10910EECS204001  
Data Structure HW3-Binary Tree

NTHU OJ 12975

Description:

In this homework, we need to construct a tree from the s-expression and implement the following functions.

**Construct\_tree**

**Traversal**

**Height**

**WeightSum**

**MaximumPathSum**

**BinaryTower**

**DeleteLeaf**

**Foldable**

Implement Functions:

1. Construct a binary tree from the given s-expression string.

S-expression is a simple expression that represents a tree structure in linear text form with matched pair of parentheses

An empty binary tree can be expressed as ()

A general binary tree can be expressed as (root(left subtree)(right subtree))

* Eg: (1(2()(3()()))(4()())) represents the tree in Fig.1

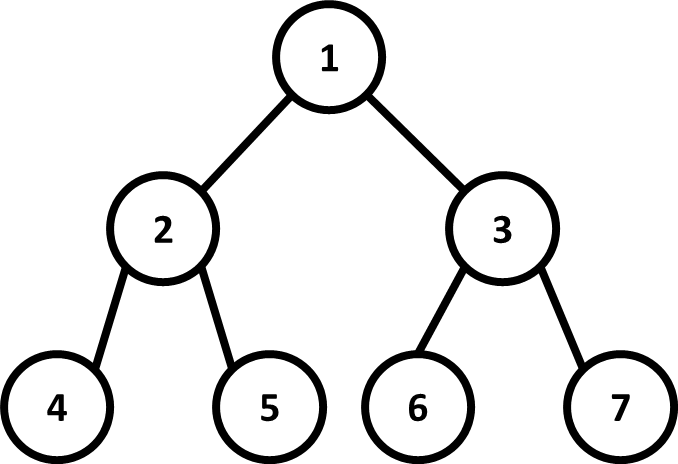


Fig 1. Example Binary Tree

Hint : Parse the given string to find the root, the left sub-tree and the right sub-tree.

Eg: (1(2(4()())(5()()))(3(6()())(7()())))

Root = 1

Left sub-tree = (2(4()())(5()()))

Right sub-tree = (3(6()())(7()()))

How to identify sub-trees?

Note that a tree must start with a ‘(‘ and end with a ‘)’

* 1. Use a stack to match ‘(‘ and ‘)’

1. Push a ‘(‘ in the beginning, and pop when seeing a ’)’
2. When the stack is empty, a sub-tree is built
3. Use a variable to accomplish the task
4. Instruction: Traverse

You need to implement 3 kinds of *traverse*, namely pre-order, in-order and post-order traversal. You should output the weight of each node in those traversal order that separated by a space. First output the pre-order traversal string. Next, output the in-order string. Last, output the post-order traversal string.

\*\*There is a space and a new line after each line of the string\*\*

Eg:

Input : (1(2()(3()()))(4()()))

Output :

1 2 4 5 3 6 7

4 2 5 1 6 3 7

4 5 2 6 7 3 1

1. Instruction: Height

You need to compute the height of the binary tree.

Input: (1(2()(3()()))(4()()))

Output: 3

1. Instruction: WeightSum

You need to calculate the sum of all nodes’ weights in the binary tree

Eg:

Input: (1(2()(3()()))(4()()))

Output: 28

1. Instruction : MaximumPathSum

You need to calculate the sum of the nodes’ weights of each path starting from the root to a leaf and output the maximum among them.

Eg:

Input: (1(2()(3()()))(4()()))

Output: 11

1. Instuction: BinaryTower

Imagine we can install towers on the nodes of the tree. Each tower can defend its parent, itself and its immediate children, as shown in Fig.2.

Find the minimum number of towers we need to defend all the nodes in the binary tree.

Eg:

Input: (1(2()(3()()))(4()()))

Output: 2

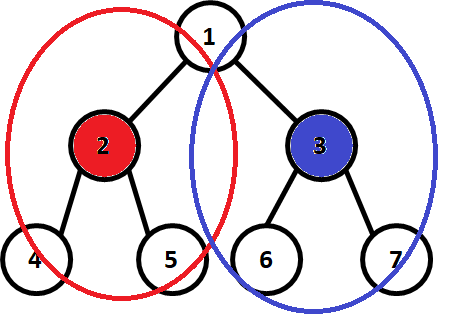


Fig.2 Binary Tower Example

1. Instruction: Foldable

Determine whether the binary tree is foldable or not.

A binary tree is **foldable** iff the left and right sub-trees of the binary tree are structure wise mirror image of each other. (without considering the node’s weight)

If the binary tree is foldable, print “Yes”; Otherwise, print “No”. (without double quotation mark)

p.s. an empty binary tree is foldable

Eg:

Input : (1(2()(3()()))(4()()))

Output : Yes

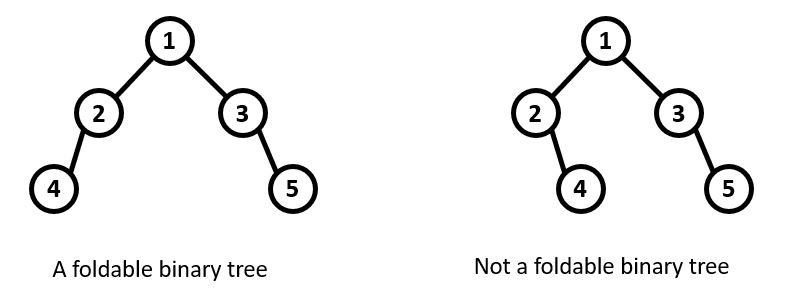


Fig.2 Foldable/Non-Foldable tree Example

1. Instruction: DeleteLeaf

Delete all the leaves in the binary tree.

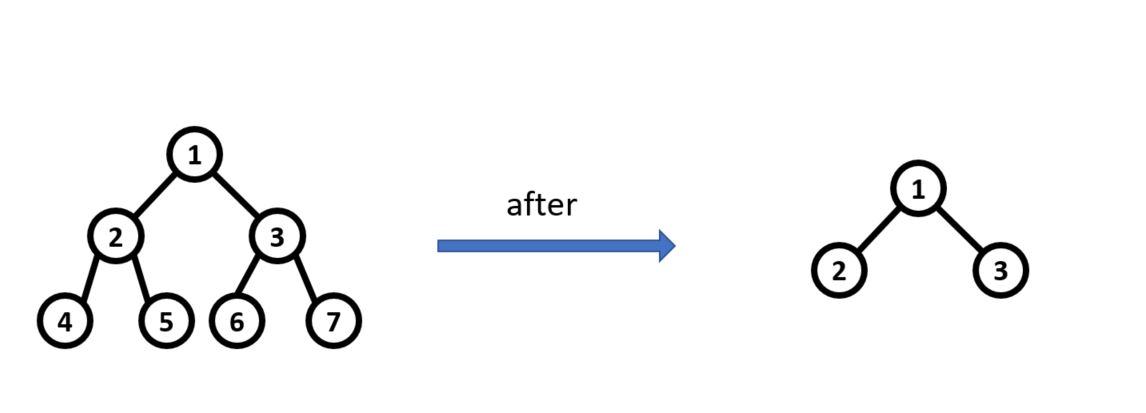


Fig.3 DeleteLeaf Example

Input:

* + Input is composed by several sets of questions.
  + A set of questions start by an s-expression string, followed by several instructions that end with the instruction “End”.
  + First, you’ll see an s-expression string. Then, you’ll see several instructions. Finally, you’ll see an “End” instruction, and after that you’ll see EOF or next s-expression string

Note:

1. The length of each s-expression is at most 10000000
2. The number of nodes in each tree is at most 1200000
3. Each nodes’ weight is between -100000 and 100000

Output:

* + Traverse: print preorder, inorder, postorder traversal of the binary tree
  + Height: print the height of the binary tree
  + WeightSum: print the sum of all the node’s weight in the binary tree
  + MaximumPathSum: print the maximum sum among all root to leaf paths of the binary tree
  + BinaryTower: print the number of towers needed to defend the hole binary tree
  + DeleteLeaf: delete all the leaves in the binary tree
  + Foldable: Print “Yes” or “No” Depending on the binary tree foldable or not. \*\*without double quotes\*\*
  + End: End of instructions of this question set.

\*\*\*The output of every Instructions except DeleteLeaf and End should followed by a new line \*\*\*

Notice:

You can use **String** in this homework. However, other kinds of STL are forbidden.

Your code needs to be submitted to NTHU OJ and iLMS before the deadline.

Please zip your code as studentID.zip and submit to iLMS HW3.(eg, 108062568.zip)