Trees (Sec 5.4, 5.6, 5.7, 12.5-6, 12.9, 16.3?)

Philip Chan

1 **Terminology (Sec 5.4)**
   - “Rooted tree” or just “tree”
   - nodes/vertices, edges/links, parent, child, root, leaf (no children)
   - level (root is level 0), height (largest level number) [Def 5.6, p241]
   - subtree
   - binary tree: each node has (at most) two children

2 **Tree Traversal (Sec 5.6)**
   - given the root of a tree, process every node in the tree systematically
   - preorder (depth-first): visit node, visit left subtree, visit right subtree (Program 5.14, p244)
   - inorder: visit left subtree, visit node, visit right subtree
   - postorder: visit left subtree, visit right subtree, visit node
   - can use a stack for non-recursive implementations of pre, in, postorder traversals
   - level-order (breadth-first): left to right, top to bottom (Program 5.16, p248)

3 **Recursive binary tree algorithms (Sec 5.7)**
   - tree is a recursive data structure, recursive algs are usually natural
   - counting the nodes and height of a tree (Program 5.17, p250)
   - printing (Program 5.18, p251)

4 **Binary Search Trees (Sec 12.5-6)**
   - given any tree node $x$, all nodes in the left subtree are smaller than $x$, and $x$ is smaller than all nodes in the right subtree (Def 12.2, p516)
   - Why this arrangement?
   - What are the operations for BST?
   - member/search
     - Worst-case of a balanced BST: $O(\log N)$ comparisons
     - Average-case of a balanced BST: $O(\log N)$ comparisons (Property 12.6, 522)
- Worst-case of a degenerate BST: $O(N)$ comparisons (Property 12.8, p523)
- Average-case of a degenerate BST: $O(N)$ comparisons

- insert: (Program 12.10, p520)
- delete/remove: three cases [book’s alg is more complicated (Program 12.16, p538)]
  - no subtrees: just remove
  - one subtree: replace the deleted node by the subtree
  - two subtrees: find the smallest key in the right subtree, replace the key in the to-be-deleted node with the smallest key, remove the node with the smallest key
- analysis of insert and delete/remove is similar to member/search

5 Balanced BST (p.571 and handout)
- AVL-tree (Adelson-Velskii and Landis)
- the heights of the two subtrees of each node differ by at most 1
- Insertion creates:
  - Outside imblanace (from left-left, right-right insertion): single rotation
  - Inside imbalance (from left-right, right-left insertion): double rotation

if we have time, we’ll discuss B Trees

6 B Trees (Sec 16.3)
- Records are stored on disk
- Minimize I/O time to the disks