

Assignment #9: Huffman, BST & Heap

Updated 1834 GMT+8 Apr 15, 2025

2025 spring, Compiled 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：

```
1  from typing import *
2  from math import pow
3  # Definition for a binary tree node.
4  class TreeNode:
5      def __init__(self, val=0, left=None, right=None):
6          self.val = val
7          self.left = left
8          self.right = right
9  class Solution:
10     def countNodes(self, root: Optional[TreeNode]) -> int:
11         self.cnt = self.h = 0
12         self.h = 0
13         def dfs(node, h):
14             if not node:
15                 self.cnt += 1
16                 if not self.h:
17                     self.h = h
18             return
19             if self.h and h == self.h and node:
20                 raise ValueError
21             dfs(node.right, h + 1)
22             dfs(node.left, h + 1)
23         try:
24             dfs(root, 0)
25         except ValueError:
26             pass
```

Python

```

27         return int(pow(2, self.h + 1)) - self.cnt - 1
28
29
30
31 if __name__ == "__main__":
32     sol = Solution()
33     print(sol.countNodes(TreeNode(1, TreeNode(2, TreeNode(4), TreeNode(5)), TreeNode(3, TreeNode(6)))))
34

```

代码运行截图 (至少包含有"Accepted")

The screenshot displays a LeetCode submission interface. On the left, the submission status is '通过' (Accepted) with a runtime of 4 ms and a memory usage of 22.93 MB. A bar chart shows the execution time distribution. The right panel shows the code in Python3, which uses a recursive DFS approach to count nodes in a binary tree.

Code (Python3):

```

# 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:
        if not root:
            return 0
        return 1 + self.countNodes(root.left) + self.countNodes(root.right)

```

Test Results:

通过 执行用时: 0 ms

Case 1: Input: [1,2,3,4,5,6], Output: 6, Expected: 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:

- 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:

```

1 from typing import *
2 from collections import deque
3 # Definition for a binary tree node.
4 class TreeNode:
5     def __init__(self, val=0, left=None, right=None):
6         self.val = val
7         self.left = left
8         self.right = right
9 class Solution:
10     def zigzagLevelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:
11         queue = deque([(root, 0)])
12         ans = []
13         lasth = -1
14         while queue:
15             node, h = queue.popleft()
16             if not node:
17                 continue

```

Python

```

18         if h != lasth:
19             lasth = h
20             if not h % 2:
21                 ans[-1].reverse()
22                 ans.append([])
23                 ans[-1].append(node.val)
24                 queue.append((node.left, h + 1))
25                 queue.append((node.right, h + 1))
26             if not h % 2:
27                 ans[-1].reverse()
28             return ans[1:]
29
30 if __name__ == "__main__":
31     sol = Solution()
32     print(sol.zigzagLevelOrder(TreeNode(3, TreeNode(9), TreeNode(20, TreeNode(15), TreeNode(7)))))
33

```

代码运行截图（至少包含有"Accepted"）

The screenshot displays a coding platform interface with two main panels. The left panel shows the problem details, including the title '通过 33 / 33 个通过的测试用例', the user 'Elated PaynePRR' who submitted on '2025.04.17 17:04', and a performance graph. The graph shows a single bar at 3ms with a 7.48% hit rate. The right panel shows the code editor with Python3 code for a zigzag level order traversal. Below the code, the '测试结果' (Test Results) section shows '通过' (Passed) with an execution time of 0ms. The input is 'root = [3,9,20,null,null,15,7]' and the output is '[[3],[20,9],[15,7]]', which matches the expected result.

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：

```

1 from heapq import heappush, heappop
2 ans = 0
3 class TreeNode:
4     def __init__(self, val, left=None, right=None):
5         self.val = val
6         self.left = left
7         self.right = right
8     def __lt__(self, other):
9         return self.val < other.val
10

```

Python

```

11 def calcWeighedLength(node, h):
12     if node and not node.left and not node.right:
13         global ans
14         ans += h * node.val
15         return
16     calcWeighedLength(node.left, h + 1)
17     calcWeighedLength(node.right, h + 1)
18 heap = []
19 n = int(input())
20 weight = map(int, input().split())
21 for i in weight:
22     heappush(heap, TreeNode(i))
23 while len(heap) >= 2:
24     node1 = heappop(heap)
25     node2 = heappop(heap)
26     node = TreeNode(node1.val + node2.val, node1, node2)
27     heappush(heap, node)
28 calcWeighedLength(node, 0)
29 print(ans)
30

```

代码运行截图（至少包含有"Accepted"）

OpenJudge
题目ID, 标题, 描述
24n2400011125
信箱
账号


CS101 / 题库（包括计概、数理题目）

题目
排名
状态
提问

#48940651提交状态

查看 提交 统计 提问

状态: **Accepted**

源代码

```

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

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)

```

基本信息

#: 48940651
题目: 04080
提交人: 颜鼎堃(24n2400011125)
内存: 3648kB
时间: 20ms
语言: Python3
提交时间: 2025-04-17 19:43:24

©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:

- 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:

```

1 from collections import deque
2 from sys import stdin
3 class TreeNode:
4     def __init__(self, val, left=None, right=None):
5         self.val = val

```

Python

```

6         self.left = left
7         self.right = right
8
9     def insert(n, p):
10         if n < p.val:
11             if p.left:
12                 insert(n, p.left)
13             else:
14                 p.left = TreeNode(n)
15         elif n > p.val:
16             if p.right:
17                 insert(n, p.right)
18             else:
19                 p.right = TreeNode(n)
20
21 data = list(map(int, stdin.read().split()))
22 head = TreeNode(data[0])
23 for num in data[1:]:
24     insert(num, head)
25 queue = deque([head])
26 ans = []
27 while queue:
28     node = queue.popleft()
29     if not node:
30         continue
31     ans.append(str(node.val))
32     queue.append(node.left)
33     queue.append(node.right)
34 print(*ans, sep=" ", end="")

```

代码运行截图（至少包含有"Accepted"）

OpenJudge

题目ID, 标题, 描述

24n2400011125

信箱

账号

CS101 / 题库（包括计概、数算题目）

题目

排名

状态

提问

#48940754提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```

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="")

```

基本信息

#:

48940754

题目:

05455

提交人:

颜鼎堃(24n2400011125)

内存:

3664kB

时间:

21ms

语言:

Python3

提交时间:

2025-04-17 19:58:45

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

```
1 18
2 1 41
3 1 43
4 1 85
5 1 24
6 2
7 1 43
8 1 54
9 2
10 1 20
11 1 54
12 1 89
13 2
14 1 58
15 2
16 2
17 2
18 2
19 2
```

unkown language

- 输出

```
1 24
2 41
3 20
4 43
5 43
6 54
7 54
8 58
```

unkown language

代码:

```
1 class heap:
2     def __init__(self):
3         self.L = []
4     def heappush(self, u):
5         self.L.append(u)
6         ind = len(self.L) - 1
7         head = (ind - 1) // 2
8         while ind and self.L[ind] < self.L[head]:
9             oth = 4 * ((head - 1) // 2) - head + 3
10            if head and self.L[head] < self.L[oth]:
11                self.L[head], self.L[oth] = self.L[oth], self.L[head]
12            self.L[ind], self.L[head] = self.L[head], self.L[ind]
13            ind = head
14            head = (ind - 1) // 2
15     def heappop(self):
16         ind = len(self.L) - 1
17         path = []
18         while ind:
19             path.append(ind)
20             ind = (ind - 1) // 2
21         path.reverse()
22         ind = 0
```

Python

```

23     for sub in path:
24         oth = 4 * ind - sub + 3
25         if oth < len(self.L) and self.L[sub] > self.L[oth]:
26             self.L[sub], self.L[oth] = self.L[oth], self.L[sub]
27         while True:
28             a = (2*oth + 2 < len(self.L) - 1) and self.L[oth] > self.L[2*oth + 2]
29             b = (2*oth + 1 < len(self.L) - 1) and self.L[oth] > self.L[2*oth + 1]
30             if a:
31                 self.L[oth], self.L[2*oth + 2] = self.L[2*oth + 2], self.L[oth]
32                 c = 2 * oth + 2
33             if b:
34                 self.L[oth], self.L[2*oth + 1] = self.L[2*oth + 1], self.L[oth]
35                 c = 2 * oth + 1
36             if a or b:
37                 oth = c
38             else:
39                 break
40
41         self.L[ind], self.L[sub] = self.L[sub], self.L[ind]
42         ind = sub
43     return self.L.pop()
44
45 h = heap()
46 for i in range(int(input())):
47     p = input().split()
48     if len(p) == 1:
49         print(h.heappop())
50     else:
51         h.heappush(int(p[1]))
52

```

代码运行截图 (至少包含有"Accepted")

OpenJudge

题目ID, 标题, 描述

24n2400011125 信箱 账号

CS101 / 题库 (包括计概、数学题目)

题目

排名

状态

提问

#48941436提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```

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()

```

基本信息

#: 48941436

题目: 04078

提交人: 颜鼎盛(24n2400011125)

内存: 4652kB

时间: 723ms

语言: Python3

提交时间: 2025-04-17 21:18:57

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:

```
1  from heapq import heappop, heappush
2  code = {}
3  decode = {}
4  heap = []
5  class TreeNode:
6      def __init__(self, ch, freq, minch="", left=None, right=None):
7          self.ch = ch
8          self.freq = freq
9          self.left = left
10         self.right = right
11         self.minch = list(ch)[0] if not minch else minch
12     def __lt__(self, other):
13         return (self.freq, self.minch) < (other.freq, other.minch)
14     def __add__(self, other):
15         return TreeNode(
16             self.ch | other.ch,
17             self.freq + other.freq,
18             min(self.minch, other.minch),
19             self if self < other else other,
20             other if self < other else self
21         )
22
23 def encode(node, prev):
24     if not node:
25         return
26     if not node.left and not node.right:
27         c = list(node.ch)[0]
28         code[c] = prev
29         decode[prev] = c
30         return
31     encode(node.left, prev + "0")
32     encode(node.right, prev + "1")
33
34 def num_to_str(get):
35     ans = ""
36     lst = 0
37     for i in range(1, len(get) + 1):
38         if get[lst:i] in decode:
39             ans += decode[get[lst:i]]
40             lst = i
41     return ans
42
43 def str_to_num(get):
44     ans = ""
45     for c in get:
46         ans += code[c]
47     return ans
48
49 for i in range(int(input())):
50     c, f = input().split()
51     heappush(heap, TreeNode({c}, int(f)))
52 while len(heap) > 1:
53     n1 = heappop(heap)
54     n2 = heappop(heap)
55     node = n1 + n2
56     heappush(heap, node)
57 head = node
58 encode(head, "")
```

Python


```
59 while True:
60     try:
61         get = input()
62         if get[0] in {"0", "1"}:
63             print(num_to_str(get))
64         else:
65             print(str_to_num(get))
66     except EOFError:
67         break
```

代码运行截图 (至少包含有"Accepted")

OpenJudge

题目ID, 标题, 描述

24n2400011125 信箱 账号

CS101 / 题库 (包括计概、数学题目)

题目

排名

状态

提问

#48941800提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
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")
```

基本信息

#: 48941800

题目: 22161

提交人: 颜鼎堃(24n2400011125)

内存: 3724kB

时间: 21ms

语言: Python3

提交时间: 2025-04-17 22:00:06

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