1. Convert Sorted List to Binary Search Tree

Medium

140278FavoriteShare

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:**

Given the sorted linked list: [-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

**Solution**

Approach1

先将链表转为数组，然后使用108的方法构建：将区间[l, r]均分，依次为左子树/根节点/右子树，递归建树

/\*\*  
 \* Definition for singly-linked list.  
 \* struct ListNode {  
 \* int val;  
 \* ListNode \*next;  
 \* ListNode(int x) : val(x), next(NULL) {}  
 \* };  
 \*/  
/\*\*  
 \* Definition for a binary tree node.  
 \* struct TreeNode {  
 \* int val;  
 \* TreeNode \*left;  
 \* TreeNode \*right;  
 \* TreeNode(int x) : val(x), left(NULL), right(NULL) {}  
 \* };  
 \*/  
class Solution {  
public:  
 TreeNode\* sortedListToBST(ListNode\* head) {  
 vector<int>nums;  
 ListNode\* p = head;  
 while(p){  
 nums.push\_back(p->val);  
 p = p->next;  
 }  
 sort(nums.begin(), nums.end());  
 TreeNode\* root = NULL;  
 create(nums, 0, nums.size() - 1, root);  
 return root;  
 }  
 void create(vector<int>&nums, int L, int R, TreeNode\* &root){  
 if(L > R)return;  
 int mid = (L + R) / 2;  
 Insert(root, nums[mid]);  
 create(nums, L, mid - 1, root);  
 create(nums, mid + 1, R, root);  
 }  
 void Insert(TreeNode\* &root, int x){  
 if(root == NULL){  
 root = new TreeNode(x);  
 return;  
 }  
 if(x > root->val)Insert(root->right, x);  
 else Insert(root->left, x);  
 }  
};

Approach2

同第一种方法，省去转数组的操作。在链表中寻找中间节点的方法为：使用两个移动速度不同的指针

lnode find\_mid(lnode head){  
 if(head == NULL)return NULL;  
 lnode p = head, q = head, p\_pre = NULL;  
 while(q && q->next){  
 p\_pre = p;  
 p = p->next;  
 q = q->next->next;  
 }  
 if(p\_pre)p\_pre->next = NULL;  
 return p;  
 }

Approach3

利用中序遍历的特性：顺序遍历链表，递归建左子树/根节点/右子树

class Solution {  
public:  
 ListNode\* head;  
 TreeNode\* sortedListToBST(ListNode\* head) {  
 this->head = head;  
 int len = find\_size(head);  
 return create(0, len - 1);  
 }  
 TreeNode\* create(int l, int r){  
 if(l > r)return NULL;  
 int mid = (l + r) / 2;  
 TreeNode\* left = create(l, mid - 1);  
 TreeNode\* root = new TreeNode(this->head->val);  
 this->head = this->head->next;  
 root->left = left;  
 root->right = create(mid + 1, r);  
 return root;  
 }  
 int find\_size(ListNode\* p){  
 int res = 0;  
 while(p){  
 res += 1;  
 p = p->next;  
 }  
 return res;  
 }   
};