Data Structures with C++ : CS189

Lecture 7-1: Binary Search Trees

Recap

- A Vector was an array that could grow.
  - O(1) Add
  - O(n) Insert
  - O(1) Access
  - O(n) Find
- A List was a chain of nodes
  - O(1) Add
  - O(1) Insert
  - O(n) Access
  - O(n) Find

Binary Search Tree

- A BST is a structure where every piece of data has up to two children.
  - Draw this as an upside down tree with a single root (at the top)
- Everything lower is somewhere on the left, and everything greater is on the right, no duplicates
- We get O(log n) for anything involving lookups because each left-right decision skips half the tree
  - "Double" or "half" is a sure sign of log
  - "The time taken doesn't increase until the input size doubles."

Binary Search Trees in STL

- BST is not an STL container on its own, it's the technology
  - Like Vector was based on arrays
- A Set is the simplest ADT that *uses* a BST, so we'll push that off for a day and concentrate on just the basic workings of a tree

BST Code

```
struct TreeNode {
        T mData;
        TreeNode *mLeft;
        TreeNode *mRight;
};
```

- There are many different ways to implement a tree
  - Information Hiding!
- We are going to have a Node like List, and two pointers to other Nodes
  - We'll call ours right and left instead of next and prev though
  - Everything smaller is on my left, everything larger is on my right
- Next week we'll make sure a tree can't get really lopsided by giving it the ability to change itself

Important Tree Methods

- Insert - Put this data at the right position
- Erase - Remove this data from the tree
- Contains - Is this data in the tree?
  - Real STL has "Find", but Find requires something from next week so we'll have just true/false
- And of course the Big 3
  - And also of course, this is only a 3 for now. Next to last week we go over the new features of C++, and I think they have 5 or 7.

Implementation

- If I start by making a pointer to the root, I can get to everything
- If I want to add/erase/find data to my left, I "go" or "walk" left
  - I use the term "walk" for when you have a pointer to a node and then set it to left or right's value. Draw the tree and "walking" is moving your finger along the arrows
    - I don't say "move" because that means something else
- When there is nothing to one side of a node, it is set to nullptr
  - A node with no left or right is called a "leaf"

This Homework

- The Remove I gave you just sets a "Dead" flag on a node
  - That is, of course, not correct
  - Part of the homework is to fix that
- Tomorrow we will talk about actually removing nodes from a tree

Other Homework

- List and Vector were written from scratch to warm you up
- The Tree assignments aren't from scratch. I'm giving you a file that you have to finish/optimize
- For the Graph assignments I'm going to give you the entire solution… with 10 bugs somewhere in it that you have to find
- After that the assignments use STL entirely for the algorithms part of the class

End

Make sure we go over the homework file I'm giving you