# Saving the USPS… With a Hash Table!

Ashlyn Duy

## *Given Data*

Before the investigation began, it was necessary to evaluate the distributions of the given data sets to understand how different data structures may perform.

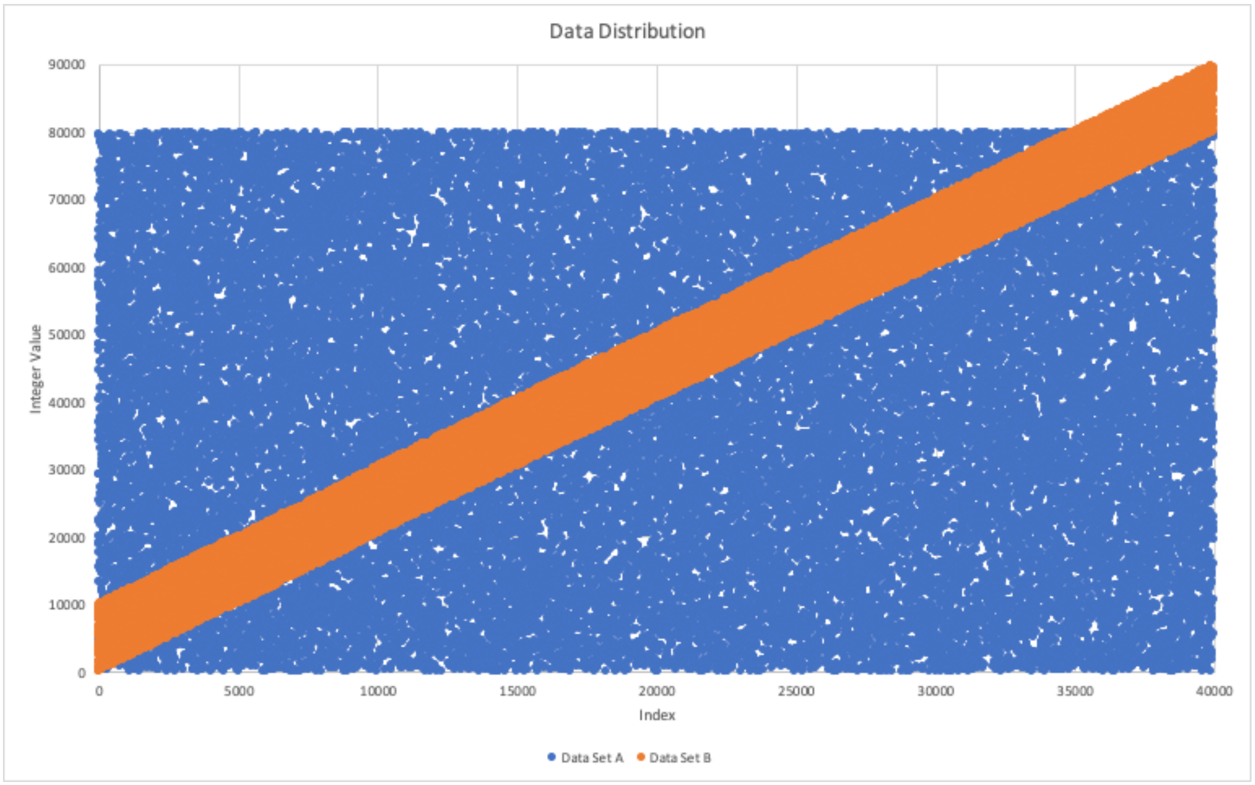

*Figure 1*

Set A is thoroughly distributed on the interval [0, 80000], compared to the highly clustered set B which instead had values that stayed within 10000 of the integer's given index. Thus, for most of the data structures it was likely that set B would have worse performance due to the nature of its distribution.

*Linked List*

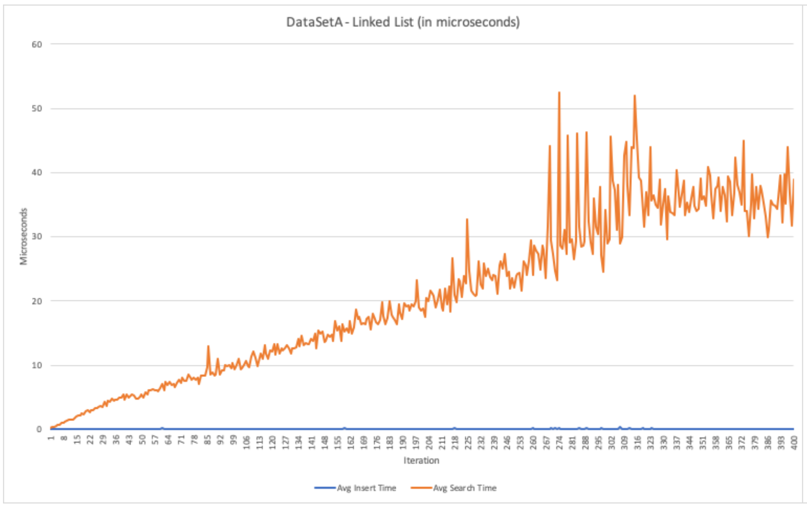

Figure 2

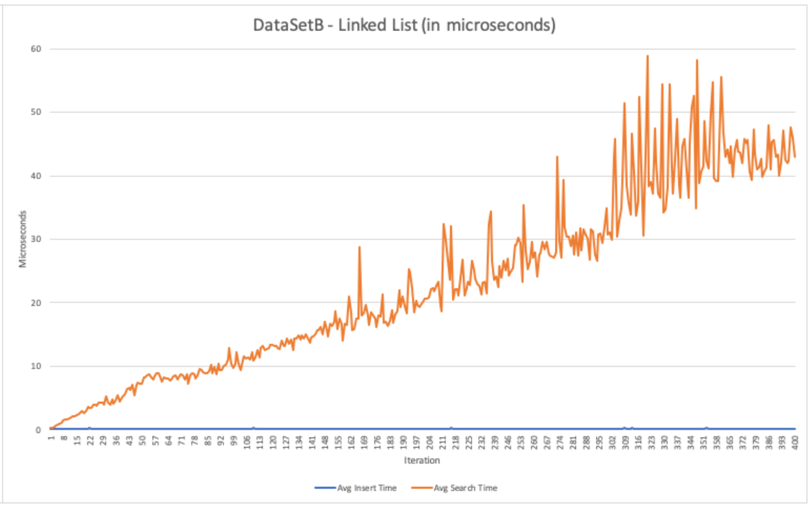

Figure 3

The current USPS implementation of a linked list was the first structure tested, yielding disappointing results which highlights the need for a change. As seen in Figures 2 and 3, the average search time is growing linearly with the number of elements created, thus achieving the anticipated complexity of O(n). Average insert times remain low due to inserting at the head of the linked list as opposed to the tail, as it was viewed as more efficient. A linked list implementation is incredibly inefficient for this large data set.

*Binary Search Tree*

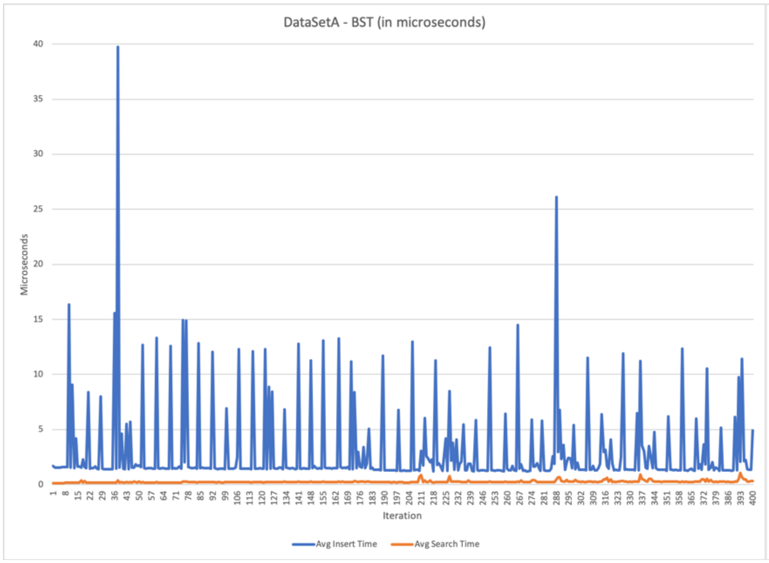

Figure 5

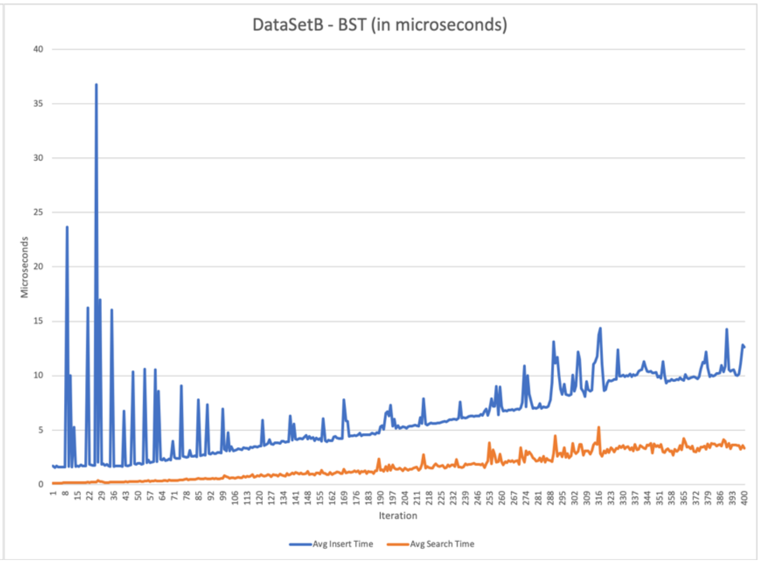

Figure 4

A BST implementation does show substantial improvements in search time, however the insert time still proves costly. Notably, duplicate values were inserted into the tree to obtain an accurate representation of the data sets. Furthermore, it seems that the nature of set B's distribution leads to a worst-case O(n) performance. While it offers an improvement over a linked list, there are better structures to be found.

*Hash Table with Chaining*

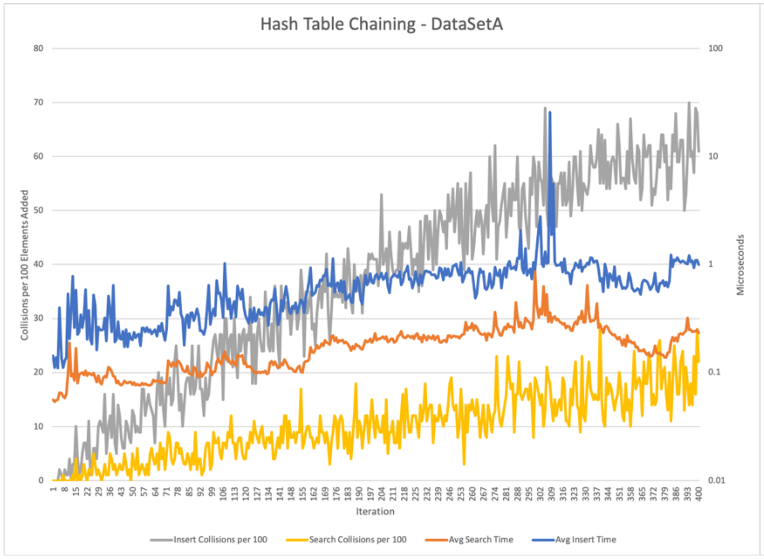

*Figure 6*

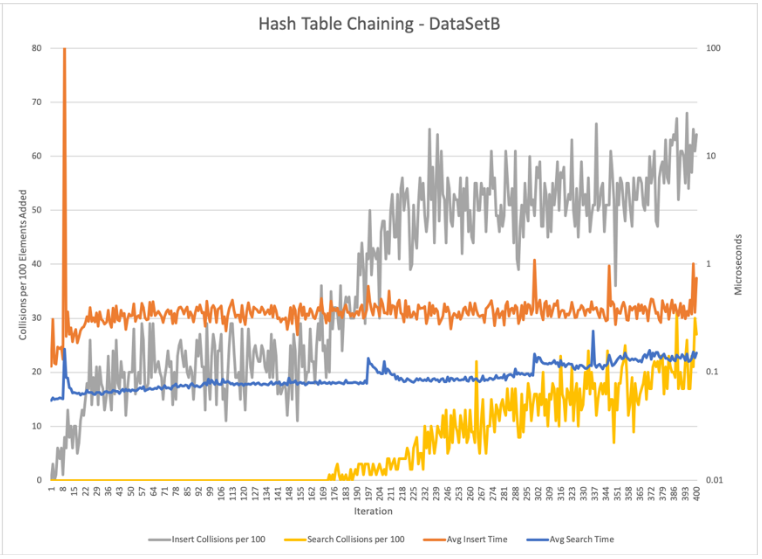

*Figure 7*

The first of the three hash table implementations, chaining, set a new bar that blew the previously tested structures away. Logarithmic scales were necessary to demonstrate the fluctuations in insertion/search time, but they were overall much faster than a BST implementation across both data sets. Search and insert time maintained an expected average O(1) performance as seen in figures 6 and 7, thus resulting in the best method thus far.

*Hash Table with Open Addressing – Linear Probing*

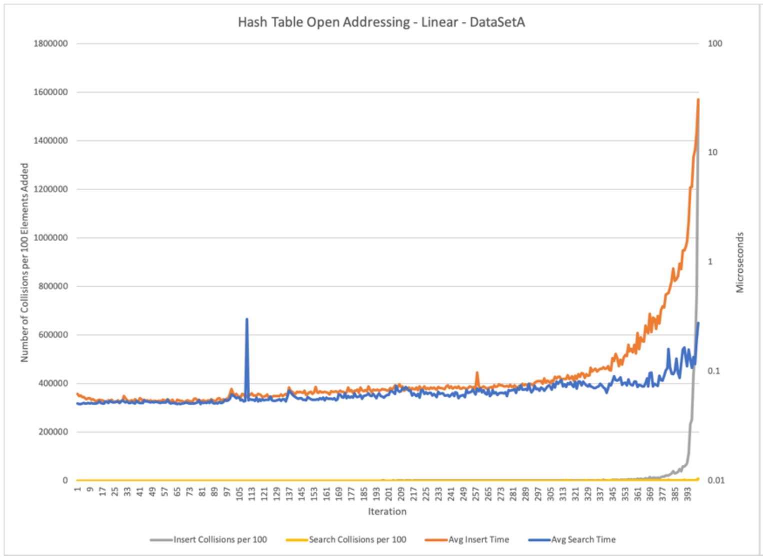

*Figure 8*

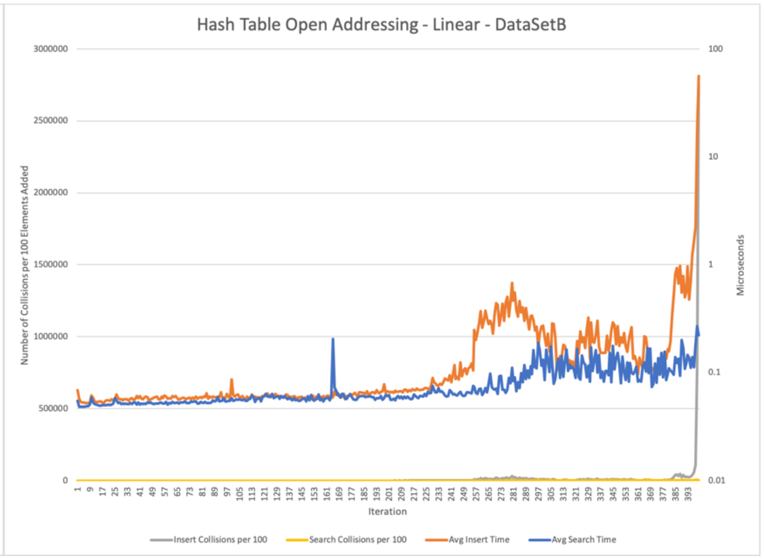

*Figure 9*

The linear probing open-addressing technique shows significant benefits over chaining the lower the load factor is on the table. Once the load factor approaches its peak, the performance begins to severely degrade as the number of collisions increase, however the linear probe still offers superior performance to chaining, therefore making it the best implementation of any structures tested thus far.

## *Hash Table with Open Addressing – Quadratic Probing*

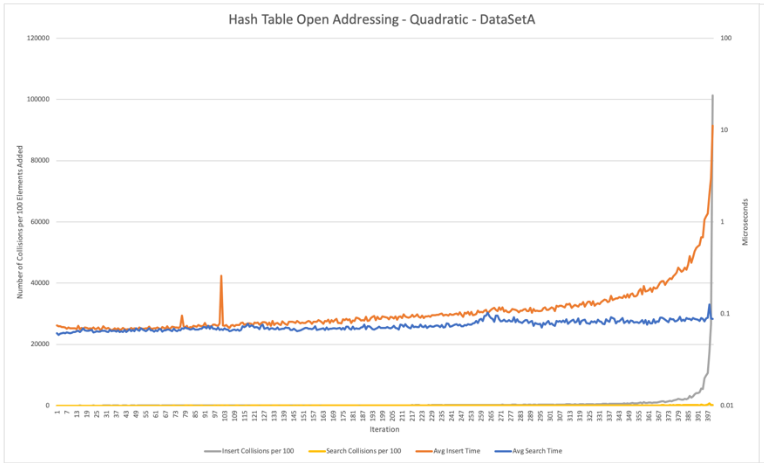

Figure 10

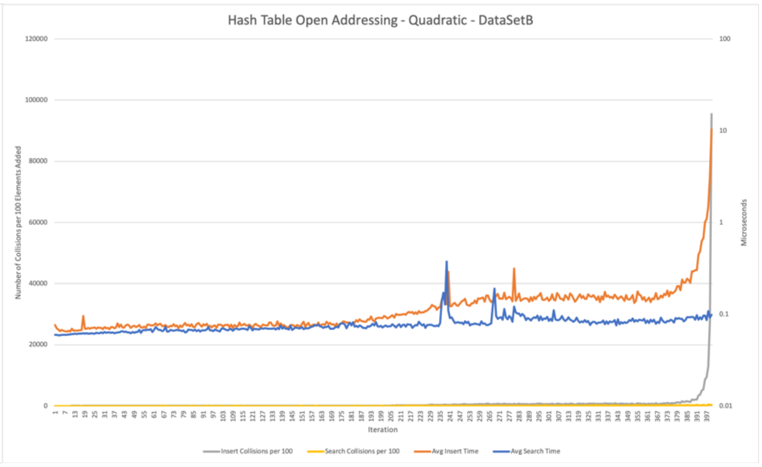

Figure 11

Quadratic probing offers the most reliable, low search times of the data structures tested. While average insert times seem to increase exponentially as the load factor on the table increases, it seems to be the best overall structure for the USPS when given some space modifications. The O(1) performance of average searches is, at times, 40,000 times better than the linked list implementation, making it an excellent structure for the USPS to migrate to.

---

## *Summary*

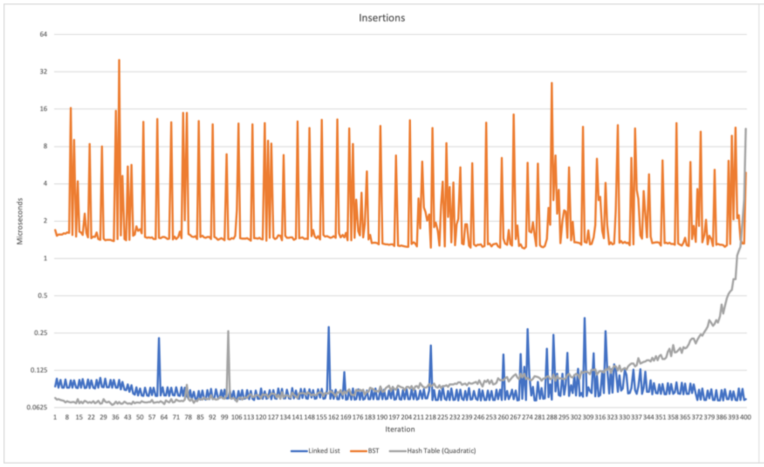

Figure 12

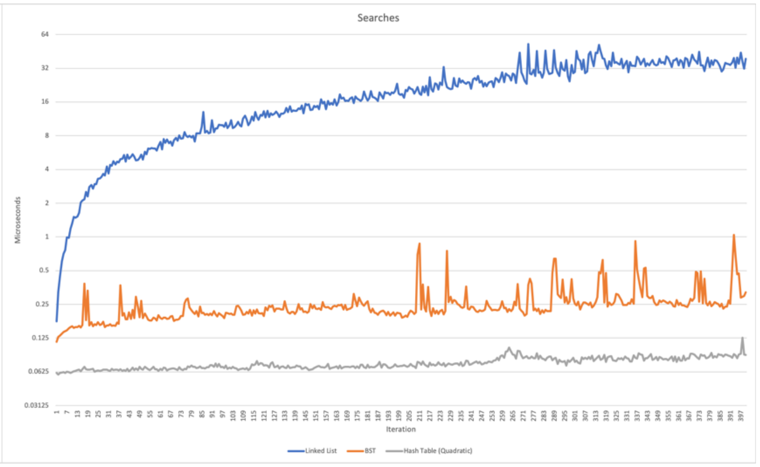

Figure 13

Overall, the hash table, specifically the quadratic probing open-addressing method, seems to be the optimal solution for the USPS. To further improve the quadratic probing implementation, I would increase the size of the hash table to bring the rising insertion times in line with the unparalleled search times. By decreasing the load factor on the hash table, the quadratic probing implementation would have the lowest insertion and search times for the USPS's needs of all the tested data structures.