有序链表转化为二叉搜索树BST\_LeetCode\_109\_ConvertSortedListToBinarySearchTree\_Medium

# 有序链表转化为二叉搜索树

## 题目介绍

/\*\*

\* 有序链表转化为二叉搜索树BST\_LeetCode\_109\_ConvertSortedListToBinarySearchTree\_Medium

\* 难度：Medium

\* DateTime：2018-10-15

\* https://leetcode.com/problems/convert-sorted-list-to-binary-search-tree/description/

\* <p>

\* 题目介绍：

\* Given a singly linked list 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:

\* <p>

\* Given the sorted linked list: [-10,-3,0,5,9],

\* <p>

\* One possible answer is: [0,-3,9,-10,null,5], which represents the following height balanced BST:

\* <p>

\* 0

\* / \

\* -3 9

\* / /

\* -10 5

\* <p>

## 思路分析

\* 思路分析：

\* 方法1：快慢指针+递归方法；

\* 递归方法：

\* 终止条件:链表头节点为null返回null;头节点的next为null，则转化为树节点并返回。

\* 递归方法内容：利用快慢指针寻找以head开头以dummyTail结尾的中间节点;

\* 设置dummyTai参数意义：保护原链表的结构不变，有的方法直接不断截断链表，以null为作为结束标志。

\* 转换为TreeNode root,递归寻找root.left和root.right。

\* 方法2: ArrayList+递归；

\* 首先将链表左右值顺序转化为int数组，然后递归转化为BST.

## Java代码

/\*\*

\* 方法1：快慢指针+递归方法；

\* 不会破坏原链表结构

\*/

public TreeNode sortedListToBST(ListNode head) {

return helper(head, null);

}

/\*\*

\* 递归方法: dummyTail作为结束标志

\*/

public TreeNode helper(ListNode head, ListNode dummyTail) {

if (head == dummyTail) return null;

if (head.next == dummyTail) return new TreeNode(head.val);//减少递归次数，删除也可以。

//利用快慢指针寻找链表的中间节点

ListNode slow = head, fast = head.next;//fast初始化head或head.next都可以

while (fast != dummyTail && fast.next != dummyTail) {

slow = slow.next;

fast = fast.next.next;

}

//转化为树的root节点

TreeNode root = new TreeNode(slow.val);

root.left = helper(head, slow);//递归寻找左子树

root.right = helper(slow.next, dummyTail);//递归寻找右子树

return root;

}

/\*\*

\* 方法2：ArrayList+递归

\*/

public TreeNode sortedListToBST2(ListNode head) {

if (head == null) return null;

if (head.next == null) return new TreeNode(head.val);

ArrayList<Integer> list = new ArrayList<Integer>();

while (head != null) {

list.add(head.val);

head = head.next;

}

return helper(list, 0, list.size() - 1);

}

/\*\*

\* 递归方法

\*/

public TreeNode helper(ArrayList<Integer> list, int start, int end) {

if (start > end) return null;//递归终止条件

if (start == end) return new TreeNode(list.get(start));//这个减少递归次数，没有也可以

int mid = (start + end) >> 1;

TreeNode root = new TreeNode(list.get(mid));

root.left = helper(list, start, mid - 1);//一开始由于start写成0，导致耽误10分钟

root.right = helper(list, mid + 1, end);

return root;

}