# Assignment #9: Huffman, BST & Heap

Updated 1834 GMT+8 Apr 15, 2025

2025 spring, Complied by 同学的姓名、院系

> **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：

```
# Definition for a binary tree node.
# class TreeNode(object):
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution(object):
    def countNodes(self, root):
        """
        :type root: Optional[TreeNode]
        :rtype: int
        """
        if not root:
            return 0
        def depth(node,record):
```

```
            depth=0
            while node:
                if record=='left':
                    depth+=1
                    node=node.left
                else:
                    depth+=1
                    node=node.right
            return depth
        lefth,righth=depth(root,'left'),depth(root,'right')
        if lefth==righth:
            return 2**lefth-1
        else:
            return self.countNodes(root.left)+self.countNodes(root.right)+1
```

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：

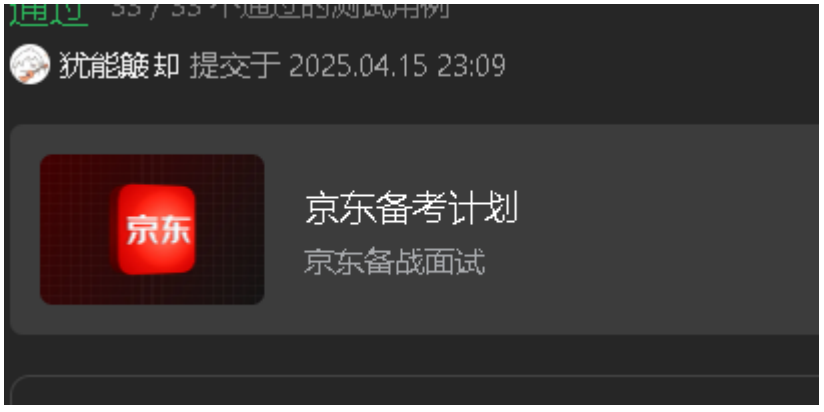
和层序遍历一样，最后相应层数翻转一下就好了

代码：

```
# Definition for a binary tree node.
# class TreeNode(object):
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution(object):
    def zigzagLevelOrder(self, root):
        """
        :type root: Optional[TreeNode]
        :rtype: List[List[int]]
        """
```

```
        if not root:
            return []
        out=[]
        out.append([root.val])
        q=deque([(root,0)])
        while q:
            node,cnt=q.popleft()
            if node.left:
                q.append((node.left,cnt+1))
                if len(out)<=cnt+1:
                    out.append([node.left.val])
                else:
                    out[cnt+1].append(node.left.val)
            if node.right:
                q.append((node.right,cnt+1))
                if len(out)<=cnt+1:
                    out.append([node.right.val])
                else:
                    out[cnt+1].append(node.right.val)
        for i in range(1,len(out),2):
            out[i].reverse()
        return out
```

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：

代码：

```
import heapq
n=int(input())
weight=[int(x) for x in input().split()]
heapq.heapify(weight)
ans=0
while len(weight)>1:
    a=heapq.heappop(weight)
    b=heapq.heappop(weight)
    combine=a+b
    ans+=combine
    heapq.heappush(weight,combine)
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：列表转集合再转列表后顺序有可能打乱啊

代码：

```
from collections import deque
class TreeNode:
    def __init__(self,val=0,left=None,right=None):
```

```
        self.val=val
        self.left=left
        self.right=right

def build(root,insertval):
    insert=TreeNode(insertval)
    if insertval>root.val:
        if root.right:
            build(root.right,insertval)
        else:
            root.right=insert
    if insertval<root.val:
        if root.left:
            build(root.left,insertval)
        else:
            root.left=insert
    return root

def bfs(root):
    ans=[]
    q=deque([root])
    while q:
        node=q.popleft()
        ans.append(node.val)
        if node.left:
            q.append(node.left)
        if node.right:
            q.append(node.right)
    return ans

s=[int(x) for x in input().split()]
root=TreeNode(s[0])
record=set()
record.add(s[0])
for i in s[1:]:
    if i in record:
        continue
    root=build(root,i)
    record.add(i)
ans=bfs(root)
print(*ans)
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

```
from collections import deque
class TreeNode:
    def __init__(self,val=0,left=None,right=None):
        self.val=val
        self.left=left
        self.right=right

def build(root,insertval):
    insert=TreeNode(insertval)
    if insertval>root.val:
        if root.right:
            build(root.right,insertval)
        else:
            root.right=insert
    if insertval<root.val:
        if root.left:
```

## 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：手搓堆还是有点困难.....

代码：

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

    def getup(self,i):
        while (i-1)//2>=0:   #当前父节点
            parent=(i-1)//2
            if self.pq[i]<self.pq[parent]:
                self.pq[i],self.pq[parent]=self.pq[parent],self.pq[i]
            i=parent

    def insert(self,i):
        self.pq.append(i)
        self.getup(len(self.pq)-1)

    def getdown(self,i):
        while 2*i+1<len(self.pq):
```

```
            child=self.getmin(i)
            if self.pq[i]>self.pq[child]:
                self.pq[i],self.pq[child]=self.pq[child],self.pq[i]
            else:
                break
            i=child

    def getmin(self,i):
        if 2*i+2>len(self.pq)-1:
            return 2*i+1
        if self.pq[2*i+1]<self.pq[2*i+2]:
            return 2*i+1
        return 2*i+2

    def delete(self):
        self.pq[0],self.pq[-1]=self.pq[-1],self.pq[0]
        ans=self.pq.pop()
        self.getdown(0)
        return ans

n=int(input())
pq=heap()
for _ in range(n):
    s=[int(x) for x in input().split()]
    if len(s)==2:
        pq.insert(s[1])
    else:
        print(pq.delete())
```

代码运行截图 （至少包含有"Accepted"）

## 状态：Accepted

源代码

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

    def getup(self,i):
        while (i-1)//2>=0:    #当前父节点
            parent=(i-1)//2
            if self.pq[i]<self.pq[parent]:
                self.pq[i],self.pq[parent]=self.pq[parent],self.pq[i]
            i=parent
```

## T22161: 哈夫曼编码树

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

思路：

代码：

```
import heapq
class Node:
    def __init__(self,weight,char=None):
        self.weight=weight
        self.char=char
        self.left=None
        self.right=None

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

def build_huffman(character):
    pq=[]
    for char,weight in character.items():
        heapq.heappush(pq,Node(weight,char))
    while len(pq)>1:
        left=heapq.heappop(pq)
        right=heapq.heappop(pq)
        combine=Node(left.weight+right.weight)
        combine.left=left
        combine.right=right
        heapq.heappush(pq,combine)
    return pq[0]

def encode(root):
    codes={}
    def traverse(node,code):
        if node.char:
            codes[node.char]=code
        else:
            traverse(node.left,code+'0')
            traverse(node.right,code+'1')
    traverse(root,'')
    return codes

def encoding(codes,string):
    encoded=''
    for char in string:
        encoded+=codes[char]
    return encoded

def decoding(root,string):
```

```
    decoded=''
    node=root
    for i in string:
        if i=='0':
            node=node.left
        else:
            node=node.right
        if node.char:
            decoded+=node.char
            node=root
    return decoded

n=int(input())
character={}
for _ in range(n):
    char,wei=input().split()
    character[char]=int(wei)
huffman=build_huffman(character)
codes=encode(huffman)
while True:
    try:
        string=input()
        if string[0] in ('0','1'):
            print(decoding(huffman,string))
        else:
            print(encoding(codes,string))
    except EOFError:
        break
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

```
import heapq
class Node:
    def __init__(self,weight,char=None):
        self.weight=weight
        self.char=char
        self.left=None
        self.right=None

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

def build_huffman(character):
    pq=[]
    for char,weight in character.items():
```

## 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。

树的题目做多了感觉还挺有套路的，但树里面的概念有点多，有些写题目时弄混了

哈夫曼编码感觉还不是很能理解