Computer Science 62
Lab 9

Wednesday, March 31, 2010

For today's lab, you get to use Eclipse again :) The lab has two goals: first, we're going to revisit sorting one last time and play with heapsort and, second, we're going to remind you about our `StopWatch` class and introduce a memory measuring class.

Getting started

Create a new java project in Eclipse and copy over all of the files from:

`/common/cs/cs062/labs/lab9/`

in to the source directory of the newly created project. Refresh your project and you should see all the files.

There are quite a few classes in there so look them over and make sure they look familiar.

Heapsort

Create a new class called `Heapsort` that implements our `Sorter` interface. Use the `ArrayListPriorityQueue` class for the heap to start with. When you construct the heap, use the constructor that takes an `ArrayList` as a parameter.

Timing

In all the files, I've included my version of `SortTimer` from lab 3. You are welcome to use your own (in fact I encourage you) if you feel more comfortable with that code. In either case, run it once to make sure it

works. Then, add your `Heapsort` as an additional sorter and compare the times. Which is faster? Do you think `Heapsort` is $O(n \log n)$?

Variants

Create another class `Heapsort2` that uses the priority queue constructor with no parameters, and then add the data items using the `add` method. Again, add this in your `SortTimer` class and compare the times with the original `Heapsort`. Which one is better? Is this what you expected?

Create a third variant `Heapsort3` that uses java's priority queue from `java.util.PriorityQueue`. Compare the timing of this version of heapsort to the other two versions. Which is better?

Measuring memory

For our next assignment, you will be measuring the memory usage of a couple of data structures. Measuring memory usage in Java is a bit tricky. I've included a `Memory` class that provides a very simple approach to measuring memory. The `usedMemory` method tells you how much memory is currently being used by your program by taking the difference between the total memory available and subtracting how much memory is free.

If you measure the amount of memory being used before and after you create/initialize an object, you can roughly measure the amount of memory taken up by that program. Experiment with this approach to memory usage measurement. A few things to try:

- Measure how much space is taken up by and `int[]` or a `double[]`. Is it what you expected?
- How does using an `ArrayList` compare to using an array (don't forget to fill in the entries of each of these if they're objects).
- What about the `Vector` class? How does the increment affect the memory usage?
- How much does memory usage vary from run to run when measuring the same thing? Can you write iterative versions to average the memory usage?