

Assignment #8: 树为主

Updated 1704 GMT+8 Apr 8, 2025

2025 spring, Compiled by 同学的姓名、院系

说明:

1. 解题与记录:

对于每一个题目，请提供其解题思路（可选），并附上使用Python或C++编写的源代码（确保已在OpenJudge, Codeforces, LeetCode等平台上获得Accepted）。请将这些信息连同显示“Accepted”的截图一起填写到下方的作业模板中。（推荐使用Typora <https://typoraio.cn> 进行编辑，当然你也可以选择Word。）无论题目是否已通过，请标明每个题目大致花费的时间。

2. **提交安排:** 提交时，请首先上传PDF格式的文件，并将.md或.doc格式的文件作为附件上传至右侧的“作业评论”区。确保你的Canvas账户有一个清晰可见的头像，提交的文件为PDF格式，并且“作业评论”区包含上传的.md或.doc附件。

3. **延迟提交:** 如果你预计无法在截止日期前提交作业，请提前告知具体原因。这有助于我们了解情况并可能为你提供适当的延期或其他帮助。

请按照上述指导认真准备和提交作业，以保证顺利完成课程要求。

1. 题目

LC108.将有序数组转换为二叉树

dfs, <https://leetcode.cn/problems/convert-sorted-array-to-binary-search-tree/>

思路:

代码:

```
# Definition for a binary tree node.
class TreeNode(object):
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
class Solution(object):
    def sortedArrayToBST(self, nums):
        """
        :type nums: List[int]
        :rtype: Optional[TreeNode]
        """
        def dfs(nums):
            mid=len(nums)//2
            lst=[None]*len(nums)
            for i in range(len(nums)):
```

```

        lst[i]=TreeNode(nums[i])
    if len(nums)==1:
        return lst[0]
    if len(nums)==2:
        lst[0].right=lst[1]
        return lst[0]
    lst[mid].left=dfs(nums[:mid])
    lst[mid].right=dfs(nums[mid+1:])
    return lst[mid]
return dfs(nums)

```

代码运行截图 (至少包含有"Accepted")



M27928:遍历树

adjacency list, dfs, <http://cs101.openjudge.cn/practice/27928/>

思路:

代码:

```

import sys
sys.setrecursionlimit(1<<30)
class TreeNode:
    def __init__(self,val=0,children=None):
        self.val=val
        self.children=[]

def dfs(root):
    if not root.children:
        print(root.val)
        return
    lst=[root]+root.children
    lst.sort(key=lambda x:x.val)
    for i in lst:
        if i==root:
            print(root.val)

```

```

        continue
    dfs(i)

n=int(input())
record={}
lst=[]
for _ in range(n):
    s=[int(x) for x in input().split()]
    lst.append(s)
    for i in s[1:]:
        if i not in record:
            record[i]=TreeNode(i)
nodes=[]
for i in lst:
    arr=i
    if arr[0] in record:
        node=record[arr[0]]
    else:
        node=TreeNode(arr[0])
    for j in arr[1:]:
        node.children.append(record[j])
    nodes.append(node)
child=list(record.values())
root=list(filter(lambda x:x not in child,nodes))[0]
dfs(root)

```

代码运行截图 (至少包含有"Accepted")

状态: Accepted

源代码

```
import sys
sys.setrecursionlimit(1<<30)
class TreeNode:
    def __init__(self, val=0, children=None):
        self.val=val
        self.children=[]

def dfs(root):
    if not root.children:
        print(root.val)
        return
    lst=[root]+root.children
    lst.sort(key=lambda x:x.val)
    for i in lst:
        if i==root:
            print(root.val)
            continue
        dfs(i)

n=int(input())
record={}
```

LC129.求根节点到叶节点数字之和

dfs, <https://leetcode.cn/problems/sum-root-to-leaf-numbers/>

思路:

代码:

```
# Definition for a binary tree node.
class TreeNode(object):
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
class Solution(object):
    def __init__(self):
        self.ans=0
    def sumNumbers(self, root):
        """
        :type root: Optional[TreeNode]
        :rtype: int
        """
        def dfs(root,s):
```

```

s+=str(root.val)
if not root.left and not root.right:
    self.ans+=int(s)
    return
if root.left:
    dfs(root.left,s)
if root.right:
    dfs(root.right,s)
dfs(root, '')
return self.ans

```

代码运行截图 (至少包含有"Accepted")



M22158:根据二叉树前中序序列建树

tree, <http://cs101.openjudge.cn/practice/22158/>

思路:

代码:

```

class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val=val
        self.left=left
        self.right=right

def post(root):
    ans=''

```

```

    if root:
        ans+=post(root.left)
        ans+=post(root.right)
        ans+=root.val
    return ans

def build(pre,ino):
    if not pre or not ino:
        return None
    rootval=pre[0]
    root=TreeNode(rootval)
    in_index=ino.index(rootval)
    root.left=build(pre[1:1+in_index],ino[:in_index])
    root.right=build(pre[1+in_index:],ino[1+in_index:])
    return root

while True:
    try:
        lst1=input()
        lst2=input()
        print(post(build(lst1,lst2)))
    except EOFError:
        break

```

代码运行截图 (至少包含有"Accepted")

4888/080提交状态

状态: Accepted

代码

```

class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val=val
        self.left=left
        self.right=right

def post(root):
    ans=''
    if root:
        ans+=post(root.left)
        ans+=post(root.right)
        ans+=root.val
    return ans

def build(pre, ino):
    if not pre or not ino:
        return None
    rootval=pre[0]
    root=TreeNode(rootval)
    in_index=ino.index(rootval)
    root.left=build(pre[1:1+in_index],ino[:in_index])
    root.right=build(pre[1+in_index:],ino[1+in_index:])

```

M24729:括号嵌套树

dfs, stack, <http://cs101.openjudge.cn/practice/24729/>

思路：吃了个大亏，啊啊啊我就说怎么代码运行的结果这么奇怪

问题 1: **TreeNode** 初始化中的默认参数问题

Python

深色版本 | 复制

```
def __init__(self, val=0, children=[]):
    self.val = val
    self.children = children
```

- 这里的 `children=[]` 是一个常见的陷阱。Python 中，默认参数只会在函数定义时被初始化一次，因此所有使用默认参数的 `TreeNode` 对象会共享同一个 `children` 列表。
- 这会导致意外行为，例如多个节点的 `children` 被错误地绑定到同一个列表。

代码：

```
import sys
sys.setrecursionlimit(1<<30)
class TreeNode:
    def __init__(self, val=0, children=None):
        self.val=val
        self.children=[]

def pre(root):
    out=''
    if root:
        out+=root.val
        for i in root.children:
            out+=pre(i)
    return out

def post(root):
    out=''
    if root:
        for i in root.children:
            out+=post(i)
        out+=root.val
    return out

s=input()
root=TreeNode(s[0])
stack=[]
nodes=[]
cur=root
for i in s[1:]:
    if i=='(':
        stack.append(i)
        nodes.append(cur)
    elif i==',':
        continue
```

```

elif i==')':
    stack.pop()
    nodes.pop()
else:
    node=TreeNode(i)
    nodes[-1].children.append(node)
    cur=node
print(pre(root))
print(post(root))

```

代码运行截图 (至少包含有"Accepted")

状态: Accepted

源代码

```

import sys
sys.setrecursionlimit(1<<30)
class TreeNode:
    def __init__(self, val=0, children=None):
        self.val=val
        self.children=[]

def pre(root):
    out=''
    if root:
        out+=root.val
        for i in root.children:
            out+=pre(i)
    return out

def post(root):
    out=''
    if root:
        for i in root.children:

```

LC3510.移除最小数对使数组有序II

doubly-linked list + heap, <https://leetcode.cn/problems/minimum-pair-removal-to-sort-array-ii/>

思路:

代码:

```

import heapq

class Solution(object):

```




```

def minimumPairRemoval(self, nums):
    """
    :type nums: List[int]
    :rtype: int
    """
    n=len(nums)
    if n<=1:
        return 0
    l=[i-1 for i in range(n)]
    r=[i+1 for i in range(n)]
    alive=[True]*n
    bad=0
    v=nums[:]
    for i in range(n-1):
        if v[i]>v[i+1]:
            bad+=1
    pq=[]
    for i in range(n-1):
        heapq.heappush(pq,(v[i]+v[i+1],i))
    cnt=0
    while bad>0:
        s,i=heapq.heappop(pq)
        j=r[i]
        if j>=n or not alive[i] or not alive[j] or v[i]+v[j]!=s:
            continue
        pi,nj=l[i],r[j]
        if pi>=0 and alive[pi] and v[pi]>v[i]:
            bad-=1
        if v[i]>v[j]:
            bad-=1
        if nj<n and alive[nj] and v[j]>v[nj]:
            bad-=1
        v[i]=s
        alive[j]=False
        r[i]=nj
        if nj<n:
            l[nj]=i
        if pi>=0 and alive[pi] and v[pi]>v[i]:
            bad+=1
        if nj<n and alive[nj] and v[i]>v[nj]:
            bad+=1
        if pi>=0 and alive[pi]:
            heapq.heappush(pq,(v[pi]+v[i],pi))
        if nj<n and alive[nj]:
            heapq.heappush(pq,(v[i]+v[nj],i))
        cnt+=1
    return cnt

```

代码运行截图 (至少包含有"Accepted")

通过 680 / 680 个通过的测试用例

 犹能颠却 提交于 2025.04.12 20:10



面向在校学生的专享特惠

完成认证享 7 折 Plus 会员，享受更多学业

2. 学习总结和收获

如果发现作业题目相对简单，有否寻找额外的练习题目，如“数算2025spring每日选做”、LeetCode、Codeforces、洛谷等网站上的题目。

啊啊啊该死的期中寄，要寄了

由于复习考试，这周基本都没怎么学数算，但是下周还是有很多考试啊啊

最后一题好难啊，自己写的代码总是超时，看答案还没怎么看懂