Intro to Data Structures

Lecture #19 – Trees and BST’s (intro)
November 10, 2013

Mark Stehlik

Outline for Today

- Intro to Trees
- Binary Search Trees (another data structure)

Trees (intro)

- Some terminology
  - The first (start) node is called the *root* of the tree
  - An *empty tree* would have no nodes
  - The successors (next nodes) are called *children* (parent, siblings)
  - A node with no children is called a *leaf*
  - A node that is not a leaf is an *interior* (*internal*) node
  - *Subtree* - a tree formed by a node and its descendants
  - Let's draw some trees…

Trees (intro)

- Some more terminology
  - Like a linked list, a tree has n nodes and n-1 links (edges)
  - *Path* - a sequence of nodes (the length of the path is the number of edges; n nodes --> path length of n-1); there is exactly one path from the root to each node
  - *Depth/level* (of a node/tree) - length of the path from the root to the node (depth of the root == 0)
  - *Height* (of the tree) - the maximum path length from the root to any node (height of 1-node tree == 0)

Binary trees

- We're interested in *binary* trees
  - A node can have 0, 1, or at most 2 children
  - Recursively, a binary tree is either
    - empty or
    - it has a root node, a left binary tree and a right binary tree
- Representation
  - Array - root at index 1; children of root are at 2 and 3; in general, for a node at index *k*, its children are at *2k* (left) and *2k+1* (right), and its parent is at *k/2*
  - Linked structure - left/right nodes

Binary Search trees

- To search efficiently, we are interested in binary trees with an ordering defined on the children of a node (called *binary search trees*)
  - the left subtree of a node will contain values less than the node
  - the right subtree of a node will contain values greater than the node
- Representation
  - Linked structure - left/right TreeNodes

Implementing a generic BST class

- OK, so for a BST node, what do I need to store?
  - Data
  - Left/right links
- And to create a BST?
  - a reference to the root (a TreeNode)
  - and where should the TreeNode class be declared?
- And we'll want it to be generic, so it will take <Anything>

Implementing a generic BST class

- So what has to be true of <Anything> this time?
  - it must be Comparable
- What methods do we implement on a data structure?
  - constructor
  - isEmpty
  - add
  - traversal (how many are there?)
  - size/count (we could update/store a value, but we'll do O(n))
  - contains/find (O(?))
  - toString
  - remove