10810EECS204001

Data Structures Homework 3

Due date: 2019/11/11 23:59

Submit to OJ: #12449

Upload code to iLMS

Submission

- Please **1)** submit your code to OJ (OJ: #12449),
  and **2)** upload the zipped file (source codes) to iLMs.
  Both should be done before the due date.
- Scores will be given based on your OJ results, and the uploaded file (the source codes) should be identical to those submitted to OJ. TAs will examine your uploaded codes.

Description

In this homework, you are asked to implement the following functions.

1. Construct a binary tree
2. Tree traversal
3. Evaluate the height of a binary tree
4. Determine whether a binary tree is complete or not
5. Determine whether a binary tree is foldable or not
6. QQ addition
7. Delete all the leaves in a binary tree

Construct a binary tree

You will be given a string (called *s-expression*), which represents a binary tree. An empty binary tree can be expressed as (), and a general binary tree can be expressed as *(root (left subtree)(right subtree))*.

For example, (1(2(4()())(5()()))(3(6()())(7()()))) represents the following tree.

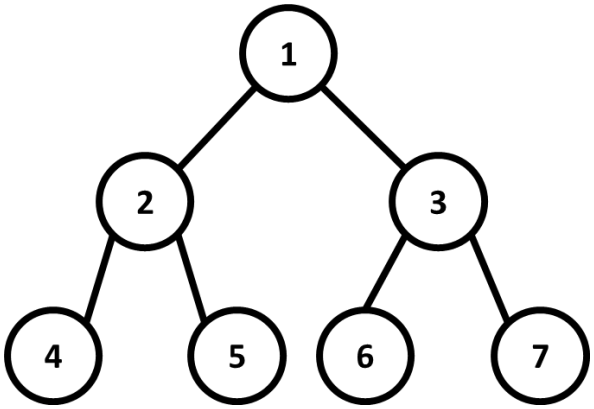

Fig. 1

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

For example, in Fig. 1

root = 1, left sub-tree = (2(4()())(5()())), right sub-tree = (3(6()())(7()()))

How to identify sub-trees?

A tree must start with a '(' and end with a ')'

method1: You can 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

method2: You can simply use a ***variable*** to do this

Tree traversal

You need to implement three kinds of traversal operations, i.e., i) *pre-order*, ii) *in-order*, and iii) *post-order.* You need to output the weights of the tree nodes in the binary tree, where each node's weight is separated by a space. For example, when you perform an *in-order* traversal on the binary tree above, the output will be: *4 2 5 1 6 3 7* . Note that every value is followed by a space, including the last one.

Evaluate the height of a binary tree

Output the height of the binary tree. For example, the height of the binary tree in Fig. 1 is 3.

Determine whether a binary tree is complete or not

Output "Complete" or "Not complete" (without double quotes). For example, in Fig. 1, the output will be:

Complete

p.s. an empty binary tree is complete.

Complete binary tree: a binary tree of depth k with n node is called **complete** iff its nodes correspond to the nodes numbered from 1 to n in the full binary tree

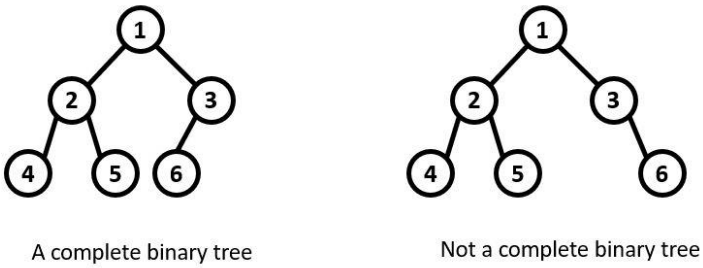

Fig. 2

Determine whether a binary tree is foldable or not

Output “Foldable” or “Not foldable” (without double quotes). For example, in Fig. 1, the ouput will be:

Foldable

p.s. an empty binary tree is foldable.

Foldable: A binary tree is **foldable** iff left and right sub-trees of the binary tree are structure wise mirror image of each other

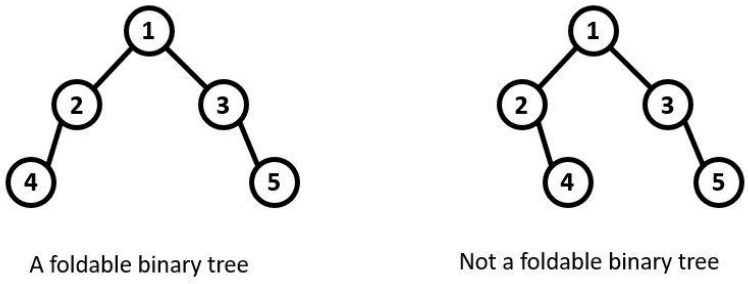

Fig. 3

QQ addition

Find a set of nodes that satisfy the following conditions

(1) Among the set of nodes, no two nodes are connected by an edge

(2) The sum of the node weights in the set is maximum

Output the sum of the selected set of nodes. For example, in Fig. 1, the output will be: 23 (1+4+5+6+7)

Please note that the sets {1,4,5} and {2,3} both satisfy condition 1, but their sums of the node weights (10 and 5, respectively) are both smaller than 23.

p.s. if the binary tree is an empty binary tree, output 0.

Delete all the leaves in a binary tree

The following figure presents an example.

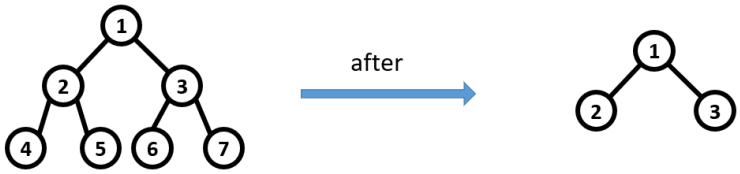

Fig. 4

You are not allow to use STL .
Implement your own stack to parse s-expression.

Input

Each test case contains multiple inputs. Each input is a string of s-expression. Please 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 node's weight is between -100000 and 100000.

Output

For each input s-expression, output the following results separated by a newline symbol.

- Pre-order traversal[#]
- In-order traversal[#]
- Post-order traversal[#]
- The height of the binary tree
- Whether the binary tree is complete or not
- Whether the binary tree is complete or not
- QQ addition
- Pre-order traversal after deleting all the leaves[#]
- In-order traversal after deleting all the leaves[#]
- Post-order traversal after deleting all the leaves[#]

[#]Note: if the tree is empty, print nothing but a new line character for the pre/in/post -order traversal results.

Sample input

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

Sample output

0
Complete
Foldable
0

1 2 4 5 3 6 7
4 2 5 1 6 3 7
4 5 2 6 7 3 1
3
Complete
Foldable
23
1 2 3
2 1 3
2 3 1