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

Updated 2214 GMT+8 March 24, 2024

2024 spring, Complied by 王业成 生命科学学院

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：Windows 11 家庭中文版 22631.3296

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：建树模拟

代码

```
# class Treenode:
    def __init__(self,value):
        self.value=value
        self.left=None
        self.right=None
def buildtree(lst):
    if len(lst)==0:
```

```
        return None
    root_value=lst[0]
    root=Treenode(root_value)
    index_=len(lst)
    for i in range(1,len(lst)):
        if lst[i]>lst[0]:
            index_=i
            break
    root.left=buildtree(lst[1:index_])
    root.right=buildtree(lst[index_:])
    return root
def parsetree(root):
    if root is None:
        return []
    else:
        output=[]
        output.extend(parsetree(root.left))
        output.extend(parsetree(root.right))
        output.append(str(root.value))
        return output
n=int(input())
lst=list(map(int,input().split()))
root=buildtree(lst)
output=parsetree(root)
print(" ".join(output))
```

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

基本信息

#:	44496912
题目:	22275
提交人:	wangyecheng
内存:	4152kB
时间:	27ms
语言:	Python3
提交时间:	2024-04-01 16:10:18

源代码

```
class Treenode:
    def __init__(self,value):
        self.value=value
        self.left=None
        self.right=None
def buildtree(lst):
    if len(lst)==0:
        return None
    root_value=lst[0]
    root=Treenode(root_value)
    index_=len(lst)
    for i in range(1,len(lst)):
        if lst[i]>lst[0]:
            index_=i
            break
    root.left=buildtree(lst[1:index_])
    root.right=buildtree(lst[index_:])
    return root
def parsetree(root):
    if root is None:
        return []
    else:
        output=[]
        output.extend(parsetree(root.left))
        output.extend(parsetree(root.right))
        output.append(str(root.value))
        return output
n=int(input())
lst=list(map(int,input().split()))
root=buildtree(lst)
output=parsetree(root)
print(" ".join(output))
```

05455: 二叉搜索树的层次遍历

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

思路:

代码

```
# from collections import deque
class Treenode:
    def __init__(self,value):
        self.value=value
        self.left=None
        self.right=None
def insert(root,value):
    if root is None:
        return Treenode(value)
    else:
        if value<root.value:
            root.left=insert(root.left,value)
        else:
            root.right=insert(root.right,value)
    return root
def buildtree(lst):
    root=None
    for i in lst:
        root=insert(root,i)
    return root
def parsetree(root):
    if root is None:
        return []
    else:
        output=[]
        a=deque()
        a.append(root)
        while a:
            i=a.popleft()
            output.append(i.value)
            if i.left!=None:
                a.append(i.left)
            if i.right!=None:
                a.append((i.right))
        return output
numbers = list(map(int, input().strip().split()))
numbers = list(dict.fromkeys(numbers))
root=buildtree(numbers)
output=parsetree(root)
print(" ".join(map(str,output)))
```

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

基本信息

#: 44497235
题目: 05455
提交人: wangyecheng
内存: 3688kB
时间: 26ms
语言: Python3
提交时间: 2024-04-01 16:46:16

源代码

```
from collections import deque
class Treenode:
    def __init__(self,value):
        self.value=value
        self.left=None
        self.right=None
def insert(root,value):
    if root is None:
        return Treenode(value)
    else:
        if value<root.value:
            root.left=insert(root.left,value)
        else:
            root.right=insert(root.right,value)
    return root
def buildtree(lst):
    root=None
    for i in lst:
        root=insert(root,i)
    return root
def parsetree(root):
    if root is None:
        return []
    else:
        output=[]
        a=deque()
        a.append(root)
        while a:
            i=a.popleft()
            output.append(i.value)
            if i.left!=None:
                a.append(i.left)
            if i.right!=None:
                a.append((i.right))
        return output
numbers = list(map(int, input().strip().split()))
numbers = list(dict.fromkeys(numbers))
root=buildtree(numbers)
output=parsetree(root)
print(" ".join(map(str,output)))
```

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:

代码

```
# class heap:
    def __init__(self):
        self.headlist=[0]
        self.size=0
    def insert(self,i):
        self.headlist.append(i)
        self.size+=1
        self.up(self.size)
    def up(self,i):
        while i//2!=0:
```

```
            if self.headlist[i]<self.headlist[i//2]:

 self.headlist[i],self.headlist[i//2]=self.headlist[i//2],self.headlist[i]
            i=i//2
    def delmin(self):
        a=self.headlist[1]
        self.headlist[1]=self.headlist[self.size]
        self.size-=1
        self.headlist.pop()
        self.down(1)
        return a
    def down(self,i):
        while i*2<=self.size:
            a=self.minchild(i)
            if self.headlist[i]>self.headlist[a]:

 self.headlist[i],self.headlist[a]=self.headlist[a],self.headlist[i]
            i=a
    def minchild(self,i):
        if i*2+1>self.size:
            return i*2
        else:
            if self.headlist[i*2]<self.headlist[i*2+1]:
                return i*2
            else:
                return i*2+1
n = int(input().strip())
bh = heap()
for _ in range(n):
    a = input().strip()
    if a[0] == '1':
        bh.insert(int(a.split()[1]))
    else:
        print(bh.delmin())
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

基本信息

#: 44497624
题目: 04078
提交人: wangyecheng
内存: 4672kB
时间: 628ms
语言: Python3
提交时间: 2024-04-01 17:21:20

源代码

```
class heap:
    def __init__(self):
        self.headlist=[0]
        self.size=0
    def insert(self,i):
        self.headlist.append(i)
        self.size+=1
        self.up(self.size)
    def up(self,i):
        while i//2!=0:
            if self.headlist[i]<self.headlist[i//2]:
                self.headlist[i],self.headlist[i//2]=self.headlist[i//2]
            i=i//2
    def delmin(self):
        a=self.headlist[1]
        self.headlist[1]=self.headlist[self.size]
        self.size-=1
        self.headlist.pop()
        self.down(1)
        return a
    def down(self,i):
        while i*2<=self.size:
            a=self.minchild(i)
            if self.headlist[i]>self.headlist[a]:
                self.headlist[i],self.headlist[a]=self.headlist[a],self
            i=a
    def minchild(self,i):
        if i*2+1>self.size:
            return i*2
        else:
            if self.headlist[i*2]<self.headlist[i*2+1]:
                return i*2
            else:
                return i*2+1
n = int(input().strip())
bh = heap()
for _ in range(n):
    a = input().strip()
    if a[0] == '1':
        bh.insert(int(a.split()[1]))
    else:
        print(bh.delmin())
```

22161: 哈夫曼编码树

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

思路:

代码

```
# class Treenode():
    def __init__(self,weight,char):
        self.weight=weight
        self.char=char
        self.left=None
        self.right=None
def buildtree(dict):
    lst=[]
    for char, weight in dict.items():
        lst.append(Treenode(weight,char))
    lst.sort(key=lambda x: [x.weight, x.char])
    while len(lst)>1:
        a=lst.pop(0)
```

```
        b=lst.pop(0)
        node=Treenode(a.weight+b.weight,a.char+b.char)
        node.left=a
        node.right=b
        lst.append(node)
        lst.sort(key=lambda x:[x.weight,x.char])
    root=lst[0]
    return root
def encode_huffman_tree(root):
    codes = {}
    def traverse(node, code):
        if node.left is None and node.right is None:
            codes[node.char] = code
        else:
            traverse(node.left, code + '0')
            traverse(node.right, code + '1')
    traverse(root, '')
    return codes
def huffman_encoding(codes, string):
    encoded = ''
    for char in string:
        encoded += codes[char]
    return encoded
def huffman_decoding(root, encoded_string):
    decoded = ''
    node = root
    for bit in encoded_string:
        if bit == '0':
            node = node.left
        else:
            node = node.right
        if node.left is None and node.right is None:
            decoded += node.char
            node = root
    return decoded
n = int(input())
characters = {}
for _ in range(n):
    char, weight = input().split()
    characters[char] = int(weight)
huffman_tree = buildtree(characters)
codes = encode_huffman_tree(huffman_tree)
strings = []
while True:
    try:
        line = input()
        strings.append(line)
    except EOFError:
        break
results = []
for string in strings:
    if string[0] in ('0','1'):
        results.append(huffman_decoding(huffman_tree, string))
    else:
        results.append(huffman_encoding(codes, string))
for result in results:
```

```
    print(result)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态：Accepted

源代码

```
class Treenode():
    def __init__(self,weight,char):
        self.weight=weight
        self.char=char
        self.left=None
        self.right=None
def buildtree(dict):
    lst=[]
    for char, weight in dict.items():
        lst.append(Treenode(weight,char))
    lst.sort(key=lambda x: [x.weight, x.char])
    while len(lst)>1:
        a=lst.pop(0)
        b=lst.pop(0)
        node=Treenode(a.weight+b.weight,a.char+b.char)
        node.left=a
        node.right=b
        lst.append(node)
        lst.sort(key=lambda x:[x.weight,x.char])
    root=lst[0]
    return root
def encode_huffman_tree(root):
    codes = {}
    def traverse(node, code):
        if node.left is None and node.right is None:
            codes[node.char] = code
        else:
            traverse(node.left, code + '0')
            traverse(node.right, code + '1')
    traverse(root, '')
    return codes
def huffman_encoding(codes, string):
    encoded = ''
    for char in string:
        encoded += codes[char]
    return encoded
def huffman_decoding(root, encoded_string):
    decoded = ''
    node = root
    for bit in encoded_string:
        if bit == '0':
            node = node.left
        else:
```

基本信息

#:	44500795
题目:	22161
提交人:	wangyecheng
内存:	3788kB
时间:	24ms
语言:	Python3
提交时间:	2024-04-01 21:48:21

晴问9.5: 平衡二叉树的建立

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

思路：

代码

```
# class Node:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None
        self.height = 1
```

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

    def insert(self, value):
        if not self.root:
            self.root = Node(value)
        else:
            self.root = self._insert(value, self.root)

    def _insert(self, value, node):
        if not node:
            return Node(value)
        elif value < node.value:
            node.left = self._insert(value, node.left)
        else:
            node.right = self._insert(value, node.right)

        node.height = 1 + max(self._get_height(node.left),
self._get_height(node.right))

        balance = self._get_balance(node)

        if balance > 1:
            if value < node.left.value: # 树形是 LL
                return self._rotate_right(node)
            else:   # 树形是 LR
                node.left = self._rotate_left(node.left)
                return self._rotate_right(node)

        if balance < -1:
            if value > node.right.value:    # 树形是 RR
                return self._rotate_left(node)
            else:   # 树形是 RL
                node.right = self._rotate_right(node.right)
                return self._rotate_left(node)

        return node

    def _get_height(self, node):
        if not node:
            return 0
        return node.height

    def _get_balance(self, node):
        if not node:
            return 0
        return self._get_height(node.left) - self._get_height(node.right)

    def _rotate_left(self, z):
        y = z.right
        T2 = y.left
        y.left = z
        z.right = T2
        z.height = 1 + max(self._get_height(z.left), self._get_height(z.right))
        y.height = 1 + max(self._get_height(y.left), self._get_height(y.right))
```

```
        return y

    def _rotate_right(self, y):
        x = y.left
        T2 = x.right
        x.right = y
        y.left = T2
        y.height = 1 + max(self._get_height(y.left), self._get_height(y.right))
        x.height = 1 + max(self._get_height(x.left), self._get_height(x.right))
        return x

    def preorder(self):
        return self._preorder(self.root)

    def _preorder(self, node):
        if not node:
            return []
        return [node.value] + self._preorder(node.left) +
self._preorder(node.right)

n = int(input().strip())
sequence = list(map(int, input().strip().split()))

avl = AVL()
for value in sequence:
    avl.insert(value)

print(' '.join(map(str, avl.preorder())))
```

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：

代码

```
# def get_father(x,lst):
    if x!=lst[x]:
        lst[x]=get_father(lst[x],lst)
    return lst[x]
def union(x,y,lst):
    a=get_father(x,lst)
    b=get_father(y,lst)
    if a==b:
        return
    lst[a]=b
num=0
while True:
    n,m=map(int,input().split())
    if n==m==0:
        break
    else:
        lst=list(range(n))
        count=0
        for i in range(m):
            a,b=map(int,input().split())
            union(a-1,b-1,lst)
        for i in range(n):
            if lst[i]==i:
                count+=1
        num+=1
        print(f"Case {num}: {count}")
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
def get_father(x,lst):
    if x!=lst[x]:
        lst[x]=get_father(lst[x],lst)
    return lst[x]
def union(x,y,lst):
    a=get_father(x,lst)
    b=get_father(y,lst)
    if a==b:
        return
    lst[a]=b
num=0
while True:
    n,m=map(int,input().split())
    if n==m==0:
        break
    else:
        lst=list(range(n))
        count=0
        for i in range(m):
            a,b=map(int,input().split())
            union(a-1,b-1,lst)
        for i in range(n):
            if lst[i]==i:
                count+=1
        num+=1
        print(f"Case {num}: {count}")
```

基本信息

#:	44501351
题目:	02524
提交人:	wangyecheng
内存:	10572kB
时间:	1255ms
语言:	Python3
提交时间:	2024-04-01 22:38:11

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。==

感觉这周作业难度好大，基本都是看着上课的代码自己边理解边摸索出来的，不过也学到了许多东西，收获满满，后续还得再花时间好好理解消化