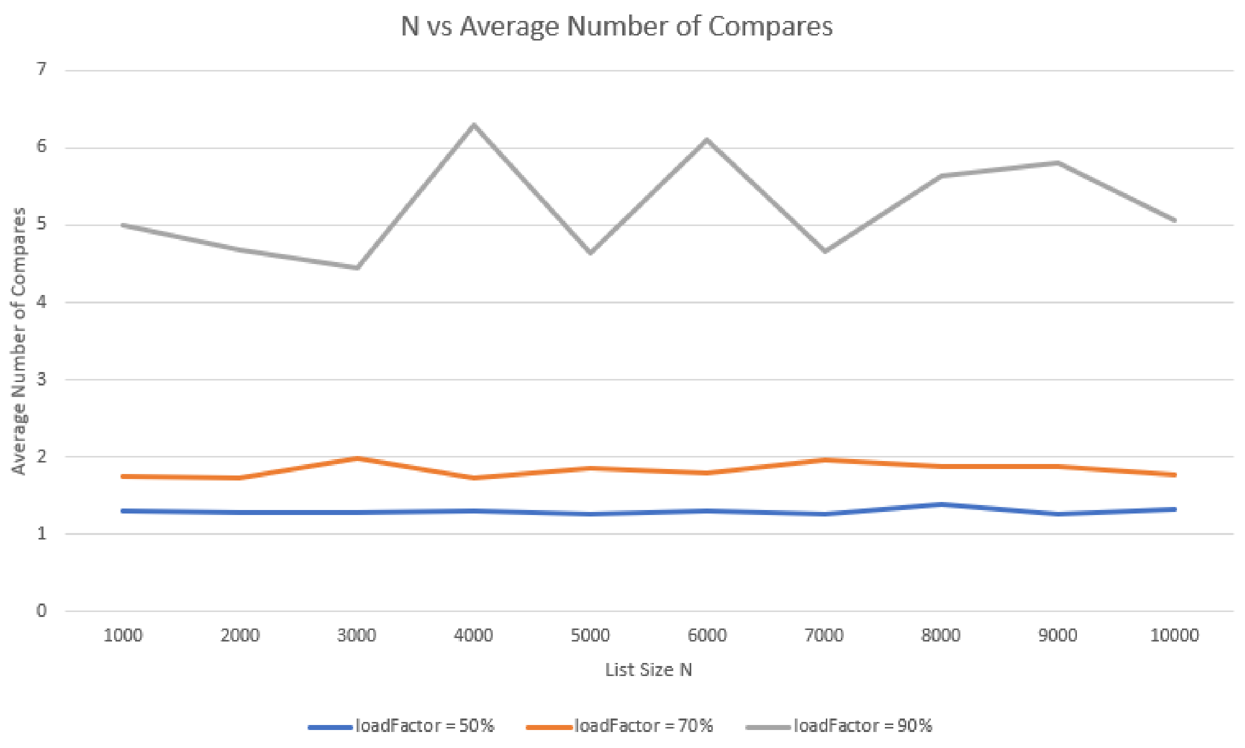
CSC 403 W20 Homework 6 Report   Name: Drake Duerdoth

Experiment 1. Effect of *load Factor* on *average number of compares*

N = #keys	loadFactor = 50%	loadFactor = 70%	loadFactor = 90%
1000	1.30	1.75	5.00
2000	1.29	1.73	4.68
3000	1.29	1.98	4.45
4000	1.30	1.73	6.30
5000	1.26	1.86	4.64
6000	1.30	1.80	6.11
7000	1.26	1.97	4.66
8000	1.39	1.87	5.63
9000	1.25	1.87	5.81
10000	1.32	1.76	5.07

Plot of *N* vs *average number of compares* for each load factor.



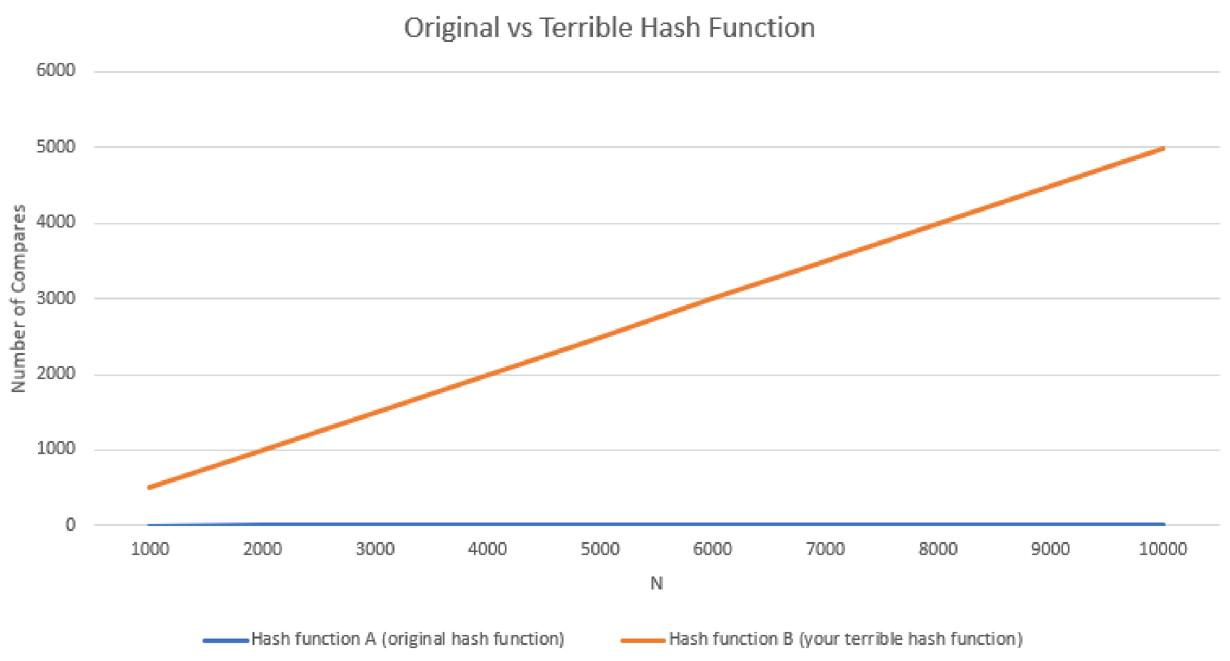
Observations/Conclusions.

The load factor was most optimal when it was set to 50%, whereas 90% was the least optimal. Order of growth was closest to constant with 50%. When the load factor was set to 90%, the number of compares had a much larger and unpredictable range.

Experiment 2. Effect of *hash function* on *average number of compares*, load factor 80%.

N = #keys	Hash function A (original hash function)	Hash function B (your terrible hash function)
1000	2.38	1.99
2000	2.70	2.11
3000	2.68	2.11
4000	2.58	2.15
5000	2.78	2.12
6000	2.66	2.12
7000	2.77	2.09
8000	2.85	2.05
9000	2.68	2.06
10000	2.81	2.13

Plot of *N* vs *average number of compares* for each hash function. Your plot will have two lines



Observations/Conclusions.

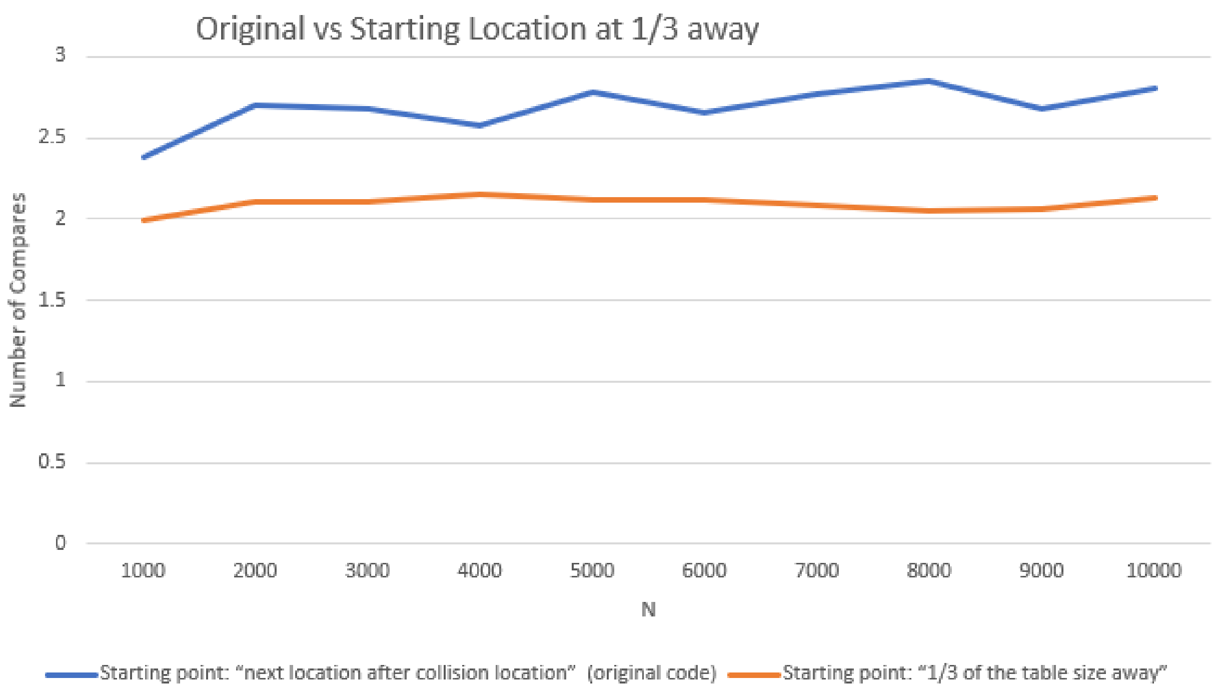
The original hash factor was much more optimal than Hash Function B. key.hashCode() & 0) caused tons of collisions. The memory address after hashing is similar for every hash, thus causing thousands of more comparisons. The order of growth became linear with the terrible function, whereas the original is constant.

Experiment 3. Effect of *probe starting point* on *average number of compares*, load factor 80%

N = #keys	Starting point: "next location after collision location" (original code)	Starting point: "1/3 of the table size away"
1000	2.38	174.41
2000	2.70	316.71

3000	2.68	489.56
4000	2.58	660.99
5000	2.78	840.35
6000	2.66	1027.03
7000	2.77	1131.17
8000	2.85	1303.09
9000	2.68	1577.55
10000	2.81	1678.97

Plot of *N* vs *average number of compares* for each Starting Point condition



Observations/Conclusions.

With the starting point at 1/3 of the table size, the order of growth barely changed, staying constant. Start was changed to

```
(i+(M/3)) % M
```

Though the number of compares stayed around the same, the get function did not work properly because it kept running into null.