两数之和

（14分）给定一个整数数组 `nums` 和一个目标值 `target`，请你在该数组中找出和为目标值的那**两个**整数，并返回他们的数组下标。

你可以假设每种输入只会对应一个答案。但是，你不能重复利用这个数组中同样的元素。

本题为核心代码模式，仅需补全函数，输入为函数参数，输出值由return返回。

样例输入

```
[2, 7, 11, 15]
9
```

样例输出

```
[0, 1]
```

样例解释

因为 nums[**0**] + nums[**1**] = 2 + 7 = 9

```python
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        dict={}
        for index,num in enumerate(nums):
            if target-num in dict:
                return [dict[target-num],index]
            dict[num] = index
        return None
```

斐波那契数

题目描述

（14分）

斐波那契数，通常用 `F(n)` 表示，形成的序列称为**斐波那契数列**。该数列由 `0` 和 `1` 开始，后面的每一项数字都是前面两项数字的和。也就是：

```
F(0) = 0,   F(1) = 1
F(N) = F(N - 1) + F(N - 2), 其中 N > 1.
```

给定 N，计算 F(N)。

示例 **1**:

```
输入：2
输出：1
解释：F(2) = F(1) + F(0) = 1 + 0 = 1.
```

示例 **2**:

```
输入：3
输出：2
解释：F(3) = F(2) + F(1) = 1 + 1 = 2.
```

示例 **3**:

```
输入：4
输出：3
解释：F(4) = F(3) + F(2) = 2 + 1 = 3.
```

提示:

- 0 <= N <= 30

本题为核心代码模式，无需处理输入输出。

样例输入

```
2
```

样例输出

```
1
```

样例输入

```
3
```

样例输出

```
2
```

样例输入

```
4
```

样例输出

```
3
```

```
class Solution:
    def fib(self, N: int) -> int:
        if N == 1:
            return 1
        elif N == 0:
            return 0
        return self.fib(N - 1)+self.fib(N - 2)
```

青蛙跳台阶

题目描述

（14分）一只青蛙一次可以跳上1级、2级、3级台阶。求该青蛙跳上一个 n 级的台阶总共有多少种跳法。结果可能很大，你需要对结果模1000000007。

从本题开始，同学们需要自己对输入输出进行处理，以及用到某些处理函数时，需要import对应的库，如对本题来说，输入为一个数字，输出使用print函数，那么我们就可以在完成函数后，在后面加上如下的代码：

```
n=input()
n=int(n)
sol=Solution()
print(sol.waysToStep(n))
```

请同学们注意上述代码的第二行，为什么这里要使用一个类型转换?
希望同学们可以通过该例子，了解python中的输入输出.

示例1:

```
输入：3
输出：4
说明：有四种跳法
```

提示：
动态规划

```python
def ex1_3(N: int) -> int:
    if N < 1:
        return 0
    if N == 1:
        return 1
    if N == 2:
        return 2
    if N == 3:
        return 4
    f1, f2, f3 = 1, 2, 4
    for i in range(4, N + 1):
        fn = f1 + f2 + f3
        f1, f2, f3 = f2, f3, fn
    return fn

N = int(input())
print(int(ex1_3(N)%(1e9+7)))
```

跳方格游戏

题目描述

（14分）给定一个长度为 n 的 0 索引整数数组 nums。初始位置为 nums[0]。

每个元素 nums[i] 表示从索引 i 向前跳转的最大长度。换句话说，如果你在 nums[i] 处，你可以跳转到任意 nums[i + j] 处：

```
0 <= j <= nums[i]
i + j < n
返回到达 nums[n - 1] 的最小跳跃次数。生成的测试用例可以到达 nums[n - 1]。
```

如何处理这个形如 C 语言中的数组的输入呢？要对上一题中的输入输出做哪些调整？
是否有一个函数，可以便利地实现输入的类型转换？

示例 **1:**

```
输入：[2,3,1,1,4]
输出：2
解释：跳到最后一个位置的最小跳跃数是 2。
     从下标为 0 跳到下标为 1 的位置，跳 1 步，然后跳 3 步到达数组的最后一个位置。
```

示例 **2:**

```
输入：[1,1,1,2,1]
输出：4
```

```
def jump(nums):
    n = len(nums)
    # print('n:',n)
    max_pos, next_pos = 0, 0
    jumps = 0
    for i in range(n-1):
        next_pos = max(next_pos, i + nums[i])
        # print(next_pos,":l r:",i + nums[i])
        if i == max_pos:
            max_pos = next_pos
            jumps += 1
            if max_pos >= n-1:
                break
        # print('next:', next_pos,' max:',max_pos)
    return jumps

input_str = input()[1:-1]
input_list = input_str.split(',')
nums = list(map(int, input_list))

print(jump(nums))  # 输出结果为 2
```

二维数组求和

题目描述

（16分）给定一个二维数组，请对其中各个一维子数组求和，并将这些子数组的和作为输出列表的一个元素。

本题将不再帮助大家处理输入输出，请自己编写代码处理。

需要注意的是，使用input输入二维数组是string类，要先将其转化为list类

提示：
尝试一下python的字符串处理函数

样例输入

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

样例输出

```
[6, 15, 24]
```

样例解释

输入string型，输出list型

样例输入

```
[[1,2,3],[4],[5,6]]
```

样例输出

```
[6,4,11]
```

```
import ast
s = input()
mat = ast.literal_eval(s)
lst=[]
for t in mat:
    lst.append(sum(t))
print(lst)
```

胡闹厨房

题目描述

（14分）你是一家饭店中唯一的厨师，现在面前有 n 份堂食订单， 你需要争分夺秒完成顾客的订单。 现在输入为一个数组cooks， 其中 cooks[i] = [durationi, Secondi] 表示做第 i 份订单所需花费的秒数为 durationi ，且完成该订单的时间不晚于 Secondi ， 否则顾客就会因为等不及而愤怒离开。

你的厨师工作从第 1 秒开始， 不能同时做两份及两份以上的饭， 且一旦开始做一份饭，就不能在未完成时更换目标。

返回你最多满足的顾客数目。

请同学们自己处理输入输出，包括把输入的string类的字符串转成二维数组的过程。

提示：
可以导入相关数据结构库

样例输入

```
[[100, 200], [200, 1300], [1000, 1250], [2000, 3200]]
```

样例输出

```
3
```

样例解释

共4份订单，此时最多可以完成其中的3份
首先做订单1，耗时100秒，在第100秒完成，在第101天开始做下一份饭。
然后做订单3，耗时1000秒，在第1100秒完成，在第1101秒开始做下一份饭。
然后做订单2，耗时200天，在第1300秒完成。
此时已无暇顾及订单4，因为做好订单4的时间：第3300秒超出了顾客的忍耐上限，该顾客已经离开。

样例输入

```
[[3,2],[4,3]]
```

样例输出

```
0
```

样例输入

```
[[1,2]]
```

样例输出

```
1
```

```python
import ast
def ex():
    s = input()
    lst = ast.literal_eval(s)
    sorted_lst = sorted(lst, key=lambda x: (x[0], x[1]))
    sorted_lst2 = sorted(lst, key=lambda x: (x[1], x[0]))
    # print(sorted_lst)
    sum_time = 0
    ans = 0
    for t in sorted_lst:
        if t[0] + sum_time <= t[1]:
            sum_time += t[0]
            # print(t)
            ans += 1
```

```
    sum_time2 = 0
    ans2 = 0
    for t in sorted_lst2:
        if t[0] + sum_time2 <= t[1]:
            sum_time2 += t[0]
            # print(t)
            ans2 += 1

    ans = max(ans,ans2)
    print(ans)


ex()
```

提取歌手和歌名

题目描述

（16分，和提取网页链接2选1）给定一段网页的 html 代码，请提取出歌手和歌名。

提示：
你需要自己处理输入和输出。
涉及到的语法：正则表达式

python 获取一行输入（以 \n 结束）：

```
temp = input()
```

python 正则表达式需要导入 re 包

```
import re
```

输入描述

输入的文本以两个 \n\n 作为结束标志。

样例输入

```
<div id="songs-list">
    <h2 class="title">classic songs</h2>
    <p class="introduction">
        song list
    </p>
    <ul id="list" class="list-group">
        <li data-view="2">yiluyouni</li>
        <li data-view="7">
```

```
            <a href="/2.mp3" singer="renxianqi">canghaiyishengxiao</a>
        </li>
        <li data-view="5">
            <a href="/6.mp3" singer="denglijun">danyuanrenchangjiu</a>
        </li>
    </ul>
</div>
```

样例输出

```
[('renxianqi', 'canghaiyishengxiao'), ('denglijun',
'danyuanrenchangjiu')]
```

```
import re
def ex1_7():
    text = ''
    while True:
        line = input()
        if line == '':
            break
        text += line
    # Alist = text.
    pattern = re.compile('<a href=".*?" singer="(.*?)">(.*?)</a>')
    info = pattern.findall(text)
    print(info)
    # print(Alist)

ex1_7()
```

提取网页中的链接

题目描述

（16分，和提取歌手2选1）给定一个文本，请提取出文本中的所有链接，以 list 形式输出。

提示：
需要你自己处理输入和输出。
涉及到 python 的正则表达式语法。

```
import re
```

输入描述

输入的文本以两个 \n\n 作为结束标志。

样例输入

```
Its after 12 noon, do you know where your rooftops are?
http://tinyurl.com/NYCRooftops
```

样例输出

```
['http://tinyurl.com/NYCRooftops']
```

样例解释

匹配到的网址有一个，['http://tinyurl.com/NYCRooftops']

样例输入

```
<a title="
校党委书记舒歌群调研马克思主义学院
" href="http://news.ustc.edu.cn/info/1055/81856.htm" target="_blank">
校党委书记舒歌群调研马克思主义学院
</a>
<a title="
中国科大合作研究在低维硼领域取得新进展
" href="http://news.ustc.edu.cn/info/1055/81977.htm" target="_blank">
中国科大合作研究在低维硼领域取得新进展
</a>
```

样例输出

```
['http://news.ustc.edu.cn/info/1055/81856.htm',
'http://news.ustc.edu.cn/info/1055/81977.htm']
```

样例解释

共匹配到两个网址：['http://news.ustc.edu.cn/info/1055/81856.htm', 'http://news.ustc.edu.cn/info/1055/81977.htm']

```
import re

text = ''
while True:
    line = input()
    if line == '':
        break
    text += line
pattern = r'(https?://\S+)'
```

```
links = re.findall(pattern, text)
res = []
for link in links:
    link = link.split(',')
    res.extend(link)
for link in res:
    if link == '':
        res.remove(link)
for link in res:
    if link[-1] == '"':
        res[res.index(link)] = link[0:-1]

print(res)
```

N 皇后

题目描述

（选做，不计分）某同学最近捡到了一个棋盘，他想在棋盘上摆放 K 个皇后。他想知道在他摆完这 K 个皇后之后，棋盘上还有多少个格子是不会被攻击到的。
注意：一个皇后会攻击到这个皇后所在的那一行，那一列，以及两条对角线。

```
输入描述：
第一行三个正整数 n,m,K，表示棋盘的行列，以及摆放的皇后的个数。
接下来 K 行，每行两个正整数 x,y，表示这个皇后被摆在了第 x 行，第 y 列，数据保证任何两个皇后都不会被摆在同一个格子里。

输出描述：
棋盘上不会被攻击到的格子数量
```

样例输入

```
12 13 6
10 4
12 10
1 1
2 3
3 2
2 6
```

样例输出

```
25
```

样例输入

```
2 2 2
1 2
2 1
```

样例输出

```
0
```

样例解释

此时没有不会被攻击的格子

```
from typing import List

def ex1_8(n, m, k, queens):
    board = [[0] * m for _ in range(n)]
    for r, c in queens:
        for i in range(n):
            board[i][c] = 1  # 标记列
        for j in range(m):
            board[r][j] = 1  # 标记行
        i, j = r, c
        while i >= 0 and j >= 0:
            board[i][j] = 1  # 标记左上方
            i, j = i - 1, j - 1
        i, j = r, c
        while i >= 0 and j < m:
            board[i][j] = 1  # 标记右上方
            i, j = i - 1, j + 1
        i, j = r, c
        while i < n and j >= 0:
            board[i][j] = 1  # 标记左下方
            i, j = i + 1, j - 1
        i, j = r, c
        while i < n and j < m:
            board[i][j] = 1  # 标记右下方
            i, j = i + 1, j + 1
    return sum(1 for i in range(n) for j in range(m) if board[i][j] ==
0)

n, m, K = map(int, input().split())
queens = []
for i in range(K):
    x, y = map(int, input().split())
    queens.append((x - 1, y - 1))  # 将行列从 1 开始改为从 0 开始
print(ex1_8(n, m, K, queens))
```