AVL Trees

4-15-2002

Opening Discussion

- What did we talk about last class?
- Do you have any questions about the assignments?
- Your minute essay last class was solving a maze using backtracking. To do this, at each step we recurse in all open directions. We also need to add logic to not go into squares that are higher up in the call stack.

Balanced Trees

- We have already discussed how a sorted binary tree can degenerate into a linked list with the wrong data. While this isn't very likely, it is still something we would like to prevent.
- If nothing else, keeping trees perfectly balanced also gives us a 38% speed boost over average performance. As such, we will look at balanced trees the next two days.

AVL Trees

- The first approach we take to take to help with the balance of trees is the AVL tree (Adelson-Velskii and Landis).
- For this tree we make sure that at each add the difference between the heights of the left and right branches of any node differ by no more than 1. We preserve this property by applying so called rotations at adds.

Approach

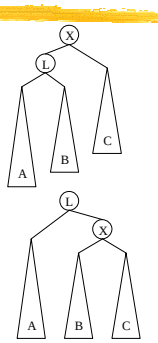
- The basic idea is that when we add something to the tree, the only nodes that could become "unbalanced" are those that we pass through on the traversal to where the node goes. In each node we store a height so we can quickly determine if something needs to be changed.
- The changes are made by moving a node from the high side to the low side, but it has to be done in a way to preserve the sorting. This is what the "rotations" do.

Four Cases for Insert

- When we find a node that is unbalanced we can easily describe four different ways in which it happened, all are based on where the new node went in relation to the children of the unbalanced node, X.
  - 1 - left of left child
  - 2 - right of left child
  - 3 - left of right child
  - 4 - right of right child

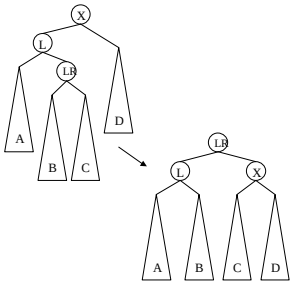
Single Rotation

- For cases 1 & 4 (outside branch too long) we can perform a single rotation. We will look at case 1.
- We "rotate" the left child up in place of X and make X its right child. We then make what had been the right child of the node we are moving the left child of X.



Double Rotation

- For cases 2 & 3 we need to do something slightly more complex.
- We bring the node from two levels down up to the top.



Removes?

- Your book does not discuss removing nodes from the tree. However, it is easy to see how that works. A remove can produce any of the same "situations" that were present in the add, but "in reverse".
- We perform the same rotations for the same situation. The main thing that makes this more complex is that it doesn't fit as easily into the recursion because things can move at two places in a delete.

Minute Essay

- Show me the trees that result from adding the numbers 1-5 to a sorted binary tree that is originally empty. Draw the whole tree for each of the additions.
- Assignment #5 is due today. You will get 10 extra points if you turn it in before midnight.