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

Updated 2214 GMT+8 March 24, 2024

2024 spring, Complied by ==王一粟 经济学院==

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

==（请改为同学的操作系统、编程环境等）==

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：先根据二叉搜索树的定义解析二叉树，确认各节点之间的关联。然后后序输出即可。

耗时：20min

代码

```python
#2200015507 王一粟
class Node:
    def __init__(self,value):
        self.val = value
        self.left = None
        self.right = None
    def get_value(self):
        return self.val
def post_order(node):
    if node is None:
        return []
    result = []
    return post_order(node.left) + post_order(node.right) + [node.get_value()]
n = int(input())
wait_list = [int(i) for i in input().split()]
root = Node(wait_list[0])
for num in wait_list[1:]:
    current_node = root
    while True:
        if current_node.get_value() < num:
            if current_node.right:
                current_node = current_node.right
            else:
                current_node.right = Node(num)
                break
        else:
            if current_node.left:
                current_node = current_node.left
            else:
                current_node.left = Node(num)
                break
post_express = post_order(root)
print(" ".join(str(i) for i in post_express))
```

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

源代码

```python
#2200015507 王一粟
class Node:
    def __init__(self,value):
        self.val = value
        self.left = None
        self.right = None
    def get_value(self):
        return self.val
def post_order(node):
    if node is None:
        return []
    result = []
    return post_order(node.left) + post_order(node.right) + [node.get_va
n = int(input())
```

基本信息

#:	44347336
题目:	22275
提交人:	2200015507-王一粟
内存:	4100kB
时间:	29ms
语言:	Python3
提交时间:	2024-03-22 23:05:03

05455: 二叉搜索树的层次遍历

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

思路：和第一题思路相同，先根据二叉搜索树的定义进行解析，然后按照层次遍历的算法输出。

耗时：20min

代码

```
#2200015507 王一粟
class Node:
    def __init__(self):
        self.val = None
        self.left = None
        self.right = None
    def setvalue(self,value):
        self.val = value
    def getvalue(self):
        return self.val

def level_express(mynode):
    result_list = []
    stack = [mynode]
    while stack:
        current_node = stack.pop(0)
        result_list.append(current_node.getvalue())
        if current_node.left:
            stack.append(current_node.left)
        if current_node.right:
            stack.append(current_node.right)
    return result_list

origin_list = [int(i) for i in input().split()]
cnt = 0
for num in origin_list:
    operate_node = Node()
    operate_node.setvalue(num)
    if cnt == 0:
        cnt = 1
        root_node = operate_node
        continue
    current_node = root_node
    while True:
        if current_node.getvalue() == num:
            break
        elif current_node.getvalue() < num:
            parent_node = current_node
            current_node = current_node.right
            if current_node is None:
                parent_node.right = operate_node
                break
        else:
            parent_node = current_node
            current_node = current_node.left
            if current_node is None:
                parent_node.left = operate_node
                break
end_result = level_express(root_node)
```

```
print(" ".join(str(i) for i in end_result))
```

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

源代码

```
#2200015507 王一粟
class Node:
    def __init__(self):
        self.val = None
        self.left = None
        self.right = None
    def setvalue(self,value):
        self.val = value
    def getvalue(self):
        return self.val

def level_express(mynode):
    result_list = []
    stack = [mynode]
```

基本信息

#:	44195662
题目:	05455
提交人:	2200015507-王一粟
内存:	3684kB
时间:	23ms
语言:	Python3
提交时间:	2024-03-13 14:46:29

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：自己手搓二叉堆的时候确实出了一点问题：在删除最小元素的时候，如果堆里只剩一个元素时候的情况。讲义里的代码更简洁，且可以运行这种情况（先让第一个元素等于最后一个元素再pop）。从应用的角度，其实也应该将没有元素的情况拎出来讨论，此时不对heap操作（heapq包会报错）

耗时：25min

代码

```
#2200015507 王一粟
class heap:
    def __init__(self):
        self.heap_list = [0]
        self.current_size = 0
    def insert(self,k):
        self.heap_list.append(k)
        self.current_size += 1
        self.up(self.current_size)
    def up(self,i):
        while i//2>0:
            if self.heap_list[i] < self.heap_list[i//2]:
                self.heap_list[i],self.heap_list[i//2] = self.heap_list[i//2],self.heap_list[i
            else:
                break
            i = i//2
    def del_min(self):
```

```
        if self.current_size > 1:
            a = self.heap_list[1]
            self.heap_list[1] = self.heap_list.pop()
            self.current_size -= 1
            self.down(1)
            return a
        elif self.current_size == 1:
            a = self.heap_list[1]
            self.heap_list.pop()
            self.current_size = 0
            return a

    def down(self,i):
        while i*2 <= self.current_size:
            if i*2+1>self.current_size:
                if self.heap_list[i]>self.heap_list[i*2]:
                    self.heap_list[i],self.heap_list[i*2]=self.heap_list[i*2],self.heap_list[i
                break
            else:
                if self.heap_list[i*2]<self.heap_list[i*2+1]:
                    mc = i*2
                else:
                    mc = i*2+1
                if self.heap_list[mc]<self.heap_list[i]:
                    self.heap_list[mc],self.heap_list[i] = self.heap_list[i],self.heap_list[mc
                    i = mc
                else:
                    break
n = int(input())
mylist = heap()
for _ in range(n):
    s = input()
    if s == "2":
        print(mylist.del_min())
    else:
        type_num,add_num = [int(i) for i in s.split()]
        mylist.insert(add_num)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
#2200015507 王一粟
class heap:
    def __init__(self):
        self.heap_list = [0]
        self.current_size = 0
    def insert(self,k):
        self.heap_list.append(k)
        self.current_size += 1
        self.up(self.current_size)
    def up(self,i):
        while i//2>0:
            if self.heap_list[i] < self.heap_list[i//2]:
                self.heap_list[i],self.heap_list[i//2] = self.heap_list
            else:
                break
            i = i//2
```

基本信息

#:	44394129
题目:	04078
提交人:	2200015507-王一粟
内存:	4144kB
时间:	606ms
语言:	Python3
提交时间:	2024-03-25 11:38:24

22161: 哈夫曼编码树

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

思路：几个比较关键的地方：1.比较大小通过修改__lt__函数完成，比较方式为先比较value再比较char；对于非叶节点，char在节点中构建方式可以通过比较两个子结点的char谁更小这一方式完成。2.encode的部分通过在函数中进行递归调用的方式解决。3.decode部分可以在每次循环后判断是否为叶结点，确定是否输出char

耗时：35min

代码

```
#2200015507 王一粟
import heapq
class Node:
    def __init__(self,val):
        self.value = val
        self.char = None
        self.left = None
        self.right = None
    def __lt__(self,other):
        if self.value == other.value:
            return ord(self.char) < ord(self.char)
        return self.value < other.value
    def get_value(self):
        return self.value
    def get_char(self):
        return self.char

def decode(ini_root,wait_string):
    now_node = ini_root
    result = ""
    for char in wait_string:
        if char == "1":
            now_node = now_node.right
```

```
            else:
                now_node = now_node.left
            if now_node.left is None:
                result += now_node.get_char()
                now_node = ini_root
        return result
    def encode(ini_root):
        codes = {}
        def parse(node,code):
            if node.left is None:
                codes[node.get_char()] = code
            else:
                parse(node.left,code+"0")
                parse(node.right,code+"1")
        parse(ini_root,"")
        return codes
    n = int(input())
    mylist = []
    for _ in range(n):
        word,freq = input().split()
        freq = int(freq)
        current_node = Node(freq)
        current_node.char = word
        mylist.append(current_node)
    heapq.heapify(mylist)
    for i in range(len(mylist)-1):
        small = heapq.heappop(mylist)
        big = heapq.heappop(mylist)
        add_node = Node(small.get_value()+big.get_value())
        if ord(small.get_char()) < ord(big.get_char()):
            add_node.char = small.get_char()
        else:
            add_node.char = big.get_char()
        add_node.left = small
        add_node.right = big
        heapq.heappush(mylist,add_node)
    root = mylist[0]
    code_dict = encode(root)
    while True:
        try:
            s = input()
            if s[0] == "0" or s[0] == "1":
                print(decode(root,s))
            else:
                ex = ""
                for element in s:
                    ex += code_dict[element]
                print(ex)
        except:
            break
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#44400788提交状态 查看 提交 统计 提问

状态: Accepted

基本信息

#: 44400788
题目: 22161
提交人: 2200015507-王一粟
内存: 3720kB
时间: 24ms
语言: Python3
提交时间: 2024-03-25 21:12:29

源代码

```
#2200015507 王一粟
import heapq
class Node:
    def __init__(self,val):
        self.value = val
        self.char = None
        self.left = None
        self.right = None
    def __lt__(self,other):
        if self.value == other.value:
            return ord(self.char) < ord(self.char)
        return self.value < other.value
    def get_value(self):
        return self.value
    def get_char(self):
        return self.char

def decode(ini_root,wait_string):
```

晴问9.5: 平衡二叉树的建立

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

思路：同讲义思路。这里列举几点编程时遇到的问题：第一次左旋和右旋函数写反了，还是要注意树型和旋转方向是相反的结果；第二个是类的写法，要注意封装在类中的函数，在书写时如果想调用类中的函数，应该加self.

耗时：35min

代码

```
#2200015507 王一粟
class Node:
    def __init__(self,val):
        self.value = val
        self.left = None
        self.right = None
        self.height = 1
class AVL:
    def __init__(self):
        self.root = None
    def insert(self,value):
        if self.root is None:
            self.root = Node(value)
        else:
            self.root =self._insert(self.root,value)
    def _get_height(self,node):
        if node is None:
            return 0
        return 1+max(self._get_height(node.left),self._get_height(node.right))

    def _get_balance(self,node):
        return self._get_height(node.left)-self._get_height(node.right)
    def _rotate_right(self,node):
```

```
        origin_root = node
        new_root = origin_root.left
        new_root_origin_right = new_root.right
        new_root.right = origin_root
        origin_root.left = new_root_origin_right
        new_root.height = 1+max(self._get_height(new_root.right),self._get_height(new_root.left)
        origin_root.height = 1+max(self._get_height(origin_root.left),self._get_height(origin_ro
        return new_root
    def _rotate_left(self,node):
        origin_root = node
        new_root = origin_root.right
        new_root_origin_left = new_root.left
        new_root.left = origin_root
        origin_root.right = new_root_origin_left
        new_root.height = 1+max(self._get_height(new_root.right),self._get_height(new_root.left)
        origin_root.height = 1 + max(self._get_height(origin_root.left), self._get_height(origin
        return new_root
    def _insert(self,node,value):
        if node is None:
            return Node(value)
        if node.value > value:
            node.left = self._insert(node.left,value)
        else:
            node.right = self._insert(node.right,value)
        node.height = self._get_height(node)
        balance = self._get_balance(node)
        if balance > 1: #左树不平衡
            if value<node.left.value: #LL,右旋
                return self._rotate_right(node)
            else: #LR,左旋+右旋
                node.left = self._rotate_left(node.left)
                return self._rotate_right(node)
        if balance < -1: #右树不平衡
            if value>node.right.value: #RR, 左旋
                return self._rotate_left(node)
            else: #RL, 右旋+左旋
                node.right = self._rotate_right(node.right)
                return self._rotate_left(node)
        return node
    def preorder(self):
        return self._preorder(self.root)
    def _preorder(self,node):
        if node is None:
            return []
        return [node.value] + self._preorder(node.left)+self._preorder(node.right)
n = int(input())
mylist = [int(i) for i in input().split()]
avl = AVL()
for element in mylist:
    avl.insert(element)
result = avl.preorder()
print(" ".join(str(i) for i in result))
```

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：采用并查集的方法，用size做了简化。果然数算还是得学经典算法做题，自己硬搞太吃瘪了

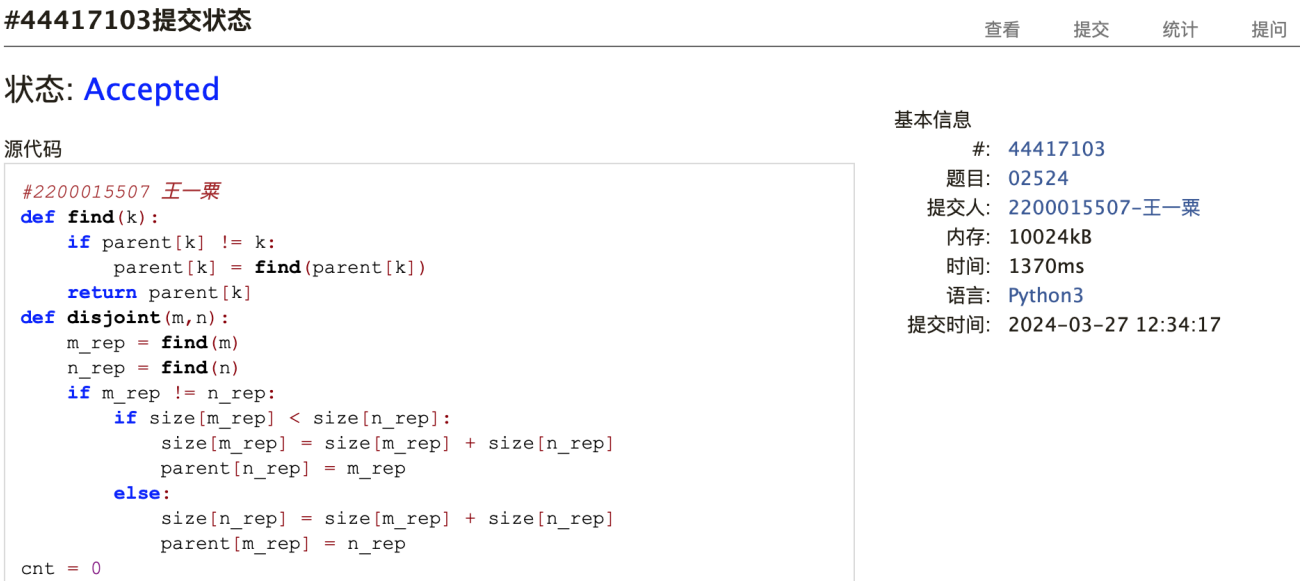
耗时：周一学并查集前，自己编程3h(1WA+3RE+1TLE)，学了并查集后15min AC

代码

```
#2200015507 王一粟
def find(k):
    if parent[k] != k:
        parent[k] = find(parent[k])
    return parent[k]
def disjoint(m,n):
    m_rep = find(m)
    n_rep = find(n)
    if m_rep != n_rep:
        if size[m_rep] < size[n_rep]:
            size[m_rep] = size[m_rep] + size[n_rep]
            parent[n_rep] = m_rep
        else:
            size[n_rep] = size[m_rep] + size[n_rep]
            parent[m_rep] = n_rep
cnt = 0
while True:
    n,m = [int(i) for i in input().split()]
    if n == 0 and m == 0:
        break
```

```
    cnt += 1
    parent = [i for i in range(n+1)]
    size = [1 for i in range(n+1)]
    for _ in range(m):
        a,b = [int(i) for i in input().split()]
        disjoint(a,b)
    result = len([i for i in range(1,n+1) if parent[i]==i])
    print(f"Case {cnt}: {result}")
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



#44417103提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
#2200015507 王一粟
def find(k):
    if parent[k] != k:
        parent[k] = find(parent[k])
    return parent[k]
def disjoint(m,n):
    m_rep = find(m)
    n_rep = find(n)
    if m_rep != n_rep:
        if size[m_rep] < size[n_rep]:
            size[m_rep] = size[m_rep] + size[n_rep]
            parent[n_rep] = m_rep
        else:
            size[n_rep] = size[m_rep] + size[n_rep]
            parent[m_rep] = n_rep
cnt = 0
```

基本信息

#: 44417103
题目: 02524
提交人: 2200015507-王一粟
内存: 10024kB
时间: 1370ms
语言: Python3
提交时间: 2024-03-27 12:34:17

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。==

这周同样跟紧了每日选做的进度，到目前为止树这块学完了老师附录以前的所有内容；

感觉数算这块还是要背一些套路的hh，很多问题其实就是一种固定的算法模式；

计概C选手补课进度：本周学完了老师的递归讲义，感觉内容不多，难度不大，不过感觉好多递归里的题目本质都是dfs诶；

这周+下周打算有空补一下dp，量多的话不确定能否一周补完