**## 1. 题目**

**### E05344:最后的最后**

http://cs101.openjudge.cn/practice/05344/

思路：队列法解决的约瑟夫问题，先创造一个n+1的队列，然后向左旋转k-1步，使应该被排除的人移动到队首，然后重复步骤每次弹出队首的人

代码：

from collections import deque  
n,k=map(int,input().split())  
  
queue=deque(range(1,n+1))  
  
ans=[]  
while len(queue) > 1:  
 queue.rotate(-(k-1))  
 ans.append(queue.popleft())  
if queue:  
 ans.append(queue[0])  
print(" ".join(map(str,ans[:-1])))

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

  
大约用时：15分钟

**### M02774: 木材加工**

binary search, http://cs101.openjudge.cn/practice/02774/

思路：二分查找，每次取当前范围的中点，并计算能切出的木材数量，若>K说明可以尝试更长的长度，更新最大长度max\_l，若<k则说明过长，则需要再左区间寻找更短的长度。

代码：

N,K=map(int,input().split())  
woods=[]  
for i in range(N):  
 wood=int(input())  
 woods.append(wood)  
  
largest=max(woods)  
left = 1  
right = largest  
max\_l=0  
  
while left <= right:  
 mid=(right+left)//2  
 count = 0  
 for j in woods:  
 count += j // mid  
  
 if count>=K:  
 max\_l=mid  
 left=mid+1  
  
 if count<K:  
 right=mid-1  
  
print(max\_l)

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

  
大约用时：15分钟

**### M07161:森林的带度数层次序列存储**

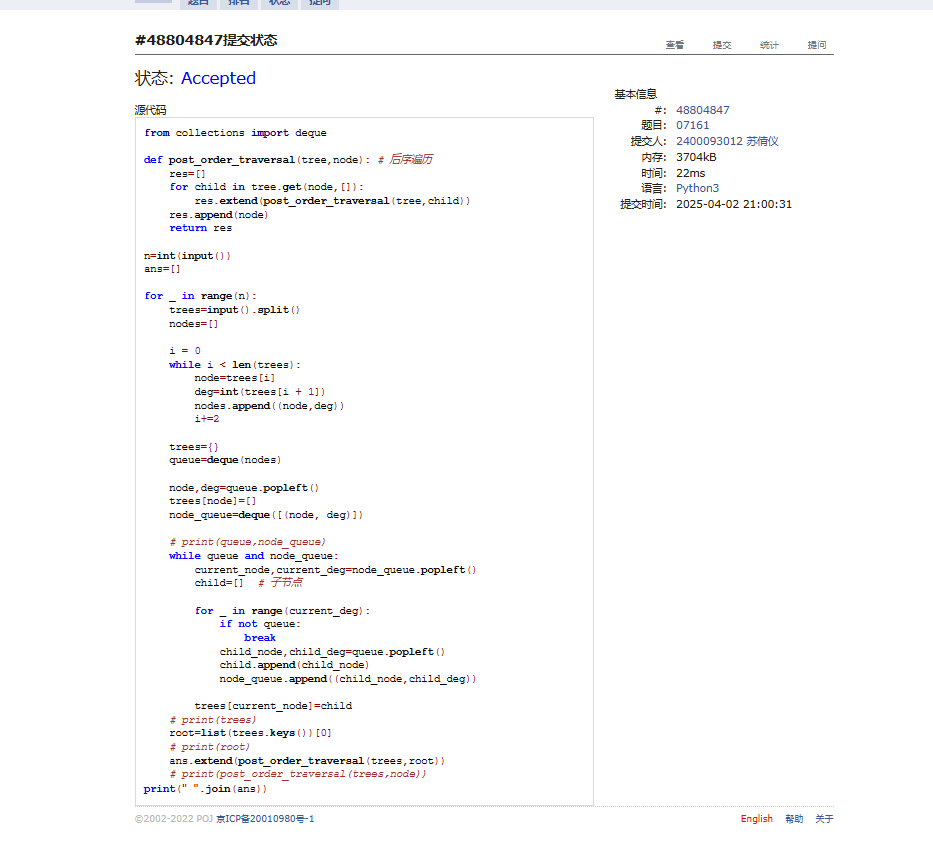
tree, http://cs101.openjudge.cn/practice/07161/

思路：用了队列和后序遍历，先每次读取一棵树的节点名称和度数，保存为 nodes 列表，然后从 queue 中取出第一个元素，表示树的根节点 node 和它的子节点数量 deg，再创建一个新队列 node\_queue用来存储当前正在处理的节点及其度数，接着遍历当前节点的度数（每次循环处理一个子节点），从queue中取出下一个节点（也就是当前节点的子节点），并添加到代表当前节点的child列表再和其度数一起添加到node\_queue准备处理下一个节点直到queue和node\_queue为空，最后将当前节点 current\_node 和 child 列表加入 trees 字典中，最后进行后序遍历，递归访问每个节点的子节点，然后将当前节点加入结果 res，并加入ans总列表中。

代码：

from collections import deque  
  
def post\_order\_traversal(tree,node): # 后序遍历  
 res=[]  
 for child in tree.get(node,[]):  
 res.extend(post\_order\_traversal(tree,child))  
 res.append(node)  
 return res  
  
n=int(input())  
ans=[]  
  
for \_ in range(n):  
 trees=input().split()  
 nodes=[]  
  
 i = 0  
 while i < len(trees):  
 node=trees[i]  
 deg=int(trees[i + 1])  
 nodes.append((node,deg))  
 i+=2  
  
 trees={}  
 queue=deque(nodes)  
  
 node,deg=queue.popleft()  
 trees[node]=[]  
 node\_queue=deque([(node, deg)])  
  
 # print(queue,node\_queue)  
 while queue and node\_queue:  
 current\_node,current\_deg=node\_queue.popleft()  
 child=[] # 子节点  
  
 for \_ in range(current\_deg):  
 if not queue:  
 break  
 child\_node,child\_deg=queue.popleft()  
 child.append(child\_node)  
 node\_queue.append((child\_node,child\_deg))  
  
 trees[current\_node]=child  
 # print(trees)  
 root=list(trees.keys())[0]  
 # print(root)  
 ans.extend(post\_order\_traversal(trees,root))  
 # print(post\_order\_traversal(trees,node))  
print(" ".join(ans))

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

  
大约用时：1小时30分钟

**### M18156:寻找离目标数最近的两数之和**

two pointers, http://cs101.openjudge.cn/practice/18156/

思路：用两个指针分别指向排序后列表的第一个和最后一个元素，并计算当前两个数之和，其中closest\_sum用于记录当前最接近目标的和，如果当前和与目标 T 的差更小，更新 closest\_sum 为 current\_sum，如果当前和与最接近的和的差相等，则比较两者的大小，选择较小的和作为最接近的和，如果当前和等于目标 T，可以直接返回当前和，而如果小于/大于目标和，则尝试增加/减少其中一个数，将指针向右/左移。

代码：

T=int(input())  
N=list(map(int,input().split()))  
S=sorted(N)  
left=0  
right=len(S)-1  
current\_sum=0  
closest\_sum = S[left] + S[right]  
while left < right:  
 current\_sum=S[left]+S[right]  
 if abs(current\_sum-T) < abs(closest\_sum-T): # 如果当前和比之前的和更接近T  
 closest\_sum = current\_sum # 更新最接近的和  
 elif abs(current\_sum - T) == abs(closest\_sum - T):# 如果当前和与最接近的和==T  
 if current\_sum < closest\_sum: # 如果当前和更小，则更新  
 closest\_sum = current\_sum # 如果存在多个解，则输出数值较小的那个  
 if current\_sum == T:  
 break  
 elif current\_sum < T:  
 left += 1  
 elif current\_sum > T:  
 right -= 1  
print(closest\_sum)

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

  
大约用时：30分钟

**### M18159:个位为 1 的质数个数**

sieve, http://cs101.openjudge.cn/practice/18159/

思路：使用了埃拉托斯特尼筛法，初始化一个大小为 num+1 的布尔数组 prime，并将 prime[0] 和 prime[1] 设置为 False（因为 0 和 1 不是素数），然后从2开始标记素数为False，并计算所有小于10^6的素数，判断从2到num的值是否个位数为1，最后输出符合条件的数。

代码：

def shaifa(num):  
 prime = [True] \* (num + 1)  
 prime[0], prime[1] = False, False  
 for i in range(2, int(num \*\* 0.5) + 1):  
 if prime[i]:  
 for j in range(i \* i, num + 1, i):  
 prime[j] = False  
 return prime  
  
prime=shaifa(10\*\*6)  
n=int(input())  
for case\_num, \_ in enumerate(range(n), start=1):  
 num=int(input())  
 if num == 0:  
 print(f"Case{case\_num}:")  
 print("NULL")  
 continue  
  
 ans=[]  
 for i in range(2,num):  
 if prime[i] and i%10==1:  
 ans.append(i)  
 print(f"Case{case\_num}:")  
 if ans:  
 print(" ".join(map(str, ans)))  
 else:  
 print("NULL")

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

  
大约用时：45分钟

**### M28127:北大夺冠**

hash table, http://cs101.openjudge.cn/practice/28127/

思路：用字典记录每个队伍的成绩（通关题目，提交次数），然后初始化一个rank列表分开记录各个队伍的通关题目数量和提交次数，再降序排序（先通关题目数再到提交次数再到队伍名字的字典序），最后输出前12个结果。

代码：

n=int(input())  
dict={}  
for \_ in range(n):  
 team,ques,yn=input().split(",")  
 if team not in dict:  
 dict[team]={0:set(),1:0}  
 # elif team in dict:  
 dict[team][1]+=1  
 if yn=="yes":  
 dict[team][0].add(ques)  
# print(dict)  
rank=[]  
for i,j in dict.items():  
 a=len(j[0])  
 sub=j[1]  
 rank.append((i,a,sub))  
rank.sort(key=lambda x:(-x[1],x[2],x[0]))  
  
for r,(a,b,c) in enumerate(rank[:12],start=1):  
 print(f'{r} {a} {b} {c}')

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

  
大约用时：30分钟

**## 2. 学习总结和收获**

<mark>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。</mark>

第一二题感觉很熟悉hhh几乎是看到题目一瞬间就想到思路了（但是基础不好，写起来还是犯了很多莫名其妙的错。第四题也能做得出来，但是一开始忘记判断“最接近”和“如果存在多个解，则输出数值较小的那个”的情况了。第五题上网找了筛法，仿照着上网找的思路写了一次结果TLE了，后来才知道要把筛法单独放出去，再先解决10^6素数才是“不需要每一次都从头开始判断”，然后也是漏了“不包括自己”这个条件，卡了好一会才做出hhh。第六题感觉还挺简单，但是条件还挺多的，对我来说还是很容易写错，但是作为第六题这个难度我很庆幸了。第三题可恶的树，拼尽全力无法战胜，写了三次不同的代码，一开始用栈，然后又写了两次队列，结果都是做不出，最后还是问了gpt然后再慢慢理解整个思路。这次AC了三题，光是第三题就卡了我一小时，我得到的教训是下次看到树我会直接跳过的TvT。