CPSC 331: Assignment 2

Matthew Allwright
30037812

Karim Beyk
30027342

Seth Campbell
10152719

November 10, 2018

AVL Tree Defenition and Properties

Question 1: Prove that $s_i \geq F_{i+1} + F_{i+2} - 1$ for all integers $i \geq -1$.

Claim:

$s_i \geq F_{i+1} + F_{i+2} - 1$ for all integers $i \geq -1$.

Proof by Strong Induction on i:

Base Cases ($i = -1, 0$):
Let $i = -1$.
Then $s_i = s_{-1} = 0 \geq 0 = 0 + 1 - 1 = F_0 + F_1 - 1 = F_{i+1} + F_{i+2} - 1$.
Let $i = 0$.
Then $s_i = s_0 = 1 \geq 1 = 1 + 1 - 1 = F_1 + F_2 - 1 = F_{i+1} + F_{i+2} - 1$.
Thus the base cases hold true.

Inductive Step:
Let k be an integer such that $k \geq 0$.
Inductive Hypothesis:
Suppose that for all integers m such that $-1 \leq m \leq k$, $s_k \geq F_{k+1} + F_{k+2} - 1$.
Inductive Claim:
$s_{k+1} \geq F_{k+2} + F_{k+3} - 1$.

According to equation 3 of assignment 2, $s_{i+1} \geq min(s_{i-1} + s_i + 1, 2s_i + 1)$ for all integers $i \geq 0$. This gives us 2 cases:

Case 1 ($s_{k-1} + s_k + 1 < 2s_k + 1$):
Then $s_{k+1} \geq s_{k-1} + s_k + 1$. It then follows that...

$$\begin{aligned} s_{k+1} &\geq s_{k-1} + s_k + 1 \\ &\geq (F_k + F_{k+1} - 1) + (F_{k+1} + F_{k+2} - 1) + 1 && \text{(By the I.H.)} \\ &= (F_{k+2} - 1) + (F_{k+3} - 1) + 1 \\ &= F_{k+2} + F_{k+3} - 1 \end{aligned}$$

Thus the inductive claim is true.

Case 2 ($s_{k-1} + s_k + 1 \geq 2s_k + 1$)*:*

Then $s_{k+1} \geq 2s_k + 1$. It then follows that...

$$\begin{aligned} s_{k+1} &\geq 2s_k + 1 \\ &\geq 2 \cdot (F_{k+1} + F_{k+2} - 1) + 1 && \text{(By the I.H.)} \\ &= 2 \cdot F_{k+3} - 1 \\ &= F_{k+3} + F_{k+3} - 1 \\ &\geq F_{k+2} + F_{k+3} - 1 \end{aligned}$$

Thus the inductive claim is true.

Insertions

Question 2: Briefly explain why the only nodes in a tree whose heights or balance factors might have changed lie on the path from the a leaf up to the root of the tree.

Both the height and the balance factor of a node are exclusively determined by the connected nodes of a greater depth.

Question 3: Briefly explain why the balance factors of the nodes on the above path are all either 2, 1, 0, -1, or -2.

Assuming the tree was indeed an AVL tree before the insertion, then all nodes of the tree must have a balance factor within $\{1, 0, -1\}$. By inserting a node, the balance factor of any preceeding node will be modified by a value of either 1 or -1, as only one node was inserted. Thus, the extreme values of the set of possible balance factors may become more "extreme" by an absolute value of 1, meaning -1 may become -2 and 1 may become 2. Plus the other values are still attainable.

Question 4: Explain why if v is some node on this path in the tree, and the height of v has not been changed, then the heights and balanced factors of all the nodes on the path that are above v have not been changed either.

Height: The height of a node depends on the maximum height of its children. If neither of the node's children's heights have changed, then the node's own height cannot have changed.

Balance Factor: The balance factor of a node is defined as the height of the left child minus the height of the right child. If neither of the node's children's balance factors have changed, then the node's own balance factor cannot have changed.

Question 5: Comparing Figures 5 and 6, confirm that node α has height h and balance factor 0 after the described rotation.

After the rotation, node α has T_2 as its left child, and T_3 as its right child. No nodes within T_2 or T_3 have been changed, and thus they each maintain the height of $h - 1$, as they did before the rotation. Because each child of α has the same height, the balance factor of α is 0. Also, the maximum height of either child of α is $h - 1$, meaning that the height of α is indeed h.

Question 6: Confirm that the node β has height $h + 1$ and balance factor 0 as well after the described rotation.

Something something copy question 5.

Question 7: Explain why the binary search tree is once again an AVL tree after the described rotation.

...but α had height $h + 2$ before the rotation, not $h + 1$...

Assuming α was the ***only*** *problem node in the tree...*

Since node α now has a balance factor of 0, node β also has a balance factor of 0, and no nodes above or below nodes α and β have been modified, the binary search tree is an AVL tree once again.

Question 8: Confirm that α and β each have a balance factor within $\{1, 0, 1\}$ and height h after this adjustment.

After the rotations, T_1 and T_3 maintain height $h-1$, and both T_{2a} and T_{2b} maintain height of either $h-1$ or $h-2$ (at least one of them must be $h-1$ because γ had height h before the rotations). Because of this, both nodes α and β will have height h. Also, because there is a chance for either T_{2a} or T_{2b} to have height $h-2$ (but not both), node β will have a balance factor of either 0 or 1, and node α will have a balance factor of either 0 or -1. Only one of these nodes can have a balance factor that is not 0.

Question 9: Explain why γ has height $h+1$ and balance factor 0 after this operation. Using this, explain why the resulting tree is an AVL tree after this operation.

Since both children of γ, nodes α and β, have height h after the rotations, node γ must have height $h+1$ after the rotations. Seeing as α and β also have the same height as each other, node γ has a balance factor of 0.

*Assuming α was the **only** problem node in the tree...*

Since node α now has a balance factor within $\{1, 0, 1\}$, node β also has a balance factor within $\{1, 0, 1\}$, node γ also a balance factor of 0, and no nodes above or below nodes α, β, and γ have been modified, the binary search tree is an AVL tree once again.

Question 10: Consider the case that α has balance factor -2, so that the subtree with root α is as shown in Figure 10. Describe two more cases that should probably be called the ***right-right*** case and the ***right-left*** case that might arise, corresponding this one, and describe the adjustments that can be used to produce AVL trees when these cases arise.

Both of these cases are mirrors of the previous two, and can be solved in a similar way.

Seeing as node α has a balance factor of -2, there must be 2 more descendants on the right of α than there are on the left. Because of this, much like the *left-left* and *left-right* cases, node α must have height of $h+2$ for some integer $h \geq 0$. Using this, we can state that the left child of α (T_1) has height $h-1$, and the right child of α has height $h+1$. We will define the right child of α to be β, and define the left child of α to be T_1. Because α is the *deepest* node with a balance factor of either -2 or 2, both children of node β must have had height $h-1$ before the insertion. If this was not true, then either β would have become the deepest node with a balance factor of either -2 or 2, or β would not have had height h before the insertion.

DO WE KEEP THAT LAST SENTENCE?

Right-Right Case:

In this case, node β will have a balance factor of -1 after the insertion. Since β has a height of $h+1$, the left and right children of β (defined as T_2 and T_3) must have heights $h-1$ and h respectively.

This is solved with a left rotation at α. T_1 will remain the left child of α, with T_2 becoming the right child. Node α then becomes the left child of β, with T_3 remaining the right child.

Because both T_1 and T_2 had height $h-1$, node α now has height h and a balance factor of 0. And since both children of β, T_3 and α, have height h, node β now has height $h+1$ and a balance factor of 0.

Thus this problem node has been fixed.

Right-Left Case:

In this case, node β will have a balance factor of 1 after the insertion. Since β has a height of $h+1$, the left and right children of β (defined as γ and T_3) must have heights h and $h-1$ respectively. We will defined the left child of γ as T_{2a}, and the right as T_{2b}. In accordance of the height h of γ, at least one of these children must have height $h-1$, while the other can have either $h-1$ or $h-2$.

This is solved with a right rotation at β, followed by a left rotation at α. T_1 will remain the left child of α, with T_{2a} becoming the right child. T_3 will remain the right child of β, with T_{2b} becoming the left child. Nodes α and β then become the left and right children of γ respectively.

Because node γ had height h before the rotations, at least of one T_{2a} and T_{2b} must have height $h-1$. Also, since α was defined as the *lowest* node with a balance factor outside of $\{-1, 0, 1\}$, the remaining child of γ must either have height $h-1$ as well, or height $h-2$ in order to keep γ's balance factor within the acceptable range. Using this, we can determine that node α will have height h and a balance factor of either 0 or 1, and node β will also have height h and a balance factor of either -1 or 0 after the rotations. Examining further, we can also determine that node γ will have height $h+1$ and a balance factor of 0 after the rotations.

Thus this problem node has been fixed.

Deletions

Question 11: Suppose that you perform a right rotation at α so that the subtree with root β after this adjustment is as shown in Figure 6. Briefly explain why α has height $h+1$ and balance factor 1 after this adjustment.

Question 12: Briefly explain why β has height $h+2$ and balance factor -1 after this adjustment.

Question 13: Briefly explain why the entire binary search tree is an AVL tree after this adjustment.

Question 14: Now consider the case that α has balance factor -2, instead, so that the subtree with root α is as shown in Figure 10. Briefly describe another three cases corresponding to this, along with the adjustments that should be made for each.

Implementing an AVL Tree

Question 15: Briefly describe the structure of an algorithm for an insertion into an AVL tree.

Question 16: Briefly describe the structure of an algorithm for a deletion from an AVL tree.

Question 17: Provide a complete `AVLDictionary.java` class for assessment.