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

Updated 2000 GMT+8 March 25, 2024

2024 spring, Complied by 钟明衡 物理学院

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_NT x64 10.0.19045

Python编程环境：Visual Studio Code 1.76.1

C/C++编程环境：Visual Studio Code 1.76.1

1. 题目

22275: 二叉搜索树的遍历

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

思路:

二叉搜索树中的每一个节点，左边的都比它小，右边的都比它大，因此可以通过大小关系来确定是左子树还是右子树，从而递归地建树

代码

```
from bisect import bisect


def suf(l):
    if len(l) <= 1:
```

```
        return l
    n = bisect(l[1:], l[0])+1
    return suf(l[1:n])+suf(l[n:])+[l[0]]


n = int(input())
l = list(map(int, input().split()))
print(' '.join(map(str, suf(l))))

```

代码运行截图

#44335806提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from bisect import bisect


def suf(l):
    if len(l) <= 1:
        return l
    n = bisect(l[1:], l[0])+1
    return suf(l[1:n])+suf(l[n:])+[l[0]]


n = int(input())
l = list(map(int, input().split()))
print(' '.join(map(str, suf(l))))
```

基本信息

#: 44335806
题目: 22275
提交人: 23n2300011505(12号娱乐选手)
内存: 3864kB
时间: 23ms
语言: Python3
提交时间: 2024-03-22 12:45:29

05455: 二叉搜索树的层次遍历

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

思路:

构建一棵二叉搜索树，小的往左大的往右，最后按层次遍历输出即可

代码

```
from collections import defaultdict as D
left, right = {}, {}
ans = D(lambda: [])


def build(n, node):
    if n < node:
        try:
            build(n, left[node])
        except KeyError:
            left[node] = n
```

```
    elif n > node:
        try:
            build(n, right[node])
        except KeyError:
            right[node] = n


def dfs(node, h):
    global ans
    ans[h].append(node)
    try:
        dfs(left[node], h+1)
    except KeyError:
        pass
    try:
        dfs(right[node], h+1)
    except KeyError:
        pass


l = list(map(int, input().split()))
for i in range(1, len(l)):
    build(l[i], l[0])
dfs(l[0], 0)
ans[0] = [l[0]]
a = []
for i in range(max(ans.keys())+1):
    a += ans[i]
print(' '.join(map(str, a)), end='')

```

代码运行截图

#44317258提交状态 查看 提交 统计 提问

状态: Accepted

基本信息

#: 44317258
题目: 05455
提交人: 23n2300011505(12号娱乐选手)
内存: 3672kB
时间: 24ms
语言: Python3
提交时间: 2024-03-20 20:15:22

源代码

```
from collections import defaultdict as D
left, right = {}, {}
ans = D(lambda: [])


def build(n, node):
    if n < node:
        try:
            build(n, left[node])
        except KeyError:
            left[node] = n
    elif n > node:
        try:
            build(n, right[node])
        except KeyError:
            right[node] = n


def dfs(node, h):
    global ans
    ans[h].append(node)
    try:
        dfs(left[node], h+1)
    except KeyError:
        pass
    try:
        dfs(right[node], h+1)
    except KeyError:
        pass


l = list(map(int, input().split()))
for i in range(1, len(l)):
    build(l[i], l[0])
dfs(l[0], 0)
ans[0] = [l[0]]
a = []
for i in range(max(ans.keys())+1):
    a += ans[i]
print(' '.join(map(str, a)), end='')
```

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:

用二叉搜索树来手搓一个堆，注意有可能把根节点pop，要特殊处理

代码

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

```
6
7
8  class Heap:
9      def __init__(self):
10         self.root = None
11
12     def add(self, val):
13         if not self.root:
14             self.root = Tree(val)
15         else:
16             self._add(self.root, val)
17
18     def _add(self, node, val):
19         if not node:
20             return Tree(val)
21         if val < node.val:
22             node.left = self._add(node.left, val)
23         else:
24             node.right = self._add(node.right, val)
25         return node
26
27     def pop(self):
28         if not self.root:
29             return None
30         node = self.root
31         while node.left:
32             node = node.left
33         self.root = self._pop(self.root)
34         return node.val
35
36     def _pop(self, node):
37         if not node.left:
38             return node.right
39         node.left = self._pop(node.left)
40         return node
41
42
43 heap = Heap()
44 for _ in range(int(input())):
45     s = input()
46     if len(s)-1:
47         heap.add(int(s.split()[1]))
48     else:
49         print(heap.pop())
50
```

代码运行截图

#44352577提交状态

查看 提交 统计 提问

状态: Accepted

基本信息

#: 44352577
题目: 04078
提交人: 23n2300011505(12号娱乐选手)
内存: 4296kB
时间: 3243ms
语言: Python3
提交时间: 2024-03-23 10:48:57

源代码

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


class Heap:
    def __init__(self):
        self.root = None

    def add(self, val):
        if not self.root:
            self.root = Tree(val)
        else:
            self._add(self.root, val)

    def _add(self, node, val):
        if not node:
            return Tree(val)
        if val < node.val:
            node.left = self._add(node.left, val)
        else:
            node.right = self._add(node.right, val)
        return node

    def pop(self):
        if not self.root:
            return None
        node = self.root
        while node.left:
            node = node.left
        self.root = self._pop(self.root)
        return node.val

    def _pop(self, node):
        if not node.left:
            return node.right
        node.left = self._pop(node.left)
        return node


heap = Heap()
for _ in range(int(input())):
    s = input()
    if len(s)-1:
        heap.add(int(s.split()[1]))
    else:
        print(heap.pop())
```

22161: 哈夫曼编码树

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

思路:

懒得写树了，把编码和权重存到堆里，每次取最小的两个，将其中较小的那个包含的每个字符的编码最前面加上'0'，较大的那个包含的每个字符的编码最前面加上'1'，然后把权重相加，字符串排序，放回堆中，如此循环直到只剩下一个元，此时编码就完成了

后面解码或者编码，就用之前得到的编码字典去操作就好

代码

```
from collections import defaultdict as D
import heapq as H
l = []
d = D(str)
n = int(input())
for _ in range(n):
    s = input().split()
    l.append((int(s[1]), list(s[0])))
H.heapify(l)
for _ in range(n-1):
    a = H.heappop(l)
    b = H.heappop(l)
    a, b = max(a, b), min(a, b)
    for c in a[1]:
        d[c] = '1'+d[c]
    for c in b[1]:
        d[c] = '0'+d[c]
    H.heappush(l, (a[0]+b[0], sorted(a[1]+b[1])))
e = {a[1]: a[0] for a in d.items()}
while True:
    try:
        s = input()
    except EOFError:
        break
    if s[0] in '01':
        A = B = ''
        for a in s:
            A += a
            try:
                B += e[A]
                A = ''
            except KeyError:
                continue
        print(B)
    else:
        print(''.join(d[a] for a in s))

```

代码运行截图

#44394334提交状态 查看 提交 统计 提问

状态: Accepted

基本信息
#: 44394334
题目: 22161
提交人: 23n2300011505(12号娱乐选手)
内存: 3656kB
时间: 25ms
语言: Python3
提交时间: 2024-03-25 12:41:44

源代码

```
from collections import defaultdict as D
import heapq as H
l = []
d = D(str)
n = int(input())
for _ in range(n):
    s = input().split()
    l.append((int(s[1]), list(s[0])))
H.heapify(l)
for _ in range(n-1):
    a = H.heappop(l)
    b = H.heappop(l)
    a, b = max(a, b), min(a, b)
    for c in a[1]:
        d[c] = '1'+d[c]
    for c in b[1]:
        d[c] = '0'+d[c]
    H.heappush(l, (a[0]+b[0], sorted(a[1]+b[1])))
e = {a[1]: a[0] for a in d.items()}
while True:
    try:
        s = input()
    except EOFError:
        break
    if s[0] in '01':
        A = B = ''
        for a in s:
            A += a
            try:
                B += e[A]
                A = ''
            except KeyError:
                continue
        print(B)
    else:
        print(''.join(d[a] for a in s))
```

晴问9.5: 平衡二叉树的建立

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

思路:

写了一个AVL树，思路就是在二叉搜索树的基础上判断平衡，平衡失调就把根节点和高的那边的一个子节点对调位置

代码

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


class Tree:
```

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

    def height(self, node):
        if node is None:
            return 0
        return node.height

    def balance_factor(self, node):
        if node is None:
            return 0
        return self.height(node.left) - self.height(node.right)

    def rotate_right(self, y):
        x = y.left
        T = x.right
        x.right = y
        y.left = T
        y.height = 1 + max(self.height(y.left), self.height(y.right))
        x.height = 1 + max(self.height(x.left), self.height(x.right))
        return x

    def rotate_left(self, x):
        y = x.right
        T = y.left
        y.left = x
        x.right = T
        x.height = 1 + max(self.height(x.left), self.height(x.right))
        y.height = 1 + max(self.height(y.left), self.height(y.right))
        return y

    def insert(self, node, val):
        if node is None:
            return Node(val)
        if val < node.val:
            node.left = self.insert(node.left, val)
        else:
            node.right = self.insert(node.right, val)
        node.height = 1 + max(self.height(node.left), self.height(node.right))
        balance = self.balance_factor(node)
        if balance > 1 and val < node.left.val:
            return self.rotate_right(node)
        if balance < -1 and val > node.right.val:
            return self.rotate_left(node)
        if balance > 1 and val > node.left.val:
            node.left = self.rotate_left(node.left)
            return self.rotate_right(node)
        if balance < -1 and val < node.right.val:
            node.right = self.rotate_right(node.right)
            return self.rotate_left(node)
        return node

```

```
    def pre(self, node, result):
        if node:
            result.append(node.val)
            self.pre(node.left, result)
            self.pre(node.right, result)


n = int(input())
tree = Tree()
for a in list(map(int, input().split())):
    tree.root = tree.insert(tree.root, a)
result = []
tree.pre(tree.root, result)
print(' '.join(map(str, result)))

```

代码运行截图

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):
        self.val = val
        self.left = None
        self.right = None
        self.height = 1


class Tree:
    def __init__(self):
        self.root = None

    def height(self, node):
        if node is None:
            return 0
        return node.height

    def balance_factor(self, node):
        if node is None:
            return 0
        return self.height(node.left) - self.height(node.righ

    def rotate_right(self, y):
        x = y.left
        T = x.right
        x.right = y
```

02524: 宗教信仰

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

思路:

这个问题可以转化为一个孤立区域数量问题，每组相同信仰都是建立一对连接，最后用bfs来计算孤立的区域数量即可

代码

```python
k = 0
while True:
    k += 1
    n, m = map(int, input().split())
    if not m:
        break
    l = [[] for _ in range(n+1)]
    for _ in range(m):
        a, b = map(int, input().split())
        l[a].append(b)
        l[b].append(a)
    f, ans = [False]*(n+1), 0
    for i in range(1, n+1):
        if not f[i]:
            ans += 1
            L, s, e = [i], 0, 1
            while e-s:
                for j in range(s, e):
                    for a in l[L[j]]:
                        if not f[a]:
                            f[a] = True
                            L.append(a)
                s, e = e, len(L)
    print('Case %d: %d' % (k, ans))

```

代码运行截图

#44398707提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
k = 0
while True:
    k += 1
    n, m = map(int, input().split())
    if not m:
        break
    l = [[] for _ in range(n+1)]
    for _ in range(m):
        a, b = map(int, input().split())
        l[a].append(b)
        l[b].append(a)
    f, ans = [False]*(n+1), 0
    for i in range(1, n+1):
        if not f[i]:
            ans += 1
            L, s, e = [i], 0, 1
            while e-s:
                for j in range(s, e):
                    for a in l[L[j]]:
                        if not f[a]:
                            f[a] = True
                            L.append(a)
                s, e = e, len(L)
    print('Case %d: %d' % (k, ans))
```

基本信息

#: 44398707
题目: 02524
提交人: 23n2300011505(12号娱乐选手)
内存: 19308kB
时间: 1176ms
语言: Python3
提交时间: 2024-03-25 18:56:27

2. 学习总结和收获

每天都在更进每日选做

第一次手搓堆和AVL树，感觉想法很易懂且巧妙，而且这类数据结构的代码模板性都很强