1、ACM 编程 敏感字段加密

题目描述：【敏感字段加密】给定一个由多个命令字组成的命令字符串：

1、字符串长度小于等于 127 字节，只包含大小写字母，数字，下划线和偶数个双引号；

2、命令字之间以一个或多个下划线_进行分割；

3、可以通过两个双引号""来标识包含下划线_的命令字或空命令字（仅包含两个双引号的命令字），双引号不会在命令字内部出现；

请对指定索引的敏感字段进行加密，替换为****** (6 个*)，并删除命令字前后多余的下划线_。如果无法找到指定索引的命令字，输出字符串 ERROR。

输入描述：输入为两行，第一行为命令字索引 K（从 0 开始），第二行为命令字符串 S。

输出描述：输出处理后的命令字符串，如果无法找到指定索引的命令字，输出字符串 ERROR

示例 展开

```
示例1
输入：1
    password__a12345678_timeout_100
输出：password_******_timeout_100
说明：

示例2
输入：2
    aaa_password_"a12_45678"_timeout__100_""_
输出：aaa_password_******_timeout_100_""
说明：
```

2、ACM 编程 IPv4 地址转换成整数

题目描述：存在一种虚拟 IPv4 地址，由 4 小节组成，每节的范围为 0~255，以#号间隔，虚拟 IPv4 地址可以转换为一个 32 位的整数，例如：

128#0#255#255，转换为 32 位整数的结果为 2147549183（0x8000FFFF）

1#0#0#0，转换为 32 位整数的结果为 16777216（0x01000000）

现以字符串形式给出一个虚拟 IPv4 地址，限制第 1 小节的范围为 1~128，即每一节范围分别为(1~128)#(0~255)#(0~255)#(0~255)，要求每个 IPv4 地址只能对

应到唯一的整数上。如果是非法 IPv4，返回 invalid IP

输入描述：输入一行，虚拟 IPv4 地址格式字符串

输出描述：输出以上，按照要求输出整型或者特定字符

补充说明：输入不能确保是合法的 IPv4 地址，需要对非法 IPv4（空串，含有 IP 地址中不存在的字符，非合法的#分十进制，十进制整数不在合法区间内）进行识别，返回特

定错误

```
示例

示例1
输入：100#101#1#5
输出：1684340997
说明：

示例2
输入：1#2#3
输出：invalid IP
说明：
```

3、ACM 编程 字符匹配

题目描述：给你一个字符串数组（每个字符串均由小写字母组成）和一个字符规律（由小写字母和.和*组成），识别数组中哪些字符串可以匹配到字符规律上。

'.' 匹配任意单个字符，'*' 匹配零个或多个任意字符；判断字符串是否匹配，是要涵盖整个字符串的，而不是部分字符串。

输入描述：第一行为空格分割的多个字符串，1<单个字符串长度<100，1<字符串个数<100
第二行为字符规律，1<=字符规律长度<=50
不需要考虑异常场景
输出描述：匹配的字符串在数组中的下标（从 0 开始），多个匹配时下标升序并用,分割，若均不匹配输出-1

```
示例

示例1
输入：ab aab abacd
        .*
输出：0,1,2
说明：ab中a匹配. b匹配* 可以全匹配；aab中a匹配. ab匹配* 可以全匹配；abacd中a匹配. bacd匹配* 可以全匹配；输出对应字符串数组下标 0,1,2

示例2
输入：ab aab
        a.b
输出：1
说明：aab中第一个a匹配a 第二个a匹配. b匹配b 可以全匹配； 输出对应字符串数组下标1
```

1、ACM 编程 生日礼物

题目描述：小牛的孩子生日快要到了，他打算给孩子买蛋糕和小礼物，蛋糕和小礼物各买一个，他的预算不超过 x 元。蛋糕 cake 和小礼物 gift 都有多种价位的可供选择。
请返回小牛共有多少种购买方案。
输入描述：第一行表示 cake 的单价，以逗号分隔
第一行表示 gift 的单价，以逗号分隔
第三行表示 x 预算
输出描述：输出数字表示购买方案的总数
补充说明：1 <= cake.length <= 10^5
1 <= gift.length <= 10^5
1 <= cake[i],gift[i] <= 10^5
1 <= x <= 2*10^5

```
示例

示例1
输入：10,20,5
        5,5,2
        15
输出：6
说明：解释：小牛有6种购买方案，所选蛋糕与所选礼物在数组中对应的下标分别是：
        第1种方案：cake [0] + gift [0] = 10 + 5 = 15；
        第2种方案：cake [0] + gift [1] = 10 + 5 = 15；
        第3种方案：cake [0] + gift [2] = 10 + 2 = 12；
        第4种方案：cake [2] + gift [0] = 5 + 5 = 10；
        第5种方案：cake [2] + gift [1] = 5 + 5 = 10；
        第6种方案：cake [2] + gift [2] = 5 + 2 = 7。
```

2、ACM 编程 求符合条件元组个数

题目描述：给定一个整数数组 nums 、一个数字 k,一个整数目标值 target，请问 nums 中是否存在 k 个元素使得其相加结果为 target，请输出所有符合条件且不重复的 k 元组的个数
数据范围
2<= nums.length <= 200
-109 <= nums[i] <= 109
-109 <= target <= 109

2 <= k <= 100
输入描述：第一行是 nums 取值：2 7 11 15
第二行是 k 的取值：2
第三行是 target 取值： 9
输出描述：输出第一行是符合条件的元祖个数： 1
补充说明：[2,7]满足，输出个数是 1

```
示例

示例1
输入：-1 0 1 2 -1 -4
      3
      0
输出：2
说明：-1 0 1,-1 -12 满足条件

示例2
输入：2 7 11 15
      2
      9
输出：1
说明：2 7 符合条件
```

3、ACM 编程 字符串化繁为简

题目描述： 给定一个输入字符串，字符串只可能由英文字母（' a ' ~ ' z '、'A'~'Z'）和左右小括号（'('、')'）组成。当字符串里存在小括号时，小括号是成对的，可以有一
个或多个小括号对，小括号对不会嵌套，小括号对内可以包含 1 个或多个英文字母，也可以不包含英文字母。当小括号对内包含多个英文字母时，这些字母之间
是相互等效的关系，而且等效关系可以在不同的小括号对之间传递，即当存在' a '和' b '等效和存在' b '和' c '等效时，' a '和' c '也等效，另外，同一个英文字母的
大写字母和小写字母也相互等效（即使它们分布在不同的括号对里）
需要对这个输入字符串做简化，输出一个新的字符串，输出字符串里只需保留输入字符串里的没有被小括号对包含的字符 （按照输入字符串里的字符顺序），
并将每个字符替换为在小括号对里包含的且字典序最小的等效字符。
如果简化后的字符串为空，请输出为"0"。
示例：
输入字符串为"never(dont)give(run)up(f)()"，初始等效字符集合为('d', 'o', 'n', 't')、('r', 'u', 'n')，由于等效关系可以传递，因此最终等效字符集合为('d', 'o',
'n', 't', 'r', 'u')，将输入字符串里的剩余部分按字典序最小的等效字符替换后得到"devedgivedp"
输入描述：input_string
输入为 1 行，代表输入字符串
输出描述：output_string
输出为 1 行，代表输出字符串
补充说明：输入字符串的长度在 1~100000 之间

示例 展开

示例1
输入：()abd
输出：abd
说明：输入字符串里没有被小括号包含的子字符串为"abd"，其中每个字符没有等效字符，输出为"abd"

示例2
输入：(abd)demand(fb)()for
输出：aemanaaor
说明：等效字符集为('a', 'b', 'd', 'f')，输入字符串里没有被小括号包含的子字符串集合为"demandfor"，将其中字符替换为字典序最小的等效字符后输出为："aemanaaor"

示例3
输入：()happy(xyz)new(wxy)year(t)
输出：happwnewwear
说明：等效字符集为('x', 'y', 'z', 'w')，输入字符串里没有被小括号包含的子字符串集合为"happynewyear"，将其中字符替换为字典序最小的等效字符后输出为："happwnewwear"

示例4
输入：()abcdefgAC(a)(Ab)(C)
输出：AAcdefgAC
说明：等效字符集为('a', 'A', 'b')，输入字符串里没有被小括号包含的子字符串集合为"abcdefgAC"，将其中字符替换为字典序最小的等效字符后输出为："AAcdefgAC"

1、ACM 编程 报文回路

题目描述：IGMP 协议中响应报文和查询报文， 是维系组播通路的两个重要报文， 在一条已经建立的组播通路中两个相邻的 HOST 和 ROUTER， ROUTER 会给 HOST 发送查询报文， HOST 收到查询报文后给 ROUTER 回复一个响应报文， 以维持相互之间互通的关系， 一旦这个关系断裂，那么这条组播通路就“异常”了。现通过某种手段， 抓取到了 HOST 和 ROUTER 两者通讯的所有响应报文和查询报文，请分析该组播通路是否“正常”

输入描述：第一行抓到的报文数量 C（C<=100）， 后续 C 行依次输入设备节点 D1 和 D2，表示从 D1 到 D2 发送了单向的报文，D1 和 D2 用空格隔开。

输出描述：组播通路是否“正常”，正常输出 True， 异常输出 False。

示例

示例1
输入：5
1 2
2 3
3 2
1 2
2 1
输出：True
说明：

示例2
输入：3
1 3
3 2
2 3
输出：False

2、ACM 编程 分割数组的最大差值

题目描述：给定一个由若干整数组成的数组 nums ，可以在数组内的任意位置进行分割，将该数组分割成两个非空子数组（即左数组和右数组），分别对子数组求和得到两个值，计算这两个值的差值，请输出所有分割方案中，差值最大的值。

输入描述：第一行输入数组中元素个数 n，1 < n <= 100000

第二行输入数字序列，以空格进行分隔，数字取值为 4 字节整数

输出描述：输出差值的最大取值

示例

示例1
输入：6
1 -2 3 4 -9 7
输出：10
说明：将数组 nums 划分为两个非空数组的可行方案有：
左数组 = [1] 且 右数组 = [-2,3,4,-9,7]，和的差值 = | 1 - 3 | = 2
左数组 = [1,-2] 且 右数组 = [3,4,-9,7]，和的差值 = |-1 - 5 | = 6
左数组 = [1,-2,3] 且 右数组 = [4,-9,7]，和的差值 = | 2 - 2 | = 0
左数组 = [1,-2,3,4] 且 右数组 = [-9,7]，和的差值 = | 6 - (-2)| = 8
左数组 = [1,-2,3,4,-9] 且 右数组 = [7]，和的差值 = |-3 - 7 | = 10
最大的差值为10

3、ACM 编程 找出两个整数数组中同时出现的整数

题目描述：现有两个整数数组，需要你找出两个数组中同时出现的整数，并按照如下要求输出：

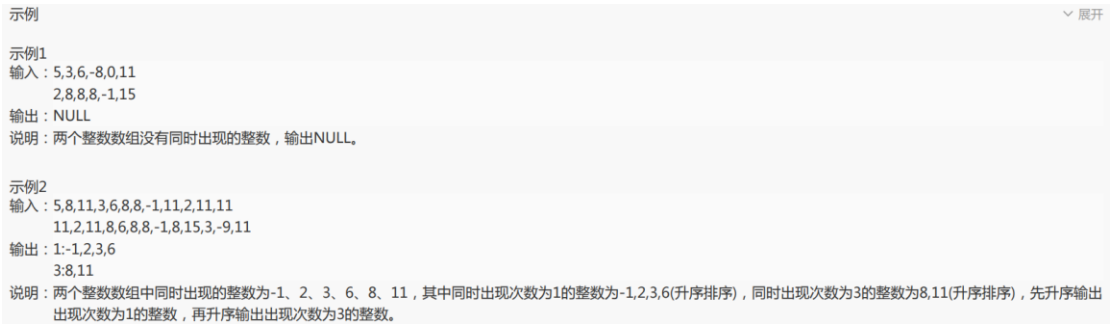
1、有同时出现的整数时，先按照同时出现次数（整数在两个数组中都出现并且出现次数较少的那个）进行归类，然后按照出现次数从小到大依次按行输出。

2、没有同时出现的整数时，输出 NULL。

输入描述：第一行为第一个整数数组，第二行为第二个整数数组，每行数据中整数与整数之间以英文逗号分隔，整数的取值范围为[-200,200]，数组长度的范围为[1,10000]之间的整数。

输出描述：按照出现次数从小到大依次按行输出，每行输出的格式为:出现次数:该出现次数下的整数升序排序的结果。

格式中的":"为英文冒号，整数间以英文逗号分隔。



1、ACM 编程 恢复数字序列

题目描述：对于一个连续正整数组成的序列，可以将其拼接成一个字符串，再将字符串里的部分字符打乱顺序。如序列 8 9 10 11 12，拼接成的字符串为 89101112，打乱一部分字符后得到 90811211。注意打乱后原来的正整数可能被拆开，比如在 90811211 中，原来的正整数 10 就被拆成了 0 和 1。

现给定一个按如上规则得到的打乱了字符的字符串，请将其还原成连续正整数序列，并输出序列中最小的数字。

输入描述：输入一行，为打乱字符的字符串和正整数序列的长度，两者间用空格分隔，字符串长度不超过 200，正整数不超过 1000，保证输入可以还原成唯一序列。

输出描述：输出一个数字，为序列中最小的数字。

示例

示例1
输入：19801211 5
输出：8
说明：还原出的序列为8 9 10 11 12，故输出8

示例2
输入：432111111111 4
输出：111
说明：还原出的序列为111 112 113 114，故输出111

2、完善核心代码编程 求幸存数之和

题目描述：给一个正整数列 nums，一个跳数 jump，及幸存数量 left。运算过程为：从索引为 0 的位置开始向后跳，中间跳过 J 个数字，命中索引为 J+1 的数字，该数被敲出，并从该点起跳，以此类推，直到幸存 left 个数为止。然后返回幸存数之和。

约束：

1）0 是第一个起跳点。

2）起跳点和命中点之间间隔 jump 个数字，已被敲出的数字不计入在内。

3）跳到末尾时无缝从头开始（循环查找），并可以多次循环。

4）若起始时 left>len(nums) 则无需跳数处理过程。

```
/**
* nums: 正整数数列，长度范围 [1,10000]
* jump: 跳数，范围 [1,10000]
* left: 幸存数量，范围 [0,10000]
* return: 幸存数之和
*/
int sumOfLeft(int[] nums,int jump,int left)
```

示例

示例1
输入：[1,2,3,4,5,6,7,8,9],4,3
输出：13
说明：从1（索引为0）开始起跳,中间跳过 4 个数字,因此依次删除 6,2,8,5,4,7 。 剩余 1,3,9,返回和为13

3、ACM 编程 最长子字符串的长度（二）

题目描述：给你一个字符串 s，字符串 s 首尾相连成一个环形 ，请你在环中找出'l'、'o'、'x'字符都恰好出现了偶数次最长子字符串的长度。

输入描述：输入是一串小写的字母组成的字符串。

输出描述：输出是一个整数

补充说明：1 <= s.length <= 5 x 10^5

s 只包含小写英文字母。

示例

示例1
输入：alolobo
输出：6
说明：最长子字符串之一是 "alolob"，它包含 'l'，'o'各 2 个，以及 0 个 'x' 。

示例2
输入：looxdolx
输出：7
说明：最长子字符串是 "oxdolxl"，由于是首尾连接在一起的，所以最后一个 'x' 和开头的 'l'是连接在一起的，此字符串包含 2 个 'l' ，2个 'o' ，2个 'x' 。

示例3
输入：bcbcbc
输出：6
说明：这个示例中，字符串 "bcbcbc" 本身就是最长的，因为 'l'、'o'、'x' 都出现了 0 次。

1、ACM 编程 关联子串

题目描述：给定两个字符串 str1 和 str2，如果字符串 str1 中的字符，经过排列组合后的字符串中，只要有一个字符串是 str2 的子串，则认为 str1 是 str2 的关联子串。
若 str1 是 str2 的关联子串，请返回子串在 str2 的起始位置；
若不是关联子串，则返回-1。
示例 1：
输入：str1="abc",str2="efghicabiii"
输出：5
解释：str2 包含 str1 的一种排列组合（"cab"），此组合在 str2 的字符串起始位置为 5（从 0 开始计数）
示例 2：str1="abc",str2="efghicaibii"
输出：-1。
预制条件：
1. 输入的字符串只包含小写字母；
2. 两个字符串的长度范围[1, 100,000]之间
3. 若 str2 中有多个 str1 的组合子串，请返回第一个子串的起始位置。
输入描述：输入两个字符串，分别为题目中描述的 str1、str2。
输出描述：如果 str1 是 str2 的关联子串，则返回子串在 str2 中的起始位置。
如果 str1 不是 str2 的关联子串，则返回-1。
若 str2 中有多个 str1 的组合子串，请返回最小的起始位置。
补充说明：输入的字符串只包含小写字母；
两个字符串的长度范围[1, 100,000]之间

示例

示例1
输入：abc efghicabiii
输出：5
说明：str2包含str1的一种排列组合（"cab"），此组合在str2的字符串起始位置为5（从0开始计数）

示例2
输入：abc efghicaibii
输出：-1
说明："abc"字符串中三个字母的各种组合（abc、acb、bac、bca、cab、cba），str2中均不包含，因此返回-1

2、ACM 编程 机场航班调度程序

题目描述：XX 市机场停放了多架飞机，每架飞机都有自己的航班号 CA3385，CZ6678，SC6508 等，航班号的前 2 个大写字母(或数字）代表航空公司的缩写，后面 4 个数字
代表航班信息。但是 XX 市机场只有一条起飞用跑道，调度人员需要安排目前停留在机场的

航班有序起飞。为保障航班的有序起飞，调度员首先按照航空公司的缩
写（航班号前 2 个字母）对所有航班进行排序，同一航空公司的航班再按照航班号的后 4 个数字进行排序最终获得安排好的航班的起飞顺序。请编写一段代码根据
输入的航班号信息帮助调度员输出航班的起飞顺序。
说明：
1、航空公司缩写排序按照从特殊符号$ & *, 0~9，A~Z 排序；
输入描述：第一行输入航班信息，多个航班号之间用逗号（“，”）分隔，输入的航班号不超过 100 个例如：
CA3385,CZ6678,SC6508,DU7523,HK4456,MK0987
备注：航班号为 6 位长度，后 4 位为纯数字，不考虑存在后 4 位重复的场景
输出描述：CA3385,CZ6678,DU7523,HK4456,MK0987,SC6508

示例

示例1
输入：CA3385,CZ6678,SC6508,DU7523,HK4456,MK0987
输出：CA3385,CZ6678,DU7523,HK4456,MK0987,SC6508
说明：输入目前停留在该机场的航班号，输出为按照调度排序后输出的有序的航班号

示例2
输入：MU1087,CA9908,3U0045,FM1703
输出：3U0045,CA9908,FM1703,MU1087
说明：

3、ACM 编程 5G 网络建设

题目描述：现需要在某城市进行 5G 网络建设，已经选取 N 个地点设置 5G 基站，编号固定为 1 到 N，接下来需要各个基站之间使用光纤进行连接以确保基站能互联互通，不同
基站之间架设光纤的成本各不相同，且有些节点之间已经存在光纤相连，请你设计算法，计算出能联通这些基站的最小成本是多少。
注意：基站的联通具有传递性，入基站 A 与基站 B 架设了光纤，基站 B 与基站 C 也架设了光纤，则基站 A 与基站 C 视为可以互相联通
输入描述：第一行输入表示基站的个数 N，其中 0<N<=20
第二行输入表示具备光纤直连条件的基站对的数目 M，其中 0<M<N*(N-1)/2
从第三行开始连续输入 M 行数据，格式为 X Y Z P，其中 X Y 表示基站的编号，0<X<=N，0<Y<=N 且 x 不等于 y，Z 表示在 X Y 之间架设光纤的成本，其中
0<Z<100，P 表示是否已存在光纤连接，0 表示未连接，1 表示已连接
输出描述：如果给定条件，可以建设成功互联互通的 5G 网络，则输出最小的建设成本；
如果给定条件，无法建设成功互联互通的 5G 网络，则输出-1

```
示例

示例1
输入：3
      3
      1 2 3 0
      1 3 1 0
      2 3 5 0
输出：4
说明：只需要在1,2以及2,3基站之间铺设光纤，其成本为3+1=4

示例2
输入：3
      1
      1 2 5 0
输出：-1
说明：3基站无法与其他基站连接，输出-1

示例3
```

```
输入：3
      3
      1 2 3 0
      1 3 1 0
      2 3 5 1
输出：1
说明：2,3基站已有光纤相连，只有要再1,3站点之间铺设光纤，其成本为1
```

1、ACM 编程 日志排序

题目描述：运维工程师采集到某产品现网运行一天产生的日志 N 条，现需根据日志时间按时间先后顺序对日志进行排序。

日志时间格式为：

H:M:S.N

H 表示小时(0-23)，M 表示分钟(0-59)，S 表示秒(0-59)，N 表示毫秒(0-999)

时间可能并没有补齐，也就是说: 01:01:01.001，也可能表示为 1:1:1.1

输入描述：第一行输入一个整数 N，表示日志条数，1<=N<=100000

接下来 N 行输入 N 个时间

输出描述：按时间升序排序之后的时间

如果有两个时间表示的时间相同，则保持输入顺序

```
示例

示例1
输入：2
      01:41:8.9
      1:1:09.211
输出：1:1:09.211
      01:41:8.9
说明：

示例2
输入：3
      23:41:08.023
      1:1:09.211
      08:01:22.0
输出：1:1:09.211
      08:01:22.0
      23:41:08.023
说明：

示例3
输入：2
      22:41:08.023
```

```
      22:41:08.23
输出：22:41:08.023
      22:41:08.23
说明：两个时间表示的时间相同，保持输入顺序
```

2、ACM 编程 查找接口成功率最优时间段

题目描述：服务之间交换的接口成功率作为服务调用关键质量特性，某个时间段内的接口失败率使用一个数组表示，数组中每个元素都是单位时间内失败率数值，数组中的数值为0~100的整数，给定一个数值(minAverageLost)表示某个时间段内平均失败率容忍值，即平均失败率小于等于 minAverageLost，找出数组中最长时间段，如果未找到则直接返回 NULL。

输入描述：输入有两行内容，第一行为{minAverageLost}，第二行为{数组}，数组元素通过空格(" ")分隔，minAverageLost 及数组中元素取值范围为 0~100 的整数，数组元素的个数不会超过 100 个。

输出描述：找出平均值小于等于 minAverageLost 的最长时间段，输出数组下标对，格式{beginIndex}-{endIndx}(下标从 0 开始)，如果同时存在多个最长时间段，则输出多个下标对且下标对之间使用空格(" ")拼接，多个下标对按下标从小到大排序。

示例 展开

示例1
输入：1
0 1 2 3 4
输出：0-2
说明：A、输入解释：minAverageLost=1，数组[0, 1, 2, 3, 4]
B、前3个元素的平均值为1，因此数组第一个至第三个数组下标，即0-2

示例2
输入：2
0 0 100 2 2 99 0 2
输出：0-1 3-4 6-7
说明：A、输入解释：minAverageLost=2，数组[0, 0, 100, 2, 2, 99, 0, 2]
B、通过计算小于等于2的最长时间段为：数组下标为0-1即[0, 0]，数组下标为3-4即[2, 2]，数组下标为6-7即[0, 2]，这三个部分都满足平均值小于等2的要求，因此输出0-1 3-4 6-7

3、ACM 编程 亲子游戏

题目描述：宝宝和妈妈参加亲子游戏，在一个二维矩阵（N*N）的格子地图上，宝宝和妈妈抽签决定各自的位置，地图上每个格子有不同的糖果数量，部分格子有障碍物。

游戏规则是妈妈必须在最短的时间（每个单位时间只能走一步）到达宝宝的位置，路上的所有糖果都可以拿走，不能走障碍物的格子，只能上下左右走。

请问妈妈在最短到达宝宝位置的时间内最多拿到多少糖果(优先考虑最短时间到达的情况下尽可能多拿糖果)。

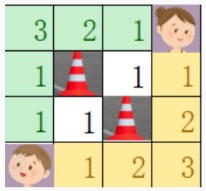
输入描述：第一行输入为 N，N 标识二维矩阵的大小
之后 N 行，每行有 N 个值，表格矩阵每个位置的值
其中：
-3：妈妈
-2：宝宝
-1：障碍
>=0：糖果数(0 表示没有糖果，但是可以走

输出描述：输出妈妈在最短到达宝宝位置的时间内最多拿到多少糖果，行末无多余空格

补充说明：地图最大 50*50

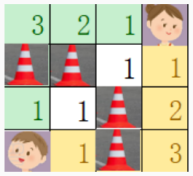
示例

示例1
输入：4
3 2 1 -3
1 -1 1 1
1 1 -1 2
-2 1 2 3
输出：9
说明：



此地图有两条最短路径可到宝宝位置，绿色线和黄色线都是最短路径6步，但黄色拿到的糖果更多，9个

示例2
输入：4
3 2 1 -3
-1 -1 1 1
1 1 -1 2
-2 1 -1 3
输出：-1
说明：



此地图妈妈无法到达宝宝位置

1、ACM 编程 数列描述

题目描述：有一个数列 a[N] (N=60)，从 a[0]开始，每一项都是一个数字。数列中 a[n+1]都是 a[n]的描述。其中 a[0]=1。

规则如下：

a[0]:1

a[1]:11(含义：其前一项 a[0]=1 是 1 个 1，即“11”。表示 a[0]从左到右，连续出现了 1 次“1”)

a[2]:21(含义：其前一项 a[1]=11，从左到右：是由两个 1 组成，即“21”。表示 a[1]从左到右，连续出现了两次“1”)

a[3]:1211(含义：其前一项 a[2]=21，从左到右：是由一个 2 和一个 1 组成，即“1211”。表示 a[2]从左到右，连续出现了 1 次“2”，然后又连续出现了 1 次“1”)

a[4]:111221(含义：其前一项 a[3]=1211，从左到右：是由一个 1、一个 2、两个 1 组成，即“111221”。表示 a[3]从左到右，连续出现了 1 次“1”，连续出现了

1 次“2”，连续出现了两次“1”)

请输出这个数列的第 n 项结果（a[n]，0≤n≤59）。

输入描述：数列的第 n 项(0≤n≤59)：

4

输出描述：数列的内容：
111221

示例

示例1
输入：4
输出：111221
说明：

2、ACM 编程 测试用例执行计划

题目描述：某个产品当前迭代周期内有 N 个特性（ ）需要进行覆盖测试，每个特性都被评估了对应的优先级，特性使用其 ID 作为下标进行标识。
设计了 M 个测试用例（ ），每个用例对应了一个覆盖特性的集合，测试用例使用其 ID 作为下标进行标识，测试用例的优先级定义为其覆
盖的特性的优先级之和。
在开展测试之前，需要制定测试用例的执行顺序，规则为：优先级大的用例先执行，如果存在优先级相同的用例，用例 ID 小的先执行。
输入描述：第一行输入为 N 和 M，N 表示特性的数量，M 表示测试用例的数量， 。
之后 N 行表示特性 ID=1 到特性 ID=N 的优先级。
再接下来 M 行表示测试用例 ID=1 到测试用例 ID=M 关联的特性的 ID 的列表。
输出描述：按照执行顺序（优先级从大到小）输出测试用例的 ID，每行一个 ID。
补充说明：测试用例覆盖的 ID 不重复。

示例

示例1
输入：5 4

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

输出：
```
3
4
1
2
```

说明：测试用例的优先级计算如下：

$$T_1 = P_{F1} + P_{F2} + P_{F3} = 1 + 1 + 2 = 4$$
$$T_2 = P_{F1} + P_{F4} = 1 + 3 = 4$$
$$T_3 = P_{F3} + P_{F4} + P_{F5} = 2 + 3 + 5 = 10$$
$$T_4 = P_{F2} + P_{F3} + P_{F4} = 1 + 2 + 3 = 6$$

按照优先级从小到大，以及相同优先级，ID小的先执行的规则，执行顺序为T3,T4,T1,T2

示例2
输入：3 3
3
1
5
1 2 3
1 2 3
1 2 3
输出：1
2
3
说明：测试用例的优先级计算如下：

$$T_1 = P_{F1} + P_{F2} + PF3 = 3 + 1 + 5 = 9$$
$$T_2 = P_{F1} + P_{F2} + PF3 = 3 + 1 + 5 = 9$$
$$T_3 = P_{F1} + P_{F2} + PF3 = 3 + 1 + 5 = 9$$

每个优先级一样，按照ID从小到大执行，执行顺序为T1,T2,T3

3、ACM 编程 贪心歌手

题目描述：一个歌手准备从 A 城去 B 城参加演出。

1) 按照合同，他必须在 T 天内赶到。

2) 歌手途径 N 座城市。

3) 歌手不能往回走。

4) 每两座城市之间需要的天数都可以提前获知。

5) 歌手在每座城市都可以在路边卖唱赚钱。经过调研，歌手提前获知了每座城市卖唱的收入预期:

如果在一座城市第一天卖唱可以赚 M，后续每天的收入会减少 D（第二天赚的钱是 M – D，第三天是 M-2D…）。如果收入减到 0 就不会再少了。

6) 歌手到达后的第二天才能开始卖唱。如果今天卖过唱，第二天才能出发。

贪心的歌手最多可以赚多少钱?

输入描述：第一行两个数字 T 和 N，中间用空格隔开。

T 代表总天数;

N 代表路上经过 N 座城市;

0 < T < 1000, 0 < N < 100

第二行 N+1 个数字，中间用空格隔开。

代表每两座城市之间耗费的时间。

其总和<=T。

接下来 N 行，每行两个数字 M 和 D，中间用空格隔开。

代表每个城市的收入预期。

0 < M < 1000, 0 < D < 100

输出描述：一个数字。代表歌手最多可以赚多少钱。以回车结束。

示例1
输入：10 2
1 1 2
120 20
90 10
输出：540
说明：总共10天，路上经过2座城市。
路上共花1+1+2=4天。
剩余6天最好的计划是在第一座城市待3天，在第二座城市待3天。
在第一座城市赚的钱：120+100+80= 300.
在第二座城市赚的钱：90+80+70 =240.
一共300 + 240 = 540。

1、ACM 编程 分割均衡字符串

题目描述：均衡串定义：字符串只包含两种字符，且两种字符的个数相同。
给定一个均衡字符串，请给出可分割成新的均衡子串的最大个数。
约定字符串中只包含大写的'X'和'Y'两种字符。
输入描述：均衡串：XXYYXY
字符串的长度[2,10000]。给定的字符串均为均衡串。
输出描述：可分割为两个子串：
XXYY
XY
补充说明：分割后的子串，是原字符串的连续子串。

```
示例

示例1
输入：XXYYXY
输出：2
说明：
```

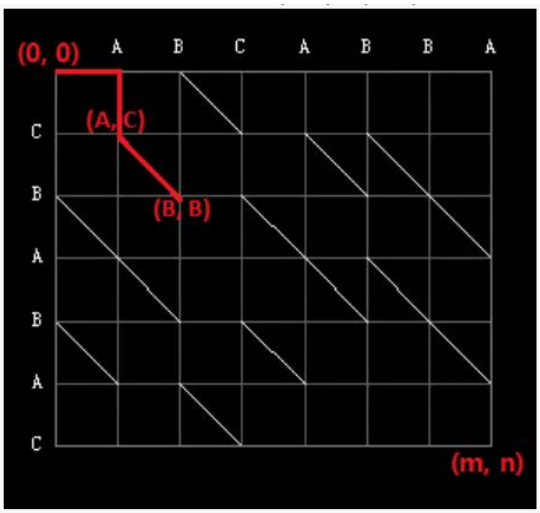
2、ACM 编程 密码解密

题目描述：给定一段"密文"字符串 s,其中字符都是经过"密码本"映射的，现需要将"密文"解密并且输出
映射的规则 ('a'-'i')分别用('1'-'9')表示；('j'-'z')分别用('10*'-'26*')表示
约束：映射始终唯一
输入描述："密文"字符串
输出描述：明文字符串
补充说明：翻译后的文本的长度在 100 以内

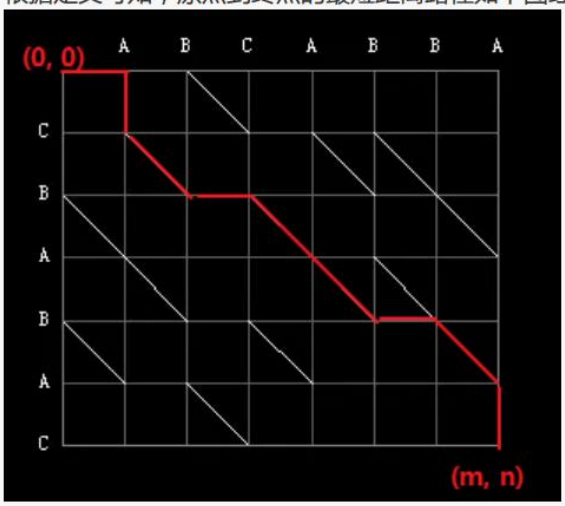
```
示例

示例1
输入：20*19*20*
输出：tst
说明：
```

3、ACM 编程 两个字符串间的最短路径问题

题目描述：给定两个字符串，分别为字符串 A 与字符串 B。例如 A 字符串为 ABCABBA，B 字符串为 CBABAC 可以得到下图 m*n 的二维数组，定义原点为(0, 0)，终点为(m, n)，
水平与垂直的每一条边距离为 1，映射成坐标系如下图。
从原点(0, 0)到(0, A)为水平边，距离为 1，从(0, A)到(A, C)为垂直边，距离为 1；假设两个字符串同一位置的两个字符相同则可以作一个斜边，如(A, C)到(B, B)最
短距离为斜边，距离同样为 1。
作出所有的斜边如下图，(0, 0)到(B, B)的距离为 1 个水平边 + 1 个垂直边 + 1 个斜边 = 3。



根据定义可知，原点到终点的最短距离路径如下图红线标记，最短距离为9：



输入描述：空格分割的两个字符串 A 与字符串 B，字符串不为“空串”，字符格式满足正则规则：[A-Z]，字符串长度 < 10000

输出描述：原点到终点的最短距离

示例

示例1
输入：ABC ABC
输出：3
说明：

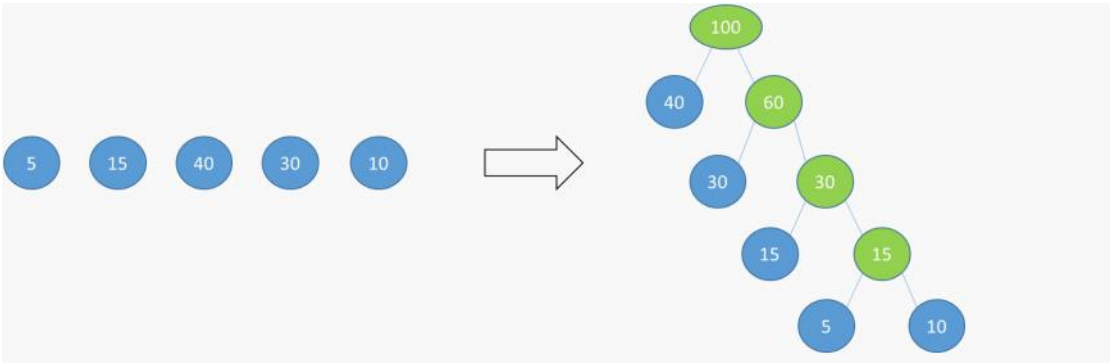
示例2
输入：ABCABBA CBABAC
输出：9
说明：

1、ACM 编程 生成哈夫曼树

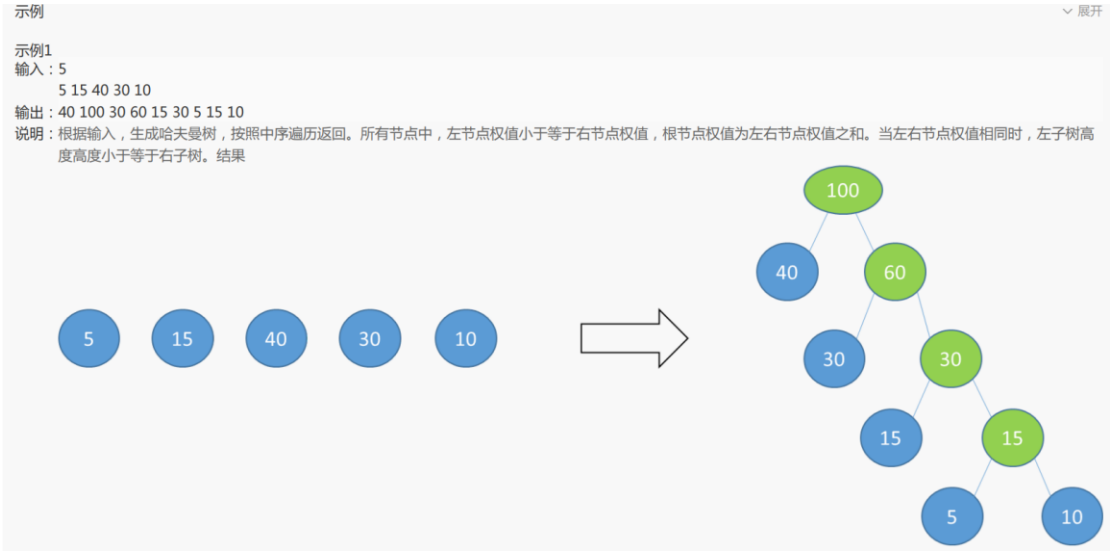
题目描述：给定长度为 n 的无序的数字数组，每个数字代表二叉树的叶子节点的权值，数字数组的值均大于等于 1。请完成一个函数，根据输入的数字数组，生成哈夫曼树，
并将哈夫曼树按照中序遍历输出。

为了保证输出的二叉树中序遍历结果统一，增加以下限制：二叉树节点中，左节点权值小于等于右节点权值，根节点权值为左右节点权值之和。当左右节点权值
相同时，左子树高度高度小于等于右子树。

注意：所有用例保证有效，并能生成哈夫曼树。
提醒：哈夫曼树又称最优二叉树，是一种带权路径长度最短的二叉树。所谓树的带权路径长度，就是树中所有的叶结点的权值乘上其到根结点的路径长度（若根
结点为 0 层，叶结点到根结点的路径长度为叶结点的层数）。
例如：
由叶子节点 5 15 40 30 10 生成的最优二叉树如下图所示，该树的最短带权路径长度为 40*1+30*2+15*3+5*4+10*4=205。



输入描述：第一行输入为数组长度，记为 N，1<=N<=1000，第二行输入无序数值数组，以空格分割，数值均大于等于 1，小于 100000
输出描述：输出一个哈夫曼树的中序遍历的数组，数值间以空格分割



2、ACM 编程 执行任务赚积分
题目描述：现有 N 个任务需要处理，同一时间只能处理一个任务，处理每个任务所需要的时间固定为 1。
每个任务都有最晚处理时间限制和积分值，在最晚处理时间点之前处理完成任务才可获得对应的积分奖励。
可用于处理任务的时间有限，请问在有限的时间内，可获得的最多积分。
输入描述：第一行为一个数 N，表示有 N 个任务，1<=N<=100
第二行为一个数 T，表示可用于处理任务的时间。1<=T<=100
接下来 N 行，每行两个空格分隔的整数(SLA 和 V)，SLA 表示任务的最晚处理时间，V 表示任务对应的积分。1<=SLA<=100, 0<=V<=100000

输出描述：可获得的最多积分

```
示例

示例1
输入：4
      3
      1 2
      1 3
      1 4
      1 5
输出：5
说明：虽然有3个单位的时间用于处理任务，可是所有任务在时刻1之后都无效。
      所以在第1个时间单位内，选择处理有5个积分的任务。1-3时无任务处理。


示例2
输入：4
      3
      1 2
      1 3
      1 4
      3 5
输出：9
说明：第1个时间单位内，处理任务3，获得4个积分
      第2个时间单位内，处理任务4，获得5个积分
      第3个时间单位内，无任务可处理
      共获得9个积分
```

3、ACM 编程 项目排期

题目描述：项目组共有 N 个开发人员，项目经理接到了 M 个独立的需求，每个需求的工作量不同，且每个需求只能由一个开发人员独立完成，不能多人合作。假定各个需求直接无任何先后依赖关系，请设计算法帮助项目经理进行工作安排，使整个项目能用最少的时间交付。

输入描述：第一行输入为 M 个需求的工作量，单位为天，用逗号隔开。

例如：X X X X

表示共有 M 个需求，每个需求的工作量分别为 X 天，X 天.....X 天。其中 0<M<30；0<X<200

第二行输入为项目组人员数量 N

例如：

5

表示共有 5 名员工，其中 0<N<10

输出描述：最快完成所有工作的天数

例如：

25

表示最短需要 25 天能完成所有工作

```
示例

示例1
输入：6 2 7 7 9 3 2 1 3 11 4
      2
输出：28
说明：共有两位员工，其中一位分配需求 6 2 7 7 3 2 1共需要28天完成，另一位分配需求 9 3 11 4 共需要27天完成，故完成所有工作至少需要28天。
```

1、ACM 编程 虚拟游戏理财

题目描述：在一款虚拟游戏中生活，你必须进行投资以增强在虚拟游戏中的资产以免被淘汰出局。现有一家 Bank，它提供有若干理财产品 m，风险及投资回报不同，你有 N（元）进行投资，能接受的总风险值为 X。

你要在可接受范围内选择最优的投资方式获得最大回报。

说明：
在虚拟游戏中，每项投资风险值相加为总风险值；
在虚拟游戏中，最多只能投资 2 个理财产品；
在虚拟游戏中，最小单位为整数，不能拆分为小数；
投资额*回报率=投资回报
输入描述：第一行：产品数(取值范围[1, 20])，总投资额(整数，取值范围[1, 10000])，可接受的总风险(整数，取值范围[1, 200])
第二行：产品投资回报率序列，输入为整数，取值范围[1,60]
第三行：产品风险值序列，输入为整数，取值范围[1,100]
第四行：最大投资额度序列，输入为整数，取值范围[1,10000
输出描述：每个产品的投资额序列
补充说明：在虚拟游戏中，每项投资风险值相加为总风险值；
在虚拟游戏中，最多只能投资 2 个理财产品；
在虚拟游戏中，最小单位为整数，不能拆分为小数；
投资额*回报率=投资回报

示例

示例1
输入：5 100 10
10 20 30 40 50
3 4 5 6 10
20 30 20 40 30
输出：0 30 0 40 0
说明：投资第二项30个单位，第四项40个单位，总的投资风险为两项相加为4+6=10

2、ACM 编程 开源项目热榜

题目描述：某个开源社区希望将最近热度比较高的开源项目出一个榜单，推荐给社区里面的开发者。对于每个开源项目，开发者可以进行关注(watch)、收藏(star)、fork、
提 issue、提交合并请求(MR)等。
数据库里面统计了每个开源项目关注、收藏、fork、issue、MR 的数量，开源项目的热度根据这 5 个维度的加权求和进行排序。

$$H = W_{watch} \times \#watch + W_{star} \times \#star + W_{fork} \times \#fork + W_{issue} \times \#issue + W_{mr} \times \#mr$$

H表示热度值，W_{watch}、W_{star}、W_{fork}、W_{issue}、W_{mr}分别表示5个统计维度的权重，
$\#watch$、$\#star$、$\#fork$、$\#issue$、$\#mr$分别表示5个统计维度的统计值。
榜单按照热度值降序排序，对于热度值相等的，按照项目名字转换为全小写字母后的字典序排序（'a','b','c',...,'x','y','z')。

输入描述：第一行输入为 N，表示开源项目的个数，$0 < N \leq 100$。
第二行输入为权重值列表，一共 5 个整型值，分别对应关注、收藏、fork、issue、MR 的权重，权重取值$0 < W \leq 50$。
第三行开始接下来的 N 行为开源项目的统计维度，每一行的格式为：
name nr_watch nr_star nr_fork nr_issue nr_mr
其中 name 为开源项目的名字，由英文字母组成，长度≤ 50，其余 5 个整型值分别为该开源项目关注、收藏、fork、issue、MR 的数量，数量取值$0 < nr \leq 1000$。
输出描述：按照热度降序，输出开源项目的名字，对于热度值相等的，按照项目名字转换为全小写字母后的字典序排序（'a'>'b'>'c'>...>'x'>'y'>'z')。

```
示例

示例1
输入：4
        8 6 2 8 6
        camila 66 70 46 158 80
        victoria 94 76 86 189 211
        anthony 29 17 83 21 48
        emily 53 97 1 19 218
输出：victoria
        camila
        emily
        anthony
说明：排序热度值计算：
        camila: 66*8 + 70*6 + 46*2 + 158*8 + 80*6 = 2784
        victoria: 94*8 + 76*6 + 86*2 + 189*8 + 211*6 = 4158
        anthony: 29*8 + 17*6 + 83*2 + 21*8 + 48*6 = 956
        emily: 53*8 + 97*6 + 1*2 + 19*8 + 218*6 = 2468
        根据热度值降序，得到结果。
```

```
示例2
输入：5
        5 6 6 1 2
        camila 13 88 46 26 169
        grace 64 38 87 23 103
        lucas 91 79 98 154 79
        leo 29 27 36 43 178
        ava 29 27 36 43 178
输出：lucas
        grace
        camila
        ava
        leo
说明：排序热度值计算：
        camila: 13*5 + 88*6 + 46*6 + 26*1 + 169*2 = 1233
        grace: 64*5 + 38*6 + 87*6 + 23*1 + 103*2 = 1299
        lucas: 91*5 + 79*6 + 98*6 + 154*1 + 79*2 = 1829
        leo: 29*5 + 27*6 + 36*6 + 43*1 + 178*2 = 922
        ava: 29*5 + 27*6 + 36*6 + 43*1 + 178*2 = 922
        根据热度值降序，对于leo和ava，热度值相等，按照字典序，ava排在leo前面，得到结果。
```

3、ACM 编程 简易内存池

题目描述：请实现一个简易内存池,根据请求命令完成内存分配和释放。

内存池支持两种操作命令，REQUEST 和 RELEASE，其格式为：

REQUEST=请求的内存大小 表示请求分配指定大小内存，如果分配成功，返回分配到的内存首地址；如果内存不足，或指定的大小为 0，则输出 error。

RELEASE=释放的内存首地址 表示释放掉之前分配的内存，释放成功无需输出，如果释放不存在的首地址则输出 error。

注意：

1.内存池总大小为 100 字节。

2.内存池地址分配必须是连续内存，并优先从低地址分配。

3.内存释放后可被再次分配，已释放的内存在空闲时不能被二次释放。

4.不会释放已申请的内存块的中间地址。

5.释放操作只是针对首地址所对应的单个内存块进行操作，不会影响其它内存块。

输入描述：· 首行为整数 N ,表示操作命令的个数，取值范围：0 < N <= 100。

接下来的 N 行,每行将给出一个操作命令，操作命令和参数之间用 “=”分割。

输出描述：见题面输出要求

```
示例1
输入：2
        REQUEST=10
        REQUEST=20
输出：0
        10
说明：
```

```
示例2
输入：5
        REQUEST=10
        REQUEST=20
        RELEASE=0
        REQUEST=20
        REQUEST=10
输出：0
        10
        30
        0
说明：第一条指令，申请地址0~9的10个字节内存，返回首地址0
        第二条指令，申请地址10~29的20字节内存，返回首地址10
        第三条指令，释放首地址为0的内存申请，0~9地址内存被释放，变为空闲，释放成功，无需输出
        第四条指令，申请20字节内存，0~9地址内存连续空间不足20字节，往后查找到30~49地址，返回首地址30
        第五条指令，申请10字节，0~9地址内存空间足够，返回首地址0
```

1、ACM 编程 万能字符单词拼写

题目描述：有一个字符串数组 words 和一个字符串 chars。

假如可以用 chars 中的字母拼写出 words 中的某个“单词”（字符串），那么我们就认为你掌握了这个单词。

words 的字符仅由 a-z 英文小写字母组成。 例如: abc

chars 由 a-z 英文小写字母和 “?”组成。其中英文问号“?”表示万能字符，能够在拼写时当做任意一个英文字母。 例如： "?" 可以当做 "a"等字母。

注意：每次拼写时，chars 中的每个字母和万能字符都只能使用一次。

输出词汇表 words 中你掌握的所有单词的个数。 没有掌握任何单词，则输出 0。

输入描述：第 1 行输入数组 words 的个数，记为 N。

从第 2 行开始到第 N+1 行依次输入数组 words 的每个字符串元素。

第 N+2 行输入字符串 chars。

输出描述：输出一个整数，表示词汇表 words 中你掌握的单词个数。

补充说明：注意：

1 <= words.length <= 100

1 <= words[i].length, chars.length <= 100

所有字符串中都仅包含小写英文字母、英文问号

```
示例

示例1
输入：4
        cat
        bt
        hat
        tree
        atach??
输出：3
说明：可以拼写字符串"cat"、"bt"和"hat"

示例2
输入：3
        hello
        world
        cloud
        welldonehoneyr
输出：2
说明：可以拼写字符串"hello"和"world"

示例3
输入：3
        apple
        car
        window
        welldoneapplec?
输出：2
说明：可以拼写字符串"apple"和"car"
```

2、ACM 编程 素数之积

题目描述：RSA 加密算法在网络安全世界中无处不在，它利用了极大整数因数分解的困难度，数据越大，安全系数越高，给定一个 32 位正整数，请对其进行因数分解，找出
是哪两个素数的乘积。
输入描述：一个正整数 num
0 < num <= 2147483647
输出描述：如果成功找到，以单个空格分割，从小到大输出两个素数，分解失败，请输出-1 -1

```
示例

示例1
输入：15
输出：3 5
说明：因数分解后，找到两个素数3和5，使得3*5=15，按从小到大排列后，输出3 5

示例2
输入：27
输出：-1 -1
说明：通过因数分解，找不到任何素数，使得他们的乘积为27，输出-1 -1
```

3、ACM 编程 员工派遣

题目描述：某公司部门需要派遣员工去国外做项目。现在，代号为 x 的国家和代号为 y 的国家分别需要 cntx 名和 cnty 名员工。部门每个员工有一个员工号(1,2,3……)，工号连
续，从 1 开始。
部长派遣员工的规则：
规则 1、从[1,k]中选择员工派遣出去
规则 2、编号为 x 的倍数的员工不能去 x 国，编号为 y 的倍数的员工不能去 y 国
问题：
找到最小的 k，使得可以将编号在[1,k]中的员工分配给 x 国和 y 国，且满足 x 国和 y 国的需求
输入描述：四个整数 x, y, cntx, cnty。(2<=x<y<=30000; x 和 y 一定是质数;1<=cntx,cnty<10^9; cntx+cnty<=10^9)
输出描述：满足条件的最小的 k。

```
示例

示例1
输入：2 3 3 1
输出：5
说明：输入说明：
        2 -表示国家代号2
        3 -表示国家代号3
        3 -表示国家2需要3个人
        1 -表示国家3需要1个人
```