

Assignment #9: Huffman, BST & Heap

Updated 1834 GMT+8 Apr 15, 2025

2025 spring, Compiled 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:

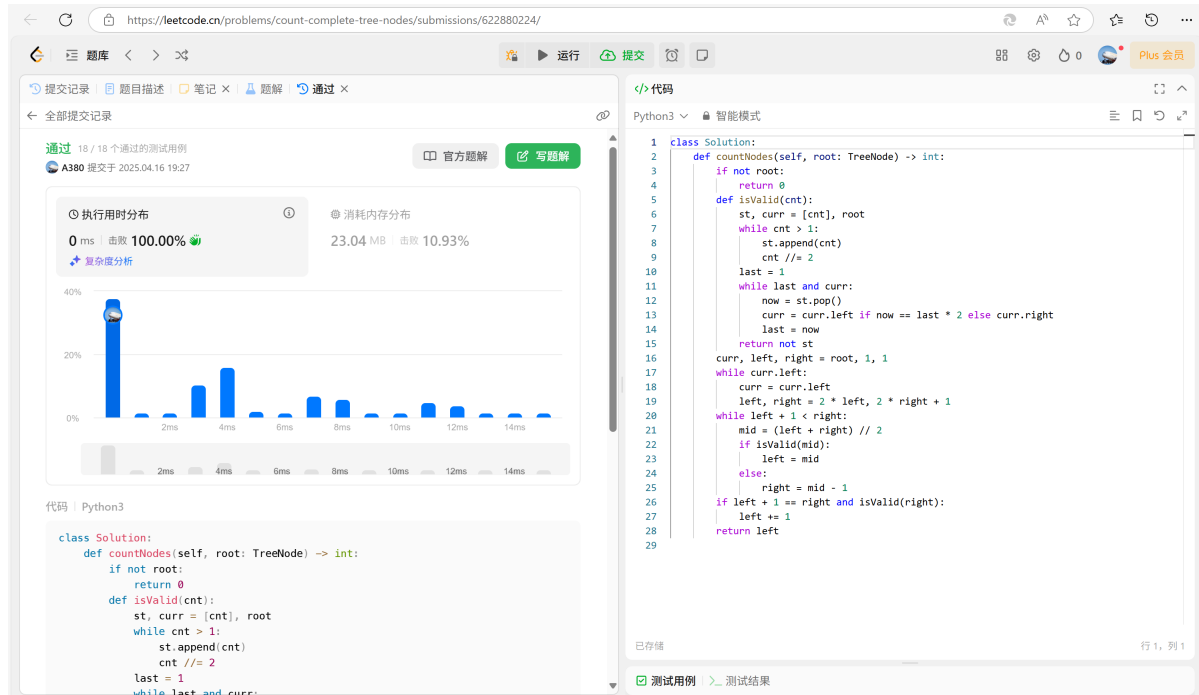
代码:

```
1 class Solution:
2     def countNodes(self, root: TreeNode) -> int:
3         if not root:
4             return 0
5         def isValid(cnt):
6             st, curr = [cnt], root
7             while cnt > 1:
8                 st.append(cnt)
9                 cnt //= 2
10            last = 1
11            while last and curr:
12                now = st.pop()
13                curr = curr.left if now == last * 2 else curr.right
14                last = now
15            return not st
16        curr, left, right = root, 1, 1
```

```

17         while curr.left:
18             curr = curr.left
19             left, right = 2 * left, 2 * right + 1
20         while left + 1 < right:
21             mid = (left + right) // 2
22             if isValid(mid):
23                 left = mid
24             else:
25                 right = mid - 1
26         if left + 1 == right and isValid(right):
27             left += 1
28         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：

```

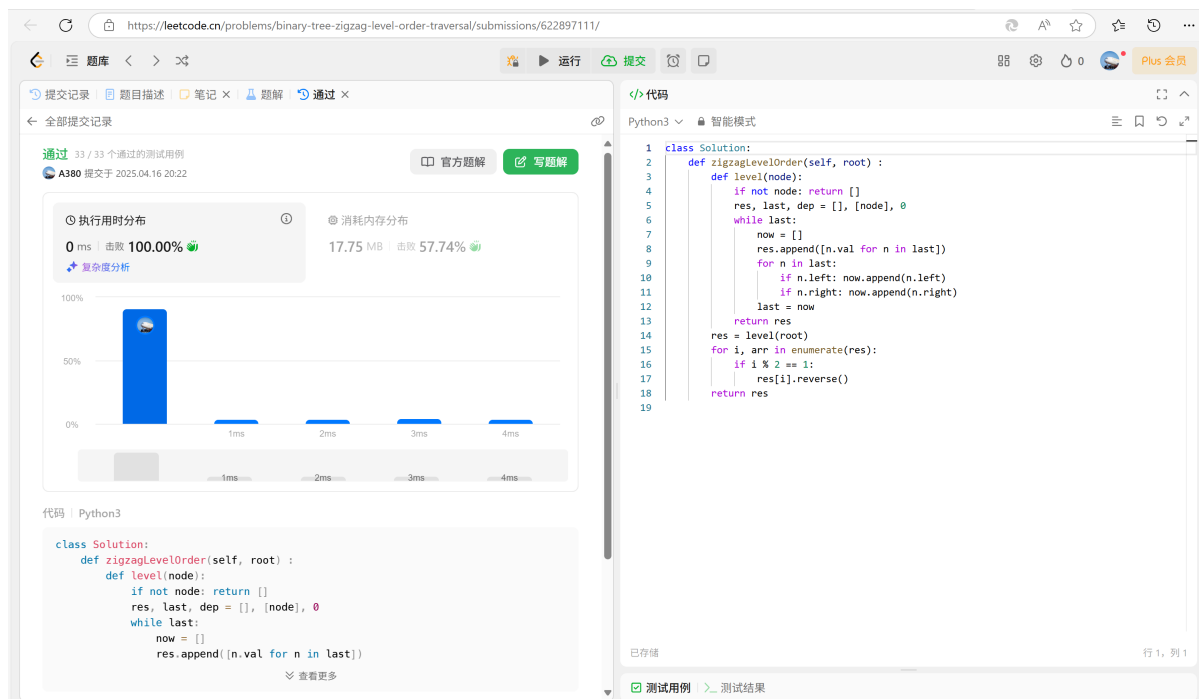
1 class Solution:
2     def zigzagLevelOrder(self, root) :
3         def level(node):
4             if not node: return []
5             res, last, dep = [], [node], 0
6             while last:
7                 now = []
8                 res.append([n.val for n in last])
9                 for n in last:
10                     if n.left: now.append(n.left)
11                     if n.right: now.append(n.right)
12                 last = now
13             return res
14         res = level(root)

```

```

15     for i, arr in enumerate(res):
16         if i % 2 == 1:
17             res[i].reverse()
18     return res

```



M04080:Huffman编码树

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

思路：

代码：

```

1  from heapq import heapify, heappop, heappush
2
3  n = int(input())
4  hp = list(map(int, input().split()))
5  heapify(hp)
6  s = 0
7
8  for _ in range(n - 1):
9      h1 = heappop(hp)
10     h2 = heappop(hp)
11     s += h1 + h2
12     heappush(hp, h1 + h2)
13
14 print(s)

```

状态: **Accepted**

源代码

```
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

M05455: 二叉搜索树的层次遍历

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

思路:

代码:

```
1 class TreeNode:
2     def __init__(self, val, left = None, right = None):
3         self.val = val
4         self.left = left
5         self.right = right
6
7 def build(nums):
8     root = TreeNode(nums[0])
9     for i in range(1, len(nums)):
10        curr, n = root, nums[i]
11        while True:
12            if n < curr.val:
13                if not curr.left:
14                    curr.left = TreeNode(n)
15                    break
16            else:
17                curr = curr.left
18            elif n > curr.val:
19                if not curr.right:
20                    curr.right = TreeNode(n)
21                    break
22            else:
23                curr = curr.right
24        else: break
25    return root
26
27 def travel(root):
28     res, last = [], [root]
```

```

29     while last:
30         res.append([n.val for n in last])
31         new = []
32         for n in last:
33             if n.left:
34                 new.append(n.left)
35             if n.right:
36                 new.append(n.right)
37         last = new
38     return res
39
40 nums = list(map(int, input().split()))
41 root = build(nums)
42 res = travel(root)
43 print(*sum(res, []), sep = " ")

```

#48932581提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: **Accepted**

源代码

```

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

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:

```
1 class my_heap:
2     def __init__(self):
3         self.arr = []
4     def heapify_up(self, i):
5         root = (i - 1) // 2
6         while root >= 0 and self.arr[root] > self.arr[i]:
7             self.arr[root], self.arr[i] = self.arr[i], self.arr[root]
8             i = root
9             root = (i - 1) // 2
10    def heapify_down(self, i):
11        while i < len(self.arr):
12            left, right = 2 * i + 1, 2 * i + 2
13            if left >= len(self.arr): return
14            if right >= len(self.arr):
15                if self.arr[left] < self.arr[i]:
16                    self.arr[left], self.arr[i] = self.arr[i],
self.arr[left]
17                return
18            if self.arr[i] <= self.arr[left] and self.arr[i] <=
self.arr[right]:
19                return
20            elif self.arr[left] < self.arr[right]:
21                self.arr[i], self.arr[left] = self.arr[left], self.arr[i]
22                i = left
23            else:
24                self.arr[i], self.arr[right] = self.arr[right], self.arr[i]
25                i = right
26        return
27    def push(self, val):
28        self.arr.append(val)
29        self.heapify_up(len(self.arr) - 1)
30    def pop(self):
31        if not self.arr: return
32        self.arr[0], self.arr[-1] = self.arr[-1], self.arr[0]
33        m = self.arr.pop()
34        self.heapify_down(0)
35        return m
36
37 n = int(input())
38 hp = my_heap()
39 for _ in range(n):
40     s = input()
41     if s[0] == "1":
42         _, val = map(int, s.split())
```

```
43         hp.push(val)
44     else:
45         print(hp.pop())
```

#48932075提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: Accepted

源代码

```
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())
```

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

©2002-2022 POJ 京ICP备20010980号-1

[English](#) [帮助](#) [关于](#)

T22161: 哈夫曼编码树

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

思路:

代码:

```
1 from heapq import heapify, heappop, heappush
2
3 class TreeNode:
4     def __init__(self, val, string = "", left = None, right = None):
5         self.val = val
```

```

6         self.string = string
7         self.left = left
8         self.right = right
9     def __lt__(self, other):
10         if self.val == other.val:
11             return self.string < other.string
12         return self.val < other.val
13
14     def build(leaves):
15         n = len(leaves)
16         heapify(leaves)
17         for _ in range(n - 1):
18             n1 = heappop(leaves)
19             n2 = heappop(leaves)
20             v12 = n1.val + n2.val
21             s12 = "".join(sorted(list(n1.string + n2.string)))
22             new = TreeNode(v12, s12, n1, n2)
23             heappush(leaves, new)
24         return leaves[0]
25
26     n = int(input())
27     leaves = []
28     for _ in range(n):
29         string, val = input().split()
30         val = int(val)
31         leaves.append(TreeNode(val, string))
32     root = build(leaves)
33
34     def str_to_code(root):
35         last = {root : ""}
36         while last:
37             new = {}
38             for n in last:
39                 if not n.left and not n.right:
40                     str_code[n.string] = last[n]
41                 if n.left:
42                     new[n.left] = last[n] + "0"
43                 if n.right:
44                     new[n.right] = last[n] + "1"
45             last = new
46
47     str_code = {}
48     str_to_code(root)
49
50     def code_to_str(code):
51         curr, pos, res = root, 0, []
52         while pos < len(code):
53             if not curr.left and not curr.right:
54                 res.append(curr.string)
55                 curr = root
56             if code[pos] == "0":
57                 curr = curr.left
58             else:
59                 curr = curr.right
60             pos += 1
61         res.append(curr.string)

```



```

62     return "".join(res)
63
64 while True:
65     try:
66         s = input()
67         if s[0] in {"0", "1"}:
68             print(code_to_str(s))
69         else:
70             res = [str_code[c] for c in s]
71             print(*res, sep = "")
72     except EOFError:
73         break

```

#48933131提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

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

```

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

```

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.