10710EECS204001  
Data Structures Homework 3

Due date: 2018/11/22 23:59

Submit to OJ: #12052

Upload code to iLMS

Submission

* Please **1)** submit your code to OJ (OJ: #12052),   
  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 zipped file (the source codes) should be identical to those submitted to OJ. TAs will examine your uploaded codes.

Description

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

**Construct\_tree**

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

For example, (1(2()())(3()())) represents the following tree.

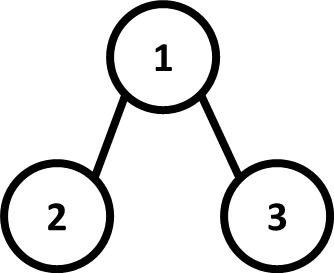


Fig. 1

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

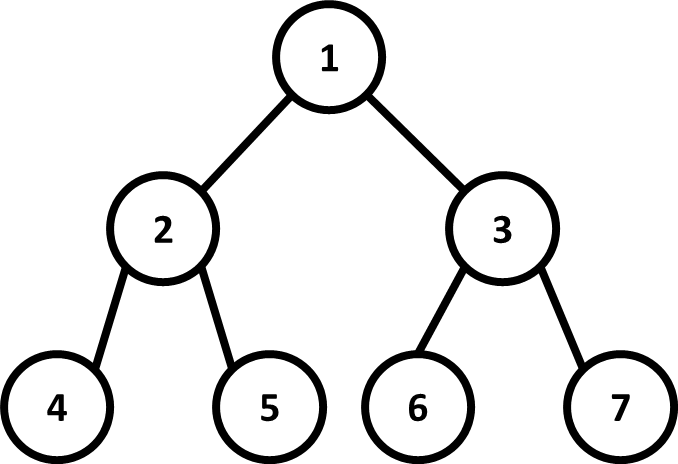


Fig. 2

In this homework, tree nodes are in linked-list representation, where every node has its own value and pointers to its left child and right child.

**Tree traversal**

You need to implement four kinds of traversal method, i.e., i) *pre-order*, ii) *in-order*, iii) *post-order*, and iv) *level-order*.

You need to output the weights of tree nodes in the binary tree, where each node weight is separated by a space. For example, when you perform a level-order traversal on the tree above, the output will be:

1 2 3 4 5 6 7

Note that every value is followed by a space, including the last one.

Also, in level-order traversal, print nodes from **left to right in the same level**.

**Tree height**

Output the height of the tree. For example, the tree height of the tree in Fig. 2 is 3.

**Weight sum**

Output the summation of the weight of every node in the tree. For example, in Fig. 2, the weight sum is 28.

**Maximum Path Sum**

Find a path from root to leaf, such that the summation of weights of the nodes in the path is maximum. For example, in Fig. 2, the maximum path sum is 11 (path: 1-3-7).

**Invert**

Invert the binary tree. That is, swap the left child and right child for every node. The following figure presents an example.

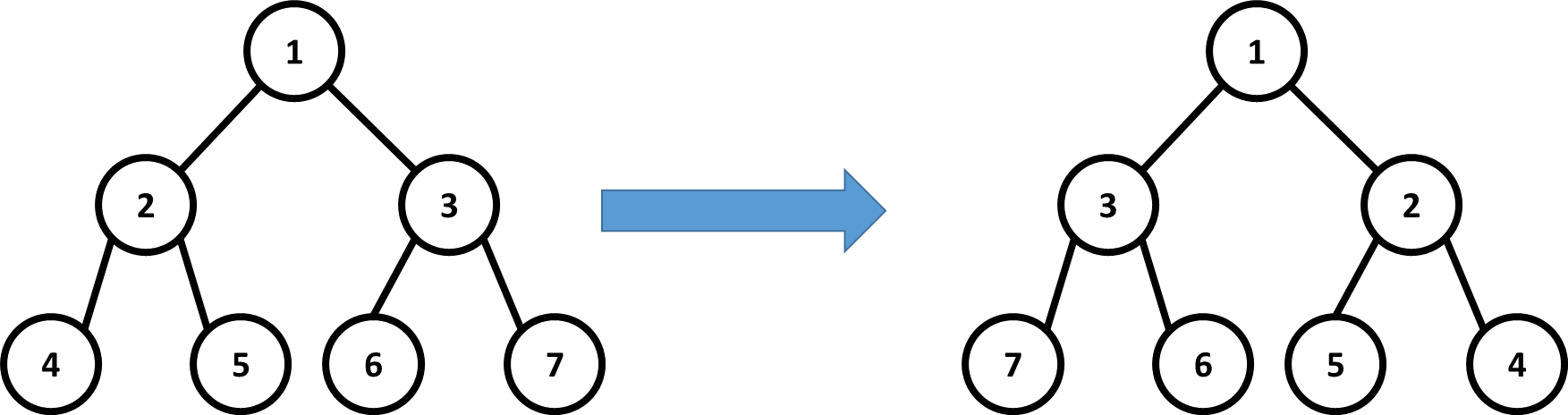


Fig. 3.

Input

Input will be a string, which is an s-expression of a binary tree

-100 <= Node weight <= 100

Output

After constructing the tree, you have to print the following information of this tree.

* In-order traversal
* Pre-order traversal
* Post-order traversal
* Level-order traversal
* Height of tree
* Weight sum of tree
* Maximum path sum of tree
* In-order traversal after inverting
* Pre-order traversal after inverting
* Post-order traversal after inverting
* Level-order traversal after inverting

The good news is, we have already implemented the main function, which contains the correct output order.

Sample input

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

Sample output

4 2 5 1 6 3 7

1 2 4 5 3 6 7

4 5 2 6 7 3 1

1 2 3 4 5 6 7

3

28

11

7 3 6 1 5 2 4

1 3 7 6 2 5 4

7 6 3 5 4 2 1

1 3 2 7 6 5 4