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

Updated 2214 GMT+8 March 24, 2024

2024 spring, Complied by 李鹏辉，元培学院

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

Windows 10 Home, PyCharm 2022.3.2 (Community Edition)

操作系统：macOS Ventura 13.4.1 (c)

Python编程环境：Spyder IDE 5.2.2, PyCharm 2023.1.4 (Professional Edition)

C/C++编程环境：Mac terminal vi (version 9.0.1424), g++/gcc (Apple clang version 14.0.3, clang-1403.0.22.14.1)

1. 题目

22275: 二叉搜索树的遍历

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

思路：about 30 mins.

代码

```
class Tree:
    def __init__(self, value):
        self.value = value
        self.leaves = []

    def print_format(self):
        re = [self.value]
```

```
        for leaf in self.leaves:
            re.extend(leaf.print_format())
        return re

    def __str__(self):
        return ' '.join(map(str, self.print_format()))


class BTree(Tree):
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

    @property
    def leaves(self):
        if self.left and self.right: return [self.left, self.right]
        elif self.left: return [self.left]
        elif self.right: return [self.right]
        else: return []


def q1():
    def helper(num_list):
        if not num_list:
            return None
        value = num_list[0]
        left_nums = [a for a in num_list if a < value]
        right_nums = [b for b in num_list if b > value]
        t = BTree(value)
        t.left = helper(left_nums)
        t.right = helper(right_nums)
        return t

    def post_order(t):
        re = []
        for leaf in t.leaves:
            re.extend(post_order(leaf))
        re.append(t.value)
        return re

    input()
    num_list = list(map(int, input().split()))
    print(' '.join(map(str, post_order(helper(num_list)))))


q1()
```

代码运行截图 （至少包含有"Accepted"）

#44502049提交状态 查看 提交 统计

状态: Accepted

源代码

```
class Tree:
    def __init__(self, value):
        self.value = value
        self.leaves = []

    def print_format(self):
```

基本信息

#: 44502049
题目: 22275
提交人: 2100017777
内存: 4052kB
时间: 28ms
语言: Python3
提交时间: 2024-04-02 00:17:56

05455: 二叉搜索树的层次遍历

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

思路：it keeps the fundamental structure of question 1 and only modifies the helper function of output. 12 mins.

代码

```
class Tree:
    def __init__(self, value):
        self.value = value
        self.leaves = []

    def print_format(self):
        re = [self.value]
        for leaf in self.leaves:
            re.extend(leaf.print_format())
        return re

    def __str__(self):
        return ' '.join(map(str, self.print_format()))


class BTree(Tree):
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

    @property
    def leaves(self):
        if self.left and self.right: return [self.left, self.right]
        elif self.left: return [self.left]
        elif self.right: return [self.right]
        else: return []


def q2():
    def helper(num_list):
        if not num_list:
```

```
            return None
        value = num_list[0]
        left_nums = [a for a in num_list if a < value]
        right_nums = [b for b in num_list if b > value]
        t = BTree(value)
        t.left = helper(left_nums)
        t.right = helper(right_nums)
        return t

    def bfs(t):
        re = []
        q = [t]
        while q:
            node = q.pop(0)
            re.append(node.value)
            q.extend(node.leaves)
        return re

    raw_num_list = list(map(int, input().split()))
    num_list = []
    for num in raw_num_list:
        if num not in num_list:
            num_list.append(num)
    print(' '.join(map(str, bfs(helper(num_list)))))


q2()
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

```
class Tree:
    def __init__(self, value):
        self.value = value
        self.leaves = []

    def print_format(self):
```

基本信息

#:	44502093
题目:	05455
提交人:	2100017777
内存:	3716kB
时间:	24ms
语言:	Python3
提交时间:	2024-04-02 00:29:14

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：about 30 mins.

代码

```
def q3():
```

```
    h = [0]
    size = 0

    def insert_h(n):
        nonlocal size
        size += 1
        i = size
        h.append(n)
        while i > 1:
            new_i = i // 2
            if h[i] < h[new_i]:
                h[i], h[new_i] = h[new_i], h[i]
                i //= 2
            else: break

    def delete_h():
        nonlocal size
        size -= 1
        print(h[1])
        h[1] = h[-1]
        h.pop()
        if size == 1: return
        i = 1
        while i * 2 <= size:
            left = i * 2
            right = left + 1
            if right > size:
                min_i = left
            else:
                if h[left] < h[right]:
                    min_i = left
                else:
                    min_i = right
            if h[i] > h[min_i]:
                h[i], h[min_i] = h[min_i], h[i]
                i = min_i
            else: break

    n = int(input())
    for _ in range(n):
        num_list = list(map(int, input().split()))
        if num_list[0] == 1:
            insert_h(num_list[1])
        else:
            delete_h()


q3()
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

状态: Accepted

源代码

```
def q3():
    h = [0]
    size = 0

    def insert_h(n):
        nonlocal size
```

基本信息

#:	44506552
题目:	04078
提交人:	2100017777
内存:	4672kB
时间:	501ms
语言:	Python3
提交时间:	2024-04-02 16:13:48

22161: 哈夫曼编码树

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

思路：about 1 hour.

代码

```python
import heapq as h


class Tree:
    def __init__(self, value):
        self.value = value
        self.leaves = []

    def print_format(self):
        re = [self.value]
        for leaf in self.leaves:
            re.extend(leaf.print_format())
        return re

    def __str__(self):
        return ' '.join(map(str, self.print_format()))


class BTree(Tree):
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

    @property
    def leaves(self):
        if self.left and self.right: return [self.left, self.right]
        elif self.left: return [self.left]
        elif self.right: return [self.right]
        else: return []


class HTree(BTree):
    def __init__(self, value, weight):
        super().__init__(value)
```

```
        self.weight = weight

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


def q4():
    def huffman(vnf):  # values and freqs
        heap = []
        for value, freq in vnf:
            h.heappush(heap, HTree(value, freq))
        while len(heap) > 1:
            left = h.heappop(heap)
            right = h.heappop(heap)
            new_tree = HTree(None, left.weight + right.weight)
            new_tree.left = left
            new_tree.right = right
            h.heappush(heap, new_tree)
        return heap[0]

    encode_dict = {}

    def encode(t, chars):

        def encode_helper(t, code=''):
            if not t.leaves:
                encode_dict[t.value] = code
            if t.left:
                encode_helper(t.left, code + '0')
            if t.right:
                encode_helper(t.right, code + '1')

        if not encode_dict:
            encode_helper(t)
        re = ''
        for char in chars:
            re += encode_dict[char]
        return re

    def decode(t, code):
        re = ''

        def decode_helper(c_t, c_code):  # current_tree/code
            nonlocal re
            if c_t.value:
                re += c_t.value
                if c_code:
                    decode_helper(t, c_code)
                else: return
            else:
                direction = c_code[0]
                c_code = c_code[1:]
                if direction == '0':
                    decode_helper(c_t.left, c_code)
```

```
                else:
                    decode_helper(c_t.right, c_code)

        decode_helper(t, code)
        return re

    n = int(input())
    vnf = []
    for _ in range(n):
        v, f = input().split()
        vnf.append((v, int(f)))
    t = huffman(vnf)
    try:
        while True:
            i_s = input()
            if i_s.startswith('0') or i_s.startswith('1'):
                print(decode(t, i_s))
            else:
                print(encode(t, i_s))
    except EOFError:
        return


q4()
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

状态: Accepted

源代码

```
import heapq as h


class Tree:
    def __init__(self, value):
        self.value = value
```

基本信息

#:	44502231
题目:	22161
提交人:	2100017777
内存:	3800kB
时间:	27ms
语言:	Python3
提交时间:	2024-04-02 01:27:07

晴问9.5: 平衡二叉树的建立

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

思路：about 1 hour.

代码

```
class Tree:
    def __init__(self, value):
        self.value = value
        self.leaves = []

    def print_format(self):
        re = [self.value]
```

```
        for leaf in self.leaves:
            re.extend(leaf.print_format())
        return re

    def __str__(self):
        return ' '.join(map(str, self.print_format()))


class BTree(Tree):
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

    @property
    def leaves(self):
        if self.left and self.right: return [self.left, self.right]
        elif self.left: return [self.left]
        elif self.right: return [self.right]
        else: return []


class AVL(BTree):
    def __init__(self, value):
        super().__init__(value)

    @property
    def height_helper(self):
        left_h = self.left.height if self.left else -1
        right_h = self.right.height if self.right else -1
        if left_h == right_h == -1:
            return 0, 0
        else:
            return max(left_h, right_h) + 1, left_h - right_h

    @property
    def height(self):
        return self.height_helper[0]

    @property
    def balance(self):
        return self.height_helper[1]

    def rotate_left(self):
        temp = self.right
        self.right = temp.left
        temp.left = self
        return temp

    def rotate_right(self):
        temp = self.left
        self.left = temp.right
        temp.right = self
        return temp


```

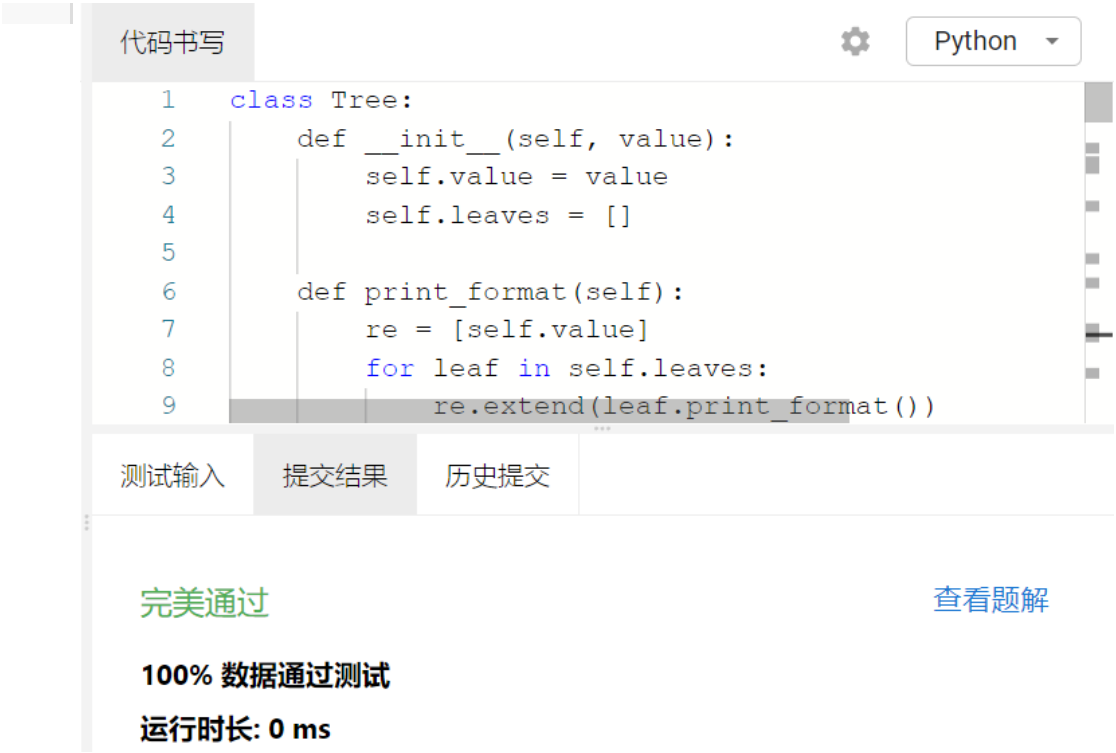
```python
def q5():
    def rotate(t):
        if t.balance == 2:
            if t.left.balance == 1:
                t = t.rotate_right()
            elif t.left.balance == -1:
                t.left = t.left.rotate_left()
                t = t.rotate_right()
        elif t.balance == -2:
            if t.right.balance == -1:
                t = t.rotate_left()
            elif t.right.balance == 1:
                t.right = t.right.rotate_right()
                t = t.rotate_left()
        return t

    def insert(t, i):
        if not t:
            return AVL(i)
        if i < t.value:
            t.left = insert(t.left, i)
        elif i > t.value:
            t.right = insert(t.right, i)
        if abs(t.balance) == 2:
            t = rotate(t)
        return t

    input()
    num_list = list(map(int, input().split()))
    t = None
    for num in num_list:
        t = insert(t, num)
    print(t)


q5()
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



02524: 宗教信仰

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

思路：about 40 mins.

代码

```
class DJSet:
    def __init__(self, n):
        self.parent = [i for i in range(n)]
        self.rank = [0] * n

    def find(self, x):
        if self.parent[x] != x:
            self.parent[x] = self.find(self.parent[x])
        return self.parent[x]

    def union(self, x, y):
        x_root = self.find(x)
        y_root = self.find(y)

        if x_root == y_root:
            return

        if self.rank[x_root] < self.rank[y_root]:
            self.parent[x_root] = y_root
        elif self.rank[x_root] > self.rank[y_root]:
            self.parent[y_root] = x_root
        else:
```

```
            self.parent[y_root] = x_root
            self.rank[x_root] += 1


def q6():
    case = 1
    while True:
        m, n = map(int, input().split())
        if m == 0 and n == 0:
            break

        ds = DJSet(m)
        for _ in range(n):
            i, j = map(int, input().split())
            ds.union(i - 1, j - 1)

        religions = set()
        for i in range(m):
            religions.add(ds.find(i))

        print("Case {}: {}".format(case, len(religions)))
        case += 1


q6()
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

状态：Accepted

源代码

```
class DJSet:
    def __init__(self, n):
        self.parent = [i for i in range(n)]
        self.rank = [0] * n

    def find(self, x):
```

基本信息

#: 44511595
题目: 02524
提交人: 2100017777
内存: 6556kB
时间: 1395ms
语言: Python3
提交时间: 2024-04-02 22:52:54

2. 学习总结和收获

如果作业题目简单，有否额外练习题目，比如：OJ“2024spring每日选做”、CF、LeetCode、洛谷等网站题目。

因为题目基本都是直接问数据结构的实现，抽象而直接，所以思路上存在不了太多改进空间，基本都是理解讲义后复现。但是细节上依然值得注意，例如空树的高度要设为-1而非0，这样才能成功递归。