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

Updated 2214 GMT+8 March 24, 2024

2024 spring, Complied by 赵策 数学科学学院

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

操作系统：Windows 11

Python编程环境：Visual Studio Code 1.86.2

1. 题目

22275: 二叉搜索树的遍历

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

思路:

代码

```
#
def preorder_to_postorder(preorder):
    if not preorder:
        return []

    root=preorder[0]
    left_subtree=[x for x in preorder if x<root]
    right_subtree=[x for x in preorder if x> root]

    left_postorder=preorder_to_postorder(left_subtree)
    right_postorder=preorder_to_postorder(right_subtree)
```

```
    return left_postorder+right_postorder+[root]


n=int(input())
preorder=list(map(int,input().split()))
postorder=preorder_to_postorder(preorder)
print(' '.join(map(str,postorder)))
```

代码运行截图 （至少包含有"Accepted"）

05455: 二叉搜索树的层次遍历

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

思路:

代码

```
#
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)
    if value<root.value:
        root.left=insert(root.left,value)
    elif value>root.value:
        root.right=insert(root.right,value)
    return root

def level_order_traversal(root):
    if root is None:
        return []
    result=[]
    queue=[root]
    while queue:
        node=queue.pop(0)
        result.append(node.value)
        if node.left:
            queue.append(node.left)
        if node.right:
            queue.append(node.right)
```

```
    return result

numbers=list(map(int,input().split()))
root=None
for num in numbers:
    root=insert(root,num)
result=level_order_traversal(root)
print(' '.join(map(str,result)))
```

代码运行截图 （至少包含有"Accepted"）

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 Minheap:
    def __init__(self):
        self.heap=[]

    def parent(self,i):
        return (i-1)//2

    def left_child(self,i):
        return 2*i+1

    def right_child(self,i):
        return 2*i+2

    def swap(self,i,j):
        self.heap[i],self.heap[j]=self.heap[j],self.heap[i]

    def insert(self,value):
        self.heap.append(value)
        self.heapify_up(len(self.heap)-1)

    def heapify_up(self,i):
        while i>0 and self.heap[i]<self.heap[self.parent(i)]:
            self.swap(i,self.parent(i))
            i=self.parent(i)
```

```
    def heapify_down(self,i):
        min_index=i
        left=self.left_child(i)
        right=self.right_child(i)
        if left<len(self.heap) and self.heap[left]<self.heap[min_index]:
            min_index=left
        if right<len(self.heap) and self.heap[right]<self.heap[min_index]:
            min_index=right
        if i!=min_index:
            self.swap(i,min_index)
            self.heapify_down(min_index)

    def extract_min(self):
        if len(self.heap)==0:
            return None
        min_val=self.heap[0]
        self.heap[0]=self.heap[-1]
        del self.heap[-1]
        self.heapify_down(0)
        return min_val

    def peek_min(self):
        if len(self.heap)==0:
            return None
        return self.heap[0]

heap=Minheap()
for _ in range(int(input())):
    operation=input().split()
    if operation[0]=='1':
        num=int(operation[1])
        Minheap.insert(heap, num)
    elif operation[0]=='2':
        min_num=Minheap.extract_min(heap)
        print(min_num)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

22161: 哈夫曼编码树

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

思路：

代码

```
#
class Node:
    def __init__(self,char=None,freq=None):
        self.char=char
        self.freq=freq
        self.left=None
        self.right=None

def build_huffman_tree(characters):
    nodes=[Node(char,freq) for char,freq in characters]
    while len(nodes)>1:
        nodes.sort(key=lambda x:(x.freq,x.char))
        left=nodes.pop(0)
        right=nodes.pop(0)
        parent=Node(freq=left.freq+right.freq)
        parent.left=left
        parent.right=right
        nodes.append(parent)
    return nodes[0]

def encode(root,mapping,prefix=''):
    if root is None:
        return
    if root.char is not None:
        mapping[root.char]=prefix
    encode(root.left,mapping,prefix+'0')
    encode(root.right,mapping,prefix+'1')

def huffman_encoding(characters,input_string):
    huffman_tree=build_huffman_tree(characters)
    mapping={}
    encode(huffman_tree,mapping)
    encoded_string=''.join(mapping[char] for char in input_string)
    return mapping,encoded_string

def huffman_decoding(root,encoded_string):
    decoded_string= ''
    current_node=root
    for bit in encoded_string:
        if bit=='0':
            current_node=current_node.left
        else:
            current_node=current_node.right
        if current_node.char is not None:
            decoded_string+=current_node.char
            current_node=root
    return decoded_string

n=int(input())
characters=[]
for _ in range(n):
    char,freq=input().split()
    characters.append((char,int(freq)))

huffman_tree=build_huffman_tree(characters)
```

```
while True:
    try:
        string=input()
        if string.isalpha():
            mapping,encoded_string=huffman_encoding(characters,string)
            print(encoded_string)
        else:
            decoded_string=huffman_decoding(huffman_tree,string)
            print(decoded_string)
    except EOFError:
        break
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

晴问9.5: 平衡二叉树的建立

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

思路:

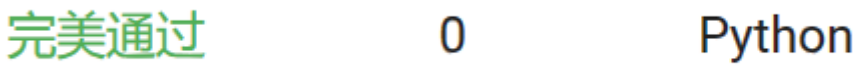
代码

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

class AVLTree:
    def getHeight(self,node):
        if not node:
            return 0
        return node.height

    def getBalance(self,node):
        if not node:
            return 0
        return self.getHeight(node.left)-self.getHeight(node.right)

    def rotateRight(self,y):
        x=y.left
        T2=x.right

        x.right=y
```

```
        y.left=T2

        y.height=max(self.getHeight(y.left),self.getHeight(y.right))+1
        x.height=max(self.getHeight(x.left),self.getHeight(x.right))+1
        return x

    def rotateLeft(self,x):
        y=x.right
        T2=y.left

        y.left=x
        x.right=T2

        x.height=max(self.getHeight(x.left),self.getHeight(x.right))+1
        y.height=max(self.getHeight(y.left),self.getHeight(y.right))+1
        return y

    def insert(self,root,val):
        if not root:
            return TreeNode(val)
        elif val<root.val:
            root.left=self.insert(root.left,val)
        else:
            root.right=self.insert(root.right,val)

        root.height=1+max(self.getHeight(root.left),self.getHeight(root.right))

        balance=self.getBalance(root)

        if balance>1:
            if val<root.left.val:
                return self.rotateRight(root)
            else:
                root.left=self.rotateLeft(root.left)
                return self.rotateRight(root)

        if balance<-1:
            if val>root.right.val:
                return self.rotateLeft(root)
            else:
                root.right=self.rotateRight(root.right)
                return self.rotateLeft(root)
        return root

    def preOrderTraversal(self,root):
        result=[]
        if root:
            result.append(root.val)
            result+=self.preOrderTraversal(root.left)
            result+=self.preOrderTraversal(root.right)
        return result

n=int(input())
nums=list(map(int,input().split()))

avl_tree=AVLTree()
```

```
root=None
for num in nums:
    root=avl_tree.insert(root,num)

prefix=avl_tree.preOrderTraversal(root)
print(' '.join(map(str,prefix)))
```

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 find_religion_count(n,m,connections):
    parent=list(range(n+1))
    def find(x):
        if parent[x]==x:
            return x
        parent[x]=find(parent[x])
        return parent[x]

    def union(x,y):
        parent[find(x)]=find(y)

    for i,j in connections:
        union(i,j)
    religions=set()
    for i in range(1,n+1):
        religions.add(find(i))
    return len(religions)

cnt=0
while True:
    cnt+=1
    n,m=map(int,input().split())
    if [n,m]==[0,0]:
        break
    connections=[]
    for _ in range(m):
        i,j=map(int,input().split())
```

```
            connections.append((i,j))
        max_religion_count=find_religion_count(n,m,connections)
        print(f'Case {cnt}: {max_religion_count}')
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

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。

挺难的，相当耗费时间，需要进一步消化。其中的Huffman编码树、AVL树、并查集相关代码都可以保存反复学习。