Assignment #6: "树"算：Huffman,BinHeap,BST,AVL,DisjointSet

Updated 2214 GMT+8 March 24, 2024

2024 spring, Complied by Xinjie Song, Phy

说明：

1）这次作业内容不简单，耗时长的话直接参考题解。

2）请把每个题目解题思路（可选），源码Python, 或者C++（已经在Codeforces/Openjudge上AC），截图（包含Accepted），填写到下面作业模版中（推荐使用 typora https://typoraio.cn ，或者用word）。AC 或者没有AC，都请标上每个题目大致花费时间。

3）提交时候先提交pdf文件，再把md或者doc文件上传到右侧“作业评论”。Canvas需要有同学清晰头像、提交文件有pdf、"作业评论"区有上传的md或者doc附件。

4）如果不能在截止前提交作业，请写明原因。

编程环境

操作系统：Windows 11 22H2

Python编程环境：PyCharm 2023.2 (Community Edition)

C/C++编程环境：g++ (x86_64-win32-seh-rev0, Built by MinGW-W64 project) 8.1.0

1. 题目

22275: 二叉搜索树的遍历

http://cs101.openjudge.cn/practice/22275/

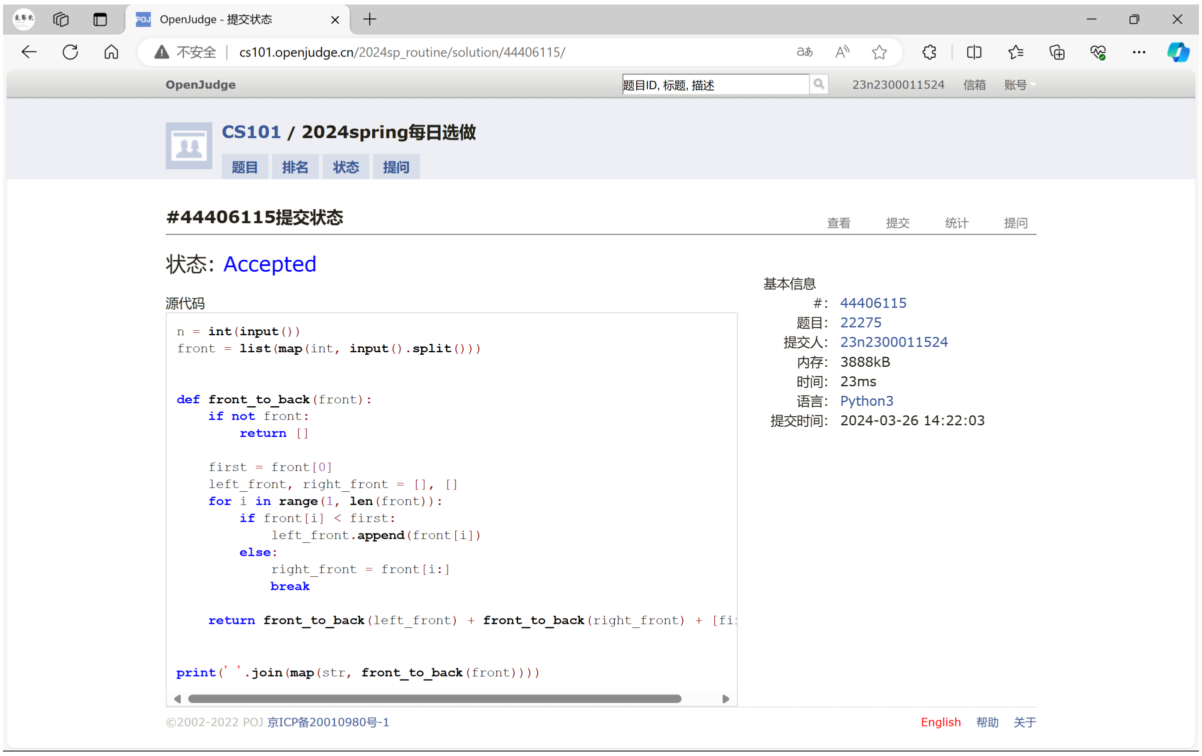
思路：递归拆分

代码

```
n = int(input())
front = list(map(int, input().split()))


def front_to_back(front):
    if not front:
        return []

    first = front[0]
    left_front, right_front = [], []
```

```python
    for i in range(1, len(front)):
        if front[i] < first:
            left_front.append(front[i])
        else:
            right_front = front[i:]
            break

    return front_to_back(left_front) + front_to_back(right_front) + [first]


print(' '.join(map(str, front_to_back(front))))
```

代码运行截图



#44406115提交状态

状态: Accepted

源代码

```python
n = int(input())
front = list(map(int, input().split()))


def front_to_back(front):
    if not front:
        return []

    first = front[0]
    left_front, right_front = [], []
    for i in range(1, len(front)):
        if front[i] < first:
            left_front.append(front[i])
        else:
            right_front = front[i:]
            break

    return front_to_back(left_front) + front_to_back(right_front) + [fi


print(' '.join(map(str, front_to_back(front))))
```

基本信息

#:	44406115
题目:	22275
提交人:	23n2300011524
内存:	3888kB
时间:	23ms
语言:	Python3
提交时间:	2024-03-26 14:22:03

05455: 二叉搜索树的层次遍历

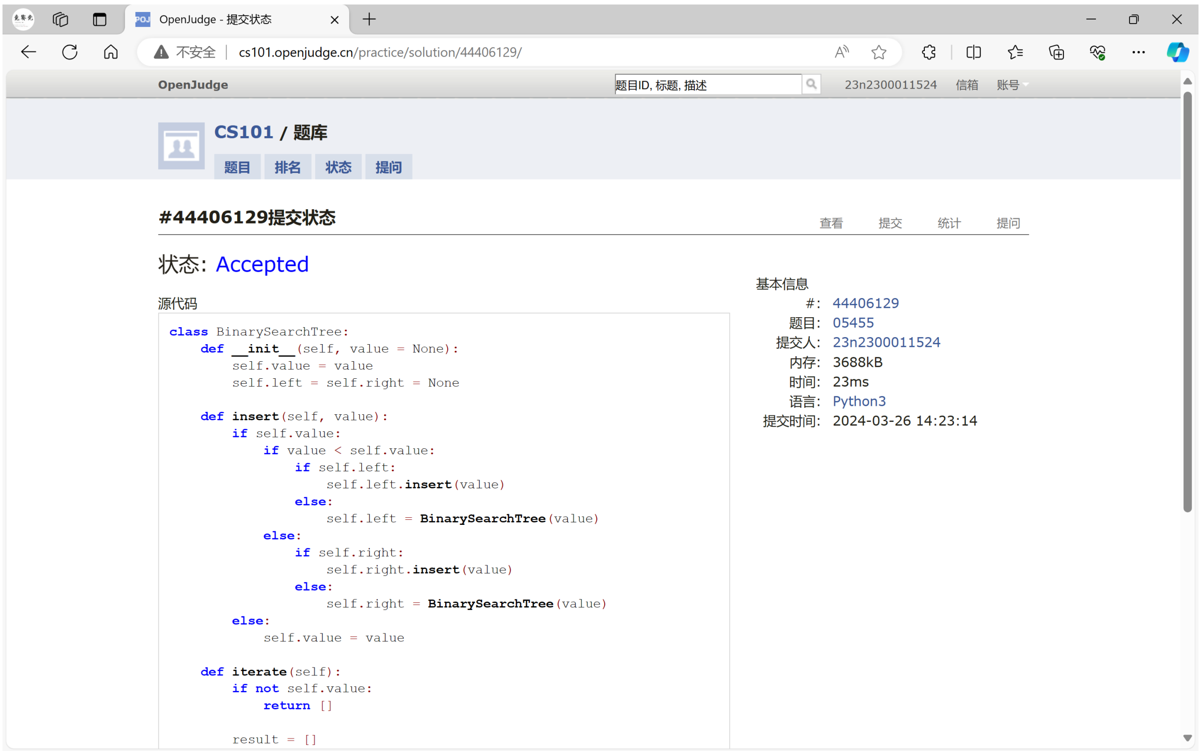
http://cs101.openjudge.cn/practice/05455/

思路：自定义类

代码

```python
class BinarySearchTree:
    def __init__(self, value = None):
        self.value = value
        self.left = self.right = None

    def insert(self, value):
        if self.value:
            if value < self.value:
                if self.left:
                    self.left.insert(value)
                else:
                    self.left = BinarySearchTree(value)
            else:
                if self.right:
                    self.right.insert(value)
                else:
                    self.right = BinarySearchTree(value)
        else:
            self.value = value

    def iterate(self):
        if not self.value:
            return []

        result = []
        if self.left:
            result = self.left.iterate()
        if self.right:
            right = self.right.iterate()
            for i in range(min(len(result), len(right))):
                result[i] = ' '.join([result[i], right[i]])
            result += right[len(result):]
        return [f'{self.value}'] + result


tree = BinarySearchTree()
dic = {}
for i in input().split():
    if i in dic:
        continue
    tree.insert(int(i))
    dic[i] = True
print(' '.join(tree.iterate()))
```

代码运行截图



#44406129提交状态

状态: Accepted

源代码

```
class BinarySearchTree:
    def __init__(self, value = None):
        self.value = value
        self.left = self.right = None

    def insert(self, value):
        if self.value:
            if value < self.value:
                if self.left:
                    self.left.insert(value)
                else:
                    self.left = BinarySearchTree(value)
            else:
                if self.right:
                    self.right.insert(value)
                else:
                    self.right = BinarySearchTree(value)
        else:
            self.value = value

    def iterate(self):
        if not self.value:
            return []

        result = []
```

基本信息

#:	44406129
题目:	05455
提交人:	23n2300011524
内存:	3688kB
时间:	23ms
语言:	Python3
提交时间:	2024-03-26 14:23:14

04078: 实现堆结构

http://cs101.openjudge.cn/practice/04078/

练习自己写个BinHeap。当然机考时候，如果遇到这样题目，直接import heapq。手搓栈、队列、堆、AVL等，考试前需要搓个遍。

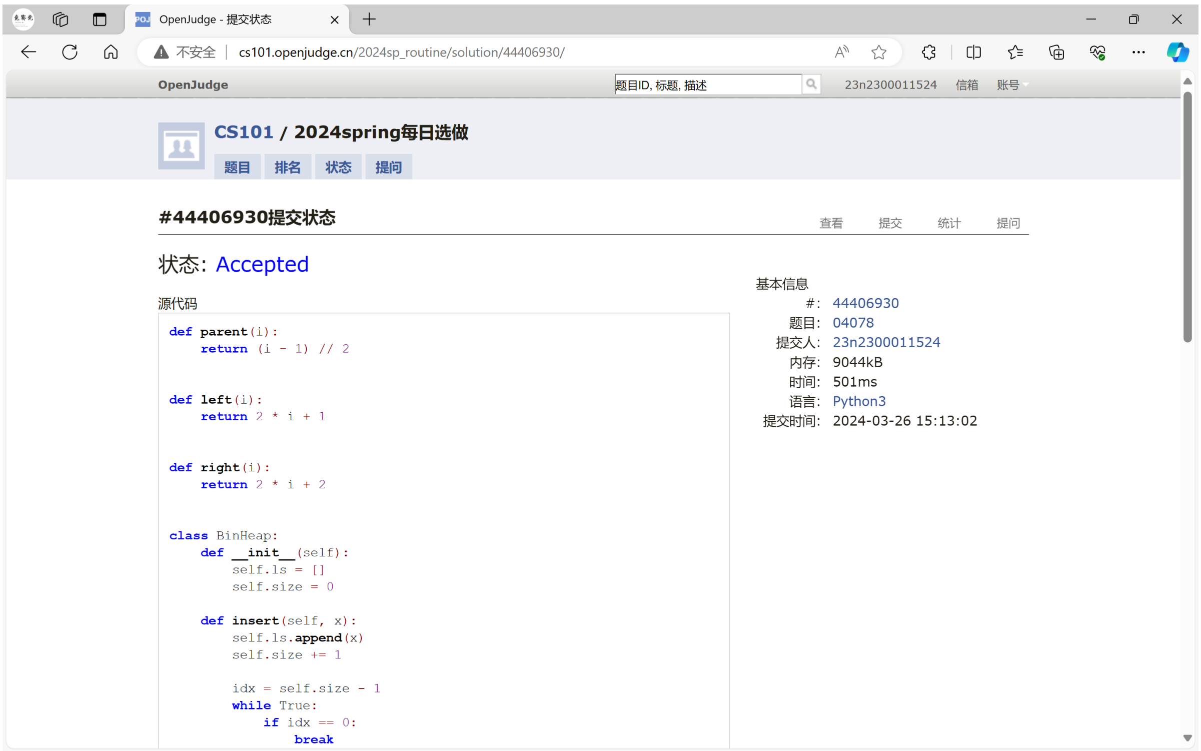
思路：统一输出节约时间

代码

```
def parent(i):
    return (i - 1) // 2


def left(i):
    return 2 * i + 1


def right(i):
    return 2 * i + 2


class BinHeap:
    def __init__(self):
        self.ls = []
        self.size = 0

    def insert(self, x):
        self.ls.append(x)
        self.size += 1

        idx = self.size - 1
        while True:
            if idx == 0:
                break

            p = parent(idx)
            if self.ls[p] > self.ls[idx]:
                self.ls[p], self.ls[idx] = self.ls[idx], self.ls[p]
            else:
                break

            idx = p

    def pop(self):
        if self.size == 0:
            return None
        elif self.size == 1:
            self.size -= 1
            return self.ls.pop()

        s = self.ls[0]
        self.ls[0] = self.ls[-1]
        self.ls.pop()
        self.size -= 1

        idx = 0
        while True:
            l, r = left(idx), right(idx)

            if l < self.size and r < self.size:
                if self.ls[l] < self.ls[r]:
                    if self.ls[idx] > self.ls[l]:
                        self.ls[idx], self.ls[l] = self.ls[l], self.ls[idx]
                        idx = l
                    else:
                        break
                else:
                    if self.ls[idx] > self.ls[r]:
                        self.ls[idx], self.ls[r] = self.ls[r], self.ls[idx]
                        idx = r
                    else:
                        break
            elif l < self.size and self.ls[idx] > self.ls[l]:
                self.ls[idx], self.ls[l] = self.ls[l], self.ls[idx]
                idx = l
            elif r < self.size and self.ls[idx] > self.ls[r]:
                self.ls[idx], self.ls[r] = self.ls[r], self.ls[idx]
                idx = r
            else:
                break
        return s


ans = []
```

```python
heap = BinHeap()
for _ in range(int(input())):
    ls = input().split()
    if ls[0] == '2':
        ans.append(str(heap.pop()))
    else:
        heap.insert(int(ls[1]))
print('\n'.join(ans))
```

代码运行截图



#44406930提交状态

状态: Accepted

源代码

```python
def parent(i):
    return (i - 1) // 2


def left(i):
    return 2 * i + 1


def right(i):
    return 2 * i + 2


class BinHeap:
    def __init__(self):
        self.ls = []
        self.size = 0

    def insert(self, x):
        self.ls.append(x)
        self.size += 1

        idx = self.size - 1
        while True:
            if idx == 0:
                break
```

基本信息

#:	44406930
题目:	04078
提交人:	23n2300011524
内存:	9044kB
时间:	501ms
语言:	Python3
提交时间:	2024-03-26 15:13:02

22161: 哈夫曼编码树

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

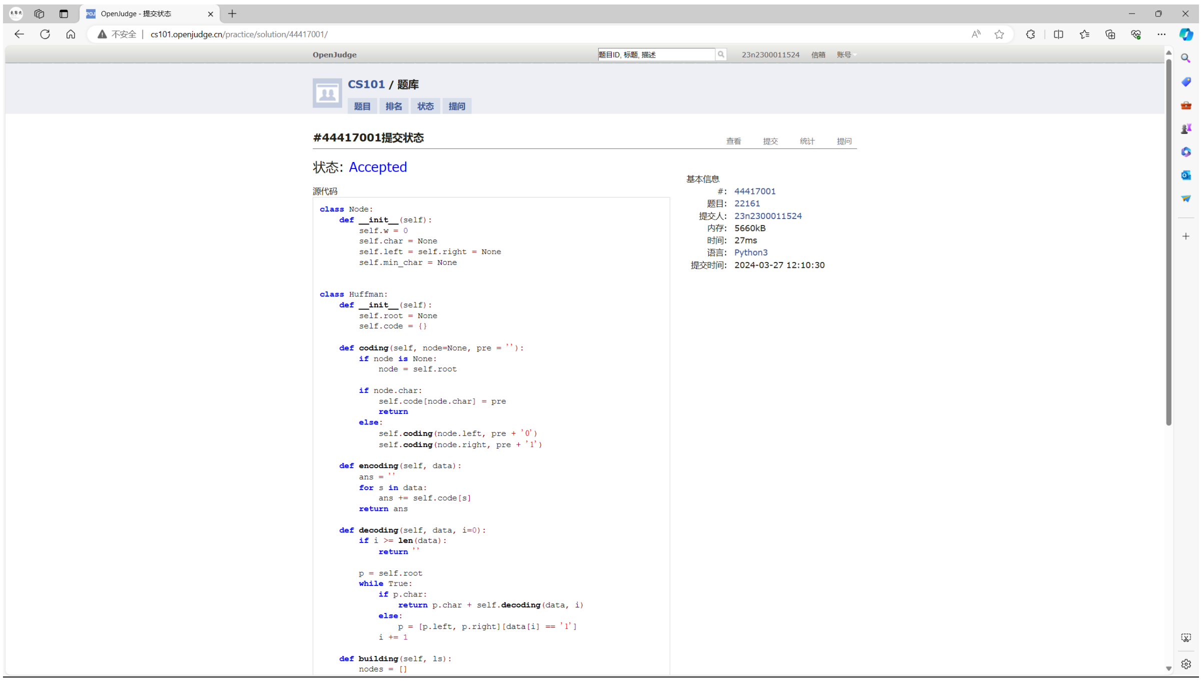
思路：无

代码

```python
class Node:
    def __init__(self):
        self.w = 0
        self.char = None
        self.left = self.right = None
        self.min_char = None


class Huffman:
    def __init__(self):
        self.root = None
        self.code = {}

    def coding(self, node=None, pre = ''):
        if node is None:
            node = self.root

        if node.char:
            self.code[node.char] = pre
            return
        else:
            self.coding(node.left, pre + '0')
            self.coding(node.right, pre + '1')

    def encoding(self, data):
        ans = ''
        for s in data:
            ans += self.code[s]
        return ans

    def decoding(self, data, i=0):
        if i >= len(data):
            return ''

        p = self.root
        while True:
            if p.char:
                return p.char + self.decoding(data, i)
            else:
                p = [p.left, p.right][data[i] == '1']
            i += 1

    def building(self, ls):
        nodes = []
        for c, w in ls:
            node = Node()
            node.char = node.min_char = c
            node.w = w
            nodes.append(node)

        for _ in range(len(ls) - 1):
            nodes.sort(key=lambda t: (t.w, t.min_char), reverse=True)
            l, r = nodes.pop(), nodes.pop()
            new_node = Node()
            new_node.left = l
            new_node.right = r
            new_node.w = l.w + r.w
            new_node.min_char = min(l.min_char, r.min_char)
            nodes.append(new_node)

        self.root = nodes[0]
        self.coding()


tree = Huffman()
ls = []
```

```
for _ in range(int(input())):
    c, w = input().split()
    ls.append((c, int(w)))
tree.building(ls)

while True:
    try:
        s = input()
    except EOFError:
        break
    if s[0] == '0' or s[0] == '1':
        print(tree.decoding(s))
    else:
        print(tree.encoding(s))
```

代码运行截图



#44417001提交状态

状态: Accepted

源代码

```
class Node:
    def __init__(self):
        self.w = 0
        self.char = None
        self.left = self.right = None
        self.min_char = None


class Huffman:
    def __init__(self):
        self.root = None
        self.code = {}

    def coding(self, node=None, pre = ''):
        if node is None:
            node = self.root

        if node.char:
            self.code[node.char] = pre
            return
        else:
            self.coding(node.left, pre + '0')
            self.coding(node.right, pre + '1')

    def encoding(self, data):
        ans = ''
        for s in data:
            ans += self.code[s]
        return ans

    def decoding(self, data, i=0):
        if i >= len(data):
            return ''

        p = self.root
        while True:
            if p.char:
                return p.char + self.decoding(data, i)
            else:
                p = [p.left, p.right][data[i] == '1']
            i += 1

    def building(self, ls):
        nodes = []
```

基本信息

#:	44417001
题目:	22161
提交人:	23n2300011524
内存:	5660kB
时间:	27ms
语言:	Python3
提交时间:	2024-03-27 12:10:30

晴问9.5: 平衡二叉树的建立

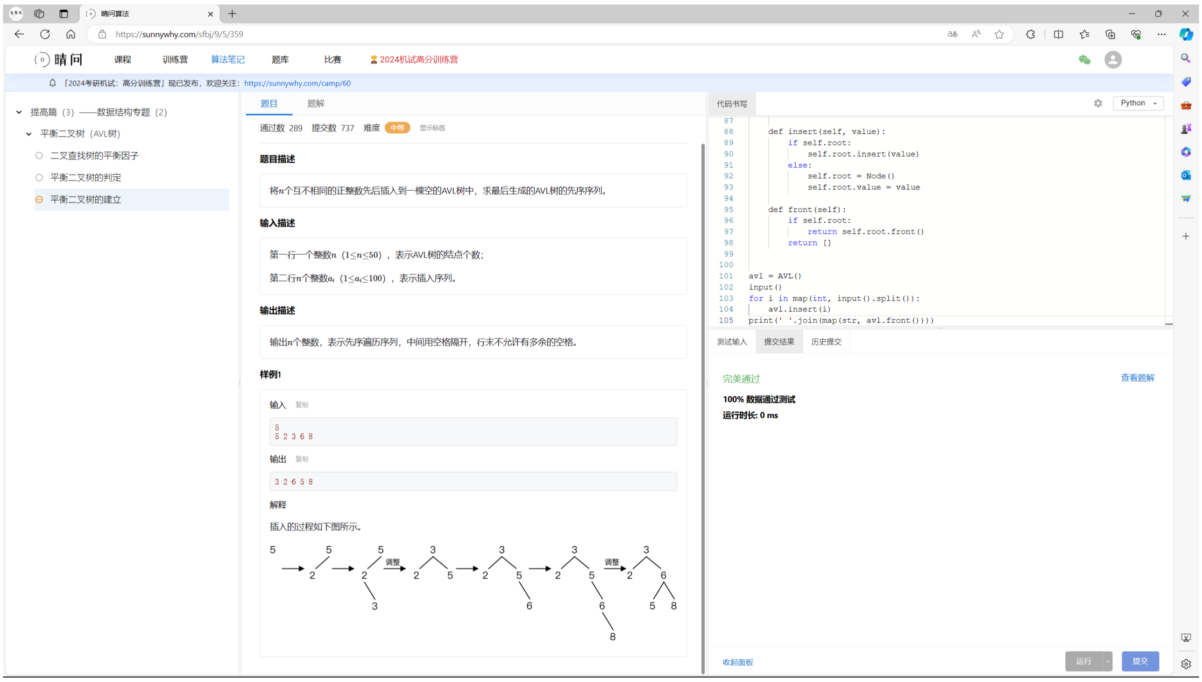
https://sunnywhy.com/sfbj/9/5/359

思路：无

代码

```
class Node:
    def __init__(self):
        self.value = None
        self.left = None
        self.right = None
        self.bal = 0
    def has_left(self):
        return self.left is not None

    def has_right(self):
        return self.right is not None

    def insert(self, value):
        if value > self.value:
            if self.has_right():
                h = self.right.insert(value)
                h = 0 if h == 0 else -1
            else:
                self.right = Node()
                self.right.value = value
                h = -1
        else:
            if self.has_left():
                h = self.left.insert(value)
                h = 0 if h == 0 else 1
            else:
                self.left = Node()
                self.left.value = value
                h = 1
        self.bal += h
        if abs(self.bal) == 2:
            self.adjust()
            return 0
        return h if h * (self.bal - h) >= 0 else 0

    def left_rotate(self):
        bal_self = max(self.bal + 1, self.right.bal) + 1
        bal_left = self.bal + 1 - min(0, self.right.bal)

        new = Node()
        new.value = self.value
        new.right = self.right.left
        new.left = self.left
        new.bal = bal_left

        self.left = new
        self.value = self.right.value
        self.right = self.right.right
        self.bal = bal_self

    def right_rotate(self):
        bal_self = min(self.bal - 1, self.left.bal) - 1
        bal_right = self.bal - 1 - max(0, self.left.bal)

        new = Node()
        new.value = self.value
        new.left = self.left.right
        new.right = self.right
        new.bal = bal_right

        self.right = new
        self.value = self.left.value
        self.left = self.left.left
        self.bal = bal_self

    def front(self):
        ans = [self.value]
        ans += self.left.front() if self.has_left() else []
        ans += self.right.front() if self.has_right() else []
        return ans

    def adjust(self):
        if self.bal < 0:
            if self.has_right() and self.right.bal > 0:
                self.right.right_rotate()
            self.left_rotate()
        else:
            if self.has_left() and self.left.bal < 0:
                self.left.left_rotate()
            self.right_rotate()


class AVL:
    def __init__(self):
        self.root = None

    def insert(self, value):
        if self.root:
            self.root.insert(value)
        else:
            self.root = Node()
            self.root.value = value

    def front(self):
        if self.root:
            return self.root.front()
        return []


avl = AVL()
input()
for i in map(int, input().split()):
    avl.insert(i)
print(' '.join(map(str, avl.front())))
```

代码运行截图



「2024考研机试：高分训练营」现已发布，欢迎关注：https://sunnywhy.com/camp/60

通过数 289 提交数 737 难度 中等 显示标签

题目描述

将n个互不相同的正整数先后插入到一棵空的AVL树中，求最后生成的AVL树的先序序列。

输入描述

第一行一个整数n（$1 \le n \le 50$），表示AVL树的结点个数；

第二行n个整数a_i（$1 \le a_i \le 100$），表示插入序列。

输出描述

输出n个整数，表示先序遍历序列，中间用空格隔开，行末不允许有多余的空格。

样例1

输入

```
5
5 2 3 6 8
```

输出

```
3 2 6 5 8
```

解释

插入的过程如下图所示。

```
    def insert(self, value):
        if self.root:
            self.root.insert(value)
        else:
            self.root = Node()
            self.root.value = value

    def front(self):
        if self.root:
            return self.root.front()
        return []


avl = AVL()
input()
for i in map(int, input().split()):
    avl.insert(i)
print(' '.join(map(str, avl.front())))
```

完美通过

100% 数据通过测试

运行时长: 0 ms

02524: 宗教信仰

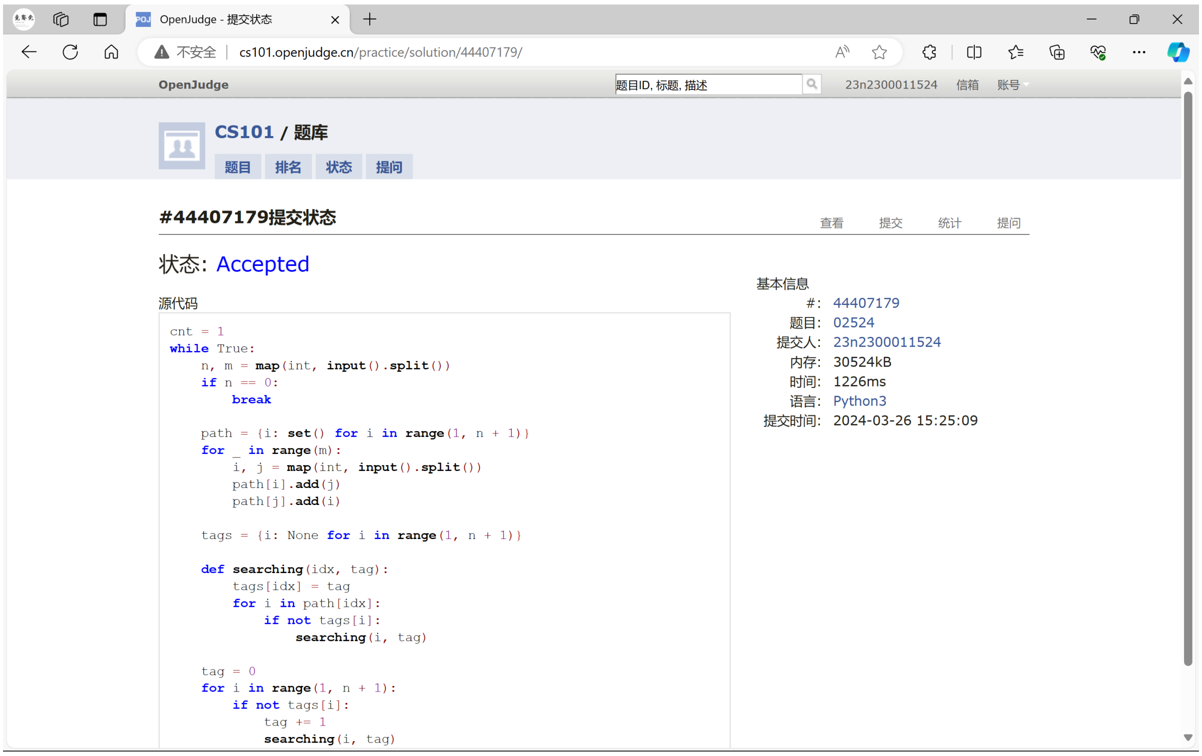
http://cs101.openjudge.cn/practice/02524/

思路：图搜索?

代码

```
cnt = 1
while True:
    n, m = map(int, input().split())
    if n == 0:
        break

    path = {i: set() for i in range(1, n + 1)}
    for _ in range(m):
        i, j = map(int, input().split())
        path[i].add(j)
        path[j].add(i)

    tags = {i: None for i in range(1, n + 1)}

    def searching(idx, tag):
        tags[idx] = tag
        for i in path[idx]:
            if not tags[i]:
                searching(i, tag)

    tag = 0
    for i in range(1, n + 1):
        if not tags[i]:
            tag += 1
            searching(i, tag)
    print(f'Case {cnt}: {tag}')
    cnt += 1
```

代码运行截图



#44407179提交状态

状态: Accepted

源代码

```
cnt = 1
while True:
    n, m = map(int, input().split())
    if n == 0:
        break

    path = {i: set() for i in range(1, n + 1)}
    for _ in range(m):
        i, j = map(int, input().split())
        path[i].add(j)
        path[j].add(i)

    tags = {i: None for i in range(1, n + 1)}

    def searching(idx, tag):
        tags[idx] = tag
        for i in path[idx]:
            if not tags[i]:
                searching(i, tag)

    tag = 0
    for i in range(1, n + 1):
        if not tags[i]:
            tag += 1
            searching(i, tag)
```

基本信息

#:	44407179
题目:	02524
提交人:	23n2300011524
内存:	30524kB
时间:	1226ms
语言:	Python3
提交时间:	2024-03-26 15:25:09

2. 学习总结和收获

除了AVL其他难度还可以，AVL最开始把自己转晕了，self.left = self写了个连通图出来，而且没有内置平衡因子，每次都要递归去算，效率低，遂推倒重来；重写时，旋转前后平衡因子的更新看过一次推到后自己再推还是比较容易的，插入同时更新平衡因子也想了好久。题解里似乎是把树的操作封装在了树里，并且有父节点的记录，估计是删除时候用？我是封在了节点里，没有记录父节点，删除时候用递归应该是可以的，改还是好改的，目前看没什么区别，就不改了。

祖传环节，让GPT写了首诗，自己又改了下。

平衡二叉搜索树，原理简单实现难，左旋右旋显神通，平衡维护在心中。
树高均衡求稳定，插入删除皆从容，红黑颜色交替舞，编码实现挑战重。
左重右轻把腰转，右重左轻身形动，维护平衡要谨慎，一步不慎全盘空。
代码实现路漫漫，逻辑严密慎思量，平衡之美在指尖，舞动诠释算法魂。
平衡二叉搜索树，左右旋转化危机，编码实现虽艰难，智慧结晶显神机。