Binary Trees

9-30-2003

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? Compare this semester's project to that from last semester. You have less freedom in what you do (though you still have a fair bit of freedom in that), but you have a lot more freedom in how you do it.

Trees

- A data structure that has a single root and a single path from any node to the root is a tree.
- We mix metaphors for biological trees and family trees when discussing them. We also draw them upside down.
- Terms: root, leaf, child, parent, sibling, ancestor, descendent, height, depth, size.

Binary Trees

- The most common type of tree that we use is a binary tree where a node can have two children called left and right.
- We can traverse these trees by recursively going to the left, then the right. Depending on when we "visit" the current node we get a pre-order, in-order, or post-order traversal.
- A breadth first traversal uses a queue instead of a stack.

Sorted Binary Trees

- Also called a binary search tree, these trees have a complete ordering of their elements and things that are less go to the left while things that are greater go to the right.
- A search on this structure starts at the root and walks down, picking the proper side to step to until the desired node is found. This has average case O(log n) performance.

Successor and Predecessor

- Unlike a hash table, we can efficiently find the successor of any given element of a binary tree. A similar algorithm can be used for predecessor.
- If the node has a right child the successor is the smallest element on the right. If not then we walk up until we find a node that this one wasn't to the left of.
- These both to O(log n) time.

Insertion

- To insert we act like we are searching until we get to the place the node should be. Once we get there we place the node.
- Obviously we have to be a bit more careful because we can't go off the edge of the structure before adding, but the idea is basically the same.

Deletion

- Deletion is easily the hardest operation on a binary search tree. We have to replace the current node with either the largest element on the left or the smallest on the right.
- If nodes keep track of their parents this can also be greatly simplified with functions that find the smallest or largest element of a subtree.

Minute Essay

- What do you see as the biggest problems with binary search problems?
- Try to get assignment #2 to a place where it compiles and does a fair bit of the functionality so you can submit it. Don't worry too much if it isn't all complete, but there will be comparative grading so if you have much less than some others it won't reflect well.