AVL Trees

CSE 373
Data Structures and Algorithms

Today’s Outline

• Announcements
  – Assignment #2 due Fri, Jan 23, at the beginning of lecture.
  – Midterm Dates:
    • Midterm #1: Friday, Jan 30th
    • Midterm #2: Friday, February 27th

• Today’s Topics:
  – Binary Search Trees
  – Balanced Binary Search Trees - (AVL Trees)

The AVL Balance Condition

Left and right subtrees of every node have equal heights differing by at most 1

Define: balance(x) = height(x.left) – height(x.right)

AVL property: -1 ≤ balance(x) ≤ 1, for every node x

• Ensures small depth
  – Will prove this by showing that an AVL tree of height h must have a lot of (i.e. Θ(2^h)) nodes
• Easy to maintain
  – Using single and double rotations

The AVL Tree Data Structure

Structural properties
1. Binary tree property
2. Balance property:
   balance of every node is between -1 and 1

Result:
Worst case depth is Θ(log n)

Ordering property
  – Same as for BST

Proving Shallowness Bound

Let S(h) be the min # of nodes in an AVL tree of height h

Claim: S(h) = S(h-1) + S(h-2) + 1

Solution of recurrence: S(h) = Θ(2^h)
(like Fibonacci numbers)
Testing the Balance Property

We need to be able to:
1. 
2. 
3. 

NULLs have height -1

AVL trees: find, insert

• AVL find:
  – same as BST find.
• AVL insert:
  – same as BST insert, except may need to “fix” the AVL tree after inserting new value.

AVL tree insert

Let x be the node where an imbalance occurs.

Four cases to consider. The insertion is in the
1. left subtree of the left child of x.
2. right subtree of the left child of x.
3. left subtree of the right child of x.
4. right subtree of the right child of x.

Idea: Cases 1 & 4 are solved by a single rotation.
Cases 2 & 3 are solved by a double rotation.

Case “#1”

Insert(6)
Insert(3)
Insert(1)

Fix: Apply Single Rotation

AVL Property violated at this node (x)

Single Rotation:
1. Rotate between x and child
Single rotation in general

<table>
<thead>
<tr>
<th>Height of tree before?</th>
<th>Height of tree after?</th>
<th>Effect on Ancestors?</th>
</tr>
</thead>
</table>

Single rotation example

Case “#3”

Insert(1)
Insert(6)
Insert(3)

Fix: Apply Double Rotation

AVL Property violated at this node (x)

Double Rotation
1. Rotate between x’s child and grandchild
2. Rotate between x and x’s new child

Double rotation in general

Double rotation, step 1
Double rotation, step 2

Imbalance at node X

Single Rotation
1. Rotate between x and child

Double Rotation
1. Rotate between x’s child and grandchild
2. Rotate between x and x’s new child

Insert into an AVL tree: a b e c d

Single and Double Rotations:
Inserting what integer values would cause the tree to need a:
1. single rotation?
2. double rotation?
3. no rotation?

Insertion into AVL tree
1. Find spot for new key
2. Hang new node there with this key
3. Search back up the path for imbalance
4. If there is an imbalance:
   case #1: Perform single rotation and exit
   case #2: Perform double rotation and exit
Both rotations keep the subtree height unchanged. Hence only one rotation is sufficient!
Hard Insert (Bad Case #1)

Insert(33)

Unbalanced?
How to fix?

Single Rotation

Hard Insert (Bad Case #2)

Insert(18)

Unbalanced?
How to fix?

Single Rotation (oops!)

Double Rotation (Step #1)

Double Rotation (Step #2)