COSC 311 WINTER 2016 Programming Project 3

Is postfix traversal of a binary search tree linear, n log n or quadratic?

Distributed (official): 3/31/2016 **Due**: 4/14/2016

Project Statement: Experimentally gauge the run-time complexity of postfix traversal of a binary search tree (BST).

Problem approach: Obtain timing results from postfix traversals of binary search trees. Fit three different curves to the data points:

An O(n) curve: `f(n) = an + b`,
An O(n log n) curve: `g(n) = a*n*log(n) + bn + c`,
And an $O(n^2)$ curve: `h(n) = an^2 + b n + c`.

Determine the errors between fitted curve and data. Use the errors to support an argument as to the best run-time complexity for postfix BST traversal.

More detail:
Obtain enough data (BST size, time) pairs to make a convincing conclusion.

In order to make a convincing argument, you need to do the following:

Ensure sufficient variety of data set sizes,
Ensure sufficient variety of data,
Ensure the BST are representative (sufficiently randomized)
Ensure your code is correct.
Take timing information on what you actually need to measure,
Compare the errors of the fitted curves against the actual data.

Data set size:

Two data points will suffice to perfectly fit a straight line.
Three data points will suffice to perfectly fit a parabola.

Therefore, you will need significantly more than four data points to measure. At the minimum, you should obtain six data points.

Range of data set size:

The maximum value of n (tree size) values should be large enough to successfully acquire timing results within a reasonable amount of elapsed wall-time (say, less than 10 minutes). Also, the size of the tree and execution of the algorithm must fit within the physical resources (i.e., memory space) of your system.

Also, the range of n must be such that meaningful measurements are obtained at all data point values. The execution time must be measurable (i.e., not too small).

The range of the data must be wide enough to allow for the fitted curves to show significant errors.

Data must be reasonable

To obtain average case behavior, you need an "average" BST. The only way to ensure that you have an average BST is to generate multiple BSTs of a given size n. Take the timing information for each BST of given size n, then average the timing results for that size n.

Ensure your code is correct

Demonstrate your code is correct by producing output of a postfix traversal for a small number of small n trees (e.g., n = 10)

Plotting:

Plot the data (independent variable is n (n is size of BST), dependent variable is time) with the fitted curves. Show the data points as visible marks, and the fitted curves overlaid on top.

The plot must have clearly labeled axes, and there must be a legend giving line shape with corresponding fitted curve.

You will need another chart, a bar chart, that compares the errors of the different fits (three bars, one for each fitted curve).

Conclusion:

Give a one paragraph summary giving your conclusion with supporting argument from the data, the plot(s) and the error chart.

Turn in:

Hardcopy of code

Example run: postfix traversal of size 10 BST (timing information not required)

Plots and charts

Conclusion

Grade based on:

Correctness of code, satisfying spec	85%
Readability, elegance, documentation	15%

How to get timing information?

You may use `System.currentTimeMillis()` or `System.nanoTime()`.

Note, there are other ways of measuring running time. Any of those are fine – some are better than using the Java System API

Measure the correct thing. Do not measure anything unrelated to postfix traversal.