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

Updated 2214 GMT+8 March 24, 2024

2024 spring, Complied by ==祁轩宇、经济学院==

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, version 23H2

Python编程环境：VSCode 1.87.1

C/C++编程环境：

1. 题目

22275: 二叉搜索树的遍历

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

思路：

代码

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

# 数组第一个元素是根节点，紧跟着是小于根节点值的节点，在根节点左侧，直至遇到大于根节点值的节点，后续节点都在根节点右侧，按照这个思路递归即可
def buildTree(preorder):
    if len(preorder) == 0:
```

```
        return None

    node = Node(preorder[0])

    idx = len(preorder)
    for i in range(1, len(preorder)):
        if preorder[i] > preorder[0]:
            idx = i
            break
    node.left = buildTree(preorder[1:idx])
    node.right = buildTree(preorder[idx:])

    return node


def postorder(node):
    if node is None:
        return []
    output = []
    output.extend(postorder(node.left))
    output.extend(postorder(node.right))
    output.append(str(node.val))

    return output


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

代码运行截图 ==（至少包含有"Accepted"）==

#44507904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

# 数组第一个元素是根节点，紧跟着是小于根节点值的节点，在根节点左侧，直至遇到大于根节点
def buildTree(preorder):
    if len(preorder) == 0:
        return None

    node = Node(preorder[0])

    idx = len(preorder)
    for i in range(1, len(preorder)):
        if preorder[i] > preorder[0]:
            idx = i
            break
    node.left = buildTree(preorder[1:idx])
    node.right = buildTree(preorder[idx:])

    return node


def postorder(node):
    if node is None:
        return []
    output = []
    output.extend(postorder(node.left))
    output.extend(postorder(node.right))
    output.append(str(node.val))

    return output


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

基本信息

#: 44507904
题目: 22275
提交人: 2100015440
内存: 4048kB
时间: 26ms
语言: Python3
提交时间: 2024-04-02 18:01:39

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
# 二叉搜索树中添加节点
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
# 层次遍历
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))  # remove duplicates
root = None
for number in numbers:
    root = insert(root, number)
traversal = level_order_traversal(root)
print(' '.join(map(str, traversal)))
```

代码运行截图 ==（至少包含有"Accepted"）==

#44508143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
# 二叉搜索树中添加节点
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
# 层次遍历
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))  # remove duplicates
root = None
for number in numbers:
    root = insert(root, number)
traversal = level_order_traversal(root)
print(' '.join(map(str, traversal)))
```

基本信息

#: 44508143
题目: 05455
提交人: 2100015440
内存: 3668kB
时间: 24ms
语言: Python3
提交时间: 2024-04-02 18:31:01

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 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):
            #print(f'i = {i}, {self.heapList}')
            self.percDown(i)
            i = i - 1
        #print(f'i = {i}, {self.heapList}')


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())
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#44508304提交状态

查看 提交 统计 提问

状态: Accepted

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]
        self.heapList[1] = self.heapList[self.currentSize]
        self.currentSize = self.currentSize - 1
        self.heapList.pop()
        self.percDown(1)
        return retval
```

基本信息

#: 44508304
题目: 04078
提交人: 2100015440
内存: 4664kB
时间: 617ms
语言: Python3
提交时间: 2024-04-02 18:48:31

22161: 哈夫曼编码树

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

思路：

代码

```

```

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

```
```

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(x):
    if parent[x] != x: # 如果不是根结点，继续循环
        parent[x] = find(parent[x])
    return parent[x]

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

Case=0
while True:
    Case+=1
    n, m = map(int, input().split())
    if (n,m)==(0,0):
        break
    parent = list(range(n + 1)) # parent[i] == i，则说明元素i是该集合的根结点

    for _ in range(m):
        a, b = map(int, input().split())
        union(a, b)

    classes = set(find(x) for x in range(1, n + 1))
    print(f'Case {Case}: {len(classes)}')
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#44508794提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def find(x):
    if parent[x] != x: # 如果不是根结点，继续循环
        parent[x] = find(parent[x])
    return parent[x]

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

Case=0
while True:
    Case+=1
    n, m = map(int, input().split())
    if (n,m)==(0,0):
        break
    parent = list(range(n + 1)) # parent[i] == i, 则说明元素i是该集合的根结点

    for _ in range(m):
        a, b = map(int, input().split())
        union(a, b)

    classes = set(find(x) for x in range(1, n + 1))
    print(f'Case {Case}: {len(classes)}')
```

基本信息

#: 44508794
题目: 02524
提交人: 2100015440
内存: 11040kB
时间: 1247ms
语言: Python3
提交时间: 2024-04-02 19:32:07

©2002-2022 POJ 京ICP备20010980号-1

English 帮助 关于

2. 学习总结和收获

个人感觉作业的题目应该不算难，主要是编程熟练度不够，看懂算法只是第一步，还需要通过练习才能熟练复现。1、2、6题难度适中，3、4、5题还有些生疏，结合题解和讲义之后完成了作业。宗教信仰这道题让我再一次感受到计概思维和数算思维的不同，虽然计概思路也可以完成，但通过合适的数据结构可以很方便地完成解题。