CS 310: AVL Trees

Chris Kauffman

Week 13-1
## BSTs: Our Focus

- AVL and Red Black Trees are complex DSs
- Focus on high-level flavor

<table>
<thead>
<tr>
<th>You Are Responsible For</th>
<th>You Are Not Responsible For</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Principles</td>
<td>- Full insert/remove</td>
</tr>
<tr>
<td>- Pictures</td>
<td>- Many cases</td>
</tr>
<tr>
<td>- What principle is</td>
<td>- Full implementation</td>
</tr>
<tr>
<td>violated?</td>
<td>- Too complex for lecture</td>
</tr>
<tr>
<td>- How would you fix it?</td>
<td>- Likely you’ll just use a library</td>
</tr>
<tr>
<td>- Draw the fixed version</td>
<td></td>
</tr>
<tr>
<td>- A few pseudocode examples</td>
<td></td>
</tr>
</tbody>
</table>
The AVL tree is named after its two Soviet inventors, Georgy Adelson-Velsky and E. M. Landis, who published it in their 1962 paper "An algorithm for the organization of information".
– Wikip: AVL Tree

- A self-balancing tree
- Operations
- Proof of logarithmic height
AVL Balance Property

T is an AVL tree if and only if

▶ T.left and T.right differ in height by at most 1
▶ AND T.left and T.right are AVL trees
Answers

T is an AVL tree if and only if

- T.left and T.right differ in height by at most 1
- AND T.left and T.right are AVL trees

1 Not AVL

Left 0, Right 1

2 AVL

3 Not AVL

Left 2, Right 0

4 AVL

5 Not AVL

80 not AVL

6 AVL

7 Not AVL

Left 2, Right 4

8 AVL

95 not AVL
Nodes and Balancing in AVL Trees

Track Balance Factor of trees

- balance = height(t.left) - height(t.right);
- Must be -1, 0, or +1 for AVL
- If -2 or +2, must fix

Don’t explicitly calculate height

- Adjust balance factor on insert/delete
- Recurse down to add/remove node
- Unwind recursion up to adjust balance of ancestors
- When unbalanced, rotate to adjust heights
- Single or Double rotation can always adjust heights by 1

```java
class Node<T>{
    Node<T> left, right;
    T data;
    int height;
}
```
private AvlNode insert( Comparable x, AvlNode t ){  
    if( t == null ){  
        // Found the spot to insert  
        t = new AvlNode( x, null, null );  
    }  
    else if( x.compareTo( t.element ) < 0 ) {  
        // Head left  
        t.left = insert( x, t.left ); // Recursively insert  
        if( (t.left.height - t.right.height) == 2 ){
            // Unbalanced - must rotate  
            if( x.compareTo( t.left.element ) < 0 ){
                // x went left-left: Single Rotation  
                t = rightRotate( t );
            } else {
                // x went left-right: Double rotation  
                t = leftRightRotate( t );
            }
        }
    }  
    // Symmetric cases for right ...
Example Rotation Codes

// Single Right rotation
Node<T> rightRotate( Node<T> t ) {
    Node<T> newRoot = t.left;
    t.left = newRoot.right;
    newRoot.right = t;
    t.height = Math.max(t.left.height,
                        t.right.height)+1;
    newRoot.height = Math.max(newRoot.left.height,
                               newRoot.right.height)+1;
    return newRoot;
}

// Left-Right Double Rotation:
// left-rotate t.left, then right-rotate t
Node<T> leftRightRotate( Node<T> t ){
    t.left = leftRotate(t.left);
    return rightRotate(t);
}
Rebalance This AVL Tree

Insert 51
Rebalancing Answer

Insert 51

35 Unbalanced, inserted right-left

Right rotate 57

Left rotate 35
Does This Accomplish our Goal?

- **Proposition:** Maintaining the AVL Balance Property during insert/remove will yield a tree with $N$ nodes and height $O(\log N)$
- **Prove it:** What do AVL trees have to do with rabbits?
AVL Properties Give $\log(N)$ height

Lemma (little theorem) *(Thm 19.3 in Weiss, pg 708)*

An AVL Tree of height $H$ has at least $F_{H+3} - 1$ nodes where $F_i$ is the *ith* Fibonacci number.

Definitions

- $F_i$: *ith* Fibonacci number (0,1,1,2,3,5,8,13,...)
- $S$: size of a tree
- $H$: height (assume roots have height 0)
- $S_H$ as smallest size AVL Tree with height $H$

Proof by Induction: Base Cases

- True for $S_0$: root
  - $H = 0$, $F_3 - 1 = 2 - 1 = 1$
- True for $S_1$: root+(left or right)
  - $H = 1$, $F_4 - 1 = 3 - 1 = 2$
Consider an Arbitrary AVL tree $T$

- $H$ is height of $T$
- $S_H$ smallest size for tree $T$
- Is smallest size $S_H = F_{H+3} - 1$
- Assume equation true for smaller trees
  - Left/Right are smaller AVL trees
  - Left/Right differ in height by at most 1
Induction Part 2

- $T$ has height $H$
- Suppose Left is 1 higher than Right
- Left Height: $H - 1$
  - Left Size: $F_{(H-1)+3} - 1 = F_{H+2} - 1$
- Right Height: $H - 2$
  - Right Size: $F_{(H-2)+3} - 1 = F_{H+1} - 1$

\[
S_H = \text{size(Left)} + \text{size(Right)} + 1 \\
= (F_{H+2} - 1) + (F_{H+1} - 1) + 1 \\
= F_{H+2} + F_{H+1} - 1 \\
= F_{H+3} - 1 \quad \blacksquare
\]
Fibonacci Growth

AVL Tree of with height $H$ has at least $F_{H+3} - 1$ nodes.

- How does $F_H$ grow wrt $H$?
- Exponentially:
  \[ F_H \approx \phi^H = 1.618^H \]
- $\phi$: The Golden Ratio
- So, \( \log(F_H) \approx H \log(\phi) \)
- Or, \( \log(N) \approx \text{height} \times \phi \)
- Or,
  \[ \log(\text{size}) \approx \text{height} \times \text{constant} \]
Next few slides...

- Adapt proof of logarithmic height for single nodes with height 1
- Nearly identical to previous proof with a few number changes
AVL Properties Give $\log(N)$ height

Lemma (little theorem) (*Thm 19.3 in Weiss, pg 708*)
An AVL Tree of height $H$ has at least $F_{H+2} - 1$ nodes where $F_i$ is the *ith* Fibonacci number.

Definitions

- $F_i$: *ith* Fibonacci number (0,1,1,2,3,5,8,13,...)
- $S$: size of a tree
- $H$: height (assume roots have height 1)
- $S_H$ as smallest size AVL Tree with height $H$

Proof by Induction: Base Cases

- True for $S_1$: root
  - $H = 1$, $F_{1+2} - 1 = 2 - 1 = 1$
- True for $S_1$: root+(left or right)
  - $H = 2$, $F_{2+2} - 1 = 3 - 1 = 2$
Consider an Arbitrary AVL tree $T$

- $H$ is height of $T$
- $S_H$ smallest size for tree $T$
- Is smallest size $S_H = F_{H+2} - 1$
- Assume equation true for smaller trees
  - Left/Right are smaller AVL trees
  - Left/Right differ in height by at most 1
Induction Part 2

- $T$ has height $H$
- Suppose Left is 1 higher than Right
- Left Height: $H - 1$
- Left Size:
  \[ F_{(H-1)+2} - 1 = F_{H+1} - 1 \]
- Right Height: $H - 2$
- Right Size:
  \[ F_{(H-2)+2} - 1 = F_H - 1 \]

\[
S_H = \text{size}(\text{Left}) + \text{size}(\text{Right}) + 1
= (F_{H+1} - 1) + (F_H - 1) + 1
= F_{H+1} + F_H - 1
= F_{H+2} - 1 \quad \blacksquare
\]