CSCI-GA.1170-003/004 Fundamental Algorithms March 13, 2020

Solutions to Problem 1 of Homework 6 (15 points)

Name: Rahul Zalkikar (rz1567) *Due: 8 pm on Friday, March 13*

Collaborators: ua388, jni215

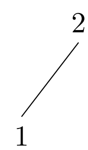
Assume that you are given a binary tree where the nodes are only labeled with either L denoting "leaf node", or I denoting "internal node". You are given the preorder tree walk of the tree.

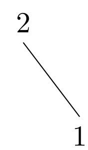
(a) (1 point) Is there a unique tree for a given preorder tree walk? If yes, provide an algorithm to construct the tree. If no, provide a counter example with the least number of nodes.

Solution:

No.

Consider the preorder $[2, 1]$ and labels $[I, L]$

The corresponding tree could be 

or 

□

(b) (5 points) You are now further told that each node either has 0 or 2 children. Is there a unique tree for a given preorder tree walk? If yes, provide an algorithm to construct the tree, argue the correctness. If no, provide a counter example with the least number of nodes.

Solution:

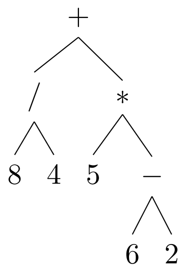
Yes. The index increments once during each call of the CONSTRUCT-TREE function which creates nodes for each recursive call that correspond to the correct node in the prefix string.

(Global var) $index = 0$

CONSTRUCT-TREE($pre, preChild$)
    $index = index + 1$
    $node =$ CreateNode($pre[index]$)
    if $preChild[index] == 2$:
        $node.left =$CONSTRUCT-TREE(pre)
        $node.right =$CONSTRUCT-TREE(pre)
    return $node$

□

The following infix expression: $(8/4) + (5 * (6 - 2))$ can be represented as the following tree:



Note that the inorder traversal of this expression tree yields the original expression and thus it is called the *infix* expression. The preorder traversal of this expression tree yields: $+/84 * 5 - 62$ and this is called a *prefix* expression. Postfix and Prefix expressions are often preferred to infix expressions because it avoids ambiguity about the order of operations.

(c) (5 points) Convince yourself that there exists a unique expression tree for a *valid* prefix expression with binary operators. Describe an algorithm to construct the expression tree for a given valid prefix expression. You may use a boolean function isOperator which takes a character and returns True if it is an operator, else returns False. (**Hint**: Use Part (b))

Solution:

(Global var) $index = 0$

EXP-TREE($prefix$)
    $index = index + 1$
    $node =$ CreateNode($prefix[index]$)
    if ISOPERATOR($node.key$) ==True:
        $node.left =$EXP-TREE($prefix$)
        $node.right =$EXP-TREE($prefix$)
    return $node$

□

(d) (4 points) In this question we will construct a *recursive* algorithm Evaluate($node$) that evaluates the expression corresponding to the expression subtree rooted at node $node$. This uses the helper function IsLeaf(x) that returns True if x is a leaf node. It also uses the helper function Solve(A,B,op) that takes two integer values and a character *op* and returns the output A *op* B where $op \in \{+, -, /, *\}$. For example, Solve$(2, 3, '+')$ returns 5.

Solution:

Evaluate($node$)
    if ISLEAF($node$) ==True:
        return $node.key$
    else :
        SOLVE(EVALUATE($node.left$),EVALUATE($node.right$),EVALUATE($node$))

□