CSE 373

APRIL 19TH – AVL OPERATIONS
ASSORTED MINUTIAE

• HW2 code grades out tonight
• HW3 due tonight
  • Last HW before midterm
• Exam review
  • Next Wednesday (in class)
  • Options for TA review session out tonight
TODAY’S LECTURE

• AVL Trees
  • Balance
  • Implementation

• Memory analysis
  • Will discuss after AVL on Friday
REVIEW

• **AVL Trees**
  • BST trees with AVL property
  • $\text{Abs}(\text{height(left)} - \text{height(right)}) \leq 1$
  • Heights of subtrees can differ by at most one
  • This property must be preserved throughout the tree
• Is this an AVL Tree? Yes!
  • Calculate balance for each node
• What about this one?
• What about this one?
  • No, 8 is out of balance
• Is this an AVL Tree?
• Is this an AVL Tree?
  • No, AVL trees must still maintain Binary Search
AVL OPERATIONS

- Since AVL trees are also BST trees, they should support the same functionality
  - Insert(key k, value v)
  - Find(key k): Same as BST!
  - Delete(key k): Not presented in this course
- For insert, we should maintain AVL property as we build
AVL OPERATIONS

• Insert(key k, value v):
  • Insert the key value pair into the dictionary
  • Verify that balance is maintained
  • If not, correct the tree

• How do we correct the tree?
AVL INSERT

- Start with the single root
• Add 7 to the tree. Is balance preserved?
  • Yes
AVL INSERT

- Add 9 to the tree. Is balance preserved?
  - No.
• How do we correct this imbalance?
  • Important to preserve binary search
AVL INSERT

- What shape do we want?
  - What then do we have as the root?
• Since 7 must be the root, we “rotate” that node into position.
AVL “ROTATION”

• To correct this case:
  • B must become the root
  • We rotate B to the root position
  • A becomes the left child of B
  • This is called the “left rotation”
AVL “ROTATION”

• Right rotation
  • Symmetric concept
  • B must become the new root
AVL “ROTATION”

• These are the “single” rotations
  • In general, this rotation occurs when an addition is made to the right-right or left-left grandchild
  • The balance might not be off on the parent! An insert might upset balance up the tree
AVL “ROTATION”

• General case
  • Suppose this tree is balanced, \{X,Y,Z\} all have the same height
AVL “ROTATION”

• **General case**
  - Suppose this tree is balanced, \{X,Y,Z\} all have the same height
  - Adding A, puts C out of balance
  - Rotate B up and pass the Y subtree to C
**AVL “ROTATION”**

- **General case**
  - Suppose this tree is balanced, \{X,Y,Z\} all have the same height
  - Adding A, puts C out of balance
  - Rotate B up and pass the Y subtree to C
  - **Perform this rotation at the lowest point of imbalance**
• Consider the above tree
  • Is it an AVL tree? Yes
• Add 16 to the tree
  • Is it unbalanced now? Where? 22
  • Also at 15, but we choose the lowest point
SINGLE ROTATION EXAMPLE

- Perform the rotation around 22
  - What rotation takes place?
• Perform the rotation around 22
  • What rotation takes place?
  • What is the resulting tree?
SINGLE ROTATION EXAMPLE

- 19 must move up to where 22 was
  - 20 changes parents
  - Balances are recomputed throughout the tree
AVL “ROTATION”

• These two rotations (right-right and left-left) are symmetric and can be solved the same way
  • Named by the location of the added node relative to the unbalanced node
  • What are the other two cases?
AVL “ROTATION”

• Right left case
  • Again, A is out of balance
  • This time, the addition (B) comes between A and C
  • In this case, the grandchild must become the root.
AVL “ROTATION”

• Identifying what should be the new root is key
• Imagine “lifting” up the root
• Where will the children have to go to maintain the search property?
AVL “ROTATION”

• This is for your reference later.
AVL “ROTATION”

- Let’s do an example. Insert(13)
AVL “ROTATION”

- Where is the imbalance?
AVL “ROTATION”

- Where is the imbalance?
AVL “ROTATION”

• Where is the imbalance? (also 7 and 10)
AVL “ROTATION”

- What must be the new root?
AVL “ROTATION”

- What must be the new root?
• What must be the new root? Why?
AVL “ROTATION”

• What does the new tree look like?
AVL “ROTATION”

• The replaced root is always a child of the new root!
NEXT CLASS

• AVL Trees
  • Even more examples!
  • Showing that this actually gives us $O(\log n)$ height
  • Showing insert is $O(1)$

• Memory analysis
  • Formalization to help with confusion from last week