Assignment #8: 树为主

Updated 1704 GMT+8 Apr 8, 2025

2025 spring, Complied by 颜鼎堃 工学院

> **说明:**
>
> 1. **解题与记录:**
>
>   对于每一个题目，请提供其解题思路（可选），并附上使用Python或C++编写的源代码（确保已在OpenJudge，Codeforces，LeetCode等平台上获得Accepted）。请将这些信息连同显示"Accepted"的截图一起填写到下方的作业模板中。（推荐使用Typora https://typoraio.cn 进行编辑，当然你也可以选择Word。）无论题目是否已通过，请标明每个题目大致花费的时间。
>
> 2. **提交安排:** 提交时，请首先上传PDF格式的文件，并将.md或.doc格式的文件作为附件上传至右侧的"作业评论"区。确保你的Canvas账户有一个清晰可见的头像，提交的文件为PDF格式，并且"作业评论"区包含上传的.md或.doc附件。
>
> 3. **延迟提交:** 如果你预计无法在截止日期前提交作业，请提前告知具体原因。这有助于我们了解情况并可能为你提供适当的延期或其他帮助。
>
> 请按照上述指导认真准备和提交作业，以保证顺利完成课程要求。

1. 题目

LC108.将有序数组转换为二叉树

dfs, https://leetcode.cn/problems/convert-sorted-array-to-binary-search-tree/

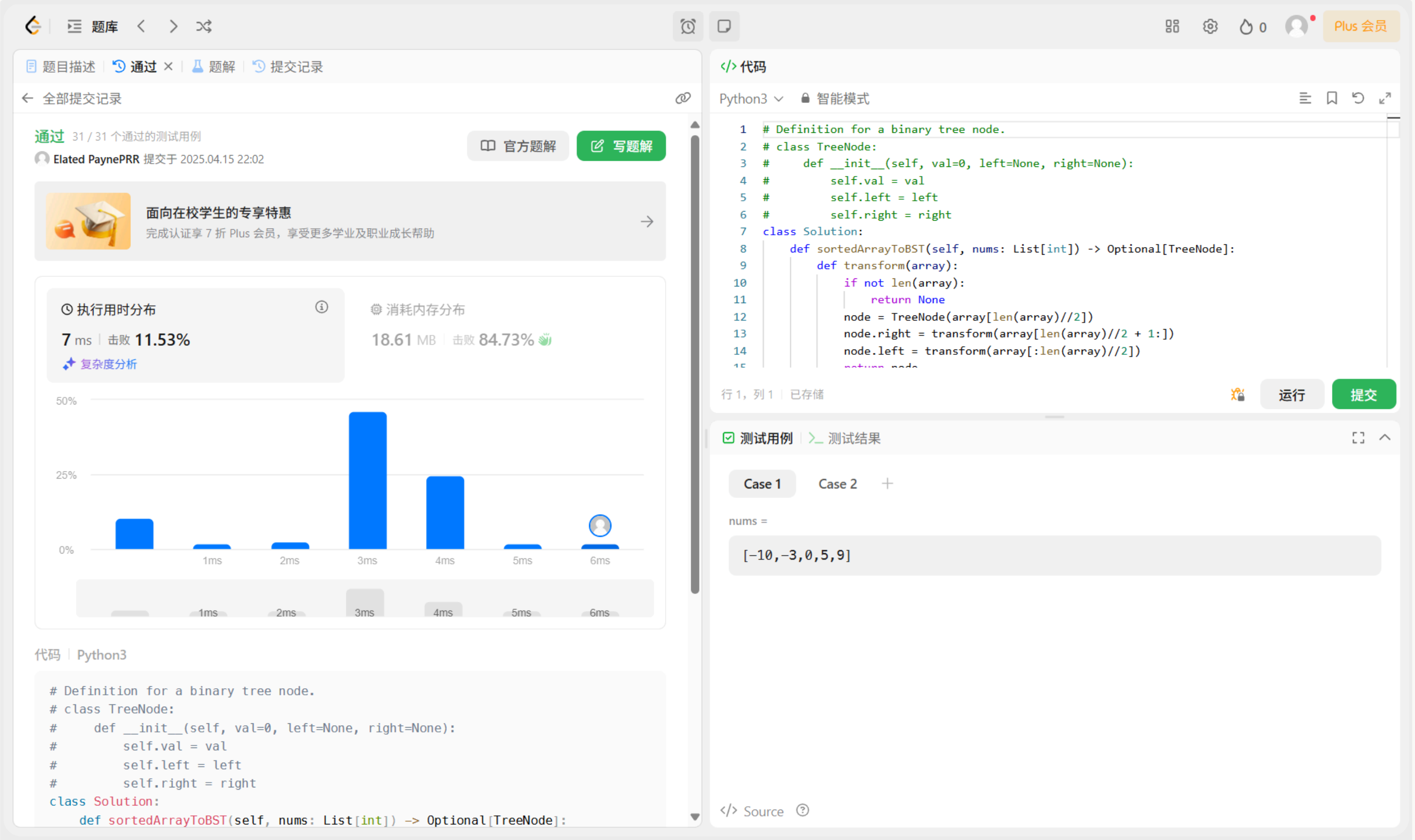
思路:

- 递归

代码:

```python
from typing import *
# 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 sortedArrayToBST(self, nums: List[int]) -> Optional[TreeNode]:
        def transform(array):
            if not len(array):
                return None
            node = TreeNode(array[len(array)//2])
            node.right = transform(array[len(array)//2 + 1:])
            node.left = transform(array[:len(array)//2])
            return node
        return transform(nums)


if __name__ == "__main__":
    sol = Solution()
    print(sol.sortedArrayToBST([-10, -3, 0, 5, 9]))

```

代码运行截图 （至少包含有"Accepted"）



通过 31 / 31 个通过的测试用例

Elated PaynePRR 提交于 2025.04.15 22:02

面向在校学生的专享特惠

完成认证享 7 折 Plus 会员，享受更多学业及职业成长帮助

执行用时分布

7 ms | 击败 11.53%

复杂度分析

消耗内存分布

18.61 MB | 击败 84.73%

代码 | Python3

```python
# 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 sortedArrayToBST(self, nums: List[int]) -> Optional[TreeNode]:
```

代码

```python
# 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 sortedArrayToBST(self, nums: List[int]) -> Optional[TreeNode]:
        def transform(array):
            if not len(array):
                return None
            node = TreeNode(array[len(array)//2])
            node.right = transform(array[len(array)//2 + 1:])
            node.left = transform(array[:len(array)//2])
```

行 1，列 1 | 已存储

nums =

```
[-10,-3,0,5,9]
```

LC-108

M27928:遍历树

adjacency list, dfs, http://cs101.openjudge.cn/practice/27928/

思路：

- 字典套树
- 这个题我调试了很久很久，有时候我甚至怀疑是python出bug了，最后发现似乎是如果把类的`__init__`方法写成这样

```python
class TreeNode:
    def __init__(self, val, subnode=[]):
        self.val = val
        self.subnode = subnode
```

就会导致所有`TreeNode`对象共用同一个`subnode`属性

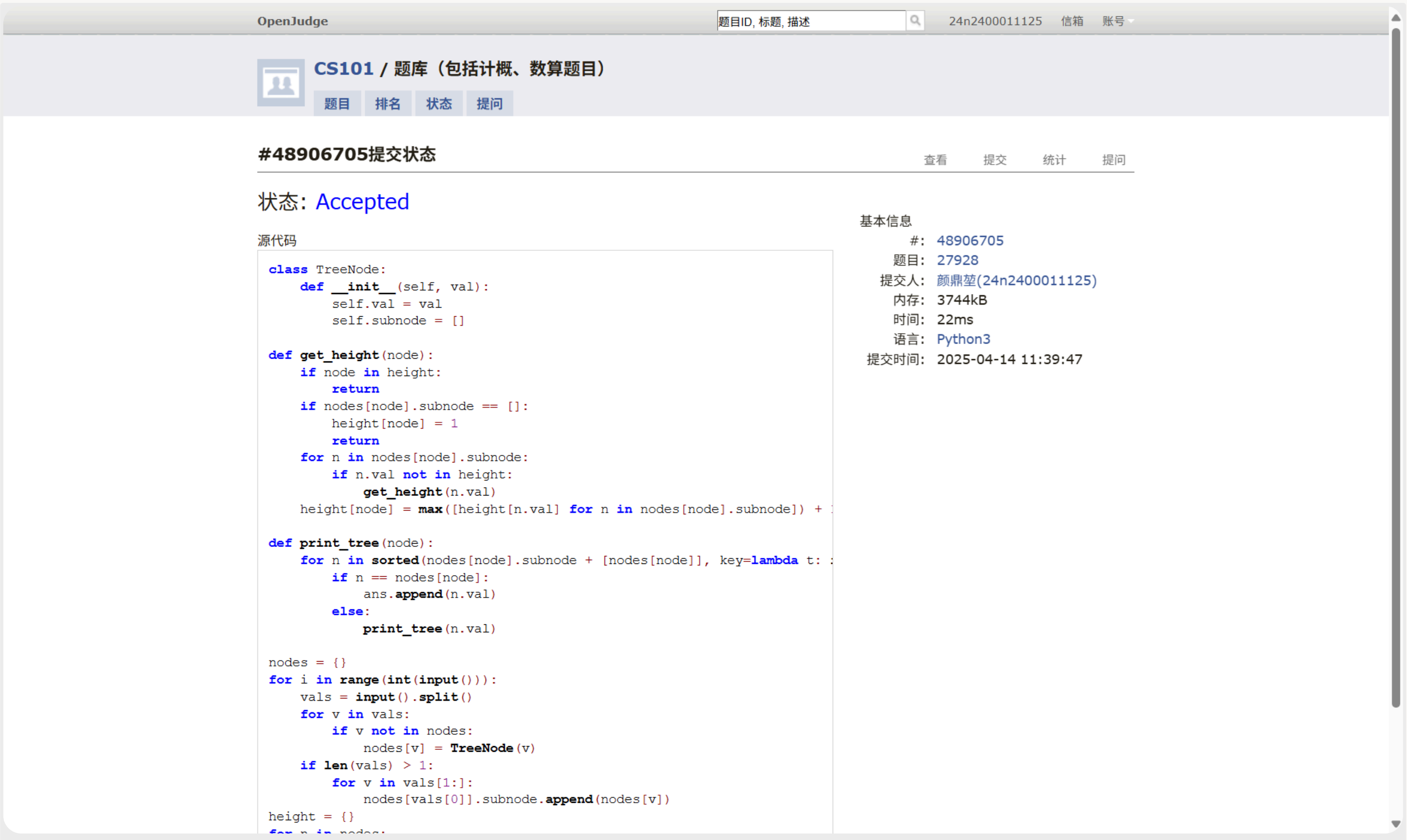
- 我大受震撼，以后坚决不在定义类时使用可变对象作为默认参数
- 最后找根节点单独写了个函数

  代码：

```python
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.subnode = []

def get_height(node):
    if node in height:
        return
    if nodes[node].subnode == []:
        height[node] = 1
        return
    for n in nodes[node].subnode:
        if n.val not in height:
            get_height(n.val)
    height[node] = max([height[n.val] for n in nodes[node].subnode]) + 1
```

```python

def print_tree(node):
    for n in sorted(nodes[node].subnode + [nodes[node]], key=lambda t: int(t.val)):
        if n == nodes[node]:
            ans.append(n.val)
        else:
            print_tree(n.val)

nodes = {}
for i in range(int(input())):
    vals = input().split()
    for v in vals:
        if v not in nodes:
            nodes[v] = TreeNode(v)
    if len(vals) > 1:
        for v in vals[1:]:
            nodes[vals[0]].subnode.append(nodes[v])
height = {}
for n in nodes:
    get_height(n)
ans = []
print_tree(max(height, key=lambda t: height[t]))
print(*ans, sep="\n")
```

代码运行截图（至少包含有"Accepted"）



#48906705提交状态

状态: Accepted

源代码

```python
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.subnode = []

def get_height(node):
    if node in height:
        return
    if nodes[node].subnode == []:
        height[node] = 1
        return
    for n in nodes[node].subnode:
        if n.val not in height:
            get_height(n.val)
    height[node] = max([height[n.val] for n in nodes[node].subnode]) +

def print_tree(node):
    for n in sorted(nodes[node].subnode + [nodes[node]], key=lambda t:
        if n == nodes[node]:
            ans.append(n.val)
        else:
            print_tree(n.val)

nodes = {}
for i in range(int(input())):
    vals = input().split()
    for v in vals:
        if v not in nodes:
            nodes[v] = TreeNode(v)
    if len(vals) > 1:
        for v in vals[1:]:
            nodes[vals[0]].subnode.append(nodes[v])
height = {}
```

基本信息

#:	48906705
题目:	27928
提交人:	颜鼎堃(24n2400011125)
内存:	3744kB
时间:	22ms
语言:	Python3
提交时间:	2025-04-14 11:39:47

OJ-27928

LC129.求根节点到叶节点数字之和

dfs, https://leetcode.cn/problems/sum-root-to-leaf-numbers/

思路：

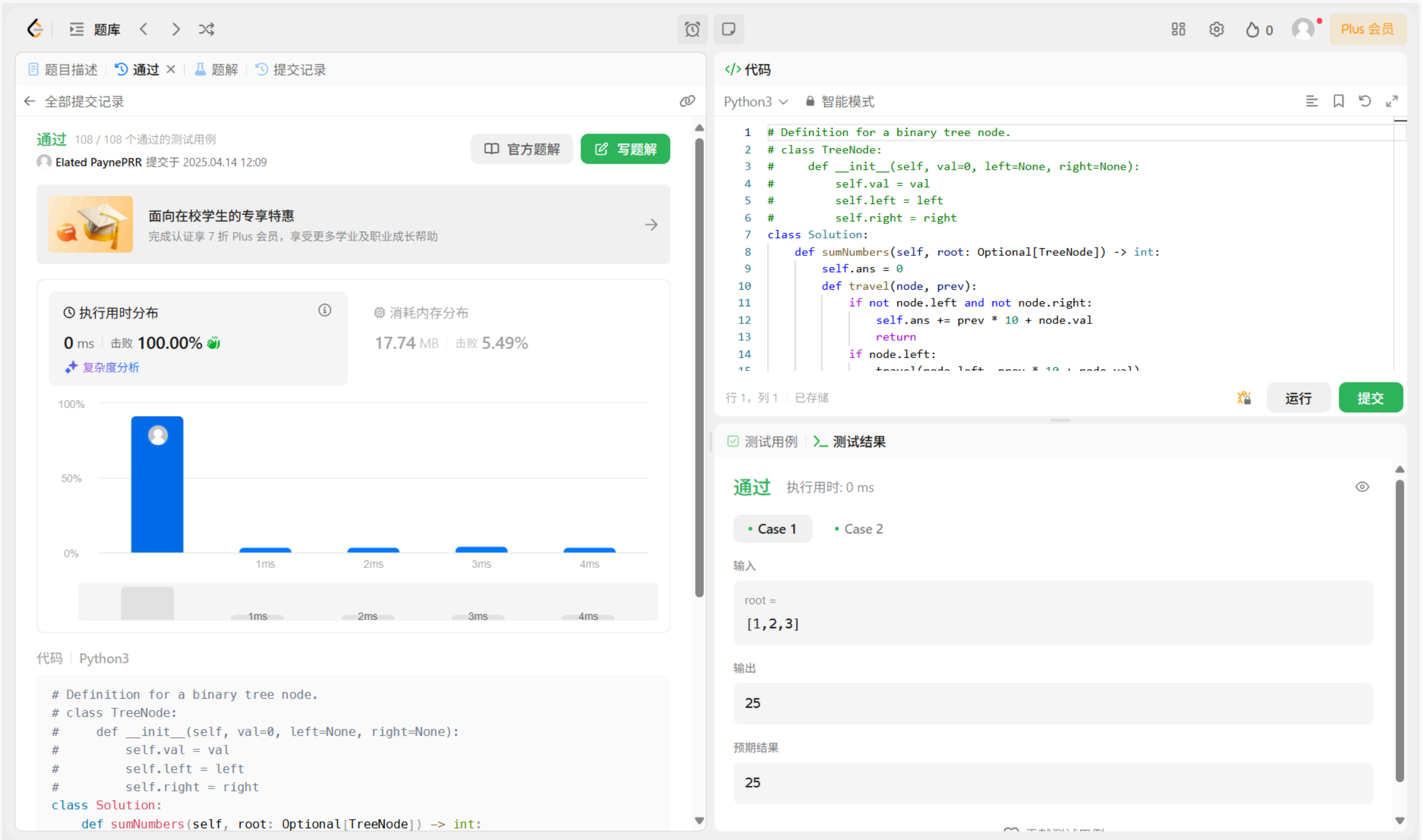
- 遍历，深搜

代码：

```python
from typing import *
# Definition for a binary tree node.
```

```python
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
class Solution:
    def sumNumbers(self, root: Optional[TreeNode]) -> int:
        self.ans = 0
        def travel(node, prev):
            if not node.left and not node.right:
                self.ans += prev * 10 + node.val
                return
            if node.left:
                travel(node.left, prev * 10 + node.val)
            if node.right:
                travel(node.right, prev * 10 + node.val)
        travel(root, 0)
        return self.ans

if __name__ == "__main__":
    sol = Solution()
    print(sol.sumNumbers(TreeNode(4, TreeNode(9, TreeNode(5), TreeNode(1)), TreeNode(0))))

```

代码运行截图 （至少包含有"Accepted"）



通过 108 / 108 个通过的测试用例

Elated PaynePRR 提交于 2025.04.14 12:09

执行用时分布 0 ms | 击败 100.00%

消耗内存分布 17.74 MB | 击败 5.49%

代码 | Python3

```python
# 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 sumNumbers(self, root: Optional[TreeNode]) -> int:
```

```python
# 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 sumNumbers(self, root: Optional[TreeNode]) -> int:
        self.ans = 0
        def travel(node, prev):
            if not node.left and not node.right:
                self.ans += prev * 10 + node.val
                return
            if node.left:
```

行 1，列 1 | 已存储

通过 执行用时: 0 ms

Case 1 Case 2

输入

```
root =
[1,2,3]
```

输出

```
25
```

预期结果

```
25
```

LC-129

M22158:根据二叉树前中序序列建树

tree, http://cs101.openjudge.cn/practice/22158/

思路：

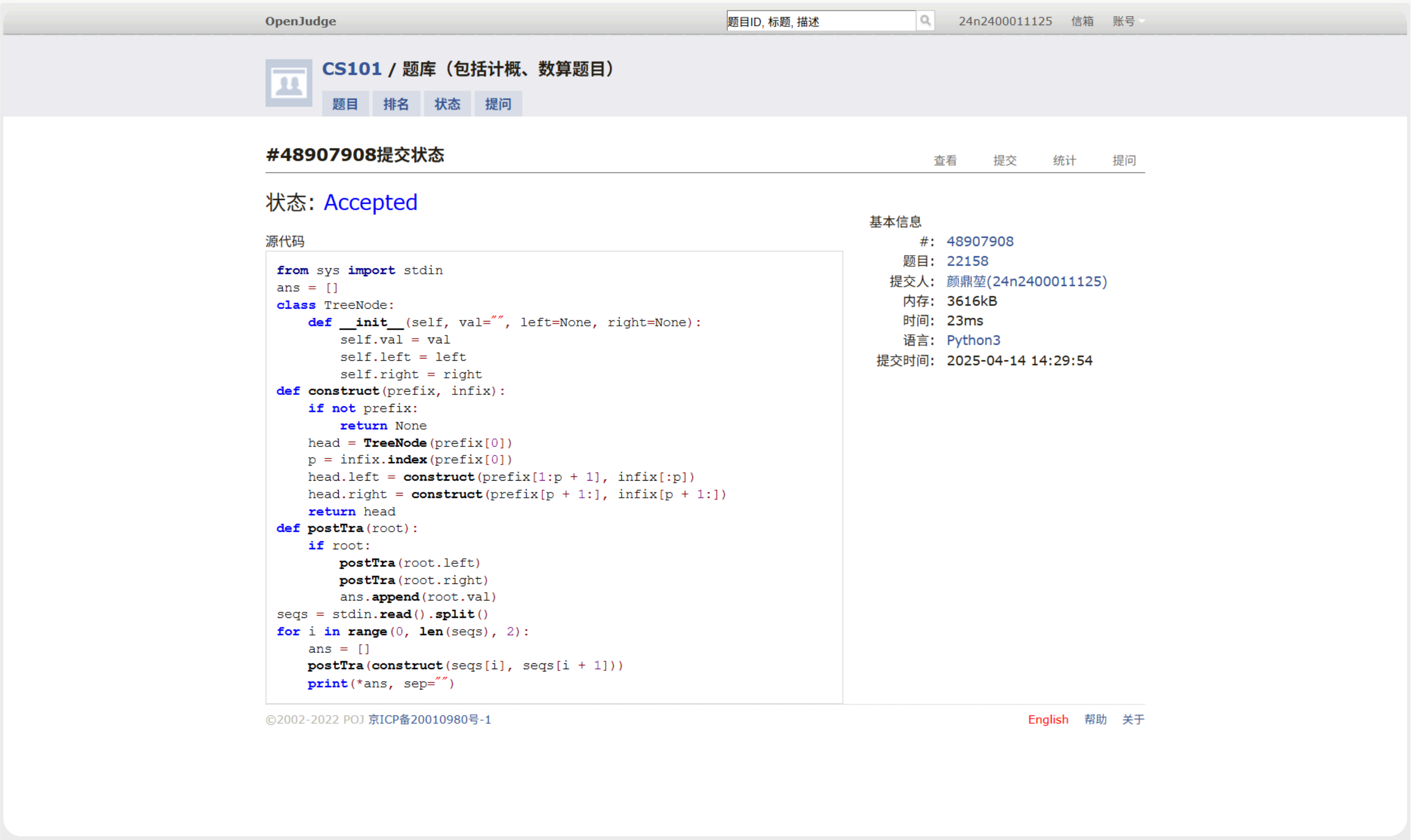
- 前序找树根，中序分左右

代码：

```python
from sys import stdin
ans = []
```

```python
class TreeNode:
    def __init__(self, val="", left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
def construct(prefix, infix):
    if not prefix:
        return None
    head = TreeNode(prefix[0])
    p = infix.index(prefix[0])
    head.left = construct(prefix[1:p + 1], infix[:p])
    head.right = construct(prefix[p + 1:], infix[p + 1:])
    return head
def postTra(root):
    if root:
        postTra(root.left)
        postTra(root.right)
        ans.append(root.val)
seqs = stdin.read().split()
for i in range(0, len(seqs), 2):
    ans = []
    postTra(construct(seqs[i], seqs[i + 1]))
    print(*ans, sep="")
```

代码运行截图 （至少包含有"Accepted"）


OpenJudge 题目ID, 标题, 描述 24n2400011125 信箱 账号

CS101 / 题库（包括计概、数算题目）

题目 排名 状态 提问

#48907908提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```python
from sys import stdin
ans = []
class TreeNode:
    def __init__(self, val="", left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
def construct(prefix, infix):
    if not prefix:
        return None
    head = TreeNode(prefix[0])
    p = infix.index(prefix[0])
    head.left = construct(prefix[1:p + 1], infix[:p])
    head.right = construct(prefix[p + 1:], infix[p + 1:])
    return head
def postTra(root):
    if root:
        postTra(root.left)
        postTra(root.right)
        ans.append(root.val)
seqs = stdin.read().split()
for i in range(0, len(seqs), 2):
    ans = []
    postTra(construct(seqs[i], seqs[i + 1]))
    print(*ans, sep="")
```

基本信息

#:	48907908
题目:	22158
提交人:	颜鼎堃(24n2400011125)
内存:	3616kB
时间:	23ms
语言:	Python3
提交时间:	2025-04-14 14:29:54

©2002-2022 POJ 京ICP备20010980号-1 English 帮助 关于

OJ-22158

M24729:括号嵌套树

dfs, stack, http://cs101.openjudge.cn/practice/24729/

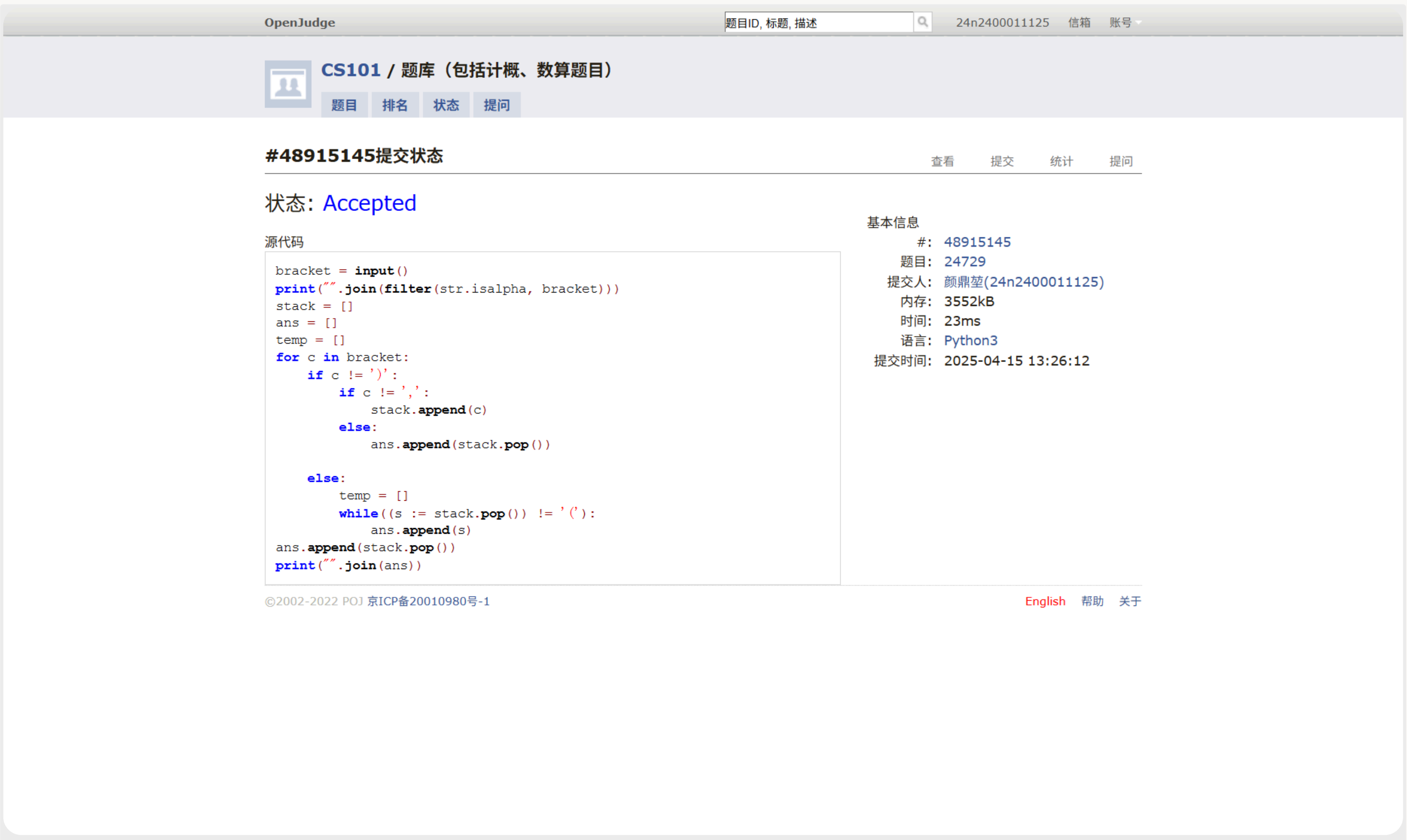
思路：

- 选择不建树直接出结果
- 前序只需要去掉括号
- 后序麻烦一些，要注意类似 `A(B,C(D,E(F,G),H(I,J,K)))` 这种前面节点是叶子结点的情况

代码：

```python
bracket = input()
print("".join(filter(str.isalpha, bracket)))
stack = []
ans = []
temp = []
for c in bracket:
    if c ≠ ')':
        if c ≠ ',':
            stack.append(c)
        else:
            ans.append(stack.pop())

    else:
        temp = []
        while((s := stack.pop()) ≠ '('):
            ans.append(s)
ans.append(stack.pop())
print("".join(ans))
```

代码运行截图 （至少包含有"Accepted"）


OpenJudge 题目ID, 标题, 描述 24n2400011125 信箱 账号

CS101 / 题库（包括计概、数算题目）

题目 排名 状态 提问

#48915145提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
bracket = input()
print("".join(filter(str.isalpha, bracket)))
stack = []
ans = []
temp = []
for c in bracket:
    if c != ')':
        if c != ',':
            stack.append(c)
        else:
            ans.append(stack.pop())

    else:
        temp = []
        while((s := stack.pop()) != '('):
            ans.append(s)
ans.append(stack.pop())
print("".join(ans))
```

基本信息

#:	48915145
题目:	24729
提交人:	颜鼎堃(24n2400011125)
内存:	3552kB
时间:	23ms
语言:	Python3
提交时间:	2025-04-15 13:26:12

©2002-2022 POJ 京ICP备20010980号-1 English 帮助 关于

OJ-24729

LC3510.移除最小数对使数组有序II

doubly-linked list + heap, https://leetcode.cn/problems/minimum-pair-removal-to-sort-array-ii/

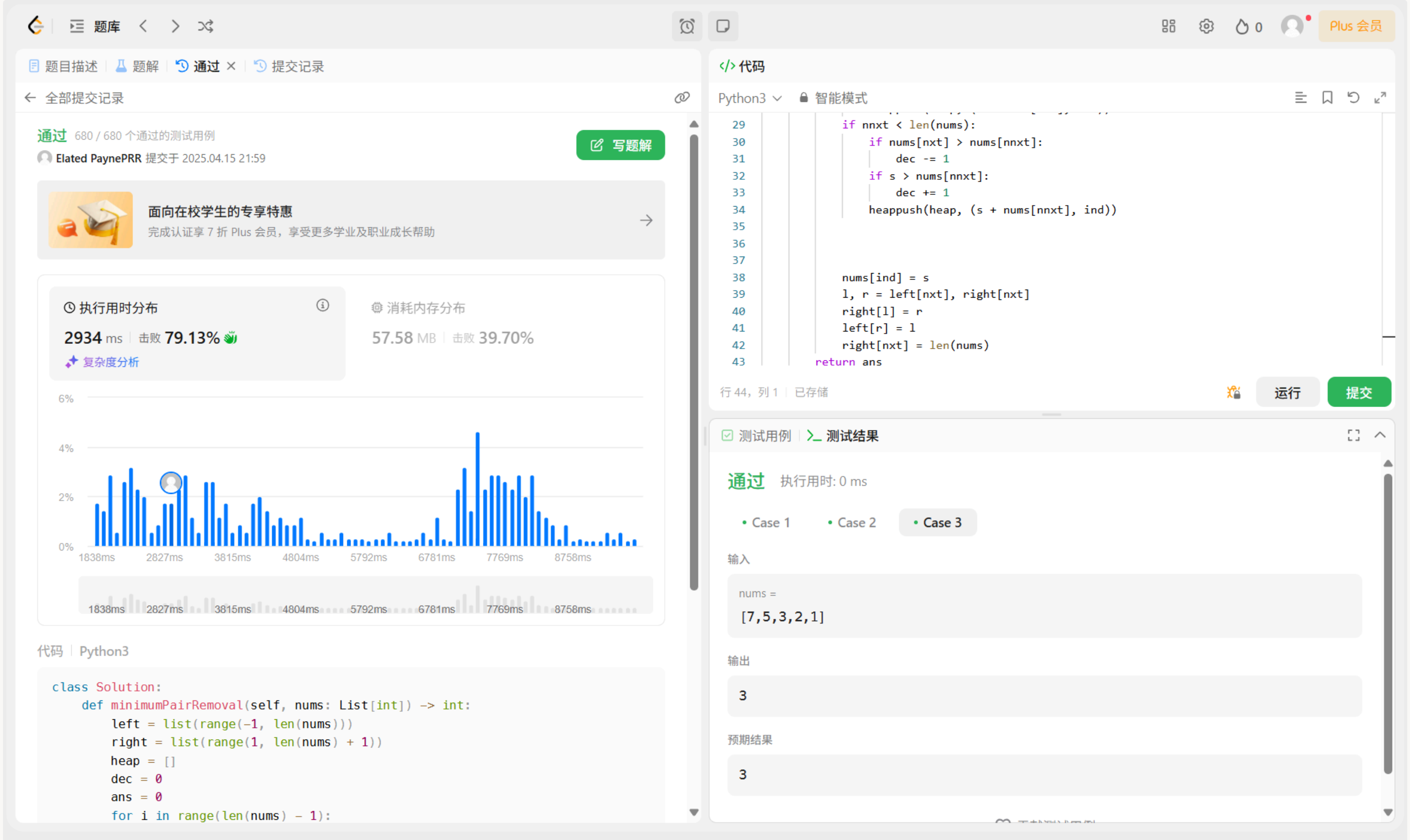
思路:

- 不会写
- 学到了一些东西，比如用并查集表示删除元素
- 抄代码都抄错好几次，还要找半天哪抄错了

代码:

```python
from typing import *
from heapq import heappush, heappop
class Solution:
    def minimumPairRemoval(self, nums: List[int]) -> int:
        left = list(range(-1, len(nums)))
        right = list(range(1, len(nums) + 1))
        heap = []
        dec = 0
        ans = 0
        for i in range(len(nums) - 1):
            if nums[i] > nums[i + 1]:
                dec += 1
            heappush(heap, (nums[i] + nums[i+1], i))
        while dec:
            s, ind = heappop(heap)
            if right[ind] >= len(nums) or s != nums[ind] + nums[right[ind]]:
                continue
            ans += 1
            lst = left[ind]
            nxt = right[ind]
            nnxt = right[nxt]
            if nums[ind] > nums[right[ind]]:
                dec -= 1

            if lst >= 0:
                if nums[lst] > nums[ind]:
                    dec -= 1
                if s < nums[lst]:
                    dec += 1
                heappush(heap, (s + nums[lst], lst))
            if nnxt < len(nums):
                if nums[nxt] > nums[nnxt]:
                    dec -= 1
                if s > nums[nnxt]:
                    dec += 1
                heappush(heap, (s + nums[nnxt], ind))



            nums[ind] = s
            l, r = left[nxt], right[nxt]
            right[l] = r
            left[r] = l
            right[nxt] = len(nums)
        return ans


if __name__ == "__main__":
    sol = Solution()
    print(sol.minimumPairRemoval([7,5,3,2,1]))

```

代码运行截图 （至少包含有"Accepted"）



题库 | 题目描述 | 题解 | 通过 | 提交记录

全部提交记录

通过 680 / 680 个通过的测试用例

Elated PaynePRR 提交于 2025.04.15 21:59

写题解

面向在校学生的专享特惠
完成认证享 7 折 Plus 会员，享受更多学业及职业成长帮助

执行用时分布
2934 ms | 击败 79.13%
复杂度分析

消耗内存分布
57.58 MB | 击败 39.70%

代码 | Python3

```python
class Solution:
    def minimumPairRemoval(self, nums: List[int]) -> int:
        left = list(range(-1, len(nums)))
        right = list(range(1, len(nums) + 1))
        heap = []
        dec = 0
        ans = 0
        for i in range(len(nums) - 1):
```

代码 | Python3 | 智能模式

```python
            if nnxt < len(nums):
                if nums[nxt] > nums[nnxt]:
                    dec -= 1
                if s > nums[nnxt]:
                    dec += 1
                heappush(heap, (s + nums[nnxt], ind))



            nums[ind] = s
            l, r = left[nxt], right[nxt]
            right[l] = r
            left[r] = l
            right[nxt] = len(nums)
        return ans
```

行 44，列 1 | 已存储 | 运行 | 提交

测试用例 | 测试结果

通过 执行用时: 0 ms

Case 1 | Case 2 | Case 3

输入

nums =
[7,5,3,2,1]

输出

3

预期结果

3

LC-3510

2. 学习总结和收获

如果发现作业题目相对简单，有否寻找额外的练习题目，如“数算2025spring每日选做”、LeetCode、Codeforces、洛谷等网站上的题目。

这次作业耗时挺长的，有一些写法和特性还不太熟练

为了写Leetcode时偷懒，写了一个Sublime Text的插件

```python
import sublime, sublime_plugin
class LeetcodeAutoInsertCommand(sublime_plugin.TextCommand):
    def run(self, edit):
        end = self.view.find(r"class Solution:", 0).end()
        if end > 0:
            end += 10
            sol_name = self.view.substr(self.view.word(end))
            ifm = """\n\n
if __name__ == \"__main__\":
    sol = Solution()
    print(sol.""" + sol_name + """())
"""
        else:
            ifm = "if __name__ == \"__main__\"\n    "
        self.view.insert(edit, 0, "from typing import *\n")
        self.view.insert(edit, self.view.size(), ifm)
```

从而按特定快捷键的时候在文档开头插入`from typing import *`，在文档末尾插入

```python
if __name__ == "__main__":
    sol = Solution()
    print(sol.sol_name())
```

其中`sol_name`是`Solution()`里面的方法名称，比如最后一题就是`minimumPairRemoval`

为了适配部分实现类题目（比如上次作业实现LRU），在找不到`sol_name`时只插入`if __name__ == "__main__":`