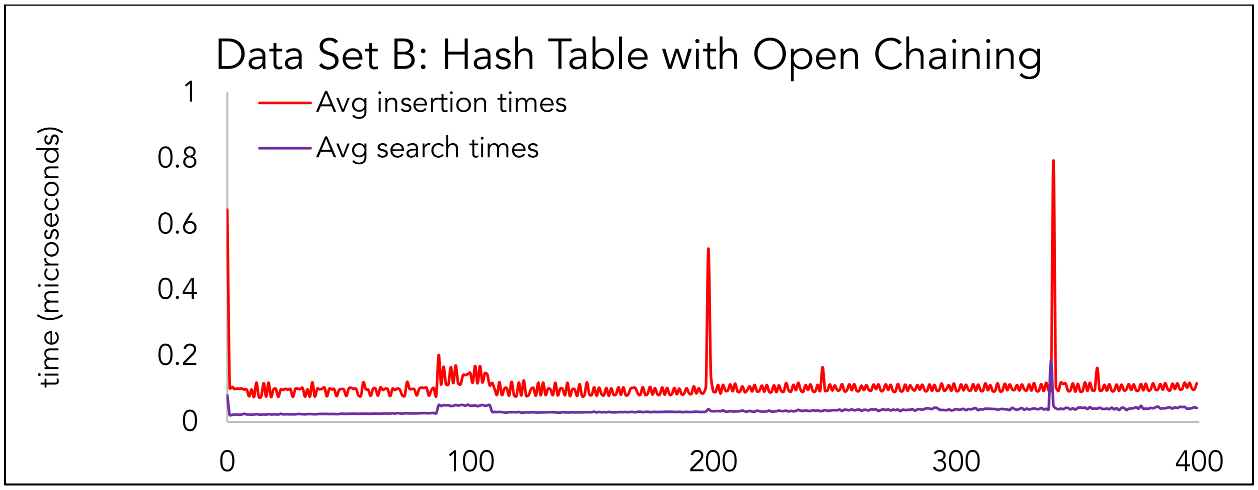
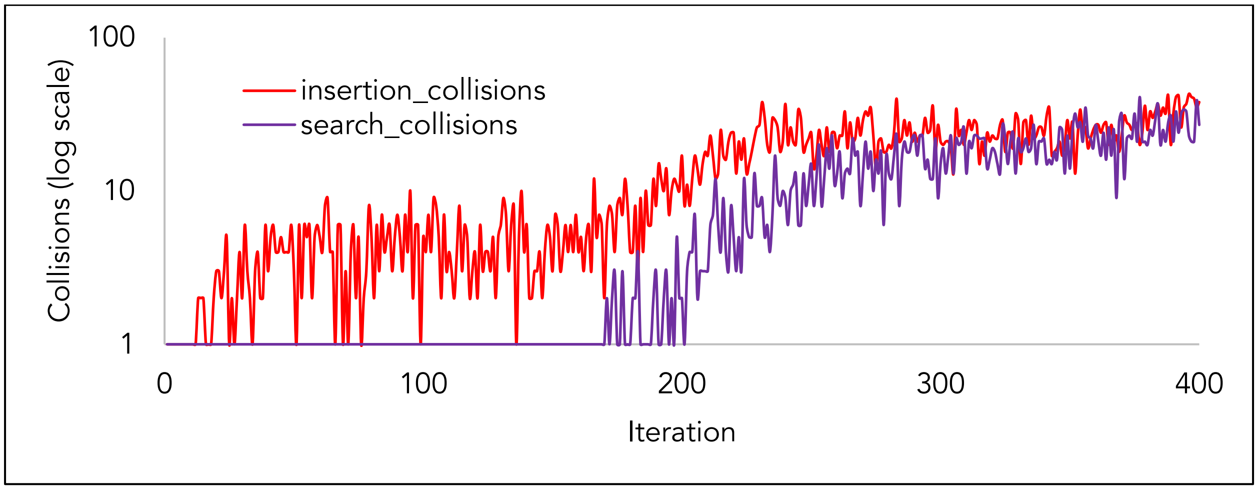


Figure 10. Chaining eliminates the problem of capacity.





SUMMARY

A graph of the insertion and search times using the various data structures quickly reveals that by implementing a hash table with chaining the USPS could dramatically increase the efficiency of their operations.

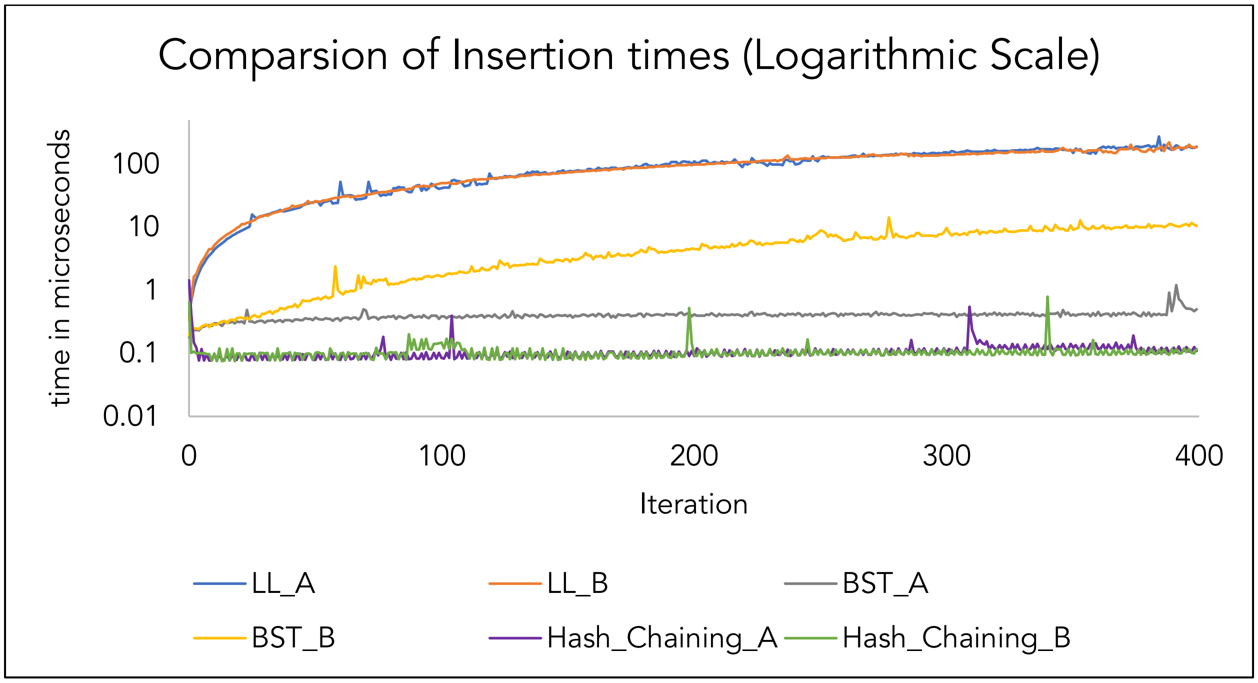


Figure 11. Comparison of insertion times shows that a hash table offers the most efficiency.

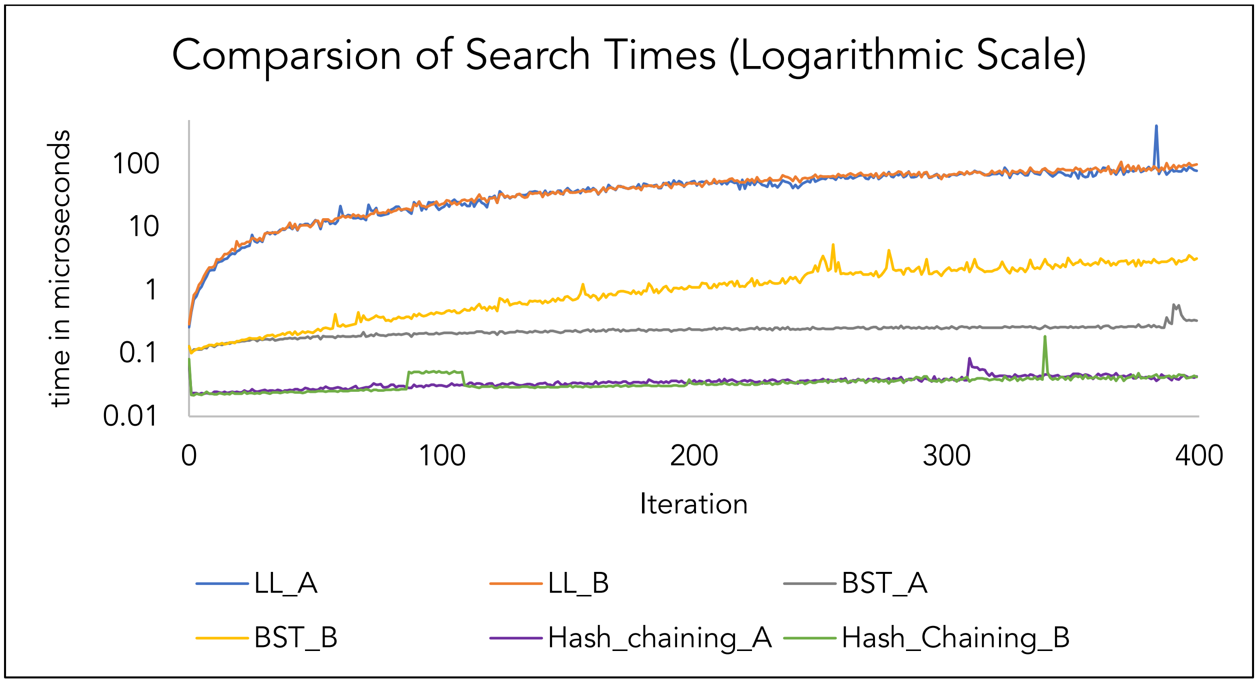


Figure 12. Comparison of search times shows that a hash table offers the most efficiency.

FURTHER IMPROVMENTS: TRIE DATA STRUCTURE

While creating a hash table with chaining may increase the efficiency of lookup and insertion times, managing the size of the hash table may create a future problem for the USPS. Considering in this scenario the USPS is using an outdated linked list data structure, it is unlikely the USPS will update their software frequently and they may benefit from a more robust, low maintenance structure. I have implemented a trie data structure that offers the benefit of quick insertion and search times that are only slightly slower than that of the hash table, but will not require maintenance as the USPS scales up operations. The chaining attribute of the hash table may cause linear time complexity if the size of the hash table is not properly managed as the USPS increases the number of shipment IDs stored. A trie would take care of this and is therefore a superior storage method in this application.

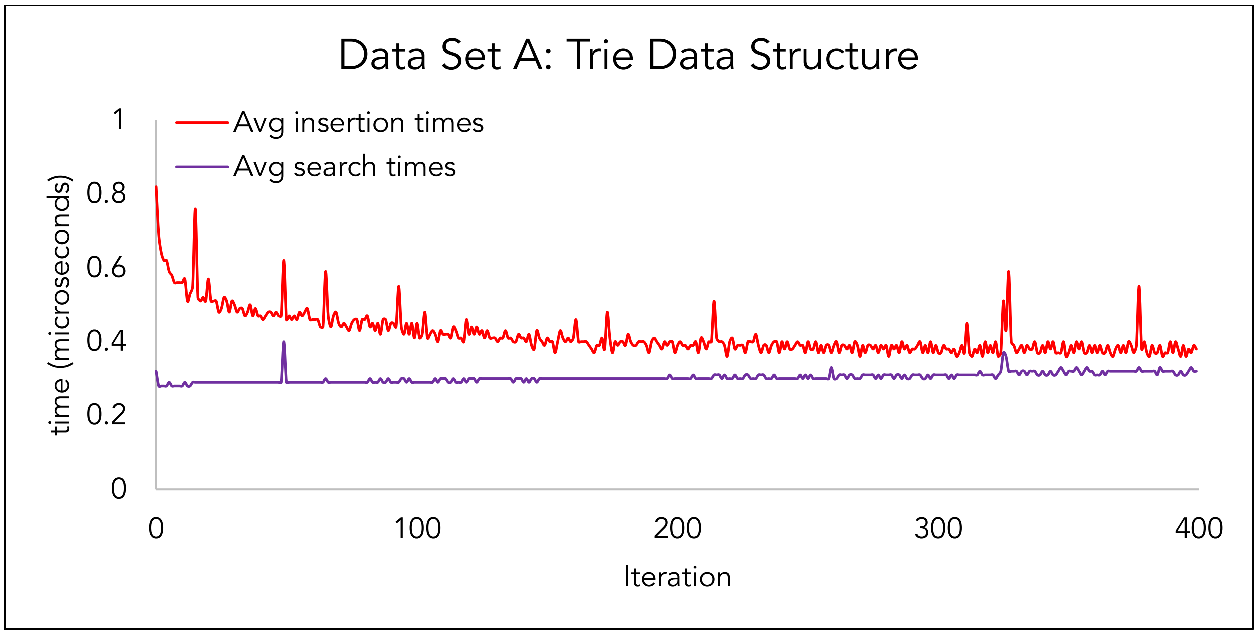


Figure 13. The trie initially will take some time as it creates nodes for storage, but quickly is able to store and lookup values in constant time.

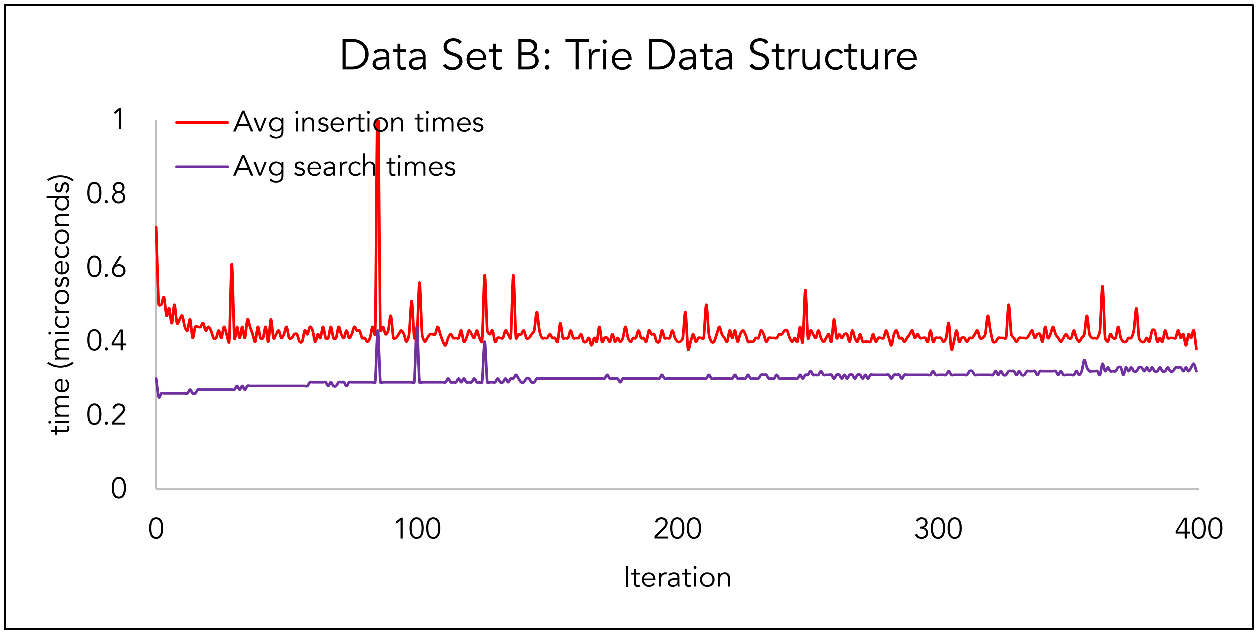


Figure 14. Storing more ordered data does not impact the capabilities of the trie.

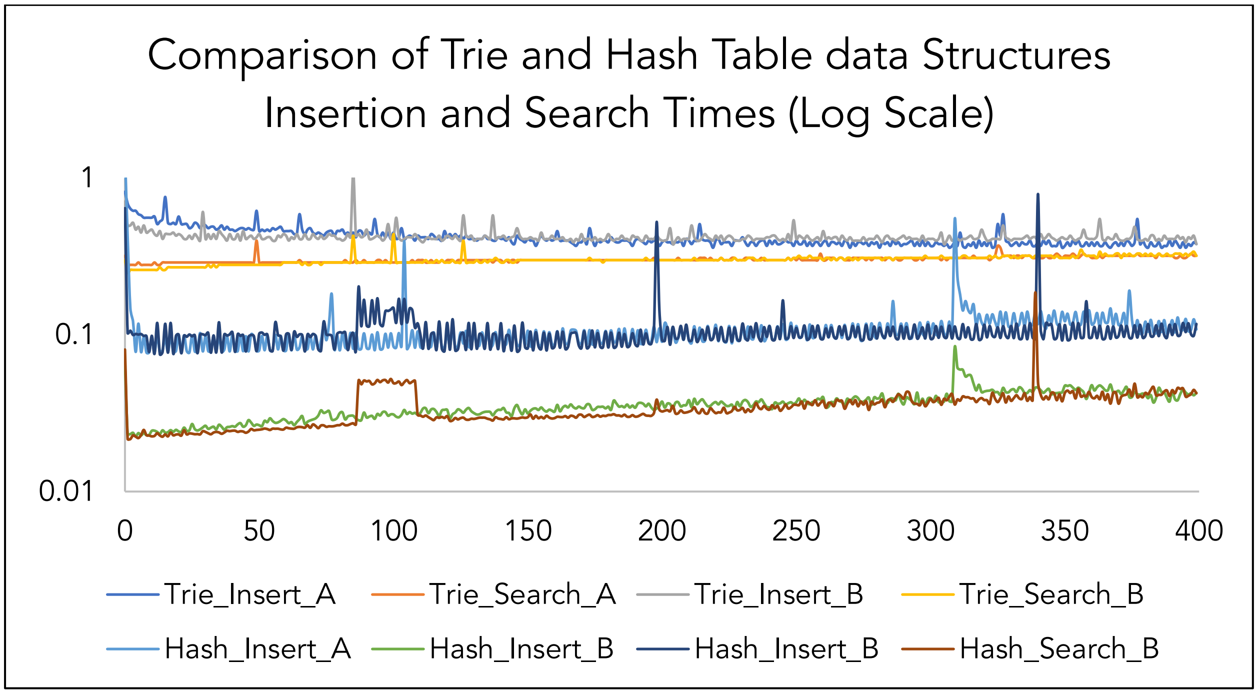


Figure 15. While hashing may be faster given 40,000 numbers to store. The hash table lacks scalability. In the event of a sudden increase in shipments, the efficiency of the hash table may suffer without maintenance (increasing the capacity of the table), while a trie would be able to maintain efficiency as its capacity is only bounded by memory.