Timing vs. Counting Operations in P2

For the P2 writeup, we ask you to compare various implementations. While we used timing in P1 (and will use it again in P3), there are two options we will allow for comparing implementations in P2: a) timing and b) counting operations. We'll say a bit about each approach below. Both have pros and cons. It can be tricky to do timing accurately (we give you some suggestions below to help with this) but in the end this is what we actually care about, and it takes into account issues like the cost of memory accesses. Counting operations may give you curves that better match what asymptotic analysis leads you to expect. You also do not need to run multiple trials and average them. But you have to decide exactly what counts as an operation. Either approach will be acceptable for the P2 writeup. In fact you might even want to try BOTH approaches for one of your experiments and see how the results compare! It will also be o.k. in the P2 writeup if you use counting operations to answer some of the questions and timing to answer others. Just make it very clear which approach you used for each question.

(a) **Timing**

One very basic guideline for timing anything is that you should do more than one trial. You could do a set number of trials and throw out the highest and lowest result and average the rest. For timing in this class, we recommend throwing out the first few trails and averaging the runtime of multiple trials as shown in the example timing code below. One thing to be aware of is that Java optimizes repeated operations, so the first few times a piece of code runs it runs more slowly. This is known as the Java virtual machine "warming up". Throw away several trials at the beginning to exclude this effect.

```
1  double totalTime = 0;
2     for (int i = 0; i < NUM_TESTS; i++) {
3       long startTime = System.currentTimeMillis();
4       // Put whatever you want to time here .....
5       long endTime = System.currentTimeMillis();
6       if (NUM_WARMUP <= i) {    // Throw away first NUM_WARMUP runs to exclude JVM warmup
7         totalTime += (endTime - startTime);
8       }
9     }
10    double averageRuntime = totalTime / (NUM_TESTS - NUM_WARMUP);
```

(b) **Counting**

For this approach, we expect you to instrument your code to count the number of operations executed. This will look like a bunch of counter++ or counter += 5 or whatever operations throughout your code. This leaves up to you the decision of what to count as an operation. There is not really a standard on this; pick something reasonable and be sure you are consistent between implementations you are trying to compare.

Whichever approach you pick, **be sure to make a copy of the code that you will instrument and place all of your instrumented code into the experiments package**. Also note that we are definitely expecting more on this writeup than we did for P1. Make sure that you refer back to the bullet points at the beginning of the WriteUp document and that for each question you have answered all parts - each sentence in a question is something you should be sure you have responded to. When in doubt please ask!