基础算法和数据结构高频题 II

扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
知乎专栏：http://zhuanlan.zhihu.com/jiuzhang
微博: http://www.weibo.com/ninechapter
官网: www.jiuzhang.com

知识点回顾

- 第三节课中提到的，遇到区间问题一般要做哪两件事？
- Sliding window 类问题通用的套路是？
- 第三节课讲到计算时复杂度时，哪一句话所代表的方法很常用？
- 在Load Balancer 问题中，我们怎样快速的删除数组中的元素？

概要

- 二分查找类问题（2题）
- BST类问题（2题）
- 二叉树类问题（3题）

二分查找类问题

Guess Number Game

http://www.lintcode.com/zh-cn/problem/guess-number-game/

http://www.jiuzhang.com/solutions/guess-number-game/

Guess Number Game

思路：

- 二分查找

二分查找思路回顾：

- 给n 个数，从小到大排序，数字不重复，再给一个待查找的数x，问x是否存在于这n个数中，存在返回下标，不存在返回-1

Guess Number Game

二分查找算法：

- 1. 找到中点mid
- 2. x和中点mid比较：
  - x<mid：砍掉mid右边
  - x>mid：砍掉mid左边
  - x=mid：退出
- 3. 重复以上两步

- Example:
- n=9 [2,5,7,9,10,13,20,32,35] x=9

Guess Number Game

◆ 小技巧总结：

- mid = l + (r - l) / 2　以保证计算中间结果不溢出
- 不知道如何下手写时，先写最简单的情况（写代码时常用思路）

Guess Number Game

- Company Tags：Google

考点：

- 电面时考查的基础算法

能力维度：

3. 基础数据结构/算法

Search for a Range

http://www.lintcode.com/zh-cn/problem/search-for-a-range/

http://www.jiuzhang.com/solutions/search-for-a-range/

Search for a Range

- 二分查找变形一：
- 给n 个数，从小到大排列，数字可重复，再给一个待查找的数x，问x是否存在于这n个数中，存在返回最小下标，不存在返回-1
- n=12 [2,5,7,9,9,9,9,10,13,20,32,35] x=9

- 二分查找变形二：
- 给n 个数，从小到大排列，数字可重复，再给一个待查找的数x，问x是否存在于这n个数中，存在返回最大下标，不存在返回-1
- n=12 [2,5,7,9,9,9,9,10,13,20,32,35] x=9

Search for a Range

思路：

- 二分查找的两种变形加起来

- Company Tags：LinkedIn

Search for a Range

考点：

- 基础算法的微小变形

能力维度：

3. 基础数据结构/算法

二分查找总结

二分查找类题目：

Closest Number in Sorted Array	Last Position of Target	Search a 2D Matrix
Maximum Number in Mountain Sequence	Find Minimum in Rotated Sorted Array	Find Peak Element
Search in a Big Sorted Array	First Bad Version	Smallest Rectangle Enclosing Black Pixels
Rectangle Enclosing Black Pixels		

BST类问题

Convert BST to Greater Tree

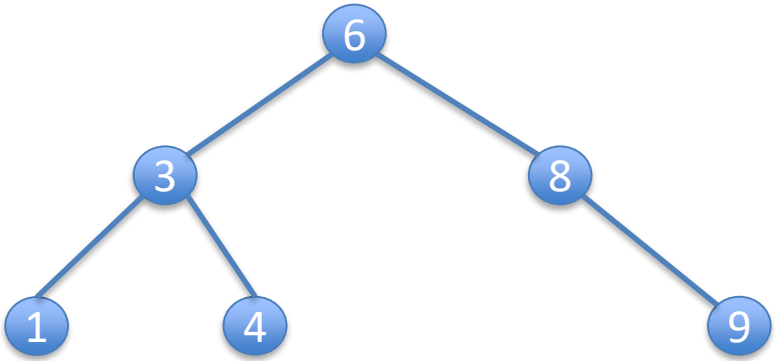
http://www.lintcode.com/zh-cn/problem/convert-bst-to-greater-tree/

http://www.jiuzhang.com/solutions/convert-bst-to-greater-tree/

Convert BST to Greater Tree

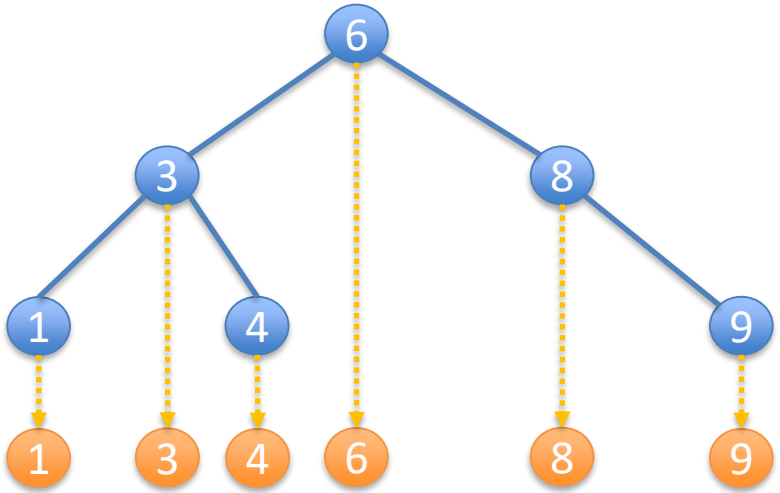
BST的性质：

- BST的定义？
  - 左边都比root小，右边都比root大



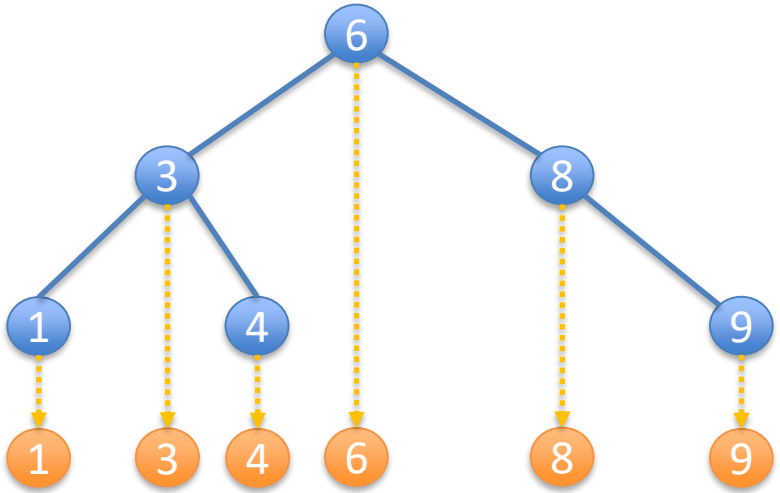
Convert BST to Greater Tree

- 把BST规范的画出来是什么样子？
  - 向下投影出的序列是从小到大的顺序



Convert BST to Greater Tree

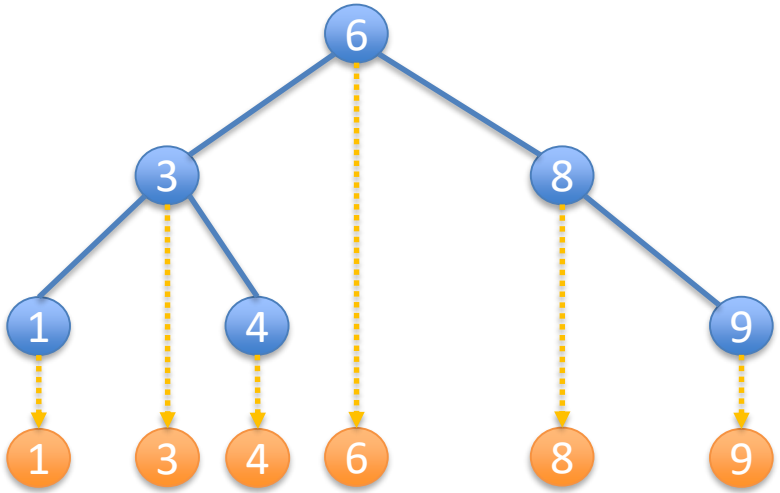
- BST的中根遍历，节点访问顺序是什么样的？
  - 小到大的顺序



Convert BST to Greater Tree

此题思路：

- 如果是右中左这样的“反中根遍历”，节点访问顺序会是什么样的？
  - 从大到小的顺序



Convert BST to Greater Tree

算法流程：

1. 定义累计求和变量sum = 0
2. 按照右中左这样的“反中根遍历”，依次访问每个节点
   ① 更新累计求和变量 sum = sum + 当前节点的值
   ② 更新当前节点的值 = sum

Convert BST to Greater Tree

代码实现：

```
12  public class Solution {
13      /**
14       * @param root the root of binary tree
15       * @return the new root
16       */
17      int sum = 0;
18
19      void dfs(TreeNode cur) {
20          if (cur == null) {
21              return;
22          }
23          dfs(cur.right);
24          sum += cur.val;
25          cur.val = sum;
26          dfs(cur.left);
27      }
28
29      public TreeNode convertBST(TreeNode root) {
30          // Write your code here
31          dfs(root);
32          return root;
33      }
34  }
```

Convert BST to Greater Tree

- 怎么理解这一段DFS（Depth First Search 深度优先搜索)？

```
19      void dfs(TreeNode cur) {
20          if (cur == null) {
21              return;
22          }
23          dfs(cur.right);
24          sum += cur.val;
25          cur.val = sum;
26          dfs(cur.left);
27      }
```

DFS的两种理解方式：

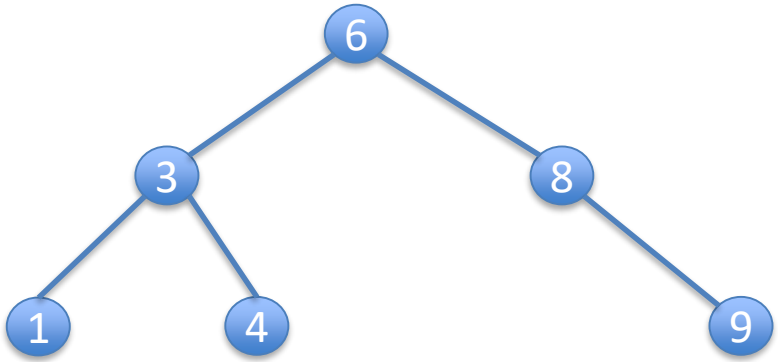
1. 按照实际执行顺序模拟
2. 按照DFS的定义宏观理解

Convert BST to Greater Tree

- 怎么理解这一段DFS (Depth First Search 深度优先搜索)?

```
19    void dfs(TreeNode cur) {
20        if (cur == null) {
21            return;
22        }
23        dfs(cur.right);
24        sum += cur.val;
25        cur.val = sum;
26        dfs(cur.left);
27    }
```

1. 按照实际执行顺序模拟



Convert BST to Greater Tree

- 怎么理解这一段DFS (Depth First Search 深度优先搜索)?

```
19    void dfs(TreeNode cur) {
20        if (cur == null) {
21            return;
22        }
23        dfs(cur.right);
24        sum += cur.val;
25        cur.val = sum;
26        dfs(cur.left);
27    }
```

2. 按照DFS的定义宏观理解：

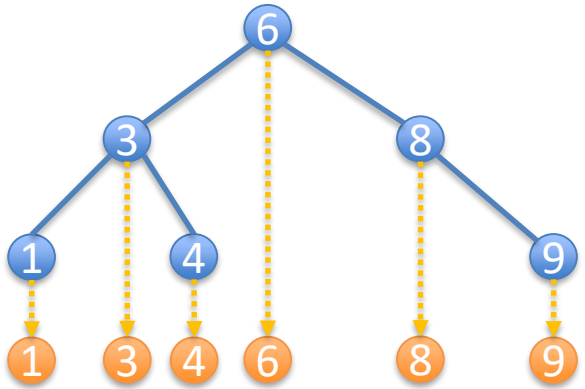
   我们这里DFS函数的定义是什么？任务是什么？

   1. “反中根遍历”的顺序访问cur这颗树每个节点
   2. sum要在当前的基础上要加上所有访问元素的值
   3. 所有访问元素要变成大于等于它的元素之和

Convert BST to Greater Tree

```
19      void dfs(TreeNode cur) {
20          if (cur == null) {
21              return;
22          }
23          dfs(cur.right);
24          sum += cur.val;
25          cur.val = sum;
26          dfs(cur.left);
27      }
28
29      public TreeNode convertBST(TreeNode root) {
30          // Write your code here
31          dfs(root);
32          return root;
33      }
34  }
```

convertBST(board)：给你root=6这棵树，帮我把这三件事干了
dfs (CEO)：好的



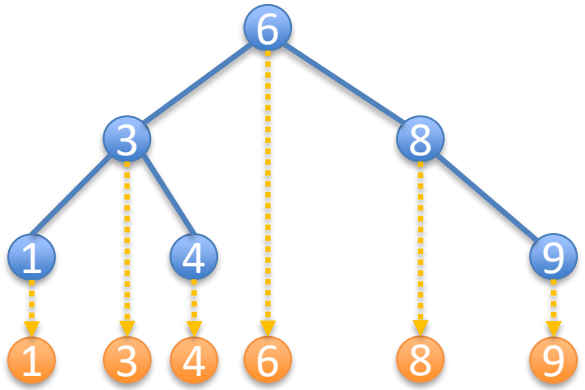
任务：

1. “反中根遍历”的顺序访问cur这颗树每个节点
2. sum要在当前的基础上要加上所有访问元素的值
3. 所有访问元素要变成大于等于它的元素之和

Convert BST to Greater Tree

```
19    void dfs(TreeNode cur) {
20        if (cur == null) {
21            return;
22        }
23        dfs(cur.right);
24        sum += cur.val;
25        cur.val = sum;
26        dfs(cur.left);
27    }
```

dfs (CEO)：我的任务是对cur=6 这棵树做三件事



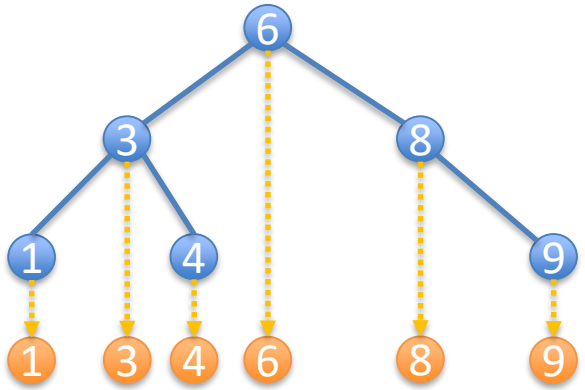
任务：

1. “反中根遍历”的顺序访问cur这颗树每个节点
2. sum要在当前的基础上要加上所有访问元素的值
3. 所有访问元素要变成大于等于它的元素之和

Convert BST to Greater Tree

```
19    void dfs(TreeNode cur) {
20        if (cur == null) {
21            return;
22        }
23        dfs(cur.right);
24        sum += cur.val;
25        cur.val = sum;
26        dfs(cur.left);
27    }
```

dfs (CEO)：我先检查下cur=6这颗树是否是空的，空的话我就完事交差了



任务：

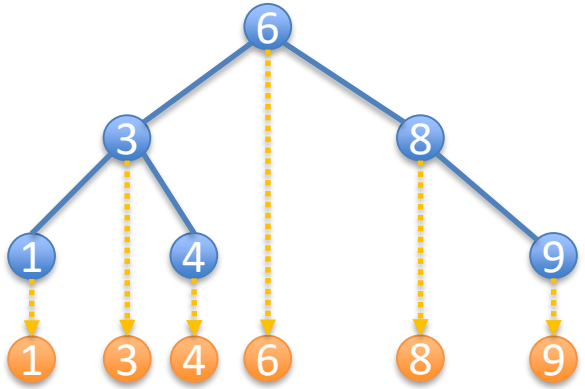
1. “反中根遍历”的顺序访问cur这颗树每个节点
2. sum要在当前的基础上要加上所有访问元素的值
3. 所有访问元素要变成大于等于它的元素之和

Convert BST to Greater Tree

```
19    void dfs(TreeNode cur) {
20        if (cur == null) {
21            return;
22        }
23        dfs(cur.right);
24        sum += cur.val;
25        cur.val = sum;
26        dfs(cur.left);
27    }
```

dfs (CEO)：既然cur=6不为空，那我们来分配下任务吧：23行的dfs，你是总监，你先帮我把8为根的这棵右子树做了，任务还是这三样，等你结果

dfs@23：👌



任务：

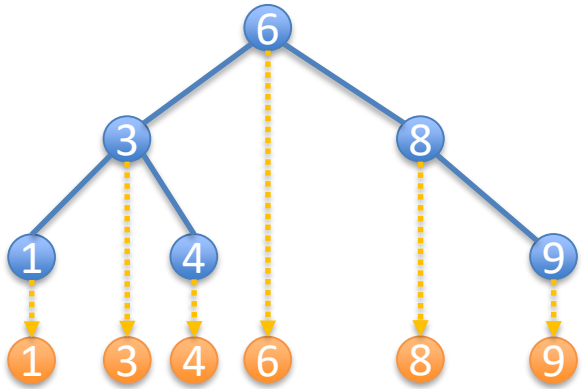
1. “反中根遍历”的顺序访问cur这颗树每个节点
2. sum要在当前的基础上要加上所有访问元素的值
3. 所有访问元素要变成大于等于它的元素之和

Convert BST to Greater Tree

```
19      void dfs(TreeNode cur) {
20          if (cur == null) {
21              return;
22          }
23          dfs(cur.right);
24          sum += cur.val;
25          cur.val = sum;
26          dfs(cur.left);
27      }
```

dfs@23：报告老大，任务已完成，sum已经加上了访问过的元素的值，所有元素的已经变成大于等于它的元素之和

dfs (CEO)：



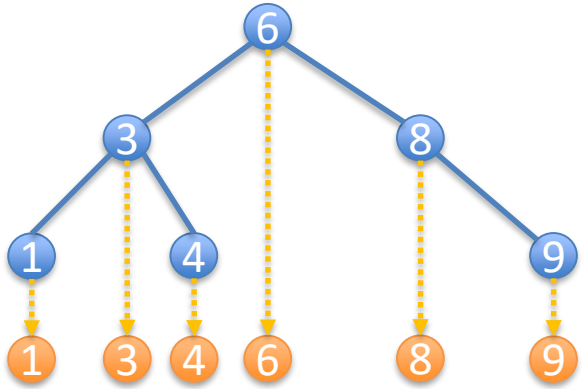
任务：

1. “反中根遍历”的顺序访问cur这颗树每个节点
2. sum要在当前的基础上要加上所有访问元素的值
3. 所有访问元素要变成大于等于它的元素之和

Convert BST to Greater Tree

```
19    void dfs(TreeNode cur) {
20        if (cur == null) {
21            return;
22        }
23        dfs(cur.right);
24        sum += cur.val;
25        cur.val = sum;
26        dfs(cur.left);
27    }
```

dfs (CEO)：@24 25这里是我自己的工作了，sum 加上当前的值，当前的值变为所有比它的大的元素之和，也就是sum



任务：

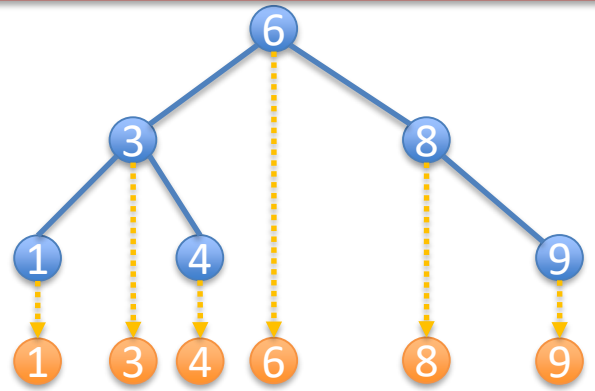
1. “反中根遍历”的顺序访问cur这颗树每个节点
2. sum要在当前的基础上要加上所有访问元素的值
3. 所有访问元素要变成大于等于它的元素之和

Convert BST to Greater Tree

```
19    void dfs(TreeNode cur) {
20        if (cur == null) {
21            return;
22        }
23        dfs(cur.right);
24        sum += cur.val;
25        cur.val = sum;
26        dfs(cur.left);
27    }
```

dfs (CEO)：好了，dfs@26，你也是总监，dfs@23和我的工作都做完了，sum现在是所有大于等于当前点的元素的和，条件已经给你创造好了，剩下的以3为根的这棵左子树就交给你了

dfs@26：放心，包在我身上



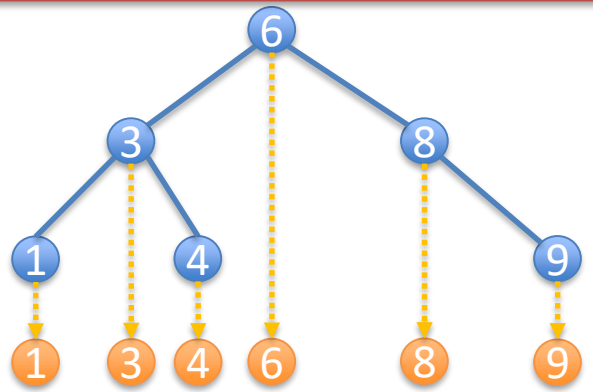
任务：

1. “反中根遍历”的顺序访问cur这颗树每个节点
2. sum要在当前的基础上要加上所有访问元素的值
3. 所有访问元素要变成大于等于它的元素之和

Convert BST to Greater Tree

```
19    void dfs(TreeNode cur) {
20        if (cur == null) {
21            return;
22        }
23        dfs(cur.right);
24        sum += cur.val;
25        cur.val = sum;
26        dfs(cur.left);
27    }
```

dfs@26：报告老大，全部做完
dfs(CEO)：不错，所有工作都做完了，可以向董事会汇报了
全体：



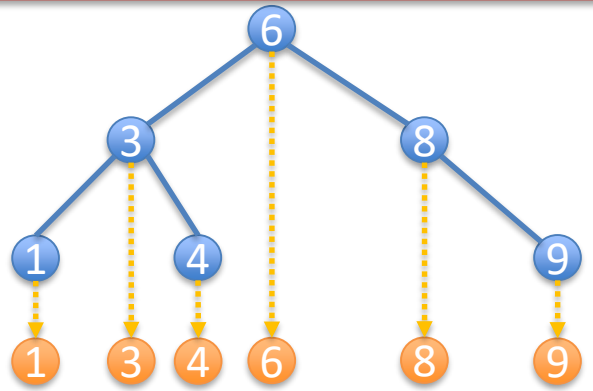
任务：

1. “反中根遍历”的顺序访问cur这颗树每个节点
2. sum要在当前的基础上要加上所有访问元素的值
3. 所有访问元素要变成大于等于它的元素之和

Convert BST to Greater Tree


```
19      void dfs(TreeNode cur) {
20          if (cur == null) {
21              return;
22          }
23          dfs(cur.right);
24          sum += cur.val;
25          cur.val = sum;
26          dfs(cur.left);
27      }
```

其实在做的过程中，总监把任务分给了经理，经理又分配给了主管，主管又分配给了各级干活的。。。他们的任务都是这三件事，直到某个干活的发现分给他的树是空的，他就啥也没干，高兴的直接交差了



任务：

1. “反中根遍历”的顺序访问cur这颗树每个节点
2. sum要在当前的基础上要加上所有访问元素的值
3. 所有访问元素要变成大于等于它的元素之和

Convert BST to Greater Tree

考点：

- 基础数据结构上的灵活操作
- BST遍历的性质

能力维度：

3. 基础数据结构/算法

Convert BST to Greater Tree

◆ 小技巧总结：

- DFS的两种理解方式：
  1. 按照实际执行顺序模拟
  2. 按照DFS的定义宏观理解

Inorder Successor in Binary Search Tree

http://www.lintcode.com/zh-cn/problem/inorder-successor-in-binary-search-tree/

http://www.jiuzhang.com/solutions/inorder-successor-in-binary-search-tree/

Inorder Successor in Binary Search Tree

名词解释：

- 节点的successor 后继就是比给定节点大的所有节点中最小的那个

Inorder Successor in Binary Search Tree

思路：

- 考虑p和root之间的关系（p 为给定的节点，要找到p节点的后继）

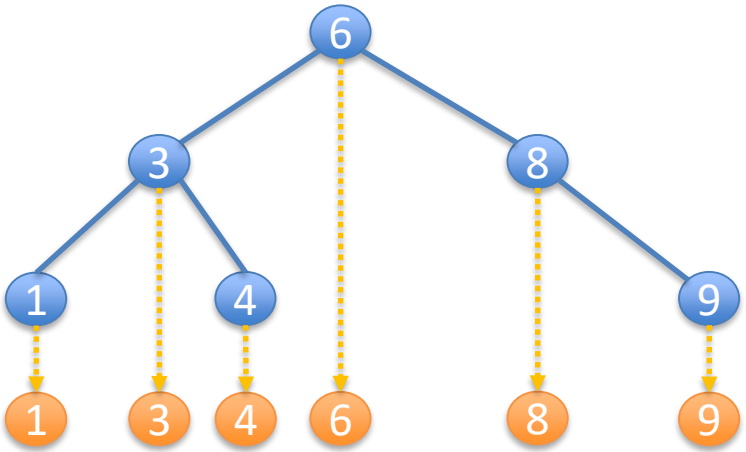
- 两种情况：
  1. root的值 =< p的值　答案就在右子树中
  2. root的值　> p的值　答案有两个备选：
     a) 就是root
     b) 左子树中继续找，并且如果找到就一定是它，因为左子树的中的元素都比根小

- 见代码

Inorder Successor in Binary Search Tree

BST与二分法的关系：

- 将BST规范的画出来，可以看到BST和二分查找非常类似，几乎是等价的，BST上有着和二分查找一样的思考方式，root就是二分查找中每次查找的mid，每次和root分情况讨论就等价于二分查找中和mid分情况讨论



Inorder Successor in Binary Search Tree

◆ 小技巧总结：

- 遇到BST上操作的问题，可以拿给定的节点（区间）与root做比较，分类讨论、分而治之

扩展问题：

- BST上求一段的和

Inorder Successor in Binary Search Tree

九章算法

- Company Tags: Facebook

考点：

- 基础数据结构上的灵活操作
- BST对根比较后的分类讨论

Inorder Successor in Binary Search Tree

能力维度：

3. 基础数据结构/算法
4. 逻辑思维/算法优化能力

BST类问题总结

- BST类问题以及其他树、二叉树上的问题常考数据结构上的灵活操作

- 解决BST类问题要从BST的性质着手（画出来从小到大、类似二分查找），经常与root比较后分类讨论

BST类问题总结

- BST类题目：
  - http://www.lintcode.com/zh-cn/problem/validate-binary-search-tree/
  - http://www.lintcode.com/zh-cn/problem/binary-search-tree-iterator/

二叉树类问题

Binary Tree Flipping

http://www.lintcode.com/zh-cn/problem/binary-tree-flipping/

http://www.jiuzhang.com/solutions/binary-tree-flipping/

Binary Tree Flipping

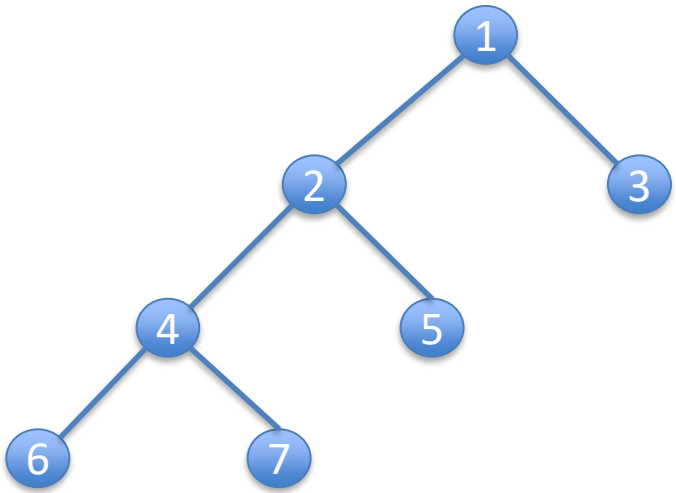
思路：

- 问题一：这到底是一棵什么样的树？
  - 试着把这个树画出来

Binary Tree Flipping

思路：

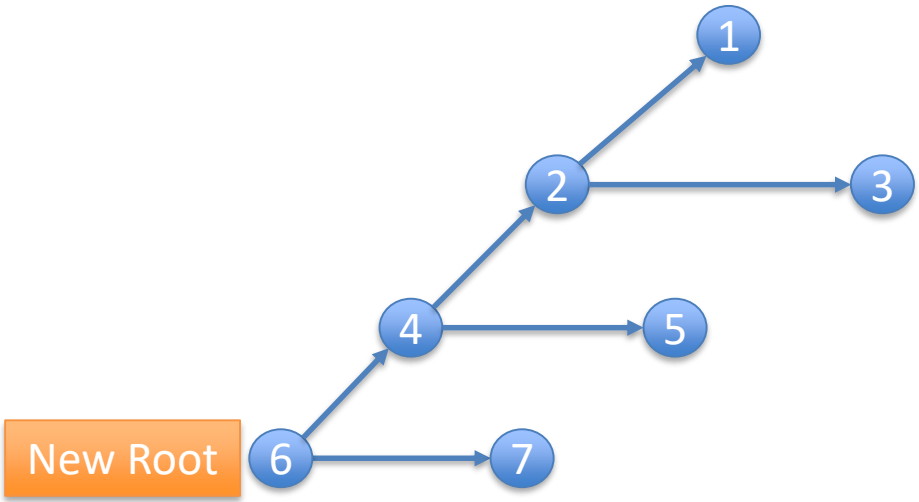
- 问题一：这到底是一棵什么样的树？
  - 试着把这个树画出来



Binary Tree Flipping

思路：

- 问题二：按题目要求翻转后是一棵什么样的树？

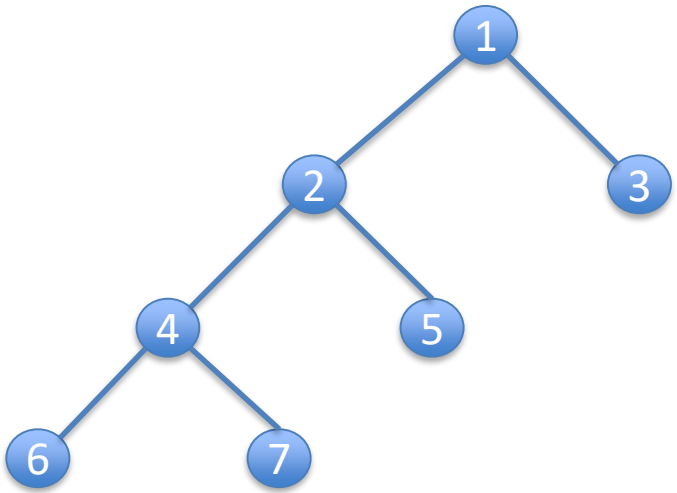


Binary Tree Flipping

- 问题三：用什么算法实现？
  - DFS 递归翻转

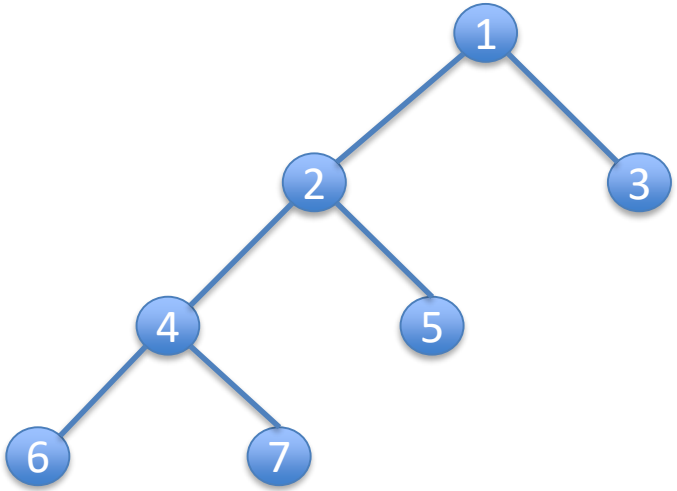
Binary Tree Flipping

- 问题四：是先翻转当前节点再递归，还是先递归再翻转？
  - 想想怎样做使得上下节点互不影响
  - 同学们会怎么选？



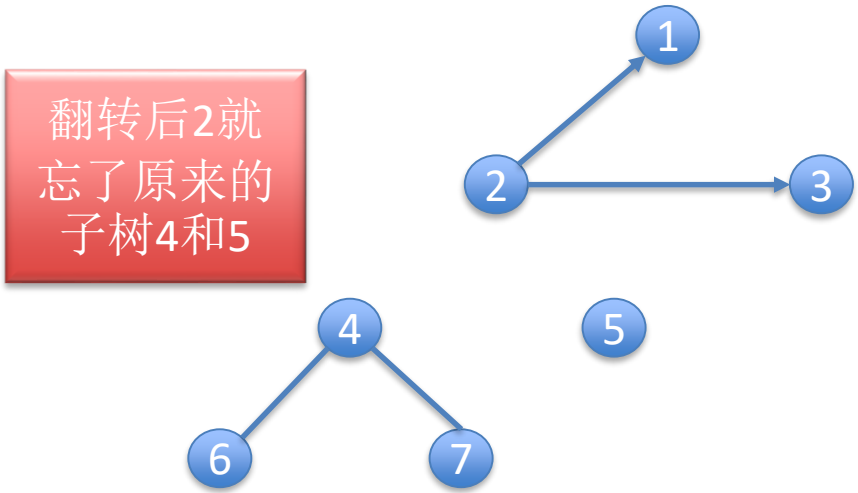
Binary Tree Flipping

- 问题四：是先翻转当前节点再递归，还是先递归再翻转？
  - 想想怎样做使得上下节点互不影响
  - 假如是先翻转再递归？



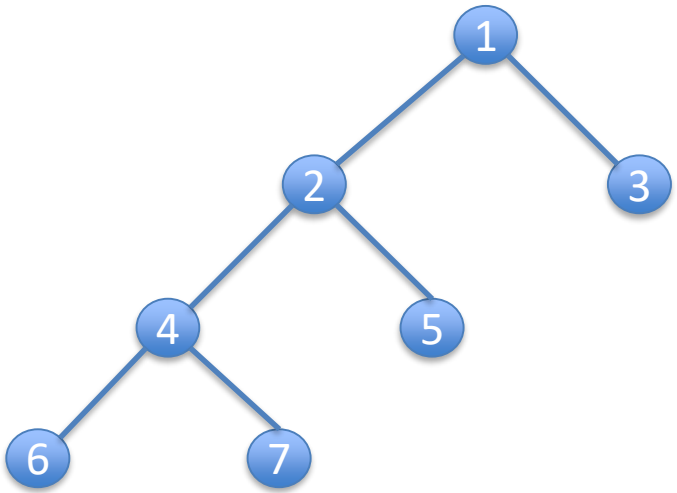
Binary Tree Flipping

- 问题四：是先翻转当前节点再递归，还是先递归再翻转？
  - 想想怎样做使得上下节点互不影响
  - 假如是先翻转再递归？



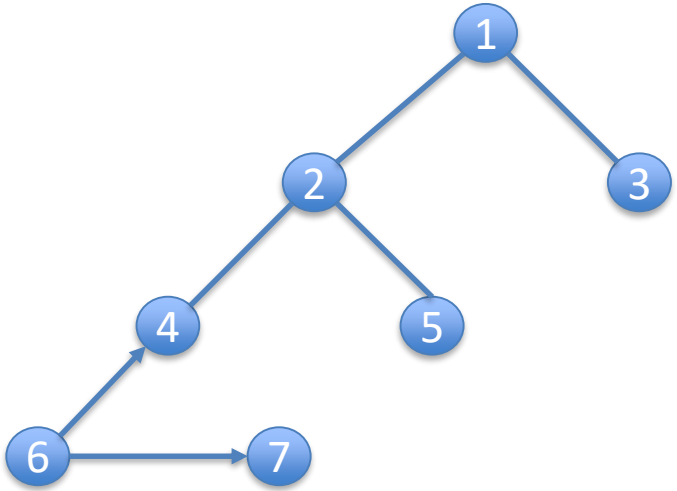
Binary Tree Flipping

- 问题四：是先翻转当前节点再递归，还是先递归再翻转？
  - 想想怎样做使得上下节点互不影响
  - 所以是？



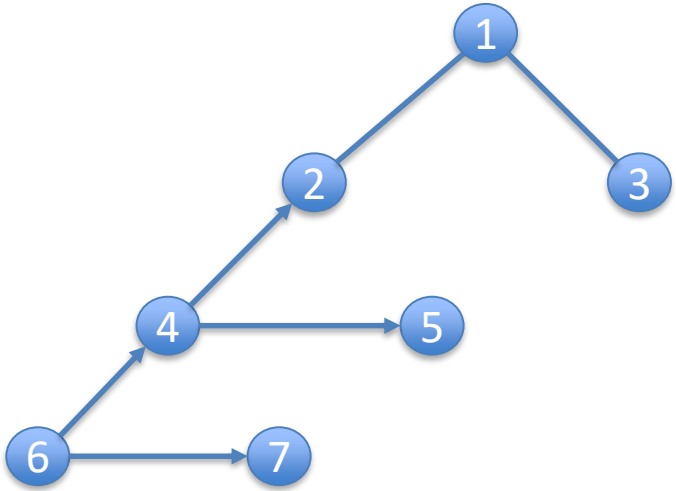
Binary Tree Flipping

- 问题四：是先翻转当前节点再递归，还是先递归再翻转？
  - 想想怎样做使得上下节点互不影响
  - 所以是？



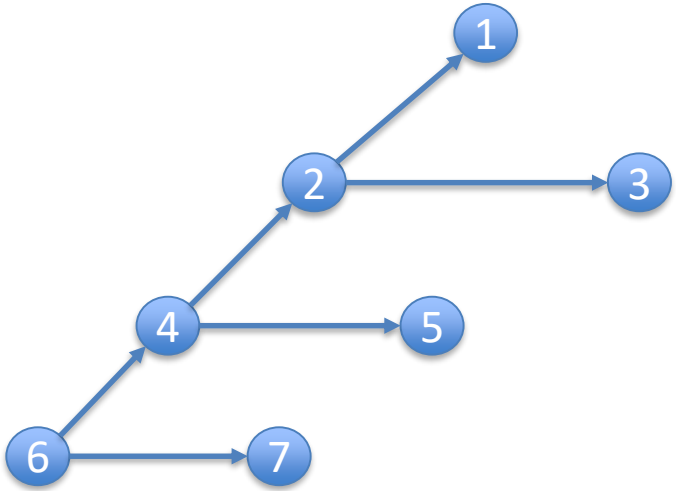
Binary Tree Flipping

- 问题四：是先翻转当前节点再递归，还是先递归再翻转？
  - 想想怎样做使得上下节点互不影响
  - 所以是？



Binary Tree Flipping

- 问题四：是先翻转当前节点再递归，还是先递归再翻转？
  - 想想怎样做使得上下节点互不影响
  - 所以是？
  - 代码实现？



Binary Tree Flipping

- Company Tags: LinkedIn

考点：

- 基础数据结构上的灵活操作
- 二叉树的遍历，遍历后的形状改变

Binary Tree Flipping

能力维度：

1. 理解问题
3. 基础数据结构/算法

Binary Tree Leaves Order Traversal

http://www.lintcode.com/zh-cn/problem/binary-tree-leaves-order-traversal/

http://www.jiuzhang.com/solutions/binary-tree-leaves-order-traversal/

Binary Tree Leaves Order Traversal

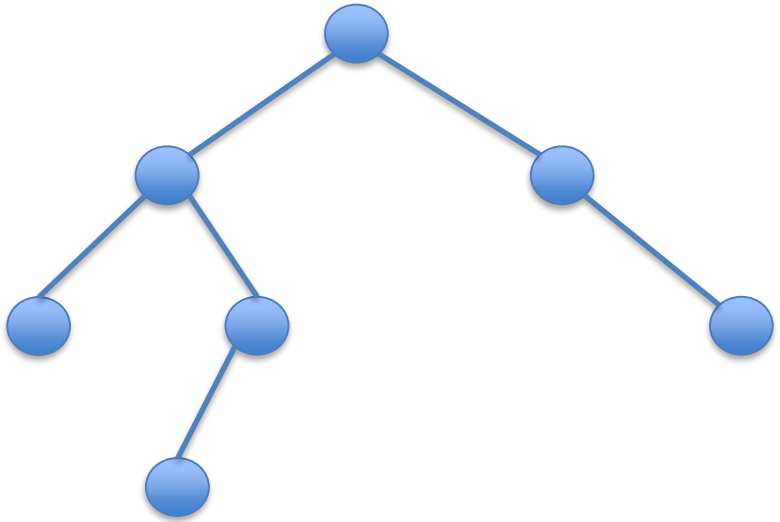
思路：

- 一层一层的剥开二叉树，如何确定这一层包含哪些节点？
  - 画出来看看

Binary Tree Leaves Order Traversal

思路：

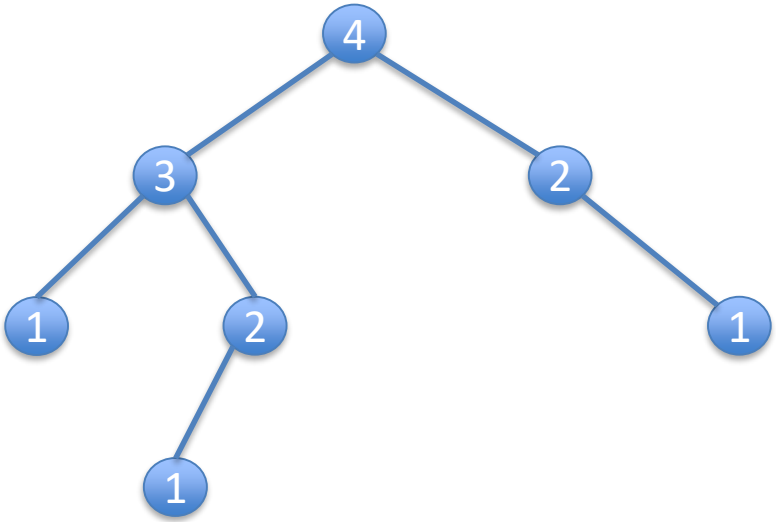
- 一层一层的剥开二叉树，如何确定这一层包含哪些节点？
  - 画出来看看



Binary Tree Leaves Order Traversal

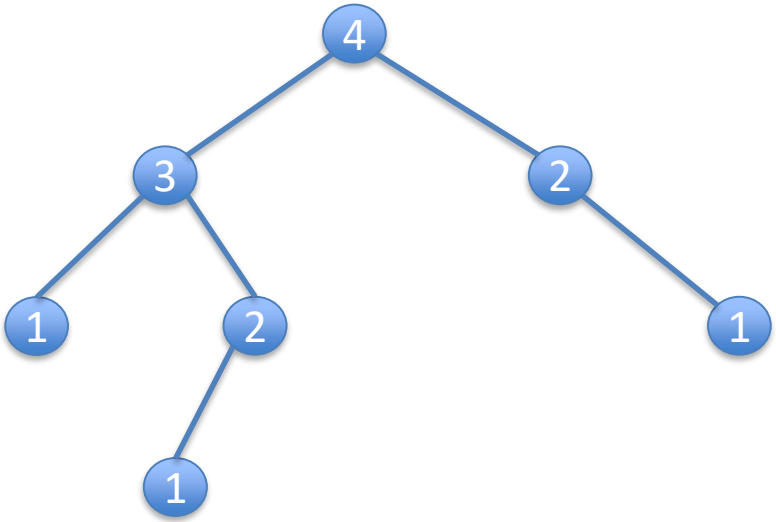
思路：

- 一层一层的剥开二叉树，如何确定这一层包含哪些节点？
  - 画出来看看



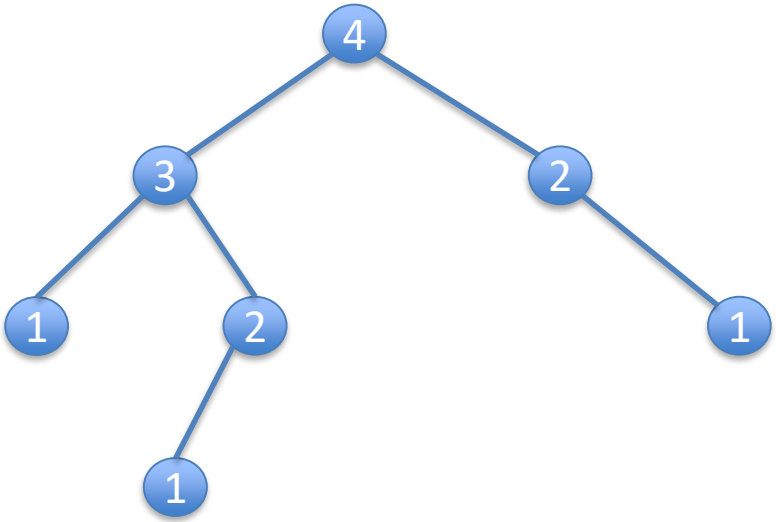
Binary Tree Leaves Order Traversal

- 看看每个子树的高度？
  - 其实第k层包含的就是所有高度为k的节点



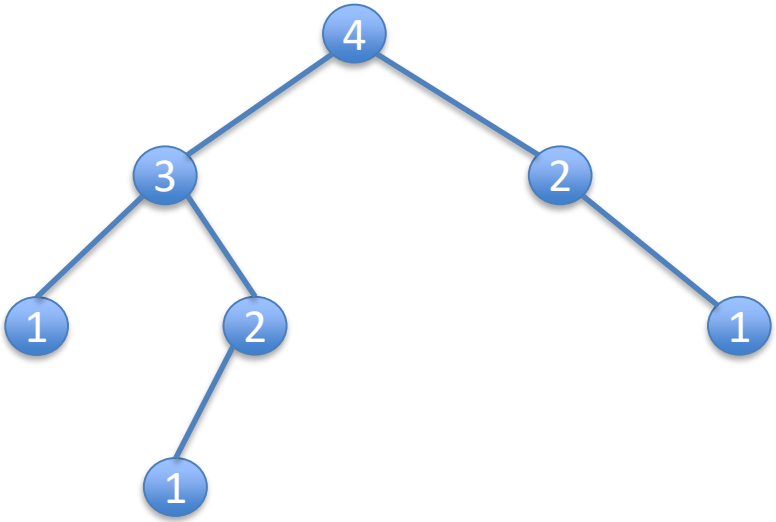
Binary Tree Leaves Order Traversal

- 同一层中，要求的顺序是从左往右



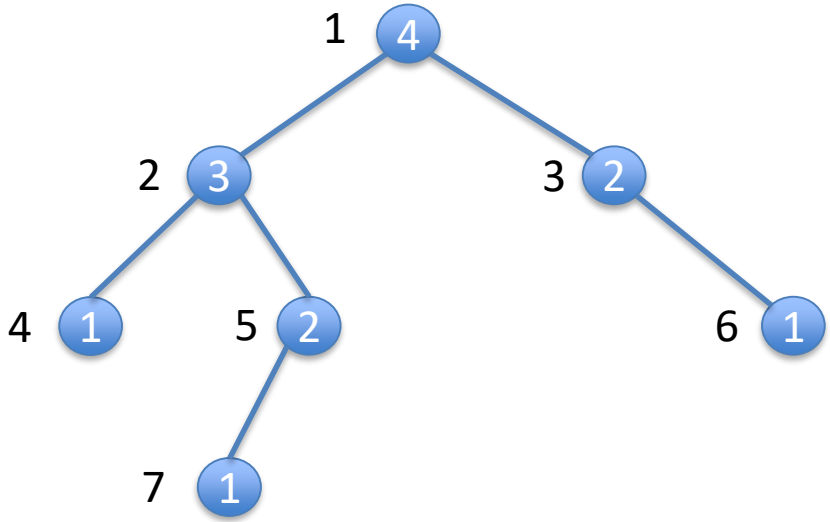
Binary Tree Leaves Order Traversal

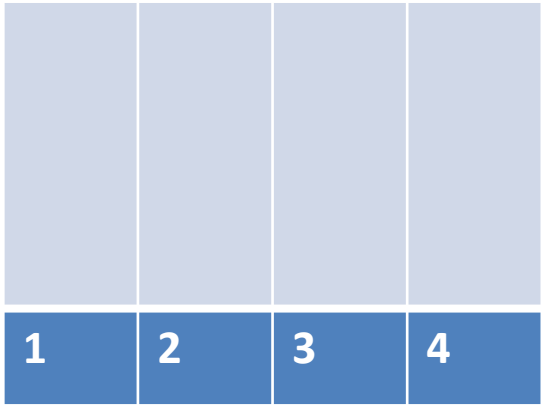
- 所以怎么求高度？怎么保存答案？
  - DFS计算节点高度，hash 保存答案



Binary Tree Leaves Order Traversal

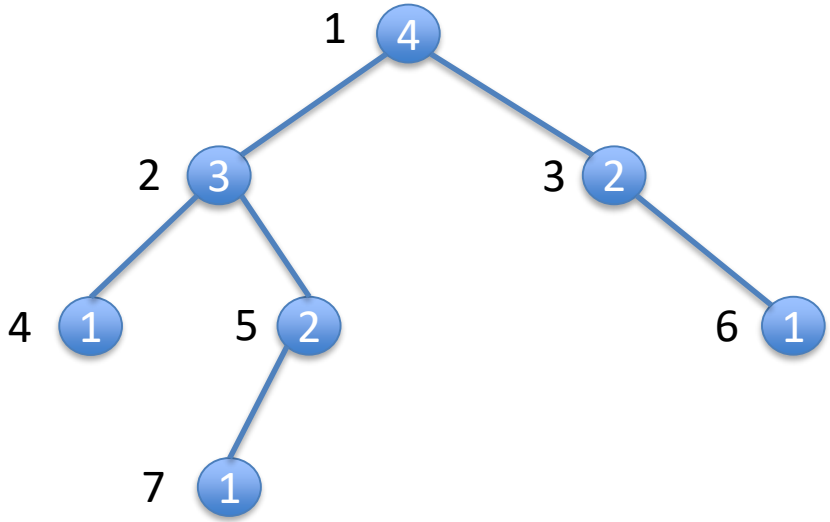
- 所以怎么求高度？怎么保存答案？
  - DFS计算节点高度，hash 保存答案





Binary Tree Leaves Order Traversal

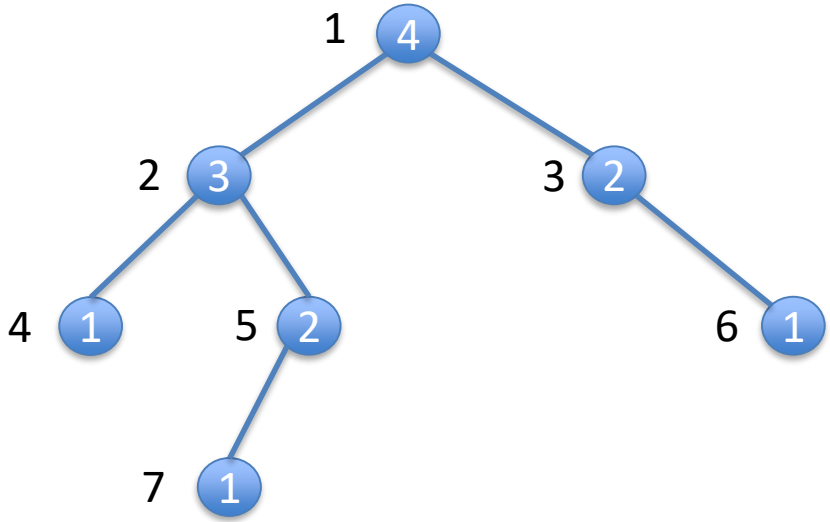
- 所以怎么求高度？怎么保存答案？
  - DFS计算节点高度，hash 保存答案



4 7 6	5 3	2	1
1	2	3	4

Binary Tree Leaves Order Traversal

- 怎么实现？
  - 见代码



4 7 6	5 3	2	1
1	**2**	**3**	**4**

Binary Tree Leaves Order Traversal

- Company Tags：LinkedIn

考点：

- 基础数据结构上的灵活操作
- 二叉树的遍历

类似题：

- http://www.lintcode.com/zh-cn/problem/maximum-subtree/

Binary Tree Leaves Order Traversal

能力维度：

3. 基础数据结构/算法

Binary Tree Vertical Order Traversal

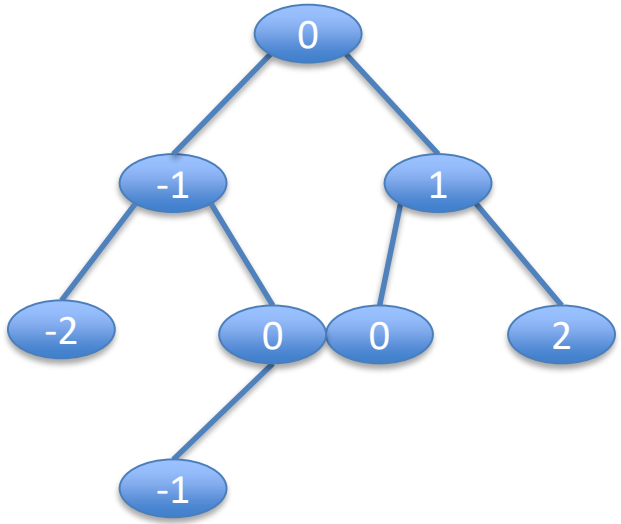
http://www.lintcode.com/zh-cn/problem/binary-tree-vertical-order-traversal/

http://www.jiuzhang.com/solutions/binary-tree-vertical-order-traversal/

Binary Tree Vertical Order Traversal

思路：

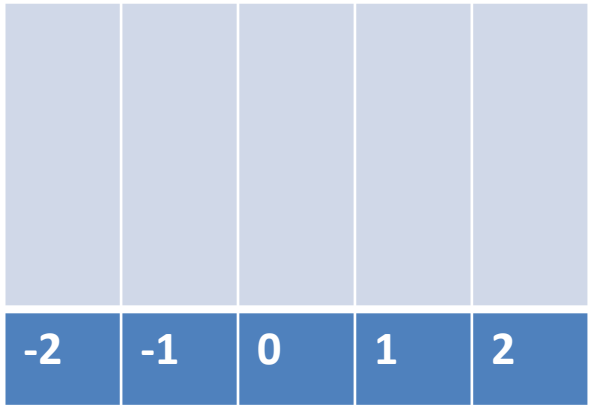
- 这一题有三个顺序:
  - 第一，列数从小到大
  - 第二，列数相同时行数从小到大
  - 第三，列数行数都相同时，从左到右

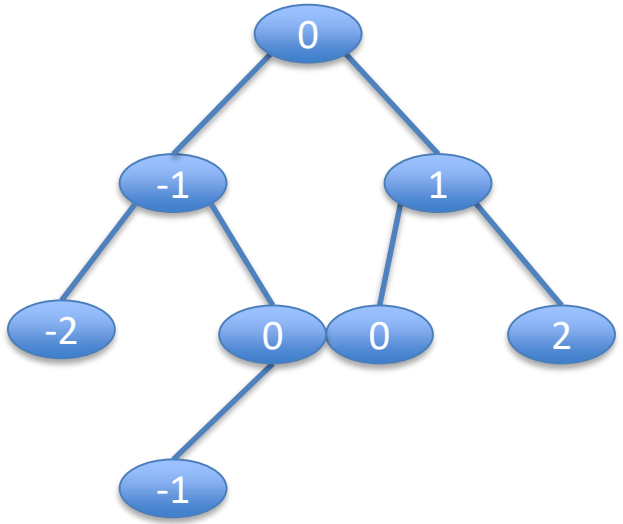


Binary Tree Vertical Order Traversal

- 怎样计算列数？
  - Root为0 向左-1　向右+1

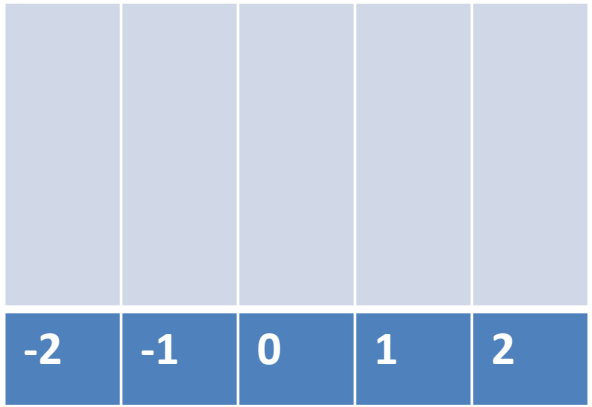
- 怎样保证第一个顺序，列数从小到大
  - Hash

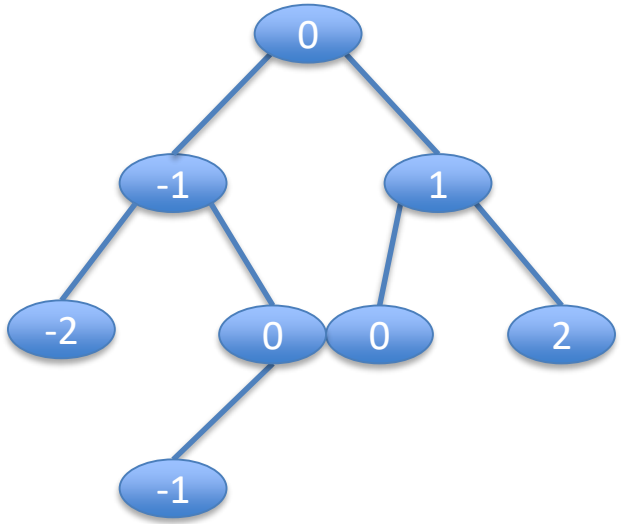




Binary Tree Vertical Order Traversal

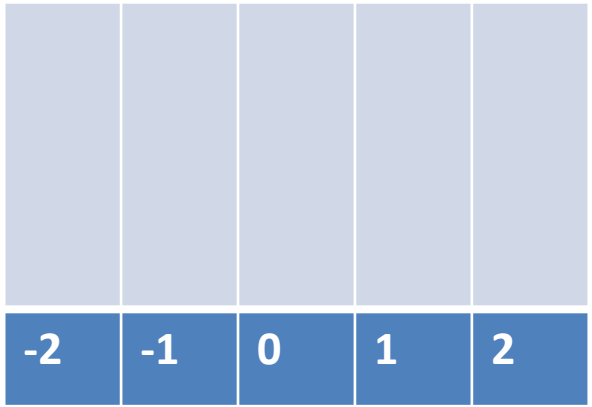
- 怎样保证第二个顺序，行数从小到大
  - BFS 一行一行的访问

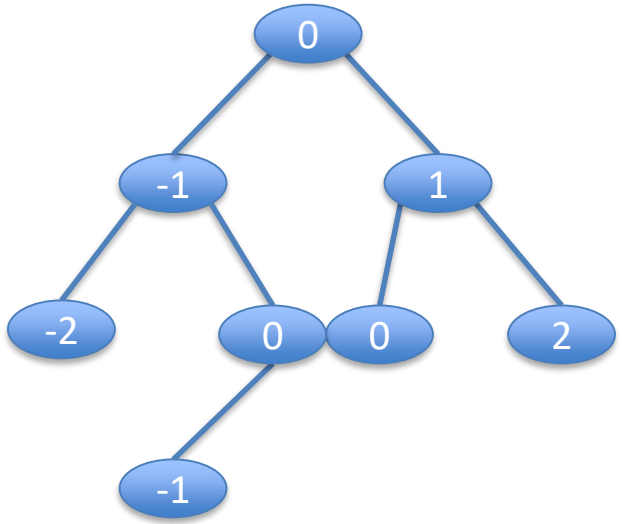




Binary Tree Vertical Order Traversal

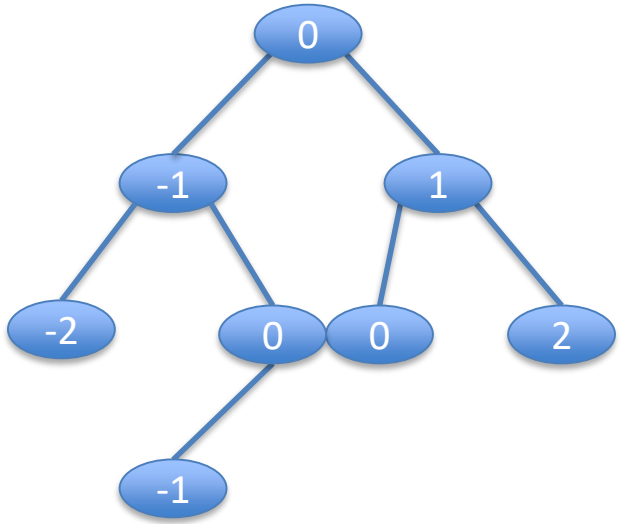
- 怎样保证第三个顺序
  - BFS已经保证了





Binary Tree Vertical Order Traversal

- 怎样实现BFS？
  - 队列



Binary Tree Vertical Order Traversal

- Company Tags：Google Facebook

考点：

- 基础数据结构上的灵活操作
- 二叉树的遍历

Binary Tree Vertical Order Traversal

能力维度：

3. 基础数据结构/算法
4. 逻辑思维/算法优化能力

二叉树类问题总结

二叉树类题目：

Binary Tree Preorder Traversal	Binary Tree Inorder Traversal	Binary Tree Postorder Traversal
Maximum Depth of Binary Tree	Balanced Binary Tree	Lowest Common Ancestor
Binary Tree Maximum Path Sum II	Binary Tree Maximum Path Sum	Binary Tree Level Order Traversal

本节课知识点总结

- Guess Number Game
  - ◆ 小技巧总结：
  - – mid = l + (r - l) / 2　以保证不溢出
  - – 不知道如何下手写时，先写最简单的情况（写代码时常用思路）

- Search for a Range

- Convert BST to Greater Tree
  - ◆ 小技巧总结：
  - – DFS的两种理解方式：
    1. 按照实际执行顺序模拟
    2. 按照DFS的定义宏观理解

本节课知识点总结

- Inorder Successor in Binary Search Tree
  - ◆ 小技巧总结：
  - – 遇到BST上操作的问题，可以拿给定的节点（区间）与root做比较，分类讨论、分而治之
- Binary Tree Flipping
- Binary Tree Leaves Order Traversal
- Binary Tree Vertical Order Traversal

扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
知乎专栏： http://zhuanlan.zhihu.com/jiuzhang
微博: http://www.weibo.com/ninechapter
官网: www.jiuzhang.com