Assignment #6: 贪心和矩阵

Updated 1925 GMT+8 Oct 17, 2023

2023 fall, Complied by Hongfei Yan（请改为同学的姓名、院系）

说明：

1）请把每个题目解题思路（可选），源码Python, 或者C++/C（已经在Codeforces/Openjudge上AC），截图（包含Accepted, 学号），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。

3）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、作业评论有md或者doc。

4）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. 必做题目

508A. Pasha and Pixels

brute force, 1100, http://codeforces.com/problemset/problem/508/A

Pasha loves his phone and also putting his hair up... But the hair is now irrelevant.

Pasha has installed a new game to his phone. The goal of the game is following. There is a rectangular field consisting of n row with m pixels in each row. Initially, all the pixels are colored white. In one move, Pasha can choose any pixel and color it black. In particular, he can choose the pixel that is already black, then after the boy's move the pixel does not change, that is, it remains black. Pasha loses the game when a 2×2 square consisting of black pixels is formed.

Pasha has made a plan of k moves, according to which he will paint pixels. Each turn in his plan is represented as a pair of numbers i and j, denoting respectively the row and the column of the pixel to be colored on the current move.

Determine whether Pasha loses if he acts in accordance with his plan, and if he does, on what move the 2×2 square consisting of black pixels is formed.

Input

The first line of the input contains three integers n, m, k ($1 \le n, m \le 1000$, $1 \le k \le 10^5$) — the number of rows, the number of columns and the number of moves that Pasha is going to perform.

The next k lines contain Pasha's moves in the order he makes them. Each line contains two integers i and j ($1 \le i \le n$, $1 \le j \le m$), representing the row number and column number of the pixel that was painted during a move.

Output

If Pasha loses, print the number of the move when the 2×2 square consisting of black pixels is formed.

If Pasha doesn't lose, that is, no 2×2 square consisting of black pixels is formed during the given k moves, print 0.

Examples

input

```
2 2 4
1 1
1 2
2 1
2 2
```

output

```
4
```

input

```
2 3 6
2 3
2 2
1 3
2 2
1 2
1 1
```

output

```
5
```

input

```
5 3 7
2 3
1 2
1 1
4 1
3 1
5 3
3 2
```

output

```
0
```

思路：

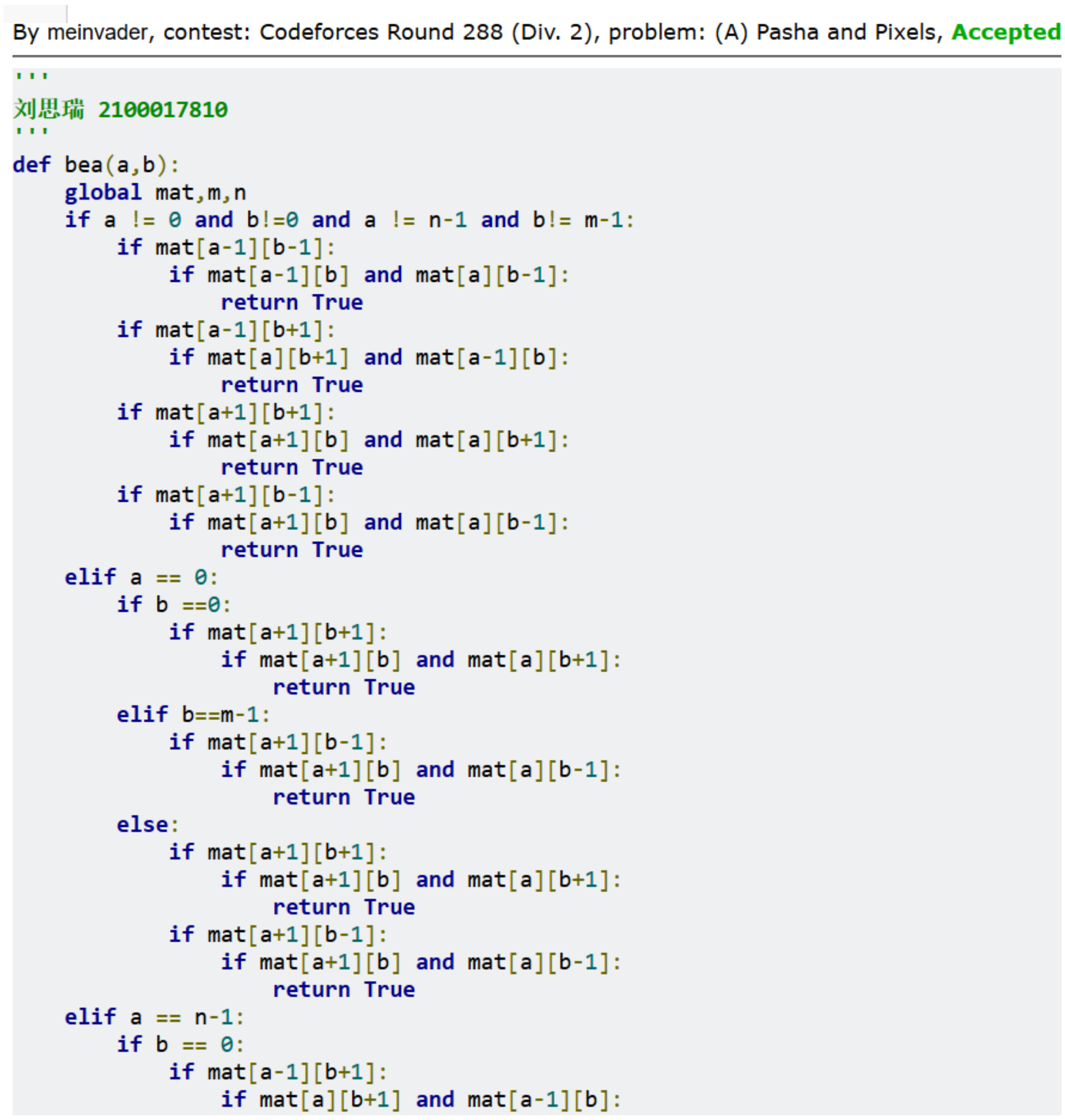
每次判断附近是否构成2x2矩阵即可，但是代码好像写的有点太复杂了。。

还有就是列表生成器会有浅拷贝的问题，要避免这个bug

代码

```python
'''
刘思瑞 2100017810
'''
def bea(a,b):
    global mat,m,n
    if a != 0 and b!=0 and a != n-1 and b!= m-1:
        if mat[a-1][b-1]:
            if mat[a-1][b] and mat[a][b-1]:
                return True
        if mat[a-1][b+1]:
            if mat[a][b+1] and mat[a-1][b]:
                return True
        if mat[a+1][b+1]:
            if mat[a+1][b] and mat[a][b+1]:
                return True
        if mat[a+1][b-1]:
            if mat[a+1][b] and mat[a][b-1]:
                return True
    elif a == 0:
        if b ==0:
            if mat[a+1][b+1]:
                if mat[a+1][b] and mat[a][b+1]:
                    return True
        elif b==m-1:
            if mat[a+1][b-1]:
                if mat[a+1][b] and mat[a][b-1]:
                    return True
        else:
            if mat[a+1][b+1]:
                if mat[a+1][b] and mat[a][b+1]:
                    return True
            if mat[a+1][b-1]:
                if mat[a+1][b] and mat[a][b-1]:
                    return True
    elif a == n-1:
        if b == 0:
            if mat[a-1][b+1]:
                if mat[a][b+1] and mat[a-1][b]:
                    return True
        elif b == m-1:
            if mat[a-1][b-1]:
                if mat[a-1][b] and mat[a][b-1]:
                    return True
        else:
            if mat[a-1][b-1]:
                if mat[a-1][b] and mat[a][b-1]:
                    return True
            if mat[a-1][b+1]:
                if mat[a][b+1] and mat[a-1][b]:
                    return True
    elif b == 0:
        if mat[a-1][b+1]:
            if mat[a][b+1] and mat[a-1][b]:
                return True
        if mat[a+1][b+1]:
            if mat[a+1][b] and mat[a][b+1]:
                return True
    elif b == m-1:
        if mat[a-1][b-1]:
            if mat[a-1][b] and mat[a][b-1]:
                return True
        if mat[a+1][b-1]:
            if mat[a+1][b] and mat[a][b-1]:
                return True
    return False


def sign(a,b):
    global mat
    mat[a][b] = True
    return bea(a,b)

j = 0
n , m , k = map(int,input().split())
mat = []
for i in range(n):
    mat.append([False]*m)
if m != 1 and n !=1 :
    for i in range(k):
        a , b = map(int,input().split())
        if sign(a-1,b-1):
            j = i+1
            break
print(j)
```

代码运行截图



By meinvader, contest: Codeforces Round 288 (Div. 2), problem: (A) Pasha and Pixels, Accepted

```python
'''
刘思瑞 2100017810
'''
def bea(a,b):
    global mat,m,n
    if a != 0 and b!=0 and a != n-1 and b!= m-1:
        if mat[a-1][b-1]:
            if mat[a-1][b] and mat[a][b-1]:
                return True
        if mat[a-1][b+1]:
            if mat[a][b+1] and mat[a-1][b]:
                return True
        if mat[a+1][b+1]:
            if mat[a+1][b] and mat[a][b+1]:
                return True
        if mat[a+1][b-1]:
            if mat[a+1][b] and mat[a][b-1]:
                return True
    elif a == 0:
        if b ==0:
            if mat[a+1][b+1]:
                if mat[a+1][b] and mat[a][b+1]:
                    return True
        elif b==m-1:
            if mat[a+1][b-1]:
                if mat[a+1][b] and mat[a][b-1]:
                    return True
        else:
            if mat[a+1][b+1]:
                if mat[a+1][b] and mat[a][b+1]:
                    return True
            if mat[a+1][b-1]:
                if mat[a+1][b] and mat[a][b-1]:
                    return True
    elif a == n-1:
        if b == 0:
            if mat[a-1][b+1]:
                if mat[a][b+1] and mat[a-1][b]:
```

23555: 节省存储的矩阵乘法

implementation, matrices, http://cs101.openjudge.cn/practice/23555/

由于矩阵存储非常耗费空间，一个长度n宽度m的矩阵需要花费n*m的存储，因此我们选择用另一种节省空间的方法表示矩阵。一个矩阵X可以表示为三元组的序列，每个三元组代表（行号，列号，元素值），如果元素值是0则我们不存储这个三元组，这样对于0很多的大型矩阵，我们节省了很多存储空间。现在我们有两个用这种方式表示的矩阵X和Y，我们想要计算这两个矩阵的乘积，并且也用三元组形式表达，该如何完成呢。

如果不知道矩阵如何相乘，可以参考：http://cs101.openjudge.cn/practice/18161

输入

输入第一行是三个整数n，m1, m2，两个矩阵X，Y的维度都是n*n，m1是矩阵X中的非0元素数，m2是矩阵Y中的非0元素数。
之后是m1行，每行是一个三元组（行号，列号，元素值），代表X矩阵的元素值，注意行列编号都从0开始。

之后是m2行，每行是一个三元组（行号，列号，元素值），代表Y矩阵的元素值，注意行列编号都从0开始。

输出

输出是m3行，代表X和Y两个矩阵乘积中的非0元素的数目，按照先行号后列号的方式递增排序。
每行仍然是前述的三元组形式。

样例输入

```
Sample Input1:
3 3 2
0 0 1
1 0 -1
1 2 3
0 0 7
2 2 1

Sample Output1:
0 0 7
1 0 -7
1 2 3

解释:
A = [
  [ 1, 0, 0],
  [-1, 0, 3],
  [0, 0, 0]
]

B = [
  [ 7, 0, 0 ],
  [ 0, 0, 0 ],
  [ 0, 0, 1 ]
]

A*B = [
[7,0,0],
[-7,0,3],
[0,0,0]]
```

样例输出

```
Sample Input2:
2 2 4
0 0 1
1 1 1
0 0 2
0 1 3
1 0 4
1 1 5

Sample Output2:
0 0 2
0 1 3
```

```
1 0 4
1 1 5

解释：
A = [
[1,0],
[0,1]
]

B = [
[2,3],
[4,5]
]

A*B = [
[2,3],
[4,5]
]
```

提示：tags: implementation,matrices

来源：2021fall-cs101, hy

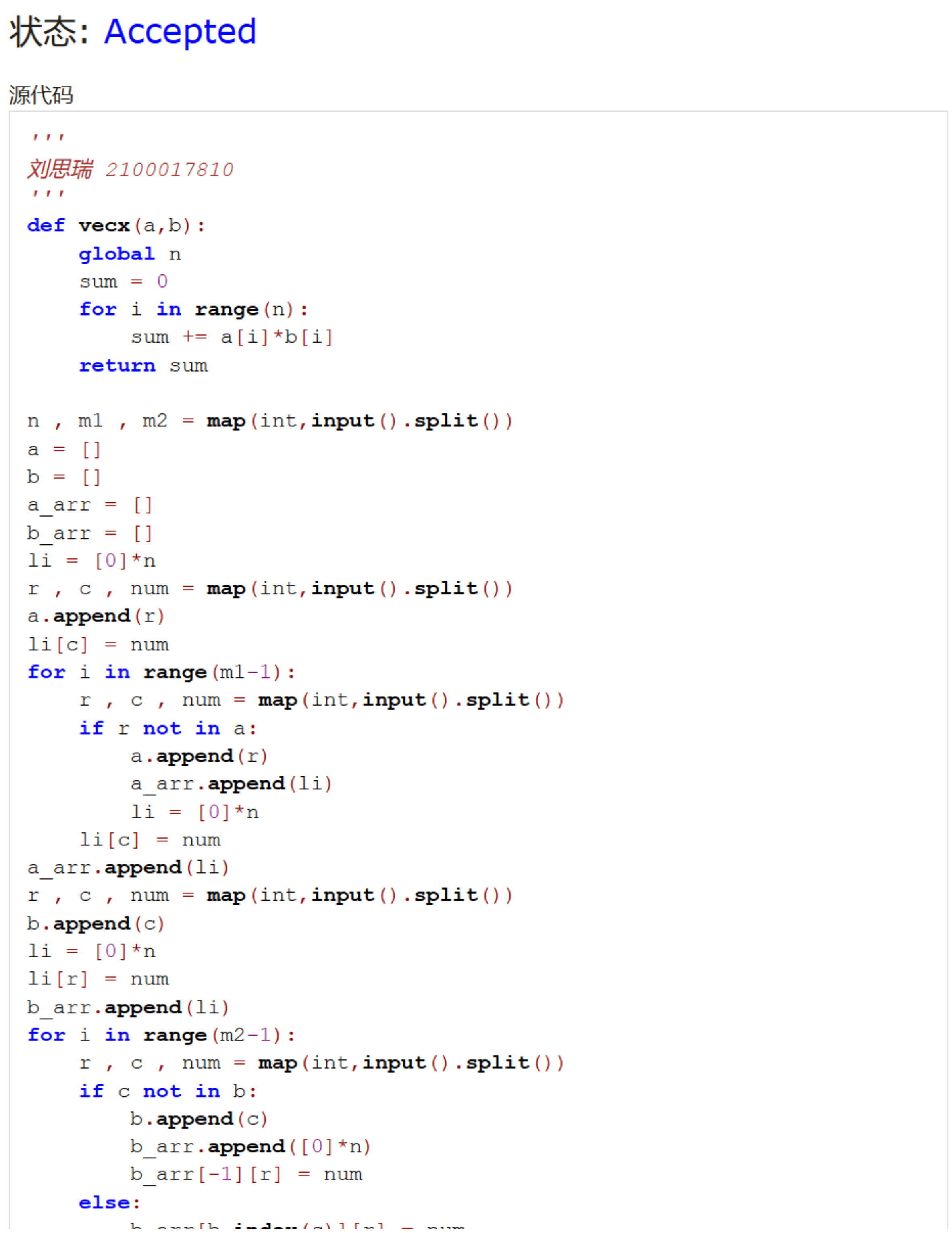
思路：

把x的非零行以及y的非零列记录下来，唯一的困难在于y是按行的顺序输入的，因此最后再进行一个排序即可

代码

```python
'''
刘思瑞 2100017810
'''
def vecx(a,b):
    global n
    sum = 0
    for i in range(n):
        sum += a[i]*b[i]
    return sum

n , m1 , m2 = map(int,input().split())
a = []
b = []
a_arr = []
b_arr = []
li = [0]*n
r , c , num = map(int,input().split())
a.append(r)
li[c] = num
for i in range(m1-1):
    r , c , num = map(int,input().split())
    if r not in a:
        a.append(r)
        a_arr.append(li)
        li = [0]*n
    li[c] = num
a_arr.append(li)
r , c , num = map(int,input().split())
b.append(c)
li = [0]*n
li[r] = num
b_arr.append(li)
for i in range(m2-1):
    r , c , num = map(int,input().split())
    if c not in b:
        b.append(c)
        b_arr.append([0]*n)
        b_arr[-1][r] = num
    else:
        b_arr[b.index(c)][r] = num
b_arr.append(li)
for i in range(len(b)-1):
    for j in range(len(b)-1-i):
        if b[j] > b[j+1]:
            b[j] , b[j+1] , b_arr[j] , b_arr[j+1] = b[j+1] , b[j] , b_arr[j+1] , b_arr[j]
for i in a:
    for j in b:
        m = vecx(a_arr[a.index(i)],b_arr[b.index(j)])
        if m:
            print(i,j,m)
```

代码运行截图


状态: Accepted

源代码

```
'''
刘思瑞 2100017810
'''
def vecx(a,b):
    global n
    sum = 0
    for i in range(n):
        sum += a[i]*b[i]
    return sum

n , m1 , m2 = map(int,input().split())
a = []
b = []
a_arr = []
b_arr = []
li = [0]*n
r , c , num = map(int,input().split())
a.append(r)
li[c] = num
for i in range(m1-1):
    r , c , num = map(int,input().split())
    if r not in a:
        a.append(r)
        a_arr.append(li)
        li = [0]*n
    li[c] = num
a_arr.append(li)
r , c , num = map(int,input().split())
b.append(c)
li = [0]*n
li[r] = num
b_arr.append(li)
for i in range(m2-1):
    r , c , num = map(int,input().split())
    if c not in b:
        b.append(c)
        b_arr.append([0]*n)
        b_arr[-1][r] = num
    else:
        b_arr[b.index(c)][r] = num
```

12560: 生存游戏

matrices, http://cs101.openjudge.cn/practice/12560/

有如下生存游戏的规则：

给定一个n*m(1<=n,m<=100)的数组，每个元素代表一个细胞，其初始状态为活着(1)或死去(0)。

每个细胞会与其相邻的8个邻居（除数组边缘的细胞）进行交互，并遵守如下规则：

任何一个活着的细胞如果只有小于2个活着的邻居，那它就会由于人口稀少死去。

任何一个活着的细胞如果有2个或者3个活着的邻居，就可以继续活下去。

任何一个活着的细胞如果有超过3个活着的邻居，那它就会由于人口拥挤而死去。

任何一个死去的细胞如果有恰好3个活着的邻居，那它就会由于繁殖而重新变成活着的状态。

请写一个函数用来计算所给定初始状态的细胞经过一次更新后的状态是什么。

注意：所有细胞的状态必须同时更新，不能使用更新后的状态作为其他细胞的邻居状态来进行计算。

输入

第一行为n和m，而后n行，每行m个元素，用空格隔开。

输出

n行，每行m个元素，用空格隔开。

样例输入

```
3 4
0 0 1 1
1 1 0 0
1 1 0 1
```

样例输出

```
0 1 1 0
1 0 0 1
1 1 1 0
```

来源：cs10116 final exam

思路：

直接判断

代码

```python
'''
刘思瑞 2100017810
'''
def survive(n_,m_):
    global a,n,m
    sum = 0
    if n_ > 0:
        sum += a[n_-1][m_]
        if m_ >0:
            sum += a[n_-1][m_-1]
        try:
            sum += a[n_-1][m_+1]
        except:
            IndexError
    if m_ > 0:
        sum += a[n_][m_-1]
        try:
            sum += a[n_+1][m_-1]
        except:
            IndexError
    try:
        sum += a[n_+1][m_]
    except:
        IndexError
    try:
        sum += a[n_+1][m_+1]
    except:
        IndexError
    try:
        sum += a[n_][m_+1]
    except:
        IndexError
    return sum

a = []
n , m = map(int,input().split())
for i in range(n):
    a.append(list(map(int,input().split())))
for i in range(n):
    for j in range(m):
        s = survive(i,j)
        out = 0
        if a[i][j] == 0:
            if s == 3:
                out = 1
        else:
            if s ==2 or s == 3:
                out = 1
        print(out,end=' ')
    print('')
```

代码运行截图

状态: Accepted

源代码

```python
'''
刘思瑞 2100017810
'''
def survive(n_,m_):
    global a,n,m
    sum = 0
    if n_ > 0:
        sum += a[n_-1][m_]
        if m_ >0:
            sum += a[n_-1][m_-1]
        try:
            sum += a[n_-1][m_+1]
        except:
            IndexError
    if m_ > 0:
        sum += a[n_][m_-1]
        try:
            sum += a[n_+1][m_-1]
        except:
            IndexError
    try:
        sum += a[n_+1][m_]
    except:
        IndexError
    try:
        sum += a[n_+1][m_+1]
    except:
        IndexError
    try:
        sum += a[n_][m_+1]
    except:
        IndexError
    return sum

a = []
n , m = map(int,input().split())
for i in range(n):
    a.append(list(map(int,input().split())))
```

04110: 圣诞老人的礼物-Santa Clau's Gifts

greedy/dp, http://cs101.openjudge.cn/practice/04110

圣诞节来临了，在城市A中圣诞老人准备分发糖果，现在有多箱不同的糖果，每箱糖果有自己的价值和重量，每箱糖果都可以拆分成任意散装组合带走。圣诞老人的驯鹿最多只能承受一定重量的糖果，请问圣诞老人最多能带走多大价值的糖果。

输入

第一行由两个部分组成，分别为糖果箱数正整数n(1 <= n <= 100)，驯鹿能承受的最大重量正整数w（0 < w < 10000），两个数用空格隔开。其余n行每行对应一箱糖果，由两部分组成，分别为一箱糖果的价值正整数v和重量正整数w，中间用空格隔开。

输出

输出圣诞老人能带走的糖果的最大总价值，保留1位小数。输出为一行，以换行符结束。

样例输入

```
4 15
100 4
412 8
266 7
591 2
```

样例输出

```
1193.0
```

思路：

先按性价比排序再贪心

代码

```python
'''
刘思瑞 2100017810
'''
n , w0 = map(int,input().split())
record = []
for i in range(n):
    v , w = map(int,input().split())
    record.append([v/w,v,w])
for i in range(n-1):
    for j in range(n-1-i):
        if record[j][0] < record[j+1][0]:
            record[j] , record[j+1] = record[j+1] , record[j]
sum = 0
for i in record:
    if i[2] < w0:
        sum+= i[1]
        w0 -= i[2]
    else:
        sum+= i[0] * w0
        break
print('%.1f' % sum)
```

代码运行截图

状态: Accepted

源代码

```
'''
刘思瑞 2100017810
'''
n , w0 = map(int,input().split())
record = []
for i in range(n):
    v , w = map(int,input().split())
    record.append([v/w,v,w])
for i in range(n-1):
    for j in range(n-1-i):
        if record[j][0] < record[j+1][0]:
            record[j] , record[j+1] = record[j+1] , record[j]
sum = 0
for i in record:
    if i[2] < w0:
        sum+= i[1]
        w0 -= i[2]
    else:
        sum+= i[0] * w0
        break
print('%.1f' % sum)
```

2. 选做题目

02659:Bomb Game （选做）

matrices, http://cs101.openjudge.cn/practice/02659/

思路:

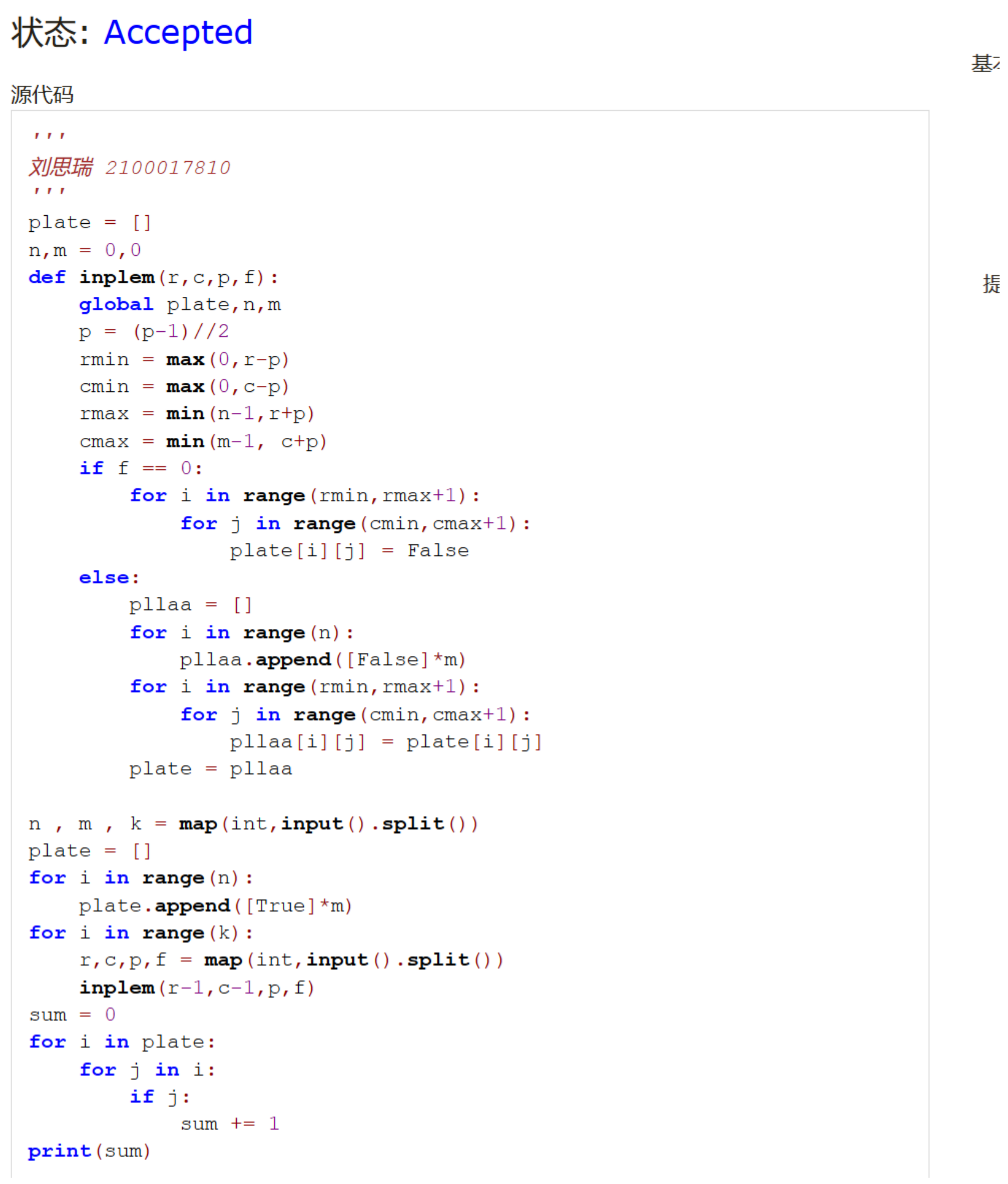
直接操作矩阵即可

代码

```
'''
刘思瑞 2100017810
'''
plate = []
n,m = 0,0
def inplem(r,c,p,f):
    global plate,n,m
    p = (p-1)//2
    rmin = max(0,r-p)
    cmin = max(0,c-p)
    rmax = min(n-1,r+p)
    cmax = min(m-1, c+p)
    if f == 0:
        for i in range(rmin,rmax+1):
            for j in range(cmin,cmax+1):
```

```python
                plate[i][j] = False
    else:
        pllaa = []
        for i in range(n):
            pllaa.append([False]*m)
        for i in range(rmin,rmax+1):
            for j in range(cmin,cmax+1):
                pllaa[i][j] = plate[i][j]
        plate = pllaa

n , m , k = map(int,input().split())
plate = []
for i in range(n):
    plate.append([True]*m)
for i in range(k):
    r,c,p,f = map(int,input().split())
    inplem(r-1,c-1,p,f)
sum = 0
for i in plate:
    for j in i:
        if j:
            sum += 1
print(sum)
```

代码运行截图



状态: Accepted

基

源代码

提

```python
'''
刘思瑞 2100017810
'''
plate = []
n,m = 0,0
def inplem(r,c,p,f):
    global plate,n,m
    p = (p-1)//2
    rmin = max(0,r-p)
    cmin = max(0,c-p)
    rmax = min(n-1,r+p)
    cmax = min(m-1, c+p)
    if f == 0:
        for i in range(rmin,rmax+1):
            for j in range(cmin,cmax+1):
                plate[i][j] = False
    else:
        pllaa = []
        for i in range(n):
            pllaa.append([False]*m)
        for i in range(rmin,rmax+1):
            for j in range(cmin,cmax+1):
                pllaa[i][j] = plate[i][j]
        plate = pllaa

n , m , k = map(int,input().split())
plate = []
for i in range(n):
    plate.append([True]*m)
for i in range(k):
    r,c,p,f = map(int,input().split())
    inplem(r-1,c-1,p,f)
sum = 0
for i in plate:
    for j in i:
        if j:
            sum += 1
print(sum)
```

CF545C: Woodcutters, dp/greedy, 1500 （选做）

https://codeforces.com/problemset/problem/545/C

思路：

代码

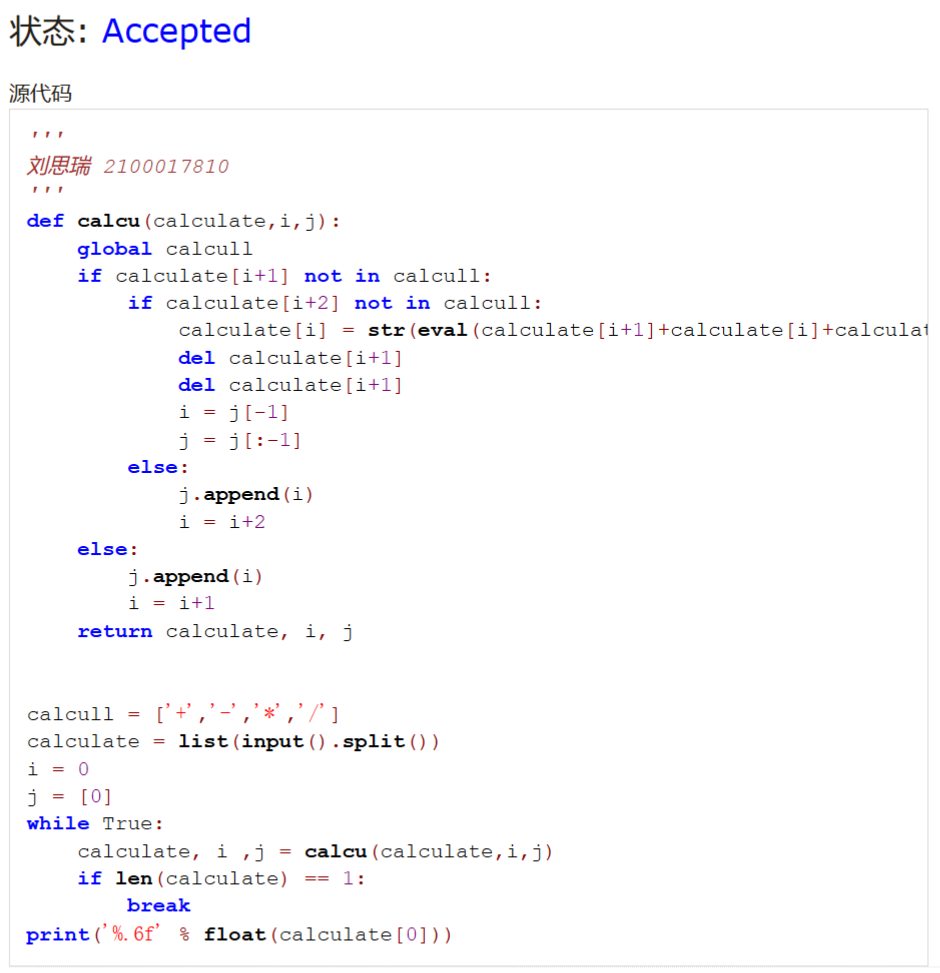
```
#

```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

3. 学习总结和收获

OJ02694:波兰表达式

这题主要是递归，但是我竟然卡在了输出的部分。。后来了解了c的双精度浮点才知道输出的格式最后ac了



状态: Accepted

源代码

```
'''
刘思瑞 2100017810
'''
def calcu(calculate,i,j):
    global calcull
    if calculate[i+1] not in calcull:
        if calculate[i+2] not in calcull:
            calculate[i] = str(eval(calculate[i+1]+calculate[i]+calculat
            del calculate[i+1]
            del calculate[i+1]
            i = j[-1]
            j = j[:-1]
        else:
            j.append(i)
            i = i+2
    else:
        j.append(i)
        i = i+1
    return calculate, i, j


calcull = ['+','-','*','/']
calculate = list(input().split())
i = 0
j = [0]
while True:
    calculate, i ,j = calcu(calculate,i,j)
    if len(calculate) == 1:
        break
print('%.6f' % float(calculate[0]))
```

OJ26977:接雨水

也是递归但是好像有更简单的做法（

状态: Accepted

源代码

```
'''
刘思瑞 2100017810
'''
columnn = []
def dp(i,key):
    global columnn
    if i == n - 1:
        return
    if key == 2:
        return
    if columnn[i+1] >= columnn[i]:
        dp(i+1,key)
        return
    else:
        for j in range(i+1,n):
            if columnn[j] >= columnn[i]:
                for k in range(i+1, j):
                    columnn[k] = columnn[i]
                dp(j,key)
                return
        key += 1
        columnn = columnn[::-1]
        dp(0,key)
        return



n = int(input())
columnn = list(map(int,input().split()))
a = sum(columnn)
dp(0,0)
print(sum(columnn)-a)
```