BTKeysAtSameLevel2 Writeup

Estimated Doubling Ratio: ~2.0

I am assuming that the *n* we are measuring (as in, O(n)) is the length of the input string, not the size of the binary tree represented by the input string. However, this should not make a difference one way or the other, because the string length is approximately 3 times the size of the tree. This is because each node besides the root, when added to the string, also adds a '(' open parenthesis before the key and a ')' close parenthesis after the key. Since the size of the tree and the length of the string differ only by a constant factor, their order of growth, and thus, their doubling ratio, are unchanged.

Based on my doubling ratio, calculated with both a balanced tree and a linear tree, my algorithm is no worse than O(n), where n is the length of the input string, as the doubling ratio of an O(n) algorithm is 2.0. The reason for this order of growth is that the body of my algorithm is a single for-loop that goes over the input string, character by character, for the length of the string. No method called within the loop or outside the loop contains any recursion or iteration. As such, my algorithm must have an order of growth of Θ(n). Since the size of the tree and length of the input string have the same order of growth, as I have shown above, this is true regardless of whether the order of growth is being measured by the size of the tree or the length of the input string.