CSI 33 LECTURE NOTES (Ojakian)

Topic 7: Trees

OUTLINE
(References: Ch. 6, 7)

1. Trees
2. Preorder, postorder, inorder
3. Binary Search Tree

---

1. Trees: Basic Terminology
   (a) Vertices, Nodes, Edges
   (b) Rooted vs. non-Rooted Tree
   (c) Root, Parent, Child, Sibling
   (d) Binary Tree
2. Programming Binary Trees
   (a)
   PROBLEM 1. *Create just a TreeNode class and build a tree from scratch.*
   (b) Typical "addressing" in binary trees.
   PROBLEM 2. *Create a BinaryTree class that uses this addressing method to: 1) add nodes and 2) get node data.*
3. Application: Data Compression
   PROBLEM 3. *Consider Problem 8 (pages 249 - 251), and discuss how to do* **without** *data compression.*
   PROBLEM 4. *Reconsider the last problem, but now* **with** *data compression.*
   PROBLEM 5. *Use the BinaryTree class to program the data compression.*
4. Recursion
   (a) Wrapper function and Recursive Helper function
   PROBLEM 6. *See recursion examples: Fibonnaci and list sums*
   (b) Check the recursion works using idea of Proof-By-Induction
   PROBLEM 7. *Verify the last recursion examples (Fibonnaci and list sums) using Proof-By-Induction.*
   (c) Recursion on Linked Structures:
      i. Pass a node reference to the recursive function
      ii. Function returns a node reference to the properly modified linked structure

iii. Notice pattern: link passed-in and the link modified are the same

iv.

PROBLEM 8. *Add a recursive append and delete to linked list, thinking inductively.*
Then verify the operation of the functions using Proof-By-Induction.

5. Expression Tree and Traversals

(a)

PROBLEM 9. *For example from book (p. 277, Figure 7.4) do three kinds of traversal.*

(b) Inorder - corresponds infix expression

(c) Postorder - corresponds postfix expression

(d) Preorder - corresponds prefix expression

(e)

PROBLEM 10. *Program in-order. Leave other two as group work.*

6. Binary Search Tree

(a) Its defining property with examples.

PROBLEM 11. *From diagrams, which are and which are not BSTs?*

(b) Recursive and non-recursive insertion: Do examples, code, and prove

(c) Recursive deletion:

i. First just code and discuss the case where the deleted node has **at most ONE child**

ii. Discuss theory for case: deleted node has TWO children

iii. Code nust a deleteMax method

iv. Then do full delete method

7. Application: Machine Learning and Decision Trees

For all this, see the coding example.

(a) Example

PROBLEM 12. *Discuss a basic decision tree for guessing what animal someone is thinking of (more to come on decision trees later ...).*

(b) Introduction to Data Frames.

(c) The goal of Machine Learning Classification via the example Data Frame: From features to target.

(d) Majority Vote and Node Purity (understood probabilistically)

(e) How to split node on a feature: Maximize purity (understood probabilistically)