Ricardo Golac Barriga

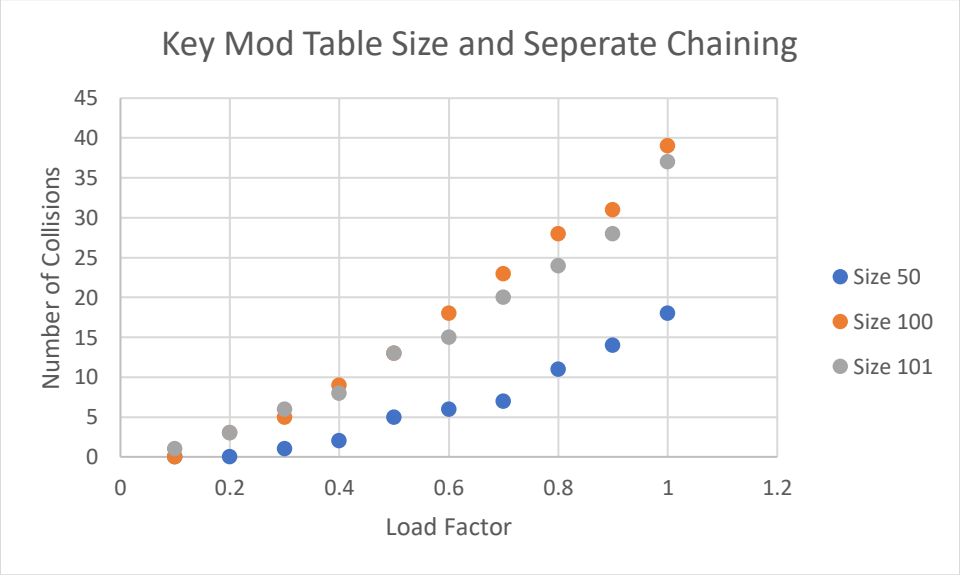
COP 3530

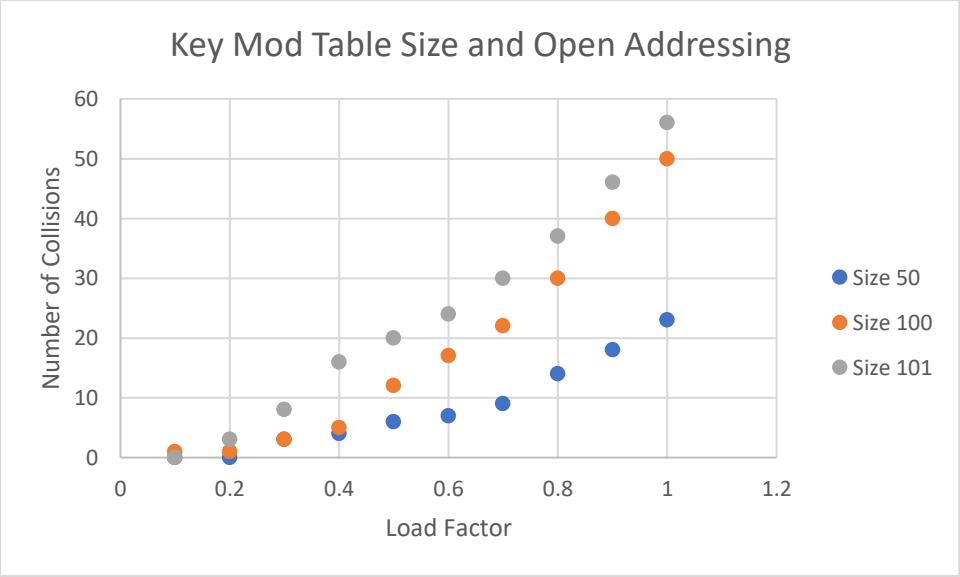
Hashing Analysis

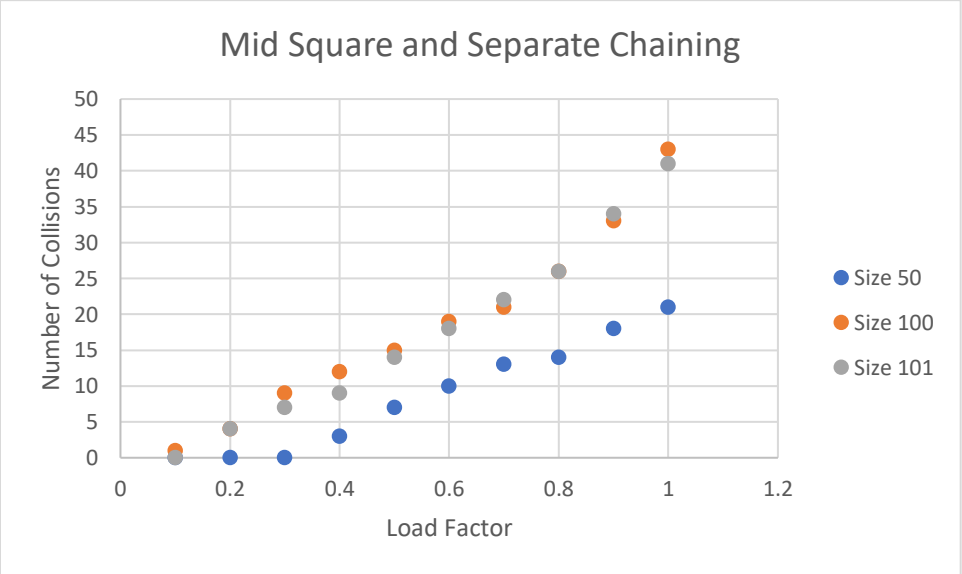
I decided to explore how 4 different implementations of a hash table faired using three different hash tables sizes. I tested all 4 implementations using sizes of 50, 100, and 101. I used two different hash functions: key mod table size and mid square. I also used two different collusion resolution schemes: separate chaining and open addressing. In total there are 12 sets of data. I filled the hash tables with random digits ranging from 0 to 3 times the tables size; this was done until the load size was 1. To generate the random numbers, I used the random class in C++ and used the uniform standard distribution feature.

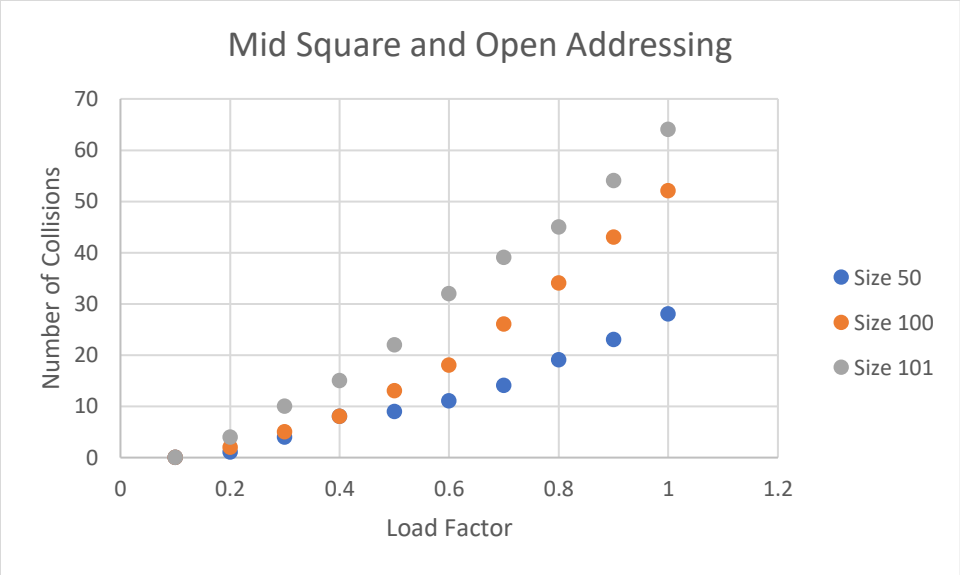
In the key mod table size hash function, I simply returned the key mod the table size. For the mid square function, I squared the key, took the middle digits, and then did mod table size. In open addressing whenever there was a collision, I would just find the next open spot and insert the key. When using separate chaining, each index in the hash table contained a linked list so when there was a collision, the value would be pushed back into the linked list at that index.

Below are 4 tables displaying the gathered data.




Key Mod Table Size and Open Addressing
Number of Collisions
60
50
40
30
20
10
0
0
0.2
0.4
0.6
0.8
1
1.2
Load Factor
Size 50
Size 100
Size 101


Mid Square and Separate Chaining
Number of Collisions
50
45
40
35
30
25
20
15
10
5
0
0
0.2
0.4
0.6
0.8
1
1.2
Load Factor
Size 50
Size 100
Size 101



There should be some margin of error for the information above as new random keys were generated for each of the separate twelve tests. Most of the trendlines for the 12 tests show that each table filled somewhat exponentially in an upward trend. So, when first filling the hash table there were few collisions, but the collisions grew exponentially, not linearly, as more keys came in; this goes for all tests regardless of hash function or collision resolution scheme.

When the collision resolution scheme was kept the same, there were some differences when using mid square over key mod size. The data using the mid square hash function tended to encounter more collisions as the load factor increased. Its possible that the random number generator could be generating smaller keys. The mid square hash function only generates high randomness when the keys are big; the squares of lower keys would only affect the low order digits of the hash table which would lead to more collisions and therefore less distribution.

When looking at the how the data differed depending on the collision resolution scheme, there were some notable differences. For the Key Mod Size Tests, there were more collisions when using open addressing for any size. This makes sense since when using open addressing, indexes are often filled up by a key that wasn't meant for that specific index, ensuring a future collision for a key that was meant for an already filled index. This can create a snowball effect that's partly responsible for the exponential behavior of open addressing. Separate chaining simply pushes a new key into the linked list in an index when a collision occurs leaving other indexes open, although this does waste more memory.

When looking at each of the 4 hash table setups, there were roughly twice as many total collisions when the hash table size was 100 rather than 50, which is to be expected. I also noticed that for tests that used open addressing, the collision count stayed close together for a hash table size of 50 and 100 for small load factors but grew apart at high load factors. When using separate chaining, the 100 and 101 hash table size data was very similar; it differed slightly when using open addressing. The 101-table size was used to see if having a prime number table size did reduce the number of collisions. This proved the case when using separate chaining because there were fewer total collisions when using the 101 size of the size of 100, but when using open addressing the 101-table size fared worse than the 100-table size.