Assignment #9: Huffman, BST & Heap

Updated 1834 GMT+8 Apr 15, 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

LC222.完全二叉树的节点个数

bfs, dfs, binary + greedy, https://leetcode.cn/problems/count-complete-tree-nodes/

如果用bfs写是简单级别，其他方法是中级难度。

思路：

- 就用dfs
- 因为只用统计最后一层中的空节点数n，若二叉树高为h，则答案为2^h-n-1

代码：

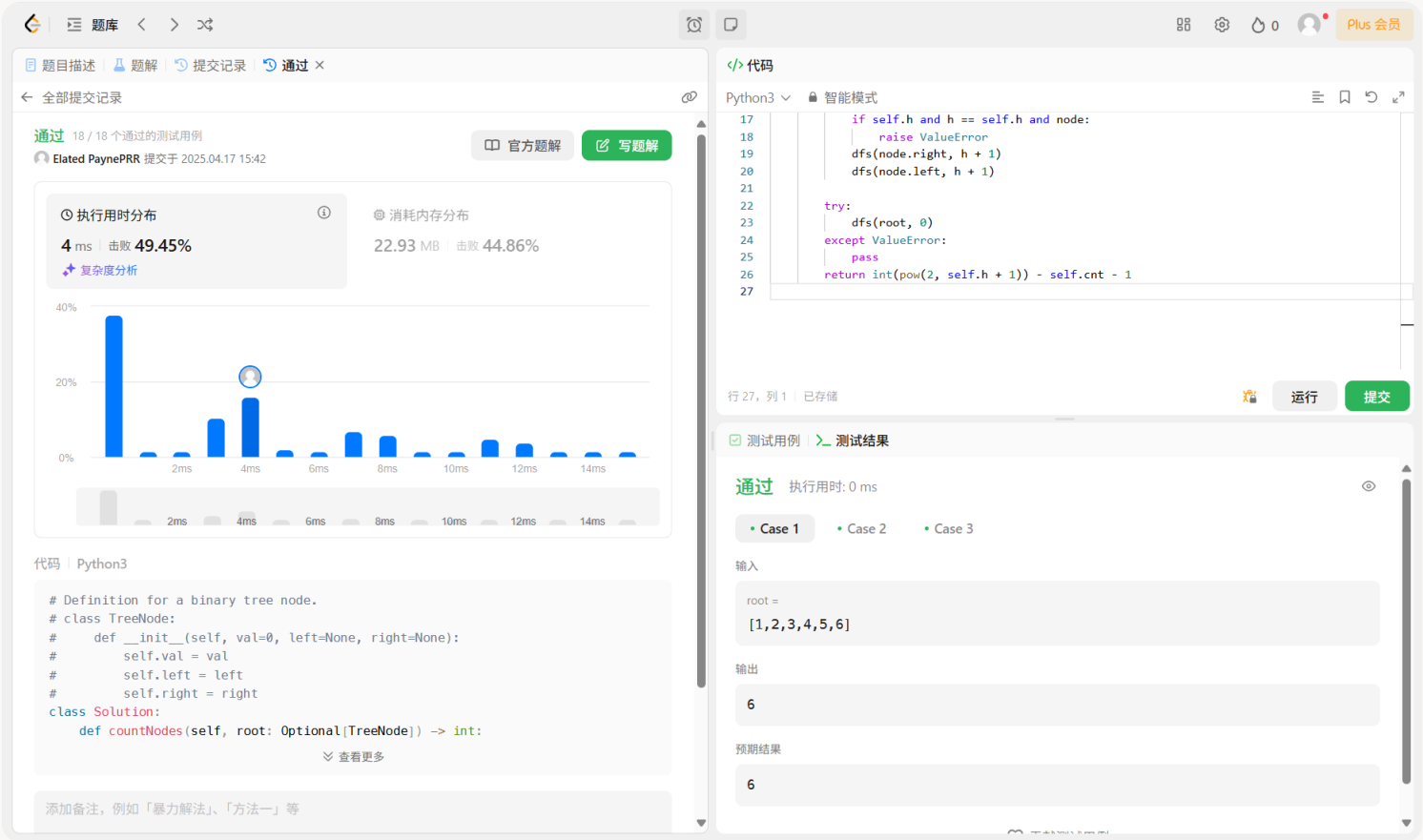
```python
from typing import *
from math import pow
# Definition for a binary tree node.
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
class Solution:
    def countNodes(self, root: Optional[TreeNode]) -> int:
        self.cnt = self.h = 0
        self.h = 0
        def dfs(node, h):
            if not node:
                self.cnt += 1
                if not self.h:
                    self.h = h
                return
            if self.h and h == self.h and node:
                raise ValueError
            dfs(node.right, h + 1)
            dfs(node.left, h + 1)
        try:
            dfs(root, 0)
        except ValueError:
            pass
```

```python
        return int(pow(2, self.h + 1)) - self.cnt - 1


if __name__ == "__main__":
    sol = Solution()
    print(sol.countNodes(TreeNode(1, TreeNode(2, TreeNode(4), TreeNode(5)), TreeNode(3, TreeNode(6)))))

```

代码运行截图 （至少包含有"Accepted"）



LC103.二叉树的锯齿形层序遍历

bfs, https://leetcode.cn/problems/binary-tree-zigzag-level-order-traversal/

思路：

- 可以考虑把 `queue` 反转
- 但为了偷懒，我选择把 `ans` 中特定层反转

代码：

Python

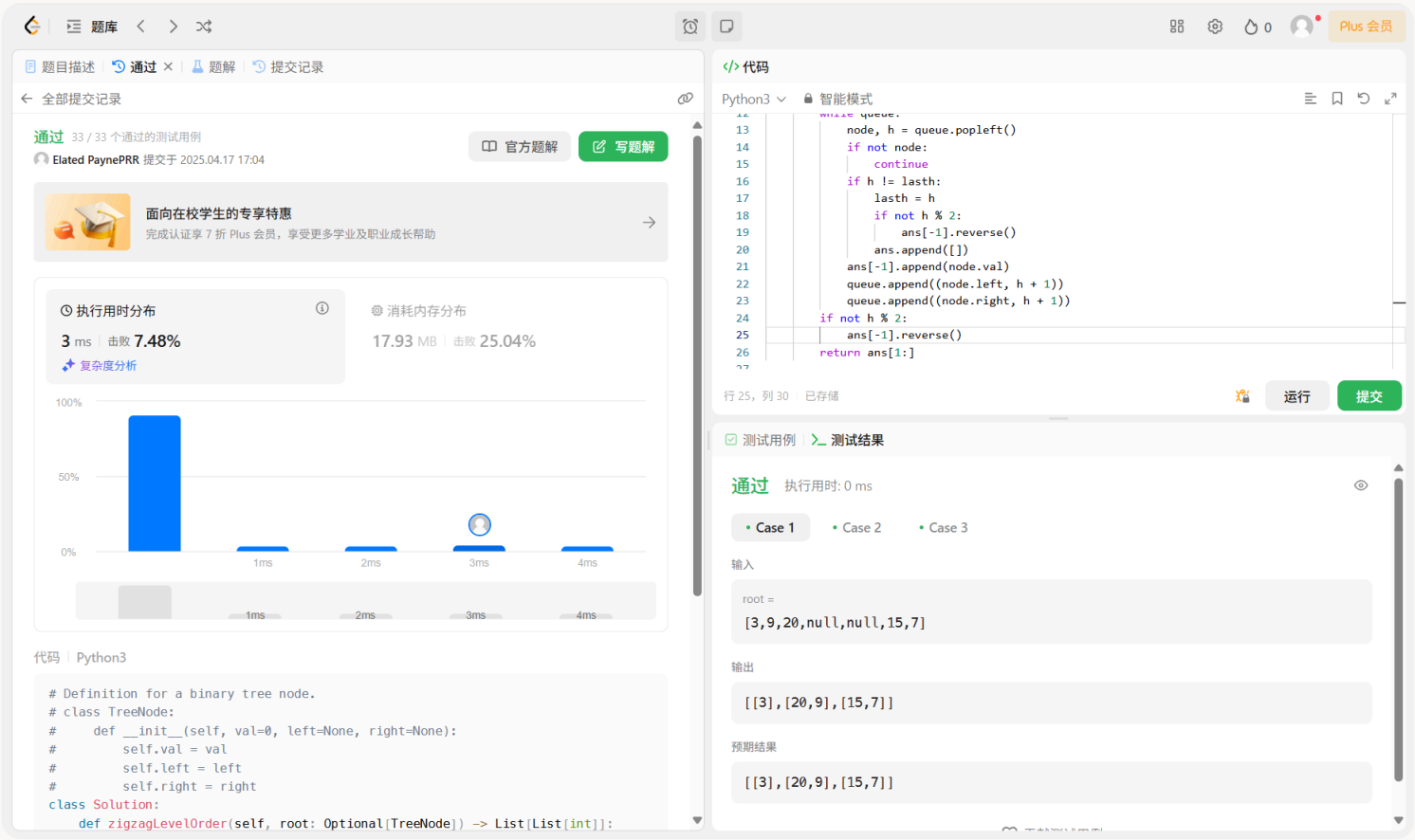
```python
from typing import *
from collections import deque
# Definition for a binary tree node.
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
class Solution:
    def zigzagLevelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:
        queue = deque([(root, 0)])
        ans = [[]]
        lasth = -1
        while queue:
            node, h = queue.popleft()
            if not node:
                continue
```

```python
            if h != lasth:
                lasth = h
                if not h % 2:
                    ans[-1].reverse()
                ans.append([])
            ans[-1].append(node.val)
            queue.append((node.left, h + 1))
            queue.append((node.right, h + 1))
        if not h % 2:
            ans[-1].reverse()
        return ans[1:]

if __name__ == "__main__":
    sol = Solution()
    print(sol.zigzagLevelOrder(TreeNode(3, TreeNode(9), TreeNode(20, TreeNode(15), TreeNode(7)))))

```

代码运行截图 （至少包含有"Accepted"）



M04080:Huffman编码树

greedy, http://cs101.openjudge.cn/practice/04080/

思路：

- 定义 `__lt__` 函数，方便直接塞进堆中

代码：

```python
from heapq import heappush, heappop
ans = 0
class TreeNode:
    def __init__(self, val, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
    def __lt__(self, other):
        return self.val < other.val

```

```python
def calcWeighedLength(node, h):
    if node and not node.left and not node.right:
        global ans
        ans += h * node.val
        return
    calcWeighedLength(node.left, h + 1)
    calcWeighedLength(node.right, h + 1)
heap = []
n = int(input())
weight = map(int, input().split())
for i in weight:
    heappush(heap, TreeNode(i))
while len(heap) ⩾ 2:
    node1 = heappop(heap)
    node2 = heappop(heap)
    node = TreeNode(node1.val + node2.val, node1, node2)
    heappush(heap, node)
calcWeighedLength(node, 0)
print(ans)

```

代码运行截图 （至少包含有"Accepted"）

M05455: 二叉搜索树的层次遍历

http://cs101.openjudge.cn/practice/05455/

思路：

- 为了去除数据中的重复，应当在构建树的过程中忽略重复元素而不是在读入时删去
- 原因是`list(set(input().split()))`无法保证与读入数据顺序相同

代码：

```python
from collections import deque
from sys import stdin
class TreeNode:
    def __init__(self, val, left=None, right=None):
        self.val = val
```

```
        self.left = left
        self.right = right

def insert(n, p):
    if n < p.val:
        if p.left:
            insert(n, p.left)
        else:
            p.left = TreeNode(n)
    elif n > p.val:
        if p.right:
            insert(n, p.right)
        else:
            p.right = TreeNode(n)

data = list(map(int, stdin.read().split()))
head = TreeNode(data[0])
for num in data[1:]:
    insert(num, head)
queue = deque([head])
ans = []
while queue:
    node = queue.popleft()
    if not node:
        continue
    ans.append(str(node.val))
    queue.append(node.left)
    queue.append(node.right)
print(*ans, sep=" ", end="")
```

代码运行截图 （至少包含有"Accepted"）

M04078: 实现堆结构

手搓实现， http://cs101.openjudge.cn/practice/04078/

类似的题目是 晴问9.7: 向下调整构建大顶堆， https://sunnywhy.com/sfbj/9/7

思路：

- `heappop`时向下渗透有些麻烦
- 最开始没有意识到，于是编了以下这组数据帮自己改错
- 输入

```unkown language
18
1 41
1 43
1 85
1 24
2
1 43
1 54
2
1 20
1 54
1 89
2
1 58
2
2
2
2
2
```

- 输出

```unkown language
24
41
20
43
43
54
54
58
```

代码：

```Python
class heap:
    def __init__(self):
        self.L = []
    def heappush(self, u):
        self.L.append(u)
        ind = len(self.L) - 1
        head = (ind - 1) // 2
        while ind and self.L[ind] < self.L[head]:
            oth = 4 * ((head - 1) // 2) - head + 3
            if head and self.L[head] < self.L[oth]:
                self.L[head], self.L[oth] = self.L[oth], self.L[head]
            self.L[ind], self.L[head] = self.L[head], self.L[ind]
            ind = head
            head = (ind - 1) // 2
    def heappop(self):
        ind = len(self.L) - 1
        path = []
        while ind:
            path.append(ind)
            ind = (ind - 1) // 2
        path.reverse()
        ind = 0
```

```
        for sub in path:
            oth = 4 * ind - sub + 3
            if oth < len(self.L) and self.L[sub] > self.L[oth]:
                self.L[sub], self.L[oth] = self.L[oth], self.L[sub]
                while True:
                    a = (2*oth + 2 < len(self.L) - 1) and self.L[oth] > self.L[2*oth + 2]
                    b = (2*oth + 1 < len(self.L) - 1) and self.L[oth] > self.L[2*oth + 1]
                    if a:
                        self.L[oth], self.L[2*oth + 2] = self.L[2*oth + 2], self.L[oth]
                        c = 2 * oth + 2
                    if b:
                        self.L[oth], self.L[2*oth + 1] = self.L[2*oth + 1], self.L[oth]
                        c = 2 * oth + 1
                    if a or b:
                        oth = c
                    else:
                        break

            self.L[ind], self.L[sub] = self.L[sub], self.L[ind]
            ind = sub
        return self.L.pop()

h = heap()
for i in range(int(input())):
    p = input().split()
    if len(p) == 1:
        print(h.heappop())
    else:
        h.heappush(int(p[1]))

```

代码运行截图 （至少包含有"Accepted"）

T22161: 哈夫曼编码树

greedy, http://cs101.openjudge.cn/practice/22161/

思路：

- 在第三题的基础上，再定义一个 `__add__` 函数方便构造子树

代码：

```python
from heapq import heappop, heappush
code = {}
decode = {}
heap = []
class TreeNode:
    def __init__(self, ch, freq, minch="", left=None, right=None):
        self.ch = ch
        self.freq = freq
        self.left = left
        self.right = right
        self.minch = list(ch)[0] if not minch else minch
    def __lt__(self, other):
        return (self.freq, self.minch) < (other.freq, other.minch)
    def __add__(self, other):
        return TreeNode(
            self.ch | other.ch,
            self.freq + other.freq,
            min(self.minch, other.minch),
            self if self < other else other,
            other if self < other else self
            )

def encode(node, prev):
    if not node:
        return
    if not node.left and not node.right:
        c = list(node.ch)[0]
        code[c] = prev
        decode[prev] = c
        return
    encode(node.left, prev + "0")
    encode(node.right, prev + "1")

def num_to_str(get):
    ans = ""
    lst = 0
    for i in range(1, len(get) + 1):
        if get[lst:i] in decode:
            ans += decode[get[lst:i]]
            lst = i
    return ans

def str_to_num(get):
    ans = ""
    for c in get:
        ans += code[c]
    return ans

for i in range(int(input())):
    c, f = input().split()
    heappush(heap, TreeNode({c}, int(f)))
while len(heap) > 1:
    n1 = heappop(heap)
    n2 = heappop(heap)
    node = n1 + n2
    heappush(heap, node)
head = node
encode(head, "")
```

```python
while True:
    try:
        get = input()
        if get[0] in {"0", "1"}:
            print(num_to_str(get))
        else:
            print(str_to_num(get))
    except EOFError:
        break
```

代码运行截图（至少包含有"Accepted"）

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。

这次作业难度不大但也耗时不短，主要原因是代码长度长以及对算法不够熟练