Data Structures Final Project

CSCI 2270

Hyden Polikoff and Nancy Yoder

April 29, 2020

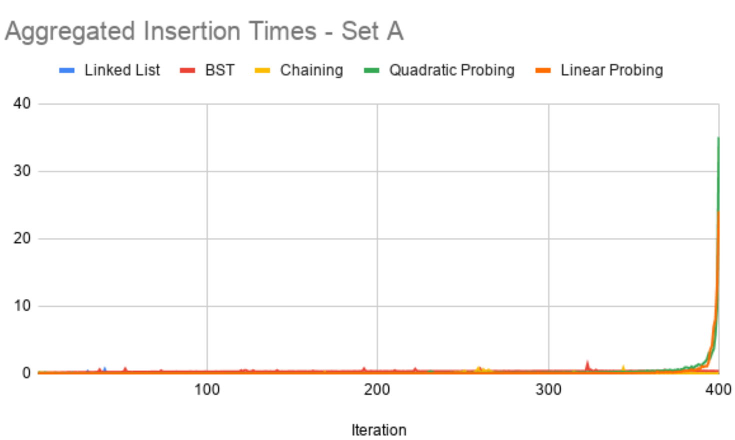
SAVE THE USPS!

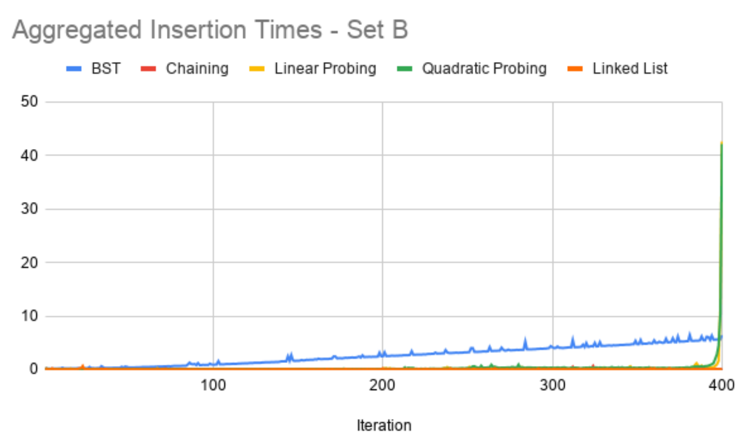
For our final project, we were asked to resolve a run-time issue the USPS was having due to a poorly chosen data structure. We were tasked with recreating the currently implemented linked list structure to establish a baseline. Then we were to follow with implementing a binary search tree and hash tables with various collision-resolutions.

Linked list has the best insertion times—big O complexity of O(1)— but it has the worst search times by a large margin when compared to other data structures. In our research, we found that there is not one data structure that has both the best insertion times *and* the best search times. Therefore, in order to pick the data structure that will work best for the USPS moving forward, we had to make some compromises to find the structure that wasn't necessarily the best at a specific operation but the most well rounded. This meant having decent times for both search and insertion.

Here are compiled graphs of both data sets

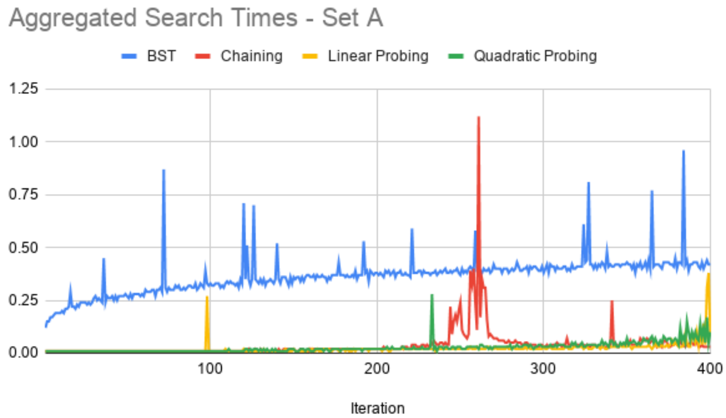
Insert Operations --

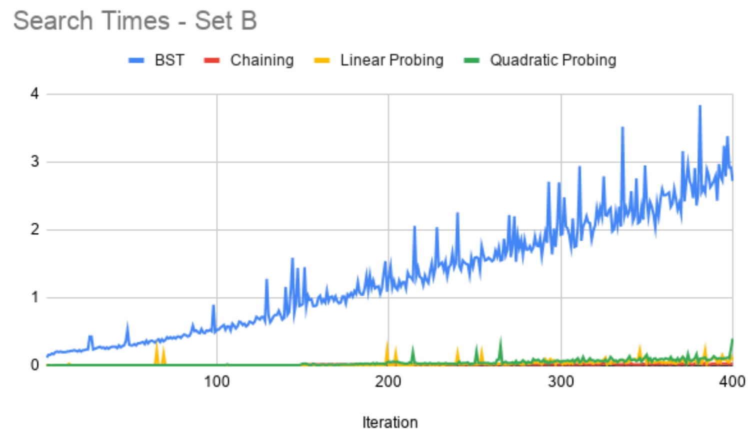




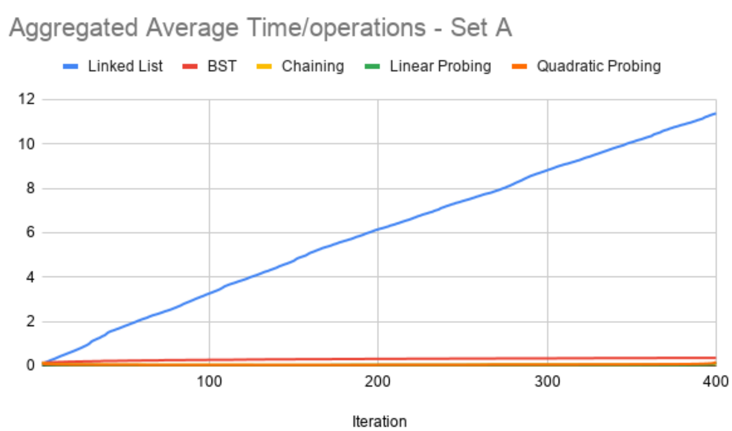
Search Operations --

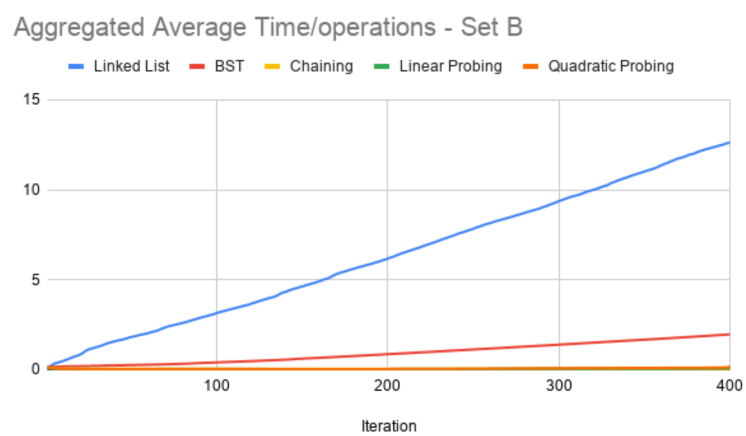
*Note: we did not include linked list in this data as it was so much slower and ruined the scale



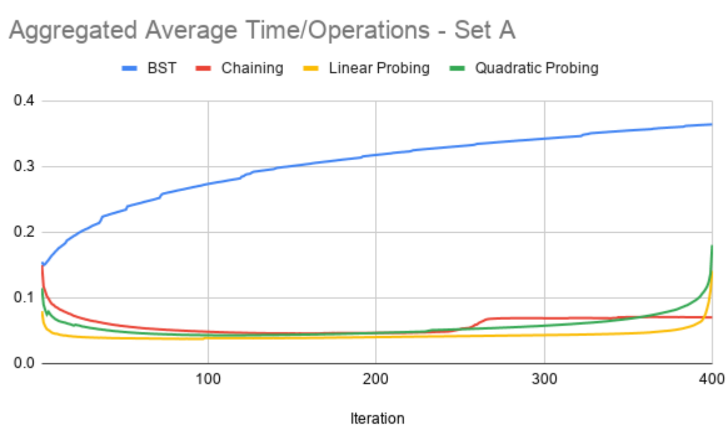


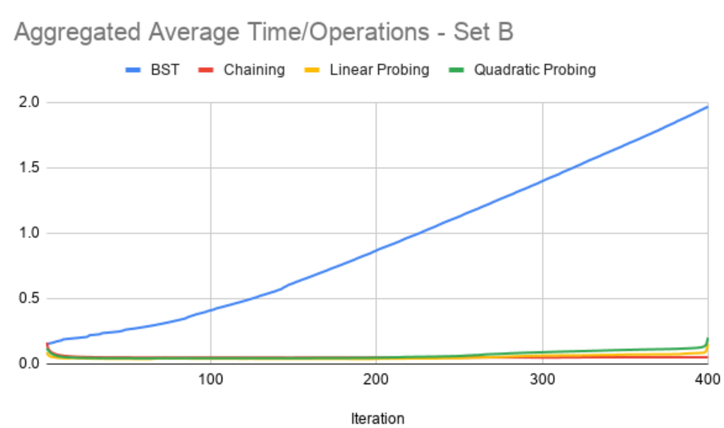
Average time per Operations ---



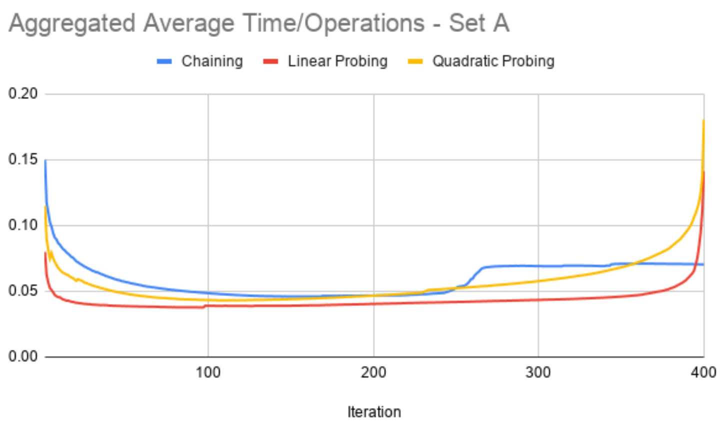


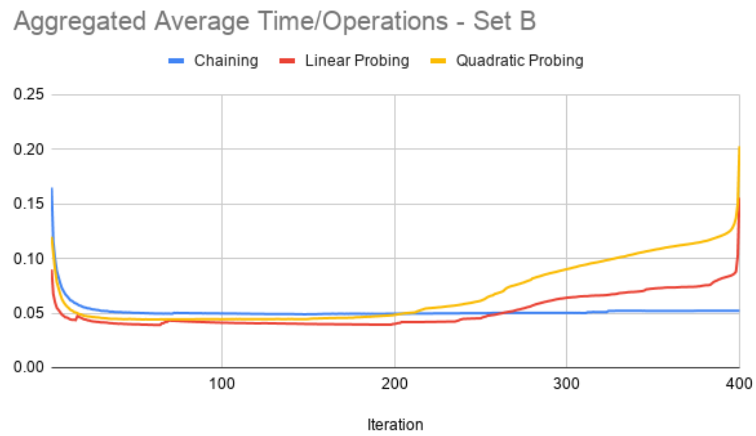
A more refined look without Linked List to better understand our data as it ruins the scale





A further refined look without Linked List or BST





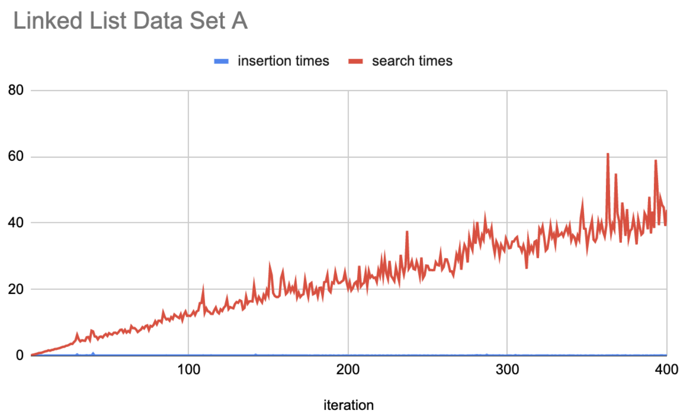
Once we reviewed the data, we came to the conclusion that the hash table data structure with chaining collision resolution, is the best data structure for the USPS to use. It proved extremely efficient with both insertion and search times. Linear and Quadratic methods performance significantly faltered once the data structure became nearly full as it then took much longer to find the next available structure The only drawback may be potentially unused space in the table resulting in a less memory efficient structure. However, this was not part of our final considerations.

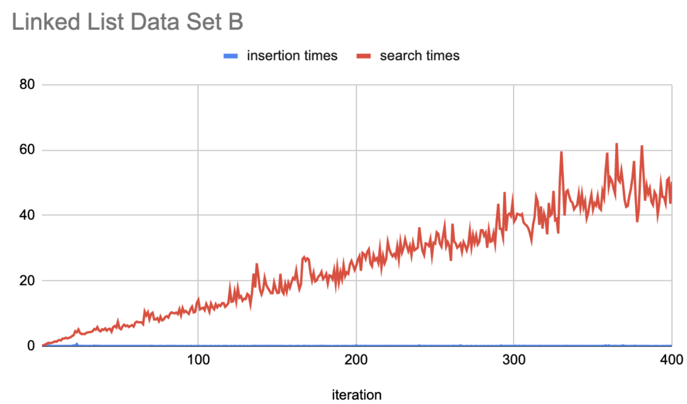
Based on our research, we came to the conclusion that data set A is initially more sorted than data set B, as the average BST search time for data set A was lower than that of data set B meaning it is a more balanced tree.

Below you can find graphs of each data structures insert and search as well as collision tracking for hash table implementations. A full set of data and chart found here:

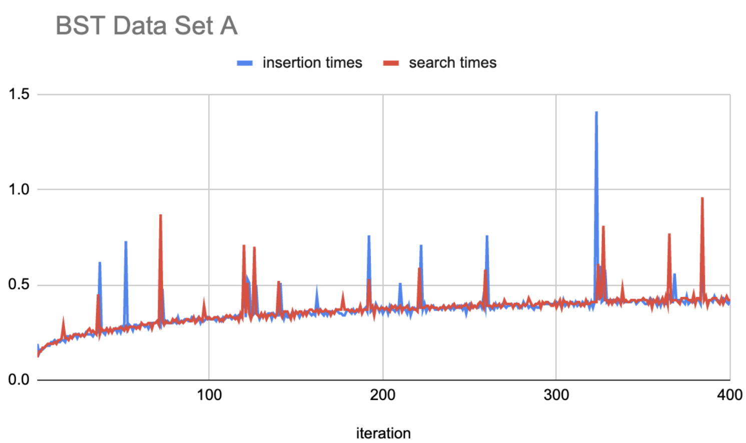
https://docs.google.com/spreadsheets/d/1Fi-oM3etp8mQWcN_AEdtOgpm7xlmAj_ecFnDCWTU-ow/edit?usp=sharing

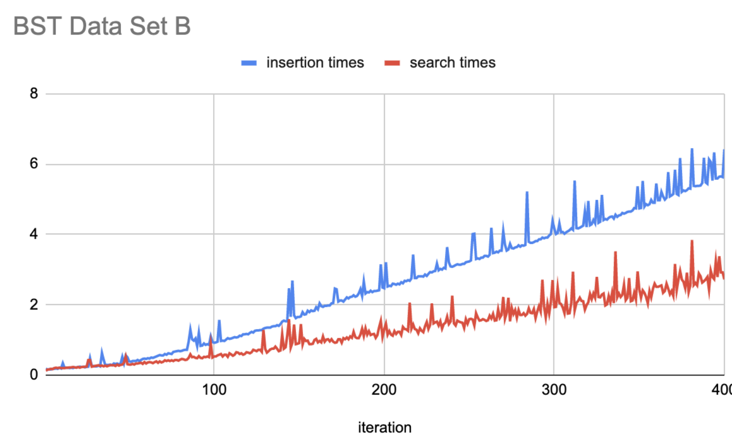
Linked List Findings





BST Findings

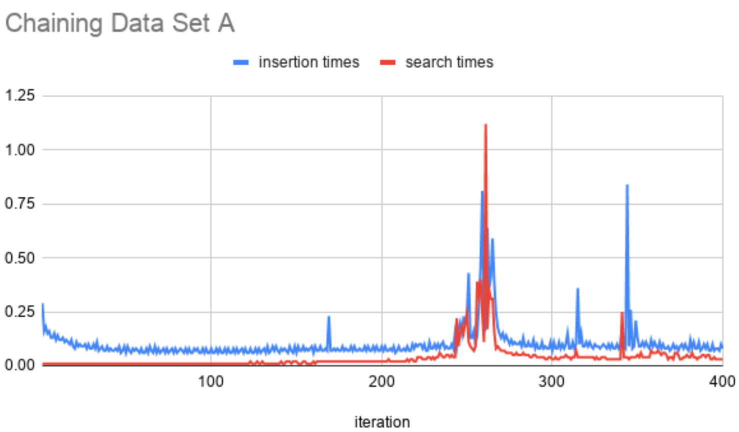


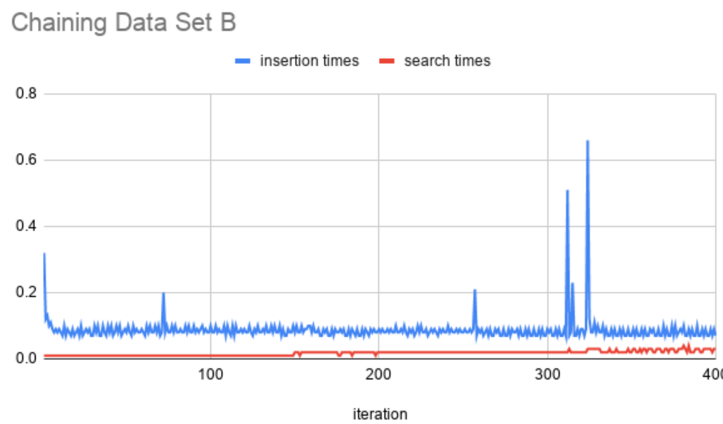


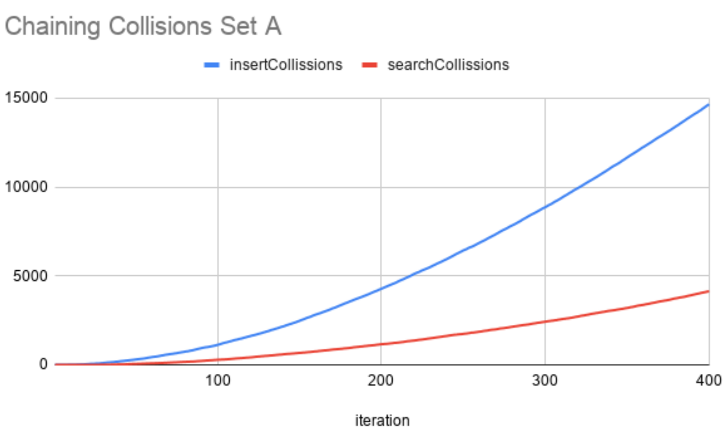
Hash Table Findings

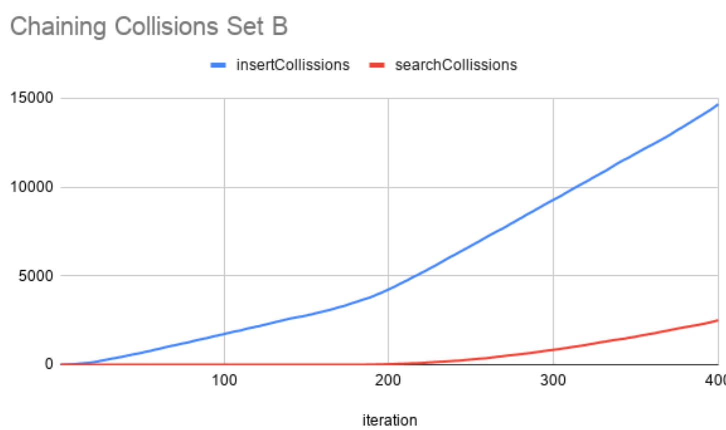
*Note: we tracked how many insert or search calls resulted in a collision, not how many collisions happened within each of those calls. Given more time, we would have been able to implement this on a more granular level and extract even better data

Chaining Collision Resolution --



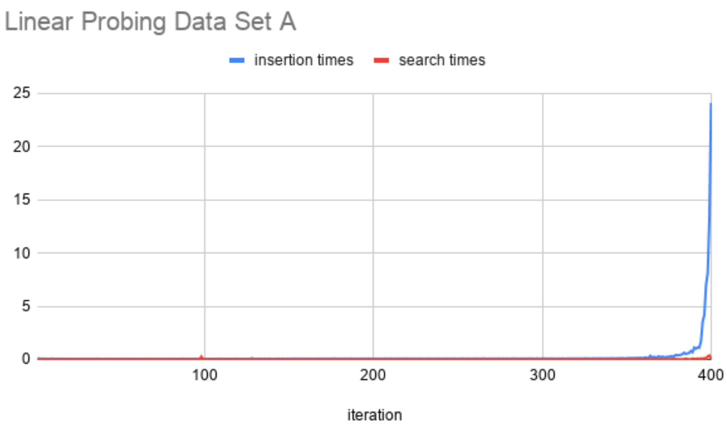


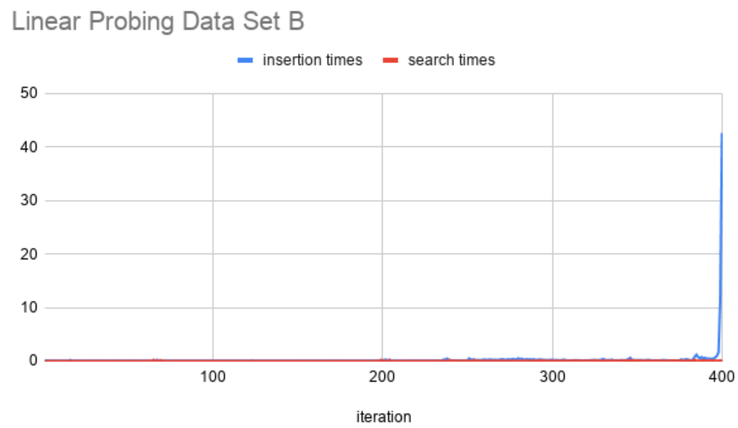


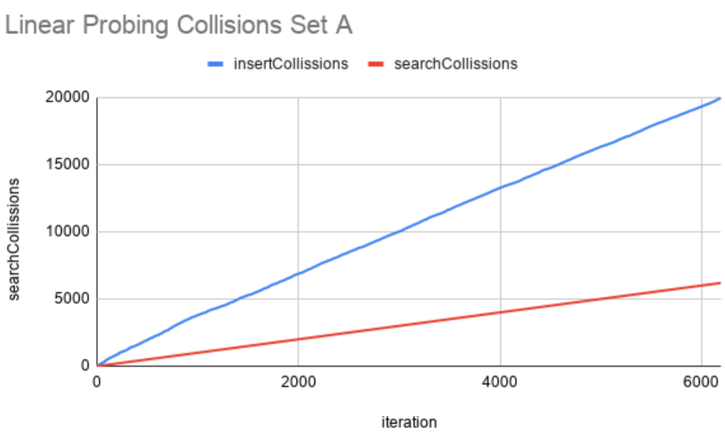


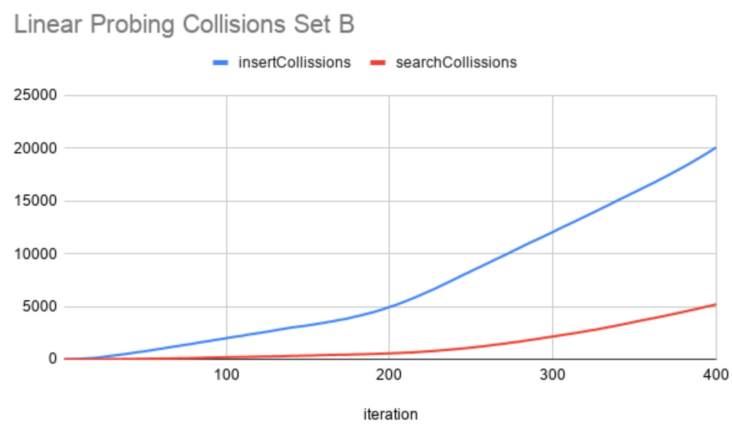
Linear Collision Resolution --

*Note: Be sure to observe the scale differences on the y-axis!









Quadratic Collision Resolution --

*Note: Be sure to observe the scale differences on the y-axis!