Programming Methodology II

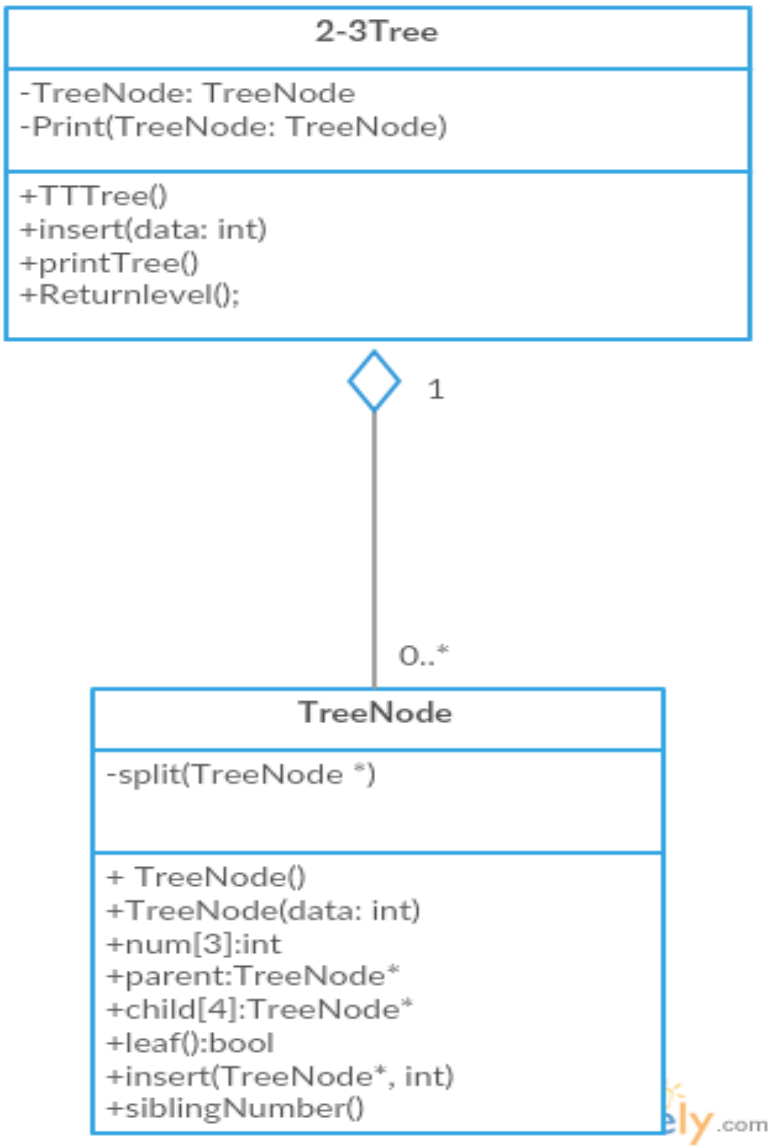
Extra Project

2-3Tree and RedBlackTree

---Kaixiang Huang

---Netid: kh595

1. Design and implement a 2-3 Balanced Search Tree. Use UML for your design. Please note you're not required to 'analyze' the requirements or analyze the design. Your implementation must support the standard 'operational contract' as given in the interface for a 2-3 tree.



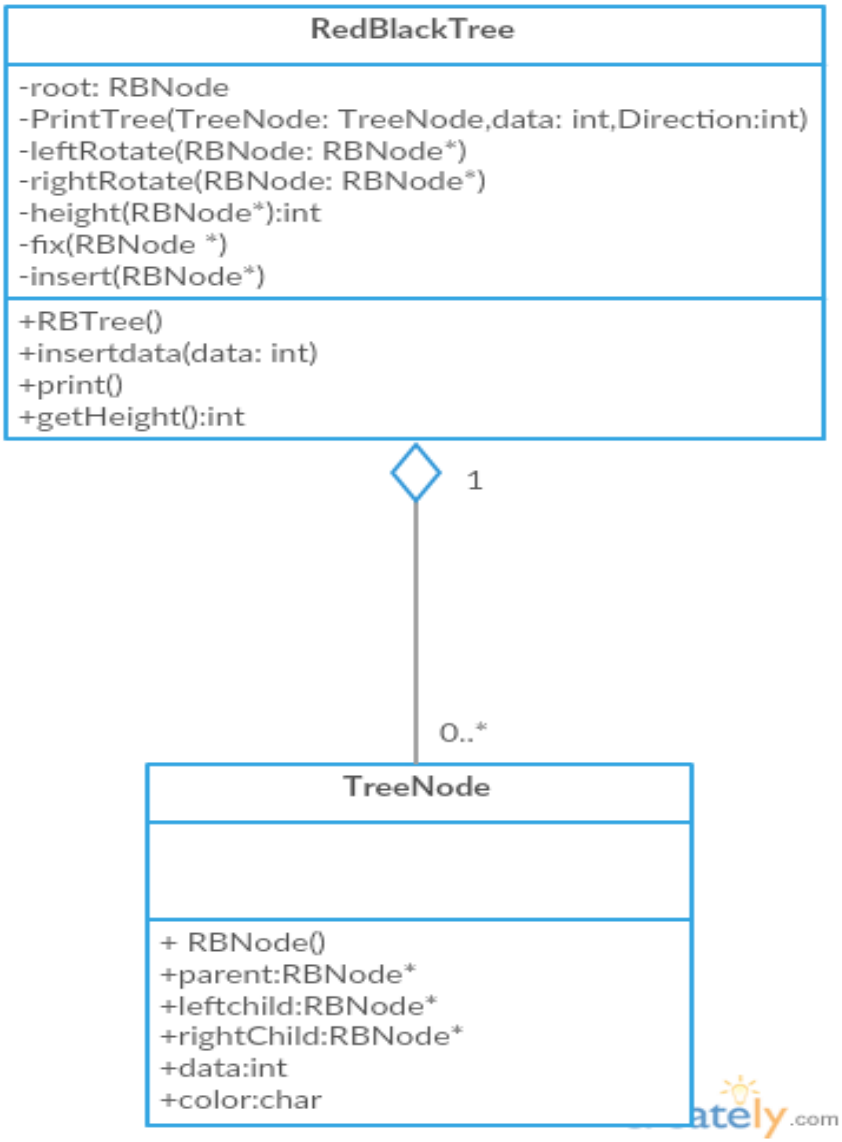
//Use 2-3 tree to store data

//Precondition: 2-3TreeNodes to help us store the data in Node type and with pointer to parent and each child

//Postcondition: A good shape of 2-3 Tree with some 2-Node and 3-Node every node with children has either two children and one data element or three children and two data elements.

2. Repeat the above for another balanced search tree implementation in the book (2-3-4 Tree, AVL, RB tree). You can choose another but please provide an explanation.

RBtree



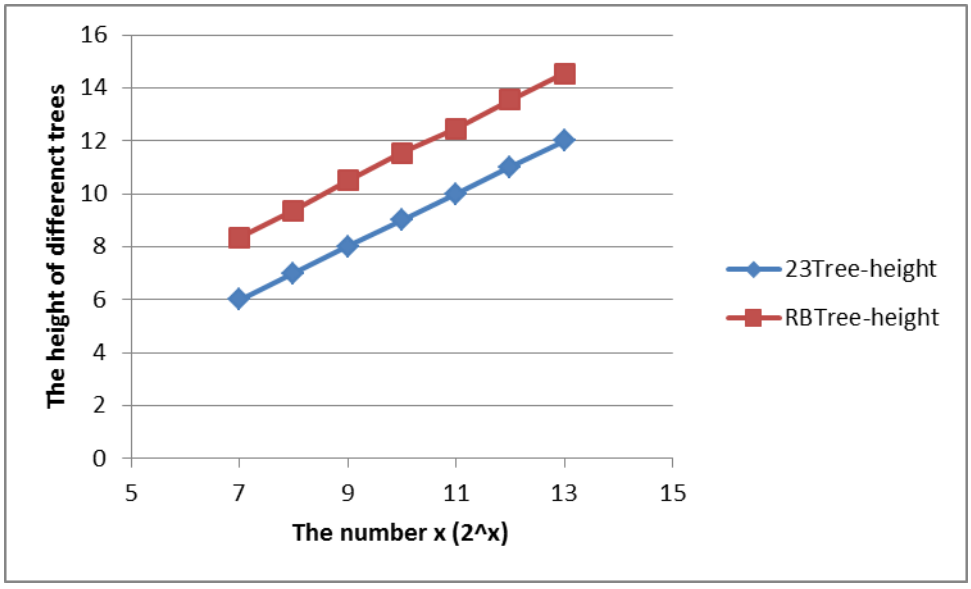
//Use RedBlack tree to store data

//Precondition: RedBlackTreeNodes to help us store the data in RBNode type and with pointer to parent, left child and right child, and its edge color to help us keep balance.

//Postcondition: A good shape of RB Tree with RBNodes. Each node of the RBTree ensures the tree remains approximately balanced during insertions and deletions.

3. Generate 100 random trees of size (=number of nodes) ranging from 64 to 8192 (factor of 2 difference):

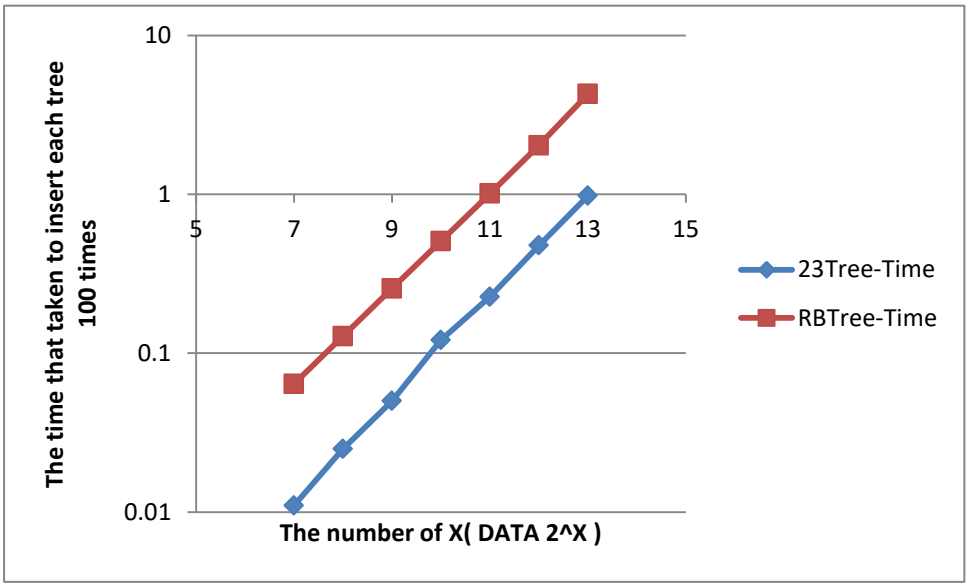
(i) For both 2-3 trees and your chosen balanced search tree: plot average height of tree at each size. Discuss.



The height of 2-3 tree is always the same in the 100 times repeat. It is always x-1 (x is the number of data 2^x), because it can store data in 2 node and 3 node, and it is always a balanced search tree. And the

height of RBTree is different, because different data can cause different tree shape, but its average is quite the same. With the increase of data to insert, and average height increased 1 for both trees.

(ii) Plot the time taken for the insert. Discuss.



The big O of two different tree’ s inserting is quiet the same, 23-Tree is clgN, RBtree is 2lgN. So the increasing rate for both trees are the same from the graph.

For 2-3 tree, insertion works by searching for the proper location of the key and adds it there. So the big O-O(lgN) of insertion is like search operation and it depends on the height of the tree.

For RB tree, it need searching operation and fix operation to fix the tree after insertion, so it take more time than 2-3Tree, but it is also a O(lgN) operation like 2-3Tree.

(iii) Discuss a real world science, engineering or a social science problem where you would have to consider memory requirements along with performance? (e.g., very very large graph).

We can use tree to store big data in social science, for example, we can use those big data to introduce new friend that user may know or user may be interested in. Since the data is hard to manage, we can use better data structure to store than to increase the speed of calculation.

So if we want to store big data and we want to reduce the big O of our operation on those data, we can use different tree to store those data that depends on the properties of different trees.