Assignment #9: Huffman, BST & Heap

Updated 1834 GMT+8 Apr 15, 2025

2025 spring, Complied by 任宇桐 物理学院

> **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

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

思路：

直接实现非常简单，学习了一下题解使用二分实现的方法。

代码：

```
# 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:
        def can_reach(s):
            direc_for_nums = {'0':lambda x: x.left, '1':lambda x:x.right}
            bin_format = bin(s)[3:]
            node = root
            for num in bin_format:
                node = direc_for_nums[num](node)
            if node:
                return True
```

```
            else:
                return False

        def binary_search():
            low = minv
            high = maxv
            while low <= high:
                mid = (low+high)//2
                if can_reach(mid):
                    low = mid+1
                else:
                    high = mid-1
            return high

        cnt = -1
        node = root
        while node:
            node = node.left
            cnt += 1
        if cnt == -1:
            return 0
        else:
            minv = 2**cnt
            maxv = 2**(cnt+1)-1
            return binary_search()
```

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：

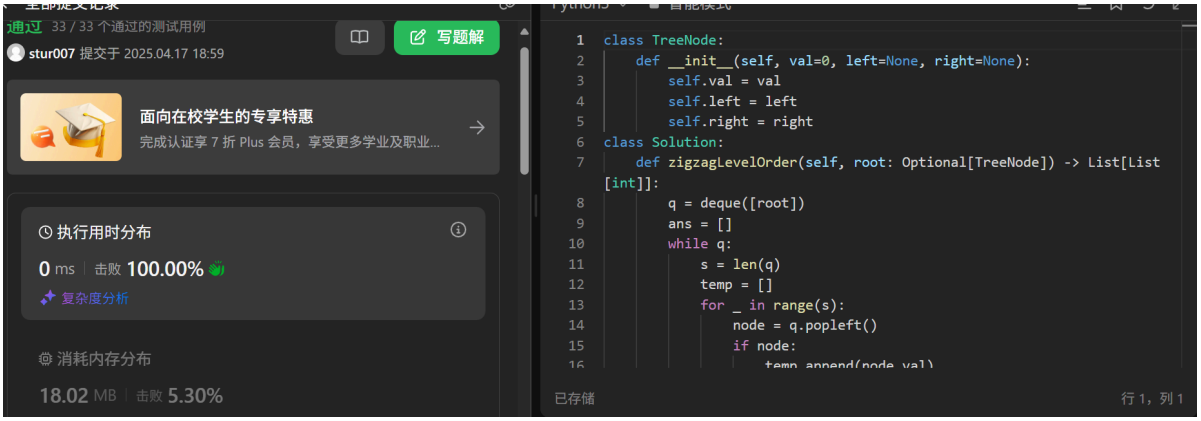
先正常做bfs，然后将对应的项反转。

代码：

```
class Solution:
    def zigzagLevelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:
        q = deque([root])
```

```
        ans = []
        while q:
            s = len(q)
            temp = []
            for _ in range(s):
                node = q.popleft()
                if node:
                    temp.append(node.val)
                    q.append(node.left)
                    q.append(node.right)
            if temp:
                ans.append(temp[:])
        for i in range(len(ans)):
            if i%2 == 1:
                ans[i] = list(reversed(ans[i]))
        return ans
```

代码运行截图 （至少包含有"Accepted"）



M04080:Huffman编码树

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

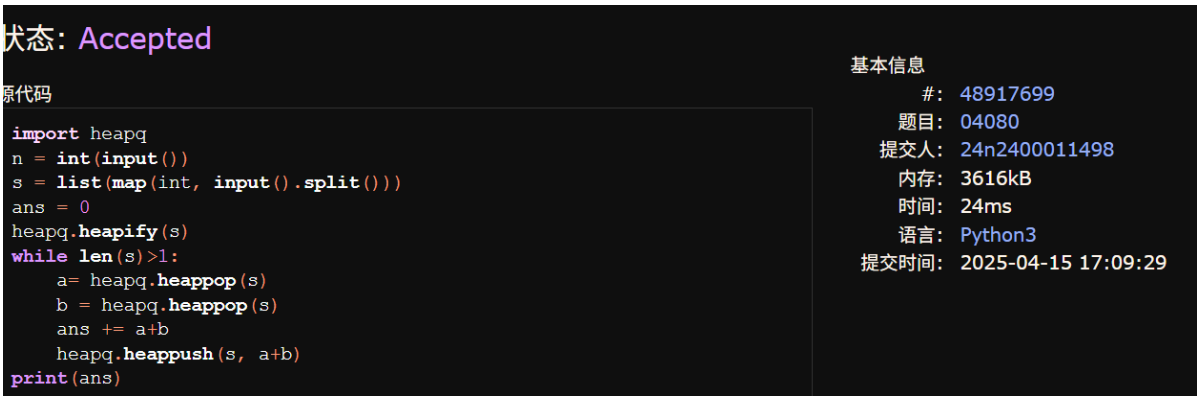
思路：

借助heapq直接实现即可。

代码：

```
import heapq
n = int(input())
s = list(map(int, input().split()))
ans = 0
heapq.heapify(s)
while len(s)>1:
    a= heapq.heappop(s)
    b = heapq.heappop(s)
    ans += a+b
    heapq.heappush(s, a+b)
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:

BST二分建树+bfs遍历

代码:

```
from collections import deque

s = list(map(int, input().split()))
s = list(dict.fromkeys(s))
class Node:
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None

    def __lt__(self, other):
        return  self.val < other.val

def parse_tree(root, node):
    if node < root:
        if root.left:
            parse_tree(root.left, node)
        else:
            root.left = node
    else:
        if root.right:
            parse_tree(root.right, node)
        else:
            root.right = node

def levelorder(root):
    q = deque([root])
    ans = []
    while q:
        node = q.popleft()
```

```
        if node:
            ans.append(node.val)
            q.append(node.left)
            q.append(node.right)
    return ans

root = Node(s[0])
for i in range(1, len(s)):
    node = Node(s[i])
    parse_tree(root, node)
print(*levelorder(root))
```

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：

使用不断递归实现即可，注意各种边界条件的处理（堆是否为空）。

代码：

```
class Heapq:
    def __init__(self):
        self.heap = []

    def heappush(self, val):
        self.heap.append(val)
        node_index = len(self.heap)-1
        while True:
            if node_index == 0:
                break
            parent_index = (node_index-1)//2
```

```
            if self.heap[parent_index] > self.heap[node_index]:
                self.heap[parent_index], self.heap[node_index]=
self.heap[node_index], self.heap[parent_index]
                node_index = parent_index
            else:
                break
    def heappop(self):
        val = self.heap[0]
        if len(self.heap) == 1:
            return self.heap.pop()
        self.heap[0] = self.heap.pop()
        node_index = 0
        while True:
            child_index_1 = 2*node_index+1
            child_index_2 = 2*node_index+2
            if child_index_1 >= len(self.heap):
                break
            elif child_index_2 >= len(self.heap):
                if self.heap[child_index_1] < self.heap[node_index]:
                    self.heap[child_index_1], self.heap[node_index] =
self.heap[node_index], self.heap[child_index_1]
                break
            else:
                min_val = min(self.heap[node_index], self.heap[child_index_1],
self.heap[child_index_2])
                if min_val == self.heap[node_index]:
                    break
                elif min_val == self.heap[child_index_1]:
                    self.heap[node_index], self.heap[child_index_1] =
self.heap[child_index_1], self.heap[node_index]
                    node_index = child_index_1
                else:
                    self.heap[node_index], self.heap[child_index_2] =
self.heap[child_index_2], self.heap[node_index]
                    node_index = child_index_2
        return val

heapq = Heapq()
n = int(input())
for _ in range(n):
    s = list(map(int, input().split()))
    if s[0] == 1:
        heapq.heappush(s[1])
    else:
        print(heapq.heappop())
```

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:

使用heapq不断弹出最小值，然后建树。

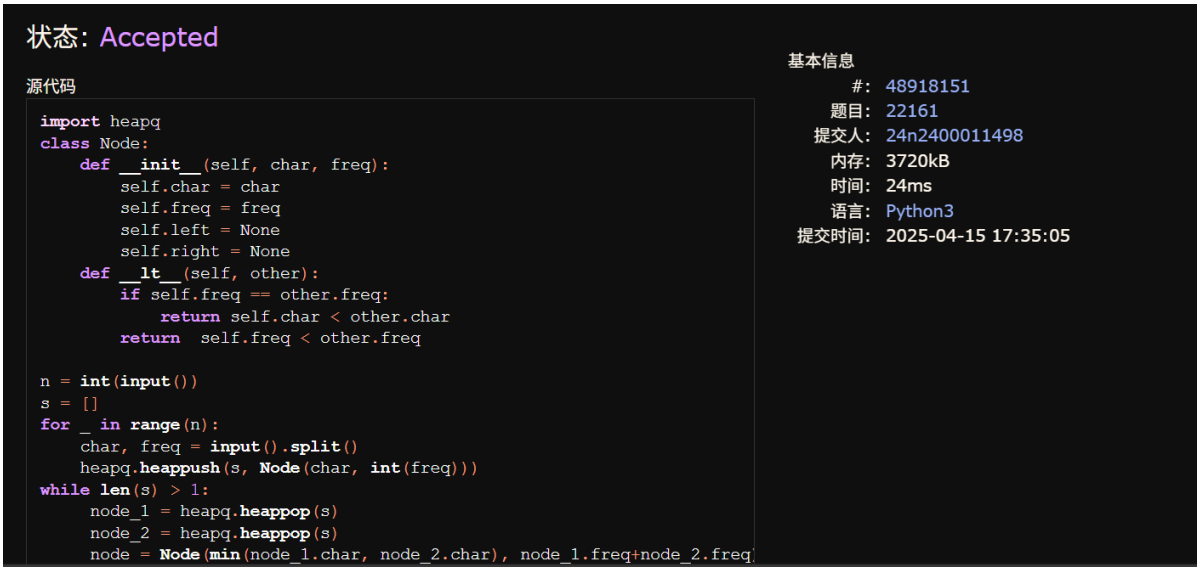
代码:

```
import heapq
class Node:
    def __init__(self, char, freq):
        self.char = char
        self.freq = freq
        self.left = None
        self.right = None
    def __lt__(self, other):
        if self.freq == other.freq:
            return self.char < other.char
        return  self.freq < other.freq

n = int(input())
s = []
for _ in range(n):
    char, freq = input().split()
    heapq.heappush(s, Node(char, int(freq)))
while len(s) > 1:
     node_1 = heapq.heappop(s)
     node_2 = heapq.heappop(s)
     node = Node(min(node_1.char, node_2.char), node_1.freq+node_2.freq)
     node.left = node_1
     node.right = node_2
     heapq.heappush(s, node)
root = s.pop()
encoding = dict()
decoding = dict()
def dfs(node, code):
    if not node.left and not node.right:
        encoding[node.char] = code
        decoding[code] = node.char
```

```
            return
        dfs(node.left, code+'0')
        dfs(node.right, code+'1')
    dfs(root, '')
    while True:
        try:
            s = input()
        except EOFError:
            break
        if s.isalpha():
            ans = ''
            for i in range(len(s)):
                ans += encoding[s[i]]
            print(ans)
        elif s.isdigit():
            ans = ''
            temp = ''
            for i in range(len(s)):
                temp += s[i]
                if temp in decoding:
                    ans += decoding[temp]
                    temp = ''
            print(ans)
```

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。

用时最长的是手搓堆结构的题目，写起来思路不是特别复杂，但是边界条件等等细节处理起来非常麻烦。