

# Assignment #D: Mock Exam下元节

Updated 1729 GMT+8 Dec 4, 2025

2025 fall, Compiled by 韩旭 元培学院

## 说明:

1. Dec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

### E29945:神秘数字的宇宙旅行

implementation, <http://cs101.openjudge.cn/practice/29945>

用时: 2min

思路: 按题目要求实现即可。

代码

```
while num>1:
    if num%2:
        print(f"{num}*3+1={num*3+1}")
        num=num*3+1
    else:
        print(f"{num}/2={int(num/2)}")
        num=int(num/2)
print("End")
```

状态: Accepted

源代码

```
num=int(input())
while num>1:
    if num%2:
        print(f"{num}*3+1={num*3+1}")
        num=num*3+1
    else:
        print(f"{num}/2={int(num/2)}")
        num=int(num/2)
print("End")
```

基本信息

#: 51131808  
题目: E29945  
提交人: 22n2200017737  
内存: 3628kB  
时间: 26ms  
语言: Python3  
提交时间: 2025-12-04 15:09:35

## E29946:删数问题

monotonic stack, greedy, <http://cs101.openjudge.cn/practice/29946>

用时: 40min

思路: 月考的时候没想通怎么用单调栈实现这个问题, 所以就写了一个贪心。基本思路是: 假设要剩下的数字个数是 `left`, 那么我们先保留倒数 `left-1` 个数字, 然后在前面所有数字中选取最小的, 这样既能保证至少满足删除数字个数的要求, 同时又能实现首位数字尽可能最小化; 然后继续保留倒数 `left-2` 个数字, 在上一轮选取的数字后面的数字中选择最小的, 以此类推, 直到选出 `left` 个数字。但是这个算法的时间复杂度可能略高。

单调栈可以用于选出某个数字之后第一个比它小(或大)的数字, 而“出栈”的过程恰好就对应了“删数”的过程。当我们找到某个数字之后第一个比它小的数后, 如果还有删数的余额, 我们就把它弹出, 然后把更小的那个数字入栈, 以此类推直到删数额度用完。这样一来, 由于栈内单调递增, 在删数额度内最小的数字总是排在最高位数, 且栈内的顺序和序列本身的顺序也是一致的。这就实现了删数额度内剩余数字最小的目标。

这道题还有一个常见的坑就是前导0的问题, 需要把输出结果用 `int()` 函数转换一下。这个在字符串或一位数字拼成更大数字的时候经常遇到, 需要注意。

代码

(i) 贪心

```

n=list(map(int,list(input()))
k=int(input())
left=len(n)-k
ans=[]
p=-1
for i in range(left):
    tmp=[]
    for j in range(p+1,len(n)-(left-i-1)):
        tmp.append((j,n[j]))
        tmp.sort(key=lambda x:x[1])
        ans.append(tmp[0][1])
        p=tmp[0][0]
print(int("".join(map(str,ans))))

```

(ii) 单调栈

```

n=list(map(int,list(input()))
k=int(input())
stack=[]

for x in n:
    while k and stack and stack[-1]>x:
        stack.pop()
        k-=1
    stack.append(x)
while k:
    stack.pop()
    k-=1
print(int("".join(map(str,stack))))

```

代码运行截图

#51132682提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: **Accepted**

源代码

```

n=list(map(int,list(input()))
k=int(input())
left=len(n)-k
ans=[]
p=-1
for i in range(left):
    tmp=[]
    for j in range(p+1,len(n)-(left-i-1)):
        tmp.append((j,n[j]))
        tmp.sort(key=lambda x:x[1])
        ans.append(tmp[0][1])
        p=tmp[0][0]
print(int("".join(map(str,ans))))

```

基本信息

#: 51132682  
 题目: E29946  
 提交人: 22n2200017737  
 内存: 3644kB  
 时间: 25ms  
 语言: Python3  
 提交时间: 2025-12-04 15:49:35

状态: Accepted

源代码

```
n=list(map(int,list(input())))\nk=int(input())\nstack=[]\n\nfor x in n:\n    while k and stack and stack[-1]>x:\n        stack.pop()\n        k-=1\n    stack.append(x)\nwhile k:\n    stack.pop()\n    k-=1\nprint(int("".join(map(str,stack))))
```

基本信息

#: 51194354  
题目: 29946  
提交人: 22n2200017737  
内存: 3624kB  
时间: 27ms  
语言: Python3  
提交时间: 2025-12-08 16:59:35

## E30091:缺德的图书馆管理员

greedy, <http://cs101.openjudge.cn/practice/30091>

用时: 25min

思路: 这道题只需要意识到非常关键的一点, 就是我们并不在意具体每个同学的身份, 而只在意最后一个离开的同学, 所以当两个同学相遇转身的时候, 我们等价地理解为他们互相“穿过去”了就可以了。所以, 每个同学离开有向左和向右两种可能性。总体最短时间就是每个同学离开的最短时间的最大值; 最长时间就是每个同学离开的最长时间的最大值。

代码

```
L=int(input())\nN=int(input())\nx=list(map(int,input().split()))\nx.sort()\nmin_time=0\nmax_time=0\nfor x in x:\n    min_time=max(min_time,min(x,L+1-x))\n    max_time=max(max_time,max(x,L+1-x))\nprint(min_time,max_time)
```

代码运行截图

状态: **Accepted**

源代码

```
L=int(input())
N=int(input())
X=list(map(int,input().split()))
X.sort()
min_time=0
max_time=0
for x in X:
    min_time=max(min_time,min(x,L+1-x))
    max_time=max(max_time,max(x,L+1-x))
print(min_time,max_time)
```

基本信息

#: 51133306  
题目: E30091  
提交人: 22n2200017737  
内存: 4020kB  
时间: 27ms  
语言: Python3  
提交时间: 2025-12-04 16:16:24

## M27371:Playfair密码

simulation, string, matrix, <http://cs101.openjudge.cn/practice/27371>

用时: 1h

思路：按照题目要求一步步实现即可。这道题在考场上没有AC，原因是第一步把j替换成i然后去重这两个步骤写在了一个循环里面，这时候如果有两个j就无法实现有效去重了。一个更好的写法如下：

```
key_new=[]
record=set() #注意空集合必须要用set(), {}默认是空字典
for i in range(len(key)):
    if key[i]=='j':
        key[i]='i'
    if key[i] not in record:
        record.add(key[i])
        key_new.append(key[i])
```

代码

```
key=list(input())
for i in range(len(key)):
    if key[i]=="j":
        key[i]="i"
for i in range(len(key)-1,-1,-1):
    tmp=key[:i]+key[i+1:]
    if key[i] in tmp:
        del key[i]

alphabet={}
for i in range(26):
    alphabet[chr(ord("a")+i)]=1
```

```

alphabet['j']=0
matrix=[[""]*5 for _ in range(5)]

count1=0
count2=0
for i in range(5):
    for j in range(5):
        if count1<len(key):
            matrix[i][j]=key[count1]
            alphabet[key[count1]]=0
            count1+=1
        else:
            while alphabet[chr(ord("a")+count2)]!=1:
                count2+=1
            matrix[i][j]=chr(ord("a")+count2)
            count2+=1
D={}
for i in range(5):
    for j in range(5):
        D[matrix[i][j]]=(i,j)

n=int(input())
for _ in range(n):
    word=list(input())
    for i in range(len(word)):
        if word[i]=="j":
            word[i]="i"
    word_split=[]
    i=0
    while i<len(word):
        if i==len(word)-1:
            if word[i]=="x":
                word_split.append(word[i]+"q")
            else:
                word_split.append(word[i]+"x")
            i+=1
            continue
        if word[i+1]==word[i]:
            if word[i]=="x":
                word_split.append(word[i]+"q")
            else:
                word_split.append(word[i]+"x")
            i+=1
        else:
            word_split.append(word[i] + word[i+1])
            i+=2
    ans=""
    def new(x):
        if x>=4:
            return 0
        else:
            return x+1

```

```

for tup in word_split:
    row1,col1=D[tup[0]]
    row2,col2=D[tup[1]]
    if row1==row2:
        new1 = matrix[row1][new(col1)]
        new2 = matrix[row2][new(col2)]
        ans += new1+new2
    elif col1==col2:
        new1 = matrix[new(row1)][col1]
        new2 = matrix[new(row2)][col2]
        ans += new1+new2
    else:
        new1 = matrix[row1][col2]
        new2 = matrix[row2][col1]
        ans += new1+new2
print(ans)

```

代码运行截图

#51135552提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: **Accepted**

源代码

```

key=list(input())
for i in range(len(key)):
    if key[i]=="j":
        key[i]="i"
for i in range(len(key)-1,-1,-1):
    tmp=key[:i]+key[i+1:]
    if key[i] in tmp:
        del key[i]

alphabet={}
for i in range(26):
    alphabet[chr(ord("a")+i)]=1
alphabet['j']=0
matrix=[[""]*5 for _ in range(5)]

count1=0
count2=0
for i in range(5):
    for j in range(5):
        if count1<len(key):
            matrix[i][j]=key[count1]
            alphabet[key[count1]]=0
            count1+=1
        else:
            while alphabet[chr(ord("a")+count2)]!=1:
                count2+=1
            matrix[i][j]=chr(ord("a")+count2)
            count2+=1

D={}
for i in range(5):
    for j in range(5):
        D[matrix[i][j]]=(i,j)

n=int(input())
for _ in range(n):
    word=list(input())

```

基本信息

#: 51135552  
 题目: 27371  
 提交人: 22n2200017737  
 内存: 3848kB  
 时间: 29ms  
 语言: Python3  
 提交时间: 2025-12-04 17:47:34

## T30201:旅行售货商问题

dp,dfs, <http://cs101.openjudge.cn/practice/30201>

用时: 50min

思路: 这道题首先需要意识到, 当我们实现一个从城市*i*开始经过每个城市再回到城市*i*的路径的时候, 其总花费是轮换对称的, 无论从哪个城市出发都可以, 所以我们不妨设从城市0出发。有了这个基础, 不难想到把较短的路径作为子问题, 把最后一个到达的城市作为状态变量。然而, 题目又要求每个城市都到达且仅到达一次, 这就不仅需要记录最后一个到达的城市, 而且要记录该路径所经过的所有城市, 这就需要使用**状态压缩DP**来实现: 即把*n*个城市每个城市有没有经过的0-1串压缩为一个二进制数, 然后通过位运算对这个数进行操作和判断。

- 如果要判断某个数位*i* (即城市*i*) 是否为1 (即是否经过), 就使用一个仅在第*i*位为1的二进制数 ( $1 \ll i$ ) 进行按位与(&);
- 把某个数位*i*改为1, 就使用 $1 \ll i$ 进行按位或(|);
- 把某个数位*i*改为0, 就使用 $\sim(1 \ll i)$ 进行按位与(&);
- 如果要反转某位的值 (比如把1变成0, 把0变成1), 可以使用 $1 \ll i$ 进行按位异或。

了解了这个技巧之后, 状态转移方程就比较容易了。dp[record][i] 的计算过程如下: 先算record中去掉*i* (使用异或) 之后剩下经过的城市, 然后遍历这些城市 (记作*j*), 考虑以*j*为终点的最少成本 dp[prev\_record][j], 再加上从*j*到*i*的成本, 取遍历过程中总成本的最小值即可。dp的初始状态为仅经过0且以0为终点, 即 dp[1][0]=0。dp更新的过程我们从小到大遍历record, 这样任何一个遍历到的record去掉某个是1的位之后一定是更小的二进制数, 也就被我们遍历过了。由于我们考虑的是从城市0出发的情况, 所以不经过0的路径我们全部设置为inf, 就不会参与到更新当中; 这样所有被更新的record就都是经过城市0的了。

最后需要让这个路径回到城市0, 所以我们遍历每个非0的城市, 考虑 dp[111...111][i]+matrix[i][0], 选取其中的最小值即可。

代码

```
n=int(input())
matrix=[list(map(int,input().split())) for _ in range(n)]
dp=[[float('inf')]*n for _ in range(1<<n)]
dp[1][0]=0

for record in range(1,1<<n):
    dp_record=dp[record]

    for last in range(n):
        if record&(1<<last)==0:
            continue
        prev_record=record^(1<<last)
        if prev_record==0:
            continue
        dp_prev=dp[prev_record]
```

```

        for i in range(n):
            if prev_record&(1<<i)==0:
                continue
            dp_record[last]=min(dp_record[last],dp_prev[i]+matrix[i][last])

ans=float('inf')
for i in range(1,n):
    ans=min(ans,dp[(1<<n)-1][i]+matrix[i][0])
print(ans)

```

代码运行截图

#51186641提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: **Accepted**

源代码

```

n=int(input())
matrix=[list(map(int,input().split())) for _ in range(n)]
dp=[[float('inf')]*n for _ in range(1<<n)]
dp[1][0]=0

for record in range(1,1<<n):
    dp_record=dp[record]

    for last in range(n):
        if record&(1<<last)==0:
            continue
        prev_record=record^(1<<last)
        if prev_record==0:
            continue
        dp_prev=dp[prev_record]

        for i in range(n):
            if prev_record&(1<<i)==0:
                continue
            dp_record[last]=min(dp_record[last],dp_prev[i]+matrix[i][last])

ans=float('inf')
for i in range(1,n):
    ans=min(ans,dp[(1<<n)-1][i]+matrix[i][0])
print(ans)

```

基本信息

#: 51186641  
 题目: 30201  
 提交人: 22n2200017737  
 内存: 99208kB  
 时间: 897ms  
 语言: PyPy3  
 提交时间: 2025-12-07 23:23:18

©2002-2022 POJ 京ICP备20010980号-1

[English](#) [帮助](#) [关于](#)

## T30204:小P的LLM推理加速

greedy, <http://cs101.openjudge.cn/practice/30204>

用时: 40min

思路：由于核的特点，想要开启  $y$  类能耗，则必须要先经过  $x$  类能耗；若想再次使用  $x$  类能耗，则也必须经过  $y$  类能耗。所以对于每个核，都只有两类选项，即“使用若干次  $x+y$ ”或者“使用一次  $x$  后再使用若干次  $y+x$ ”。 $y$  不能单独拿出来被使用。因此，我们只需要选择  $x+y$  最小的核使用若干次，以及  $x$  较小的核使用第一次。**注意这里单独使用较小的  $x$  的每周期平均能耗可能比最小的  $x+y$  要小**，所以根据贪心的原则，应该从  $x$  最小的核开始先尝试“使用一次  $x$ ”，然后再使用若干次最小的  $x+y$ ；或者尝试最小和第二小的核均先“使用一次  $x$ ”，然后使用若干次最小的  $x+y$ ，以此类推。这就是说，把各个核的  $x$  数组排序，再选出前缀和，剩余的能耗均用于最小的  $x+y$ ，最后选出所有方式中周期数最大的一种即可。

代码

```
n,m=map(int,input().split())
core_x=[]
core_sum=[]
for i in range(n):
    x,y=map(int,input().split())
    core_x.append(x)
    core_sum.append(x+y)
core_sum.sort()
sum_min=core_sum[0]
core_x.sort()
pre_x=[]
tmp=0
for x in core_x:
    tmp+=x
    pre_x.append(tmp)

count=(m//sum_min)*2
for i in range(n):
    pre=pre_x[i]
    count_tmp=(i+1)+((m-pre)//sum_min)*2
    if count_tmp>count:
        count=count_tmp
print(count)
```

代码运行截图

状态: **Accepted**

源代码

```
n,m=map(int,input().split())
core_x=[]
core_sum=[]
for i in range(n):
    x,y=map(int,input().split())
    core_x.append(x)
    core_sum.append(x+y)
core_sum.sort()
sum_min=core_sum[0]
core_x.sort()
pre_x=[]
tmp=0
for x in core_x:
    tmp+=x
    pre_x.append(tmp)

count=(m//sum_min)*2
for i in range(n):
    pre=pre_x[i]
    count_tmp=(i+1)+((m-pre)//sum_min)*2
    if count_tmp>count:
        count=count_tmp
print(count)
```

基本信息

#: 51199949  
题目: 30204  
提交人: 22n2200017737  
内存: 19388kB  
时间: 324ms  
语言: Python3  
提交时间: 2025-12-08 22:23:41

## 2. 学习总结和收获

这次月考 AC3，整体完成度还可以，但没来得及看后面两个比较难的题，而且第四题因为一个细节性的逻辑错误遗憾 WA，所以这种implementation的题还需要更加冷静细致。LLM 那个题的基本贪心思路倒是很快想到了，但是走了另一个误区，把预算先投在若干次  $x + y$  上再去扣除若干个  $x$ ，但实际应该先考虑若干个最小的  $x$ ，因为这些单步平均每个周期的能耗有时会更小，剩余预算再统一投入到最便宜的  $x + y$  上。这个抽象过程其实仍然是对“平均每个周期的能耗”的贪心。

此外还有一些新的收获：删数问题让我对单调栈的核心思想更加深化，即保持一个局部最优的递增结构，本质上就等于实时完成“删除不必要元素”的动作；而旅行商那题第一次让我比较系统地跑通了完整的状压 DP（二进制数表示子集、位运算实现判断），其核心就是大大提高了状态表示的潜力，此前只能把状态用几个值来表示，但是现在通过二进制数的压缩，可以把很丰富的history都压缩进一个状态变量里面，非常巧妙。另外还有一些特定的细节问题需要注意，比如前导0的处理、旅行商问题轮换对称性的发现、用集合来更快地记录和查找重复字符（bfs中也类似）等。