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

2024 spring, Complied by 胡豪俊 工学院

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：Windows11

Python编程环境：Visual Studio Code

1. 题目

22275: 二叉搜索树的遍历

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

代码

```
class Node:
    def __init__(self,val):
        self.val=val
        self.left=None
        self.right=None

def build(preorder,inorder):
    if not preorder or not inorder:
        return None
    root_val=preorder[0]
    root=Node(root_val)
    root_index=inorder.index(root_val)
    root.left=build(preorder[1:root_index+1],inorder[:root_index])
    root.right=build(preorder[root_index+1:], inorder[root_index+1:])
    return root


def postorder(root):
```

```
    if not root:
        return []
    if root.left is None and root.right is None:
        return [root.val]
    result=[]
    result+=postorder(root.left)
    result+=postorder(root.right)
    result+=[root.val]
    return result


n=input()
preorder=list(map(int,input().split()))
inorder=sorted(preorder)
root=build(preorder,inorder)
result=postorder(root)
print(' '.join(map(str,result)))
##写完这个对树更有感觉了，除了这个代码还参考了题解里很短的那个代码，感觉被碾压了
```

#44497163提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class Node:
    def __init__(self,val):
        self.val=val
        self.left=None
        self.right=None

def build(preorder,inorder):
    if not preorder or not inorder:
        return None
    root_val=preorder[0]
    root=Node(root_val)
    root_index=inorder.index(root_val)
    root.left=build(preorder[1:root_index+1],inorder[:root_index])
    root.right=build(preorder[root_index+1:], inorder[root_index+1:])
    return root


def postorder(root):
    if not root:
        return []
    if root.left is None and root.right is None:
        return [root.val]
    result=[]
    result+=postorder(root.left)
    result+=postorder(root.right)
    result+=[root.val]
    return result


input()
preorder=list(map(int,input().split()))
inorder=sorted(preorder)
root=build(preorder,inorder)
result=postorder(root)
print(' '.join(map(str,result)))
```

基本信息

#: 44497163
题目: 22275
提交人: 22n2200011800
内存: 4092kB
时间: 28ms
语言: Python3
提交时间: 2024-04-01 16:37:58

English 帮助 关于

05455: 二叉搜索树的层次遍历

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

代码

```
class TreeNode:
    def __init__(self,value):
        self.value=value
        self.left=None
```

```
        self.right=None

def insert(node,value):
    if node is None:
        return TreeNode(value)
    if value<node.value:
        node.left=insert(node.left,value)
    elif value>node.value:
        node.right=insert(node.right,value)
    return node

def level_order_traversal(root):
    queue=[root]
    traversal=[]
    while queue:
        node=queue.pop(0)
        traversal.append(node.value)
        if node.left:
            queue.append(node.left)
        if node.right:
            queue.append(node.right)
    return traversal

numbers=list(map(int,input().strip().split()))
numbers=list(dict.fromkeys(numbers))
root=None
for number in numbers:
    root=insert(root,number)
traversal=level_order_traversal(root)
print(' '.join(map(str,traversal)))
##和上一题有点区别，不过总体的思路经过思考后还是能理顺的
```

#44497241提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class TreeNode:
    def __init__(self,value):
        self.value=value
        self.left=None
        self.right=None

def insert(node,value):
    if node is None:
        return TreeNode(value)
    if value<node.value:
        node.left=insert(node.left,value)
    elif value>node.value:
        node.right=insert(node.right,value)
    return node

def level_order_traversal(root):
    queue=[root]
    traversal=[]
    while queue:
        node=queue.pop(0)
        traversal.append(node.value)
        if node.left:
            queue.append(node.left)
        if node.right:
            queue.append(node.right)
    return traversal

numbers=list(map(int,input().strip().split()))
numbers=list(dict.fromkeys(numbers))
root=None
for number in numbers:
    root=insert(root,number)
traversal=level_order_traversal(root)
print(' '.join(map(str,traversal)))
```

基本信息

#: 44497241
题目: 05455
提交人: 22n2200011800
内存: 3656kB
时间: 26ms
语言: Python3
提交时间: 2024-04-01 16:46:43

English 帮助 关于

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。

代码

```
class BinHeap:
    def __init__(self):
        self.heapList=[0]
        self.currentSize=0

    def percUp(self,i):
        while i//2>0:
            if self.heapList[i]<self.heapList[i//2]:
                tmp=self.heapList[i//2]
                self.heapList[i//2]=self.heapList[i]
                self.heapList[i]=tmp
            i=i//2

    def insert(self,k):
        self.heapList.append(k)
        self.currentSize=self.currentSize+1
        self.percUp(self.currentSize)

    def percDown(self,i):
```

```
        while (i*2)<=self.currentSize:
            mc=self.minChild(i)
            if self.heapList[i]>self.heapList[mc]:
                tmp=self.heapList[i]
                self.heapList[i]=self.heapList[mc]
                self.heapList[mc]=tmp
            i=mc

    def minChild(self,i):
        if i*2+1>self.currentSize:
            return i*2
        else:
            if self.heapList[i*2]<self.heapList[i*2+1]:
                return i*2
            else:
                return i*2+1

    def delMin(self):
        retval=self.heapList[1]
        self.heapList[1]=self.heapList[self.currentSize]
        self.currentSize=self.currentSize-1
        self.heapList.pop()
        self.percDown(1)
        return retval

    def buildHeap(self,alist):
        i=len(alist)//2
        self.currentSize=len(alist)
        self.heapList=[0]+alist[:]
        while (i > 0):
            self.percDown(i)
            i=i-1


n=int(input().strip())
bh=BinHeap()
for _ in range(n):
    inp=input().strip()
    if inp[0]=='1':
        bh.insert(int(inp.split()[1]))
    else:
        print(bh.delMin())
```

##Binheap手搓实在是错了很多次，后面看了题解才发现在minChild那一步返回的数值有一个忘记加一了，细节还是要注意。当然前面的堆的实现的代码还是可以记一记的。

#44497054提交状态

查看 提交 统计 提问

状态: Accepted

基本信息

#: 44497054
题目: 04078
提交人: 22n2200011800
内存: 4700kB
时间: 609ms
语言: Python3
提交时间: 2024-04-01 16:29:35

源代码

```
class BinHeap:
    def __init__(self):
        self.heapList=[0]
        self.currentSize=0

    def percUp(self,i):
        while i//2>0:
            if self.heapList[i]<self.heapList[i//2]:
                tmp=self.heapList[i//2]
                self.heapList[i//2]=self.heapList[i]
                self.heapList[i]=tmp
            i=i//2

    def insert(self,k):
        self.heapList.append(k)
        self.currentSize=self.currentSize+1
        self.percUp(self.currentSize)

    def percDown(self,i):
        while (i*2)<=self.currentSize:
            mc=self.minChild(i)
            if self.heapList[i]>self.heapList[mc]:
                tmp=self.heapList[i]
                self.heapList[i]=self.heapList[mc]
                self.heapList[mc]=tmp
            i=mc

    def minChild(self,i):
        if i*2+1>self.currentSize:
            return i*2
        else:
            if self.heapList[i*2]<self.heapList[i*2+1]:
                return i*2
            else:
                return i*2+1

    def delMin(self):
        retval=self.heapList[1]
```

22161: 哈夫曼编码树

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

代码

```
import heapq

class Node:
    def __init__(self,weight,char=None):
        self.weight=weight
        self.char=char
        self.left=None
        self.right=None

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

def build_huffman_tree(characters):
    heap=[]
    for char,weight in characters.items():
        heapq.heappush(heap,Node(weight,char))

    while len(heap)>1:
        left=heapq.heappop(heap)
```

```
        right=heapq.heappop(heap)
        merged=Node(left.weight+right.weight,min(left.char,right.char))
        merged.left=left
        merged.right=right
        heapq.heappush(heap,merged)

    return heap[0]

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=build_huffman_tree(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)
##理解题目想表达的东西理解了很久，最后在题解的帮助下做出来了，这题对树的应用已经出神入化了
```

#44496965提交状态

查看 提交 统计 提问

状态: Accepted

基本信息

#: 44496965
题目: 22161
提交人: 22n2200011800
内存: 3688kB
时间: 23ms
语言: Python3
提交时间: 2024-04-01 16:20:00

源代码

```
import heapq

class Node:
    def __init__(self,weight,char=None):
        self.weight=weight
        self.char=char
        self.left=None
        self.right=None

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

def build_huffman_tree(characters):
    heap=[]
    for char,weight in characters.items():
        heapq.heappush(heap,Node(weight,char))

    while len(heap)>1:
        left=heapq.heappop(heap)
        right=heapq.heappop(heap)
        merged=Node(left.weight+right.weight,min(left.char,right.char))
        merged.left=left
        merged.right=right
        heapq.heappush(heap,merged)

    return heap[0]

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')
```

晴问9.5: 平衡二叉树的建立

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

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:
                return self._rotate_right(node)
            else:
                node.left=self._rotate_left(node.left)
                return self._rotate_right(node)

        if balance<-1:
            if value>node.right.value:
                return self._rotate_left(node)
            else:
                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())))
##第一次在这个新网站上写题目，一开始还提交错了好几次没注意语言，总的来说这种要自己从头手写的都比较麻烦，代码会相对来说长一点
```

02524: 宗教信仰

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

代码

```
def init_set(n):
    return list(range(n))

def get_father(x,father):
    if father[x]!=x:
        father[x]=get_father(father[x],father)
    return father[x]

def join(x,y,father):
    fx=get_father(x,father)
    fy=get_father(y,father)
    if fx==fy:
        return
    father[fx]=fy

def is_same(x,y,father):
```

```
    return get_father(x,father)==get_father(y,father)

def main():
    case_num=0
    while True:
        n,m=map(int,input().split())
        if n==0 and m==0:
            break
        count=0
        father=init_set(n)
        for _ in range(m):
            s1,s2=map(int,input().split())
            join(s1-1,s2-1,father)
        for i in range(n):
            if father[i]==i:
                count+=1
        case_num+=1
        print(f"Case {case_num}: {count}")

if __name__=="__main__":
    main()
```

##树的思想的应用，这题用了题解里的主函数的写法（虽然和直接写感觉这里差别也不大），前面实现起来其实比较自然。

#44497378提交状态

查看 提交 统计 提问

状态: Accepted

基本信息

#: 44497378
题目: 02524
提交人: 22n2200011800
内存: 10524kB
时间: 1170ms
语言: Python3
提交时间: 2024-04-01 17:02:32

源代码

```
def init_set(n):
    return list(range(n))

def get_father(x,father):
    if father[x]!=x:
        father[x]=get_father(father[x],father)
    return father[x]

def join(x,y,father):
    fx=get_father(x,father)
    fy=get_father(y,father)
    if fx==fy:
        return
    father[fx]=fy

def is_same(x,y,father):
    return get_father(x,father)==get_father(y,father)

def main():
    case_num=0
    while True:
        n,m=map(int,input().split())
        if n==0 and m==0:
            break
        count=0
        father=init_set(n)
        for _ in range(m):
            s1,s2=map(int,input().split())
            join(s1-1,s2-1,father)
        for i in range(n):
            if father[i]==i:
                count+=1
        case_num+=1
        print(f"Case {case_num}: {count}")

if __name__=="__main__":
    main()
```

2. 学习总结和收获

太难了，几乎每道题我都被卡了挺久，最终要不对着题解看自己哪里没想到要不像实现堆结构和平衡二叉树建立这种比较标准的模板一样的东西就直接照着题解学了。虽然我自己没能力从无到有想出来，但是还是通过这些资料学到了很多东西。马上要期中考试了时间很紧张，需要更加规划好时间对数算的难点进行学习。