# Assignment #7: April 月考

Updated 1557 GMT+8 Apr 3, 2024

2024 spring, Complied by 同学的姓名、院系

**说明：**

1）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。

2）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。

3）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 Ventura 13.4.1 (c)

Python编程环境：Spyder IDE 5.2.2, PyCharm 2023.1.4 (Professional Edition)

C/C++编程环境：Mac terminal vi (version 9.0.1424), g++/gcc (Apple clang version 14.0.3, clang-1403.0.22.14.1)

## 1. 题目

### 27706: 逐词倒放

http://cs101.openjudge.cn/practice/27706/

思路：

栈，送分题

代码

```
ipt=[str(x) for x in input().split()]
stack=[]
ans=[]
for j in range(len(ipt)):
    stack.append(ipt[j])
while stack:
    i=stack.pop()
    ans.append(i)
print(' '.join(ans))
```

代码运行截图 （至少包含有"Accepted"）


## 27951: 机器翻译

http://cs101.openjudge.cn/practice/27951/

思路：

栈、根据题意做即可；竟然不需要考虑时间复杂度。。。直接在栈里查找就可以

代码

```
MNlst=[int(x) for x in input().split()]
M,N=MNlst[0],MNlst[1]
word_lst=[int(x) for x in input().split()]
instack=[]
count=0
for i in range(N):
    ipt=word_lst[i]
    if ipt not in instack:
        instack.append(ipt)
        count+=1
    if len(instack)>M:
        instack.pop(0)
print(count)
```

代码运行截图 （至少包含有"Accepted"）

**#44542686提交状态**

查看 提交 统计 提问

状态: Accepted

源代码

```
MNlst=[int(x) for x in input().split()]
M,N=MNlst[0],MNlst[1]
word_lst=[int(x) for x in input().split()]
instack=[]
count=0
for i in range(N):
    ipt=word_lst[i]
    if ipt not in instack:
        instack.append(ipt)
        count+=1
    if len(instack)>M:
        instack.pop(0)
print(count)
```

基本信息

#: 44542686
题目: 27951
提交人: zxk
内存: 3648kB
时间: 27ms
语言: Python3
提交时间: 2024-04-06 09:04:10



English 帮助 关于

# 27932: Less or Equal

http://cs101.openjudge.cn/practice/27932/

思路:

这道题竟然是卡的题。。。。明明很简单；数据估计很大，用了heapq，结果发现直接得到的堆并不是有序的。。。然后通过heappop做了升序列表，第一步解决；关于截断是最坑的。。。普通的判断很好实现，问题出在k=0，只要第一个数大于1（不能等于1）就可以算

代码

```
import heapq

nklst=[int(x) for x in input().split()]
n,k=nklst[0],nklst[1]
data=[int(x) for x in input().split()]

heapq.heapify(data)

std_data=[]
while data:
    std_data.append(heapq.heappop(data))

stack=[]
while std_data:
    stack.append(std_data.pop(0))
    if len(stack)==k:
        break
if len(stack)==k:
    if std_data and std_data[0]!=stack[-1]:
        print(stack[-1])
    elif not std_data:
        print(stack[-1])
    else:
        print(-1)
else:
    if k==0:
```

```
        if stack and stack[0]>1:
            print(1)
        else:
            print(-1)
    else:
        print(-1)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

**#44523516提交状态**

查看 提交 统计 提问

状态: Accepted

源代码

```
import heapq

nklst=[int(x) for x in input().split()]
n,k=nklst[0],nklst[1]
data=[int(x) for x in input().split()]

heapq.heapify(data)

std_data=[]
while data:
    std_data.append(heapq.heappop(data))

stack=[]
while std_data:
    stack.append(std_data.pop(0))
    if len(stack)==k:
        break
if len(stack)==k:
    if std_data and std_data[0]!=stack[-1]:
        print(stack[-1])
    elif not std_data:
        print(stack[-1])
    else:
        print(-1)
else:
    if k==0:
        if stack and stack[0]>1:
            print(1)
        else:
            print(-1)
    else:
        print(-1)
```

基本信息

#: 44523516
题目: 27932
提交人: zxk
内存: 9876kB
时间: 469ms
语言: Python3
提交时间: 2024-04-03 23:32:42

## 27948: FBI树

http://cs101.openjudge.cn/practice/27948/

思路:

这题挺简单的，逆天的问题出在了好久没用python的指数，是**不是^\。。。latex和mma打多了就忘了，而且python的^\是有意义的，导致总是不知道为啥就没法切片，试了很多切片方法。。。最后不得已喂给gpt，告诉我写错了字符。。。唉

代码

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

```
    def postorder(self):
        rtn=[]
        if self.left:
            rtn.extend(self.left.postorder())
        if self.right:
            rtn.extend(self.right.postorder())
        rtn+=[self.value]
        return rtn

def BuildTree(astr,n):
    if n==0:
        if astr=='1':
            typ='I'
        else:
            typ='B'
        node=Node(typ)
        return node
    else:
        leftchild=BuildTree(astr[:2**(n-1)],n-1)
        rightchild=BuildTree(astr[2**(n-1):],n-1)
        if leftchild.value==rightchild.value:
            typ=leftchild.value
        else:
            typ='F'
        node=Node(typ)
        node.left=leftchild
        node.right=rightchild
        return node

N=int(input())
S=str(input())
Tree=BuildTree(S,N)
print(''.join(Tree.postorder()))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

**#44542894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.value=val
        self.left=None
        self.right=None

    def postorder(self):
        rtn=[]
        if self.left:
            rtn.extend(self.left.postorder())
        if self.right:
            rtn.extend(self.right.postorder())
        rtn+=[self.value]
        return rtn

def BuildTree(astr,n):
    if n==0:
        if astr=='1':
            typ='I'
        else:
            typ='B'
        node=Node(typ)
        return node
    else:
        leftchild=BuildTree(astr[:2**(n-1)],n-1)
        rightchild=BuildTree(astr[2**(n-1):],n-1)
        if leftchild.value==rightchild.value:
            typ=leftchild.value
        else:
            typ='F'
        node=Node(typ)
        node.left=leftchild
        node.right=rightchild
        return node

N=int(input())
S=str(input())
Tree=BuildTree(S,N)
print(''.join(Tree.postorder()))
```

基本信息

#: 44542894
题目: 27948
提交人: zxk
内存: 3868kB
时间: 26ms
语言: Python3
提交时间: 2024-04-06 09:14:02



English 帮助 关于

# 27925: 小组队列

http://cs101.openjudge.cn/practice/27925/

思路:

这道题还是挺有趣的，感觉数据量会比较大，所以准备建树，但是好像树的层级也特别少啊。。。遂不知道建一个什么数据结构好；最终决定设计一个双层的队列，外层的值表示组号，内层的值是学生编号，也因此需要先建立一个人--组号的字典用于查找入队人的组号；对于入队的人，先检索他的组是否在整体队列的子队列里，如果是就找到组号的位置，在这个位置的队列结构里插入这个人，如果不在，那就新建一个组，并把ta放到组里，然后把新的组插到整体队列里；对于出队，直接从整体队列的以一个组的队列里出一个人，之后检查一下是不是变成了空组，避免之后入队的人进错组

代码

```
class queue:
    def __init__(self,v):
        self.que=[]
        self.name=v

    def show_que(self):
        s=[]
        for x in self.que:
```

```
            s.append(x.name)
        return s

t=int(input())
teams_lst=[]
people={}
for _ in range(t):
    ateam=[int(x) for x in input().split()]
    for p in ateam:
        people[p]=_+1
ans_lst=[]
Queue=queue('all')
while True:
    ipt=[x for x in input().split()]
    if len(ipt)==2:
        pe=int(ipt[1])
        tm=people[pe]
        if tm in Queue.show_que():
            idx=Queue.show_que().index(tm)
            Queue.que[idx].que.append(pe)
        else:
            it=queue(tm)
            it.que.append(pe)
            Queue.que.append(it)
    else:
        if ipt[0]=='DEQUEUE':
            a=Queue.que[0].que.pop(0)
            if Queue.que[0].que==[]:
                Queue.que.pop(0)
            ans_lst.append(a)
        else:
            break
for x in ans_lst:
    print(x)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

**#44543068提交状态** 查看 提交 统计 提问

状态: Accepted

源代码

```
class queue:
    def __init__(self,v):
        self.que=[]
        self.name=v

    def show_que(self):
        s=[]
        for x in self.que:
            s.append(x.name)
        return s

t=int(input())
teams_lst=[]
people={}
for _ in range(t):
    ateam=[int(x) for x in input().split()]
    for p in ateam:
        people[p]=_+1
ans_lst=[]
Queue=queue('all')
while True:
    ipt=[x for x in input().split()]
    if len(ipt)==2:
        pe=int(ipt[1])
        tm=people[pe]
        if tm in Queue.show_que():
            idx=Queue.show_que().index(tm)
            Queue.que[idx].que.append(pe)
        else:
            it=queue(tm)
            it.que.append(pe)
            Queue.que.append(it)
    else:
        if ipt[0]=='DEQUEUE':
            a=Queue.que[0].que.pop(0)
            if Queue.que[0].que==[]:
                Queue.que.pop(0)
            ans_lst.append(a)
        else:
            break
for x in ans_lst:
    print(x)
```

基本信息

| | |
|---|---|
| #: | 44543068 |
| 题目: | 27925 |
| 提交人: | zxk |
| 内存: | 5372kB |
| 时间: | 89ms |
| 语言: | Python3 |
| 提交时间: | 2024-04-06 09:21:21 |



# 27928: 遍历树

http://cs101.openjudge.cn/practice/27928/

思路:

这道题是一个没见过的遍历方法，光遍历是怎么递归的都快搞不清了。。。搞清楚之后发现也挺难实现的？直觉上决定不用做一个树的数据结构，用父子关系即可，遂开始考虑字典；因为输入的数据是以parent--children的结构，所以可以先建立一个节点字典，用parent做key，用children列表做value（排序一次，用于后续的遍历），但是这时相当于只记录了向下的关系，并没建立向上的关系，于是考虑了再建一个根字典，即将每个child做key，节点的值做value，这时只要从任意一个节点开始递归查找，只要某个节点的父节点并未出现在key中，说明这个父节点就是整体的根节点；之后开始遍历，对于某个子树，如果它没子节点，说明只需要输出它自己的值即可；如果它有子节点，就把值放到子节点遍历结果的中间；返回遍历结果；之后我们就需要找到根节点，然后对根节点进行遍历，输出即可

代码

```
def traversal(root):
    ans = []
    chd = node_dct[root].copy()
```

```
    if not chd:
        return [root]
    else:
        while chd:
            new_root = chd.pop(0)
            if new_root < root:
                ans.extend(traversal(new_root))
            else:
                chd.insert(0,new_root)
                break
        ans.append(root)
        while chd:
            new_root = chd.pop(0)
            ans.extend(traversal(new_root))
        return ans


def find_root(p):
    if p not in root_lst:
        return p
    else:
        return find_root(parent_dct[p])

n = int(input())
node_dct = {}
parent_dct = {}
for i in range(n):
    ipt = [int(x) for x in input().split()]
    root = ipt.pop(0)
    children = sorted(ipt)
    node_dct[root] = children

    for x in children:
        parent_dct[x] = root
root_lst = list(parent_dct.keys())
Root = find_root(root_lst[0])
ans = traversal(Root)
for i in ans:
    print(i)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

**#44521160提交状态**

查看 提交 统计 提问

状态: Accepted

基本信息

#: 44521160
题目: 27928
提交人: zxk
内存: 3716kB
时间: 26ms
语言: Python3
提交时间: 2024-04-03 19:38:54

源代码

```
def traversal(root):
    ans = []
    chd = node_dct[root].copy()
    if not chd:
        return [root]
    else:
        while chd:
            new_root = chd.pop(0)
            if new_root < root:
                ans.extend(traversal(new_root))
            else:
                chd.insert(0,new_root)
                break
        ans.append(root)
        while chd:
            new_root = chd.pop(0)
            ans.extend(traversal(new_root))
        return ans


def find_root(p):
    if p not in root_lst:
        return p
    else:
        return find_root(parent_dct[p])

n = int(input())
node_dct = {}
parent_dct = {}
for i in range(n):
    ipt = [int(x) for x in input().split()]
    root = ipt.pop(0)
    children = sorted(ipt)
    node_dct[root] = children

    for x in children:
        parent_dct[x] = root
root_lst = list(parent_dct.keys())
Root = find_root(root_lst[0])
ans = traversal(Root)
for i in ans:
    print(i)
```

# 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。

本次考试AC4，感觉自己的代码能力有了比较大的提升，没做出来的两道题一道是遍历树时间不够了，一道是LoE的k=0没考虑到

本次考试收获如下：

.**

·树的结构并不是唯一的，也不是说树就必须递归；适当修改可以用在比如队列上

·树的结构也并不一定做成Node结构，用字典、列表也可以比较好地实现

·几种快速排序的方法还需要再熟悉熟悉

·要看数据的范围，尤其需要考虑边界条件

希望下次考试能冲一下AK