GeekBand 极客班

互联网人才 + 油站!

二叉树和分治算法

www.geekband.com

GeekBand 极客班 互联网人才+油站！

极客班携手网易云课堂，针对热门IT互联网岗位，联合业内专家大牛，紧贴企业实际需求，量身打造精品实战课程。

专业课程 + **项目碾压**

• 顶尖专家技能私授	• 学员混搭线上组队
• 贴合企业实际需求	• 一线项目实战操练
• 互动交流直播答疑	• 业内大牛辅导点评

www.geekband.com

二叉树和分治算法

大纲

1. 二叉树介绍
2. 先序/中序/后序 Preorder / inorder / postorder
3.分治算法 Divide & Conquer
4.二叉树的宽度优先遍历
5.二叉搜索树

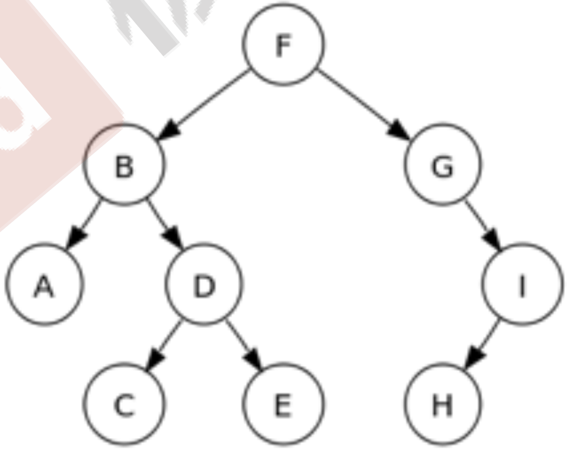
翻转二叉树

Homebrew 作者Mark Howell面试被Google 拒，因为不会翻转二叉树?

```
    1                      1
   / \                    / \
  2   3          =>      3   2
     /                    \
    4                      4
```

二叉树

二叉树，是指对于树中的每个节点而言，至多有左右两个子节点，即任意节点的度小于等于2



概念

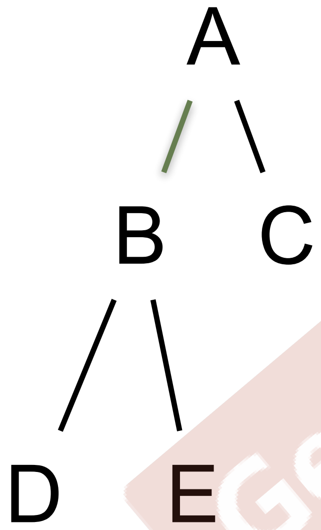
高度：从根节点到某个节点的路径长度称为该节点的层数(level)，根节点为第0层，非根节点的层数是其父节点的层数加1。树的高度定义为该树中层数最大的叶节点的层数加1，即相当于于从根节点到叶节点的最长路径加1。

满二叉树(full binary tree)：如果一棵二叉树的任何结点，或者是叶节点，或者左右子树都存在，则这棵二叉树称作满二叉树。

完全二叉树(complete binary tree)：如果一棵二叉树最多只有最下面的两层节点度数可以小于2，并且最下面一层的节点都集中在该层最左边的连续位置上，则此二叉树称作完全二叉树。

Binary Tree DFS Traversal

Binary Tree Traversal



Preorder: **A** $\underline{\text{BDE}}$ $\underline{\text{C}}$

Postorder: $\underline{\text{DEB}}$ $\underline{\text{C}}$ **A**

inorder: $\underline{\text{DBE}}$ **A** $\underline{\text{C}}$

DFS 代码

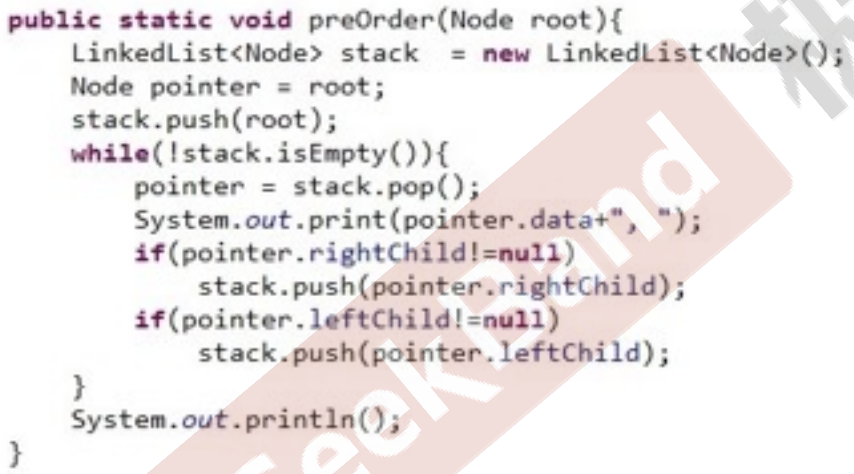
```
void preOrderTraversal(TreeNode *root) {
    if (!root) {
        return;
    }
    visit(root);
    preOrderTraversal(root->left);
    preOrderTraversal(root->right);
}
```

```
void inOrderTraversal(TreeNode *root) {
    if (!root) {
        return;
    }
    inOrderTraversal(root->right);
    visit(root);
    inOrderTraversal(root->left);
}
```

```
void postOrderTraversal(TreeNode *root) {
    if (!root) {
        return;
    }
    postOrderTraversal(root->left);
    postOrderTraversal(root->right);
    visit(root);
}
```

Traverse Iteration

Stack: Preorder

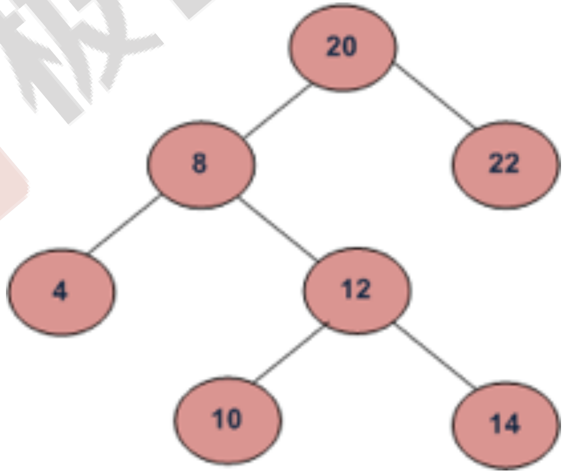


```
public static void preOrder(Node root){
    LinkedList<Node> stack  = new LinkedList<Node>();
    Node pointer = root;
    stack.push(root);
    while(!stack.isEmpty()){
        pointer = stack.pop();
        System.out.print(pointer.data+", ");
        if(pointer.rightChild!=null)
            stack.push(pointer.rightChild);
        if(pointer.leftChild!=null)
            stack.push(pointer.leftChild);
    }
    System.out.println();
}
```

http://www.cnblogs.com/dolphin0520/archive/2011/08/25/2153720.html

Find Next in Binary Tree

In-order traverse a binary tree with parent links, find the next node to visit given a specific node.



Followup: without parent link?

Divide & Conquer Algorithm

分解（Divide）：将原问题分解为若干子问题，这些子问题都是原问题规模较小的实例。
解决（Conquer）：递归地求解各子问题。如果子问题规模足够小，则直接求解。
合并（Combine）：将所有子问题的解合并为原问题的解。

- Merge Sort
- Quick Sort
- Most of the Binary Tree Problems !

Tree Depth

Compute the depth of a binary tree.

```
int treeDepth(TreeNode *node) {
    if (node == NULL)
        return 0;
    else
        return max(treeDepth(node->left),
treeDepth(node->right)) + 1;
}
```

Subtree

Tree1 and Tree2 are both binary trees nodes having value, determine if Tree2 is a subtree of Tree1.

Rebuild Binary Tree

Pre-order + In-order

Preorder sequence: **A** B D E C F
Inorder sequence: D B E **A** F C

```
        A
      /   \
    B       C
   / \     /
  D   E  F
```

Balanced Binary Tree

Determine if a binary tree is a balanced tree.

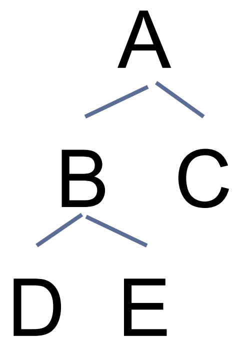
一颗二叉树是平衡的，当且仅当左右两个子树的高度差的绝对值不超过1，并且左右两个子树都是一棵平衡二叉树。

Path Sum

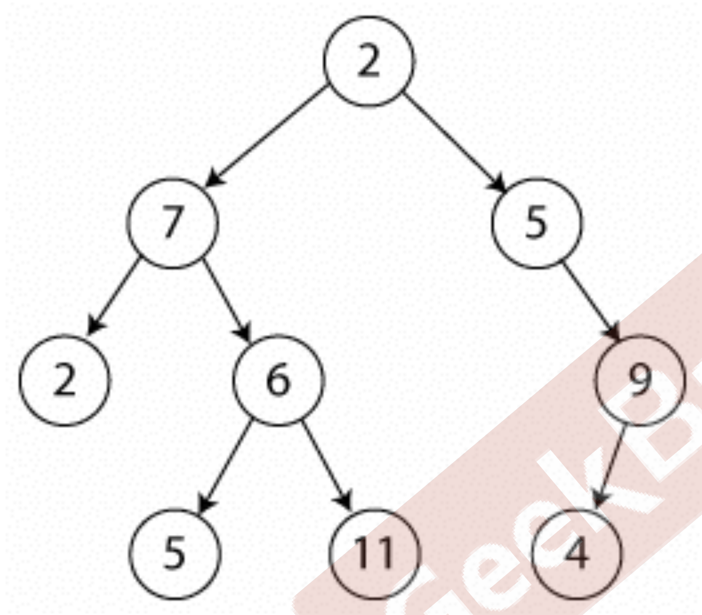
Get all the paths (always starts from the root) in a binary tree, whose sum would be equal to given value.

Lowest Common Ancestor(LCA)

Given a binary tree and two nodes. Find the lowest common ancestor of the two nodes in the tree.
For example, the LCA of D & E is B.



Shortest Path in Two Nodes



http://www.fusu.us/2013/06/p6-shortest-path-in-binary-tree.html

Binary Tree DFS Traversal

Template: https://github.com/dongfeiwww/boolan/blob/master/class4/dfs.java

Binary Tree BFS Traversal

Template: https://github.com/dongfeiwww/boolan/blob/master/class4/bfs.c

Binary Tree to Linked Lists

Covert a binary tree to linked lists. Each linked list is correspondent to all the nodes at the same level.

Binary Tree Level Order Traversal

- 2 Queues
- 1 Queue + Dummy Node
- 1 Queue (best)

Binary Tree Level Order Traversal II

Given a binary tree, return the bottom-up level order traversal of its nodes' values. (ie, from left to right, level by level from leaf to root).

For example: Given binary tree {3,9,20,#,#,15,7},

```
    3
   / \
  9  20
    /  \
   15   7
```

return its bottom-up level order traversal as:

```
[
  [15,7],
  [9,20],
  [3]]
```

Binary Tree Zigzag Level Order Traversal

Given a binary tree, return the zigzag level order traversal of its nodes' values. (ie, from left to right, then right to left for the next level and alternate between).

For example: Given binary tree {3,9,20,#,#,15,7},

```
    3
   / \
  9  20
    /  \
   15   7
```

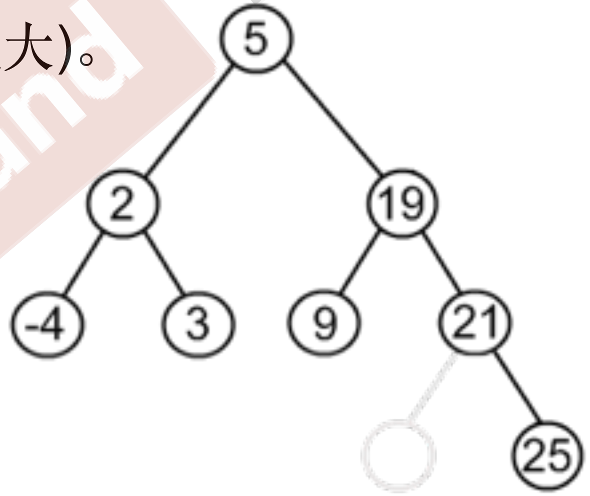
return its zigzag level order traversal as:

```
[
  [3],
  [20,9],
  [15,7]]
```

Binary Search Tree

Binary Search Tree

二分查找树(Binary Search Tree, BST)是二叉树的一种特例，对于二分查找树的任意节点，该节点储存的数值一定比左子树的所有节点的值大比右子树的所有节点的值小(该节点储存的数值一定比左子树的所有节点的值小比右子树的所有节点的值大)。



Is Binary Search Tree

错误例子:

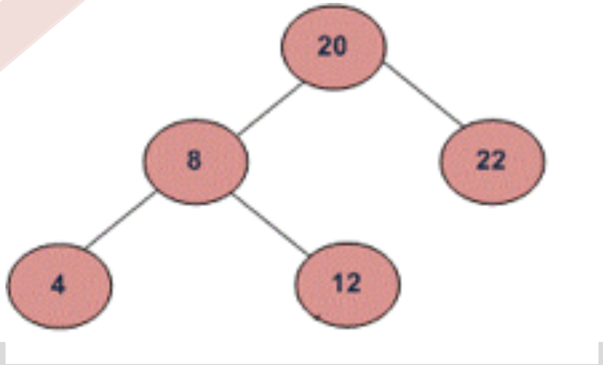
```
bool isValidBST(TreeNode* root) {
        if(root==NULL)
            return true;
        else if(root->left&&!root->right)
            return isValidBST(root->left)&&(root->val>root->left->val);
        else if(!root->left&&root->right)
            return isValidBST(root->right)&&(root->val<root->right->val);
        else if(root->left&&root->right)
            return isValidBST(root->left)&&isValidBST(root->right)
&&(root->val>root->left->val)&&(root->val<root->right->val);
        else
            return true;
    }
```

```
  10
 / \
5  15
   /  \
  6   20
```

Print Range in a Binary Search Tree

Given two values k1 and k2 (where k1 < k2) and a root pointer to a Binary Search Tree. Print all the keys of tree in range k1 to k2. i.e. print all x such that k1<=x<=k2 and x is a key of given BST. Print all the keys in increasing order.

For example, if k1 = 10 and k2 = 22, then your function should print 12, 20 and 22.



Sorted Array to Binary Search Tree

Convert a sorted array(increasing order) to a balanced BST.

Input: Array {1, 2, 3}

Output: A Balanced BST

```
     2
   /  \
  1    3
```

Input: Array {1, 2, 3, 4}

Output: A Balanced BST

```
      3
    /  \
   2    4
 /
1
```

Homework:

Given a binary tree, check whether it is a mirror of itself (ie, symmetric around its center).

For example, this binary tree is symmetric:

```
    1
   / \
  2   2
 / \ / \
3  4 4  3
```

But the following is not:

```
  1
 / \
2   2
 \   \
  3   3
```

Binary Search Tree Iterator

Design an iterator over a binary search tree with the following rules:

- Elements are visited in ascending order (i.e. an in-order traversal)
- next() and hasNext() queries run in O(1) time in average.

Example
For the following binary search tree, in-order traversal by using iterator is [1, 6, 10, 11, 12]

```
       10
     /    \
    1      11
     \       \
      6       12
```

参考： http://stackoverflow.com/questions/4581576/implementing-an-iterator-over-a-binary-search-tree