108 Convert Sorted Array to Binary Search Tree

2018年3月30日 21:35

Question:

Given an array where elements are sorted in ascending order, convert it to a height balanced BST.
For this problem, a height-balanced binary tree is defined as a binary tree in which the depth of the two subtrees of *every* node never differ by more than 1.
Example:
Given the sorted array: [-10,-3,0,5,9],
One possible answer is: [0,-3,9,-10,null,5], which represents the following height balanced BST:

```
0
   /\
  -3  9
  /  /
 -10  5
```

来自 <https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree/description/> s
将一个按照升序排列的有序数组，转换为一棵高度平衡二叉搜索树。
此题中，一个高度平衡二叉树是指一个二叉树*每个节点*的左右两个子树的高度差的绝对值不超过1。

Solution for Python3:

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None
# Recursive Version:
class Solution1:
    def sortedArrayToBST(self, nums):
        """
        :type nums: List[int]
        :rtype: TreeNode
        """
        if not len(nums):
            return None
        mid = len(nums) // 2
        root = TreeNode(nums[mid])
        root.left = self.sortedArrayToBST(nums[:mid])
        root.right = self.sortedArrayToBST(nums[mid+1:])
        return root

# Iterative Version:
class Solution2:
    def sortedArrayToBST(self, nums):
        """
        :type nums: List[int]
        :rtype: TreeNode
        """
        from collections import deque
        import math
        if not len(nums):
            return None
        root = TreeNode(0)
        nodeDeque = deque([root])
        LIndexDeque = deque([0])
        RIndexDeque = deque([len(nums) - 1])
        while nodeDeque:
            curNode = nodeDeque.pop()
            L = LIndexDeque.pop()
            R = RIndexDeque.pop()
            mid = L + math.ceil((R - L) / 2)
            curNode.val = nums[mid]
            if L <= mid - 1:
                curNode.left = TreeNode(0)
                nodeDeque.append(curNode.left)
                LIndexDeque.append(L)
                RIndexDeque.append(mid - 1)
            if mid + 1 <= R:
                curNode.right = TreeNode(0)
                nodeDeque.append(curNode.right)
                LIndexDeque.append(mid + 1)
                RIndexDeque.append(R)
        return root

class Solution3:
    def sortedArrayToBST(self, nums):
        """
        :type nums: List[int]
        :rtype: TreeNode
        """
        from collections import deque
        import math
        if not len(nums):
            return None
        root = TreeNode(0)
        nodeDeque = deque([root, 0, len(nums) - 1])
        while nodeDeque:
            curNode = nodeDeque.popleft()
            L = nodeDeque.popleft()
            R = nodeDeque.popleft()
            mid = L + math.ceil((R - L) / 2)
            curNode.val = nums[mid]
            if L <= mid - 1:
                curNode.left = TreeNode(0)
                nodeDeque.append(curNode.left)
                nodeDeque.append(L)
                nodeDeque.append(mid - 1)
            if mid + 1 <= R:
                curNode.right = TreeNode(0)
                nodeDeque.append(curNode.right)
                nodeDeque.append(mid + 1)
                nodeDeque.append(R)
        return root
```

Solution for C++:

```cpp
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
// Recursive Version:
class Solution {
public:
    TreeNode* sortedArrayToBST(vector<int>& nums) {
        if (nums.empty()) {
           return NULL;
        }
        int mid = nums.size() / 2;
        TreeNode* root = new TreeNode(nums[mid]);
        // vector<int> L(nums.begin(), nums.begin() + mid);
        // vector<int> R(nums.begin() + mid + 1, nums.end());
        // root->left = sortedArrayToBST(L);
        // root->right = sortedArrayToBST(R);
        root->left = sortedArrayToBST(*new vector<int>(nums.begin(), nums.begin() + mid));
        root->right = sortedArrayToBST(*new vector<int>(nums.begin() + mid + 1, nums.end()));
        return root;
    }
};

// Iterative Version:
class Solution2 {
public:
    TreeNode* sortedArrayToBST(vector<int>& nums) {
        if (nums.empty()) {
           return NULL;
        }
        TreeNode* root = new TreeNode(0);
        queue<TreeNode*> nodeQue;
        nodeQue.push(root);
        queue<int> LRIndexQue;
        LRIndexQue.push(0);
        LRIndexQue.push(nums.size() - 1);
        int L = 0, R = 0, mid = 0;
        while (!nodeQue.empty()) {
           TreeNode* curNode = nodeQue.front();
           nodeQue.pop();
           L = LRIndexQue.front();
           LRIndexQue.pop();
           R = LRIndexQue.front();
           LRIndexQue.pop();
           mid = L + ceil((R - L) / 2.0);
           curNode->val = nums[mid];
           if (L <= mid - 1) {
               curNode->left = new TreeNode(0);
               nodeQue.push(curNode->left);
               LRIndexQue.push(L);
               LRIndexQue.push(mid - 1);
           }
           if (mid + 1 <= R) {
               curNode->right = new TreeNode(0);
               nodeQue.push(curNode->right);
               LRIndexQue.push(mid + 1);
               LRIndexQue.push(R);
           }
        }
        return root;
    }
};
```

Appendix:

Python 向上向下取整函数:

import math   math.ceil()/math.floor()

Python deque(iterable, maxsize):

1) deque初始化传入参数必须是可迭代对象，int类型这种就不能直接传入，可以转换成list传入。
2) 如：d = deque([2]) 就把数字2初始化传入队列中。

C++ 创建匿名vector向量:

1) 创建匿名vector数组指针：new vector<int>(v.begin(),v.begin()+6),但是这样返回的是指针。
2) 原因是vector<int> *vv = new vector<int>(v.begin(),v.begin()+6)。

3) 所以在用到匿名vector数组时可以用：*new vector<int>(v.begin(),v.begin()+6)。

C++两整数相除

1) 得整数：a/b=c都是整数。
2) 要得到小数结果：a/double(b)=c　把a和b其中一个转换成小数。
3) 进而再对小数结果向上向下取整。