Man Tik Li
Assignment 4

Benchmarking

1. A short (one paragraph) explanation of why the benchmark harness is structured as it is. Why is there an initial run that isn't counted in the min/max/average? Why are we measuring multiple runs?

   The goal of the benchmarks is structured this way is to let the system warm-up. If the program counted the min/max/average, it would affect the program's run time. The warm-up called the initial run it tends to take longer than usual to run. It's warm-up some hardware in the machine, as well as the JVM. For instance, when we visit a website for the first time, it takes longer to load the page. It is because it takes the initial run of everything from the page. If we revisit, it will take less time because the cache is stored in the client's browser.

2. Reasonable operational profiles for use of the RedBlackBST in the following use cases. Each of these profiles, for our purposes, is simply a percentage of insert/delete/lookup operations (3 numbers that sum to 100 in each case).

   1. A logging data structure, which mostly records new data, with occasional queries and edits (deletions)

      Insert: 80
      Read: 10
      Delete: 10

   2. A read-heavy database, with occasional updates and deletions.

      Insert: 10
      Read: 80
      Delete: 10

3. A description of what steps you took in preparation for your performance measurements. For example, what other software is or isn't running on your machine? (For this assignment, you may ignore our discussion of frequency scaling's effect on performance tests, since disabling it is quite inconvenient and many of you are probably using laptops.)

   While running the benchmark, I opened chrome with a few tabs, a preview, text editor for compare data, and a MS word for this assignment. I don't see those programs had siphoning a lot to the CPU.

4. For each of your operational profiles, report the following, which should be produced by Benchmark.main() once you update the hard-coded operational profile.

1. Warmup time
    1. Insert-Heavy: 281491104ns
    2. Read-Heavy: 150756933ns

2. Minimum iteration time
    1. Insert-Heavy: 92955146ns
    2. Read-Heavy: 57153865ns

3. Maximum iteration time
    1. Insert-Heavy: 175381906ns
    2. Read-Heavy: 94528750ns

4. Average iteration time
    1. InsertHeavy: 111369295ns
    2. ReadHeavy: 66837312ns

5. Which of your operational profiles has higher *throughput*? What about the red-black tree might explain this outcome?

   According to the data from question #4, Insert heavy programs have higher throughput because it takes time to insert all the values and balance the red-black leaves' structure. The red-black tree optimal for the operations; it makes sure that the tree is balanced, or the node is on the floor or ceiling, or even the size and rank of the node.

6. Are your warmup times noticeably different from the “main” iteration times? Most likely yes, but either way, why might we *expect* a significant difference?

   My warmup times are higher than the main iteration times. The reason of the significant difference is the hardware, and the Java JVM needed to warmup before the main iteration begin to run.

7. The numbers for operational profile (c) aren’t actually that useful. Look at the benchmarking code, and explain why those numbers, as reported, tell us nothing about the performance of using the BST in a real application for a read-only workload.

   The read operation in benchmarking is only an integer value that we defined. The real application has more complex values or data set. When implementing BST, each application would have a different result.

8. Assume each run (each call to runOps) simulates the activity of one remote request. Based on the average execution time for each operational profile, what would the throughput be for each of your profiles? (i.e., requests/second)

   Insert: 1 Requests / 111369295ns. Around 8.98 requests/second (1 Request / 0.111s)
   Read: 1 Requests / 66837312ns. Around 14.96 requests/second (1 Request / 0.668s)

9. Assuming each remote user makes 5 requests/minute, your program's resource usage scales linearly, and we are only interested in CPU execution time, how many concurrent remote users could you support on your machine (again, do this for each operational profile) without degrading performance or overloading the system?

   For insert heavy (8.98 requests/second), it would be 540 requests/minute, and it would be handling about 108 concurrent users. For read-heavy (14.96 requests/second), it would be 900 requests/minute and able to handle 180 current users.

10. What aspects of load are we not testing, which could possibly reduce the capacity of your machine to service requests?

    The insert operation does not completely test to see the program's action when the disk filled up. The possibility could be remote access. If the testing involves remote access, the testing time between the remote requests would be different.