Michael Gross & Matthew Zarifa

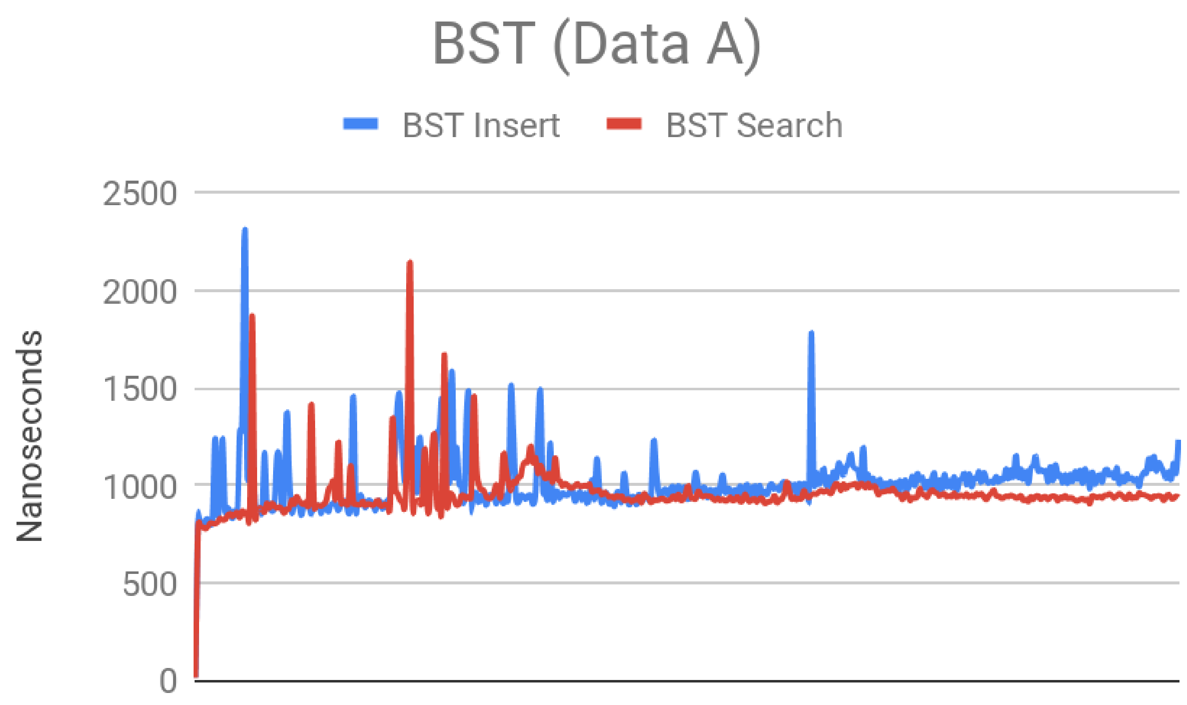
CSCI 2270

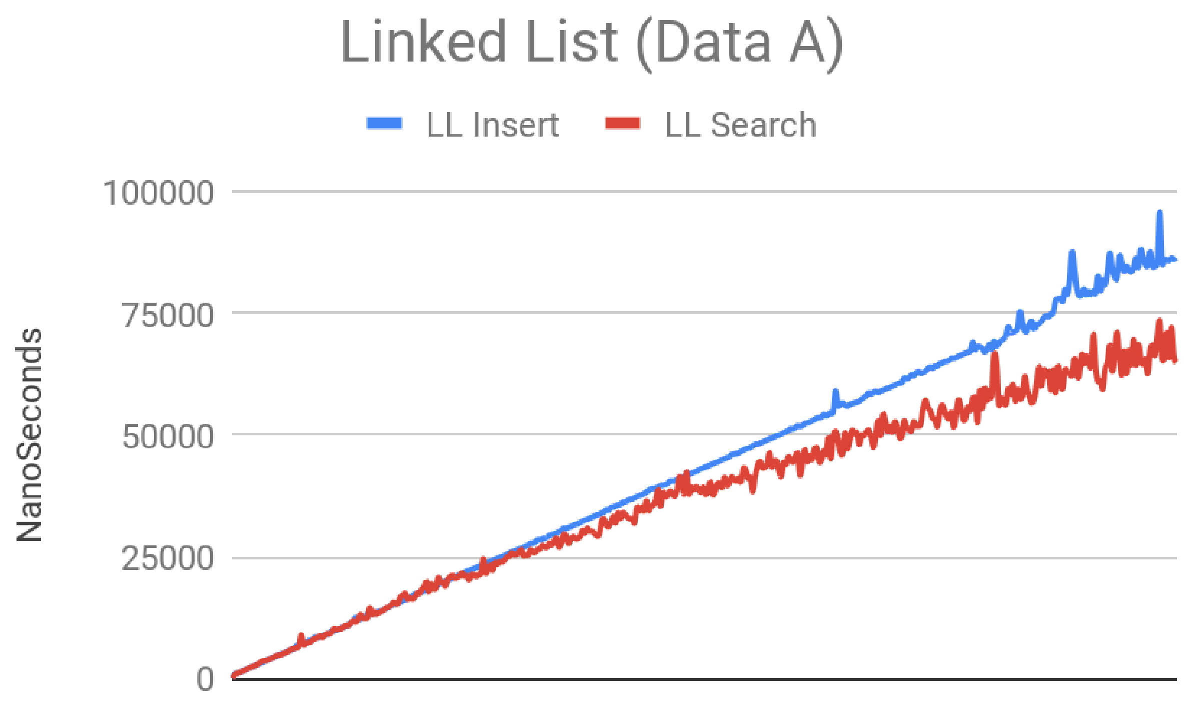
Final Project Report

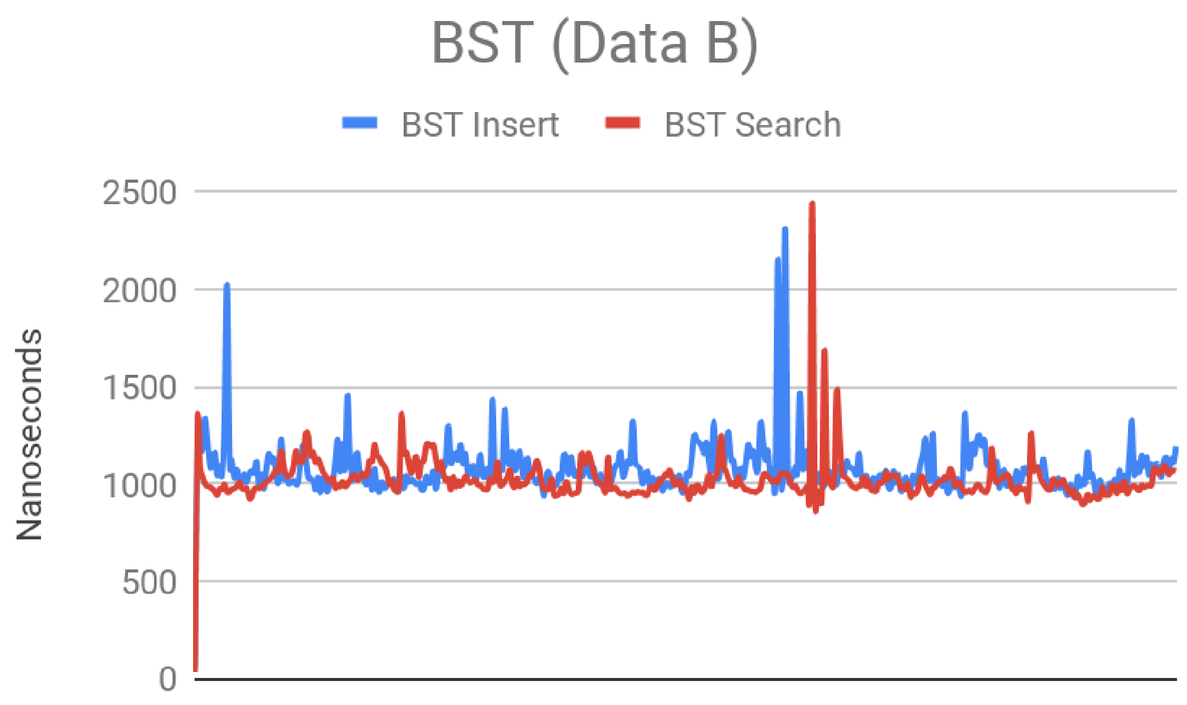
The figures showed the fundamental expectations for the times for a linear hash, chain hash, quadratic hash, BST, and a linked list. The linked list showed the worst time due to it being O(n) time, so it linearly increases and results in a very large time. In contrast, a quadratic hash was consistently lower on time for inserts (avg time: 793 ns) but was average for searches (avg time: 750 ns). Any of the non-hash methods would require a large amount of time to both process data and find specific data, so while their implementation is easy, the payoff is substantially worse. The best method would be the Binary Search Tree method out of the two, as it had consistently lower times when compared to the Linked List. However, both fell short of the hash-based storage methods, the overall best storage method seeming to be the quadratic hash function.

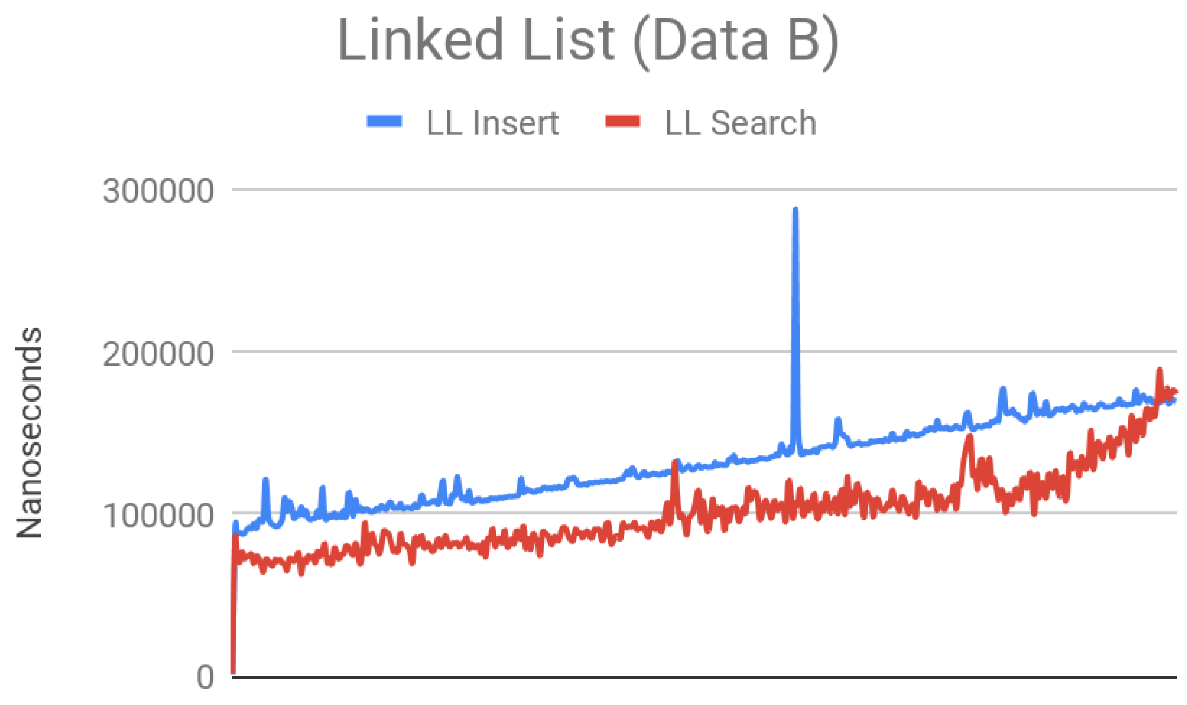
In general, all graphs showed random spikes in time, but all showed consistent trends. These random spikes in time can be attributed to CPU clock speeds and processes running at that time. Times were different between the two group members when running the same data sets, and even the data sets themselves would output different times from one execution to another. This also can be credited to the extremely small time interval being used (nanoseconds).

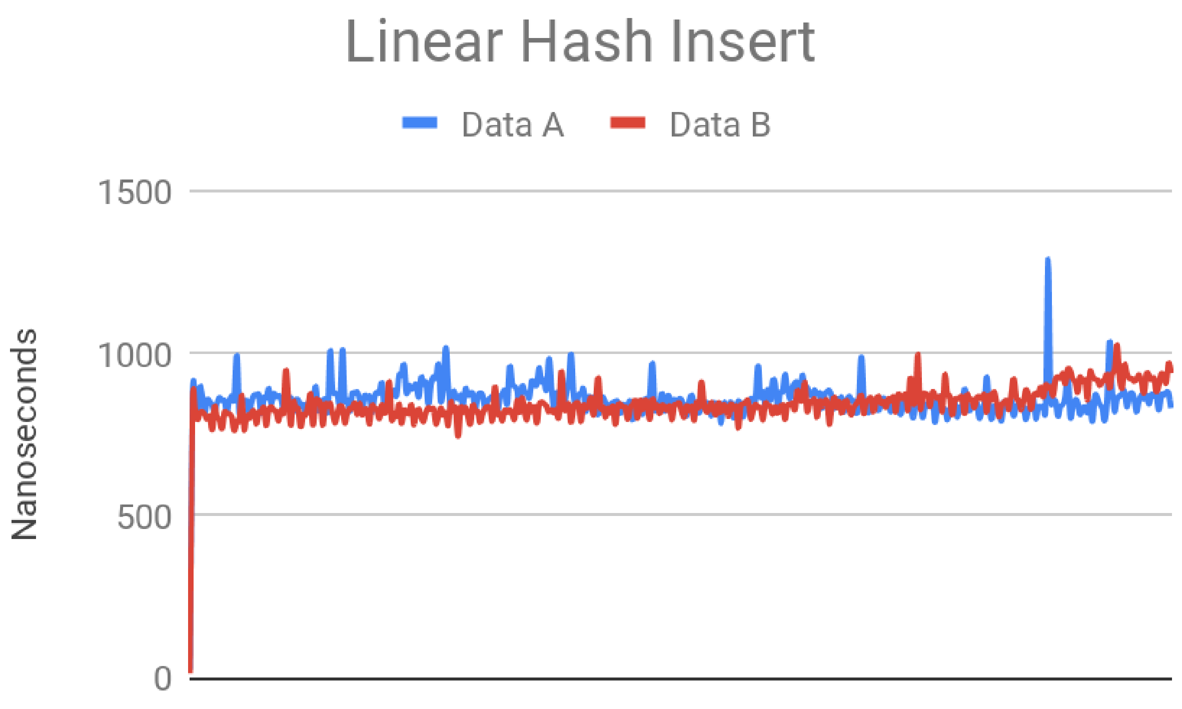
All data structures except the linked list proved similar times to the quadratic hashing. The linked list method underperformed significantly. This is due to its O(n) run time, compared to O(1) or O(log) time that the other tested data structures have (see figures below).

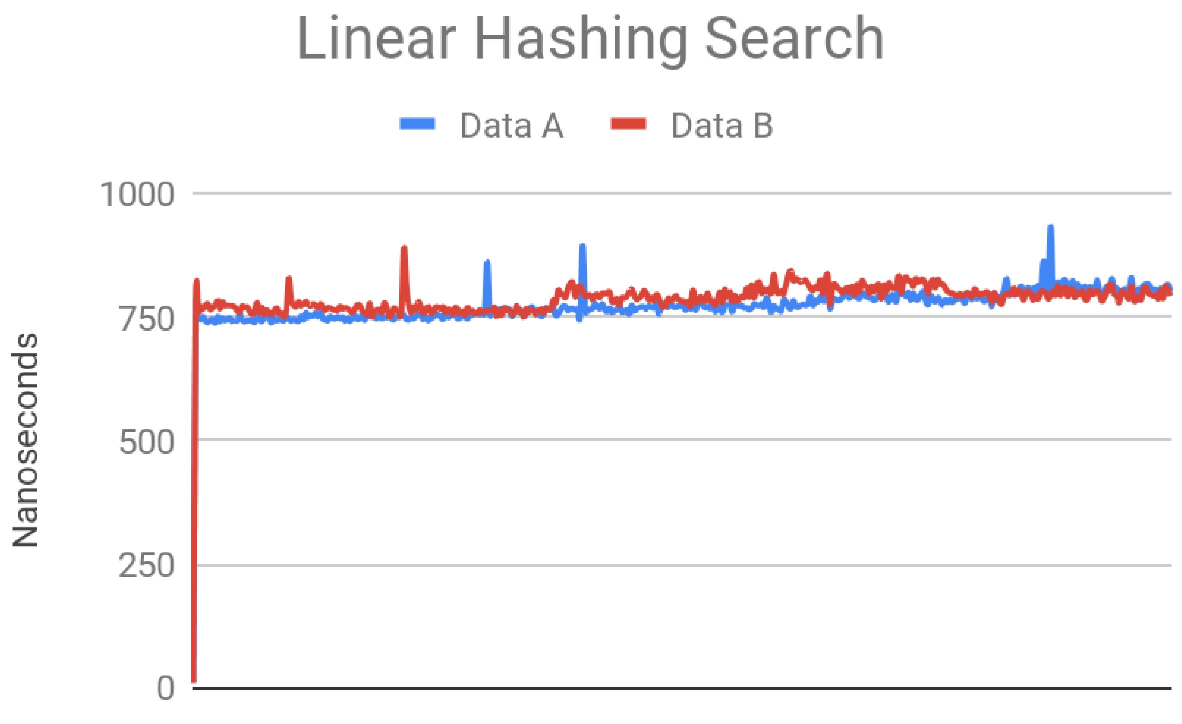

BST (Data A)
BST Insert
BST Search
2500
2000
1500
1000
500
0
Nanoseconds

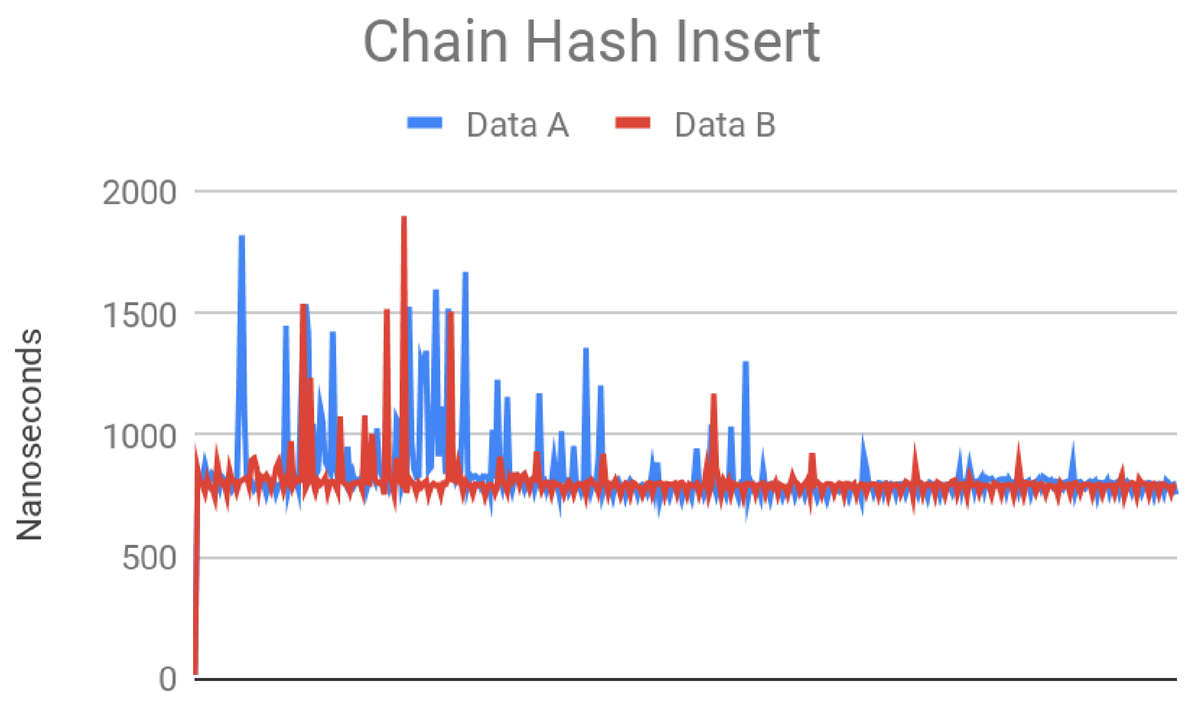

Linked List (Data A)
LL Insert
LL Search
100000
75000
50000
25000
0
NanoSeconds

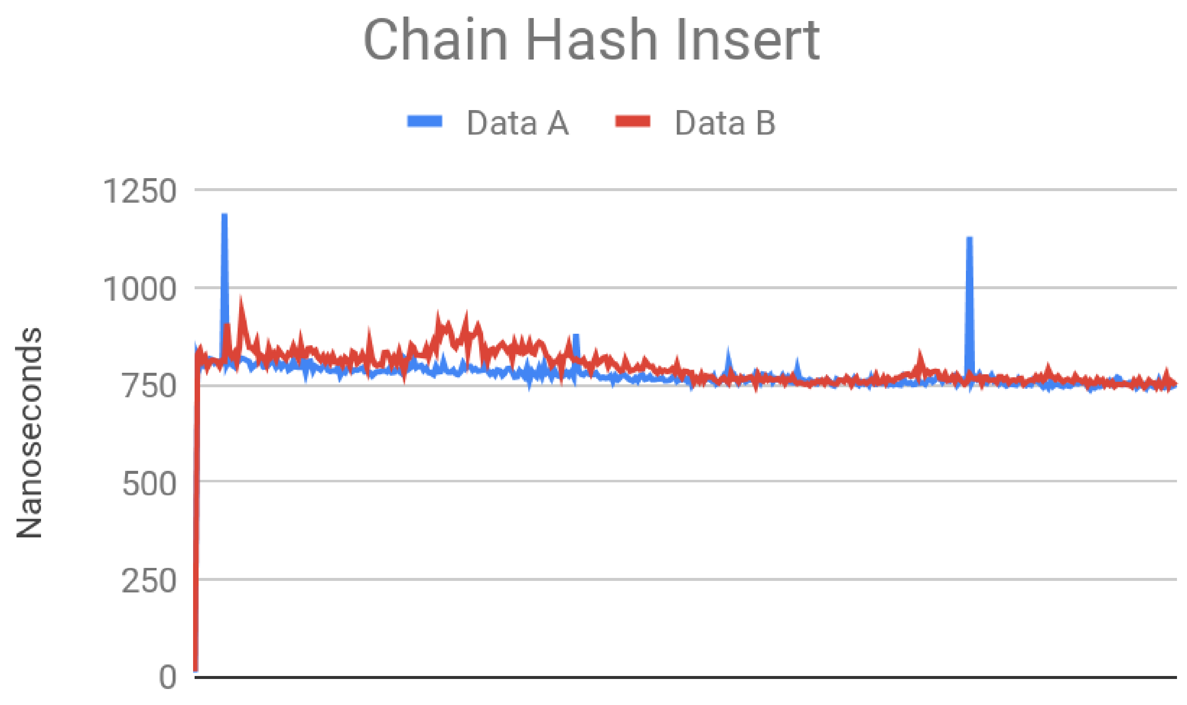

BST (Data B)
BST Insert
BST Search
2500
2000
1500
1000
500
0
Nanoseconds

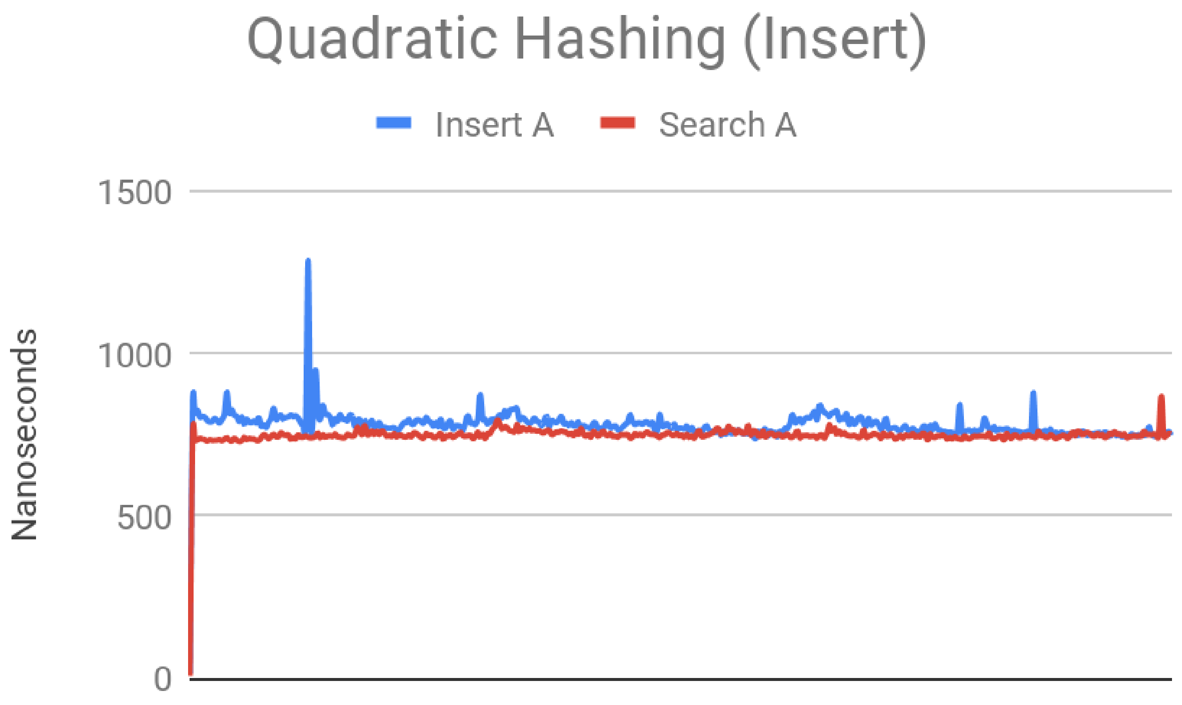

Linked List (Data B)
LL Insert
LL Search
300000
200000
100000
0
Nanoseconds

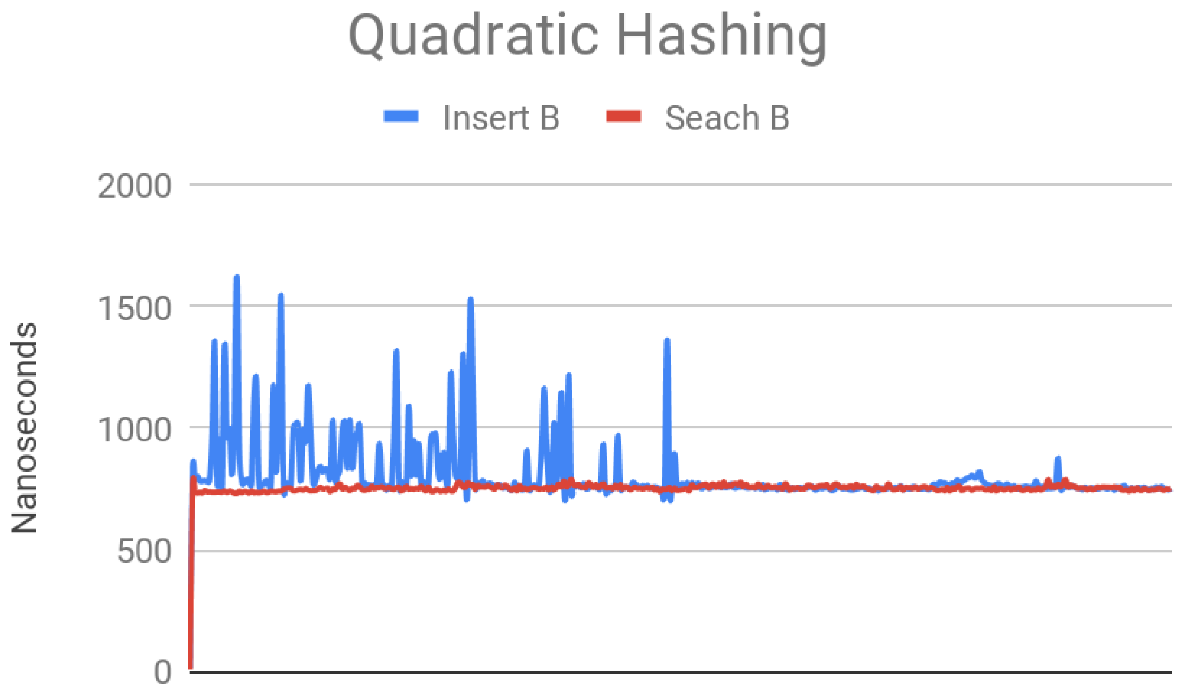

Linear Hash Insert
Data A
Data B
1500
1000
500
0
Nanoseconds


Linear Hashing Search
Data A
Data B
1000
750
500
250
0
Nanoseconds


Chain Hash Insert
Data A
Data B
2000
1500
1000
500
0
Nanoseconds


Chain Hash Insert
Data A
Data B
1250
1000
750
500
250
0
Nanoseconds


Quadratic Hashing (Insert)
Insert A
Search A
1500
1000
500
0
Nanoseconds


Quadratic Hashing
Insert B
Seach B
2000
1500
1000
500
0
Nanoseconds