

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

Updated 2214 GMT+8 March 24, 2024

2024 spring, Compiled by 陈奕好 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

(请改为同学的操作系统、编程环境等)

操作系统：macOS Sonoma 14.4 (23E214)

Python编程环境：PyCharm 2023.3.1 (Professional Edition)

1. 题目

22275: 二叉搜索树的遍历

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

思路：先建树，加入一个insert模块，用于每次放置新数字，postOrder用于后序输出。

代码

```
1 class TreeNode:
2     def __init__(self, value):
3         self.val = value
4         self.left = None
5         self.right = None
6
```

```

7
8 def insert(root, value):
9     if root is None:
10         return TreeNode(value)
11     elif value > root.val:
12         root.right = insert(root.right, value)
13     else:
14         root.left = insert(root.left, value)
15     return root
16
17
18 def postOrder(root):
19     if root is None:
20         return []
21     return postOrder(root.left) + postOrder(root.right) + [root.val]
22
23
24 N = int(input())
25 arr = list(map(int, input().split()))
26 root = None
27 for value in arr:
28     root = insert(root, value)
29 print(*postOrder(root))
30

```

代码运行截图 (至少包含有"Accepted")

#44402292提交状态

[查看](#)

[提交](#)

状态: **Accepted**

源代码

基本信息

#: 44402292

题目: 22275

提交人: 23n230001103

内存: 4100kB

时间: 29ms

语言: Python3

提交时间: 2024-03-25 2

```

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

def insert(root, value):
    if root is None:
        return TreeNode(value)
    elif value > root.val:
        root.right = insert(root.right, value) #
    else:
        root.left = insert(root.left, value)
    return root

def postOrder(root):
    if root is None:
        return []
    return postOrder(root.left) + postOrder(root.right) + [root.val]

```

05455: 二叉搜索树的层次遍历

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

思路：和上面一样，但追加了levelOrder板块

代码

```
1 class TreeNode:
2     def __init__(self, value):
3         self.val = value
4         self.left = None
5         self.right = None
6
7
8 def insert(root, value):
9     if root is None:
10         return TreeNode(value)
11     elif value > root.val:
12         root.right = insert(root.right, value) #
13     else:
14         root.left = insert(root.left, value)
15     return root
16
17
18 def postOrder(root):
19     if root is None:
20         return []
21     return postOrder(root.left) + postOrder(root.right) + [root.val]
22
23
24 def levelOrder(root):
25     queue = [root]
26     result = []
27     while queue:
28         current = queue.pop(0)
29         result.append(current.val)
30         if current.left is not None:
31             queue.append(current.left)
32         if current.right is not None:
33             queue.append(current.right)
34     return result
35
36
37 # N = int(input())
38 arr = list(map(int, input().split()))
39 vaild = set()
```

```

40 root = None
41 for value in arr:
42     if value in vaild:
43         continue
44     vaild.add(value)
45     root = insert(root, value)
46 print(*levelOrder(root))

```

代码运行截图 (至少包含有"Accepted")

#44402407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.val = value
        self.left = None
        self.right = None

def insert(root, value):
    if root is None:
        return TreeNode(value)
    elif value > root.val:
        root.right = insert(root.right, value) #
    else:
        root.left = insert(root.left, value)
    return root

def postOrder(root):
    if root is None:
        return []

```

基本信息

#: 44402407
 题目: 05455
 提交人: 23n2300011030(陈奕好)
 内存: 3676kB
 时间: 24ms
 语言: Python3
 提交时间: 2024-03-25 23:11:31

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：insert_key不断比较parent和本身

extract_min最为关键，把最大项移动到最开始，重新建树

min_heapify则是把列表归位

代码

```

1 class MinHeap:
2     def __init__(self):
3         self.heap = []
4

```

```

5     def parent(self, i):
6         return (i - 1) // 2
7
8     def left_child(self, i):
9         return 2 * i + 1
10
11    def right_child(self, i):
12        return 2 * i + 2
13
14    def get_min(self):
15        return self.heap[0]
16
17    def insert_key(self, k):
18        self.heap.append(k)
19        i = len(self.heap) - 1
20        while i != 0 and self.heap[self.parent(i)] > self.heap[i]:
21            self.heap[i], self.heap[self.parent(i)] = self.heap[self.parent(i)],
self.heap[i]
22            i = self.parent(i)
23
24    def decrease_key(self, i, new_val):
25        self.heap[i] = new_val
26        while i != 0 and self.heap[self.parent(i)] > self.heap[i]:
27            self.heap[i], self.heap[self.parent(i)] = self.heap[self.parent(i)],
self.heap[i]
28            i = self.parent(i)
29
30    def extract_min(self):
31        if len(self.heap) <= 0:
32            return float('inf')
33        if len(self.heap) == 1:
34            return self.heap.pop()
35        root = self.heap[0]
36        self.heap[0] = self.heap.pop()
37        self.min_heapify(0)
38        return root
39
40    def min_heapify(self, i):
41        l = self.left_child(i)
42        r = self.right_child(i)
43        smallest = i
44        if l < len(self.heap) and self.heap[l] < self.heap[i]:
45            smallest = l
46        if r < len(self.heap) and self.heap[r] < self.heap[smallest]:
47            smallest = r
48        if smallest != i:
49            self.heap[i], self.heap[smallest] = self.heap[smallest], self.heap[i]
50            self.min_heapify(smallest)
51
52
53    heap = MinHeap()
54    for i in range(int(input())):

```

```
55     opt = list(map(int, input().split()))
56     if opt[0] == 1:
57         heap.insert_key(opt[1])
58     else:
59         print(heap.extract_min())
60
61
```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

#44402782提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: **Accepted**

源代码

```
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 get_min(self):
        return self.heap[0]

    def insert_key(self, k):
        self.heap.append(k)
        i = len(self.heap) - 1
```

基本信息

#: 44402782
题目: 04078
提交人: 23n2300011030(陈奕好)
内存: 4132kB
时间: 858ms
语言: Python3
提交时间: 2024-03-26 00:02:23

22161: 哈夫曼编码树

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

思路: 抄了题解, 确实题解代码好

代码

```
1 import heapq
2
3 class Node:
4     def __init__(self, weight, char=None):
5         self.weight = weight
6         self.char = char
7         self.left = None
8         self.right = None
9
```

```
10     def __lt__(self, other):
11         if self.weight == other.weight:
12             return self.char < other.char
13         return self.weight < other.weight
14
15
16     def build_huffman_tree(characters):
17         heap = []
18         for char, weight in characters.items():
19             heapq.heappush(heap, Node(weight, char))
20
21         while len(heap) > 1:
22             left = heapq.heappop(heap)
23             right = heapq.heappop(heap)
24             merged = Node(left.weight + right.weight)
25             merged.left = left
26             merged.right = right
27             heapq.heappush(heap, merged)
28
29
30         return heap[0]
31
32
33     def encode_huffman_tree(root):
34         codes = {}
35
36         def traverse(node, code):
37             if node.char:
38                 codes[node.char] = code
39             else:
40                 traverse(node.left, code + '0')
41                 traverse(node.right, code + '1')
42
43         traverse(root, '')
44         return codes
45
46
47     def huffman_encoding(codes, string):
48         encoded = ''
49         for char in string:
50             encoded += codes[char]
51         return encoded
52
53
54     def huffman_decoding(root, encoded_string):
55         decoded = ''
56         node = root
57         for bit in encoded_string:
58             if bit == '0':
59                 node = node.left
60             else:
61                 node = node.right
```

```

62
63         if node.char:
64             decoded += node.char
65             node = root
66
67     return decoded
68
69
70 n = int(input())
71 characters = {}
72 for _ in range(n):
73     char, weight = input().split()
74     characters[char] = int(weight)
75
76 #string = input().strip()
77 #encoded_string = input().strip()
78
79 # 构建哈夫曼编码树
80 huffman_tree = build_huffman_tree(characters)
81
82 # 编码和解码
83 codes = encode_huffman_tree(huffman_tree)
84
85 strings = []
86 while True:
87     try:
88         line = input()
89         if line:
90             strings.append(line)
91         else:
92             break
93     except EOFError:
94         break
95
96 results = []
97 #print(strings)
98 for string in strings:
99     if string[0] in ('0', '1'):
100         results.append(huffman_decoding(huffman_tree, string))
101     else:
102         results.append(huffman_encoding(codes, string))
103
104 for result in results:
105     print(result)
106

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

状态: Accepted

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():
        node = Node(weight, char)
        heap.append(node)
    heapq.heapify(heap)
    while len(heap) > 1:
        left = heapq.heappop(heap)
        right = heapq.heappop(heap)
        merged = Node(left.weight + right.weight, None)
        merged.left = left
        merged.right = right
        heapq.heappush(heap, merged)
    return heapq.heappop(heap)
```

基本信息

#: 44403914
题目: 22161
提交人: 23n2300011030(陈奕好)
内存: 3696kB
时间: 26ms
语言: Python3
提交时间: 2024-03-26 09:54:22

晴问9.5: 平衡二叉树的建立

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

思路: 这里询问了copilot, 了解了“旋转”

代码

```
1 class TreeNode:
2     def __init__(self, value):
3         self.val = value
4         self.left = None
5         self.right = None
6         self.height = 1
7
8
9 def insert(root, value):
10     if root is None:
11         return TreeNode(value)
12     elif value > root.val:
13         root.right = insert(root.right, value) #
14     else:
15         root.left = insert(root.left, value)
16
17     root.height = 1 + max(get_height(root.left), get_height(root.right))
18
19     balance = get_balance(root)
20
21     # 如果节点失衡, 那么有4种情况需要处理
22     # Case 1 - 左左
```

```

23     if balance > 1 and value < root.left.val:
24         return right_rotate(root)
25
26     # Case 2 - 右右
27     if balance < -1 and value > root.right.val:
28         return left_rotate(root)
29
30     # Case 3 - 左右
31     if balance > 1 and value > root.left.val:
32         root.left = left_rotate(root.left)
33         return right_rotate(root)
34
35     # Case 4 - 右左
36     if balance < -1 and value < root.right.val:
37         root.right = right_rotate(root.right)
38         return left_rotate(root)
39     return root
40
41 def get_height(root):
42     if root is None:
43         return 0
44     else:
45         return root.height
46
47
48 def get_balance(root):
49     if root is None:
50         return 0
51     return get_height(root.left) - get_height(root.right)
52
53
54 def left_rotate(z):
55     y = z.right
56     T2 = y.left
57     y.left = z
58     z.right = T2
59     z.height = 1 + max(get_height(z.left), get_height(z.right))
60     y.height = 1 + max(get_height(y.left), get_height(y.right))
61     return y
62
63
64 def right_rotate(y):
65     x = y.left
66     T2 = x.right
67     x.right = y
68     y.left = T2
69     y.height = 1 + max(get_height(y.left), get_height(y.right))
70     x.height = 1 + max(get_height(x.left), get_height(x.right))
71     return x
72
73
74 def preOrder(root):

```

```

75     if root is None:
76         return []
77     return [root.val] + preOrder(root.left) + preOrder(root.right)
78
79
80 N = int(input())
81 arr = list(map(int, input().split()))
82
83 root = None
84 for value in arr:
85     root = insert(root, value)
86 # print(get_height(root))
87 print(*preOrder(root))
88 # print(*ans)
89
90

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

完美通过

100% 数据通过测试

运行时长: 0 ms

02524: 宗教信仰

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

思路：一开始没想到用并查集，确实好用

代码

```

1  def find(x):
2      if parent[x] != x:
3          parent[x] = find(parent[x])
4      return parent[x]
5
6  def union(x, y):

```

```
7     parent[find(x)] = find(y)
8
9     case = 0
10    while True:
11        n, m = map(int, input().split())
12        if n == 0 and m == 0:
13            break
14        parent = list(range(n+1))
15        for _ in range(m):
16            i, j = map(int, input().split())
17            union(i, j)
18        religions = len(set(find(i) for i in range(1, n+1)))
19        case += 1
20        print("Case %d: %d" % (case, religions))
21
```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

#44406889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:
        parent[x] = find(parent[x])
    return parent[x]

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

case = 0
while True:
    n, m = map(int, input().split())
    if n == 0 and m == 0:
        break
    parent = list(range(n+1))
    for _ in range(m):
        i, j = map(int, input().split())
        union(i, j)
    religions = len(set(find(i) for i in range(1, n+1)))
    case += 1
    print("Case %d: %d" % (case, religions))
```

基本信息

#: 44406889
题目: 02524
提交人: 23n2300011030(陈奕好)
内存: 11020kB
时间: 1224ms
语言: Python3
提交时间: 2024-03-26 15:10:52

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。

并查集大法

```
1 def find(x):
2     if parent[x] != x:
3         parent[x] = find(parent[x])
4     return parent[x]
5
6 def union(x, y):
7     parent[find(x)] = find(y)
```