Program #3 Sorting

CSE 464/564

For this assignment, we will be completing three separate exercises from the textbook to explore sorting algorithms: problems 2.1.31 and 2.1.32 on page 268 and 2.3.25 on page 307. Here are the specifications.

1. *Doubling test*: Write a client that that performs a doubling test for sort algorithms. Start at n equal to 1000, and print *n*, the predicted number of seconds, the actual number of seconds, and the ratio as *n* Use your program to validate that *insertion sort* and *selection sort* are quadratic for random inputs, and formulate and test a hypothesis for *shellsort*.
2. *Plot running times*. Write a client that uses *StdDraw()* to plot the average running time of the algorithm for random inputs and various values of array size. You may add one or two more command-line arguments. Strive to design a useful tool.
3. *Cutoff to insertion sort*. Implement *quicksort* with a cutoff to *insertion sort* to subarrays with fewer than *m*elements, and empirically determine the value of *m* for which *quicksort* runs fastest in your computing environment to sort random arrays of *n* doubles, for n = 10^3, 10^4, 10^5, and 10^6. Plot the average running time for *m* from 0 to 30 for each value of *m*. Note: you need to add a three-argument *sort()* method to Algorithm 2.2 for sorting subarrays such that the call to *sort(a, lo, hi)* sorts the subarray a[lo .. hi].

What to turn in:

1. The java files that you develop. These will be graded for style in addition to accuracy.
2. A document (pdf) that answers:
    1. the questions from part 1 above: verify that *insertion sort* and *selection sort* are quadratic for random inputs.
    2. the question from part 1 above: Give and test a hypothesis for the performance of *shellsort* for random inputs.
    3. the question from part 3 above: empirically determine the value of *m* for which *quicksort* runs fastest in your computing environment to sort random arrays of *n*

You may use the *insertion sort* and the *selection sort* code from the textbook or from the Java library. For *quicksort*, you will need to code this yourself since you need to modify the code. Use the drawing tool that you develop in part 2 above in answering the questions from part 1 and from part 3. That is, your pdf answer will include text and graphs to make your point.