Assignment #6: 贪心和矩阵

Updated 1925 GMT+8 Oct 17, 2023

2023 fall, Complied by Xinjie Song, Phy

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

操作系统：Windows 11 22H2

Python编程环境：PyCharm 2023.2 (Community Edition)

C/C++编程环境：g++ (x86_64-win32-seh-rev0, Built by MinGW-W64 project) 8.1.0

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 ≤ *n*, *m* ≤ 1000, 1 ≤ *k* ≤ 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
```

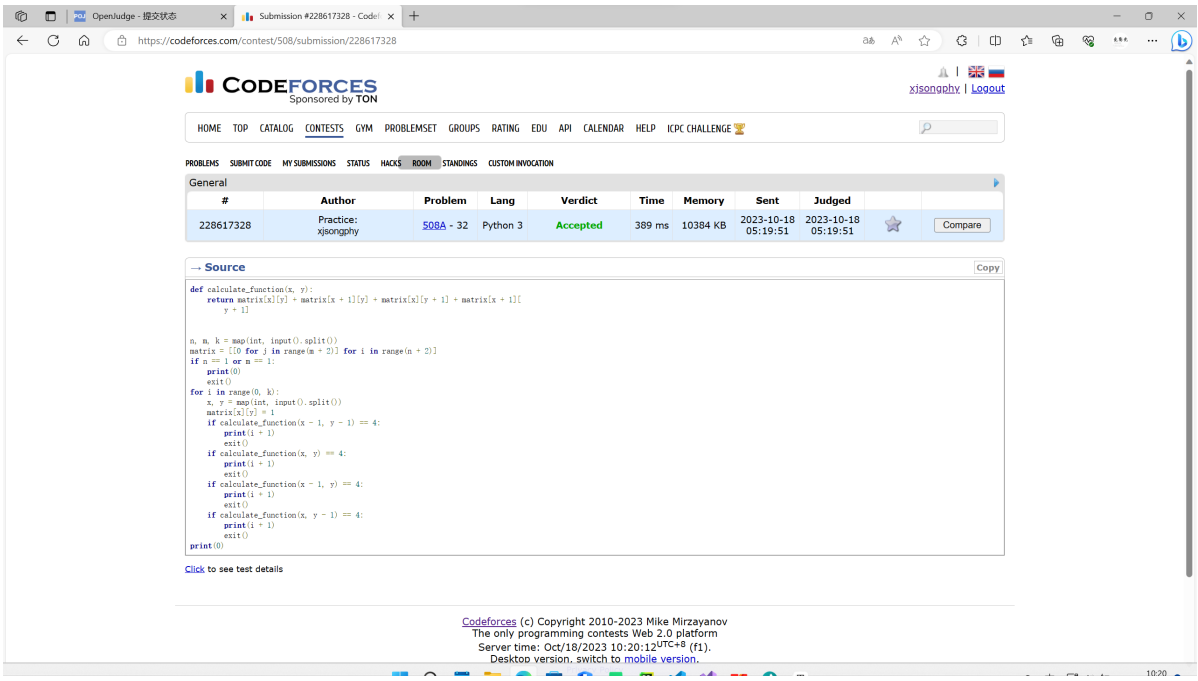
思路：模拟法，执行完每一步检查是否形成正方形，若形成则输出当前步数并退出

代码

```
def calculate_function(x, y):
    return matrix[x][y] + matrix[x + 1][y] + matrix[x][y + 1] + matrix[x + 1][
        y + 1]


n, m, k = map(int, input().split())
matrix = [[0 for j in range(m + 2)] for i in range(n + 2)]
if n == 1 or m == 1:
    print(0)
    exit()
for i in range(0, k):
    x, y = map(int, input().split())
    matrix[x][y] = 1
    if calculate_function(x - 1, y - 1) == 4:
        print(i + 1)
        exit()
    if calculate_function(x, y) == 4:
        print(i + 1)
        exit()
    if calculate_function(x - 1, y) == 4:
        print(i + 1)
        exit()
    if calculate_function(x, y - 1) == 4:
        print(i + 1)
        exit()
print(0)
```

代码运行截图



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

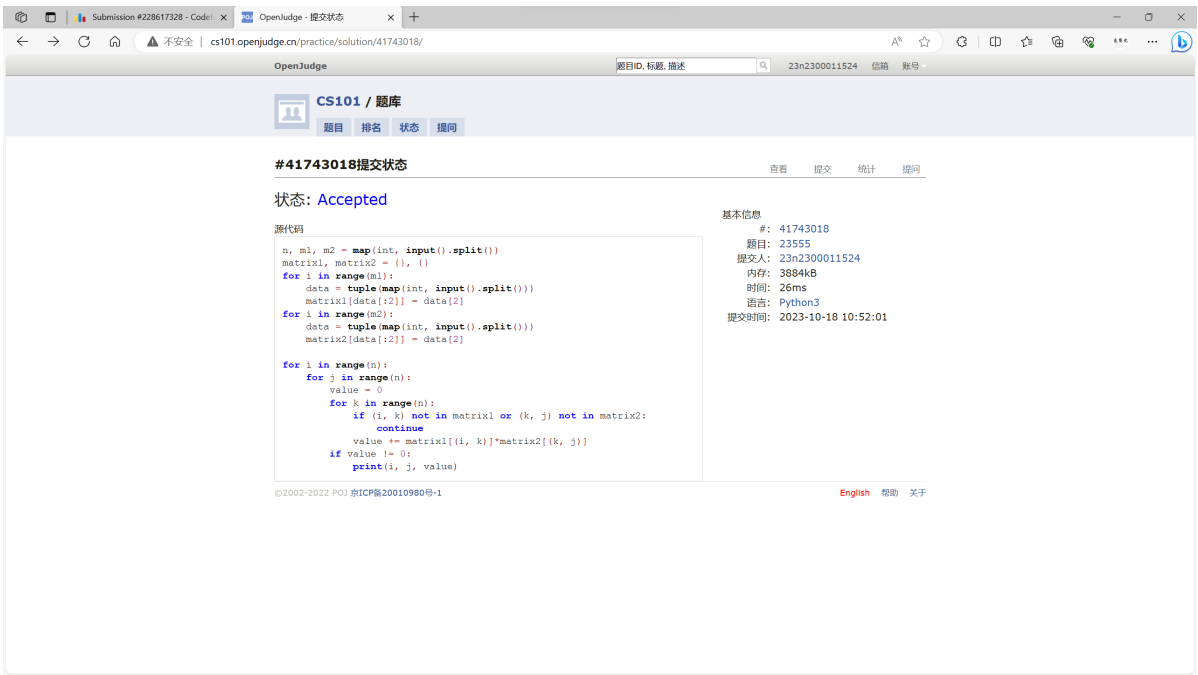
思路：正常思路

代码

```
n, m1, m2 = map(int, input().split())
matrix1, matrix2 = {}, {}
for i in range(m1):
    data = tuple(map(int, input().split()))
    matrix1[data[:2]] = data[2]
for i in range(m2):
    data = tuple(map(int, input().split()))
    matrix2[data[:2]] = data[2]

for i in range(n):
    for j in range(n):
        value = 0
        for k in range(n):
            if (i, k) not in matrix1 or (k, j) not in matrix2:
                continue
            value += matrix1[(i, k)]*matrix2[(k, j)]
        if value != 0:
            print(i, j, value)
```

代码运行截图



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：在外侧补充0来方便计算某个细胞周围的活细胞数

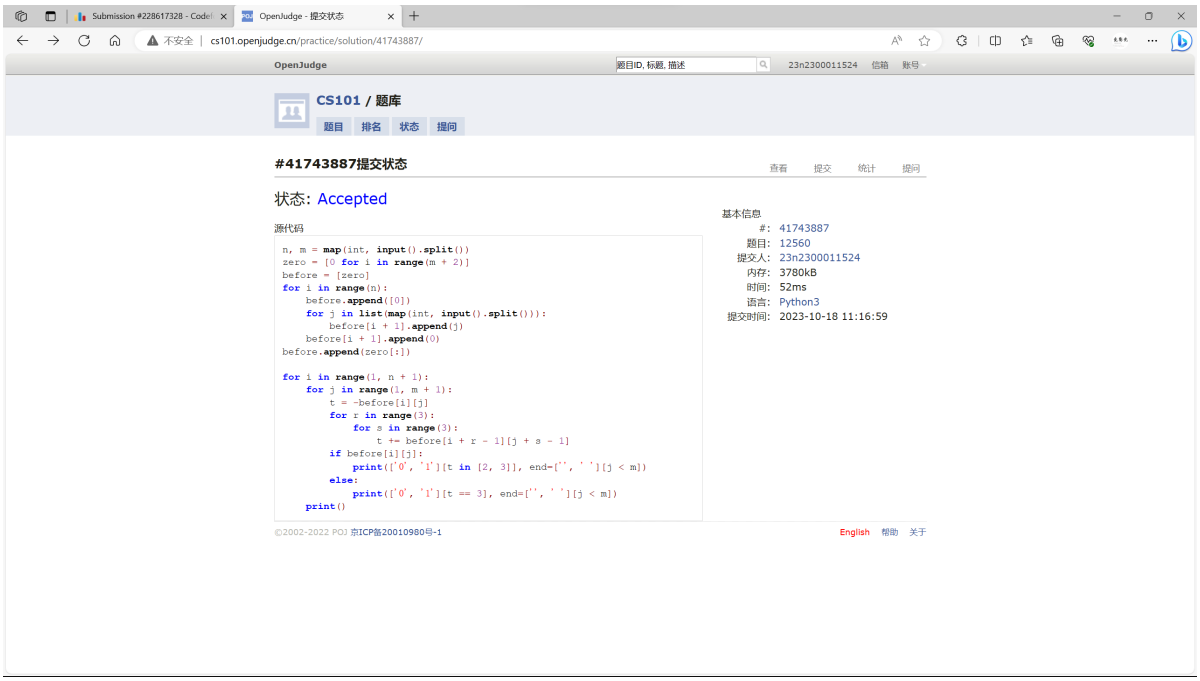
代码

```
n, m = map(int, input().split())
zero = [0 for i in range(m + 2)]
before = [zero]
for i in range(n):
    before.append([0])
```

```
    for j in list(map(int, input().split())):
        before[i + 1].append(j)
    before[i + 1].append(0)
before.append(zero[:])

for i in range(1, n + 1):
    for j in range(1, m + 1):
        t = -before[i][j]
        for r in range(3):
            for s in range(3):
                t += before[i + r - 1][j + s - 1]
        if before[i][j]:
            print(['0', '1'][t in [2, 3]], end=['', ' '][j < m])
        else:
            print(['0', '1'][t == 3], end=['', ' '][j < m])
    print()
```

代码运行截图



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

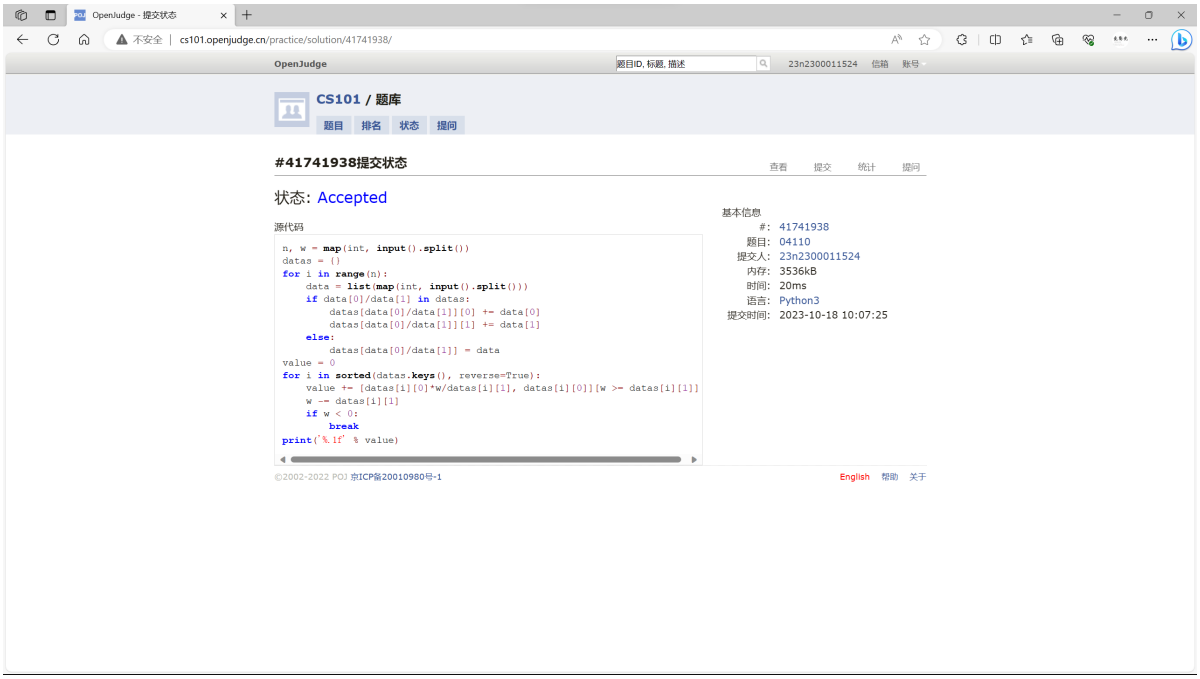
```
1193.0
```

思路：简单贪心，优先放入性价比大的糖果

代码

```
n, w = map(int, input().split())
datas = {}
for i in range(n):
    data = list(map(int, input().split()))
    if data[0]/data[1] in datas:
        datas[data[0]/data[1]][0] += data[0]
        datas[data[0]/data[1]][1] += data[1]
    else:
        datas[data[0]/data[1]] = data
value = 0
for i in sorted(datas.keys(), reverse=True):
    value += [datas[i][0]*w/datas[i][1], datas[i][0]][w >= datas[i][1]]
    w -= datas[i][1]
    if w < 0:
        break
print('%.1f' % value)
```

代码运行截图



2. 选做题目

02659:Bomb Game （选做）

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

思路：模拟法；如果目标在范围里，则范围内每个值加1；如果目标不在范围里，考虑到最多尝试99次，讲增量设置为-100可以保证该区域最终数值不会大于0；模拟结束后，先找到最大值，再统计最大值的个数，即为可能位置的个数。

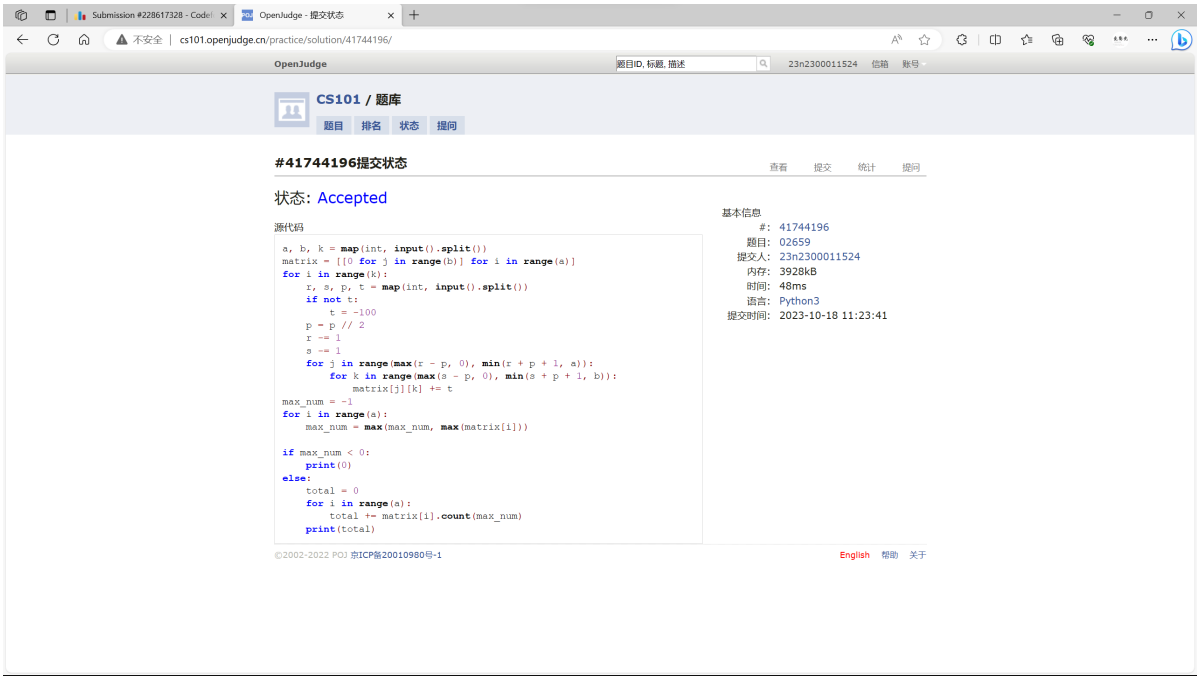
代码

```
a, b, k = map(int, input().split())
matrix = [[0 for j in range(b)] for i in range(a)]
for i in range(k):
    r, s, p, t = map(int, input().split())
    if not t:
        t = -100
    p = p // 2
    r -= 1
    s -= 1
    for j in range(max(r - p, 0), min(r + p + 1, a)):
        for k in range(max(s - p, 0), min(s + p + 1, b)):
            matrix[j][k] += t
max_num = -1
for i in range(a):
    max_num = max(max_num, max(matrix[i]))

if max_num < 0:
    print(0)
```

```
else:
    total = 0
    for i in range(a):
        total += matrix[i].count(max_num)
    print(total)
```

代码运行截图



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

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

思路：考虑到两棵树中间的位置只能由这两棵树占据，因此不必考虑其他的树，为了节约时间，读取数据同时对数据进行处理，这就会导致砍倒某棵树时，右侧的树信息未知，因此如果某棵树左侧有空间，那么就将这棵树倒向左侧；否则，才考虑能否倒向右侧。需要注意的是，最外侧的树一定是可以砍倒的，但如果只有一棵树，即这棵树既是最左侧的树，又是最右侧的树，这时注意不要输出2。

代码

```
n, total = int(input()), 1
trees = {}

for i in range(n):
    trees[i] = list(map(int, input().split())) + [0]
    if i == 0:
        trees[0][2] = 1
    elif i > 1:
        if trees[i - 1][0] - trees[i - 2][0] > [0, trees[i - 2][1]][trees[i - 2]
[2] == -1] + trees[i - 1][1]:
            trees[i - 1][2] = 1
```

```
            total += 1
        elif trees[i][0] - trees[i - 1][0] > trees[i - 1][1]:
            trees[i - 1][2] = -1
            total += 1
print(total + [1, 0][n == 1])
```

代码运行截图

3. 学习总结和收获

最近做题时，第一次AC后，一般会继续缩短代码、减少运行时间；平时按《算法基础与在线实践》上的顺序做题，有时间便会做新加入的题目。这次作业的最后一题，思考时间比较长，但最后发现贪心策略很简单，主要是没有注意到两棵树中间的位置只能由这两棵树占据，导致完成时间较长。

截至2023年10月20日，OJ完成题目78道，CF完成题目32道。