Red-Black Trees

10-9-2003

Opening Discussion

- Do you have any questions about the quiz?
- What did we talk about last class?
- Do you have any questions about the assignment?
- Should the test be open book/notes?

Rules of Red-Black Trees

- Every element is either red or black.
- The root is black.
- Every NULL child is considered black.
- If a node is red then both its children must be black.
- All paths from a node to a leaf under it must have the same number of black nodes.

Rotations

- We will be using the same single rotations with red-black trees that we used with AVL trees. Only the situations doesn't have to be the same.
- Can someone describe a single rotation to me?

Insertion

- As with an AVL tree we begin by inserting the new node in the way of a normal sorted binary tree. After that we have to clean up the tree in some way if we have violated the properties of a red-black tree.
- To fix the tree we do something similar to what was done in the AVL tree, but using colors instead of heights. We walk up the tree and fix the violations that we find.

Insertion Fixing

- At the beginning of every loop our "current" node is red and if the parent is the root it is black. There is also never more than one violation of the red-black properties. Generally it is a red node having a red child.
- The advantage of the red-black tree is that we never do more than 2 rotations to fix the tree.

The Loop

- While the parent is red we do a loop.
  - If we are to the left of our grandparent.
    - If our uncle is red we make our parent and our uncle black and our grandparent red, then make our grandparent the current.
  - Otherwise if we are to the right of our parent move to the parent and rotate the current left. Either way we make our parent black and our grandparent red then rotate right.
  - Mirror this logic if we are on the right.
- Finish by setting the root black.

Deletion

- The delete your book uses is a bit different from what we did. The main difference is that they don't actually move the node we are deleting. Instead, they change the value stored in it. This has less impact on the red-black structure, though we can make ours look like that by remembering to set the color in the replacement node to what it is replacing.

Deletion Fixing

- After the delete we again have to fix things. This function is distinctly different from the fixing that is used in insertion. In this regard, the code can be more complex than the AVL tree.
- We'll go into the details of this next class.

Code

- We can spend some time validating the binary tree code that we have now and then start making a red-black tree from it once we know that it works.

Minute Essay

- Do you think that balanced binary trees are worth the trouble?