Lab 9: Sort Performance

November 6, 2018

1 Introduction

We have been discussing several sorting algorithms in class. As we talked about, there are many dimensions along which we can measure the quality of an algorithm, including time, space, elegance, and simplicity. In this project, we will focus on the running times of algorithms. We will compare actual running times of various algorithms implemented in Java. We will also verify that the running times correspond to our theoretical analyses.

2 Tasks

2.1 Sort Implementations

I have provided Java implementations of each of the sorting methods we discussed in class: selection, insertion, bubble, merge, and quick sorts. Open `TimedSorts.java` and examine the code. Make sure you understand the implementations. The user interface relies on command line arguments. This means that you supply information to the Java runtime at the time you invoke it via the command line. The first command line argument is the name of the sort, the second command line argument is the size of the array to sort. For example, if you wish to execute selection sort on an array of 50,000 elements, you would issue this command:

```
java TimedSorts selection 50000
```

Compile the code and run it a few times to familiarize yourself with the interface. The `switch` statement in the `main()` method indicates the valid sort names. See if you can figure out how the command line arguments are used in `main()`.

If you are using NetBeans, you provide command line arguments by opening the Run menu, clicking on Set Project Configurations, and then clicking on Customize... This will open a window with a text field labeled "Arguments:". Type your command line arguments in this field.

If you are using Eclipse, open the Run menu, click on Run Configurations..., click on the Arguments tab, and enter your command line arguments int Program arguments textbox.

2.2 Sort Measurements

We need some mechanism to measure the running times of our algorithms. The `System` class in the `java.util` package provides a number of useful things. It's worth taking a few minutes and looking at it. The helpful method we want to use here is `System.nanoTime()`. This returns an

integer representing the time at which the method is called. We will call it once before the sort starts and once after. The difference between these times is the number of nanoseconds that the sort actually took. This is then divided by 1,000,000 and reported as milliseconds in the output of the `main()` method.

Do the following things:

1. Create a table with rows for each sort and columns for the size of the arrays. The array sizes should run from 25,000 up to 250,000 in increments of 25,000.
2. Run each sort four times at each array size. Enter the average of the runtimes for each sort at each array size in the table.
3. Graph the results. Use a spread sheet and its graphing features, other graphing software, or draw it by hand.
4. Visually verify that your results match the analyses we did in class. Do the curves for the $\Theta(n^2)$ algorithms look like parabolas? Do the curves for the $\Theta(n \log n)$ algorithms look like something between straight lines and parabolas?
5. Recall that order analysis is interested in the long-term growth of a function. We can characterize this roughly by looking at what happens to the running time when you double the size of the input. Compute the following ratios of running times for each sort and record them in a table.
   (a) (time for 50000)/(time for 25000)
   (b) (time for 100000)/(time for 50000)
   (c) (time for 200000)/(time for 100000)

   Do the ratios match your expectations?
6. Create a document including all your tables, calculations, and answers to any questions asked above. Provide an introduction and conclusion. What was the purpose of this study? What did you learn? Were your hypotheses (the Big-Theta analyses) confirmed? Why or why not? What other factors might come into play that affect the running times of these algorithms that might account for any variations that you observed?

3 Submission

1. The lab is due in class on Wednesday, November 13, 2018.
2. This lab is worth 30 points.