David Schulz

CS 4981 021

5/13/2022

**Exercise 4: Optimization Challenge – Binary Search Tree**

* Time spent on optimizations: 3 hours
* What optimization techniques did you use?
  + Java’s TreeMap instead of a custom BinarySearchTree
  + The TreeMap maps each input value to its depth in the tree
  + TreeMap’s floorKey() and ceilingKey() methods are used to get the parent nodes if the new node is inserted into the right or left respectively
  + The get() method is used to get the depth of the floor and ceiling keys
* What was the impact (runtime improvement) of the optimizations you used?
  + Testing with Sample Input 3 on Kattis
  + “Brute force” BinarySearchTree runtime: 3789000 ns
  + TreeMap runtime: 2112000 ns
  + Runtime improvement: 1677000 ns
* What was your ultimate runtime as reported by Kattis for your final attempt?
  + 0.63 s

Graphical user interface, text, application, Teams

Description automatically generated

* What is the algorithmic runtime complexity and memory complexity of your modified solution?
  + Runtime complexity: O(n log n)
    - The algorithm has a for loop that iterates for every input value and then uses TreeMap’s floorKey(), ceilingKey(), and get() within the loop, which all have a time complexity of O(log n)
  + Memory complexity: O(n)
    - The TreeMap has one entry for each of the N input values