CSCI 104L Lecture 24: Splay Trees

Question 1. Starting with an empty AVL tree, do the following operations, in sequence:
INSERT: $1, 2, 3, 12, 9, 13, 7, 4, 6, 5, 8$
DELETE: $4, 1$
INSERT: $1, 14, 11$
DELETE: $3, 13, 12, 11, 14, 2, 5, 7, 8, 9$

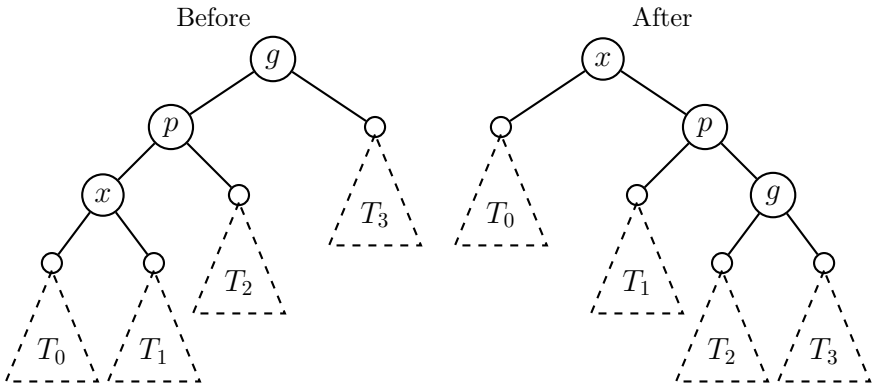
Splay Trees

Question 2. We saw AVL trees, which guaranteed a search time of $O(\log n)$, where n is the number of nodes in the tree. Is it reasonable to optimize to guarantee the worst-case lookup time is $O(\log n)$, or are there other things we should be worried about?

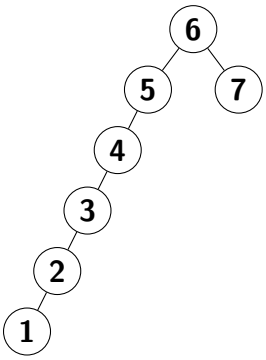
In a *splay tree*, our goal is that recently-used data should be near the top. We add and search as per a normal binary search tree, except when we're done, we're going to splay it to the root.

If we just inserted x and it is now a child of the root (whether because we inserted it there or because it splayed up to that location), the adjustment is easy: just do a single rotation

Otherwise, if x is not a child of the root (and is also not a child of the root), we're going to bring it up two steps. There are two possibilities (four if you count the mirrors). If x, the parent, and grandparent form a zig-zag, then we do exactly a double rotation. Otherwise:



Question 3. Consider the following splay tree; what does it look like after `find(1)`?



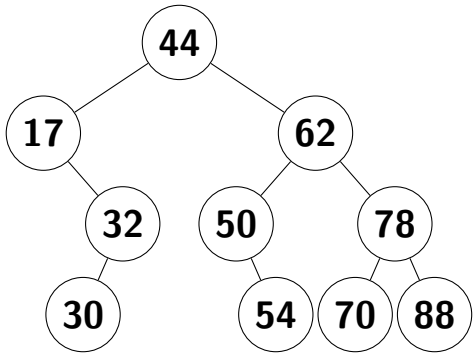
Question 4. From the tree that resulted, what does it look like after `find(3)`?

Question 5. Suppose we start with an initially empty splay tree and then insert the values $1, 2, \ldots n$ in order and then call `find(1)`. What is the total running time? What is the average time per operation?

Deletion

When deleting from a Splay Tree, first use the Binary Search Tree deletion algorithm. Then, take the parent of the deleted node (which is probably the predecessor or successor of the value you wanted to delete), and splay it to the root.

Question 6. Consider the following splay tree; what should it do if you `delete(44)`?



The "Splay Trees Are Awesome" Conjecture: there is a conjecture that, for any sequence of binary search tree operations, splay trees are asymptotically as fast as any other implementation (basic, AVL, Red/Black, etc). Some sets can be done in less than $O(\log n)$ time, and the conjecture is that if one binary search tree implementation can do it, so can splay trees.

Question 7. Starting from an empty splay tree, execute the following operations in sequence:
INSERT: 1, 2, 4, 5, 6, 8, 7, 3
REMOVE: 1, 2, 3