# Assignment #9: Huffman, BST & Heap
Updated 1834 GMT+8 Apr 15, 2025
2025 spring, Complied by 李振硕、信息管理系

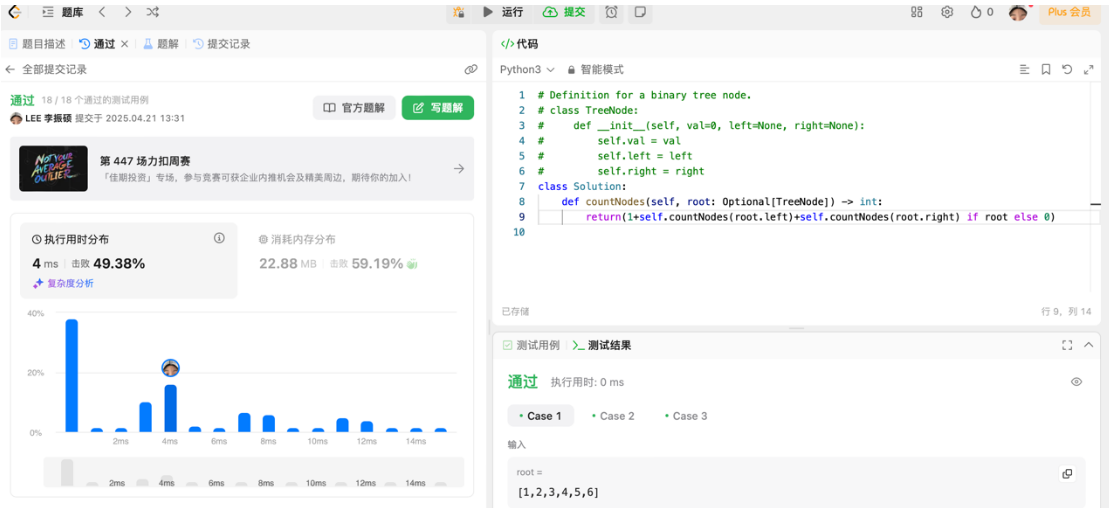
## 1. 题目

### LC222.完全二叉树的节点个数

dfs, https://leetcode.cn/problems/count-complete-tree-nodes/

思路：

代码：



通过 18 / 18 个通过的测试用例
LEE 李振硕 提交于 2025.04.21 13:31

第 447 场力扣周赛
「佳期投资」专场，参与竞赛可获企业内推机会及精美周边，期待你的加入！

4 ms | 击败 49.38%

22.88 MB | 击败 59.19%

```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 countNodes(self, root: Optional[TreeNode]) -> int:
        return(1+self.countNodes(root.left)+self.countNodes(root.right) if root else 0)
```

通过 执行用时: 0 ms

输入

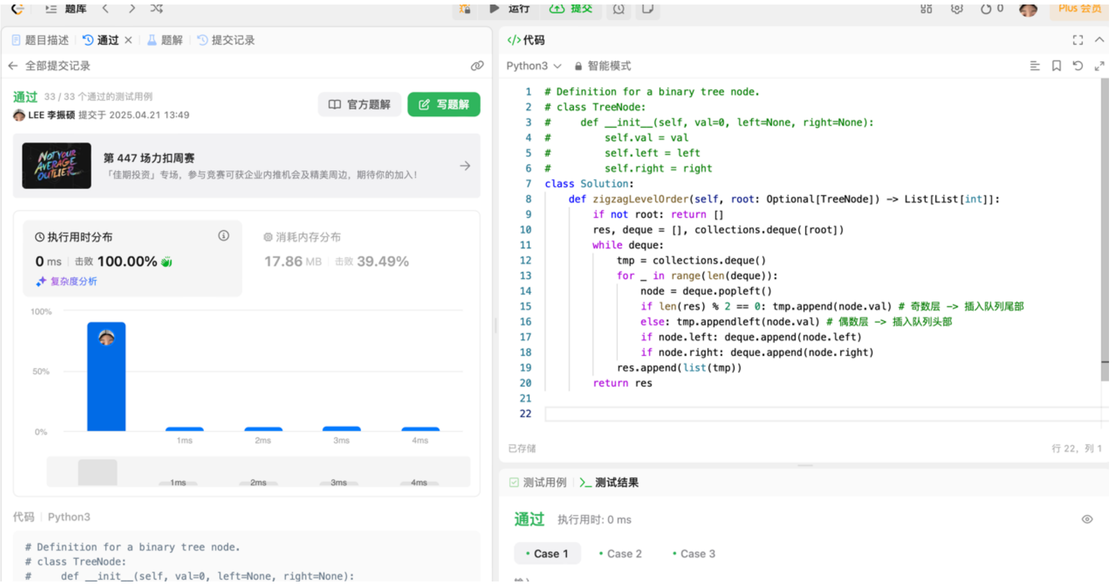
root =
[1,2,3,4,5,6]

### 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：

代码：



通过 33 / 33 个通过的测试用例
LEE 李振硕 提交于 2025.04.21 13:49

第 447 场力扣周赛
「佳期投资」专场，参与竞赛可获企业内推机会及精美周边，期待你的加入！

0 ms | 击败 100.00%

17.86 MB | 击败 39.49%

代码 | Python3

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
```

```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 zigzagLevelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:
        if not root: return []
        res, deque = [], collections.deque([root])
        while deque:
            tmp = collections.deque()
            for _ in range(len(deque)):
                node = deque.popleft()
                if len(res) % 2 == 0: tmp.append(node.val) # 奇数层 -> 插入队列尾部
                else: tmp.appendleft(node.val) # 偶数层 -> 插入队列头部
                if node.left: deque.append(node.left)
                if node.right: deque.append(node.right)
            res.append(list(tmp))
        return res

```

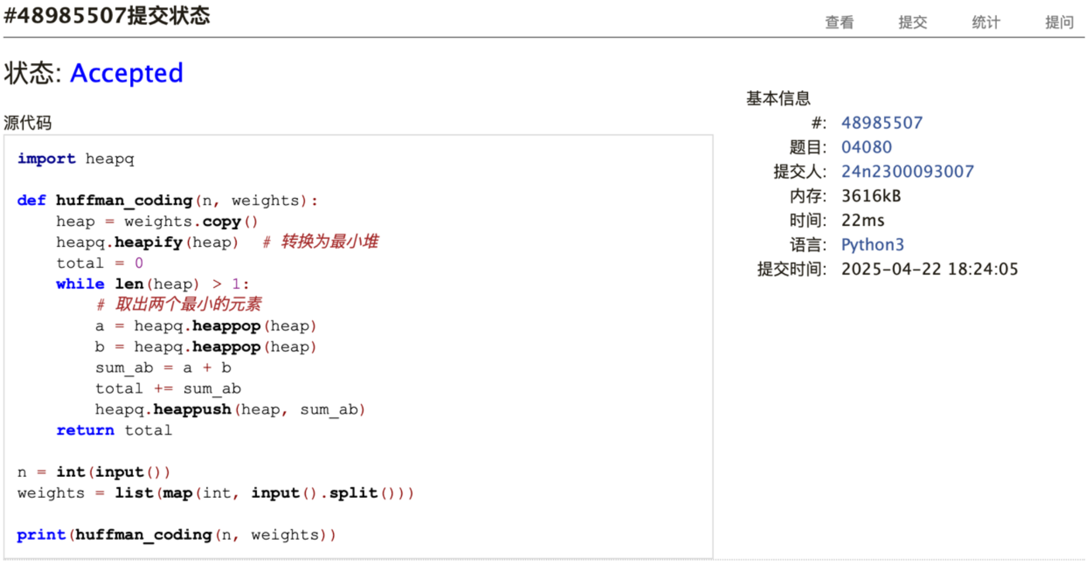
通过 执行用时: 0 ms

### M04080:Huffman 编码树

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

思路:

代码:



#48985507提交状态

状态: Accepted

源代码

```python
import heapq

def huffman_coding(n, weights):
    heap = weights.copy()
    heapq.heapify(heap)   # 转换为最小堆
    total = 0
    while len(heap) > 1:
        # 取出两个最小的元素
        a = heapq.heappop(heap)
        b = heapq.heappop(heap)
        sum_ab = a + b
        total += sum_ab
        heapq.heappush(heap, sum_ab)
    return total

n = int(input())
weights = list(map(int, input().split()))

print(huffman_coding(n, weights))
```

基本信息

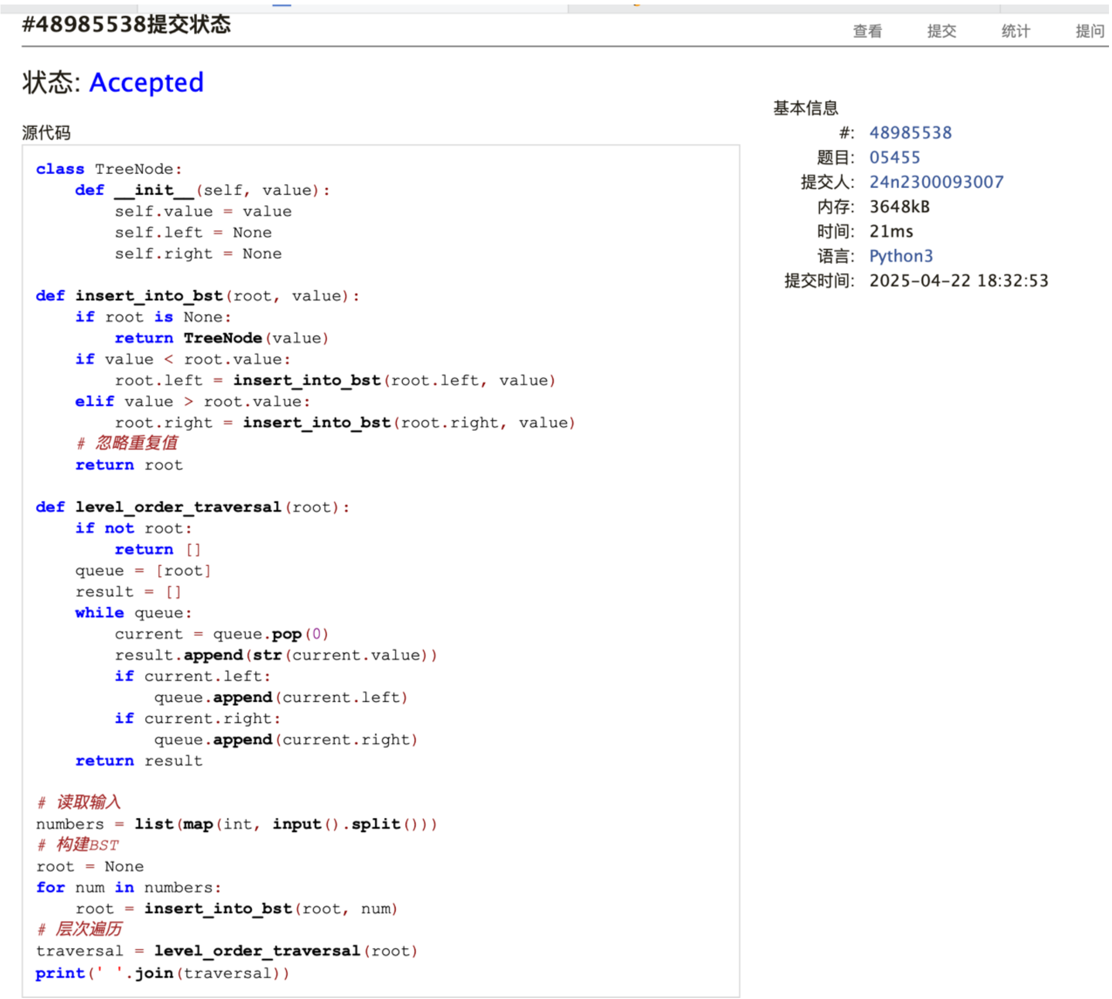
#:	48985507
题目:	04080
提交人:	24n2300093007
内存:	3616kB
时间:	22ms
语言:	Python3
提交时间:	2025-04-22 18:24:05

### M05455: 二叉搜索树的层次遍历

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

思路:

代码：



#48985538提交状态

状态: Accepted

源代码

```
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

def insert_into_bst(root, value):
    if root is None:
        return TreeNode(value)
    if value < root.value:
        root.left = insert_into_bst(root.left, value)
    elif value > root.value:
        root.right = insert_into_bst(root.right, value)
    # 忽略重复值
    return root

def level_order_traversal(root):
    if not root:
        return []
    queue = [root]
    result = []
    while queue:
        current = queue.pop(0)
        result.append(str(current.value))
        if current.left:
            queue.append(current.left)
        if current.right:
            queue.append(current.right)
    return result

# 读取输入
numbers = list(map(int, input().split()))
# 构建BST
root = None
for num in numbers:
    root = insert_into_bst(root, num)
# 层次遍历
traversal = level_order_traversal(root)
print(' '.join(traversal))
```

基本信息

#:	48985538
题目:	05455
提交人:	24n2300093007
内存:	3648kB
时间:	21ms
语言:	Python3
提交时间:	2025-04-22 18:32:53

### M04078: 实现堆结构

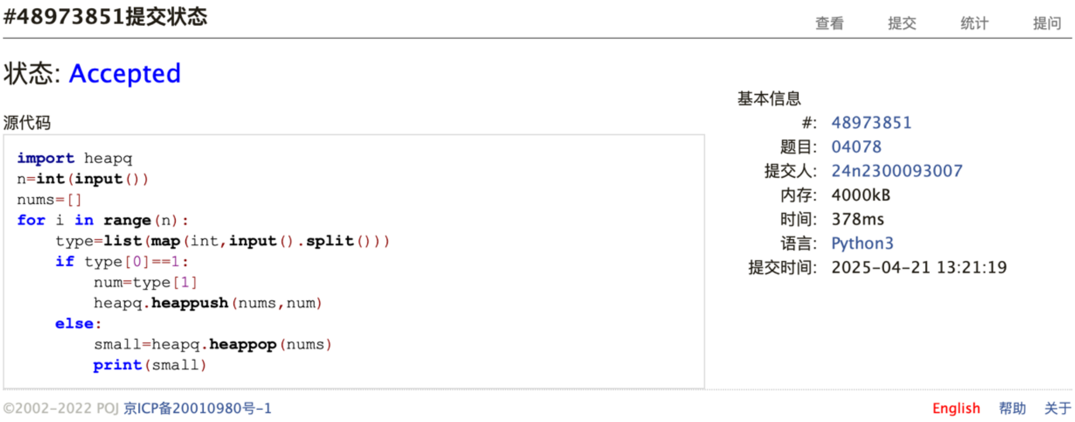
手搓实现，http://cs101.openjudge.cn/practice/04078/

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：

代码：



#48973851提交状态

状态: Accepted

源代码

```
import heapq
n=int(input())
nums=[]
for i in range(n):
    type=list(map(int,input().split()))
    if type[0]==1:
        num=type[1]
        heapq.heappush(nums,num)
    else:
        small=heapq.heappop(nums)
        print(small)
```

基本信息

#:	48973851
题目:	04078
提交人:	24n2300093007
内存:	4000kB
时间:	378ms
语言:	Python3
提交时间:	2025-04-21 13:21:19

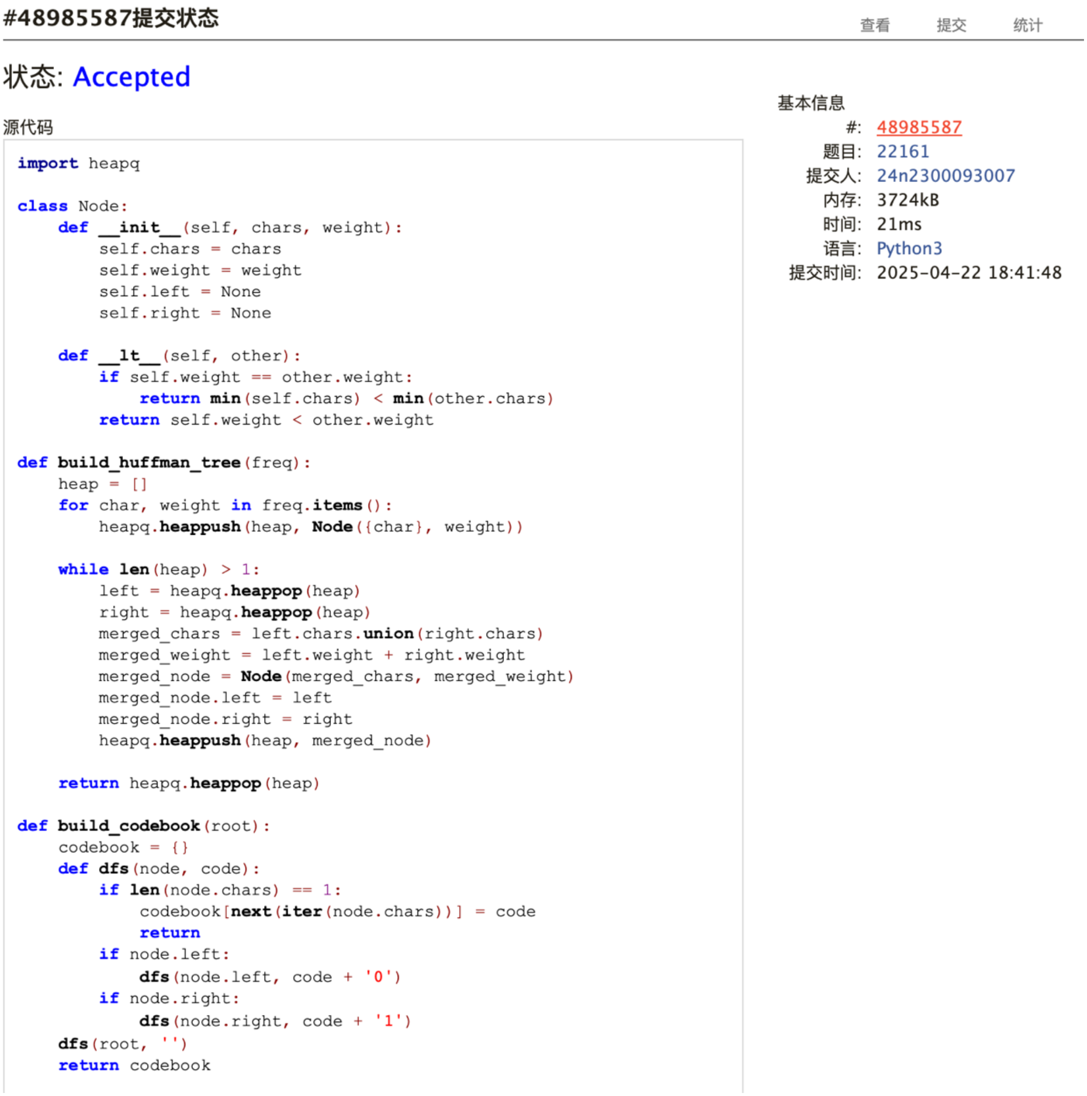
©2002-2022 POJ 京ICP备20010980号-1

### T22161: 哈夫曼编码树

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

思路：

代码：



#48985587提交状态

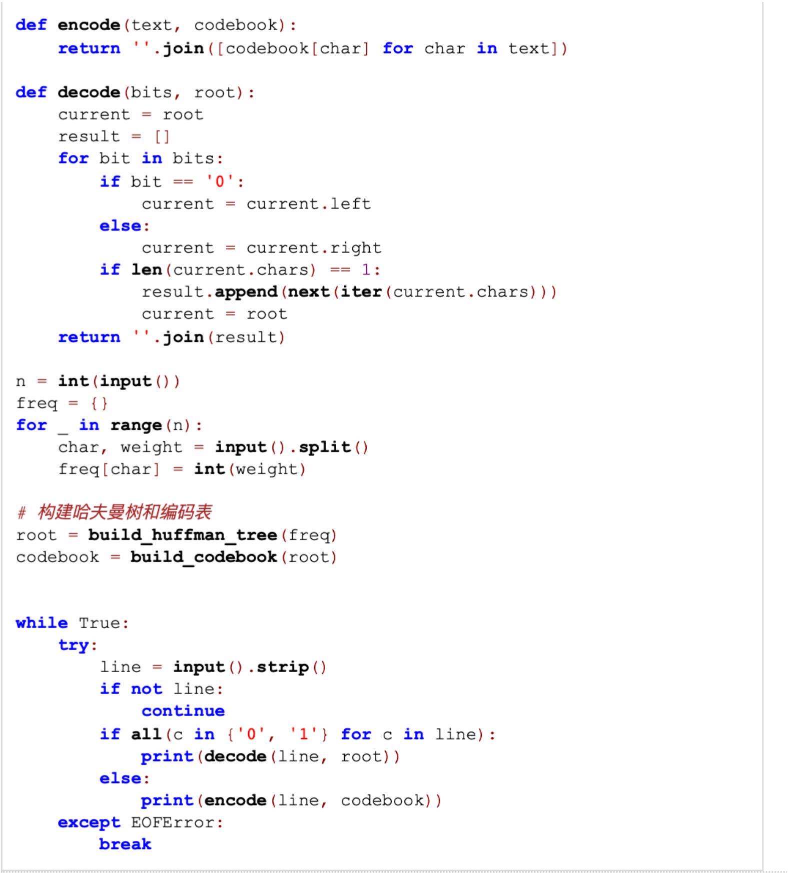
状态: Accepted

源代码

```python
import heapq

class Node:
    def __init__(self, chars, weight):
        self.chars = chars
        self.weight = weight
        self.left = None
        self.right = None

    def __lt__(self, other):
        if self.weight == other.weight:
            return min(self.chars) < min(other.chars)
        return self.weight < other.weight

def build_huffman_tree(freq):
    heap = []
    for char, weight in freq.items():
        heapq.heappush(heap, Node({char}, weight))

    while len(heap) > 1:
        left = heapq.heappop(heap)
        right = heapq.heappop(heap)
        merged_chars = left.chars.union(right.chars)
        merged_weight = left.weight + right.weight
        merged_node = Node(merged_chars, merged_weight)
        merged_node.left = left
        merged_node.right = right
        heapq.heappush(heap, merged_node)

    return heapq.heappop(heap)

def build_codebook(root):
    codebook = {}
    def dfs(node, code):
        if len(node.chars) == 1:
            codebook[next(iter(node.chars))] = code
            return
        if node.left:
            dfs(node.left, code + '0')
        if node.right:
            dfs(node.right, code + '1')
    dfs(root, '')
    return codebook
```

基本信息

#:	48985587
题目:	22161
提交人:	24n2300093007
内存:	3724kB
时间:	21ms
语言:	Python3
提交时间:	2025-04-22 18:41:48



```
def encode(text, codebook):
    return ''.join([codebook[char] for char in text])

def decode(bits, root):
    current = root
    result = []
    for bit in bits:
        if bit == '0':
            current = current.left
        else:
            current = current.right
        if len(current.chars) == 1:
            result.append(next(iter(current.chars)))
            current = root
    return ''.join(result)

n = int(input())
freq = {}
for _ in range(n):
    char, weight = input().split()
    freq[char] = int(weight)

# 构建哈夫曼树和编码表
root = build_huffman_tree(freq)
codebook = build_codebook(root)


while True:
    try:
        line = input().strip()
        if not line:
            continue
        if all(c in {'0', '1'} for c in line):
            print(decode(line, root))
        else:
            print(encode(line, codebook))
    except EOFError:
        break
```

## 2．学习总结和收获

通过这次作业，明白了堆的结构以及实现的方法。但我还是对二叉树的用法以及结构不太熟悉，我要多多做题复习相关知识。