

# Problem 109: Convert Sorted List to Binary Search Tree

## Problem Information

**Difficulty:** Medium

**Acceptance Rate:** 65.48%

**Paid Only:** No

**Tags:** Linked List, Divide and Conquer, Tree, Binary Search Tree, Binary Tree

## Problem Description

Given the `head` of a singly linked list where elements are sorted in **ascending order**, convert it to a **height-balanced binary search tree**.

**Example 1:**



**Input:** head = [-10,-3,0,5,9] **Output:** [0,-3,9,-10,null,5] **Explanation:** One possible answer is [0,-3,9,-10,null,5], which represents the shown height balanced BST.

**Example 2:**

**Input:** head = [] **Output:** []

**Constraints:**

\* The number of nodes in `head` is in the range `[0, 2 \* 104]`. \*  $-105 \leq \text{Node.val} \leq 105$

## Code Snippets

C++:

```

/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
 * right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* sortedListToBST(ListNode* head) {
        }
    };
}

```

### Java:

```

/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * };
 */
/**
 * Definition for a binary tree node.
 * class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) { this.val = val;
 * this.left = left; this.right = right; }
 * };
 */
class Solution {
    public TreeNode sortedListToBST(ListNode head) {
        if (head == null) return null;
        if (head.next == null) return new TreeNode(head.val);
        // Find the middle node
        ListNode slow = head;
        ListNode fast = head;
        ListNode prev = null;
        while (fast != null && fast.next != null) {
            prev = slow;
            slow = slow.next;
            fast = fast.next.next;
        }
        // Now slow is the middle node
        // Recur for left half
        prev.next = null;
        TreeNode left = sortedListToBST(head);
        // Recur for right half
        TreeNode right = sortedListToBST(slow.next);
        // Create the root node
        TreeNode root = new TreeNode(slow.val);
        root.left = left;
        root.right = right;
        return root;
    }
}

```

```

* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public TreeNode sortedListToBST(ListNode head) {

}
}

```

### Python3:

```

# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:

    def sortedListToBST(self, head: Optional[ListNode]) -> Optional[TreeNode]:

```