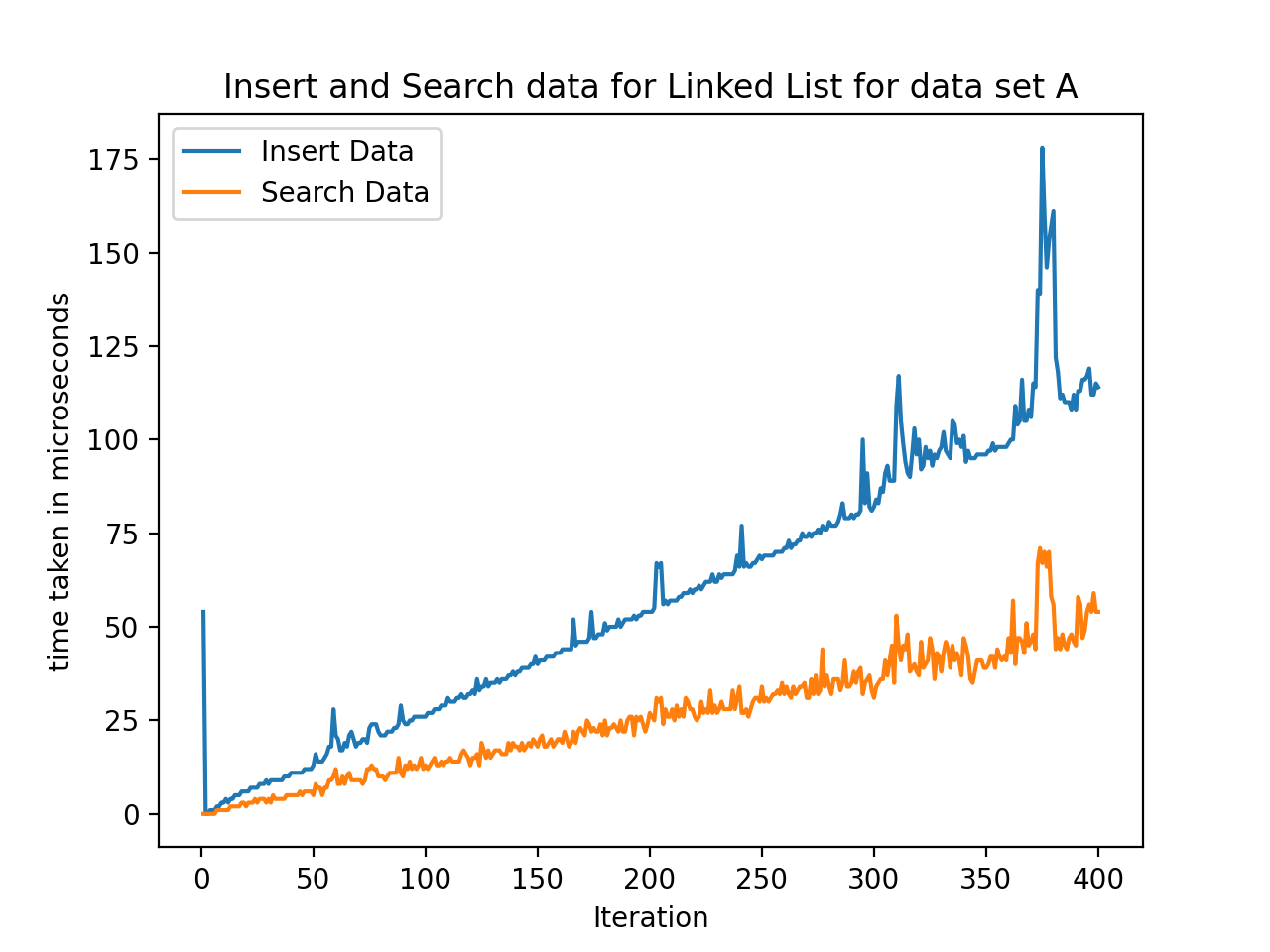
Jack Blackburn

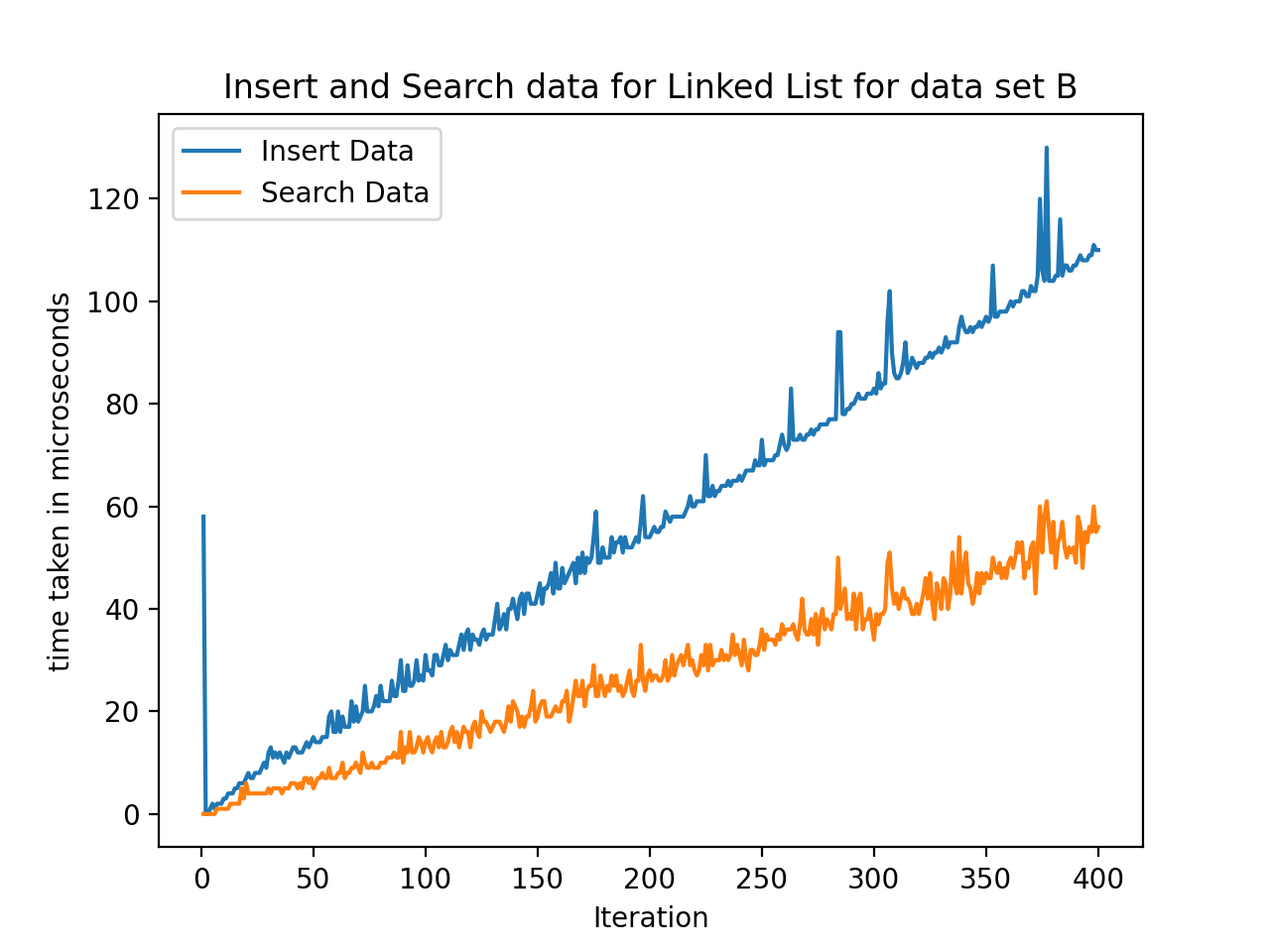
**Save The USPS Report**

For my experiment my results were about what I expected. The Linked List got slower and slower the bigger it got which makes sense because for inserts it has to traverse to the end of the list every time to insert a new element. And for searches it also has to traverse more and more elements to reach the desired node. The Binary Search Tree was much faster than any other data structure I tested at inserts, especially with the random data in data set A, and still faster than everything else in data set B. BST searches weren’t as fast as it’s inserts but still faster than the LL searches and about the same as the hash table search times. For my hash table implementations, the linear probe was the slowest, then quadratic, then the chaining. Linear Probing saw collisions increase steadily the bigger the table got but insert times stayed steady no matter how many collisions there were. Quadratic probing however, saw insert times skyrocket near the end of the inserts for Data Set A. My chaining tests saw steady insert and search times no matter the amount of collisions.

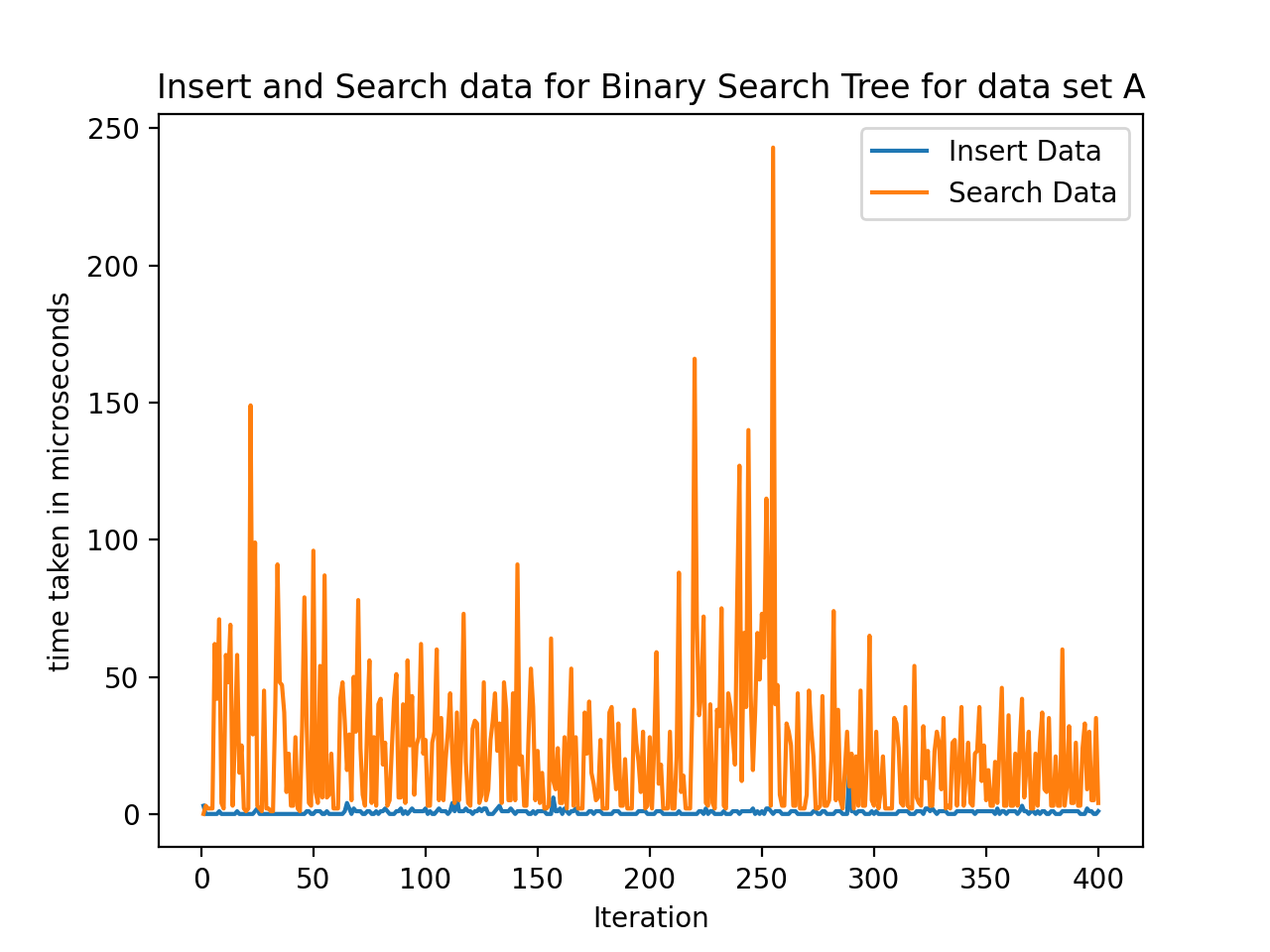
The best data structure performance for this experiment was the BST. The BST had excellent insert times when the data set was randomized and only a little bit slower when the data was organized.The search times didn’t stand out as much as the insert times but they were also very good overall. The hash table with chaining had similar search times as the BST but it’s insert times were much slower, so the hash table with chaining was a close second to the BST.



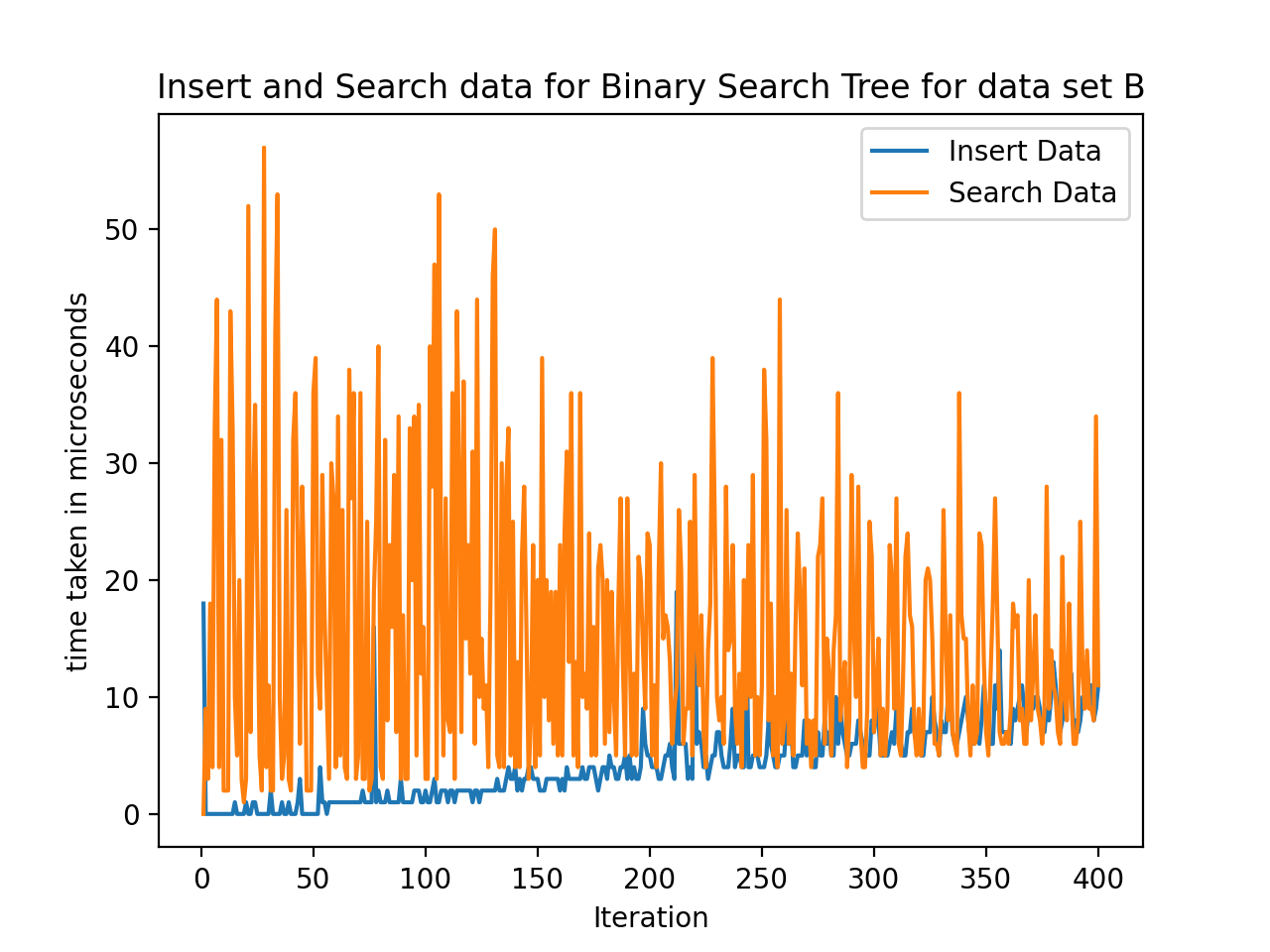
**Figure 1: Average insertion and search times for a Linked List with Data Set A**

****

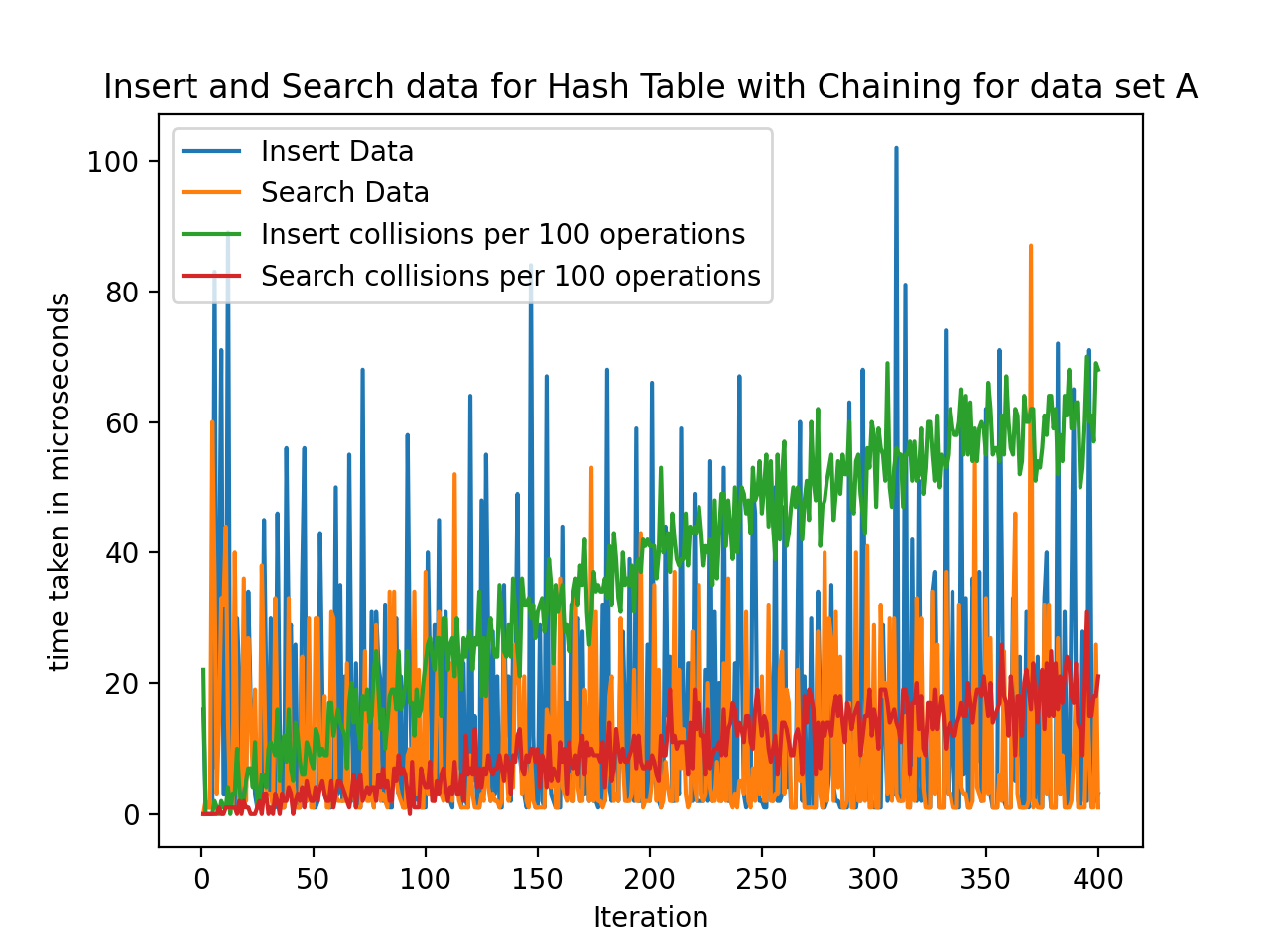
**Figure 2: Average insertion and search times for a Linked List with Data Set B**

****

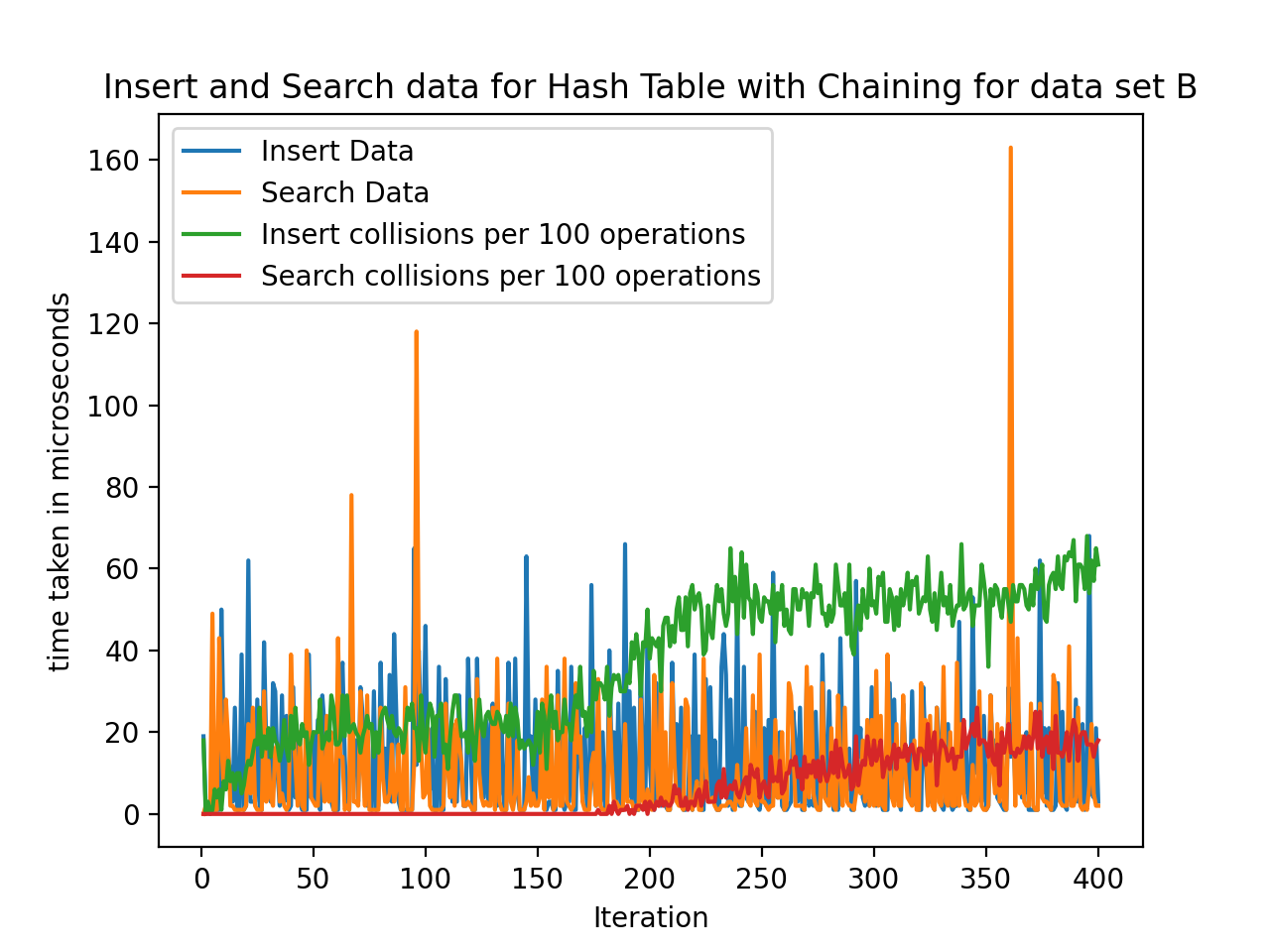
**Figure 3: Average insertion and search times for a Binary Search Tree with Data Set A**

****

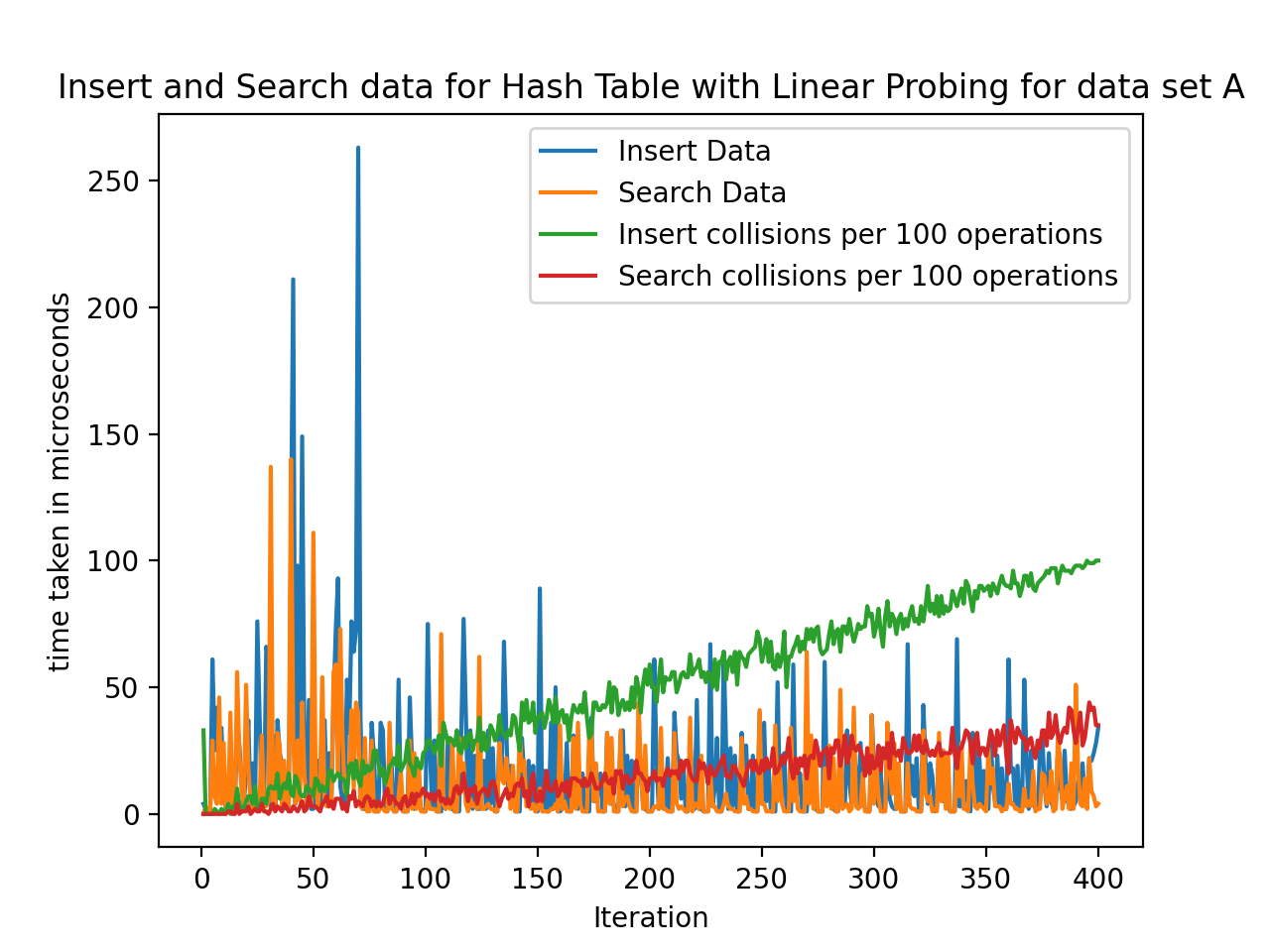
**Figure 4: Average insertion and search times for a Binary Search Tree with Data Set B**

****

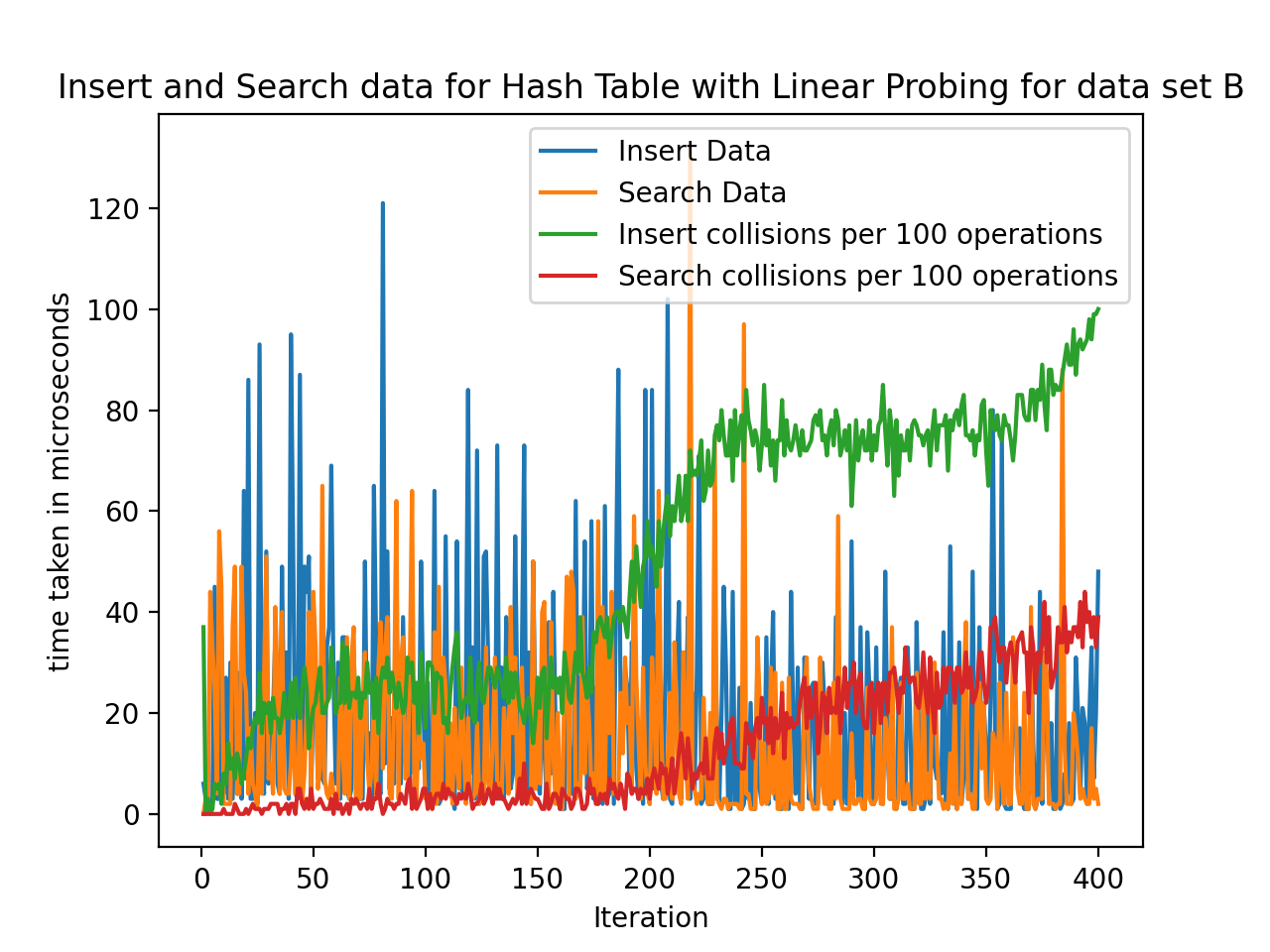
**Figure 5: Average insertion and search times for a Hash Table using Chaining with Data Set A**

****

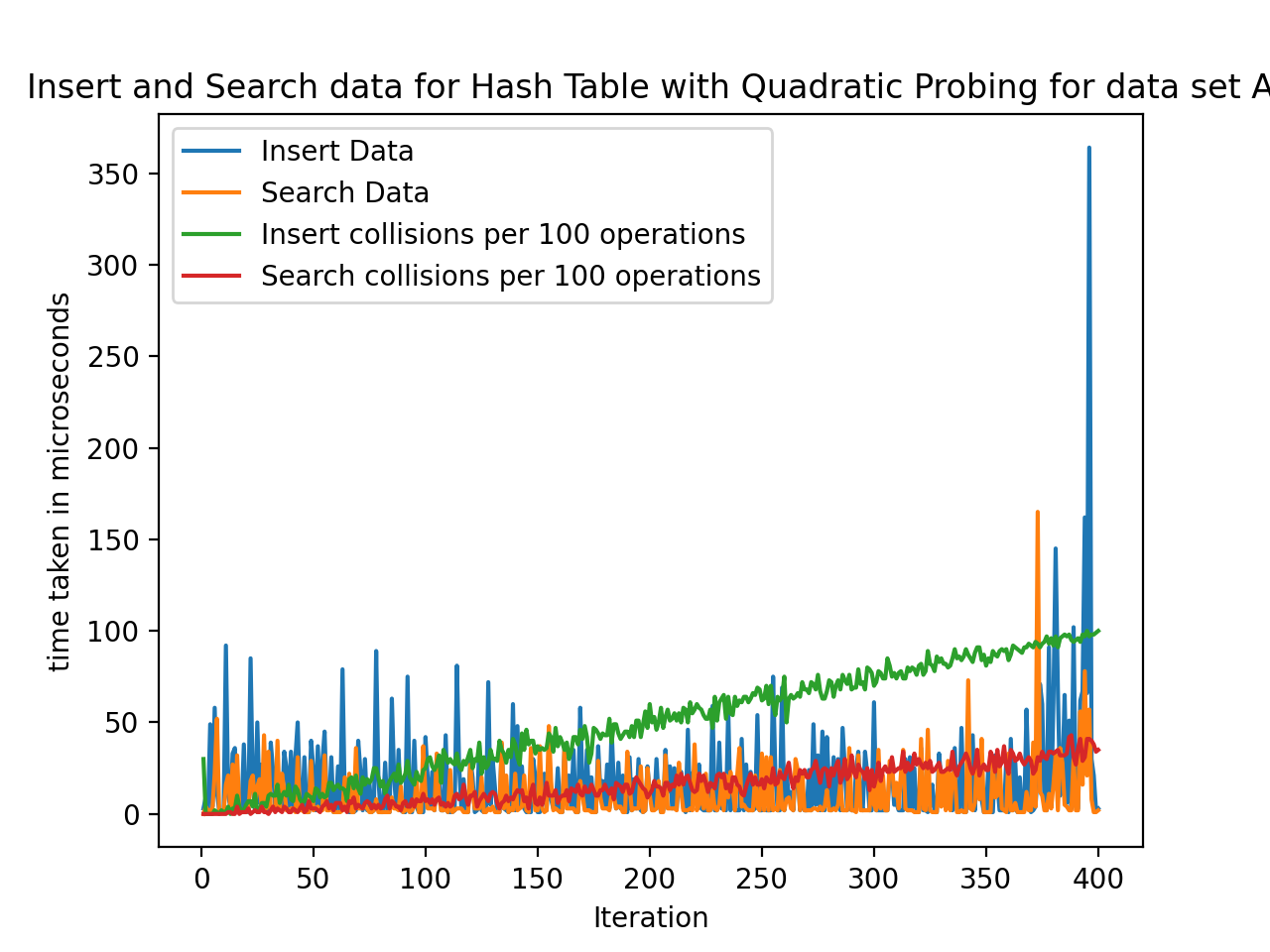
**Figure 6: Average insertion and search times for a Hash Table using Chaining with Data Set B**

****

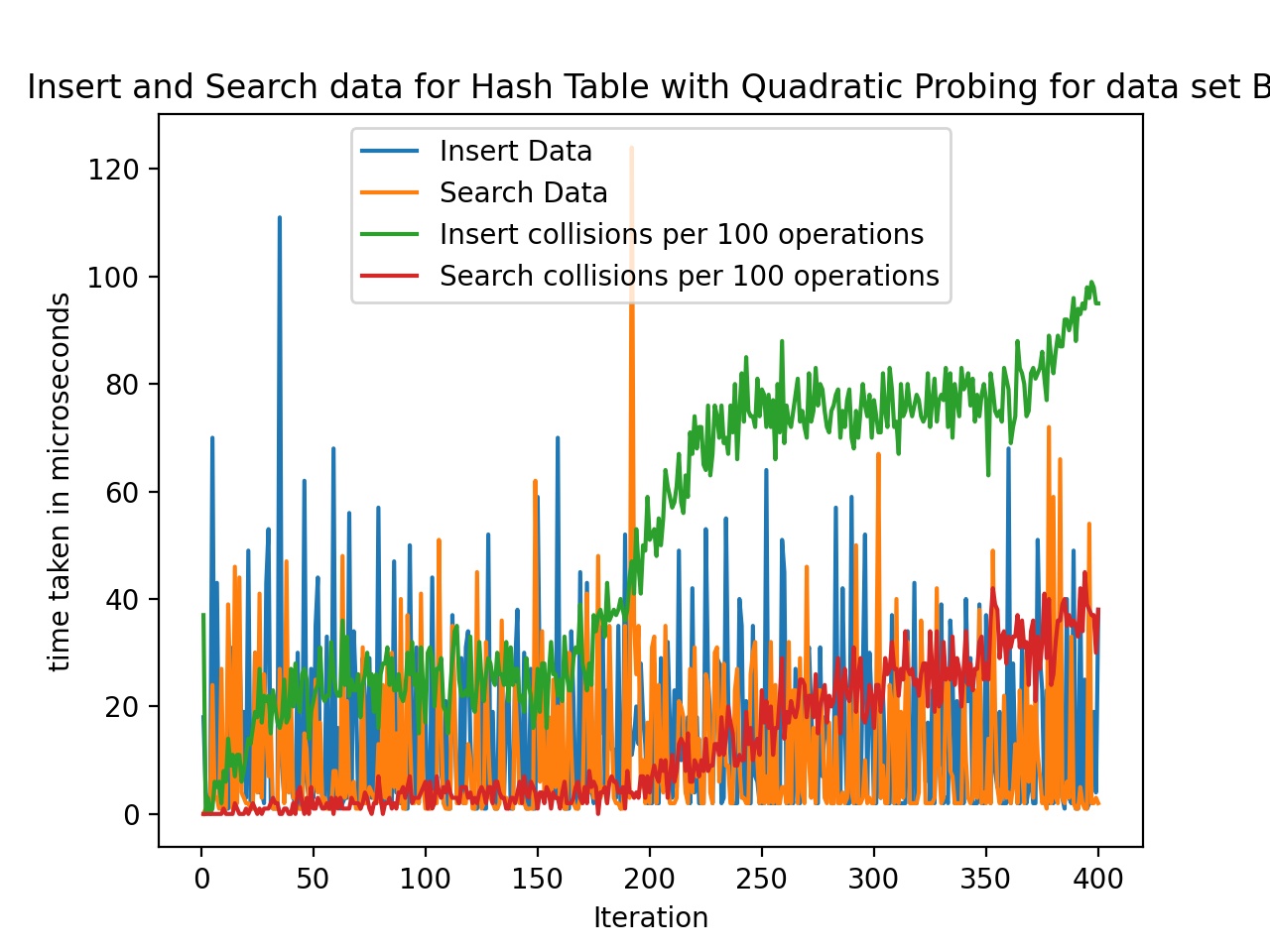
**Figure 7: Average insertion and search times for a Hash Table using Linear Probing with Data Set A**

****

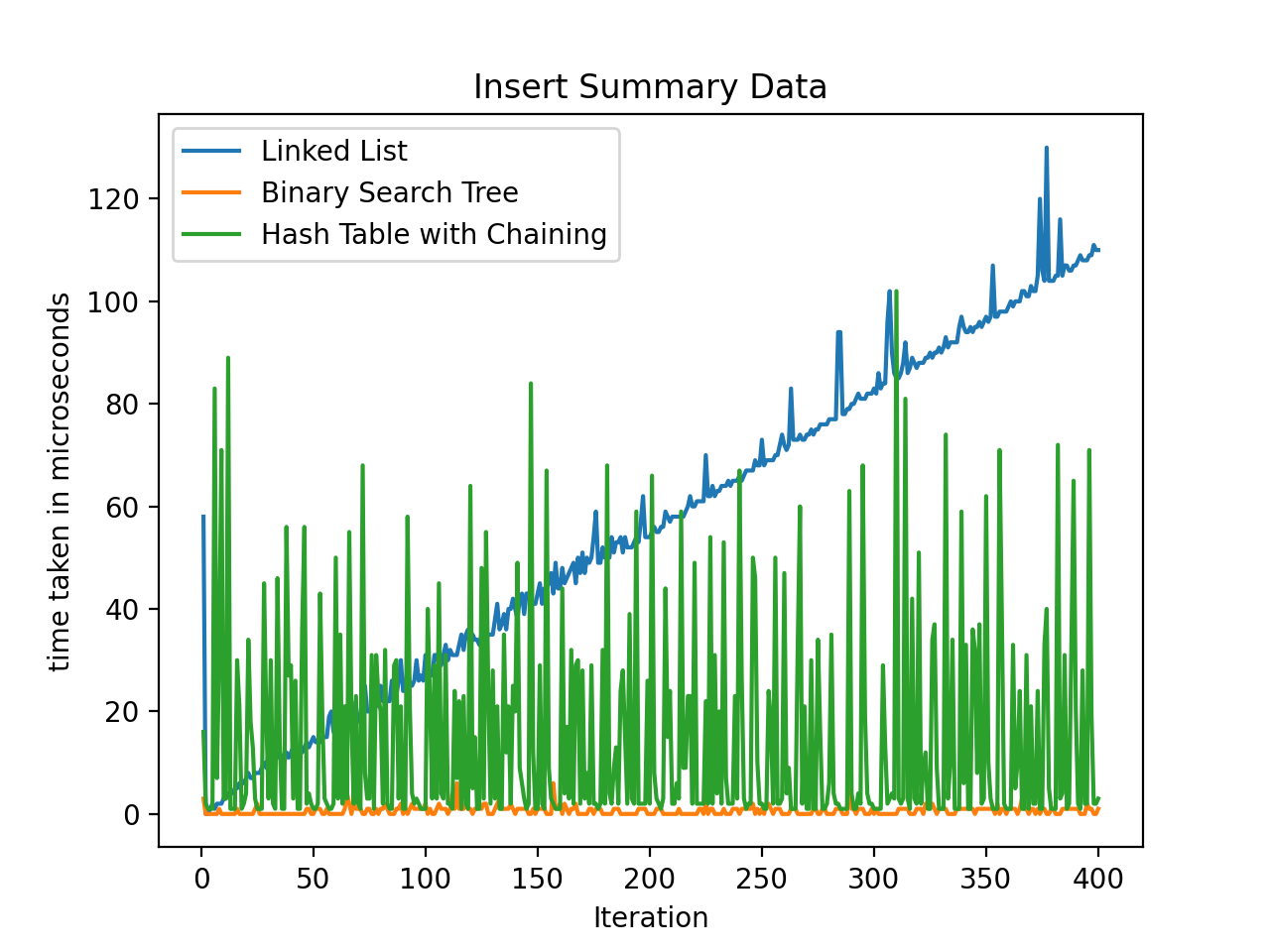
**Figure 8: Average insertion and search times for a Hash Table using Linear Probing with Data Set B**

****

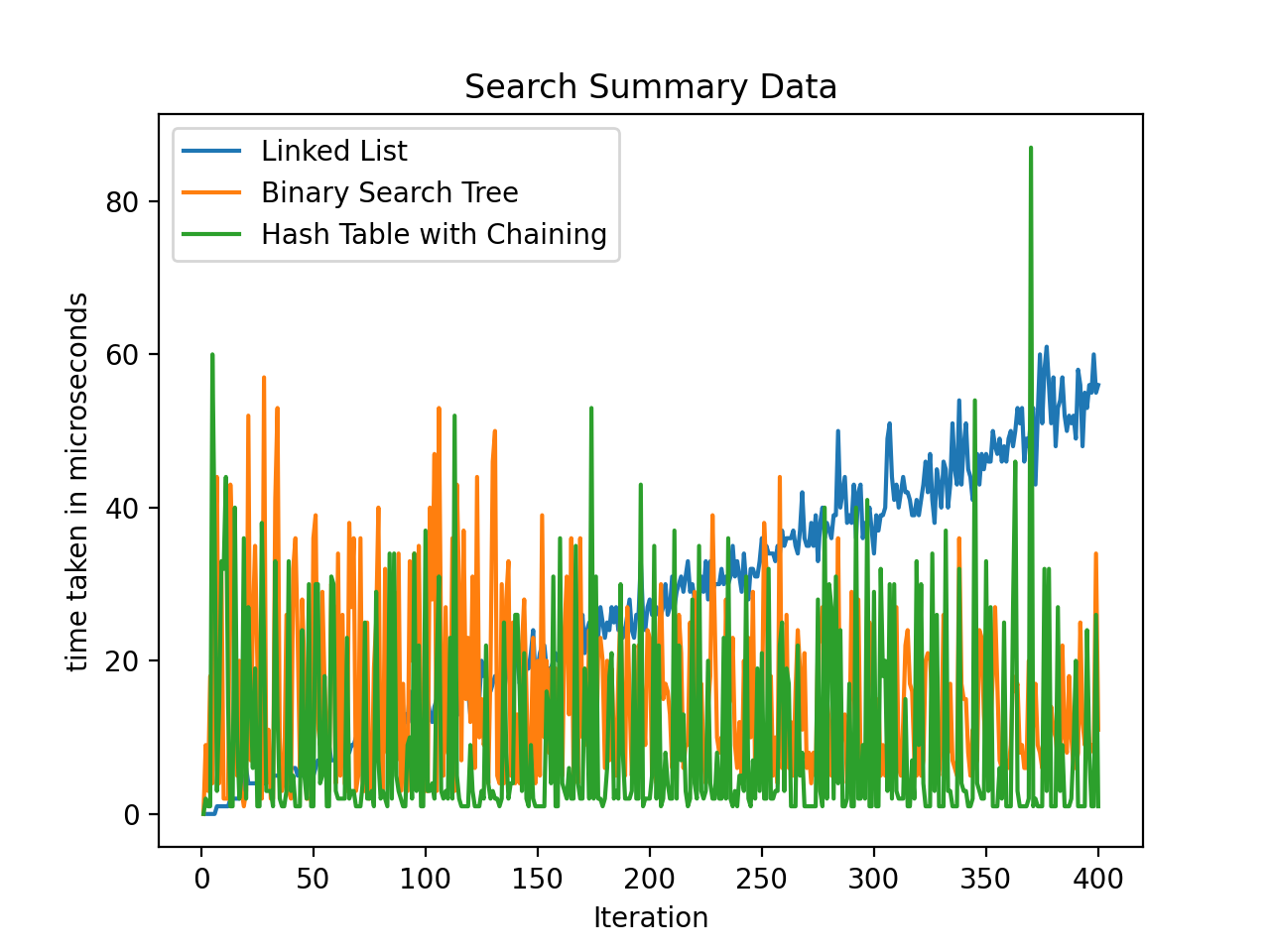
**Figure 9: Average insertion and search times for a Hash Table using Quadratic Probing with Data Set A**

****

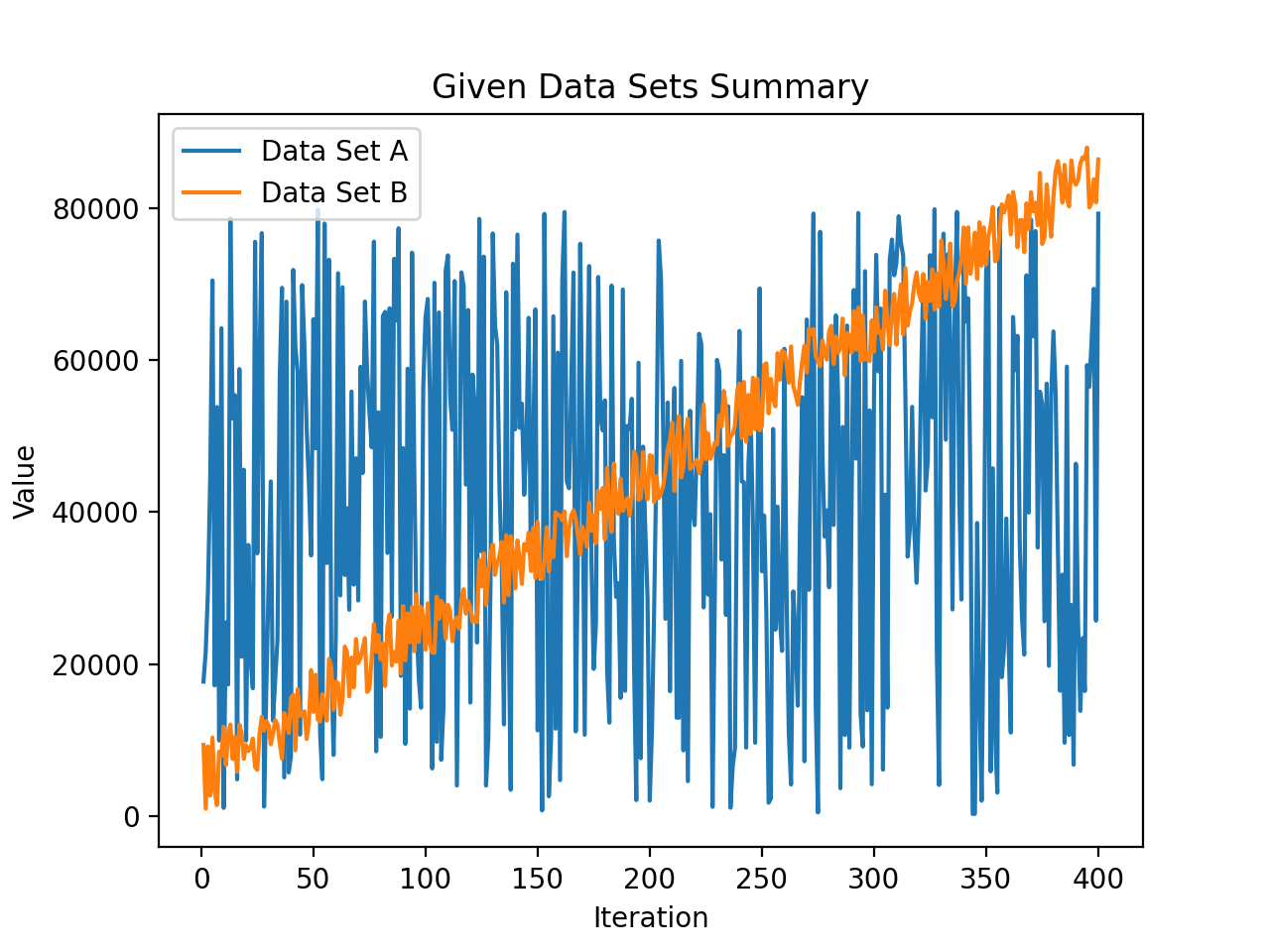
**Figure 10: Average insertion and search times for a Hash Table using Quadratic Probing with Data Set B**

****

**Figure 11: Insert Summary Data with Binary Search Tree, Linked List, and Hash Table with Chaining**

****

**Figure 12: Search Summary data for Linked List, Binary search Tree, and Hash Table with Chaining**

****

**Figure 13: Data Set A and B placed into a graph with every 100th value plotted**