EE 2331 Data Structures and Algorithms, Semester B, 2009/10

Tutorial 11: Reconstruction of Binary Trees

Week 11 (1st April, 2010)

The tasks of tutorial exercises are divided into three levels. Level 1 is the basic tasks. You should have enough knowledge to complete them after attending the lecture. Level 2 is the advanced tasks. You should able to tackle them after revision. Level 3 is the challenge tasks which may be out of the syllabus and is optional to answer. I expect you to complete task A in the tutorial.

Outcomes of this tutorial

1. Able to reconstruct the binary tree from the given inorder and postorder traversal sequences using recursion

An inorder traversal sequence and a postorder traversal sequence unambiguously define a binary tree. For example, the following traversals define the tree shown in Fig. 1.

Inorder Traversal:	**H B J A F D G C E**
Postorder Travesal:	**H J B F G D E C A**

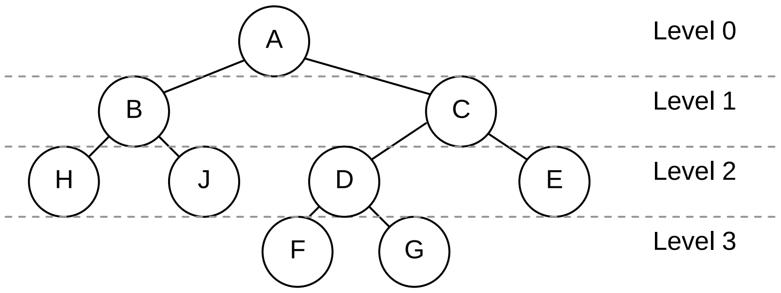

Figure 1. An example of the reconstructed tree.

Please refer to p98 to p106 of lecture notes 09 for the detail description of the recursive algorithm.

In this tutorial, the program will read in the inorder and postorder traversal sequences and build the corresponding binary tree.

The traversal sequences are stored in a text file with the following format:

Row	**Content**	**Remark**
1st	$C_1\ C_2\ C_3 \ldots C_n$	The inorder traversal sequence
2nd	$C_1\ C_2\ C_3 \ldots C_n$	The postorder traversal sequence

The structure of tree is defined as follows:

```
typedef char TreeElement;              //define the tree element (char)
typedef struct _treenode {             //define the tree node structure
        TreeElement data;
        struct _treenode *left, *right;
} TreeNode;

typedef struct _tree {                 //define the tree structure
        struct _treenode *root;
} Tree;
```

Task A (Level 2): Reconstruction of binary tree

Design the recursive function **TreeNode* reconstructTreeUsingInPost(TreeElement inorder[], int in_lo, int in_hi, TreeElement postorder[], int post_lo, int post_hi)** to reconstruct the linked list represented tree from the given inorder and postorder traversal sequence.

The function accepts an array of inorder sequence (*inorder*[]), an array of postorder sequence (*postorder*[]), the starting (*in_lo*) and ending indices (*in_hi*) of the inorder array, the starting (*post_lo*) and ending indices (*post_hi*) of the postorder array. It should return the root pointer of the tree.

Expected Output:

```
Enter the file name for testing: test1.txt

The inorder sequence is: HBJAFDGCE
The postorder sequence is: HJBFGDECA
The linked list tree is:
    E
  C
      G
    D
      F
A
    J
  B
    H
The inorder sequence is: HBJAFDGCE
```

Discussion: Can you reconstruct the tree by considering the preorder and postorder sequences? Why or why not? Illustrate with an example.