

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:

```
1  # Definition for a binary tree node.
2  # class TreeNode:
3  #     def __init__(self, x):
4  #         self.val = x
5  #         self.left = None
6  #         self.right = None
7  # Recursive Version:
8  class Solution1:
9      def sortedArrayToBST(self, nums):
10         """
11         :type nums: List[int]
12         :rtype: TreeNode
13         """
14         if not len(nums):
15             return None
16         mid = len(nums) // 2
17         root = TreeNode(nums[mid])
18         root.left = self.sortedArrayToBST(nums[:mid])
19         root.right = self.sortedArrayToBST(nums[mid+1:])
20         return root
21
22 # Iterative Version:
23 class Solution2:
24     def sortedArrayToBST(self, nums):
25         """
26         :type nums: List[int]
27         :rtype: TreeNode
28         """
29         from collections import deque
30         import math
31         if not len(nums):
32             return None
33         root = TreeNode(0)
34         nodeDeque = deque([root])
35         LIndexDeque = deque([0])
36         RIndexDeque = deque([len(nums) - 1])
37         while nodeDeque:
38             curNode = nodeDeque.pop()
39             L = LIndexDeque.pop()
40             R = RIndexDeque.pop()
```

```

41         mid = L + math.ceil((R - L) / 2)
42         curNode.val = nums[mid]
43         if L <= mid - 1:
44             curNode.left = TreeNode(0)
45             nodeDeque.append(curNode.left)
46             LIndexDeque.append(L)
47             RIndexDeque.append(mid - 1)
48         if mid + 1 <= R:
49             curNode.right = TreeNode(0)
50             nodeDeque.append(curNode.right)
51             LIndexDeque.append(mid + 1)
52             RIndexDeque.append(R)
53     return root
54
55 class Solution3:
56     def sortedArrayToBST(self, nums):
57         """
58         :type nums: List[int]
59         :rtype: TreeNode
60         """
61         from collections import deque
62         import math
63         if not len(nums):
64             return None
65         root = TreeNode(0)
66         nodeDeque = deque([root, 0, len(nums) - 1])
67         while nodeDeque:
68             curNode = nodeDeque.popleft()
69             L = nodeDeque.popleft()
70             R = nodeDeque.popleft()
71             mid = L + math.ceil((R - L) / 2)
72             curNode.val = nums[mid]
73             if L <= mid - 1:
74                 curNode.left = TreeNode(0)
75                 nodeDeque.append(curNode.left)
76                 nodeDeque.append(L)
77                 nodeDeque.append(mid - 1)
78             if mid + 1 <= R:
79                 curNode.right = TreeNode(0)
80                 nodeDeque.append(curNode.right)
81                 nodeDeque.append(mid + 1)
82                 nodeDeque.append(R)
83         return root

```

Solution for C++:

```

1  /**
2   * Definition for a binary tree node.
3   * struct TreeNode {
4   *     int val;
5   *     TreeNode *left;
6   *     TreeNode *right;
7   *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
8   * };
9   */
10 // Recursive Version:
11 class Solution {
12 public:
13     TreeNode* sortedArrayToBST(vector<int>& nums) {
14         if (nums.empty()) {
15             return NULL;
16         }
17         int mid = nums.size() / 2;
18         TreeNode* root = new TreeNode(nums[mid]);

```

```

19         // vector<int> L(nums.begin(), nums.begin() + mid);
20         // vector<int> R(nums.begin() + mid + 1, nums.end());
21         // root->left = sortedArrayToBST(L);
22         // root->right = sortedArrayToBST(R);
23         root->left = sortedArrayToBST(*new vector<int>(nums.begin(), nums.begin() + mid));
24         root->right = sortedArrayToBST(*new vector<int>(nums.begin() + mid + 1, nums.end()));
25         return root;
26     }
27 };
28
29 // Iterative Version:
30 class Solution2 {
31 public:
32     TreeNode* sortedArrayToBST(vector<int>& nums) {
33         if (nums.empty()) {
34             return NULL;
35         }
36         TreeNode* root = new TreeNode(0);
37         queue<TreeNode*> nodeQueue;
38         nodeQueue.push(root);
39         queue<int> LRIndexQueue;
40         LRIndexQueue.push(0);
41         LRIndexQueue.push(nums.size() - 1);
42         int L = 0, R = 0, mid = 0;
43         while (!nodeQueue.empty()) {
44             TreeNode* curNode = nodeQueue.front();
45             nodeQueue.pop();
46             L = LRIndexQueue.front();
47             LRIndexQueue.pop();
48             R = LRIndexQueue.front();
49             LRIndexQueue.pop();
50             mid = L + ceil((R - L) / 2.0);
51             curNode->val = nums[mid];
52             if (L <= mid - 1) {
53                 curNode->left = new TreeNode(0);
54                 nodeQueue.push(curNode->left);
55                 LRIndexQueue.push(L);
56                 LRIndexQueue.push(mid - 1);
57             }
58             if (mid + 1 <= R) {
59                 curNode->right = new TreeNode(0);
60                 nodeQueue.push(curNode->right);
61                 LRIndexQueue.push(mid + 1);
62                 LRIndexQueue.push(R);
63             }
64         }
65         return root;
66     }
67 };

```

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: $a/\text{double}(b)=c$ 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。