Assignment: Performance Constrained Version Of "BTKeysAtSameLevel"

Avraham Leff

September 22, 2022

1 Assignment-Specific Packaging

The general packaging is unchanged from the basic "Homework Requirements" (see slides from first lecture and "Homework Policies for COM 2545" on Piazza).

This assignment's "DIR" **must be named** *BTKeysAtSameLevel2*, and your application's Java class **must be named** *BTKeysAtSameLevel2.java*. Your write-up file **must be named** *BTKeysAtSameLevel2.pdf*.

2 Motivation

The purpose of this assignment is for you to apply the *order of growth* and *performance measurement* skills that you've been learning in textbook and lectures.

3 Background

You previously implemented the *BTKeysAtSameLevel* assignment in which you wrote a program to **compute "keys at same binary tree level"**. Please refer to the requirements document for that assignment if necessary: the core requirements are unchanged. All that you have to do in this assignment is to implement the same requirements as before **subject to a performance requirement**.

This assignment may therefore be very easy ☺: you can simply copy (don't forget to rename the class!) your code from the previous assignment and you're done. All that's left is the *non-programming* component.

Alternatively: you may want to use your doubling ratio infrastructure to validate that your implememntation meet requirements (below).

4 Requirements

> Aside from using the JDK, for this assignment, you may **only use code written by yourself**!

The assignment consists of both "programming" and "non-programming" components.

4.1 Non-Programming

This portion of the assignment is worth 30%.

Your writeup must contain, in this order:

1. Your estimated "doubling ratio" for your implementation.
2. In no more than three paragraphs, explain why your solution "costs" the order-of-growth that your "Big-O" claim.

Your explanation must sketch your design in sufficient detail that your "Big-O" claims follows naturally. Don't give me the code since I can see that for myself! Pseudo-code is the right level of detail. If appropriate, consider incorporating a diagram into your discussion.

This component will be graded on your *"communication skills"*: clarity, correctness, succinctness, and "convincingness". Your grade will also reflect the disparity between your "Big-O" claim and the one that I measure.

You may "explain by reference" to material covered in lecture or textbook or to "well-known" cs-knowledge. If you choose to use a "reference to covered material", you **must offer a convincing explanation** that this material is relevant to solving our problem.

4.2 Programming

Once the tree is built, depending on the algorithm you use, the order of growth could be as good as $O(n)$; O(n log n) performance is certainly feasible. That said, given the cost of processing the input String, I'm fine with a ratio of as large as 3.5. Points will be deducted if performance exceeds this ratio.

This portion of the assignment is worth 70%, and will be graded for "correctness" and ability to meet the "Big-O" requirements at scale.

Please review the general requirements for a programming assignment! I've tried to reduce the chances of "mistakes" occurring through the use of a "skeleton interface", but ultimately, **you are responsible** for ensuring that I can compile and test your code without incident.

1. Begin by downloading a skeleton `BTKeysAtSameLevel.java` class from this git repository.
2. Rename the class as `BTKeysAtSameLevel2.java`.
3. Then, implement the stubbed methods of *BTKeysAtSameLevel2*.
4. Not mandatory, but you may want to verify that your code meets the performance requirement. If it doesn't, you may want to revise your initial implementation.

Note: I'm allowed (in addition to running the performance test(s)) to run "correctness" tests.