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

Updated 2214 GMT+8 March 24, 2024

2024 spring, Complied by 数学科学学院 王镜廷 2300010724

**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：Windows11 专业版

Python编程环境：VSCode 1.86.2, with extension Python and python version 3.12.2

# 1. 题目

## 22275: 二叉搜索树的遍历

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

用时：约15分钟

思路：注意到由二叉搜索树的性质，直接对值进行排序即可得到中序遍历，从而问题即化归为先前所做的由前序遍历和中序遍历建树。

代码

```
class node :
    def __init__ (self, left, right, val) :
        self.left = left
        self.right = right
        self.val = val

def getIndex(c, s) :
    for i in range(len(s)) :
        if c == s[i] :
            return i
    return -1

def ToPostString(s) :
    s1 = ""
    if s.left != None :
        s1 = ToPostString(s.left) + " "
    s2 = ""
    if s.right != None :
        s2 = ToPostString(s.right) + " "
    return s1 + s2 + str(s.val)

def BuildTree_pre_in(sp, si) :
    if sp == [] :
        return None
    l = getIndex(sp[0], si)
#    print(sp, si, l, sep = " ")
    return node(BuildTree_pre_in(sp[1 : l + 1], si[0 : l]), BuildTree_pre_in(sp[l + 1 : len(sp)

n = int(input())
s1 = list(map(int, input().split()))
s2 = sorted(s1)
print(ToPostString(BuildTree_pre_in(s1, s2)))
```

代码运行截图 （至少包含有"Accepted"）

#44340333提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class node :
    def __init__ (self, left, right, val) :
        self.left = left
        self.right = right
        self.val = val

def getIndex(c, s) :
    for i in range(len(s)) :
        if c == s[i] :
            return i
    return -1

def ToPostString(s) :
    s1 = ""
    if s.left != None :
        s1 = ToPostString(s.left) + " "
    s2 = ""
    if s.right != None :
        s2 = ToPostString(s.right) + " "
    return s1 + s2 + str(s.val)
```

基本信息

#: 44340333
题目: 22275
提交人: 23n2300010724
内存: 4080kB
时间: 28ms
语言: Python3
提交时间: 2024-03-22 16:59:47

# 05455: 二叉搜索树的层次遍历

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

用时：约20分钟

思路：依题意模拟，依次插入建树后输出即可

代码

```
class node :
    def __init__(self, left, right, val) :
        self.left = left
        self.right = right
        self.val = val

def insert(x, s) :
    if s == None :
        return node(None, None, x)
    if x > s.val :
        return node(s.left, insert(x, s.right), s.val)
    if x < s.val :
        return node(insert(x, s.left), s.right, s.val)
    return s

def ToQString(s) :
    q = [s]
    res = []
    while q != [] :
        u = q[0]
        del(q[0])
        res.append(u.val)
        if u.left != None :
            q.append(u.left)
        if u.right != None :
            q.append(u.right)
    return " ".join(str(item) for item in res)

s = None
s1 = list(map(int, input().split()))
for i in s1 :
    s = insert(i, s)
print(ToQString(s))
```

代码运行截图 （至少包含有"Accepted"）

#44198794提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class node :
    def __init__(self, left, right, val) :
        self.left = left
        self.right = right
        self.val = val

def insert(x, s) :
    if s == None :
        return node(None, None, x)
    if x > s.val :
        return node(s.left, insert(x, s.right), s.val)
    if x < s.val :
        return node(insert(x, s.left), s.right, s.val)
    return s

def ToQString(s) :
    q = [s]
    res = []
    while q != [] :
        u = q[0]
        del(q[0])
```

基本信息

#: 44198794
题目: 05455
提交人: 23n2300010724
内存: 3676kB
时间: 23ms
语言: Python3
提交时间: 2024-03-13 16:48:18

# 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。

用时：约30分钟

思路：按照课上讲的方法实现即可

代码

```
class BinHeap : # the top of the heap is the smallest
    def __init__(self) :
        self.item = [0]
        self.size = 0
    def percUp(self, i) :
        if i == 1 :
            return
        if self.item[i] < self.item[i // 2] :
            self.item[i] ^= self.item[i // 2]
            self.item[i // 2] ^= self.item[i]
            self.item[i] ^= self.item[i // 2]
            self.percUp(i // 2)
    def insert(self, x) :
        self.item.append(x)
        self.size += 1
        self.percUp(self.size)
    def percDown(self, i) :
        if i * 2 > self.size :
            return
        u = i * 2
        if 2 * i + 1 <= self.size :
            if self.item[2 * i + 1] < self.item[2 * i] :
                u = 2 * i + 1
        if self.item[u] < self.item[i] :
            self.item[i] ^= self.item[u]
            self.item[u] ^= self.item[i]
            self.item[i] ^= self.item[u]
            self.percDown(u)
    def delTop(self) :
        if self.size == 0 :
            return None
        res = self.item[1]
        self.item[1] = self.item[self.size]
        self.item.pop()
        self.size -= 1
        self.percDown(1)
        return res
    def heapify(self, items) :
        self.item.extend(items)
        self.size = len(self.item) - 1
        i = self.size // 2
        while i >= 1 :
            self.percDown(i)
            i -= 1

q = BinHeap()
```

```
n = int(input())
for i in range(n) :
    s = list(map(int, input().split()))
    if s[0] == 1 :
        q.insert(s[1])
    else :
        print(q.delTop())
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44408730提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class BinHeap : # the top of the heap is the smallest
    def __init__(self) :
        self.item = [0]
        self.size = 0
    def percUp(self, i) :
        if i == 1 :
            return
        if self.item[i] < self.item[i // 2] :
            self.item[i] ^= self.item[i // 2]
            self.item[i // 2] ^= self.item[i]
            self.item[i] ^= self.item[i // 2]
            self.percUp(i // 2)
    def insert(self, x) :
        self.item.append(x)
        self.size += 1
        self.percUp(self.size)
    def percDown(self, i) :
        if i * 2 > self.size :
            return
        u = i * 2
        if 2 * i + 1 <= self.size :
```

基本信息

#: 44408730
题目: 04078
提交人: 23n2300010724
内存: 4132kB
时间: 882ms
语言: Python3
提交时间: 2024-03-26 16:44:37

# 22161: 哈夫曼编码树

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

用时：约30分钟

思路：实现Huffman编码的过程中记录树上每个节点对应的字符集

代码

```
import heapq

class node :
    def __init__(self, chars, val, left = None, right = None, isleaf = True, char = None) :
        self.chars = chars
        self.val = val
        self.left = left
        self.right = right
        self.isleaf = isleaf
        self.char = char
    def __lt__(self, other) :
        return self.val < other.val or (self.val == other.val and min(self.chars) < min(other.c
    def merge(self, other) :
        return node(self.chars.union(other.chars), self.val + other.val, self, other, isleaf =
    def getcode(self, c) :
        if self.isleaf == True and c == self.char:
            return ""
        if c in self.left.chars :
            return "0" + self.left.getcode(c)
        if c in self.right.chars :
            return "1" + self.right.getcode(c)
    def decode(self, root, u) :
        if self.isleaf :
            return self.char + root.decode(root, u)
        if u == "" :
            return ""
        if u[0] == "0" :
            return self.left.decode(root, u[1 : len(u)])
        else :
            return self.right.decode(root, u[1 : len(u)])
    def encode(self, s) :
        res = ""
        for i in s :
            res = res + self.getcode(i)
        return res

def Huffman(s1) :
    heapq.heapify(s1)
    while len(s1) >= 2 :
        x = heapq.heappop(s1)
        y = heapq.heappop(s1)
        heapq.heappush(s1, x.merge(y))
    return heapq.heappop(s1)


n = int(input())
```

```
s1 = []
for i in range(n) :
    u = input().split()
    s1.append(node(set([u[0]]), float(u[1]), char = u[0]))
H = Huffman(s1)
while True :
    try :
        s = input()
        if ord(s[0]) == ord('0') or ord(s[0]) == ord('1') :
            print(H.decode(H, s))
        else :
            print(H.encode(s))
    except :
        break
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44446850提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import heapq

class node :
    def __init__(self, chars, val, left = None, right = None, isleaf = 
        self.chars = chars
        self.val = val
        self.left = left
        self.right = right
        self.isleaf = isleaf
        self.char = char
    def __lt__(self, other) :
        return self.val < other.val or (self.val == other.val and min(se
    def merge(self, other) :
        return node(self.chars.union(other.chars), self.val + other.val,
    def getcode(self, c) :
        if self.isleaf == True and c == self.char:
            return ""
        if c in self.left.chars :
            return "0" + self.left.getcode(c)
        if c in self.right.chars :
            return "1" + self.right.getcode(c)
    def decode(self, root, u) :
        if self.isleaf :
            return self.char + root.decode(root, u)
        if u == "" :
            return ""
```

基本信息

#: 44446850
题目: 22161
提交人: 23n2300010724
内存: 3680kB
时间: 25ms
语言: Python3
提交时间: 2024-03-29 17:16:06

# 晴问9.5: 平衡二叉树的建立

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

用时：约90分钟

思路：按照AVL的算法实现即可

代码

```
class Node :
    def __init__(self, val) : # making a new leaf node with value = val
        self.val = val
        self.left = None
        self.right = None
        self.height = 1
    def upd_height(self) :
        if not self.left and not self.right :
            self.height = 1
            return
        if not self.left :
            self.height = self.right.height + 1
            return
        if not self.right :
            self.height = self.left.height + 1
            return
        self.height = max(self.left.height, self.right.height) + 1
    def balanceness(self) :
        u = 0
        v = 0
        if self.left != None :
            u = self.left.height
        if self.right != None :
            v = self.right.height
    #    print(u, v)
        return u - v
class AVL :
    def __init__(self) :
        self.root = None
    def _rotate_right(self, node) :
    #    print("rotate right")
        T1 = node.left.right
        tmp = node.left
        tmp.right = node
        tmp.right.left = T1
        tmp.right.upd_height()
        tmp.upd_height()
        return tmp
    def _rotate_left(self, node) :
    #    print("rotate left")
        T1 = node.right.left
        tmp = node.right
        tmp.left = node
        tmp.left.right = T1
        tmp.left.upd_height()
        tmp.upd_height()
```

```
    return tmp
def _rebalance(self, node) :
#    print("rebalance")
    if node.balanceness() >= 2 :
        if node.left.balanceness() == 1 :
#            print("LL")
            node = self._rotate_right(node)
            return node
        else :
#            print("LR")
            node.left = self._rotate_left(node.left)
            node = self._rotate_right(node)
            return node
    if node.balanceness() <= -2 :
        if node.right.balanceness() == -1 :
#            print("RR")
            node = self._rotate_left(node)
            return node
        else :
#            print("RL")
            node.right = self._rotate_right(node.right)
            node = self._rotate_left(node)
            return node
#    print("nothing")
    return node
def _insert(self, value, node) :
    if not node :
#        print("added")
        return Node(value)
    if node.val < value :
#        print("going right")
        node.right = self._insert(value, node.right)
        node.upd_height()
        node = self._rebalance(node)
        return node
    if node.val > value :
#        print("going left")
        node.left = self._insert(value, node.left)
        node.upd_height()
        node = self._rebalance(node)
        return node
def insert(self, value) :
    if not self.root :
        self.root = Node(value)
    else :
        self.root = self._insert(value, self.root)
```

```
    def delete(self, value) : # nothing will be done if value doesn't exist
        pass
    def _delete(self, value, node) :
        pass
    def traversal(self, mode, node) :
        if not node :
            return []
        if mode == "pre" :
            return [node.val] + self.traversal("pre", node.left) + self.traversal("pre", node.r
        if mode == "mid" :
            return self.traversal("mid", node.left) + [node.val] + self.traversal("mid", node.r
        if mode == "post" :
            return self.traversal("post", node.left) + self.traversal("post", node.right) + [no
    def __str__(self) : # preorder traversal as default
        if not self.root :
            return ""
        return " ".join(str(item) for item in self.traversal("pre", self.root))

n = int(input())
T = AVL()
u = list(map(int, input().split()))
for i in u :
    T.insert(i)
    #print(T)
print(T)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

完美通过

**100% 数据通过测试**

**运行时长: 0 ms**

---

**语言: Python**

```python
class Node :
    def __init__(self, val) : # making a new leaf no
        self.val = val
        self.left = None
        self.right = None
        self.height = 1
    def upd_height(self) :
        if not self.left and not self.right :
            self.height = 1
            return
        if not self.left :
            self.height = self.right.height + 1
            return
        if not self.right :
            self.height = self.left.height + 1
            return
        self.height = max(self.left.height, self.rig
    def balanceness(self) :
        u = 0
```

## 02524: 宗教信仰

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

用时：约30分钟

思路：实现并查集即可

代码

```python
class DisjointSet :
    def __init__(self, items) :
        self.sizes = dict(zip(items, [1] * len(items)))
        self.rep = dict(zip(items, items))
        self.length = len(items)
    def getrep(self, i) :
        if self.rep[i] == i :
            return i
        res = self.getrep(self.rep[i])
        self.rep[i] = res
        return self.rep[i]
    def check_if_same(self, i, j) :
        return (self.getrep(i) == self.getrep(j))
    def merge(self, i, j) :
        x = self.getrep(i)
        y = self.getrep(j)
        if x == y :
            return
        u = self.sizes[x]
        v = self.sizes[y]
        if u < v :
            self.rep[x] = y
            self.sizes[y] = u + v
        else :
            self.rep[y] = x
            self.sizes[x] = u + v
    def count(self) :
        res = 0
        for item in self.rep :
            if self.getrep(item) == item :
                res += 1
        return res

Case = 0
while True :
    Case += 1
    n, m = map(int, input().split())
    if n == 0 and m == 0 :
        break
    s = DisjointSet([i for i in range(1, n + 1)])
    for i in range(m) :
        u, v = map(int, input().split())
        s.merge(u, v)
    print(f"Case {Case}: {s.count()}")
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44445700提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class DisjointSet :
    def __init__(self, items) :
        self.sizes = dict(zip(items, [1] * len(items)))
        self.rep = dict(zip(items, items))
        self.length = len(items)
    def getrep(self, i) :
        if self.rep[i] == i :
            return i
        res = self.getrep(self.rep[i])
        self.rep[i] = res
        return self.rep[i]
    def check_if_same(self, i, j) :
        return (self.getrep(i) == self.getrep(j))
    def merge(self, i, j) :
        x = self.getrep(i)
        y = self.getrep(j)
        if x == y :
            return
        u = self.sizes[x]
        v = self.sizes[y]
        if u < v :
```

基本信息

#: 44445700
题目: 02524
提交人: 23n2300010724
内存: 16448kB
时间: 1593ms
语言: Python3
提交时间: 2024-03-29 16:09:28

# 2. 学习总结和收获

这周学到的AVL算法是以前学的时候没太搞懂的部分，实现之后感觉还是理解的更好了。同时，这周在代码里面开始学着使用heapq, set, dict等python自带的实现，确实很好用（而且找不到函数的时候可以问ChatGPT）。而且现在调代码的时候也感觉更熟练了一些。