SAVE THE USPS

CSCI 2270 Spring 2020 Final Project

Julia Troni and Annikka Turmala

CSCI 2270 CU Boulder Spring 2020

TAs: Archana Anand, Abhidip Bhattacharyya

DATA

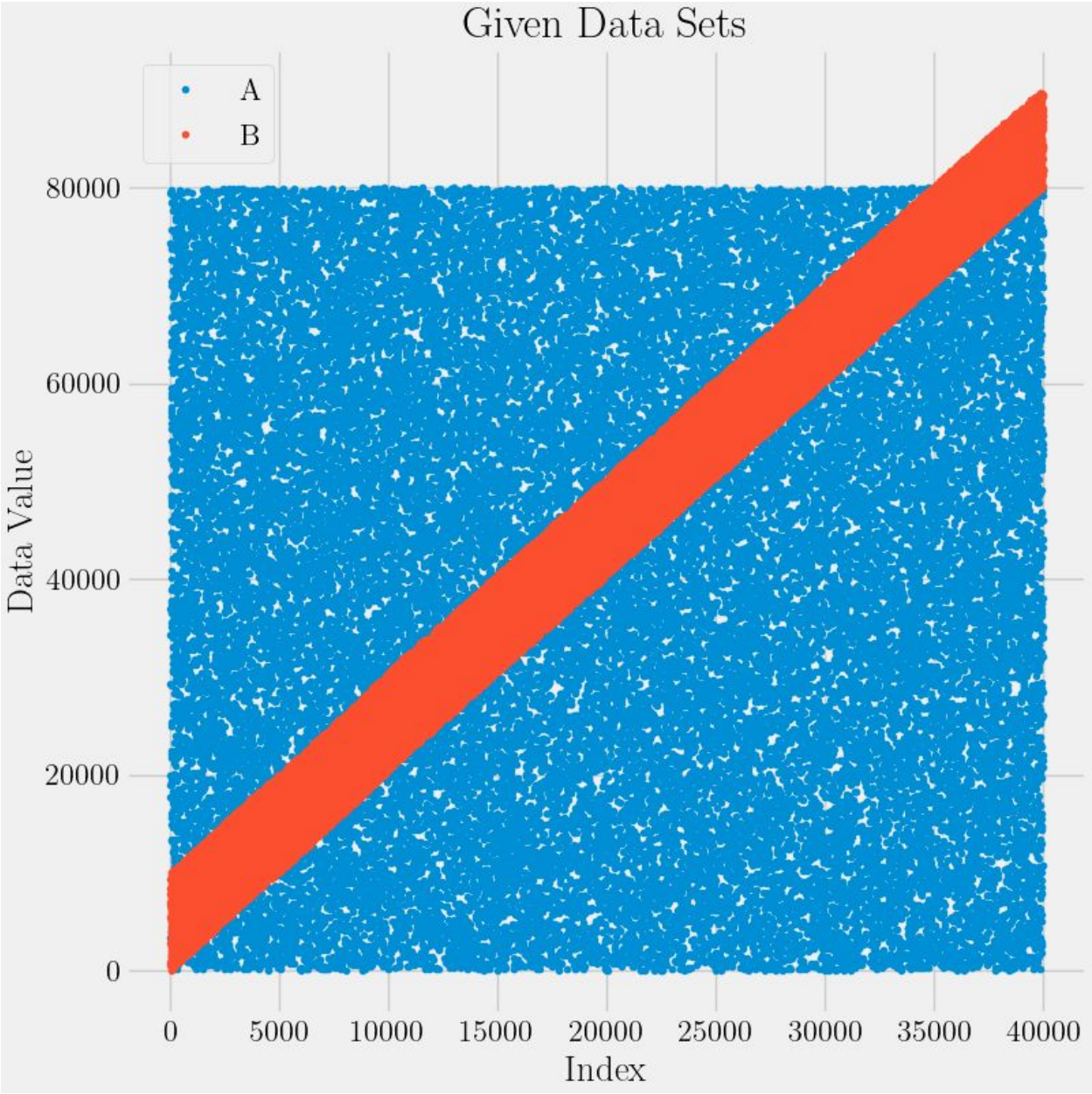

Figure 1

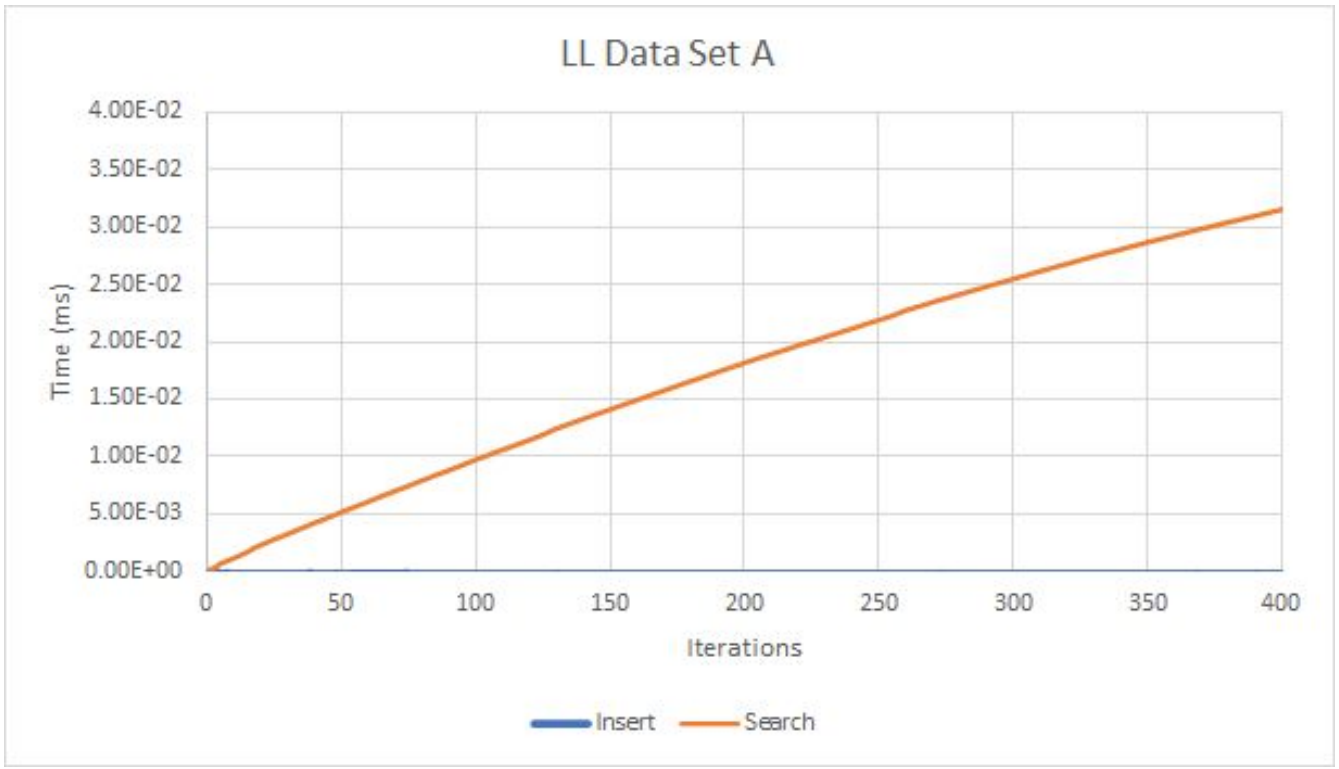

Figure 2

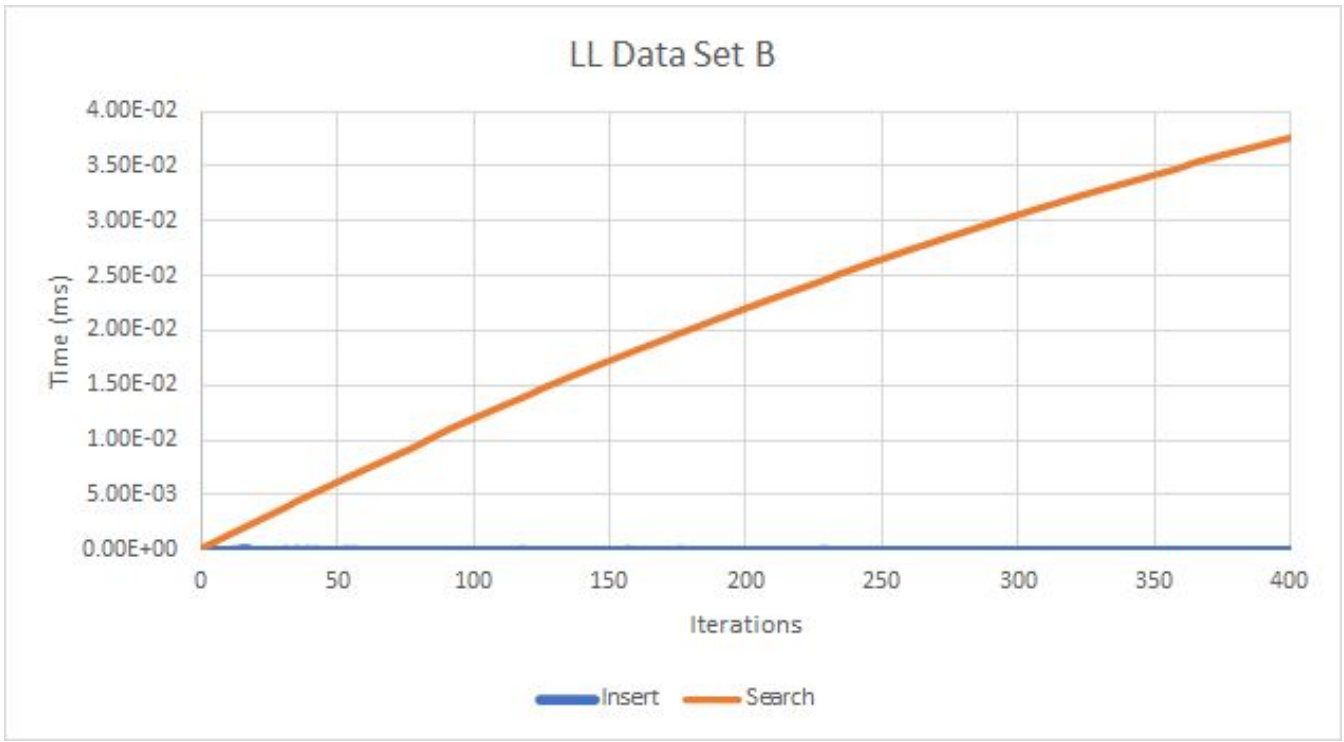

Figure 3

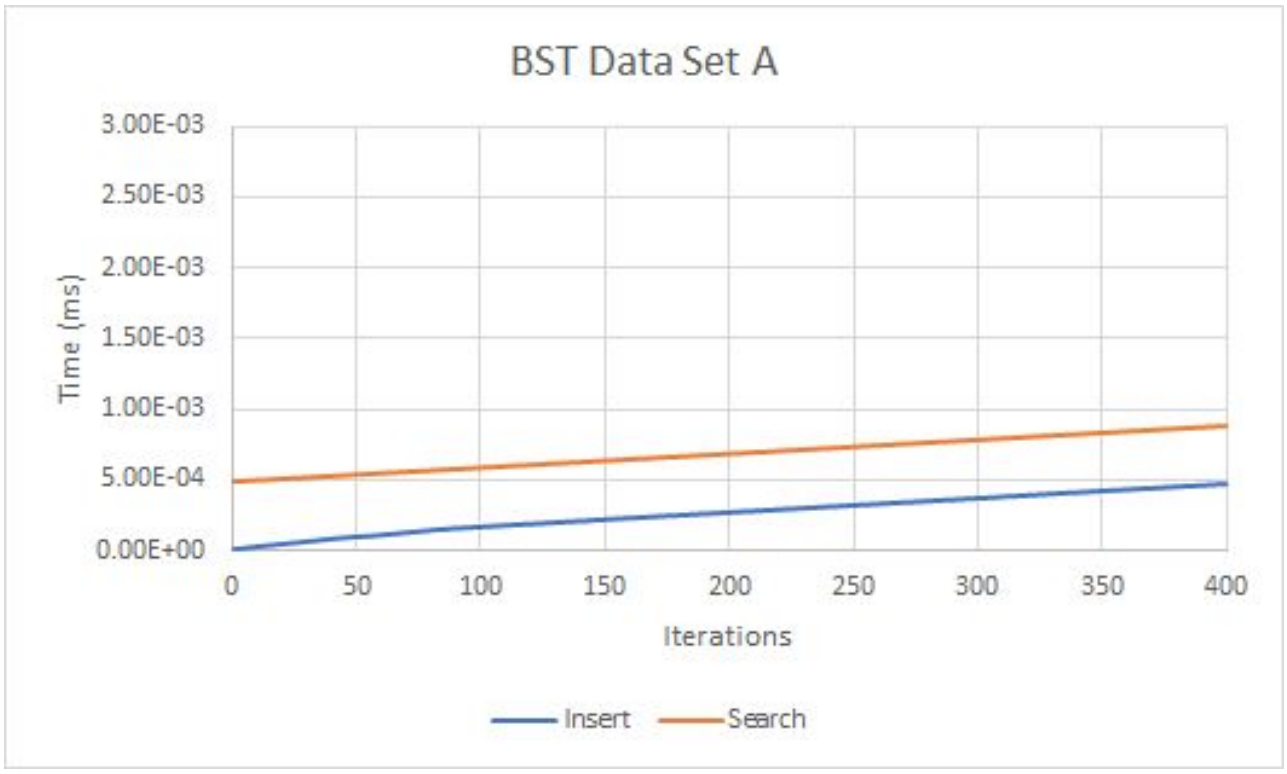
BST Data Set A
3.00E-03
2.50E-03
2.00E-03
1.50E-03
1.00E-03
5.00E-04
0.00E+00
Time (ms)
0
50
100
150
200
250
300
350
400
Iterations
Insert
Search

Figure 4


BST Data Set B
3.00E-03
2.50E-03
2.00E-03
1.50E-03
1.00E-03
5.00E-04
0.00E+00
Time (ms)
0
50
100
150
200
250
300
350
400
Iterations
Insert
Search

Figure 5

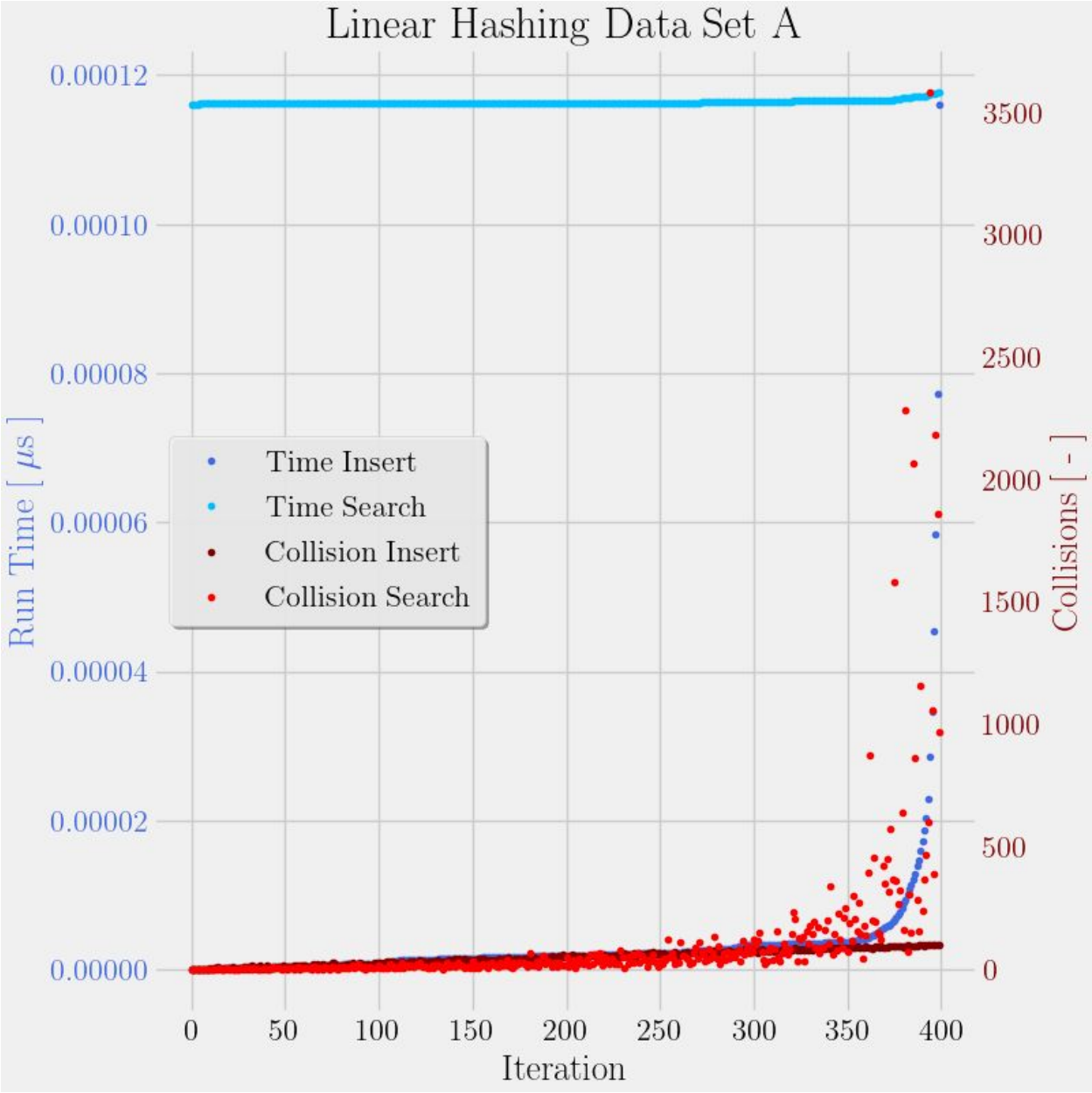

Figure 6

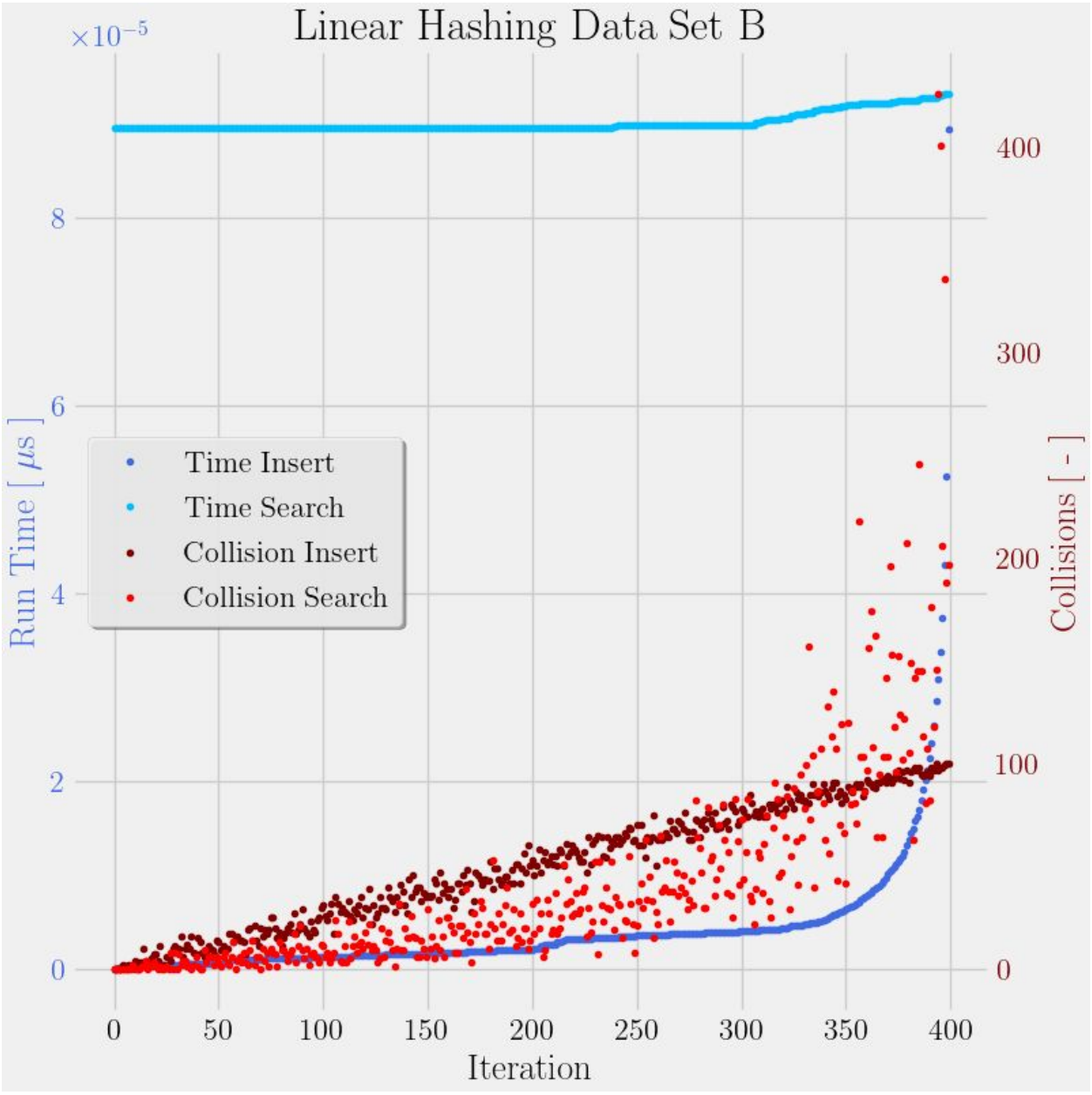
Linear Hashing Data Set B
×10⁻⁵
Run Time [μs]
Collisions [-]
Iteration
Time Insert
Time Search
Collision Insert
Collision Search
0
2
4
6
8
0
100
200
300
400
0
50
100
150
200
250
300
350
400

Figure 7

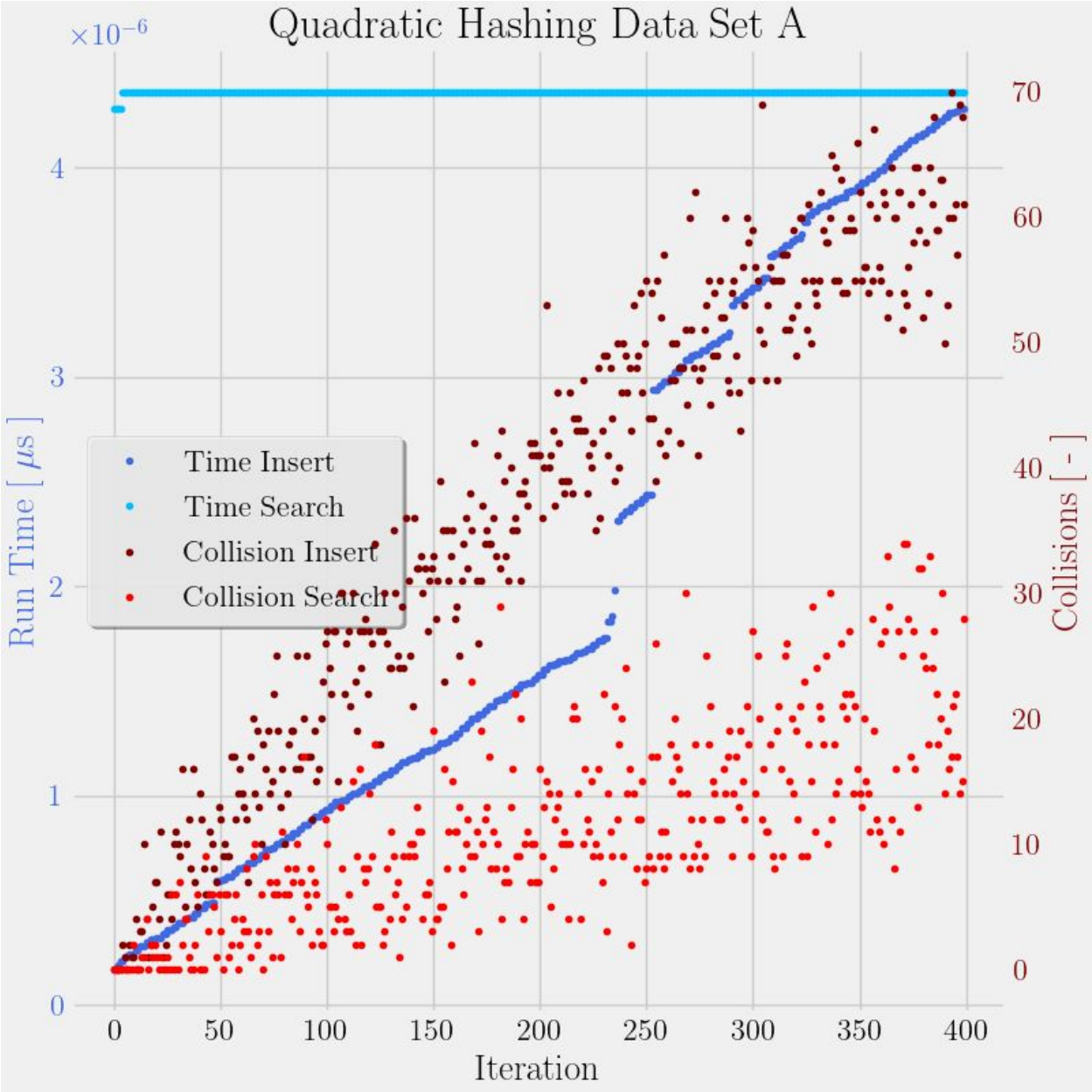
Quadratic Hashing Data Set A
×10⁻⁶
Run Time [μs]
Collisions [-]
Iteration
Time Insert
Time Search
Collision Insert
Collision Search
0
1
2
3
4
0
10
20
30
40
50
60
70
0
50
100
150
200
250
300
350
400

Figure 8

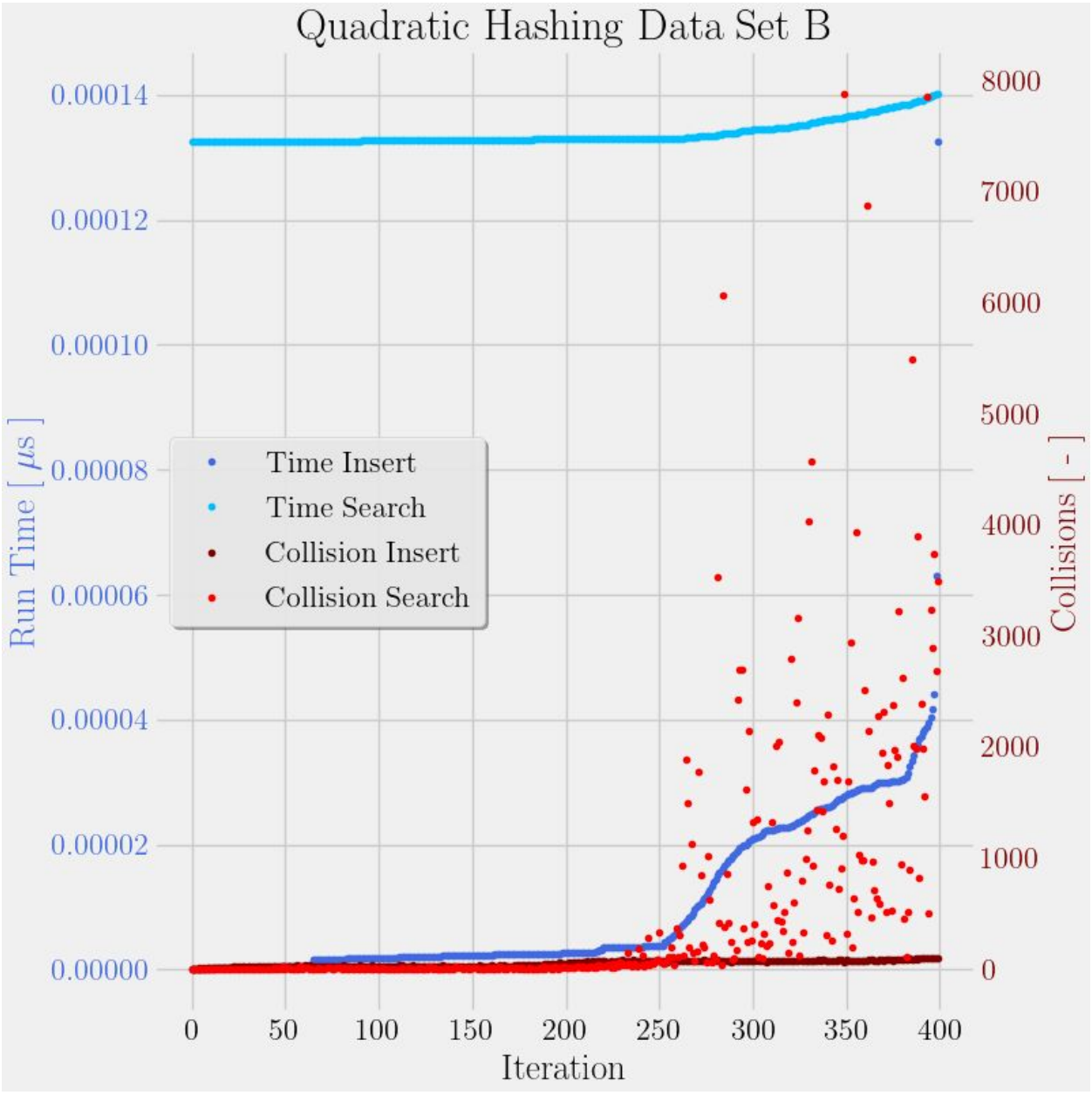
Quadratic Hashing Data Set B
Run Time [μs]
Collisions [-]
Iteration
Time Insert
Time Search
Collision Insert
Collision Search
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250
300
350
400

Figure 9

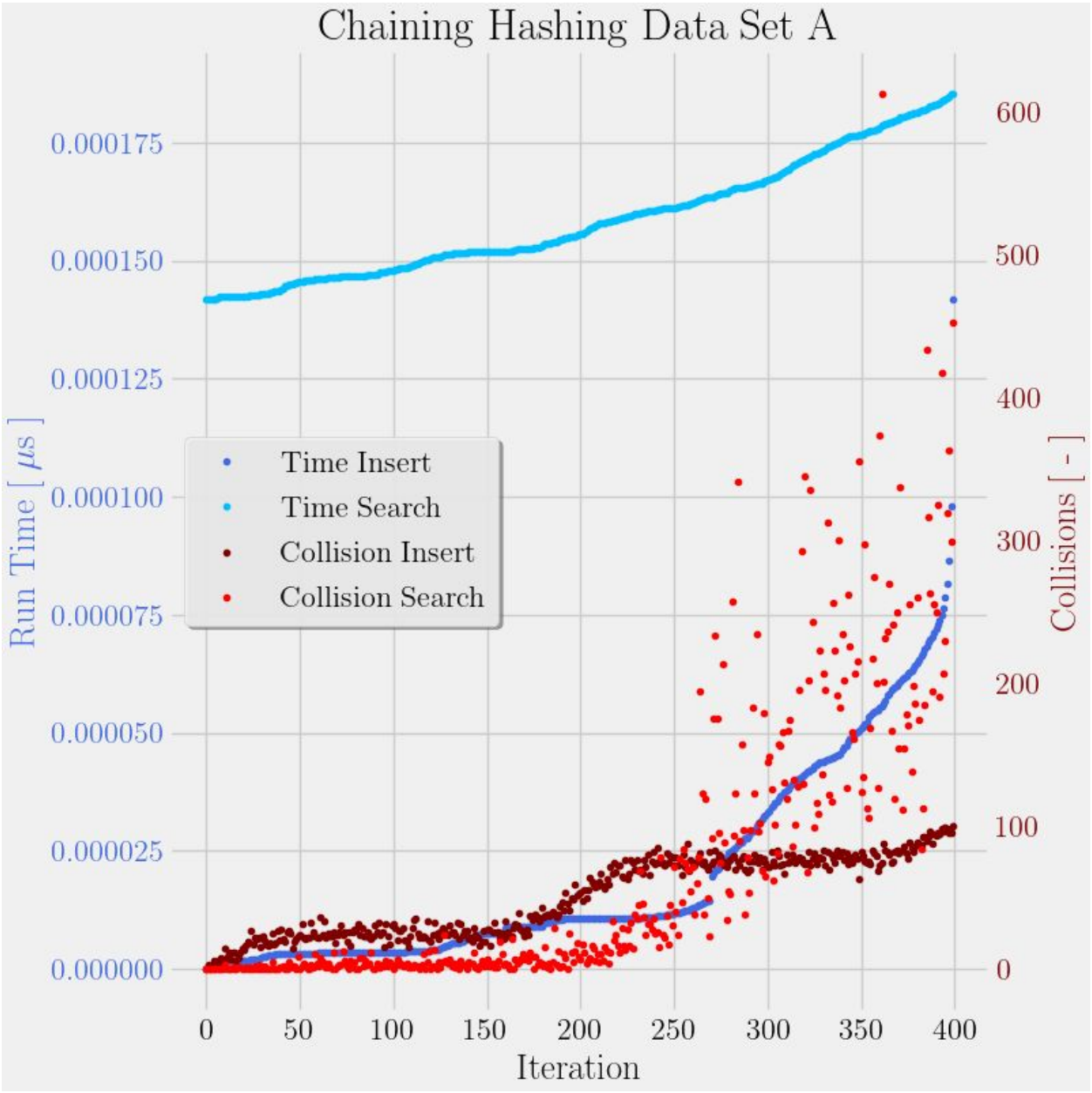
Chaining Hashing Data Set A
Time Insert
Time Search
Collision Insert
Collision Search
Run Time [μs]
Collisions [-]
Iteration
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
600
500
400
300
200
100
0
0
50
100
150
200
250
300
350
400

Figure 10

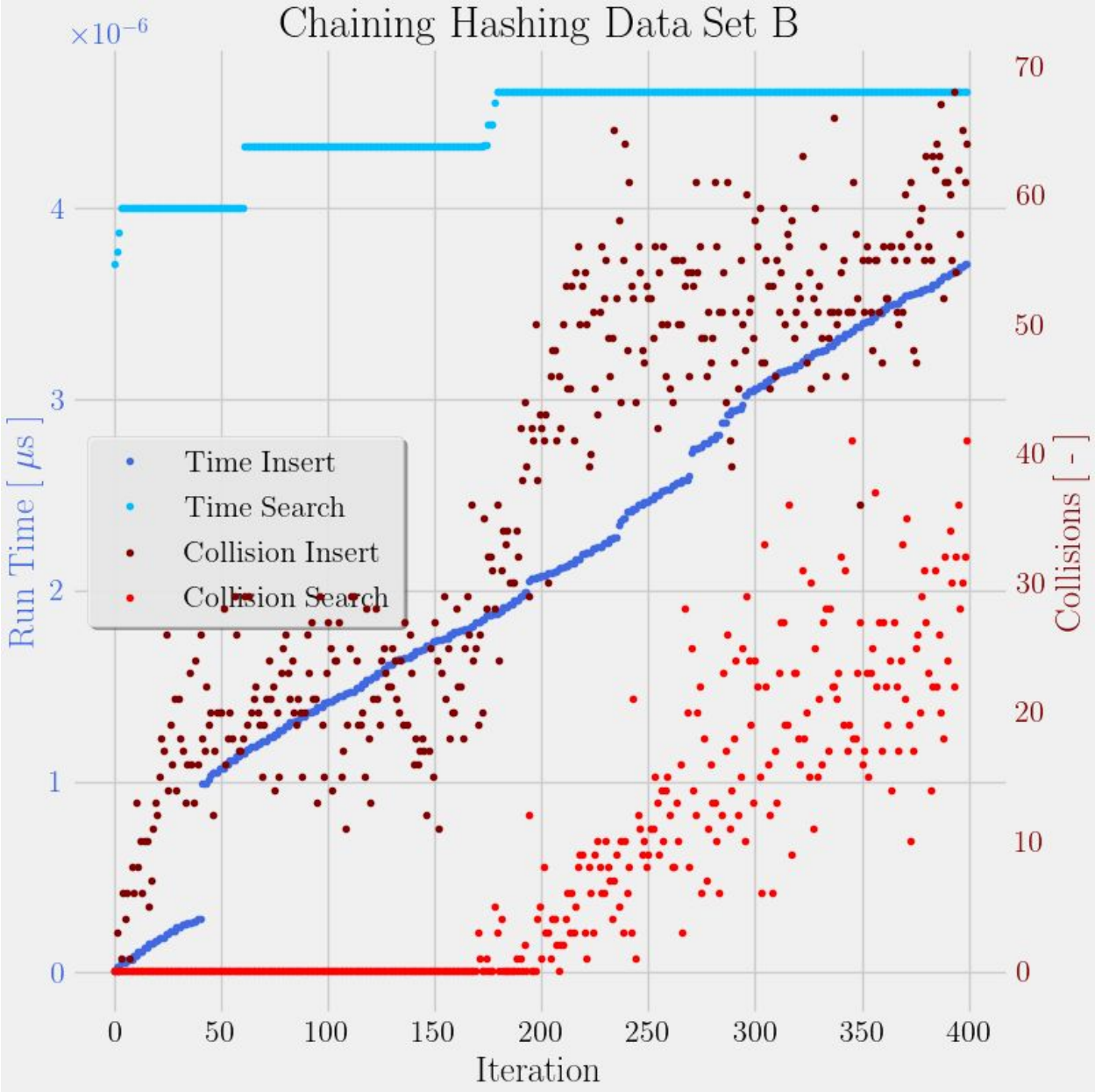
Chaining Hashing Data Set B
×10−6
Run Time [µs]
Collisions [-]
Iteration
Time Insert
Time Search
Collision Insert
Collision Search
0
1
2
3
4
0
10
20
30
40
50
60
70
0
50
100
150
200
250
300
350
400

Figure 11

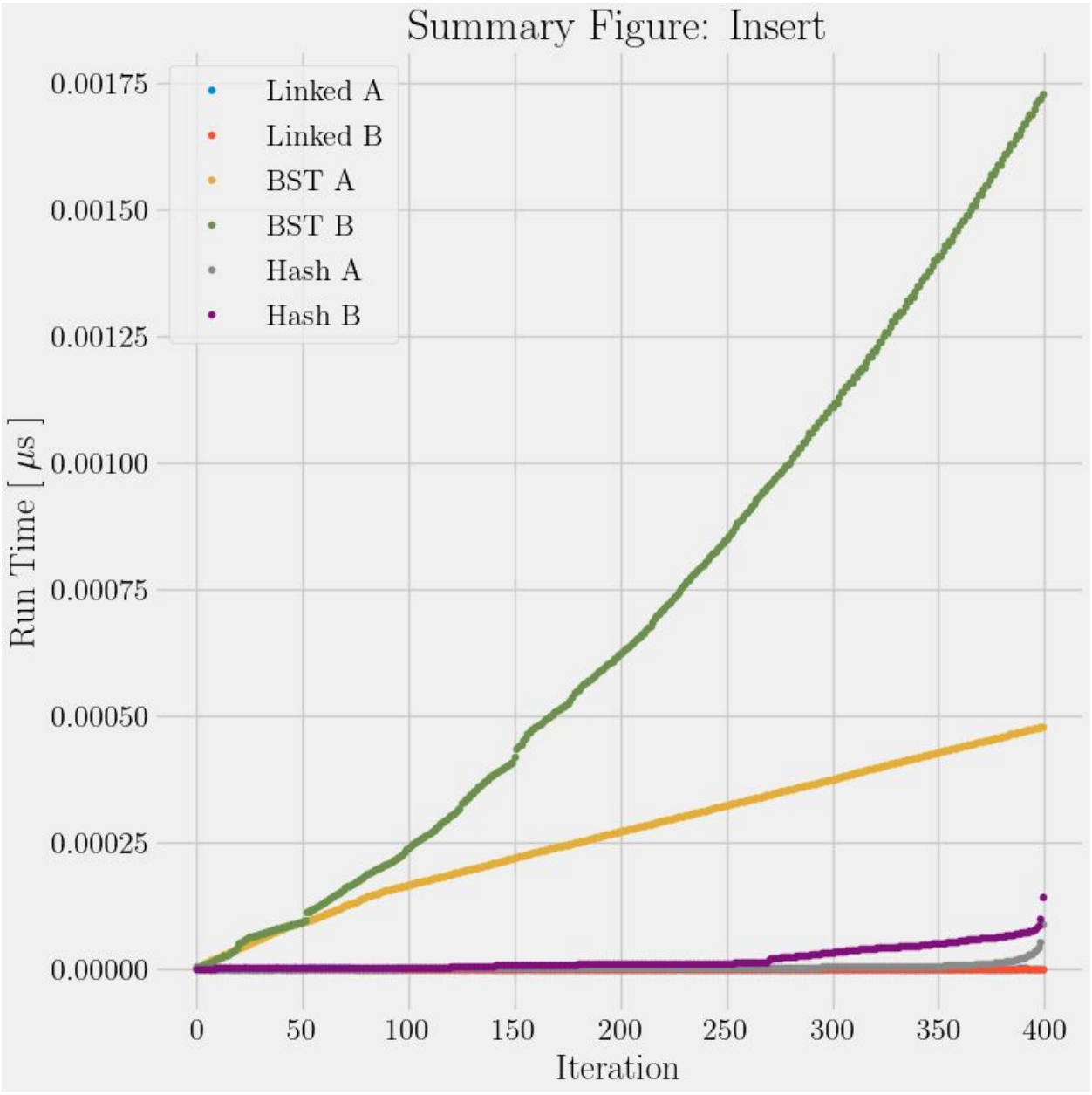

Figure 12 (Hash table graphed uses quadratic probing)

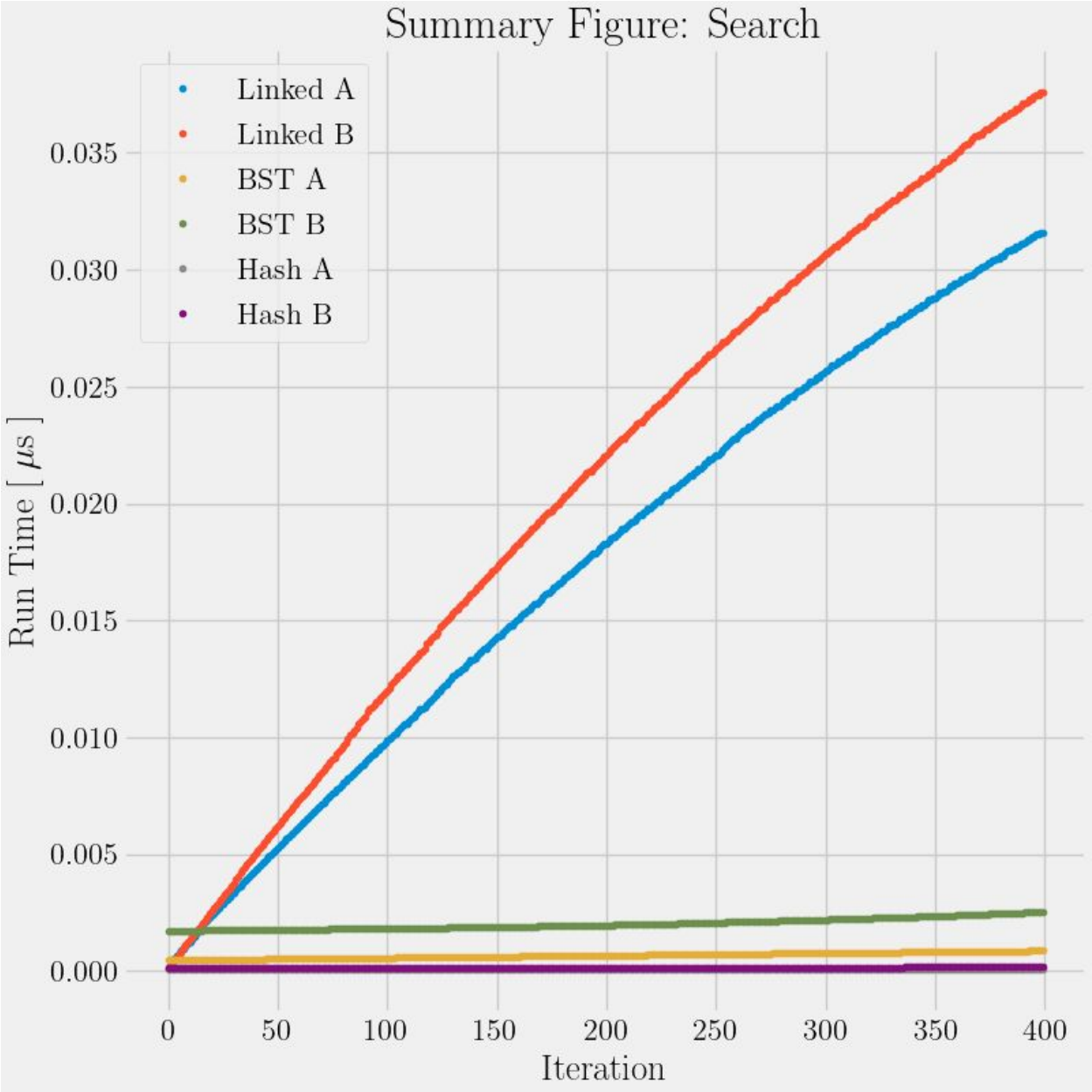

Figure 13 (Hash table graphed uses quadratic probing)

Analysis of Data

Data sets A and B both contain 40,000 data values, however data set A is unordered while Data set B is linearly correlated [Figure 1]. This can impact search and insert run time in different ways for different data structures. As seen from figure 2 and 3, the insertion time for the linked list is O(1) for the both data sets because each element is inserted at

the head [Figure 2, 3].

For the BST, the insertion and search time for data set B is significantly larger than for data set A [Figure 4 and 5]. This is likely because the elements in data set B are in ascending order which results in a BST that is more skewed. In a skewed tree, the height is greater logN, so the time complexity for insert and search will be larger and more proportional to the number of elements, N.

Both hashing with linear probing and hashing with chaining perform better with ordered data (set B) [Figures 6, 7, 10, 11]. This is because there were less collisions so the insertion and search time is faster. However, hashing with quadratic probing performed much better with unordered data for inserting, searching, and collision reduction [Figure 8,9].

Overall, a linked list performed best overall for inserting, with a constant O(1) time, while the BST performed the worst [Figure 12]. To search, hash tables using quadratic probing were the most efficient and linked lists performed the worst [Figure 13].

Conclusion

Ater analyzing all data structure types and how they perform inserting and searching large amounts of data, we conclude that USPS should implement a hash table algorithm that utilizes quadratic probing for collisions. Even though a linked list performed best for inserting data, it's search time performance was O(n) time. Similarly, the binary search tree performed well with searching, but had large worst-case run time for inserting. Since USPS delivers approximately 20 million packages a day, and will need to insert and search efficiently, it would be highly inefficient to use a linked list or a binary search tree. So, even though a quadratic hash table is not the fastest at inserting, it is the only data structure out of the three that has both efficient insert and search run time, thus the USPS should implement a hash table algorithm using quadratic probing.