Topic 08: AVL Trees

- AVL Property
- Insertion
- Deletion
- Summing it all up
AVL Trees

• AVL trees are special BSTs:
  - Keep track of **height** of every node
  - Never let left child & right child differ by more than 1

• **Leads to** $h = O(\lg n)$

AVL stands for “Georgy Adelson-Velsky and Evgenii Landis”
AVL Trees

Total Tree Height = 4
AVL Trees

This example generally assumes that the left tree is bigger...

But AVL trees don't require this. Either can be larger!
AVL Trees

h=4

h=3

height = 2

height = 1

height = 2

height = 2
AVL Trees
AVL Trees

h=0

h=1

height = 0

height = 1

h=2

height = 1

h=3

height = 2

height = 1

h=4

height = 2
AVL Trees
AVL Trees

height = 2

h=4

h=3

h=2

h=1

h=1

h=0

h=0

h=0

h=0
AVL Trees
Insight:
“Balanced” doesn't mean “perfect.”

This tree has only 12 nodes...out of 31 possible! (height=4)
AVL Trees

Claim: The height of an AVL tree is $O(\lg n)$

**Hint:** “Prove that in an AVL tree of height $h$, there are at least $F_h$ nodes, where $F_h$ is the $h$th Fibonacci number.” (Fibonacci numbers grow exponentially!)

- $F_0 = 0$
- $F_1 = 1$
- $F_2 = 1$
- $F_3 = 2$
- $F_4 = 3$

**In-Class:**
Break into groups, and prove this claim by structural induction.
Proof:

**Base Case (height = 0):**
A leaf node has height=0, and exactly one node.

\[ F_0 = 0 \], so this base case is obviously true.

**Base case (height = 1):**
A tree of height 1 has either 2 or 3 nodes.

\[ F_1 = 1 \], so this base case is also obviously true.
Proof (cont'd):

Inductive Case (height $\geq 2$):

Consider a subtree with height $h \geq 2$. Since the height of this node is $h$, at least one of the node's children has height $h-1$; the height of the other is either $h-1$ or $h-2$. We choose to assume that it has height $h-2$, since that will give us the smallest lower bound.
AVL Trees

Proof (cont'd):

Inductive Case (height \( \geq 2 \)):

By the I.H. we know that the first child has \( \textbf{at least} \) \( F_{h-1} \) nodes, and the second has \( F_{h-2} \). Thus, this subtree has \( \textbf{at least} \)

\[
1 + F_{h-1} + F_{h-2}
\]

nodes, which is (by definition of Fibonacci) greater than \( F_h \). Thus, the inductive case also holds.
Proof (cont'd):

Conclusion:

We have shown that an AVL tree of height \( h \) has at least \( F_h \) nodes.

We know that the Fibonacci sequence grows exponentially. Thus, we know that

\[
 n = \Omega(2^h)
\]

and thus

\[
 h = O(lg \ n)
\]
Topic 08: AVL Trees

- AVL Property
- Insertion
- Deletion
- Summing it all up
AVL Trees

• How to insert into an AVL Tree?
  - Insert new leaves
  - Update heights
  - Do rotations if necessary
In these two trees...

Any insertion will cause the height to grow.

But no insertion can possibly violate the AVL property.
Inserting into an AVL Tree  
(Examples)

But what about this tree?  
Some insertions require rotations to fix them.
Inserting into an AVL Tree  
(Examples)

Nothing to fix!
Inserting into an AVL Tree (Examples)

Inserting node C into this tree causes an imbalance at A. Can you see why?
Inserting into an AVL Tree
(Examples)

Rotate right at A to fix this tree.
Inserting node C into this tree causes an imbalance. But rotating right at A would **not** help us.

Can you see why?
Inserting into an AVL Tree (Examples)

Rotate left at B, and then this is equivalent to the previous case.
Inserting into an AVL Tree
(Examples)

Rotate left at B, and then this is equivalent to the previous case.
Inserting into an AVL Tree
(Examples)

Conclusions

Rotations \textbf{NEVER} required.

But the height grows!

Rotations sometimes required.

But the height \textbf{NEVER} grows!
A Cool Property of Insertions...

If two children have the same height before insertion...

- One might grow (and thus the whole subtree grow)
- But no rotation will be required (at this level)

If the two children have different heights **AND** we insert into the taller one...

- Rotation might be required
- But the overall height will never grow

But can you prove it?
(we already proved the base cases...)
Proof Strategy

• We will use a **Proof by Cases**
  - Draw a general a picture as we can
  - Start by handling the simplest cases
  - Keep going until all possibilities have been covered
Proving the Theorem
(Base Cases)

Empty Tree → h=0 → h=0 → h=1

h=0
Proving the Theorem
(Inductive Step)

Two Possible Forms
BEFORE Insertion

Imbalance
Impossible

Imbalance Possible
(if we insert on this side)

Imbalance Impossible
(if we insert on this side)
Proving the Theorem
(Inductive Step)

If any imbalance developed, then we must have inserted into the taller subtree.
By the I.H., the only way that an insertion can cause this subtree to grow larger is if its two children are the same height.
Thus, we know that both subtrees have height $h-2$. 
Case 1:
We inserted the new node into the left subtree, and it grew to height $h-1$.

Rotate right at the root!
Proving the Theorem (Inductive Step)

After Rotation...

A
WAS: h-2
NOW: h-1

B
h-2

C
h-2
Proving the Theorem (Inductive Step)

After Rotation...
Fill in the new heights.
Note that the total height **did not** increase!

A
**WAS**: h-2
**NOW**: h-1

B  

**h-2**

C  

**h-2**

h

h-1
Case 2: We inserted the new node into the right subtree, and it grew to height $h-1$. 
Case 2 (cont'd):
By the I.H. (again!), both children of B had height $h-3$ before the insertion.

One of them has grown to $h-2$.

NOTE:
B1,B2 might be swapped...it won't matter in the rotations that follow.
Case 2 (cont'd): We start with a left rotation.
Proving the Theorem
(Inductive Step)

Case 2 (cont'd):
Now a right rotation is possible...

<table>
<thead>
<tr>
<th>Diagram Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>h-2</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>h-2</td>
</tr>
<tr>
<td>B1</td>
</tr>
<tr>
<td>h-3</td>
</tr>
<tr>
<td>B2</td>
</tr>
<tr>
<td>h-3</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>h-2</td>
</tr>
</tbody>
</table>

WAS: h-3
NOW: h-2
Proving the Theorem
(Inductive Step)
Proving the Theorem (Inductive Step)

Case 2 (cont'd):
Finally, recalculate heights.

The total height didn't grow!
A Cool Property of Insertions...

<table>
<thead>
<tr>
<th>If two children have the same height before insertion...</th>
<th>If the two children have different heights AND we insert into the taller one...</th>
</tr>
</thead>
<tbody>
<tr>
<td>• One might grow (and thus the whole subtree grow)</td>
<td>• Rotation might be required</td>
</tr>
<tr>
<td>• But no rotation will be required (this level)</td>
<td>• But the overall height will never grow</td>
</tr>
</tbody>
</table>

Proved!
AVL Trees – Insertion (Review)

- To insert in an AVL tree:
  - Do normal insertion
  - Do “rebalance” back toward the root
    - Increase some heights
    - Maybe do 1 or 2 rotations
      - End updates after the rotations

- Cost:
  - $O(lg \ n)$ to insert
  - $O(lg \ n)$ to update height fields
  - $O(1)$ to do rotations
Topic 08: AVL Trees

- AVL Property
- Insertion
- Deletion
- Summing it all up
Deletion in an AVL Tree

Two steps to deletion:

- Delete
- Rebalance
  - Same rebalance algorithm as insert, except:
  - Rotations a multiple levels sometimes required!
Multiple Rotations

- As with insertion, our algorithm works up from the deletion point
  - Update heights
  - Perform rotations where there are imbalances

- The rotation after a deletion can reduce height of a subtree – which means that multiple rotations are possible
Multiple Rotations

Suppose we delete L from the tree.
Multiple Rotations

This creates an imbalance at K, which we fix with a single rotation.
This creates a new imbalance at H, which we fix with another rotation.
Multiple Rotations

This reduced the size of the tree – meaning that we could have had even more rotations, further up.
AVL Trees – Deletion

• To delete from an AVL tree:
  - Do normal deletion
  - Do “rebalance” back toward the root
    • Increase some heights
    • Do rotations (maybe at multiple levels)

• Cost:
  - $O(lg \ n)$ to delete
  - $O(lg \ n)$ to update height fields
  - $O(lg \ n)$ to do rotations
Topic 08: AVL Trees

- AVL Property
- Insertion
- Deletion
- Summing it all up
AVL Trees (summary)

- AVL Property: child heights differ by no more than 1
  - Does **not** balance the tree
  - But close enough
  - Guaranteed: \( h = O(\lg n) \)

- Insertion/Deletion
  - Do the normal op
  - Then “fixup” back toward the root
AVL Trees (summary)

- Two rotation types during an AVL rebalance:
  - Single rotation at the root (of the subtree)
  - Double rotation, first at child, then root

- Choose between them by checking which grandchild-subtree is largest
AVL Trees (summary)

The result:

Insert, Delete, Search

all guaranteed

$O(\lg n)$ cost!