Assignment #9: Huffman, BST & Heap

Updated 1834 GMT+8 Apr 15, 2025

2025 spring, Complied by **袁奕 2400010766 数院**

> **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

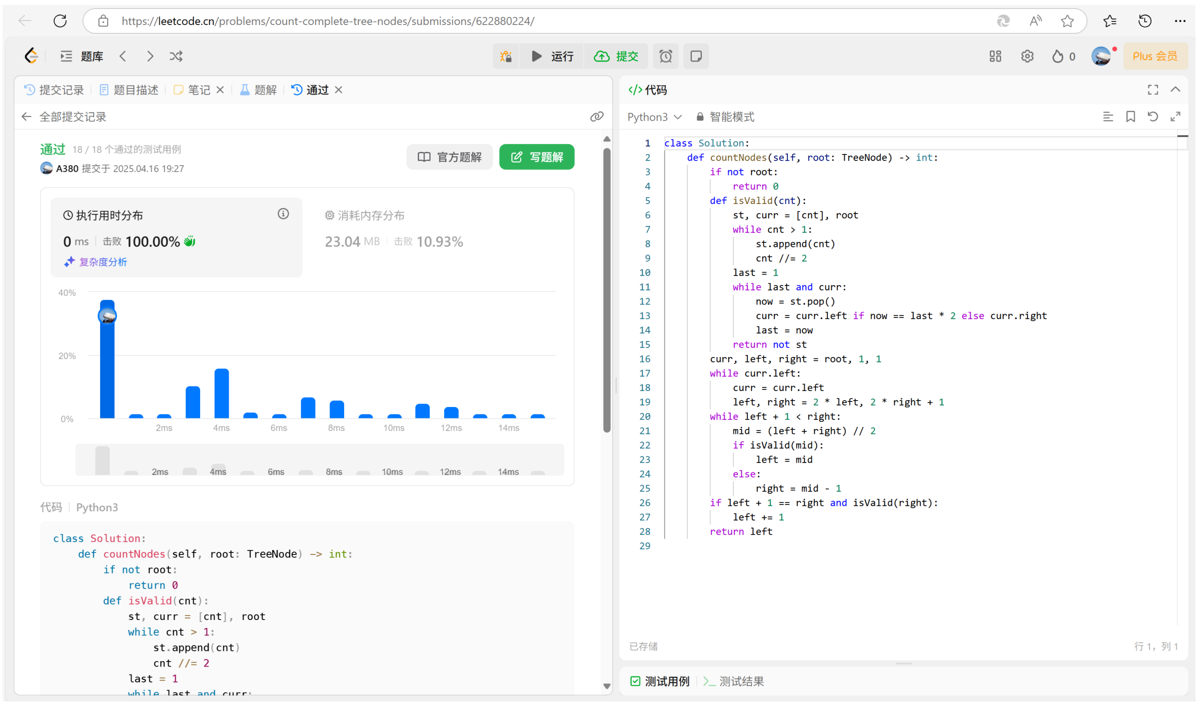
LC222.完全二叉树的节点个数

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

思路：

代码：

```python
class Solution:
    def countNodes(self, root: TreeNode) -> int:
        if not root:
            return 0
        def isValid(cnt):
            st, curr = [cnt], root
            while cnt > 1:
                st.append(cnt)
                cnt //= 2
            last = 1
            while last and curr:
                now = st.pop()
                curr = curr.left if now == last * 2 else curr.right
                last = now
            return not st
        curr, left, right = root, 1, 1
        while curr.left:
            curr = curr.left
            left, right = 2 * left, 2 * right + 1
        while left + 1 < right:
            mid = (left + right) // 2
            if isValid(mid):
                left = mid
            else:
                right = mid - 1
        if left + 1 == right and isValid(right):
            left += 1
        return left
```



https://leetcode.cn/problems/count-complete-tree-nodes/submissions/622880224/

通过 18 / 18 个通过的测试用例

A380 提交于 2025.04.16 19:27

执行用时分布 0 ms | 击败 100.00%

消耗内存分布 23.04 MB | 击败 10.93%

代码 | Python3

```python
class Solution:
    def countNodes(self, root: TreeNode) -> int:
        if not root:
            return 0
        def isValid(cnt):
            st, curr = [cnt], root
            while cnt > 1:
                st.append(cnt)
                cnt //= 2
            last = 1
            while last and curr:
                now = st.pop()
                curr = curr.left if now == last * 2 else curr.right
                last = now
            return not st
        curr, left, right = root, 1, 1
        while curr.left:
            curr = curr.left
            left, right = 2 * left, 2 * right + 1
        while left + 1 < right:
            mid = (left + right) // 2
            if isValid(mid):
                left = mid
            else:
                right = mid - 1
        if left + 1 == right and isValid(right):
            left += 1
        return left
```

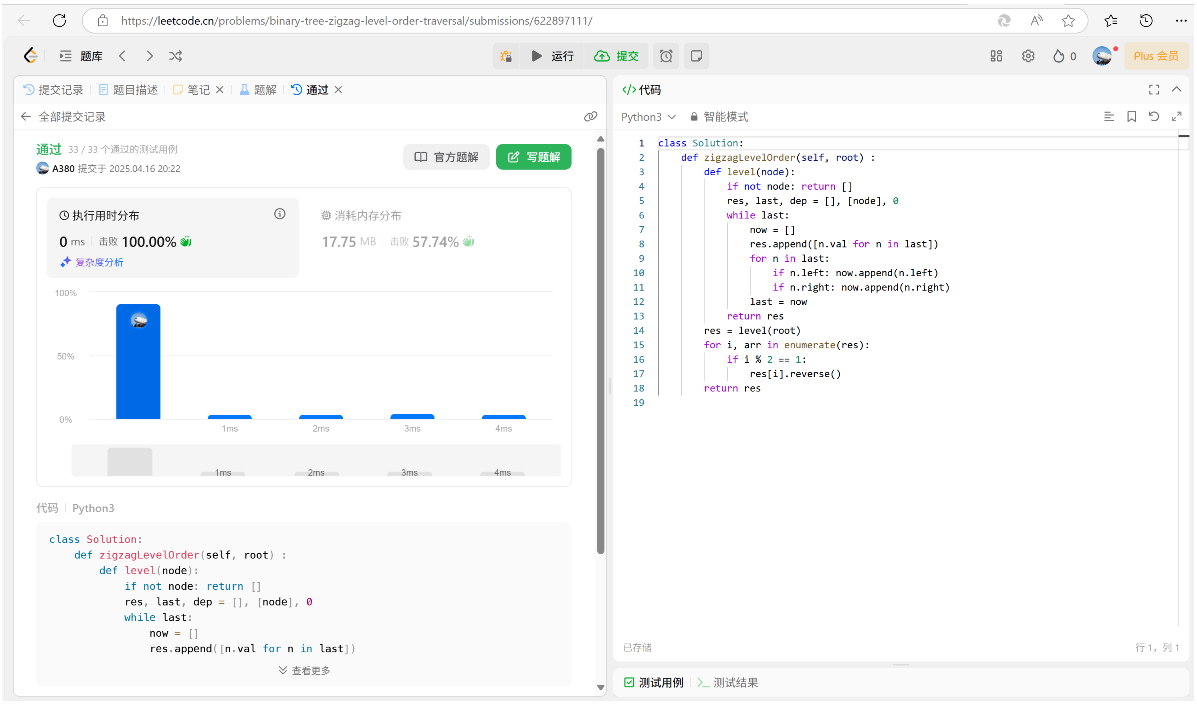
LC103.二叉树的锯齿形层序遍历

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

思路：开始用递归层序遍历, 导致 $O(n^2)$ 复杂度. 应该用 bfs 方法

代码：

```python
class Solution:
    def zigzagLevelOrder(self, root) :
        def level(node):
            if not node: return []
            res, last, dep = [], [node], 0
            while last:
                now = []
                res.append([n.val for n in last])
                for n in last:
                    if n.left: now.append(n.left)
                    if n.right: now.append(n.right)
                last = now
            return res
        res = level(root)
        for i, arr in enumerate(res):
            if i % 2 == 1:
                res[i].reverse()
        return res
```



https://leetcode.cn/problems/binary-tree-zigzag-level-order-traversal/submissions/622897111/

通过 33 / 33 个通过的测试用例

A380 提交于 2025.04.16 20:22

执行用时分布 0 ms | 击败 100.00%

消耗内存分布 17.75 MB | 击败 57.74%

代码 | Python3

```python
class Solution:
    def zigzagLevelOrder(self, root) :
        def level(node):
            if not node: return []
            res, last, dep = [], [node], 0
            while last:
                now = []
                res.append([n.val for n in last])
                for n in last:
                    if n.left: now.append(n.left)
                    if n.right: now.append(n.right)
                last = now
            return res
        res = level(root)
        for i, arr in enumerate(res):
            if i % 2 == 1:
                res[i].reverse()
        return res
```

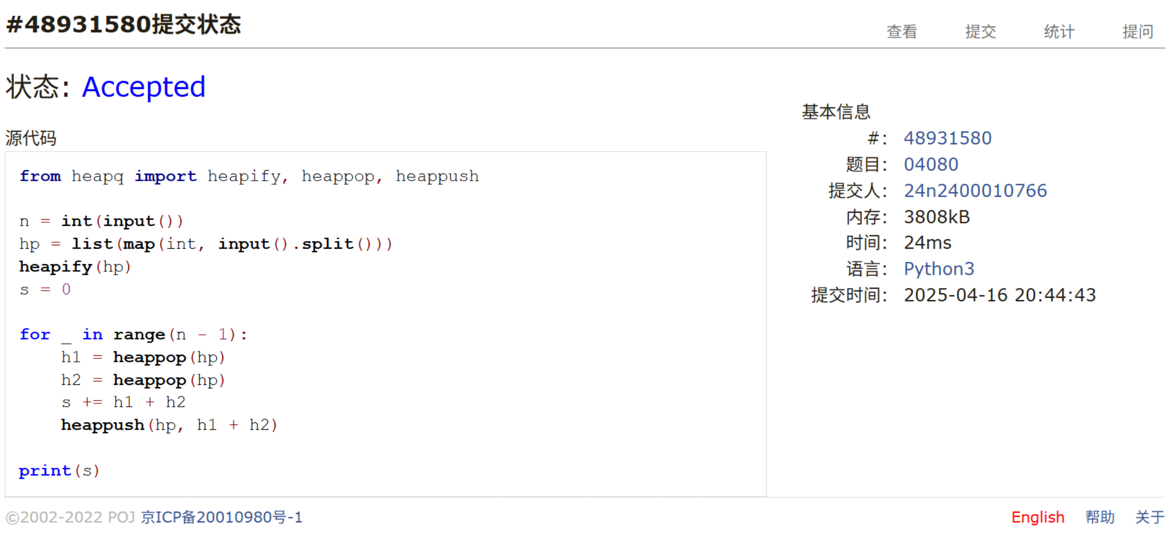
M04080:Huffman编码树

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

思路：

代码：

```python
from heapq import heapify, heappop, heappush

n = int(input())
hp = list(map(int, input().split()))
heapify(hp)
s = 0

for _ in range(n - 1):
    h1 = heappop(hp)
    h2 = heappop(hp)
    s += h1 + h2
    heappush(hp, h1 + h2)

print(s)
```


#48931580提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
from heapq import heapify, heappop, heappush

n = int(input())
hp = list(map(int, input().split()))
heapify(hp)
s = 0

for _ in range(n - 1):
    h1 = heappop(hp)
    h2 = heappop(hp)
    s += h1 + h2
    heappush(hp, h1 + h2)

print(s)
```

基本信息

#:	48931580
题目:	04080
提交人:	24n2400010766
内存:	3808kB
时间:	24ms
语言:	Python3
提交时间:	2025-04-16 20:44:43

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

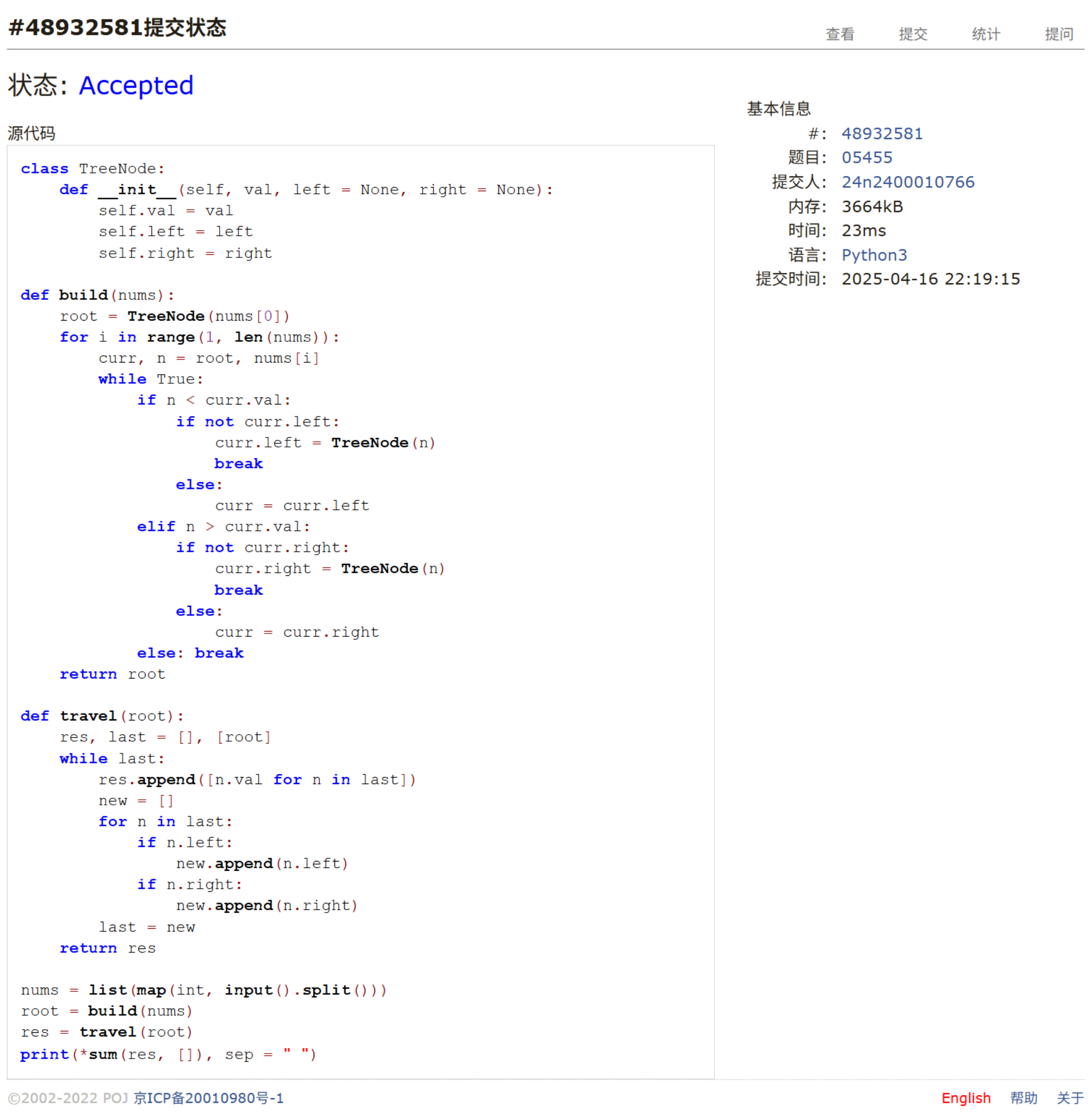
M05455: 二叉搜索树的层次遍历

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

思路:

代码:

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

def build(nums):
    root = TreeNode(nums[0])
    for i in range(1, len(nums)):
        curr, n = root, nums[i]
        while True:
            if n < curr.val:
                if not curr.left:
                    curr.left = TreeNode(n)
                    break
                else:
                    curr = curr.left
            elif n > curr.val:
                if not curr.right:
                    curr.right = TreeNode(n)
                    break
                else:
                    curr = curr.right
            else: break
    return root

def travel(root):
    res, last = [], [root]
    while last:
        res.append([n.val for n in last])
        new = []
        for n in last:
            if n.left:
                new.append(n.left)
            if n.right:
                new.append(n.right)
        last = new
    return res

nums = list(map(int, input().split()))
root = build(nums)
res = travel(root)
print(*sum(res, []), sep = " ")
```



#48932581提交状态

查看 提交 统计 提问

状态: Accepted

源代码

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

def build(nums):
    root = TreeNode(nums[0])
    for i in range(1, len(nums)):
        curr, n = root, nums[i]
        while True:
            if n < curr.val:
                if not curr.left:
                    curr.left = TreeNode(n)
                    break
                else:
                    curr = curr.left
            elif n > curr.val:
                if not curr.right:
                    curr.right = TreeNode(n)
                    break
                else:
                    curr = curr.right
            else: break
    return root

def travel(root):
    res, last = [], [root]
    while last:
        res.append([n.val for n in last])
        new = []
        for n in last:
            if n.left:
                new.append(n.left)
            if n.right:
                new.append(n.right)
        last = new
    return res

nums = list(map(int, input().split()))
root = build(nums)
res = travel(root)
print(*sum(res, []), sep = " ")
```

基本信息

#:	48932581
题目:	05455
提交人:	24n2400010766
内存:	3664kB
时间:	23ms
语言:	Python3
提交时间:	2025-04-16 22:19:15

©2002-2022 POJ 京ICP备20010980号-1

English 帮助 关于

M04078: 实现堆结构

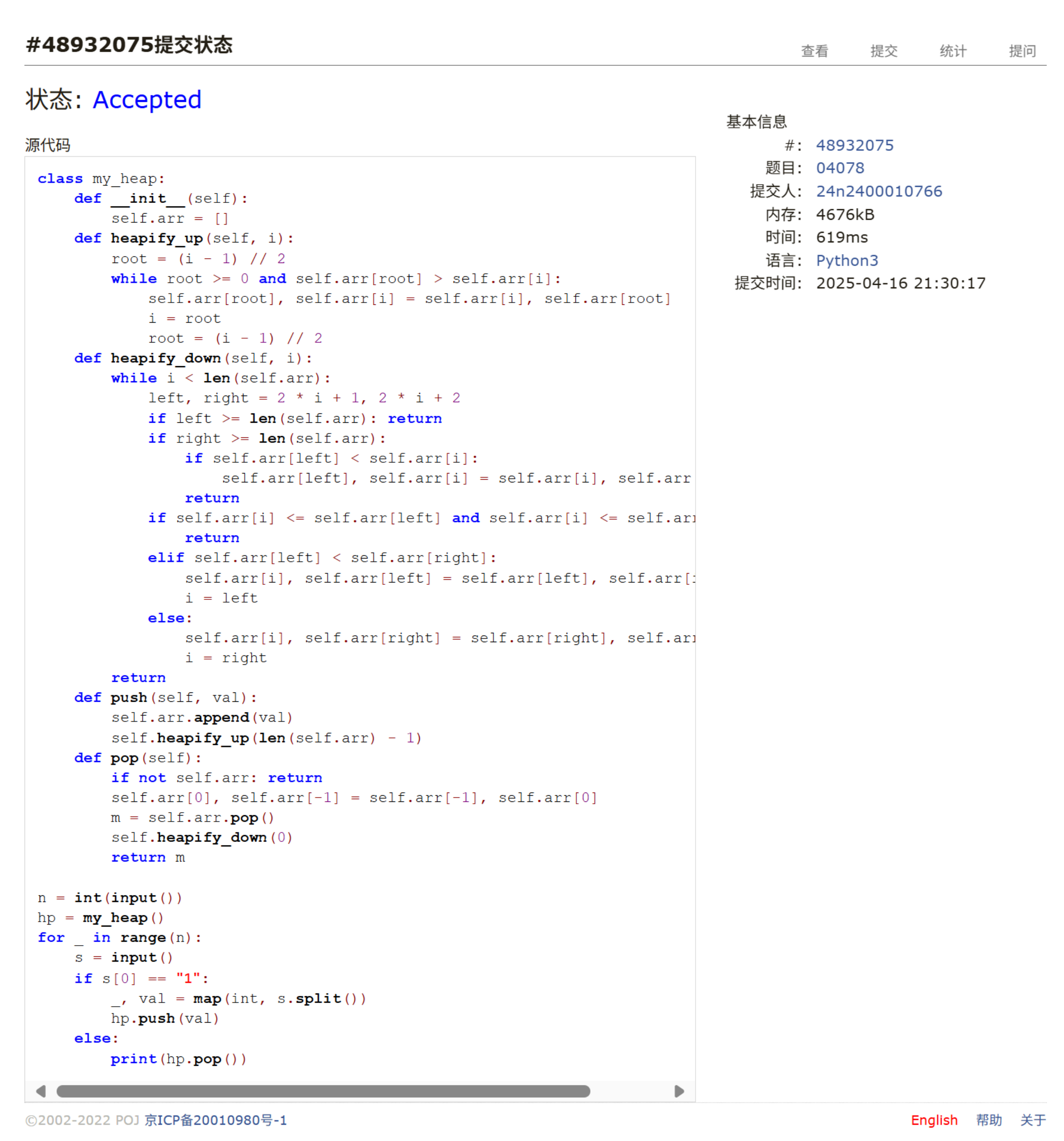
手搓实现，http://cs101.openjudge.cn/practice/04078/

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：

```python
class my_heap:
    def __init__(self):
        self.arr = []
    def heapify_up(self, i):
        root = (i - 1) // 2
        while root >= 0 and self.arr[root] > self.arr[i]:
            self.arr[root], self.arr[i] = self.arr[i], self.arr[root]
            i = root
            root = (i - 1) // 2
    def heapify_down(self, i):
        while i < len(self.arr):
            left, right = 2 * i + 1, 2 * i + 2
            if left >= len(self.arr): return
            if right >= len(self.arr):
                if self.arr[left] < self.arr[i]:
                    self.arr[left], self.arr[i] = self.arr[i], self.arr[left]
                return
            if self.arr[i] <= self.arr[left] and self.arr[i] <= self.arr[right]:
                return
            elif self.arr[left] < self.arr[right]:
                self.arr[i], self.arr[left] = self.arr[left], self.arr[i]
                i = left
            else:
                self.arr[i], self.arr[right] = self.arr[right], self.arr[i]
                i = right
        return
    def push(self, val):
        self.arr.append(val)
        self.heapify_up(len(self.arr) - 1)
    def pop(self):
        if not self.arr: return
        self.arr[0], self.arr[-1] = self.arr[-1], self.arr[0]
        m = self.arr.pop()
        self.heapify_down(0)
        return m

n = int(input())
hp = my_heap()
for _ in range(n):
    s = input()
    if s[0] == "1":
        _, val = map(int, s.split())
        hp.push(val)
    else:
        print(hp.pop())
```



#48932075提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```python
class my_heap:
    def __init__(self):
        self.arr = []
    def heapify_up(self, i):
        root = (i - 1) // 2
        while root >= 0 and self.arr[root] > self.arr[i]:
            self.arr[root], self.arr[i] = self.arr[i], self.arr[root]
            i = root
            root = (i - 1) // 2
    def heapify_down(self, i):
        while i < len(self.arr):
            left, right = 2 * i + 1, 2 * i + 2
            if left >= len(self.arr): return
            if right >= len(self.arr):
                if self.arr[left] < self.arr[i]:
                    self.arr[left], self.arr[i] = self.arr[i], self.arr
                return
            if self.arr[i] <= self.arr[left] and self.arr[i] <= self.arr
                return
            elif self.arr[left] < self.arr[right]:
                self.arr[i], self.arr[left] = self.arr[left], self.arr[
                i = left
            else:
                self.arr[i], self.arr[right] = self.arr[right], self.arr
                i = right
        return
    def push(self, val):
        self.arr.append(val)
        self.heapify_up(len(self.arr) - 1)
    def pop(self):
        if not self.arr: return
        self.arr[0], self.arr[-1] = self.arr[-1], self.arr[0]
        m = self.arr.pop()
        self.heapify_down(0)
        return m

n = int(input())
hp = my_heap()
for _ in range(n):
    s = input()
    if s[0] == "1":
        _, val = map(int, s.split())
        hp.push(val)
    else:
        print(hp.pop())
```

基本信息

#:	48932075
题目:	04078
提交人:	24n2400010766
内存:	4676kB
时间:	619ms
语言:	Python3
提交时间:	2025-04-16 21:30:17

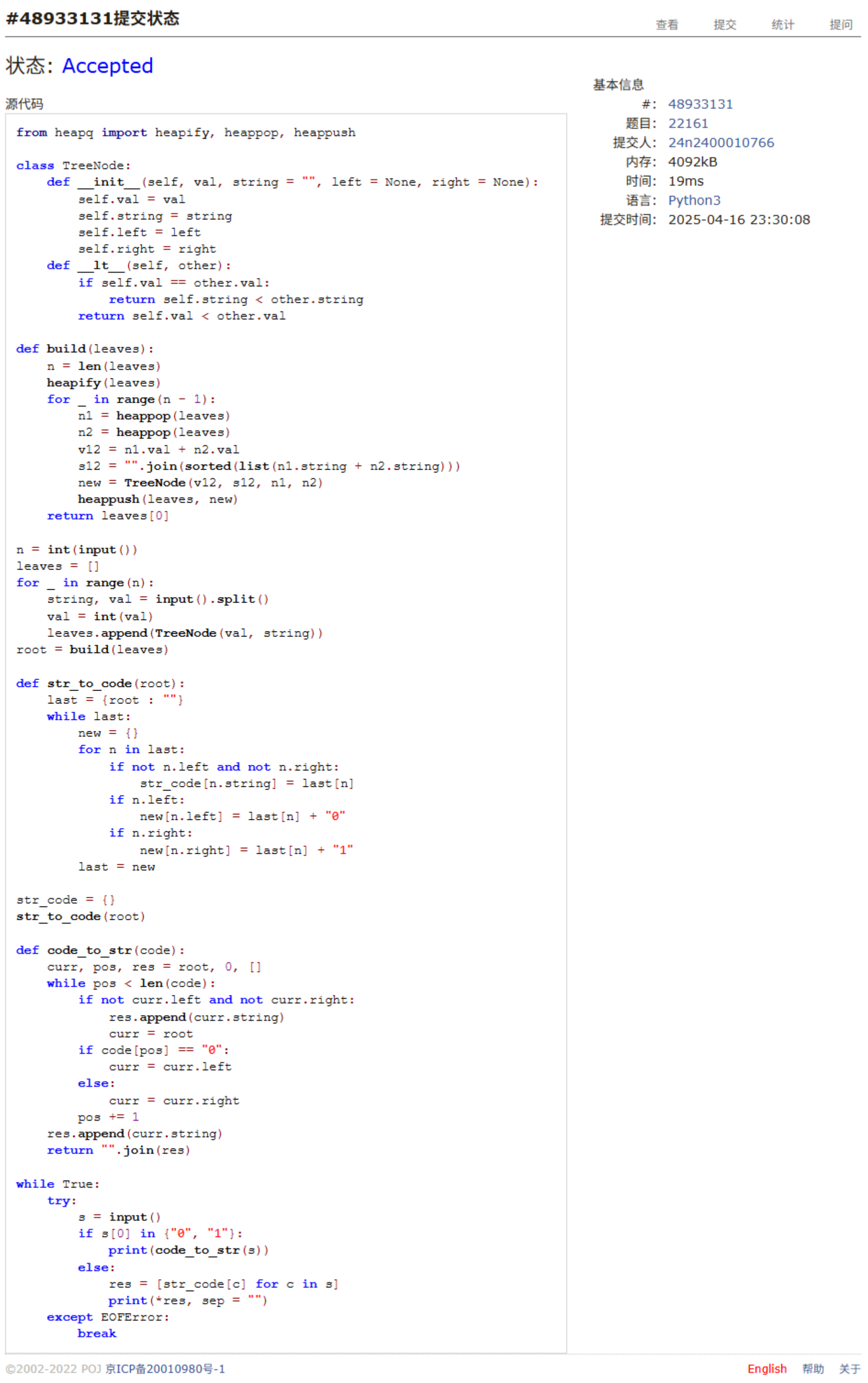
T22161: 哈夫曼编码树

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

思路:

代码:

```python
from heapq import heapify, heappop, heappush

class TreeNode:
    def __init__(self, val, string = "", left = None, right = None):
        self.val = val
        self.string = string
        self.left = left
        self.right = right
    def __lt__(self, other):
        if self.val == other.val:
            return self.string < other.string
        return self.val < other.val

def build(leaves):
    n = len(leaves)
    heapify(leaves)
    for _ in range(n - 1):
        n1 = heappop(leaves)
        n2 = heappop(leaves)
        v12 = n1.val + n2.val
        s12 = "".join(sorted(list(n1.string + n2.string)))
        new = TreeNode(v12, s12, n1, n2)
        heappush(leaves, new)
    return leaves[0]

n = int(input())
leaves = []
for _ in range(n):
    string, val = input().split()
    val = int(val)
    leaves.append(TreeNode(val, string))
root = build(leaves)

def str_to_code(root):
    last = {root : ""}
    while last:
        new = {}
        for n in last:
            if not n.left and not n.right:
                str_code[n.string] = last[n]
            if n.left:
                new[n.left] = last[n] + "0"
            if n.right:
                new[n.right] = last[n] + "1"
        last = new

str_code = {}
str_to_code(root)

def code_to_str(code):
    curr, pos, res = root, 0, []
    while pos < len(code):
        if not curr.left and not curr.right:
            res.append(curr.string)
            curr = root
        if code[pos] == "0":
            curr = curr.left
        else:
            curr = curr.right
        pos += 1
    res.append(curr.string)
    return "".join(res)

while True:
    try:
        s = input()
        if s[0] in {"0", "1"}:
            print(code_to_str(s))
        else:
            res = [str_code[c] for c in s]
            print(*res, sep = "")
    except EOFError:
        break
```



#48933131提交状态

状态: Accepted

基本信息

#: 48933131
题目: 22161
提交人: 24n2400010766
内存: 4092kB
时间: 19ms
语言: Python3
提交时间: 2025-04-16 23:30:08

源代码

```python
from heapq import heapify, heappop, heappush

class TreeNode:
    def __init__(self, val, string = "", left = None, right = None):
        self.val = val
        self.string = string
        self.left = left
        self.right = right
    def __lt__(self, other):
        if self.val == other.val:
            return self.string < other.string
        return self.val < other.val

def build(leaves):
    n = len(leaves)
    heapify(leaves)
    for _ in range(n - 1):
        n1 = heappop(leaves)
        n2 = heappop(leaves)
        v12 = n1.val + n2.val
        s12 = "".join(sorted(list(n1.string + n2.string)))
        new = TreeNode(v12, s12, n1, n2)
        heappush(leaves, new)
    return leaves[0]

n = int(input())
leaves = []
for _ in range(n):
    string, val = input().split()
    val = int(val)
    leaves.append(TreeNode(val, string))
root = build(leaves)

def str_to_code(root):
    last = {root : ""}
    while last:
        new = {}
        for n in last:
            if not n.left and not n.right:
                str_code[n.string] = last[n]
            if n.left:
                new[n.left] = last[n] + "0"
            if n.right:
                new[n.right] = last[n] + "1"
        last = new

str_code = {}
str_to_code(root)

def code_to_str(code):
    curr, pos, res = root, 0, []
    while pos < len(code):
        if not curr.left and not curr.right:
            res.append(curr.string)
            curr = root
        if code[pos] == "0":
            curr = curr.left
        else:
            curr = curr.right
        pos += 1
    res.append(curr.string)
    return "".join(res)

while True:
    try:
        s = input()
        if s[0] in {"0", "1"}:
            print(code_to_str(s))
        else:
            res = [str_code[c] for c in s]
            print(*res, sep = "")
    except EOFError:
        break
```

©2002-2022 POJ 京ICP备20010980号-1

2. 学习总结和收获

感觉本次作业代码长度普遍较长. 应该养成OOP的好习惯, 并且写完一部分后逐个模块进行测试.