EECS 560
Lab 4 report
Haaris Chaudhry

Organization and Data gathering

In this lab, students had to determine how much time certain hash tables took with respect to how much data was loaded into each hash table. The program necessary for the lab is fairly simple and is organized as follows:

1. First, we create the necessary variables. The load factors, which are the ratios of data to data slots, were loaded into an array starting at .2 and adding .1 until we got to .9.
2. The number of buckets for each hash table was set to 600011.
3. We were given an implementation of a class (The Timer class) that could be used to determine the duration that a piece of code took to execute (in seconds).
4. We use two nested for-loops, the outer for-loop iterates through the load factor array, while the inner for-loop iterates through each random generation of values. We use i and j as the index values for each loop. The j value is used as a seed for a random number from 0 to RAND_MAX.
5. The number of values that we use is set to the load factor * 600011 for each iteration of the outer loop. This would be 8 iterations.
6. The inner loop is used to create random numbers in an array that is size (load factor * 600011) and stores these values into each hash table. We calculate the time it takes for each hash table to obtain all values by using an object of the Timer class. This would be 5 iterations.
7. We know that we're going to obtain 8*5 =40 duration values for each hash table, so an array was used for each hash table to store the durations.
8. After gather data, another for-loop was used to iterate through the duration arrays to obtain average times. These average times were then printed to the screen in a readable format.

The following is data gather during one execution of the program:

For load factor 0.2:
openAverage is: 0.0132966s
quadraticAverage is: 0.004997s
doubleAverage is: 0.0055264s

For load factor 0.3:
openAverage is: 0.0187034s
quadraticAverage is: 0.0072608s
doubleAverage is: 0.008204s

For load factor 0.4:
openAverage is: 0.0274054s
quadraticAverage is: 0.0103054s
doubleAverage is: 0.011343s

For load factor 0.5:
openAverage is: 0.036575s
quadraticAverage is: 0.0138952s

doubleAverage is: 0.0148564s

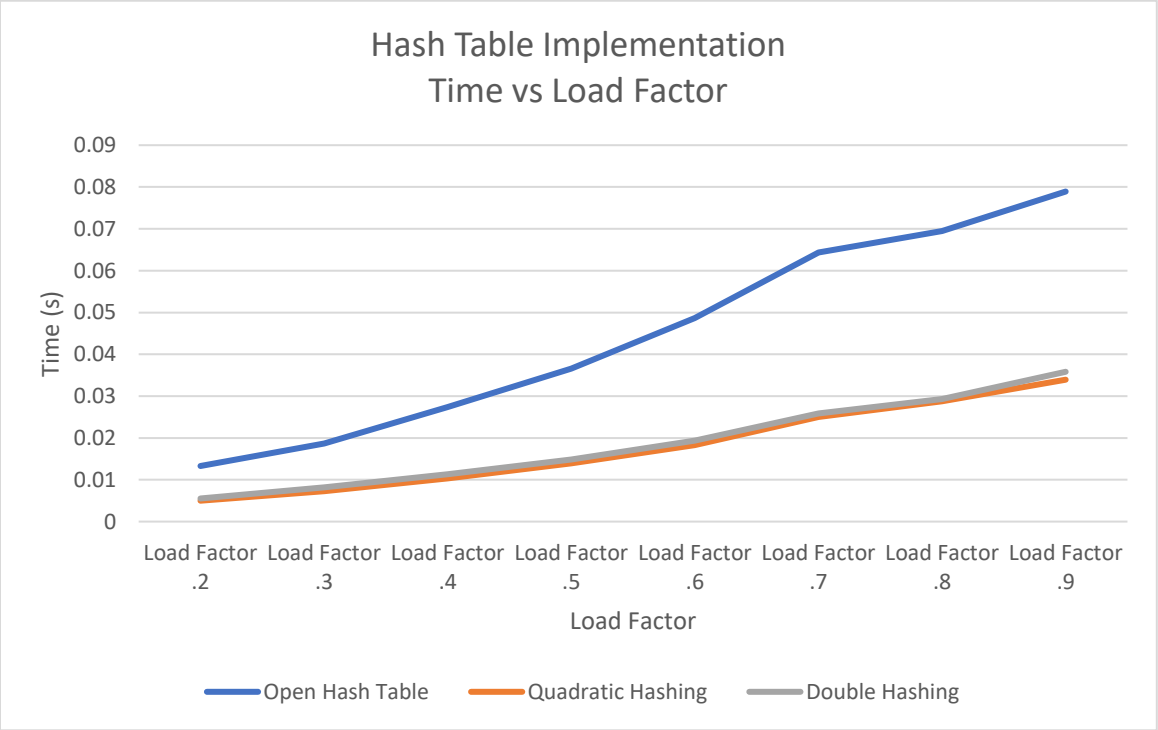
For load factor 0.6:
openAverage is: 0.04867s
quadraticAverage is: 0.0182956s
doubleAverage is: 0.019344s

For load factor 0.7:
openAverage is: 0.0643688s
quadraticAverage is: 0.025047s
doubleAverage is: 0.0258992s

For load factor 0.8:
openAverage is: 0.0694498s
quadraticAverage is: 0.028789s
doubleAverage is: 0.029296s

For load factor 0.9:
openAverage is: 0.0789186s
quadraticAverage is: 0.0339298s
doubleAverage is: 0.0358188s

Summary of Results
Using Microsoft Excel, the following graph was obtained:



It's obvious that the time for insertion in the open hash table is significantly longer than that of the quadratic or double hashing methods.

Conclusion

It comes as no surprise that open hashing results in a longer time for insertion. If we think about the implementation, we know that open hashing requires the use of traversing linked lists in each bucket before finding a spot to insert a value. Therefore, there could be a bucket, or several buckets, where linked lists become extremely long.

Neither quadratic hashing nor double hashing face the performance hit of using linked lists, and are therefore much better suited when it comes to time performance. Quadratic hashing is slightly faster than double hashing all throughout the experiment, however, the difference in time between the two is very small.