Pre-Class Work for “Red-black trees—part I” CS110

Create a Google document to record all your answers. Be ready to paste a link to your Google document into a class poll (make sure the document’s settings are such that one can assess its content from the link).

Question 1. (Exercise 13.1-3 Cormen et al.)

[Estimate: 5 minutes] Let a relaxed red-black tree (RBT) be a binary search tree that satisfies red-black properties 1, 3, 4, and 5. In other words, the root may be either red or black. Consider a relaxed RBT, *T*, whose root is red. If we color the root of *T* black but make no other changes to *T* , is the resulting tree a RBT? Explain your answer.

E.g. 1

R

B B

R R R R

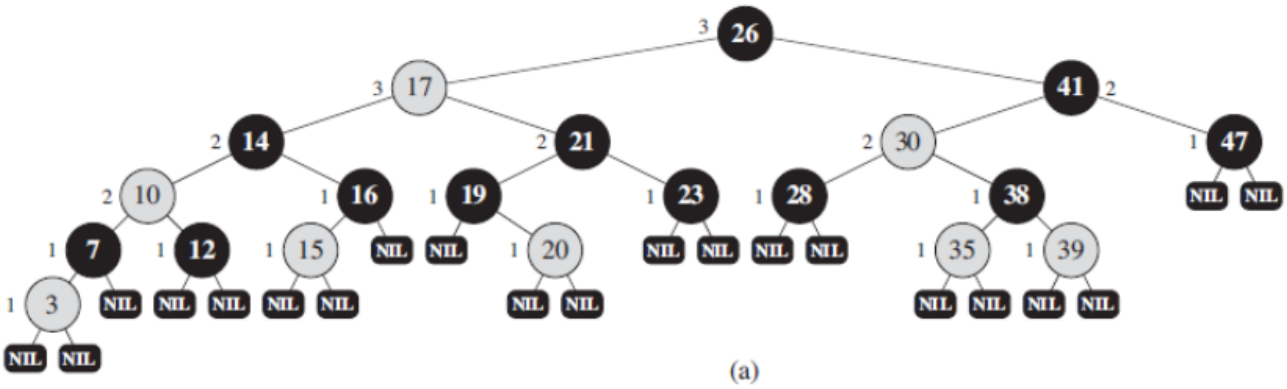
NIL NIL NIL NIL NIL NIL NIL NIL

Yes, if we have a relaxed tree like in example 1 and color the root black we can still have a RBT. This is because the two kid nodes of the root will be black (Rule 4), and there is no rule against having consecutive black nodes.

Question 2. (Exercise 13.1-5 Cormen et al.)

[Estimate: 5 minutes] Show that the longest simple path from a node *x* in a RBT to a descendant leaf has length at most twice that of the shortest simple path from node *x* to a descendant leaf.

E.g. 2



(a)

The class readings give the RBT above as an example. Without counting the NIL nodes, the shortest path is from the root to the right-most node with the value 47 (length 2 if we count the root). The longest path is from the root node to the left-most node with the value 3 (length 6 if we count the root node).

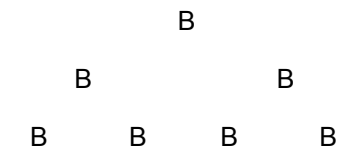
Question 3. (Exercise 13.1-6 Cormen et al.)

[Estimate: 5 minutes] What is the largest possible number of internal nodes in a RBT with black-height *k*? What is the smallest possible number?

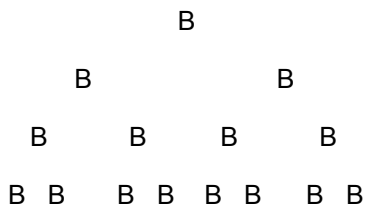
I believe the formula is exponential, something similar to k^2 + k.

For example a tree with black height 2 (example 3) has a max of 6 black nodes. And a tree with black height 3 (example 4) has a max of 14 black nodes.

E.g. 3



E.g. 4



Question 4. Left-Right Balance

[Estimate: 7 minutes] We can define the left-right balance of a binary search tree (BST) as the ratio of the number of nodes in the left sub-tree to the number of nodes in the right. Answer the following questions:

1. If a BST can be successfully colored red-black, then find bounds on the left-right balance of that tree, or prove why there is no bound on the left-right balance of that tree.

   Given the rule that all simple paths from the node to descendant leaves must contain the same number of black nodes and that a red node must have 2 black nodes as children, there is a general bound to how unbalanced a tree can be.

2. Does your answer change as the number of nodes in the tree increases? Explain your answer.

   This should not change as the number of nodes increases given that the rules must always be followed, meaning there should always be the same amount of imbalance no matter the height or levels in the tree.

Question 5. From Binary Search Trees to Red-Black Trees

[Estimate: 5 minutes] Consider the Node class you have worked on for the previous lessons on BSTs. Describe how you could start from that class to adapt it to be applicable to RBTs. Enumerate what different attributes and methods you would need, and any other changes you would need to make to the code. [Challenge] Implement these changes in Python, and clearly demonstrate that your code works by applying it to a few test cases.

The first thing we need is to add an extra attribute to the class - color. Next, we need to add some constraints, like having the child nodes of a red node be black, or the path from a node to leaf nodes to contain the same number of black nodes. We must also add the possibility to change the color of a node.