二分法

1. 初始化左右边界（low和high）为数据集的两端
2. 当左边界小于等于右边界时执行以下步骤： a. 计算中间位置 mid = (low + high) // 2 b. 如果目标值等于中间元素，返回 mid c. 如果目标值小于中间元素，更新右边界为 mid - 1 d. 如果目标值大于中间元素，更新左边界为 mid + 1
3. 如果左边界大于右边界，则目标值不存在于数据集中

```
def binary_search(arr, target):
    low, high = 0, len(arr) - 1

    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid  # 找到目标值，返回索引
        elif arr[mid] < target:
            low = mid + 1
        else:
            high = mid - 1

    return -1  # 目标值不存在
```

使用例子

```
data = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
target_value = 7
result = binary_search(data, target_value)

if result != -1:
    print(f"目标值 {target_value} 在数组中的索引是 {result}")
else:
    print(f"目标值 {target_value} 不存在于数组中")
```

质数

```
prime_number = [True] * (x+1)
for i in range(2,x+1):
    if prime_number[i]:
        for j in range(2*i,x+1,i):
            prime_number[j] = False
```

欧拉筛法

```
def euler_sieve(n):
    primes = [True for _ in range(n + 1)]
    p = 2
    while p * p <= n:
        if primes[p]:
            for i in range(p * p, n + 1, p):
                primes[i] = False
        p += 1
    return [num for num in range(2, n + 1) if primes[num]]

# 输入一个数
num = int(input("输入一个数字: "))

# 判断是否是质数并输出结果
result = euler_sieve(num)
print(f"{num} {'是' if result else '不是'} 质数")
```

matrix相乘

```
for i in range(n):
    for j in range(n):
        for N in range(n):
            matrix_ans[i][j] += matrix_X[i][N] * matrix_Y[N][j]
```

matrix input,output的方法

```
#input
n, m = map(int, input().split()) #n是行，m是列
matrix = [[int(num) for num in input().split()] for _ in range(n)]

#output
for row in matrix:
print(" ".join(map(str,row)))
```

greedy解决背包问题

```
n,w = map(int,input().split()) #n行，w是能承受的重量
candies = []
for _ in range(n):
    value,weight = map(int,input().split())
    candies.append((value/weight,value,weight)) #tuple的形式存储在list

candies.sort(reverse = True)

total_value = 0
for i,v,weight in candies:
```

```
    if w >= weight: #背包容量 > 目前糖果的重量
        total_value += v #整箱糖果装进来
        w -= weight #剩余容量
    else:
        total_value += w*i #只能装一箱糖果中的部分
        break

print("{:.1f}".format(total_value))
```

dp解决背包问题

```
t,m=map(int,input().split())
sum_value=[]
for i in range(m):
    sum_value.append([int(x) for x in input().split()]) #存储每个物品的价值和重量

dp=[0]*(t+1)
for i in range(m):
    for j in range(t,sum_value[i][0]-1,-1):  #sum_value[i][0]是当前物品重量
        if j >= sum_value[i][0]:
            dp[j]=max(dp[j],dp[j-sum_value[i][0]]+sum_value[i][1]) #选择当前物品或不选择当前物品，哪个总价值更大

print(dp[-1])
```

语法

语法

1.截取

```
s = 'abcdef'
s[1:5] #[start,end(不包括该数字),step]
'bcde'
```

2.dictionary

```
print dict.keys()      # 输出所有键
print dict.values()    # 输出所有值
dict.get(key, default=None)-- 返回指定键的值，如果值不在字典中返回default值
```

3.转换

```
chr(x)-- 将一个整数转换为一个字符
ord(x)-- 将一个字符转换为它的整数值
hex(x)-- 将一个整数转换为一个十六进制字符串
oct(x)-- 将一个整数转换为一个八进制字符串
bin(x)-- 将一个整数转换为一个二进制字符串
```

4.有关数字的转换（import math）

```
abs(x)-- 返回数字的绝对值，如abs(-10) 返回 10
ceil(x)-- 返回数字的上入整数，如math.ceil(4.1) 返回 5
cmp(x, y)-- 如果 x < y 返回 -1, 如果 x == y 返回 0, 如果 x > y 返回 1
exp(x)-- 返回e的x次幂(ex),如math.exp(1) 返回2.718281828459045
fabs(x)-- 以浮点数形式返回数字的绝对值，如math.fabs(-10) 返回10.0
floor(x)-- 返回数字的下舍整数，如math.floor(4.9)返回 4
log(x)-- 如math.log(math.e)返回1.0,math.log(100,10)返回2.0
log10(x)-- 返回以10为基数的x的对数，如math.log10(100)返回 2.0
max(x1, x2,...)-- 返回给定参数的最大值，参数可以为序列。
min(x1, x2,...)-- 返回给定参数的最小值，参数可以为序列。
modf(x)-- 返回x的整数部分与小数部分，两部分的数值符号与x相同，整数部分以浮点型表示。
pow(x, y)-- x**y 运算后的值。
round(x [,n])-- 返回浮点数x的四舍五入值，如给出n值，则代表舍入到小数点后的位数。
sqrt(x)-- 返回数字x的平方根
```

5.有关字符串的内建函数

```
string.capitalize()-- 把字符串的第一个字符大写
string.count(str, beg=0, end=len(string))-- 返回 str 在 string 里面出现的次数，如果 beg 或者 end 指定则返回指定范围内 str 出现的次数
string.endswith(obj)-- 检查字符串是否以 obj 结束，如果是，返回 True,否则返回 False.
string.find(str)-- 检测 str 是否包含在 string 中，如果是返回开始的索引值，否则返回-1
string.isalpha()-- 至少有一个字符并且所有字符都是字母
string.isdecimal()-- 十进制数字
string.isdigit()-- 数字
string.islower()-- 检查是否全小写
string.isnumeric()-- 数字字符
string.join(seq)-- 以 string 作为分隔符，将 seq 中所有的元素(的字符串表示)合并为一个新的字符串
string.replace(str1, str2,  num=string.count(str1))-- 把 string 中的 str1 替换成 str2,如果 num 指定，则替换不超过 num 次.
string.rfind(str)-- 返回字符串最后一次出现的位置，如果没有匹配项则返回 -1。
string.rindex( str )-- 不过是返回最后一个匹配到的子字符串的索引号。
string.rstrip()-- 删除 string 字符串末尾的空格.
string.swapcase()-- 翻转 string 中的大小写。
```

6.for循环

```
for i in [2,-2]-- 在2和-2之间循环
for i in range(-2,2)-- 在-2到1之间循环（-2，-1，0，1）
```

打怪兽 Q神无聊的时候经常打怪兽。现在有一只怪兽血量是b，Q神在一些时刻可以选择一些技能打怪兽，每次释放技能都会让怪兽掉血。 现在给出一些技能ti,xi，代表这个技能可以在ti时刻使用，并且使得怪兽的血量下降xi。这个打怪兽游戏有个限制，每一时刻最多可以使用m个技能（一个技能只能用一次）。如果技能使用得当，那么怪兽会在哪一时刻死掉呢?

```
s = int(input())
for _ in range(s):
    n,m,b = map(int,input().split(" "))    #n=技能，m=每个时刻最多使用的技能，b=
血量
    d = {}
    for _ in range(n):
        t,x = map(int,input().split(" "))    #t=时刻，x=血量下降
        if t not in d.keys():    #如果字典中没有记录过这一时刻
            d[t] = [x]    #将x添加到对应的列表中
        else:
            d[t].append(x)    #比如同一时刻有两种技能，{1:[5,5]} -- t:d[t]=[x]

    for i in d.keys():    #遍历每个时刻
        d[i].sort(reverse = True)    #对每个时刻里的技能按血量下降，降序排列
        d[i] = sum(d[i][:m])    #取排序后的前m个技能
    dp = sorted(d.items())    #对{时刻:血量下降}按照时刻顺序排列

    for i in dp:    #遍历每个{时刻：血量下降}
        b -= i[1]    #怪兽血量减去i[1]（血量下降）
        if b <= 0:
            print(i[0])    #print时刻
            break
    else:    #如果循环结束时怪兽血量仍然大于0
            print("alive")
```

计算鞍点 给定一个5*5的矩阵，每行只有一个最大值，每列只有一个最小值，寻找这个矩阵的鞍点。 鞍点指的是矩阵中的一个元素，它是所在行的最大值，并且是所在列的最小值。

```
matrix = []
for _ in range(5):
    row = list(map(int,input().split()))
    matrix.append(row)

found = False

for i in range(5):
    max_row_point = max(matrix[i])
    j = matrix[i].index(max_row_point)
    column = [matrix[k][j] for k in range(5)]
    min_column_point = min(column)
```

```
        if max_row_point == min_column_point:
            print(i+1,j+1,max_row_point)
            found = True
            break

    if not found:
        print("not found")
```

决战双十一 双十一来临之际，某猫推出了一系列眼花缭乱的活动。Casper希望借此机会，狠狠薅一把羊毛。他在购物车里准备了n个商品，分别来自店铺si，双十一狂欢价为pi。除了跨店的满200减30活动以外，总共m家店铺，每个都有自己的店铺券qj - xj，表示在该店铺中，商品狂欢价金额之和满qj即可减掉xj。

由于最终成交金额只有在支付页面才能看到，而Casper现在就想知道自己最终能薅到多少羊毛，希望你能帮他计算出最终的成交价。

注意每一家店铺只能使用一张店铺券；跨店满减计算的总价是双十一狂欢价的金额总和，每满200减30。

n和m，分别表示有n个商品和m家店铺（1 < n <10000，1 < m < 100）

接下来n行分别是商品i的店铺si与它的双十一狂欢价pi（1 <= si <= m，1<= i<= n）

最后m行中，每一行表示第j家店铺的优惠券，用qj - xj表示，满足qj >= xj（1 <= j <= m）

```
n,m = map(int,input().split())   #n=商品，m=店铺
total_price = 0
store_price = [0 for i in range(m)]
for _ in range(n):
    s,p = map(int,input().split())   #s=店铺，p=商品价格
    total_price += p
    store_price[s-1] += p

discount = (total_price//200)*30
for i in range(m):
    q,x = map(int,input().split("-"))   #q-x=优惠券
    if store_price[i] >= q:
        discount += x

print(total_price - discount)
```

2022决战双十一 又是一年双十一，某猫一如既往推出一系列活动，去年尝到甜头的Casper希望今年继续。今年他希望从m个店铺中购买n个商品，每个商品可能在一个或多个店铺中以不同的标价出售。此次跨店满减的活动升级为每满300减50，此外每个店铺也可能有多张不同档位的店铺券"q-x"，希望你能够帮助Casper算出他最少花多少钱买到这n个商品（每个商品只买一件） 注意，每一家店铺的店铺券只能用一张，对于任意一张店铺券"q-x"，他表示在当前商铺购买的所有商品标价达到q时，最终应付款可以减x。 而跨店满减活动可以叠加使用，它是指所有商品标价之和每满300，可以减去50。

输入 第一行为两个整数 n 和 m ，分别表示有 n 个商品和 m 个店铺（1 < n < 9，1 < m < 6） 接下来 n 行分别是 n 个商品的相关价格，其中第 i 行表示出售商品 i 的店铺及其标价，具体形式为s_i: p(i,s_i)，其中 s_i 为店

铺编号（1 <= s_i <= m），p(i,s_i) 为这个店铺关于这件商品的报价，每个店铺的标价用空格分开。 最后 m 行中，每一行表示一个店铺的优惠券，用 q-x 表示，满足 q >= x，每张优惠券间用空格分开。

输出 最低的成交价

```
def plans(n, price, count, all_plans, plan):  # 递归列出所有购买方案
    if count == n + 1:
        all_plans.append(plan[:])
        return
    for i in price[count].keys():
        plan.append(i)
        plans(n, price, count + 1, all_plans, plan)
        plan.pop()
    return


def buy(n, m, price, coupon):
    all_plans = list()  # 列出所有购买方案
    plans(n, price, 1, all_plans, [])
    # for i in all_plans:
    #     print(i)
    final_price = list()  # 每个方案的最终价格
    for plan in all_plans:  # 对每个购买方案
        totals_rsp = list()  # 每个店铺的总价
        prices = [price[i][plan[i - 1]] for i in range(1, n + 1)]  # 每个商
品的价格
        total = sum(prices)  # 所有商品的总价
        total -= total // 300 * 50  # 跨店满减
        for i in range(1, m + 1):  # 对每个店铺
            prices_rsp = [price[j + 1][plan[j]]
                          for j in range(n) if plan[j] == i]  # 每个商品在该
店铺的价格
            totals_rsp.append(sum(prices_rsp))  # 该店铺的总价
        store = 0
        for total_rsp in totals_rsp:
            store += 1
            discount = 0
            for j in coupon[store]:
                if total_rsp >= j[0]:
                    discount = max(j[1], discount)
            total -= discount
        final_price.append(total)
    # print(final_price)
    return min(final_price)


n, m = map(int, input().split())
price = dict()
coupon = dict()
for i in range(n):
    price_i = dict()
    price_raw = list(input().split())
```

```
        for j in price_raw:
            price_i[int(list(j.split(':'))[0])] = int(list(j.split(':'))[1])
        price[i + 1] = price_i
    for i in range(m):
        coupon_i = list()
        coupon_raw = list(input().split())
        for j in range(len(coupon_raw)):
            coupon_i.append(tuple(map(int, coupon_raw[j].split('-'))))
        coupon[i + 1] = coupon_i
    # print(n, m, price, coupon)
    print(buy(n, m, price, coupon))
```