Assignment #9: Mock Exam立冬前一天

Updated 1658 GMT+8 Nov 6, 2025

2025 fall, Complied by **韩旭 元培学院**

> **说明：**
>
> 1. Nov月考： **AC3** 。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本人头像，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

E29982:一种等价类划分问题

hashing, http://cs101.openjudge.cn/practice/29982

用时：20min

思路

这道题首先写一个`digit_sum()`函数来计算某数字的各位和。这个函数也可以像下面这样通过递归完成。

```
def digit_sum(num, record=0):
    if num//10==0:
        return record+num
    else:
        return digit_sum(num//10,record+num%10)
```

此后，我们遍历所有范围内的数字，计算其各位和，然后利用key list和对应的字典将这些数字按各位和归类。最后将key list排序并输出即可。

代码

```
def digit_sum(num):
    a=0
    for j in range(4,-1,-1):
        x=num//(10**j)
        a+=x
        num-=x*(10**j)
    return a

m,n,k=map(int,input().split(sep=","))
values=[]
ans={}
for i in range(m+1,n):
    s=digit_sum(i)
    if s%k==0:
        if s//k in values:
            ans[s//k].append(i)
        else:
            values.append(s//k)
            ans[s//k]=[i]
values.sort()
for v in values:
    print(",".join(map(str,ans[v])))
```

代码运行截图

#50726064提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def digit_sum(num):
    a=0
    for j in range(4,-1,-1):
        x=num//(10**j)
        a+=x
        num-=x*(10**j)
    return a

m,n,k=map(int,input().split(sep=","))
values=[]
ans={}
for i in range(m+1,n):
    s=digit_sum(i)
    if s%k==0:
        if s//k in values:
            ans[s//k].append(i)
        else:
            values.append(s//k)
            ans[s//k]=[i]
values.sort()
for v in values:
    print(",".join(map(str,ans[v])))
```

基本信息

#:	50726064
题目:	E29982
提交人:	22n2200017737
内存:	3664kB
时间:	25ms
语言:	Python3
提交时间:	2025-11-06 15:30:24

E30086:dance

greedy, http://cs101.openjudge.cn/practice/30086

用时：20min

思路

这道题首先按照学生的身高排序，然后遍历各个学生，只需要考虑其下一个学生是否能与她配对即可。如果下一个学生都不能配对，那么所有学生就都无法配对，整体配对也就失败了。如果可以配对，就将`i`和`i+1`记录下来，然后继续从`i+2`开始遍历即可。我在月考的时候写的循环部分可以优化如下：

```
def judge(students):
    for i in range(N):
        if students[2*i+1] - students[2*i] > D:
            print("No")
            return
    print("Yes")
judge(students)
```

代码

```
N,D=map(int,input().split())
students=list(map(int,input().split()))
students.sort()
#print(students)
ans=True
record=[]
for i in range(2*N-1):
    if i not in record:
        if students[i+1]-students[i]>D:
            ans=False
        else:
            record.append(i)
            record.append(i+1)
if not ans:
    print("No")
else:
    print("Yes")
```

代码运行截图

#50726554提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
N,D=map(int,input().split())
students=list(map(int,input().split()))
students.sort()
#print(students)
ans=True
record=[]
for i in range(2*N-1):
    if i not in record:
        if students[i+1]-students[i]>D:
            ans=False
        else:
            record.append(i)
            record.append(i+1)
if not ans:
    print("No")
else:
    print("Yes")
```

基本信息

#: 50726554
题目: E30086
提交人: 22n2200017737
内存: 3624kB
时间: 21ms
语言: Python3
提交时间: 2025-11-06 15:52:17

English 帮助 关于

M25570: 洋葱

matrices, http://cs101.openjudge.cn/practice/25570

用时：15min

思路

本题的核心是如何判断哪一个元素属于哪一层。我使用了如下方法：即把矩阵坐标平移到以原点为中心，然后判断x和y的绝对值的最大值，并按照n的奇偶性进行简单的分类讨论。然后把各层的数字相加即可。

另一种判断层数的方法是`max(i,j,n-1-i,n-1-j)`（这里用min或者max都可以），这同样利用了层数定义的对称性。

代码

```
import math
n=int(input())
onion=[]
for _ in range(n):
    onion.append(list(map(int,input().split())))
layer=0
if n%2==0:
    layer=int(n/2)
else:
    layer=int((n+1)/2)
ans={i+1:0 for i in range(layer)}
for i in range(n):
```

```
    for j in range(n):
        x=i-(n-1)/2
        y=j-(n-1)/2
        l=math.ceil(max(abs(x),abs(y)))
        if n%2:
            l+=1
        ans[l]+=onion[i][j]
final=0
for i in range(layer):
    final=max(final,ans[i+1])
print(final)
```

代码运行截图

#50726897提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import math
n=int(input())
onion=[]
for _ in range(n):
    onion.append(list(map(int,input().split())))
layer=0
if n%2==0:
    layer=int(n/2)
else:
    layer=int((n+1)/2)
ans={i+1:0 for i in range(layer)}
for i in range(n):
    for j in range(n):
        x=i-(n-1)/2
        y=j-(n-1)/2
        l=math.ceil(max(abs(x),abs(y)))
        if n%2:
            l+=1
        ans[l]+=onion[i][j]
final=0
for i in range(layer):
    final=max(final,ans[i+1])
print(final)
```

基本信息

#: 50726897
题目: M25570
提交人: 22n2200017737
内存: 3992kB
时间: 29ms
语言: Python3
提交时间: 2025-11-06 16:07:48

English 帮助 关于

M28906:数的划分

dfs, dp, http://cs101.openjudge.cn/practice/28906

用时：1h

思路

我这里采用的是dp的方法，这个方法更为简练。状态转移方程的发现非常关键，假设`dp[n][k]`表示将n划分成k份的分法数目，那么有如下两种分法，一种是最小份为1，方法等同于剩下的n-1划分成k-1份的分法数目；另一种是最小份大于1，那么分法数目等价于：把n-k划分成k份的分法。因为把n-k划分成k份的每个方法，给每一份加1，就可以得到一种把n划分成k份的方法，且最小份一定大于1，反之亦然，所以是充要的。总结成状态转移方程就是：`dp[n][k]=dp[n-1][k-1]+dp[n-k][k]`（当然还需要保证n-k为正数）。按照状态转移方程进行动态规划即可。

这道题使用dfs也可以。月考的时候我不太熟悉dp的思路，从而尝试了dfs的思路，但没有实现有效的剪枝。具体来说，dfs的递归函数需要：（1）返回题目所求的分法数目，而不需要返回所有的分法；（2）在枚举的时候保证非降序，避免同一种分法被重复枚举。以下是AI给出的dfs写法。

```
from functools import lru_cache

@lru_cache(None)
def dfs(n, k, min_part):
    # 把 n 划成 k 份，每份 >= min_part，按非降序计数
    if k == 0:
        return 1 if n == 0 else 0
    if n < k * min_part:          # 最小都塞不满，剪枝
        return 0
    if k == 1:                    # 只剩一份，只能把剩下全给它
        return 1

    total = 0
    # 当前这一份取 x，后面保持非降序（下一层最小值仍为 x）
    for x in range(min_part, n // k + 1): # 右边界剪枝，不需要遍历到n
        total += dfs(n - x, k - 1, x)
    return total

n, k = map(int, input().split())
print(dfs(n, k, 1))
```

代码

```
N,K=map(int,input().split())
dp=[[0]*(K+1) for _ in range(N+1)]
dp[0][0]=1
for n in range(1,N+1):
    for k in range(1,K+1):
        dp[n][k]+=dp[n-1][k-1]
        if n-k>0:
            dp[n][k]+=dp[n-k][k]
print(dp[N][K])
```

代码运行截图

#50779906提交状态

查看 提交 统计 提问

状态：Accepted

源代码

```
N,K=map(int,input().split())
dp=[[0]*(K+1) for _ in range(N+1)]
dp[0][0]=1
for n in range(1,N+1):
    for k in range(1,K+1):
        dp[n][k]+=dp[n-1][k-1]
        if n-k>0:
            dp[n][k]+=dp[n-k][k]
print(dp[N][K])
```

基本信息

#: 50779906
题目: M28906
提交人: 22n2200017737
内存: 3932kB
时间: 24ms
语言: Python3
提交时间: 2025-11-10 16:01:59

English 帮助 关于

M29896:购物

greedy, http://cs101.openjudge.cn/practice/29896

用时：40min

思路

这道题依然是思路比较难想的贪心。关键点在于注意到它需要选择硬币使得**区间**$[1,X]$**上的整数完全被覆盖**，所以这是一个不断扩展连续覆盖区间长度的贪心。具体而言，我们维护可以覆盖的区间右端点`right`，表示我们目前的硬币可以覆盖$[1,right]$的区间。然后我们需要在所有的硬币面值中，选择尽量能让`right`变得更大的硬币，这就是贪心所在。这里有两个要求：（1）为了完整覆盖，新硬币的面值最多是`right+1`；（2）在此基础上，面值越大越好。根据这个条件，在`coins`的面值列表里排序筛选，直到右端点大于等于X即可。一旦所有面值都比`right+1`大，此题无解，因为无法完整覆盖某个区间。

通过这道题我理解了，贪心不一定是简单地从面值最大的硬币开始考虑（虽然很多贪心是这样的），而是要一步一步选取合适的硬币，使得我们的**目标**（这里是能连续覆盖的区间长度）最大。所以，不仅要考虑最大化，还要在能满足目标的基础上最大化。一开始选的硬币往往是比较小的，因为需要满足连续覆盖的目标；但后面这个约束就比较宽松了，就可以选择更大面值的硬币。如果一上来就选择大面值的硬币，区间长度是比较大了，但是连续覆盖不能满足，后面为了补充连续覆盖，就可能会选出能被小硬币平替的大硬币——这就是没有考虑到贪心和目标的一致性。

代码

```
X,N=map(int,input().split())
coins=list(map(int,input().split()))
coins.sort()
right=0
ans=0
while right<X:
    v=0
    for c in coins:
        if c<=right+1:
            v=c
```

```
        else:
            break
    if v==0:
        ans=-1
        break
    else:
        right+=v
        ans+=1
print(ans)
```

代码运行截图

#50787432提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
X,N=map(int,input().split())
coins=list(map(int,input().split()))
coins.sort()
right=0
ans=0
while right<X:
    v=0
    for c in coins:
        if c<=right+1:
            v=c
        else:
            break
    if v==0:
        ans=-1
        break
    else:
        right+=v
        ans+=1
print(ans)
```

基本信息

#:	50787432
题目:	29896
提交人:	22n2200017737
内存:	3608kB
时间:	21ms
语言:	Python3
提交时间:	2025-11-11 00:51:20

English 帮助 关于

T25353:排队

greedy, http://cs101.openjudge.cn/practice/25353

用时：20min（先听了答案）

思路：

这道题的贪心体现在：直接遍历身高数组，挑选出一定可以被交换顺序到开头的那些同学（因而他们也可以互相交换顺序），然后对他们进行字典序排序。然后继续如此操作，把剩下的同学进行交换和排序。

假设挑出来一组同学S后，因为剩下的同学被证明没法交换到开头，所以剩下的同学要么比它前面的最大值至少小$D+1$，要么比它前面的最小值至少大$D+1$。所以，如果想要把后面一个身高较小的同学x往前交换，一定会存在一个最开始排在x前面的同学y没法和他交换。假设$y \in S$，如果$h_y > h_x + D$，那么x就比S里面每一个元素都小，那么把x往前交换就会导致不能把S中身高较大的同学往后放；如果$h_y + D < h_x$，那么x比S中的同学都高，交换也

没有意义。假设y在被挑出的那组同学之外，那么x就更不可能交换到前面影响被挑出的那组同学了。总结来说，据字典序的排序特征，贪心是有效的：只需要优先保证能交换到开头的那部分是从小到大排序的。

具体实现上，我们需要在遍历过程中维护一个最大值和最小值，只要后面出现的身高和前面出现过的最大值、最小值的差都小于等于D，就保证可以和前面所有同学交换，因而就可以换到开头了。把这些选出来的同学加入一个`buffer`数组里面，并且在`check`数组里标记为True（此后就不会遍历到了），然后对`buffer`数组排序即可。

代码

```
N,D=map(int,input().split())
heights=[]
for _ in range(N):
    heights.append(int(input()))
check=[False]*N
ans=[]
while False in check:
    i=0
    buffer=[]
    maxh,minh=0,10**9
    while i<len(heights):
        if check[i]:
            i+=1
            continue
        maxh=max(maxh,heights[i])
        minh=min(minh,heights[i])
        if maxh-heights[i]<=D and heights[i]-minh<=D:
            buffer.append(heights[i])
            check[i]=True
        i+=1
    buffer.sort()
    ans+=buffer
for i in ans:
    print(i)
```

代码运行截图

#50792589提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
N,D=map(int,input().split())
heights=[]
for _ in range(N):
    heights.append(int(input()))
check=[False]*N
ans=[]
while False in check:
    i=0
    buffer=[]
    maxh,minh=0,10**9
    while i<len(heights):
        if check[i]:
            i+=1
            continue
        maxh=max(maxh,heights[i])
        minh=min(minh,heights[i])
        if maxh-heights[i]<=D and heights[i]-minh<=D:
            buffer.append(heights[i])
            check[i]=True
        i+=1
    buffer.sort()
    ans+=buffer
for i in ans:
    print(i)
```

基本信息

#:	50792589
题目:	25353
提交人:	22n2200017737
内存:	10500kB
时间:	391ms
语言:	Python3
提交时间:	2025-11-11 15:47:25

2. 学习总结和收获

这次月考前面三题整体还可以，花费的时间略有一点长（20+20+15，主要是dance那个题在错误解法上浪费了一点时间，可以多思考一下解法再写代码）。洋葱那个题类似旋转矩阵，重点是找到每个层的坐标有什么共同点（比较像八皇后怎么判断在同一个斜线上），这里就需要利用旋转不变性，要么是平移坐标，要么是考虑`x,y,n-1-x,n-1-y`四个的最大值或者最小值——这些量都是旋转不变的。

后三题涉及的算法比较难，需要在思路上再积累经验。数的划分主要是要积累经验，也就是说，如果有多种划分方式可能重复计数的情况，需要再加入一个排序的约束，来防止这种重复计数。另外，剪枝的具体策略也值得学习。购物和排队这两个题都属于比较难的贪心了，主要是目标不再是简单的最大化总量或者最小化消耗。所以，它并不等同于“选当前最大的”，而是要围绕**具体目标**（如区间连续覆盖、字典序最小）去设计可行的贪心策略。这里面要特别关注**约束**，也就是说贪心是在特定约束（比如说购物那个题，约束就是必须覆盖连续区间，所以不能一味选最大的硬币；排序那个题，约束就是交换的身高差限制，所以不是所有同学都能被挑出来）。