1

Trees in C++

CSC 340

April 6, 2016

2

Overview

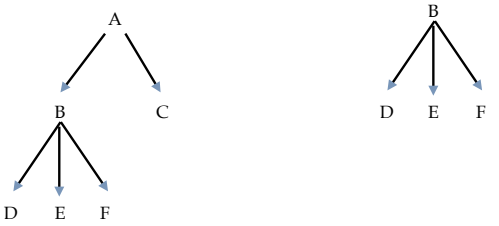
- Terminology
- Binary Trees
  - Array
  - Nodes/Pointers
- Traversals

3

Terminology

- Trees are composed of ***nodes*** and ***edges***
- Trees are hierarchical
  - Parent-child relationship between nodes
  - Ancestor-descendant relationship between nodes
- Subtree of a tree is any node and its descendants
- A general tree **T** is a set of one or more nodes such that **T** is partitioned into disjoin subsets:
  - A single node, **r**, the root
  - Sets that are general trees, called subtrees of **r**

Tree Examples



4

4

On the left, a general tree

On the right, a subtree of the tree on the left

Terminology

- Parent of node n
  the node directly above node n in the tree
- Child of node n
  any node directly below node n in the tree
- Root
  the only node in the tree with no parent
- Subtree of node n
  A tree that consists of a child (if any) of node n and that child's descendants

5

5

Terminology

- Leaf
  A node with no children
- Siblings
  Nodes with a common parent
- Ancestor of node n
  A node on the path from the root to n
- Descendant of node n
  A node on the path from n to a leaf

6

6

Overview

- Terminology
- Binary Trees
  - Array
  - Nodes/Pointers
- Traversals

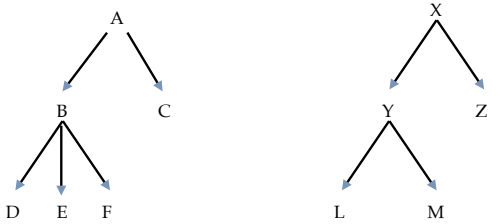
7

Binary Trees

- A binary tree is a set T of nodes such that either
  - T is empty, or
  - T is partitioned into three disjoint sets:
    - A single node r, the root node
    - Two possibly empty sets that are binary trees, called the left subtree of r and the right subtree of r

8

Tree Examples

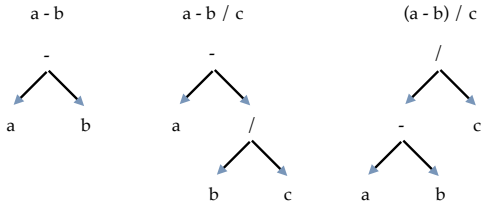


9

On the left, a general tree - not a binary tree - why?
On the right, a binary tree

Algebraic Expressions

- Binary trees can represent algebraic expressions



10

10

Note where the leaves and subtrees are - indicates precedence. More on this in CSC 600

Binary Search Tree

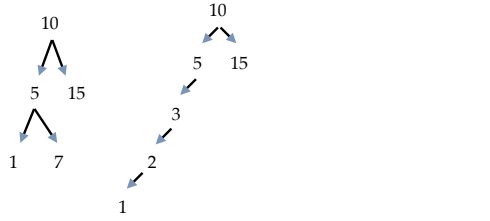
- A binary tree that has the following properties:
  - n's value is greater than all values in n's left subtree (TL)
  - n's value is less than all values in n's right subtree (TR)
  - Both TL and TR are binary search trees

11

11

Binary Search Tree Height

- Height
  The height of a tree is the number of nodes along the longest path from the root to a leaf



12

12

Left: height 3

Right: height 5

Binary Search Tree Height

13

- Level of node n in a tree T
  - If n is the root of T, it is at level 1
  - If n is not the root of T, its level is 1 greater than the level of its parent
- Height of a tree T defined in terms of the levels of its nodes
  - If T is empty, its height is 0
  - If T is not empty, its height is equal to the maximum level of all of its nodes

Binary Search Tree Height

14

- Recursive definition of height:
  - If T is empty, height is 0
  - If T is not empty,
    height(T) = 1 + max(height(TL), height(TR))

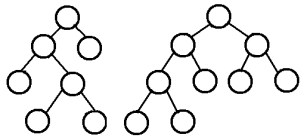
Full Binary Tree

15

- A binary tree of height h is full if nodes at levels less than h have two children each
- A full binary tree (sometimes proper binary tree or 2-tree) is a tree in which every node other than the leaves has two children.
- Recursive definition:
  - If T is empty, T is a full binary tree of height 0
  - If T is not empty and has height h > 0, T is a full binary tree if its root's subtrees are both full binary trees of height h - 1

Complete Binary Tree

- A binary tree of height h is complete if
  - It is full to level h - 1, and
  - Level h is filled from left to right



full tree complete tree

16

A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.

Complete Binary Tree

- Another definition:
- A binary tree of height h is complete if
  - All nodes at levels <= h - 2 have two children each and
  - When a node at level h - 1 has children, all nodes to its left at the same level have two children each, and
  - When a node at level h - 1 has one child, it is a left child

17

Balanced Binary Tree

- A binary tree is balanced if the heights of any node's subtrees differ by no more than 1
- Complete binary trees are balanced
- Full binary trees are complete and balanced

18

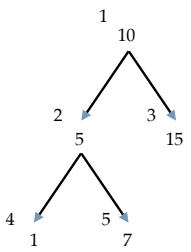
Possible Representations

- Array based
  - Uses an array of tree nodes
  - Requires the creation of a free list that keeps track of available nodes
- Pointer based
  - Nodes have two pointers that link the nodes in the tree

19

19

Array Representation



0	10
1	5
2	15
3	1
4	7
5	
6	

20

20

Level by level numbering of complete binary tree
Root at index 0
Left node at 2 * (index + 1)
Right node at 2 * (index + 1) + 1

Node Representation

- Let's design this:
  - What are the objects we need?
  - What fields?

21

21

Tree, Node
Tree: Node * root
Node: T value, Node * left, Node * right

Overview

- Terminology
- Binary Trees
  - Array
  - Nodes/Pointers
- Traversals

22

22

Traversals of a Binary Tree

- A traversal visits every node in the tree
- You do something with or to a node during a traversal (for example, display the value)
- General form of a recursive traversal algorithm:

```
traverse( binary_tree )
   if binary_tree is not empty
      traverse( left_subtree of binary_tree's root )
      traverse( right_subtree of binary_tree's root )
```

23

23

Note that we're missing the root in this general form

Traversals of a Binary Tree

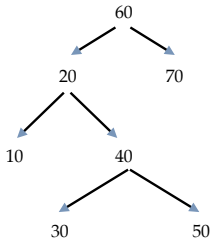
- Preorder traversal
  Visit root ***BEFORE*** visiting its subtrees (i.e. before the recursive calls)
- Inorder traversal
  Visit root ***BETWEEN*** visiting its subtrees (i.e. between the recursive calls)
- Postorder traversal
  Visit root ***AFTER*** visiting its subtrees (i.e. after the recursive calls)

24

24

Traversals of a Binary Tree

- What is the result of displaying each node with each traversal?



25

Preorder: 60 20 10 40 30 50 70
Inorder: 10 20 30 40 50 60 70
Postorder: 10 30 50 40 20 70 60

Write code