Kelly Breedlove
Dr. VanDrunen
Project 5
7 April 2015

Methodology:

For all of my questions, I tested the runtime of the three methods required by the Map interface - put, get, and contains key – for the specified type of hash map. My experiment code was largely reused from project 3, which was originally based on your test code. I made an abstract Map Experiment class with classes that extend it for Basic Hash Map, Perfect Hash Map, and Linear Probe Hash Map. The runtime tests for put, get and contains key all use a nested stopwatch class to time them. Each test populates a map with the given number of pairs, then the stopwatch times as put, get, or contains key are called on each pair.

Question: How do the running times for put, get, and contains key on large data compare when using a basic hash map or linear probing?

Runtime (milliseconds)

| Map Type | Method | 10 | 100 | 250 | 500 | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Basic Hash Map | put | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| | get | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | containsKey | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 42 | 0 | 0 |
| Linear Probe Hash Map | put | 0 | 0 | 0 | 2 | 37 | 88 | 280 | 510 | 684 | 946 | 1123 |
| | get | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | containsKey | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |

Table 2. Running times for basic map operations from Basic Hash Map and Linear Probe Hash Map on a large range of sizes

The pairs for this experiment were scanned in from a text file with Sacramento crime rates. Pairs ranging from 10 to 7000 were tested. A large range of pair sizes were tested in order to show trends in running times related to map sizes. The tests on these pairs were conducted as described above, and the data is given in the above table.

As the data shows, most of the running times for operations on maps of all tested sizes are negligible, the exception being Linear Probe Hash Map's put operation. This trend implies that Linear Probe Hash Map is not a good choice when a lot of put operations will be called. Usually that would mean that Linear Probe Hash Map is still a good option when more get operations would be called than put operations, although in this case Basic Hash Map performed better on put and about the same on get. So, according to this data Basic Hash Map is more efficient than Linear Probe Hash Map.

Runtime (milliseconds)

| Map Type | Method | Max T1 | Max T2 | Max T3 |
|---|---|---|---|---|
| Basic Hash Map | put | 127 | 21 | 32 |
| | get | 1 | 2 | 6 |
| | containsKey | 8 | 1 | 0 |
| Linear Probe Hash Map | put | 1799 | 199 | 218 |
| | get | 33 | 1 | 1 |
| | containsKey | 1 | 1 | 1 |

Table 3. Running times for basic map operations from Basic Hash Map and Linear Probe Hash Map (Max is equal to 7000 for Trial 1-3)

The pairs for this experiment were scanned in from a text file with Sacramento crime rates. Pairs of size 7000, Max, were tested in three trials, T1 through T3. A large number was used for this in order to see efficiency of running times for Basic Hash Map and Linear Probe Hash Map on larger data sets. The tests on these pairs were conducted as described above, and the data is given in the above table.

As the data shows, the contains key and get methods are consistently small. However, the put times for both Basic Hash Map and Linear Probe Hash Map have much larger running times. Specifically, the put times for Linear Probe Hash Map are higher than the put times for Basic Hash Map. This is consistent with the findings in Table 2. Once again, this would support the conclusion that Basic Hash Map is more efficient than Linear Probe Hash Map.

One reason Linear Probe Hash Map's put operation may have shown to have such large running times in both of these data tables could be because of the rehashing done periodically in the put operation. In order to optimize Linear Probe Hash Map, a different approach to rehashing should be taken.

Question: How do the running times for put, get, and contains key on small data compare when using a basic hash map, linear probing, and perfect hashing?

| | | Runtime (milliseconds) | | | |
|---|---|---|---|---|---|
| Map Type | Method | 5 | 10 | 15 | 20 |
| Basic Hash Map | put | 0 | 0 | 0 | 0 |
| | get | 0 | 0 | 0 | 0 |
| | containsKey | 0 | 0 | 0 | 0 |
| Linear Probe Hash Map | put | 0 | 0 | 0 | 0 |
| | get | 0 | 0 | 0 | 0 |
| | containsKey | 0 | 0 | 0 | 0 |
| Perfect Hash Map | construction | 94 | 83 | 75 | 84 |
| | put | 0 | 0 | 0 | 0 |
| | get | 0 | 0 | 0 | 0 |
| | containsKey | 0 | 0 | 0 | 0 |

Table 1. Running times for basic map operations from Basic Hash Map, Linear Probe Hash Map, and Perfect Hash Map on a small range of sizes

The pairs for this experiment were scanned in from a text file with first and last names. 5, 10, 15, and 20 pairs were tested. The reason for choosing relatively small numbers of pairs is because anything larger than 25 pairs could not construct a perfect hash map, even when given 20 or so minutes. I also found that a perfect hash map could not be constructed from the Sacramento crime data. The tests on these pairs were conducted as described above, and the data is given in the above table.

As the data shows, all of the running times for operations on maps of this size were negligible. The only significant running time is the construction of the perfect hash map. In order to make a more useful conclusion, perfect hash maps of larger sizes would need to be tested. It is possible that as the operation running times for basic hash map and linear probe hash map increase, the operation running times for perfect hash map may stay small, making the construction time worth while.