Convert Sorted List to Binary Search Tree

Given a singly linked list where elements are sorted in ascending order, convert it to a height balanced BST.

Solution 1

count is a function to calculate the size of list.

Key words: inorder traversal.

```
class Solution {
public:
    ListNode *list;
    int count(ListNode *node){
        int size = 0;
        while (node) {
            ++size;
            node = node->next;
        }
        return size;
    }

    TreeNode *generate(int n){
        if (n == 0)
            return NULL;
        TreeNode *node = new TreeNode(0);
        node->left = generate(n / 2);
        node->val = list->val;
        list = list->next;
        node->right = generate(n - n / 2 - 1);
        return node;
    }

    TreeNode *sortedListToBST(ListNode *head) {
        this->list = head;
        return generate(count(head));
    }
};
```

written by vaputa original link here

Solution 2

```
private ListNode node;

public TreeNode sortedListToBST(ListNode head) {
    if(head == null){
        return null;
    }

    int size = 0;
    ListNode runner = head;
    node = head;

    while(runner != null){
        runner = runner.next;
        size ++;
    }

    return inorderHelper(0, size - 1);
}

public TreeNode inorderHelper(int start, int end){
    if(start > end){
        return null;
    }

    int mid = start + (end - start) / 2;
    TreeNode left = inorderHelper(start, mid - 1);

    TreeNode treenode = new TreeNode(node.val);
    treenode.left = left;
    node = node.next;

    TreeNode right = inorderHelper(mid + 1, end);
    treenode.right = right;

    return treenode;
}
```

written by treesbug original link here

Solution 3

```
class Solution {
public:
    TreeNode *sortedListToBST(ListNode *head)
    {
        return sortedListToBST( head, NULL );
    }

private:
    TreeNode *sortedListToBST(ListNode *head, ListNode *tail)
    {
        if( head == tail )
            return NULL;
        if( head->next == tail )    //
        {
            TreeNode *root = new TreeNode( head->val );
            return root;
        }
        ListNode *mid = head, *temp = head;
        while( temp != tail && temp->next != tail )    // 寻找中间节点
        {
            mid = mid->next;
            temp = temp->next->next;
        }
        TreeNode *root = new TreeNode( mid->val );
        root->left = sortedListToBST( head, mid );
        root->right = sortedListToBST( mid->next, tail );
        return root;
    }
};
```

written by AllenYick original link here