Assignment #7: 20250402 Mock Exam

Updated 1624 GMT+8 Apr 2, 2025

2025 spring, Complied by <mark>同学的姓名、院系</mark>

姓名：李彦臻
学号：2300010821
学院：数学科学学院
AC 数：4 道题

> **说明：**
>
> 1. **月考**：AC?<mark>（请改为同学的通过数）</mark> 。考试题目都在“题库（包括计概、数算题目）”里面，按照数字题号能找到，可以重新提交。作业中提交自己最满意版本的代码和截图。
>
> 2. **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。
>
> 3. **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。
>
> 4. **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

E05344:最后的最后

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

思路：利用循环链表即可；先创建一个循环链表（节点包含编号 1 到 n）。随后，找到第 k 个节点，并将其从链表中移除；接着，重复这个过程，直到只剩下一个节点就 ok 了。
用时：10mins

代码：

```python

```
```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
class CircularLinkedList:
    def __init__(self):
        self.head = None

    # 创建含有 n 个节点的循环链表
    def create(self, n):
        self.head = Node(1)
        current = self.head
        for i in range(2, n + 1):
            current.next = Node(i)
            current = current.next
        current.next = self.head  # 构成循环

    # 模拟约瑟夫问题
    def joseph(self, k):
        result = []
        current = self.head
        prev = None

        # 找到最后一个节点（方便删除操作）
        while current.next != self.head:
            current = current.next
        prev = current  # prev 是最后一个节点
        current = self.head  # current 从头开始

        # 直到只剩下一个节点
        while current.next != current:
            # 移动 k-1 步
            for _ in range(k - 1):
                prev = current
                current = current.next
            # 删除第 k 个节点
            result.append(current.data)
            prev.next = current.next
            current = current.next
```

```
        return result

n, k = map(int, input().split())
# 创建并处理循环链表
cll = CircularLinkedList()
cll.create(n)
killed = cll.joseph(k)
# 输出结果 print(" ".join(map(str, killed)))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

M02774: 木材加工

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

思路：直接使用二分查找即可，思路很常规，就是构造一个函数 suit(a)，用于验证长度 a 是否符合题意；随后，先验证长度为 1 是否 ok，如果 ok，就构造 up = 10001（不合题的下限）和 down = 1（合题的上限），并不断的验证中值是否成立，进而将上限和下限逼近即可；最后，在 up-down=1 的时候输出 down 即可~
用时：10mins

代码：

```python

```
```
n, k = map(int,input().split())
length = []
for i in range(n):
    length.append(int(input()))

def suit(a): # 用于验证长度 a 是否符合题意的函数
    sum = 0 # 能够切割出来的原木个数
    for i in range(n):
        b = length[i] // a
        sum += b
    if sum >= k: # 合题
        return 1
    else:return 0 # 否则不合题

if suit(1) == 0: # 先验证 1 是否 ok
    print("0") # 不 ok 就输出 0
else:
    up = 10001 # 不合题的下限
    down = 1 # 合题的上限
    while up - down > 1:
        a = (up + down) // 2
        if suit(a) == 0:
            up = a
        else:down = a
    print(down)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

M07161:森林的带度数层次序列存储

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

思路：

代码：

```python

```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

M18156:寻找离目标数最近的两数之和

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

思路：直接使用双指针方法即可~（对于每一次判断，首先，如果当下值是 t 则就已经结束，结束循环；接着，更新最佳值；最后，根据当下值与 t 的大小关系移动指针）
用时：15mins

代码：

```python
```

```
t = int(input())
num = list(map(int,input().split()))
num.sort()
n = len(num)

best = num[0] + num[n - 1] # 当下的最佳逼近
i = 0 # 左指针
j = n - 1 # 右指针
while i < j:
    k = num[i] + num[j]

    # 首先，如果 k 是 t 则就已经结束
    if k == t:
        best = t
        break

    # 接着，更新最佳值
    if abs(k - t) < abs(best - t):
        best = k
    elif abs(k - t) == abs(best - t):
        best = min(k, best)

    # 最后，移动指针
    if k > t:
        j -= 1
    else:i += 1

print(best)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

M18159:个位为 1 的质数个数

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

思路：先用欧拉筛找出 1~10001 内的素数，并只保留个位数为 1 的素数；接着，利用 bisect 函数处理每个案例即可（因为总共只有 306 个符合题意的素数，因此使用用 bisect 方法不会 TLE（bisect 的时间复杂度是 o(lnn)）！ ）~
用时：15mins

代码：

```python
# 先用欧拉筛的方法把 1~10001 内的素数找出来
primes=[]
status=['True']*(10**4 + 1)# 用对应位置的 T/F 来记录每个数是否为素数
for i in range(2,10**4 + 1):
    if status[i] == 'True':# 如果轮到 i 的时候仍没被扔掉，则一定是素数！
        primes.append(i)
    # 接下来进行下一步操作：把 i 的倍数扔掉
    # 欧拉筛的第一步简化：只扔 i 的素数倍数！因为此时所有的合数仍会被扔掉~
    for prime in primes:
```

```
        if prime*i > 10**4:
            break
        else:
            status[prime*i]='False'#把 i 的 p 倍扔掉
        # 欧拉筛的关键第二步简化：只扔 i 的“小于等于 i 的最小素因数”的素数倍数！
        # 因为这样会确保每个合数只被扔掉一次！
        if i % prime == 0:
            break

# 接着，只保留个位为 1 的素数
good = []
for p in primes:
    if p % 10 == 1:
        good.append(p)

# 然后，建立一个函数用于处理每个案例
# 由于 good 列表的长度为 306，因此用 bisect 方法不会 TLE（是 o(lnn)）
from bisect import bisect_left
def find(a):
    idx = bisect_left(good, a)
    result.append(good[:idx])

# 最后，处理输入、并输出即可
result = []
n = int(input())
for i in range(n):
    a = int(input())
    find(a)
for i in range(n):
    print(f"Case{i + 1}:")
    if len(result[i]) == 0:
        print("NULL")
    else:
        print(" ".join(map(str,result[i])))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

M28127:北大夺冠

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

思路：

代码：

```python

```

代码运行截图 ==（AC 代码截图，至少包含有"Accepted"）==

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>

这次考试因为时间和我的专业课冲突了，所以我就没有去参加线下的考试，是在线上进行的；我大概花了一个半小时，AC 了四道题；这个成绩对我个人来讲，并不算满意。其实这次考试给我的感觉是题目都偏简单，但是第 3 题因为我对树以及数据结构那一块儿掌握的不是很好，所以在考试时间内并没有做出来；而第 6 题的话，则是因为时间不够了，我就没有去做了。

对我做出来的四道题而言：第一题的话我觉得很常规，直接利用循环链表的基本性质就可以了。第二题的话，我因为之前做过好几道类似思路的题目，所以当时一看到题目就条件反射的想到了二分查找的思路，也是比较快的做出来了。第四题也是一个经典的使用双指针方法做的题目：每次判断的时候，直接根据当下值与目标值的大小关系来移动左指针和右指针就可以了，可以说是一道比较简单的题。对于第五题的话，我的思路很朴素，直接先用欧拉筛找出 1~10001 内的素数，并只保留个位数为 1 的素数；接着，利用 bisect 函数处理每个案例即可。我一开始做这道题目的时候并没有想到这道题目会这么快就能做出来，我还以为需要再改进一下算法才能 AC，这个方法估计肯定会 TLE；结果没想到第一次提交就直接 AC 了。后来我仔细一想，因为总共只有 306 个符合题意的素数，而 bisect 的时间复杂度是 o(lnn)，因此使用 bisect 方法并不会 TLE，所以确实能够用这个方法 AC。。可以说这道题的难度远比我想象中要低。

总的来讲，这次考试的结果我并不是特别满意，后续还是需要多精进一下自己对树、图还有数据结构那一块的理解；另外，还得多做点练习题，才能够取得真正的进步。