



2026 TGU-ACM 程序设计新生赛

试题册

天津工业大学 ACM 集训队

2026 年 10 月 11 日

试题列表

A	An ACMer's Life from Zero
B	Bring Your Partner
C	Constructive Problem, Again
D	Do You Want to Start a Band?
E	Evening of the Starlight Festival
F	For Stalin, Let's Sort!
G	Grand Finale of the Witch Trial
H	How Hard Can Math Be?
I	It's Time for hrz to Sleep
J	Journey of Liyu Beneath the Sea
K	Keep RiNG in Order
L	Look, It's Yesterday Once More

本试题册共 12 题，17 页。

如果您的试题册缺少页面，请立即通知志愿者。

A. An ACMer’s Life from Zero

时间限制：1 秒

内存限制：128 MiB

输入方式：标准输入

输出方式：标准输出

题目背景



Fun Fact：以上画面出自 2026 年 9 月播出的 Re：从零开始的异世界生活 S4E17。

剑圣 6154 正在准备与雷德开战。在他们战前嘴炮的时候，6154 忽然听到了远处传来了一声神秘的呼喊：“天津工业大学 ACM 社团正在举办新生赛！”，他想，可是我觉得 ACM 很神圣啊，于是就想赶忙跑到软件园 C205 参赛。

并且为了告知 ACM 社团的负责同学，他一边跑一边大喊：I want to be an acmer!!，然而，由于他跑得实在太快，再加上喊声被呼啸的风吹得断断续续，ACM 社团的同学并没有听清他说了什么。

于是，他们找到了正在参加新生赛的你，希望你能够准确地复述出 6154 刚才喊出的那句话。

作为一名未来的 ACMer，相信这对你来说一定不是问题。

题目描述

请输出下面这句话。

I want to be an acmer!!

请注意保留字母大小写、单词间的空格和末尾的两个感叹号。

输入格式

本题没有输入。

输出格式

输出一行 I want to be an acmer!!。

样例

样例输入 1	样例输出 1
(无输入)	I want to be an acmer!!

B. Bring Your Partner

时间限制：1 秒

内存限制：128 MiB

输入方式：标准输入

输出方式：标准输出

题目背景



选择一个伙伴，帮你度过今晚。

由于 Hailuo 看过很多的奶龙视频了，他知道奶蛋是一个轮椅角色，只要有奶蛋出现，他一定会选择奶蛋。否则，他会从所有伙伴里选择字典序最小的那个。

现在，有 n 个伙伴站在 Hailuo 的面前，他的头脑逐渐变得混沌，因此想请你来根据他的规则，帮他选择伙伴。

题目描述

现在给出 n 个名字，你要从中选出一位伙伴。选择规则如下：

1. 如果这些名字中出现了 `naidan`，一定选择 `naidan`。
2. 否则，选择出现过的名字中字典序最小的一个。

名字可能重复出现，且不保证测试用例的名字种类与样例相同。

输入格式

第一行包含一个整数 n ，表示给出的名字数量。

接下来 n 行，每行输入一个字符串，表示一个名字。

输出格式

输出一行，为选中的名字。

数据范围

- $1 \leq n \leq 10^5$ 。
- 每个名字的长度在 1 到 30 之间，只包含小写英文字母 `a` 到 `z`。
- 同一个名字可以出现多次。

B. Bring Your Partner（续）

样例

样例输入 1	样例输出 1
5 naishi naigou naidan feidudu niulai	naidan

样例输入 2	样例输出 2
4 niulai naishu feidudu naigou	feidudu

样例输入 3	样例输出 3
3 gagadilaxia hola andi	andi

样例解释

样例 1 中出现了 `naidan`,因此必须选择它。样例 2 中没有 `naidan`,出现过的名字中字典序最小的是 `feidudu`。

字典序说明

字典序可以理解为“像查字典一样比较两个字符串”。

比较两个字符串时，从左到右依次比较对应位置上的字符：

- 如果第一次出现不同的字符，那么字符较小的字符串字典序更小。
- 如果一直比较到其中一个字符串结束，前面的字符都相同，那么较短的字符串字典序更小。

例如：

- `"abc"` 的字典序小于 `"abd"`，因为前两个字符相同，而第三个字符中 `'c' < 'd'`。
- `"abc"` 的字典序小于 `"abcd"`，因为 `"abc"` 是 `"abcd"` 的前缀，并且长度更短。
- `"apple"` 的字典序小于 `"banana"`，因为第一个字符就有 `'a' < 'b'`。

因此，对若干字符串按照字典序从小到大排序，与我们平时在字典中查找单词时的顺序类似。

C. Constructive Problem, Again

时间限制：2 秒

内存限制：256 MiB

输入方式：标准输入

输出方式：标准输出

题目描述

给定一个正整数 n ，请构造一个 n 行 n 列的整数矩阵 A ，满足以下所有条件：

- 每个元素都在 1 到 n 之间，即 $1 \leq A_{i,j} \leq n$ 。
- 同一行中的元素两两不同，即对任意 $1 \leq i \leq n$ 以及 $1 \leq j < k \leq n$ ，都有 $A_{i,j} \neq A_{i,k}$ 。
- 同一列中的元素两两不同，即对任意 $1 \leq j \leq n$ 以及 $1 \leq i < k \leq n$ ，都有 $A_{i,j} \neq A_{k,j}$ 。

不同行且不同列的两个元素可以相同。可以证明，对于数据范围内的每个 n ，都存在满足条件的矩阵。请输出任意一个满足条件的矩阵。

输入格式

输入一行，包含一个整数 n ，表示矩阵的行数和列数。

输出格式

输出 n 行，每行 n 个整数，表示你构造的矩阵。若有多种合法答案，输出任意一种即可。

数据范围

对于所有测试数据， $1 \leq n \leq 5000$ 。

样例

样例输入 1	样例输出 1
2	1 2 2 1

样例输入 2	样例输出 2
3	2 1 3 1 3 2 3 2 1

输入输出提示

本题输出量较大，建议使用较快的输入输出方式。
使用 C++ 时，可以在程序开头加入以下语句关闭同步流，关闭同步流后，注意不要混用 `cin`, `cout` 和 `scanf`, `printf`。

```
std::ios::sync_with_stdio(false);  
std::cin.tie(nullptr);
```

输出换行时使用 `'\n'`，而不是 `std::endl`，以免频繁刷新缓冲区。
使用 Python 时，请尽可能减少 `print()` 的调用次数，否则会超时。

D. Do You Want to Start a Band?

时间限制：2 秒

内存限制：256 MiB

输入方式：标准输入

输出方式：标准输出

题目背景

shuiyin 学长在高考完的暑假闲来无事开始看起了 MyGO。起初觉得无聊，但越看越觉得……好美味。

少女因为乐队相遇，在一次次误解、争执与磨合中逐渐产生羁绊，最终一起站上了舞台。看着五个人在聚光灯下尽情演奏，shuiyin 学长不禁感叹：“我要看一辈子乐队番口牙！”

为了方便理解乐队成员之间的羁绊，shuiyin 学长决定把少女们之间复杂的情感进行一番数学处理。毕竟，只要把情感变成数字，问题就会从青春群像剧变成一道算法题。

题目描述

shuiyin 学长喜欢上了一支乐队，乐队共有 n 名成员，第 i 名成员的演出值为 a_i 。

如果两位成员在舞台上相邻站立，那么会产生 $|a_i - a_j|$ 点违和感。所有成员可以任意改变站位顺序，重新排成一列，随后选择一个位置 i ，满足 $1 \leq i < n$ ，将队伍分成前后两个连续部分。**每个部分内部**相邻成员的违和感均不能超过 k ，两个部分之间则不受此限制。

shuiyin 学长想知道，是否存在一种站位方式，使得乐队能够登台演出。

特别地，当 $n = 1$ 时，不需要进行划分，直接认为满足条件。

输入格式

第一行包括两个整数 n, k ，分别表示成员数量以及允许的最大违和。

第二行有 n 个整数 a_i ，表示每个乐队成员的演出值 $a_1, a_2, \dots, a_n$ 。

输出格式

对于每个样例，输出一行：

- 如果存在一种合法排列方式，输出 YES；
- 否则，输出 NO。

数据范围

- $1 \leq n \leq 5 \times 10^5$ ；
- $0 \leq k \leq 10^9$ ；
- 对每个演出值 a_i ，保证每个演出值都是正整数： $1 \leq a_i \leq 10^9$ 。

样例

样例输入 1	样例输出 1
5 5 4 5 11 6 9	YES

样例输入 2	样例输出 2
5 1 4 5 11 6 9	NO

E. Evening of the Starlight Festival

时间限制：2 秒

内存限制：256 MiB

输入方式：标准输入

输出方式：标准输出

题目背景

一年一度的星灯节即将开始。城市中共有 n 盏星灯需要维修。由于星灯内部的能量正在逐渐耗尽，每盏星灯都有自己的最后维修期限。小鲤只有一套维修工具，因此在任意时刻，他都只能维修一盏星灯。现在，小鲤想知道：他最多能够成功维修多少盏星灯？

题目描述

共有 n 盏星灯，对于第 i 盏星灯：

- 维修它需要连续花费 t_i 分钟；
- 必须在第 d_i 分钟结束之前完成维修。

小鲤从第 0 分钟开始工作，并且在任意时刻只能维修一盏星灯。如果第 i 盏星灯的维修工作在第 d_i 分钟之前或恰好在第 d_i 分钟完成，那么这盏星灯就被成功维修。

小鲤可以自由决定维修哪些星灯，以及这些星灯的维修顺序。请你求出，小鲤最多能够成功维修多少盏星灯。

输入格式

第一行输入一个正整数 n ，表示星灯的数量。
接下来 n 行，每行输入两个正整数 t_i, d_i ，分别表示第 i 盏星灯需要的维修时间和最后维修期限。

输出格式

输出一个整数，表示小鲤最多能够成功维修的星灯数量。

数据范围

对于所有数据：
 $1 \leq n \leq 2 \times 10^5$ ； $1 \leq t_i \leq 10^9$ ； $1 \leq d_i \leq 10^9$ 。

样例

样例输入 1	样例输出 1
5 3 5 2 6 4 6 1 7 5 8	3

样例解释

例如，小鲤可以依次维修第 2、第 1、第 4 盏星灯。它们需要的维修时间分别为 2,3,1，因此三盏星灯的完成时间依次为 2,5,6。
对应的最后维修期限分别为 6,5,7，三盏星灯都能够在自己的期限内完成维修。因此至少可以成功维修 3 盏星灯。可以证明，不存在成功维修 4 盏星灯的方案，所以答案为 3。

F. For Stalin, Let's Sort!

时间限制：2 秒

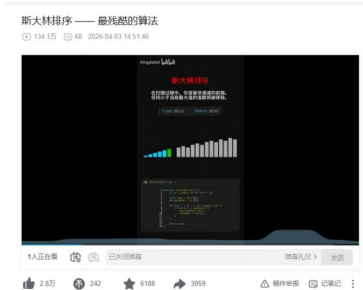
内存限制：256 MiB

输入方式：标准输入

输出方式：标准输出

题目背景

Hailuo 在刷视频的时候看到了一个排序方法，于是好奇地想实现一下，但他的代码能力太差了，写了 998244353 次都没办法复现这个排序，于是他找到了你来帮忙。



题目描述

给定一个长度为 n 的整数序列，从左到右依次处理每个元素。第一个元素一定保留。此后，若当前元素大于或等于最近一个被保留的元素，就保留它；否则删除它。请你输出最后留下的序列。

输入格式

第一行包含一个整数 T ，表示测试组数。
对于每组测试数据：

- 第一行包含一个整数 n ，表示序列长度。
- 第二行包含 n 个整数 $a_1, a_2, \dots, a_n$ ，表示原序列。

输出格式

对于每组测试数据，输出一行，按原顺序给出所有保留的整数，相邻整数用一个空格分隔。

数据范围

$1 \leq T \leq 10^4$ ， $1 \leq n \leq 2 \times 10^5$ ，所有测试数据的 n 之和不超过 2×10^5 ， $-10^9 \leq a_i \leq 10^9$ 。

样例

样例输入 1	样例输出 1
3 5 3 1 3 2 4 4 -2 -2 -3 0 1 7	3 3 4 -2 -2 0 7

样例解释

第一组中，3 被保留；1 小于最近保留的 3，被删除；下一个 3 与最近保留值相等，因此被保留。继续处理后，保留的是 3,3,4。

G. Grand Finale of the Witch Trial

时间限制：1 秒

内存限制：256 MiB

输入方式：标准输入

输出方式：标准输出

题目背景

致一路陪伴至此的你：

从满是苦痛的魔女岛，到这场迟来的夏夜烟花。跨越绝望、眼泪和无数次断裂的命运，十三位少女，终于稳稳降落在这名为“日常”的奇迹之中。

属于她们的故事，或许在天台的合影中暂时画上了休止符。

但在没有魔法的时空里，她们的青春、明天，连同未来的无限可能，才刚刚开始。

愿你在现实的漫长岁月中，也能拥有驱散阴霾的灿烂烟火，以及一双无论何时都会紧紧相握的手。

《魔法少女的魔女审判》官方后日谈

魔女审判已经结束，但——属于她们的时光还远没有结束。这天，二阶堂希罗和樱羽艾玛正为由谁来做晚饭而发愁。为此，两人决定进行一次特别的“魔女审判”，来决定今晚究竟由谁负责做饭。

题目描述

审判开始时，桌面上摆放着两组证据，第一组有 a 个证据，第二组有 b 个证据。

两人轮流进行审判。每一轮，当前玩家可以选择以下三种方式之一，对证据提出异议：

- 对第一组中的 1 个证据提出异议；
- 对第二组中的 1 个证据提出异议；
- 同时对第一组和第二组各 1 个证据提出异议。

每个证据只能被提出异议 1 次。由二阶堂希罗先手，两人都会采取对自己最有利的策略。当轮到一名玩家时，如果已经无法对任何证据提出异议，则该玩家判输。请你判断最终谁能够赢得这场“魔女审判”。

G. Grand Finale of the Witch Trial（续）

输入格式

第一行输入一个整数 T ，表示测试用例的组数。
接下来 T 行，每行输入两个整数 a, b ，分别表示第一组和第二组证据的数量。

输出格式

对于每组测试用例，输出一行：

- 若二阶堂希罗获胜，输出 `yuanshennb`；
- 若樱羽艾玛获胜，输出 `mingchaonb`。

数据范围

- $1 \leq t < 1000$ ；
- $0 \leq a, b < 1000$ ；
- 对于所有测试用例， $\sum_{i=1}^t a_i < 3000$ ， $\sum_{i=1}^t b_i < 3000$ 。

样例

样例输入 1	样例输出 1
2	yuanshennb
1 1	yuanshennb
1 2	

样例解释

对于第一个测试用例，初始时证据情况为 $(1, 1)$ ，希罗可以选择对两组证据各一个提出异议，轮到艾玛操作时证据情况为 $(0, 0)$ ，无法操作，希罗获胜。

对于第二个测试用例，初始时证据情况为 $(1, 2)$ ，希罗可以选择第一组的一个证据提出异议，此时证据情况为 $(0, 2)$ ，随后两人分别进行一次操作，轮到艾玛操作时证据情况为 $(0, 0)$ ，无法操作，希罗获胜。

H. How Hard Can Math Be?

时间限制：1 秒
输入方式：标准输入

内存限制：128 MiB
输出方式：标准输出

题目背景

fusun 正在研究整数拆分时最大公约数的变化规律，他遇到了一个不会的问题，来找你请教。

题目描述

给定 T 组测试数据。对于每组测试数据，给定一个正整数 n ，你需要找到两个正整数 a, b ，满足：

- $a + b = n$;
- $\gcd(a, b)$ 尽可能大。

其中， $\gcd(a, b)$ 表示 a 与 b 的最大公约数。如果有多组 (a, b) 都能使 $\gcd(a, b)$ 达到最大值，请输出其中 a 最小的一组。

输入格式

第一行包含一个整数 T ，表示测试数据的组数。
接下来 T 行，每行包含一个整数 n 。

输出格式

对于每组测试数据，输出一行两个正整数 a, b ，用一个空格分隔。

数据范围

- $1 \leq T \leq 100$;
- $2 \leq n \leq 10^9$;

样例

样例输入 1	样例输出 1
3	3 3
6	3 6
9	1 6
7	

样例解释

对于 $n = 6$ ，输出的两个数之和为 6，其最大公约数为 3。对于 $n = 9$ ，输出的两个数之和为 9，其最大公约数为 3。对于 $n = 7$ ，输出的两个数之和为 7，其最大公约数为 1。

I. It's Time for hrz to Sleep

时间限制：2 秒

内存限制：256 MiB

输入方式：标准输入

输出方式：标准输出

题目描述

hrz 学长一直被失眠所困扰，因为他想睡着的条件太苛刻了。只有当昨天和前天睡眠时间的差值为 x 时，他今天才可以安稳地睡着。

由于长期失眠，hrz 学长的记忆力出现了一个严重的问题——他记不住前两天的睡觉时长了！但是他能模糊的记起很多数字，他坚信前两天的睡眠时长就在其中。

你现在要做的是从这些数字中找出两个差值为 x 的数字分别作为 hrz 学长昨天和前天的睡眠时长，并统计可行解的数量。**特别的**， $x \neq 0$ 。

形式化的：给你一个长度为 n 的数组 $a_1, a_2, \dots, a_n$ ，以及一个整数 x ，请你求满足 $a_i - a_j = x$ 的有序二元对 (i, j) 的数量，其中 $i \neq j$ ，不同位置的相同数字视为不同。

输入格式

第一行输入一个正整数 T ，表示数据组数，接下来输入 T 组测试数据。每组测试数据包含两行：

第一行输入一个正整数 n ，和一个整数 x 。

第二行输入 n 个整数 $a_1, a_2, \dots, a_n$ 。

保证单个测试点内所有测试数据的 n 之和不超过 10^5 。

输出格式

对于每组数据输出一个整数，表示找到的满足条件的有序二元对数量。

数据范围

- $1 \leq T \leq 3000, 1 \leq n \leq 10^5$;
- $-10^9 \leq x \leq 10^9, x \neq 0, -10^9 \leq a_i \leq 10^9$;
- 对于所有测试用例， $\sum n \leq 10^5$ 。

样例

样例输入 1	样例输出 1
3 5 2 1 2 3 4 5 4 1 1 1 2 3 5 -3 -1 -1 0 1 2	3 3 2

样例输入 2	样例输出 2
2 6 5 5 5 0 0 5 5 6 -1 -2 -2 1 2 3 3	8 3

J. Journey of Liyu Beneath the Sea

时间限制：2 秒

内存限制：256 MiB

输入方式：标准输入

输出方式：标准输出

题目描述

本题为交互题。

鲤鱼在一次海底探险中，发现了一处由 n 个古老装置组成的遗迹。这些装置从左到右编号为 $1, 2, \dots, n$ 。鲤鱼得知其中恰好有两个装置仍然保存着能量。由于这种能量没有具体形态，无法直接观测，因此必须使用专门的探测器进行扫描。

每一次，鲤鱼都可以选择一段连续的装置编号区间 $[l, r]$ 进行扫描，探测器会告诉他：**在这段区间中，一共有多少个拥有能量的装置**。由于整个遗迹中只有两个有效装置，因此一次扫描的结果只可能是 0、1 或 2。

然而，探测器的电量已经所剩无几，最多只能再使用 50 次。请你帮助鲤鱼，在不超过 50 次扫描的情况下，确定两个有效装置的位置。

输入格式

第一行输入一个正整数 n ，表示装置的数量。

输出格式

本题为交互题，具体输出方式见「交互格式」。

温馨提示：

- 输出答案时，应当从小到大输出（即先输出较小的位置，再输出较大的位置）；
- 与交互器交互时，不应输出多余的空格，否则可能会得到 Wrong Answer 的结果。

数据范围

对于所有测试数据，满足 $2 \leq n \leq 2 \times 10^5$ 。

隐藏的两个位置满足 $1 \leq x < y \leq n$ 。

样例

样例输入 1	样例输出 1
10	? 2 6
1	? 7 10
1	? 1 3
0	? 4 7
1	? 8 9
1	! 4 9

询问次数限制

对于每组测试数据，你最多可以进行 50 次形如：? l r 的询问。

如果询问次数超过 50 次，则会被判定为答案错误。

J. Journey of Liyu Beneath the Sea（续）

交互格式

每次输出询问后，都必须立即刷新输出缓冲区，否则程序可能会一直等待交互器返回结果。

C++

可以使用：

```
cout << "? " << l << " " << r << endl;
```

其中 endl 会自动刷新输出缓冲区。
也可以使用：

```
cout << "? " << l << " " << r << '\n';  
cout.flush();
```

当确定答案后，也应输出最终答案并刷新，例如：

```
cout << "! " << x << " " << y << endl;
```

C

可以使用：

```
printf("? %d %d\n", l, r);  
fflush(stdout);
```

其中 fflush(stdout) 用于刷新标准输出缓冲区。
当确定答案后，也应输出最终答案并刷新，例如：

```
printf("! %d %d\n", x, y);  
fflush(stdout);
```

Python

可以使用：

```
print("? ", l, r, flush=True)
```

其中 flush=True 会立即刷新输出缓冲区。
也可以使用：

```
import sys  
  
print("? ", l, r)  
sys.stdout.flush()
```

当确定答案后，也应输出最终答案并刷新，例如：

```
print("!", x, y, flush=True)
```

样例解释

假设两个有效装置的位置分别为 4 和 9。

询问	返回值	说明
? 2 6	1	区间 [2,6] 中恰好包含一个有效装置。
? 7 10	1	区间 [7,10] 中也恰好包含一个有效装置。
? 1 3	0	区间 [1,3] 中不存在有效装置。
? 4 7	1	区间 [4,7] 中恰好包含一个有效装置。
? 8 9	1	区间 [8,9] 中恰好包含一个有效装置。

最终程序确定两个有效装置的位置分别为 4 和 9，因此输出 ! 4 9。

注意：该样例仅用于说明交互格式，并不代表一种正确的询问策略。
在不同实际测试中，有效装置的位置不同。

K. Keep RiNG in Order

时间限制：2 秒

内存限制：256 MiB

输入方式：标准输入

输出方式：标准输出

题目背景

Hailuo 是 Live House RiNG 的管理员，负责安排今天所有人的出场顺序。

一大早，第一次参加演出的 Mutsumi 就找到了他。由于有些害羞，她始终不太敢独自走上舞台。犹豫了很久后，她小声提出了一个请求：希望自己能够在 Saki 演出之后上场。对她来说，只要先看到 Saki 在舞台上演奏的样子，就能安心不少，也能获得走上舞台的勇气。

然而，这段对话正好被休息室里的 Rikki 听见了，她立刻不高兴起来——按照原本的安排，排在 Saki 后面的人明明是 Tomorin。

“竟敢无视灯？”

立希当场要求 Hailuo 无论如何都要保证 Tomorin 在 Saki 之后出场，而且不能因为其他人的请求随便改变这一点。

没过多久，其他成员也陆续找了过来。有人希望在某个人之前出场，因为她想早点结束演出；有人坚持必须在某个人之后登台；还有人表示，如果自己的出场顺序被改到某个人前面，就绝对不会接受……

面对越来越多的要求，Hailuo 很快发现事情已经完全超出了自己的处理能力。于是，他找到了聪明的你，希望你能够根据所有人的要求，判断是否存在一种合理的出场顺序，并帮他安排好今天的演出。

题目描述

今天有 n 位成员准备登台演出，她们分别编号为 $1, 2, \dots, n$ 。她们向你提出了 m 条要求，每条要求指定成员 u 必须在成员 v 之前出场。

你需要安排所有成员的出场顺序方案，满足以上 m 条要求。如果存在多个合法方案，输出任意一个即可；如果不存在合法方案，输出 -1 。

K. Keep RiNG in Order（续）

输入格式

第一行包含两个整数 n, m ，分别表示成员数量和要求数量。
接下来 m 行，每行包含两个整数 u, v ，表示成员 u 必须在成员 v 之前出场。

输出格式

如果不存在合法顺序，输出一行一个整数 -1 。
否则输出一行 n 个整数，表示所有成员的一种合法出场顺序。每位成员必须恰好出现一次。

数据范围

对于所有测试数据， $1 \leq n \leq 2 \times 10^5$ ， $0 \leq m \leq 2 \times 10^5$ ， $1 \leq u, v \leq n$ ，且 $u \neq v$ 。保证同一条要求不会重复出现。

样例

样例输入 1	样例输出 1
5 5 1 4 5 1 2 3 3 4 2 5	2 5 1 3 4

样例输入 2	样例输出 2
3 3 1 2 2 3 3 1	-1

样例解释

样例一中，2 在 5 和 3 之前，5 在 1 之前，1 和 3 都在 4 之前，因此给出的顺序满足所有要求。样例二中，三位成员的要求首尾相接，无法安排合法顺序。

L. Look, It's Yesterday Once More

时间限制：1 秒

内存限制：256 MiB

输入方式：标准输入

输出方式：标准输出

题目背景

随着时间的流逝，刚进入大学的大一懵懂小生如今已经变成了大四老登，要毕业的他不得不开始实习上班。

某天，瓢泼大雨，老登在公司门口摔了一跤，好不容易爬到工位，迎接他的是没人愿意做、于是顺手丢给实习生的 dirty work，以及没有统一口径随着时间反复横跳的需求。

每当老登做完手头的工作，向需求方询问接下来该干什么时，得到的回答都可能截然不同。

遗憾的是，需求方自始至终都没有想好过需要什么。只有在收到询问后，他们才会灵机一动，现场构思出一个需求。于是，老登只能一边盯着时间，一边在各个工作阶段之间被需求踢来踢去。

终于，他崩溃了，发出了：

“这 β 班谁爱上谁上。”

的声音。

在历经千辛万苦完成工作之后，老登终于下班了，但饿了一天的老登因为工资很低只能去便利店买点泡面饱腹。

然而刚刚走出便利店门口，感到疲倦的他揉了揉眼睛，却发现自己回到了开始工作的第一天。

好消息是，上一次工作的记忆并没有消失。虽然需求方依旧会现场发挥，但经历过完整的一轮折磨后，老登已经彻底摸清了他们的规律。

为了尽快完成工作下班加训，他带着这些记忆，再次踏入了公司。

Oh please don't let me die, Waiting for your touch.

No don't give up on life, This endless dead end.

题目描述

为了方便管理，老登把整个工作拆成了 n 个工作阶段，编号为 $1, 2, \dots, n$ 。他需要从阶段 1 开始，最终推进到阶段 n 。

老登在时刻 0 位于工作阶段 1，工作完成当且仅当首次达到工作阶段 n 。

对于每个工作阶段 u ，需求方一共可能给出 d_u 种不同的需求。为了方便描述，将这些需求依次编号为 $0, 1, \dots, d_u - 1$ 。

如果老登在整数时刻 t 位于工作阶段 u ，并在此时向需求方询问接下来的安排，那么需求方现编出的恰好是第 $t \bmod d_u$ 种需求。

第 j 种需求会花费老登 $w_{u,j}$ 单位时间，并在完成后进入工作阶段 $v_{u,j}$ 。一旦老登选择向需求方询问，他必须立刻开始执行需求方在该时刻提出的需求。

老登可以选择暂时不询问，而是在当前工作阶段摸鱼等待任意非负整数时间。

请你帮助老登计算一下从工作阶段 1 开始，最早需要多少时间才能完成工作。

并且老登还希望知道，对于每种需求是否有可能出现在一种最快完成工作的方案中。

L. Look, It's Yesterday Once More（续）

输入格式

第一行包含两个整数 n, m ($2 \leq n \leq 2 \times 10^5$, $n - 1 \leq m \leq 2 \times 10^5$), 分别表示工作阶段的数量和需求的总数。

接下来 m 行, 每行包含三个整数 u, v, w ($1 \leq u, v \leq n$, $1 \leq w \leq 10^9$), 表示工作阶段 u 存在一种需求, 需要花费 w 的时间完成, 并且在完成后进入工作阶段 v 。

对于同一个工作阶段 u , 它的所有需求按照在输入中出现的先后顺序, 依次编号为 $0, 1, \dots, d_u - 1$ 。

数据保证对于所有 $1 \leq u < n$, 都有 $d_u \geq 1$ 。

数据保证从工作阶段 1 出发可以到达工作阶段 n 。

输出格式

第一行输出一个整数 T , 表示老登最早完成工作所需要的时间。

第二行输出一个长度为 m 的 01 串 s , 对于输入的第 i 个需求, 如果存在一种最快完成工作的方案执行过此需求, 则 $s_i = 1$, 否则 $s_i = 0$ 。

样例

样例输入 1	样例输出 1
3 5 1 2 1 1 2 1 2 2 1 2 2 1 2 3 1	3 11011