CS32 Spring 2021

Week 8

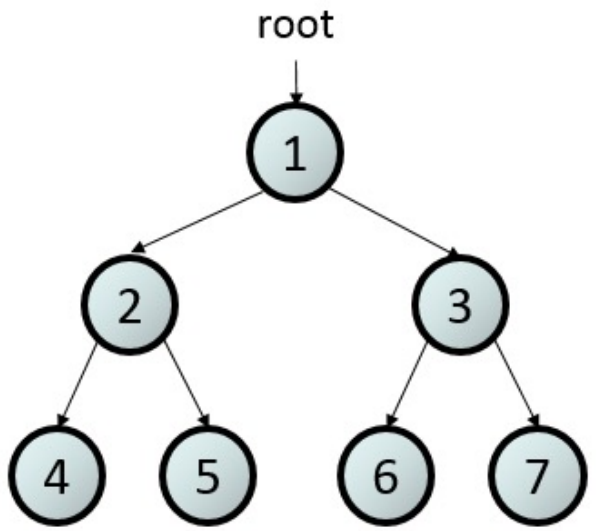
TA: Manoj Reddy

LA: Katherine Zhang

Outline

- Trees

Trees



- Very useful in organizing information
- TreeNode can have more than 2 children (array of TreeNode pointers)
- Binary tree has at most 2 child nodes
  - Left child
  - Right child
- Root is represented using TreeNode pointer (TreeNode*)
- Operations: search, insert, delete etc.
- TreeNodes are randomly ordered in memory

```
struct TreeNode{
  int val;
  TreeNode *left;
  TreeNode *right;
};
```

Tree Traversal

- Order of node traversal
  - Preorder
  - Inorder
  - Post-order
- Draw a line around nodes in counter-clockwise direction
- Level-order
  - Implemented using queue
  - Draw a horizontal line from left to right

```
void PreOrder(Node *cur)
{
   if (cur == NULL)                // if empty, return…
       return;

   cout << cur->value;        // Process the current node

   PreOrder(cur->left);      // Process left sub-tree
   PreOrder(cur-> right);  // Process right sub-tree
}
```

Problem

- Find the maximum depth of a tree
  - *int maxDepth(TreeNode* root){…};*
- Return true if 2 trees are exactly the same
  - *bool sameTrees(TreeNode* root1, TreeNode* root2){…};*

Binary Search Tree

- Special binary tree with following properties:
  - For every node X in the tree:
    - All nodes in X's left subtree must be less than X
    - All nodes in X's right subtree must be greater than X
- Operations (*If balanced*)
  - Search: O(log n)
  - Insert: O(log n)
  - Delete: O(log n) //3 cases involved, see slides
- If tree is unbalanced, above operations: O(n)
- 2-3 Trees, Red-Black Trees, AVL Trees
  - Improved versions of binary search tree that ensures trees are balanced!

Problem

- Get the max value of a BST
  - *int maxBST(TreeNode* root){};*
- Check if a binary tree is a BST
  - *bool isBST(TreeNode* root){ };*