

第四周练习

张睿恒 2024302182001

T1 LeetCode108 简单

```
/*
 * 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* sortedArrayToBST(vector<int>& nums) {
        return build(nums, 0, nums.size() - 1);
    }
private:
    TreeNode* build(const vector<int>& nums, int l, int r) {
        if (l > r) return nullptr;
        const int m = (l+r)/2;
        return new TreeNode(nums[m], build(nums, l, m-1), build(nums, m+1, r));
    }
};
```

Lee X | Accepted X



← All Submissions



Accepted 31 / 31 testcases passed
itheng submitted at Oct 29, 2025 18:07



Solution

⌚ Runtime



5 ms | Beats **18.39%**

Analyze Complexity

🐏 Memory

23.02 MB | Beats **22.69%**



Code | C++

```
/**
```

T2 LeetCode98 简单

```
/**  
 * 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:  
    bool isValidBST(TreeNode* root) {  
        stack<TreeNode*> stack;  
        TreeNode* pred = NULL;  
  
        while (root != NULL || !stack.empty()) {  
            while (root != NULL) {  
                stack.push(root);  
                root = root->left;  
            }  
            root = stack.top();  
            stack.pop();  
            if (pred && pred->val >= root->val) return false;  
            pred = root;  
            root = root->right;  
        }  
  
        return true;  
    }  
};
```

Lee X | Accepted X

← All Submissions



Accepted 86 / 86 testcases passed

Itheng submitted at Oct 29, 2025 18:16



Solution

Runtime

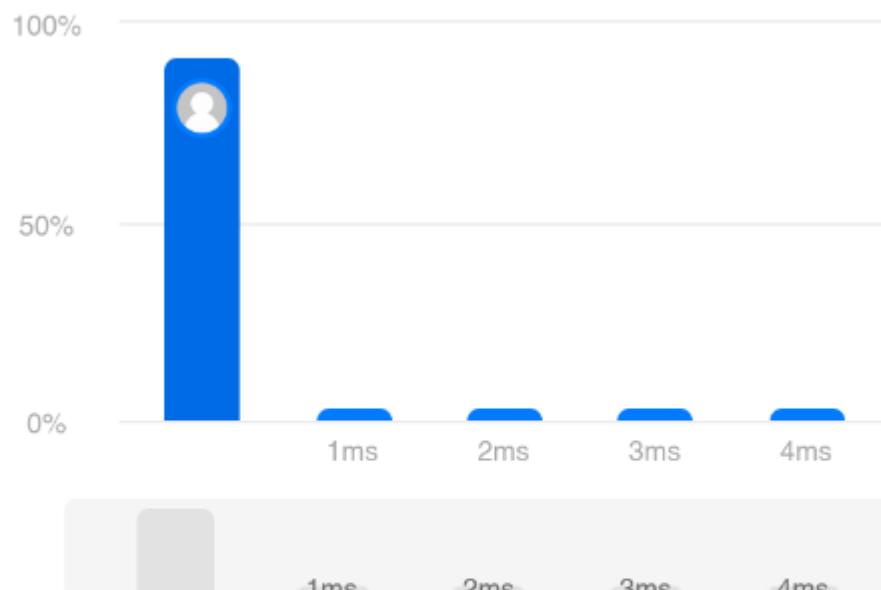


0 ms | Beats 100.00%

Analyze Complexity

Memory

21.87 MB | Beats 75.31%



Code | C++

```
/**
```

T3 LeetCode701 中等

```
/**  
 * 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* insertIntoBST(TreeNode* root, int val) {  
        if(root == NULL) return new TreeNode(val);  
        if(root->val > val) root->left = insertIntoBST(root->left, val);  
        else root->right = insertIntoBST(root->right, val);  
  
        return root;  
    }  
};
```

Lee X

Accepted X



← All Submissions



Accepted 35 / 35 testcases passed

Itheng submitted at Oct 29, 2025 18:28



Solution

⌚ Runtime

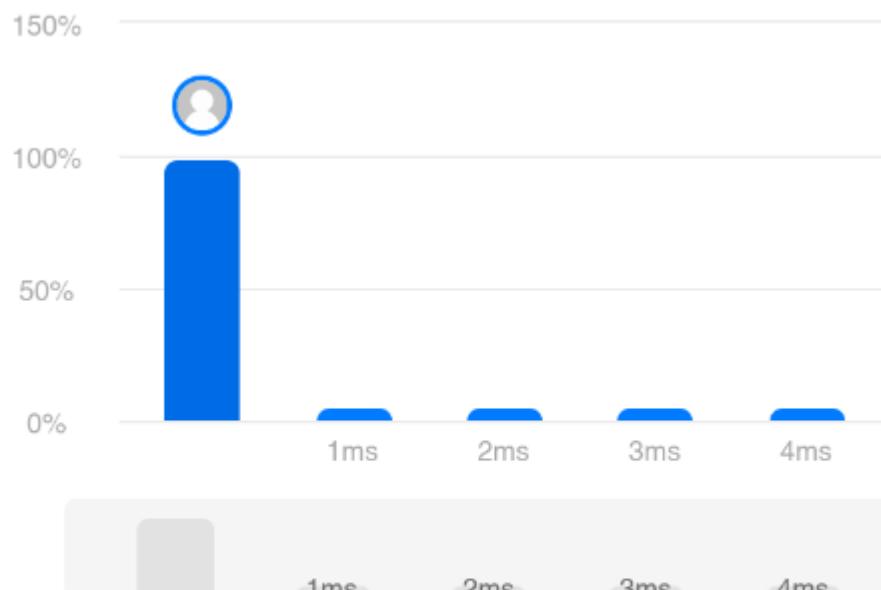


0 ms | Beats 100.00% 🎉

⭐ Analyze Complexity

🐏 Memory

59.08 MB | Beats 99.73% 🎉



Code | C++

```
/**
```

T4 LeetCode230 中等

```
class Solution {
public:
    int kthSmallest(TreeNode* root, int k) {
        int cntl = count(root->left);

        if (cntl == k-1) return root->val;
        if (cntl >= k) return kthSmallest(root->left,k);
        return kthSmallest(root->right, k-1-cntl);
    }

private:
    int count(TreeNode* root) {
        if (root == NULL) return 0;
        return 1+count(root->left)+count(root->right);
    }
};
```

Lee X | Accepted X

← All Submissions



Accepted 93 / 93 testcases passed

Itheng submitted at Oct 29, 2025 18:48



Solution

Runtime

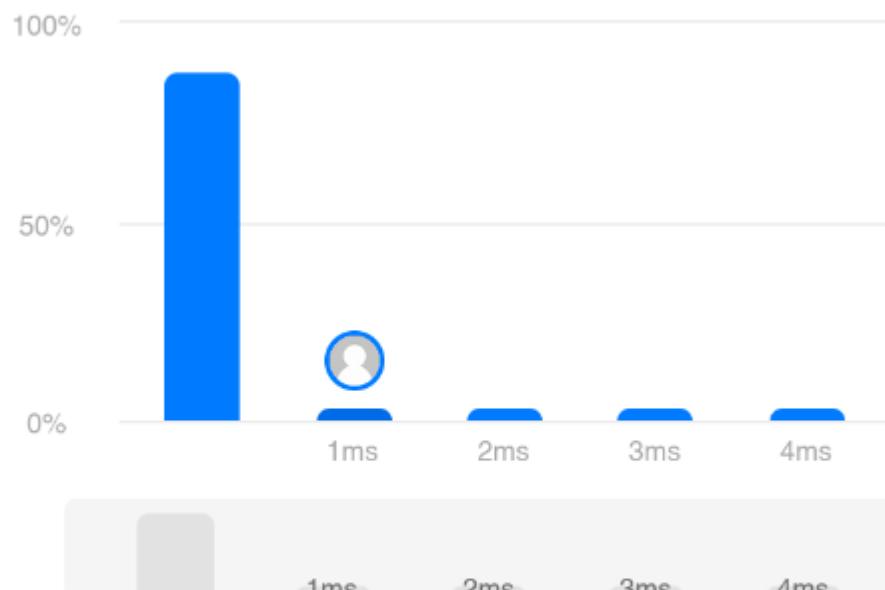


1 ms | Beats 11.78%

Analyze Complexity

Memory

24.10 MB | Beats 99.31% 🎉



Code | C++

```
class Solution {
```

T5 LeetCode501 中等

```

/**
 * 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:
    vector<int> findMode(TreeNode* root) {
        vector <int> ans;
        int cnt = 0;
        int maxCnt = 0;

        inorder(root,cnt,maxCnt,ans);
        return ans;
    }
private:
    TreeNode* pre = NULL;
    void inorder(TreeNode* now,int& cnt,int& maxCnt,vector <int>& ans){
        if(now == NULL) return ;

        inorder(now->left,cnt,maxCnt,ans);
        upd(now,cnt,maxCnt,ans);
        inorder(now->right,cnt,maxCnt,ans);
        return ;
    }

    void upd(TreeNode* now,int& cnt,int& maxCnt,vector <int>& ans){
        if(pre && pre->val == now->val) ++cnt;
        else cnt=1;

        if(cnt > maxCnt){
            maxCnt = cnt;
            ans = {now->val};
        }
        else if(cnt == maxCnt){
            ans.push_back(now->val);
        }
    }

    pre = now;
    return ;
}

```

```
    }  
};
```

Lee X | Accepted X

← All Submissions



Accepted 24 / 24 testcases passed

ltheng submitted at Oct 29, 2025 19:04



Solution

Runtime



0 ms | Beats 100.00%

Analyze Complexity

Memory

25.22 MB | Beats 67.39%



Code | C++

```
/**
```

T6 LeetCode106 中等

```
/*
 * 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* buildTree(vector<int>& inorder, vector<int>& postorder) {
        unordered_map<int, int> inToIndex;

        for (int i=0;i<inorder.size();++i) inToIndex[inorder[i]] = i;

        return build(inorder, 0, inorder.size()-1, postorder, 0, postorder.size()-1, inToIndex);
    }

private:
    TreeNode* build(const vector<int>& inorder, int inStart, int inEnd, const vector<int>& postorder, int postStart, int postEnd, const unordered_map<int, int>& inToIndex){
        if (inStart > inEnd) return NULL;

        const int rootVal = postorder[postEnd];
        const int rootInIndex = inToIndex.at(rootVal);
        const int leftSize = rootInIndex - inStart;

        TreeNode* root = new TreeNode(rootVal);
        root->left = build(inorder, inStart, rootInIndex - 1, postorder, postStart,
                            postStart + leftSize - 1, inToIndex);
        root->right = build(inorder, rootInIndex + 1, inEnd, postorder,
                             postStart + leftSize, postEnd - 1, inToIndex);
        return root;
    }
};
```

Lee X

Accepted X



← All Submissions



Accepted 202 / 202 testcases passed

Itheng submitted at Oct 29, 2025 19:40



Solution

⌚ Runtime



1 ms | Beats 81.20% 🎉

⭐ Analyze Complexity

🐏 Memory

27.73 MB | Beats 11.31%

⭐ Analyze Complexity

30%

20%

10%

0%

10ms

20ms



Code | C++

/**

T7 LeetCode307 困难

```

class SegmentTree {
public:
    explicit SegmentTree(vector<int>& nums) : n(nums.size()), tree(n * 4) {
        if (n > 0) build(nums, 0, 0, n - 1);
    }

    void update(int index, int val) {
        myUpdate(0, 0, n - 1, index, val);
    }

    int sumRange(int left, int right) {
        return query(0, 0, n - 1, left, right);
    }

private:
    int n;
    vector<int> tree;

    void build(vector<int>& nums, int indTree, int low, int high) {
        if (low == high) {
            tree[indTree] = nums[low];
            return;
        }
        int mid = (low + high) / 2;
        build(nums, 2 * indTree + 1, low, mid);
        build(nums, 2 * indTree + 2, mid + 1, high);
        tree[indTree] = merge(tree[2 * indTree + 1], tree[2 * indTree + 2]);
    }

    void myUpdate(int indTree, int low, int high, int i, int val) {
        if (low == high) {
            tree[indTree] = val;
            return;
        }
        int mid = (low + high) / 2;
        if (i <= mid)
            myUpdate(2 * indTree + 1, low, mid, i, val);
        else
            myUpdate(2 * indTree + 2, mid + 1, high, i, val);

        tree[indTree] = merge(tree[2 * indTree + 1], tree[2 * indTree + 2]);
    }

    int query(int indTree, int low, int high, int i, int j) {
        if (i <= low && j >= high) return tree[indTree];
        if (j < low || i > high) return 0;
        int mid = (low + high) / 2;

```

```

        int leftSum = query(2 * indTree + 1, low, mid, i, j);
        int rightSum = query(2 * indTree + 2, mid + 1, high, i, j);
        return merge(leftSum, rightSum);
    }

    int merge(int left, int right) {
        return left + right;
    }
};

class NumArray {
public:
    NumArray(vector<int>& nums) : tree(nums) {}

    void update(int index, int val) {
        tree.update(index, val);
    }

    int sumRange(int left, int right) {
        return tree.sumRange(left, right);
    }

private:
    SegmentTree tree;
};

```

 Lee X |  Accepted X

← All Submissions 

Accepted 16 / 16 testcases passed

 lth... submitted at Oct 29, 2025 20:01



 Solution

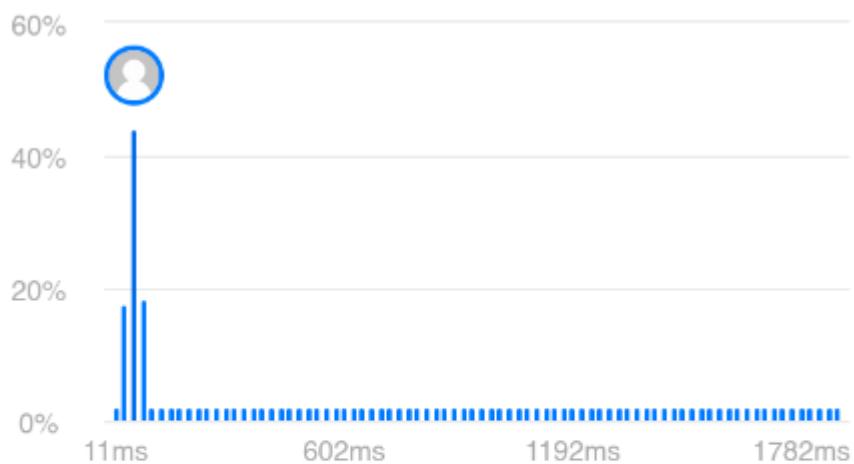
 Runtime 

79 ms | Beats **27.18%**

 Analyze Complexity

 Memory

182.42 MB | Beats **27.66%**



Code | C++

 class SegmentTree {

T8 LeetCode327 困难

```

class Fenwick {
public:
    explicit Fenwick(int n) : bit(n + 1, 0) {}

    void update(int index, int delta) {
        for (++index; index < (int)bit.size(); index += index & -index)
            bit[index] += delta;
    }

    int query(int index) const {
        int sum = 0;
        for (++index; index > 0; index -= index & -index)
            sum += bit[index];
        return sum;
    }

    int rangeQuery(int left, int right) const {
        if (left > right) return 0;
        return query(right) - (left == 0 ? 0 : query(left - 1));
    }
}

private:
    vector<int> bit;
};

class Solution {
public:
    int countRangeSum(vector<int>& nums, int lower, int upper) {
        const int n = nums.size();
        vector<long long> prefix(n + 1, 0);

        for (int i = 0; i < n; ++i)
            prefix[i + 1] = prefix[i] + nums[i];

        // 离散化所有前缀和及其偏移量
        vector<long long> allVals;
        allVals.reserve(prefix.size() * 3);
        for (auto s : prefix) {
            allVals.push_back(s);
            allVals.push_back(s - lower);
            allVals.push_back(s - upper);
        }

        sort(allVals.begin(), allVals.end());
        allVals.erase(unique(allVals.begin(), allVals.end()), allVals.end());

        auto getIndex = [&](long long val) {

```

```
        return int(lower_bound(allVals.begin(), allVals.end(), val) -  
allVals.begin());  
    };  
  
Fenwick bit(allVals.size());  
int ans = 0;  
  
for (auto s : prefix) {  
    int left = getIndex(s - upper);  
    int right = getIndex(s - lower);  
    ans += bit.rangeQuery(left, right);  
    bit.update(getIndex(s), 1);  
}  
  
return ans;  
}  
};
```

Lee X | Accepted X



← All Submissions



Accepted 67 / 67 testcases passed

Itheng submitted at Oct 29, 2025 20:07

Solution

Runtime

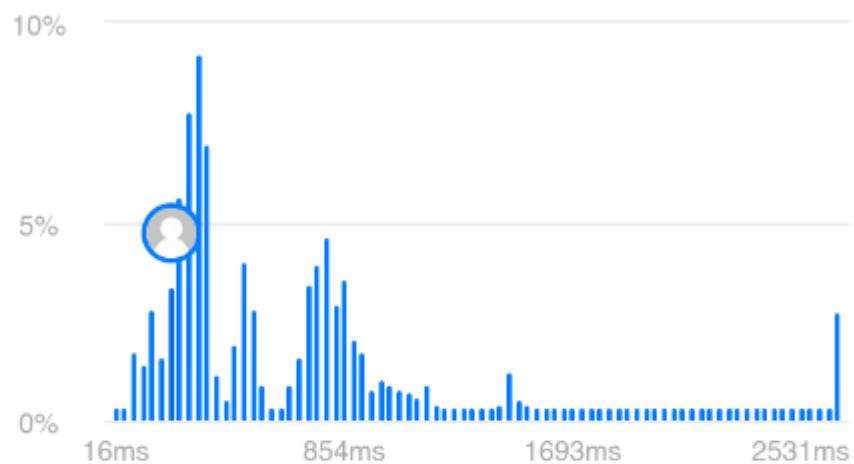


221 ms | Beats **90.76%**

Analyze Complexity

Memory

88.23 MB | Beats **90.12%**



16ms 854ms 1693ms 2531ms

Code | C++

class Fenwick {