

2022  icpc 国际大学生程序设计竞赛
亚洲区域赛（南京站）

正式赛

2022 年 12 月 18 日



试题列表

A	停停，昨日请不要再重现
B	索道
C	智巧灵蕈大竞逐
D	聊天程序
E	树的染色
F	三角形
G	邪恶铭刻
H	工厂重现
I	完美回文
J	完美匹配
K	堆里的 NaN
L	命题作文
M	清空水箱

本试题册共 13 题，26 页。
如果您的试题册缺少页面，请立即通知志愿者。

由 SUA 程序设计竞赛命题组命题。

<https://sua.ac/>

承办方



命题方

签到成功 这是你的
签到奖励



竞赛过程中访问非竞赛网页是违反竞赛规则的行为。
如果您有兴趣（我们很荣幸），
请在竞赛后扫描二维码。

Problem A. 停停，昨日请不要再重现

继 2018, 2019, 2020 和 2021 年成功承办赛事之后，南京航空航天大学（NUAA）将连续第五年承办国际大学生程序设计竞赛（ICPC）。

在 2018 与 2019 年，“中二之力”队与“三个顶俩”队为清华大学赢得了冠军。在 2020 与 2021 年，北京大学的“逆十字”队连续赢得冠军。该队也在达卡举办的第 45 届国际大学生程序设计竞赛全球总决赛中获得了亚军，创造了东大陆区域过去六年来的最佳成绩。让我们恭喜他们，同时也非常期待他们在 2022 年南京站的表现！

今年，将会有约 500 支队伍参与南京站的竞赛。本次竞赛将会颁发至多 35 项金奖，70 项银奖与 105 项铜奖。让我们期待选手们出色的表现！

虽然由于疫情，我们（又一次）无法在南京相聚，我们仍然需要感谢竞赛组委会与志愿者们的努力付出。感谢你们为本次竞赛做出的贡献！



2018 国际大学生程序设计竞赛亚洲区域赛（南京站）

在 2018 年的竞赛中，K 题《袋鼠谜题》要求选手为以下游戏构造一个操作序列：

谜题由一个 n 行 m 列的网格 ($1 \leq n, m \leq 20$) 组成，且有一些（至少 2 只）袋鼠位于网格中。玩家的目标是控制袋鼠并把它们聚集在同一个格子中。一些格子里有墙，袋鼠无法进入这些有墙的格子，而其它格子是空的。袋鼠可以从一个空格子移动到上，下，左，右相邻的另一个空格子中。

游戏开始时，每个空格子里都有一只袋鼠。玩家可以通过键盘上 U, D, L, R 四个按键控制袋鼠的移动。所有袋鼠会同时根据您按下的按键移动。

选手需要构造一个长度至多为 5×10^4 且由 U, D, L, R 组成的操作序列以达成目标。

在 2020 年的竞赛中，A 题《啊，昨日重现》要求选手构造一张输入地图，以证明以下代码并不是上述问题的解：

```
#include <bits/stdc++.h>
using namespace std;
string s = "UDLR";
int main()
{
    srand(time(NULL));
    for (int i = 1; i <= 50000; i++) putchar(s[rand() % 4]);
    return 0;
}
```

在 2021 年的竞赛中，A 题《呀，昨日再次重现》同样要求选手为以下游戏构造操作序列：

本题中，网格中的每个格子都有恰好一只袋鼠。您需要构造一个仅由字符 ‘U’，‘D’，‘L’ 和 ‘R’ 组成的操作序列。在应用该操作序列后，所有袋鼠必须聚集在指定格子 (a, b) 中。操作序列的长度不能超过 $3(n - 1)$ 。同往常一样，所有袋鼠会根据您的命令同时移动。

在 2022 年的竞赛中，袋鼠题又回来啦！我们不知道为什么命题组的成员们那么喜欢袋鼠，但题目如下：

给定一张 n 行 m 列的网格，在位于第 i_h 行第 j_h 列的格子上有一个洞，其它每个格子都是空地并且都有一只袋鼠。

相似地，袋鼠可以被键盘上的 U, D, L, R 键控制。所有袋鼠会同时根据按下的按键移动。具体来说，对于一只位于第 i 行第 j 列的格子（用 (i, j) 表示）上的袋鼠：

1. 按键 U：它会移动到 $(i - 1, j)$ 。
2. 按键 D：它会移动到 $(i + 1, j)$ 。
3. 按键 L：它会移动到 $(i, j - 1)$ 。
4. 按键 R：它会移动到 $(i, j + 1)$ 。

如果一只袋鼠踩到了洞（也就是说， $i = i_h$ 且 $j = j_h$ ）或者移动到了网格外面，它将被从网格上移除。

问题在于， i_h 与 j_h 的值是未知的。您只知道一个仅由字符 ‘U’，‘D’，‘L’，‘R’ 组成的操作序列，以及一个整数 k 表示应用这个操作序列之后，网格上恰有 k 只袋鼠存留。

请计算有多少位置可能存在洞。也就是说，计算满足以下条件的整数对 (i_h, j_h) 的数量：

- $1 \leq i_h \leq n$, $1 \leq j_h \leq m$ 。
- 洞位于 (i_h, j_h) 。
- 应用给定的操作序列后，网格上恰有 k 只袋鼠存留。

Input

有多组测试数据。第一行输入一个整数 T 表示测试数据组数，对于每组测试数据：

第一行输入三个整数 n , m 与 k ($1 \leq n, m \leq 10^3$, $0 \leq k < n \times m$) 表示网格的大小以及应用操作序列后网格上存留的袋鼠数量。

第二行输入一个字符串 $s_1 s_2 \cdots s_l$ ($s_i \in \{\text{'U', 'D', 'L', 'R'}\}$, $1 \leq l \leq 10^6$) 表示操作序列。

保证所有数据 $n \times m$ 之和以及操作序列长度之和均不超过 10^6 。

Output

每组数据输出一行一个整数表示有多少位置可能存在洞。

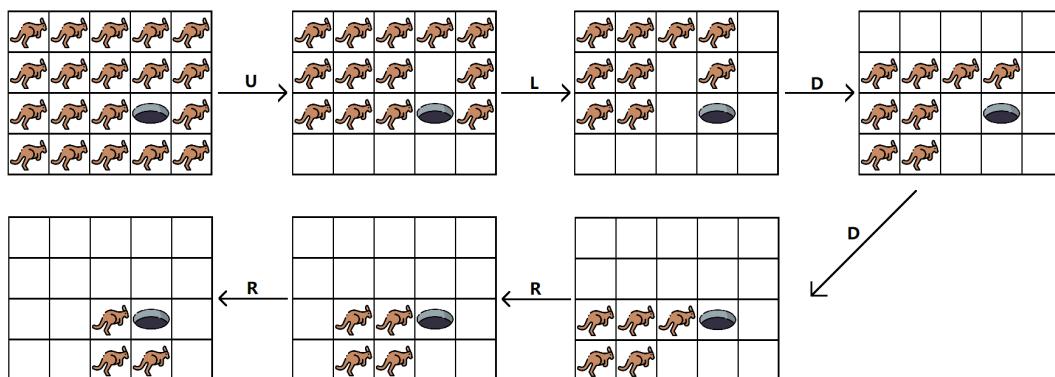
Example

standard input	standard output
3	2
4 5 3	20
ULDDRR	0
4 5 0	
UUUUUUUU	
4 5 10	
UUUUUUUU	

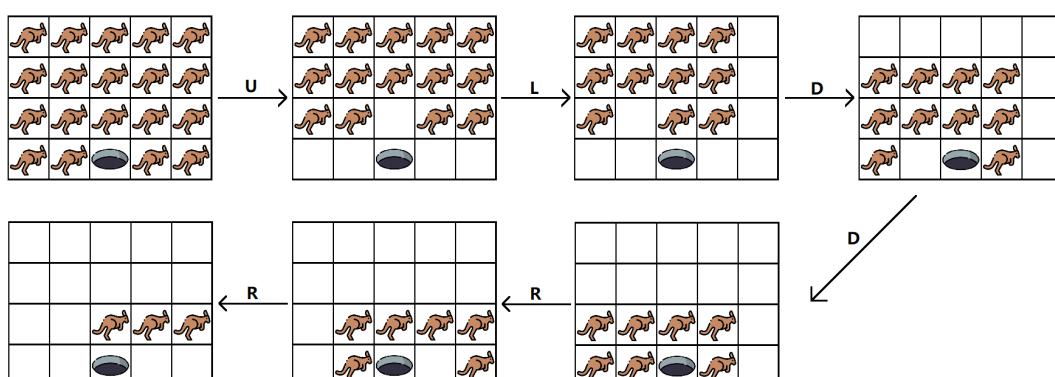
Note

对于第一组样例数据，有 2 个位置可能存在洞。

第一个可能的位置是 (3, 4)。



第二个可能的位置是 (4, 3)。



Problem B. 索道

紫金山是南京的名山之一，而其中的紫金山索道曾是中国最长的吊椅式索道。游客们可以从山脚搭乘索道并一路直达山顶，总用时不超过十分钟。



旧紫金山索道。MiNe 摄。通过 CC BY 2.0 协议授权。

为了建设索道，除了山脚和山顶的索道站，还需要沿路架设支撑塔。工程师们已经在距离索道入口 0 单位距离和 $(n + 1)$ 单位距离的位置建设了索道站，并勘察了在 $1, 2, \dots, n$ 单位距离架设支撑塔的成本，分别是 a_1, a_2, \dots, a_n 。

出于工程因素考虑，部分位置必须架设支撑塔。同时为了保证索道的安全性，相邻支撑塔或索道站之间的距离必须小于等于 k 。也就是说，假设最终建设两座索道站与架设 m 座支撑塔的位置为 $b_0, b_1, b_2, \dots, b_m, b_{m+1}$ ，其中 $0 \leq m \leq n$ ， $0 = b_0 < b_1 < b_2 < \dots < b_m < b_{m+1} = n + 1$ ，那么对于所有 $1 \leq i \leq m + 1$ 有 $b_i - b_{i-1} \leq k$ 。

同时，为了使建设方案更加灵活，工程师们会对成本序列进行 q 次临时的修改。第 i 次修改形如 (p_i, v_i) ，意思是临时将 a_{p_i} 的值修改为 v_i 。

您需要对每次修改求出，符合以上要求的前提下，架设支撑塔的最小总成本。

请再次注意，所有修改都是临时且相互独立的。也就是说，在计算一次修改的答案之后，此次修改将会被撤销，成本序列将会恢复原来的样子。

Input

有多组测试数据。第一行输入一个整数 T 表示测试数据组数，对于每组测试数据：

第一行输入两个整数 n 与 k ($1 \leq n \leq 5 \times 10^5$, $1 \leq k \leq \min(n + 1, 3 \times 10^3)$) 表示架设支撑塔的备选位置数以及相邻支撑塔或索道站之间的最大距离。

第二行输入 n 个整数 a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^9$)，其中 a_i 表示在 i 单位距离架设支撑塔的成本。

第三行输入一个长度为 n 的二进制字符串 $s_1 s_2 \dots s_n$ ($s_i \in \{0, 1\}$)。如果 $s_i = '1'$ 则必须在 i 单位距离架设支撑塔。如果 $s_i = '0'$ 则您可以决定是否在 i 单位距离架设支撑塔。

第四行输入一个整数 q ($1 \leq q \leq 3 \times 10^3$) 表示修改的次数。

对于接下来 q 行，第 i 行输入两个整数 p_i 与 v_i ($1 \leq p_i \leq n$, $1 \leq v_i \leq 10^9$) 表示第 i 次修改。

保证所有数据 n 之和不超过 2×10^6 。另外保证所有数据 k 之和与 q 之和均不超过 10^4 。

Output

每次修改输出一行一个整数表示临时应用该修改后，架设支撑塔的最小总成本。

Example

standard input	standard output
3	206
10 3	214
5 10 7 100 4 3 12 5 100 1	0
0001000010	100
2	
2 3	
6 15	
5 6	
1 1 1 1 1	
00000	
1	
3 100	
5 6	
1 1 1 1 1	
00100	
1	
3 100	

Note

对于第一组样例数据：

- 应用第一次修改后，成本序列变为 $\{5, 3, 7, 100, 4, 3, 12, 5, 100, 1\}$ 。应选择在 2, 4, 6 与 9 单位距离架设支撑塔以最小化总成本。
- 应用第二次修改后，成本序列变为 $\{5, 10, 7, 100, 4, 15, 12, 5, 100, 1\}$ 。应选择在 1, 4, 5, 8 与 9 单位距离架设支撑塔以最小化总成本。

对于第二组样例数据，无需架设支撑塔。

对于第三组样例数据，必须在 3 单位距离架设支撑塔，因此答案就是 a_3 。

Problem C. 智巧灵蕈大竞逐

您正作为世界知名游戏《原神》中的旅行者游历提瓦特大陆。在旅经蒙德，璃月和稻妻之后，现在您来到了名为“须弥”的国度。

蕈兽是存在于须弥的一种敌对生物。它们是真菌进化后的生物，具有更强的产生与保护孢子的能力。虽然它们外表十分可爱，但实际上十分危险。蕈兽也是一种三相众物，因此火元素与雷元素将对它们产生相反的影响。受到火元素攻击后，蕈兽将进入枯焦状态，攻击速度减慢，但是会造成更大的伤害。相反地，雷元素的攻击将导致蕈兽进入活化状态，攻击速度会提升。由于不同种类的蕈兽可以抵挡不同元素的不同技能，可以预见，它们将成为旅行者的强力伙伴。您开始考虑如何与蕈兽组成队伍，并让它们听从您的指令。



幸运的是，在“智巧灵蕈大竞逐”活动期间，您可以控制蕈兽与对手进行战斗。您被邀请参加“月莲杯”驯兽师大赛，并可以使用名为“意智宝珠”的道具捕获并控制蕈兽，组成自己的蕈兽小队。然而，为了取得大赛的胜利，您还需要通过完成“潜能焕发”挑战来唤醒蕈兽的潜能，解锁它们的特殊技能。

在潜能焕发挑战中，旅行者需要将“花花琼脂”调配成蕈兽喜欢的组合，蕈兽吸收后可唤醒潜能。



具体地，挑战开始时，您会得到一个初始的花花琼脂组合，可用一个 $n \times m$ 大小的矩阵来表示。对于所有 $1 \leq i \leq n$ 与 $1 \leq j \leq m$ ，每个位置 (i, j) 都有一个花花琼脂。矩阵里每个位置的值代表琼脂的种类，如果两个位置的值相同，则两个位置上的琼脂种类也相同。

您可以进行 3 种操作：对调，旋转和预置。

对调：对于两个相邻的花花琼脂，您可以使用一次对调操作来交换它们的位置。

具体来说，两个位于 (x_1, y_1) 和 (x_2, y_2) 的花花琼脂被认为是相邻的，当且仅当 $|x_1 - x_2| + |y_1 - y_2| = 1$ 。交换之后，位于 (x_1, y_1) 和 (x_2, y_2) 的花花琼脂将分别变成之前位于 (x_2, y_2) 和 (x_1, y_1) 的琼脂。



旋转：对于任意四个组成 2×2 大小的子方阵的花花琼脂，您可以通过一次旋转操作来顺时针移动它们的位置。

具体来说，一个 2×2 方阵意味着任意四个位于 (x, y) , $(x, y+1)$, $(x+1, y+1)$ 和 $(x+1, y)$ 的花花琼脂（满足 $1 \leq x < n$ 和 $1 \leq y < m$ ）。在旋转操作后，位于 (x, y) , $(x, y+1)$, $(x+1, y+1)$ 和 $(x+1, y)$ 的花花琼脂会分别变成之前位于 $(x+1, y)$, (x, y) , $(x, y+1)$ 和 $(x+1, y+1)$ 的琼脂。



预置：对于任意形成 $n' \times m'$ 大小的子矩阵 ($1 \leq n' \leq n$, $1 \leq m' \leq m$) 的花花琼脂，您可以通过一次预置操作将它们一次性覆盖成一个预先给出的且大小相同的配方。用一个大小为 $n' \times m'$ 的矩阵 \mathbf{F} 来表示一个配方。

具体来说，对于每一个位于 $(x+i, y+j)$ 的花花琼脂（满足 $1 \leq x \leq n-n'+1$, $1 \leq y \leq m-m'+1$, $0 \leq i < n'$ 和 $0 \leq j < m'$ ），在使用配方 \mathbf{F} 的预置操作后，位于 $(x+i, y+j)$ 的花花琼脂将变成 \mathbf{F} 中位于 (i, j) 的琼脂。



挑战指定了最终需要形成的花花琼脂目标组合。也就是说，在完成所有操作之后，矩阵里每个位置的花花琼脂的类型都必须和目标组合的相应位置一致。

Input

每个测试文件仅有一组测试数据。

第一行输入三个整数 n, m, k ($2 \leq n, m \leq 20, 1 \leq k \leq 20$) 表示矩阵的大小以及预置配方数量。

对于接下来 n 行，第 i 行输入一个字符串 $s_{i,1}s_{i,2} \dots s_{i,m}$ 。其中 $s_{i,j}$ 表示在初始组合中，位于第 i 行第 j 列的琼脂的种类。

接下来是一个空行。

对于接下来 n 行，第 i 行输入一个字符串 $t_{i,1}t_{i,2} \dots t_{i,m}$ 。其中 $t_{i,j}$ 表示在目标组合中，位于第 i 行第 j 列的琼脂的种类。

接下来会输入 k 个预置配方。对于第 p 个预置配方：

首先是一个空行。

接下来一行输入两个整数 n_p 和 m_p ($1 \leq n_p \leq n, 1 \leq m_p \leq m$) 表示预置配方对应的子矩阵大小。

对于接下来 n_p 行，第 i 行输入一个字符串 $f_{i,1}^{(p)}f_{i,2}^{(p)} \dots f_{i,m_p}^{(p)}$ 。其中 $f_{i,j}^{(p)}$ 表示在第 p 个配方中，位于第 i 行第 j 列的琼脂的种类。

总共有 62 种不同种类的花花琼脂，用小写字母（‘a’ 到 ‘z’），大写字母（‘A’ 到 ‘Z’）以及数码（‘0’ 到 ‘9’）来表示。两个花花琼脂被认为是种类相同的，当且仅当它们对应的字符相同。所有输入的字符串只会包含这 62 种字符。

Output

如果不存在合法的操作序列，请输出 “-1”（不包含引号）。

否则，在第一行输出一个数字 r ($0 \leq r \leq 4 \times 10^5$) 表示所需要的操作步数。

接下来输出 r 行，每行包含三个整数 op, x 和 y ，表示一个作用于当前花花琼脂组合的操作。您需要按执行操作的顺序输出所有操作。可用操作如下：

- $-4 x y$: 交换位于 (x, y) 和 $(x - 1, y)$ 的琼脂。要求满足 $1 < x \leq n$ 以及 $1 \leq y \leq m$ 。
- $-3 x y$: 交换位于 (x, y) 和 $(x + 1, y)$ 的琼脂。要求满足 $1 \leq x < n$ 以及 $1 \leq y \leq m$ 。
- $-2 x y$: 交换位于 (x, y) 和 $(x, y - 1)$ 的琼脂。要求满足 $1 \leq x \leq n$ 以及 $1 < y \leq m$ 。
- $-1 x y$: 交换位于 (x, y) 和 $(x, y + 1)$ 的琼脂。要求满足 $1 \leq x \leq n$ 以及 $1 \leq y < m$ 。
- $0 x y$: 顺时针旋转位于 $(x, y), (x, y + 1), (x + 1, y + 1)$ 和 $(x + 1, y)$ 的琼脂，要求满足 $1 \leq x < n$ 以及 $1 \leq y < m$ 。
- $op x y$: 使用一次预置配方 op ，将所有位于 (i, j) 的琼脂（满足 $x \leq i \leq x + n_{op} - 1$ 以及 $y \leq j \leq y + m_{op} - 1$ ）替换成预置配方。要求满足 $1 \leq op \leq k, 1 \leq x \leq n - n_{op} + 1$ 以及 $1 \leq y \leq m - m_{op} + 1$ 。

“位于 (x, y) 的琼脂”指的是位于从上到下第 x 行，从左到右第 y 列的琼脂。

除了操作总数不超过 4×10^5 的限制外，我们额外限制预置操作 ($1 \leq op \leq k$) 数量不能超过 400。可以证明如果存在合法操作序列，则至少存在一个操作序列满足以上条件。

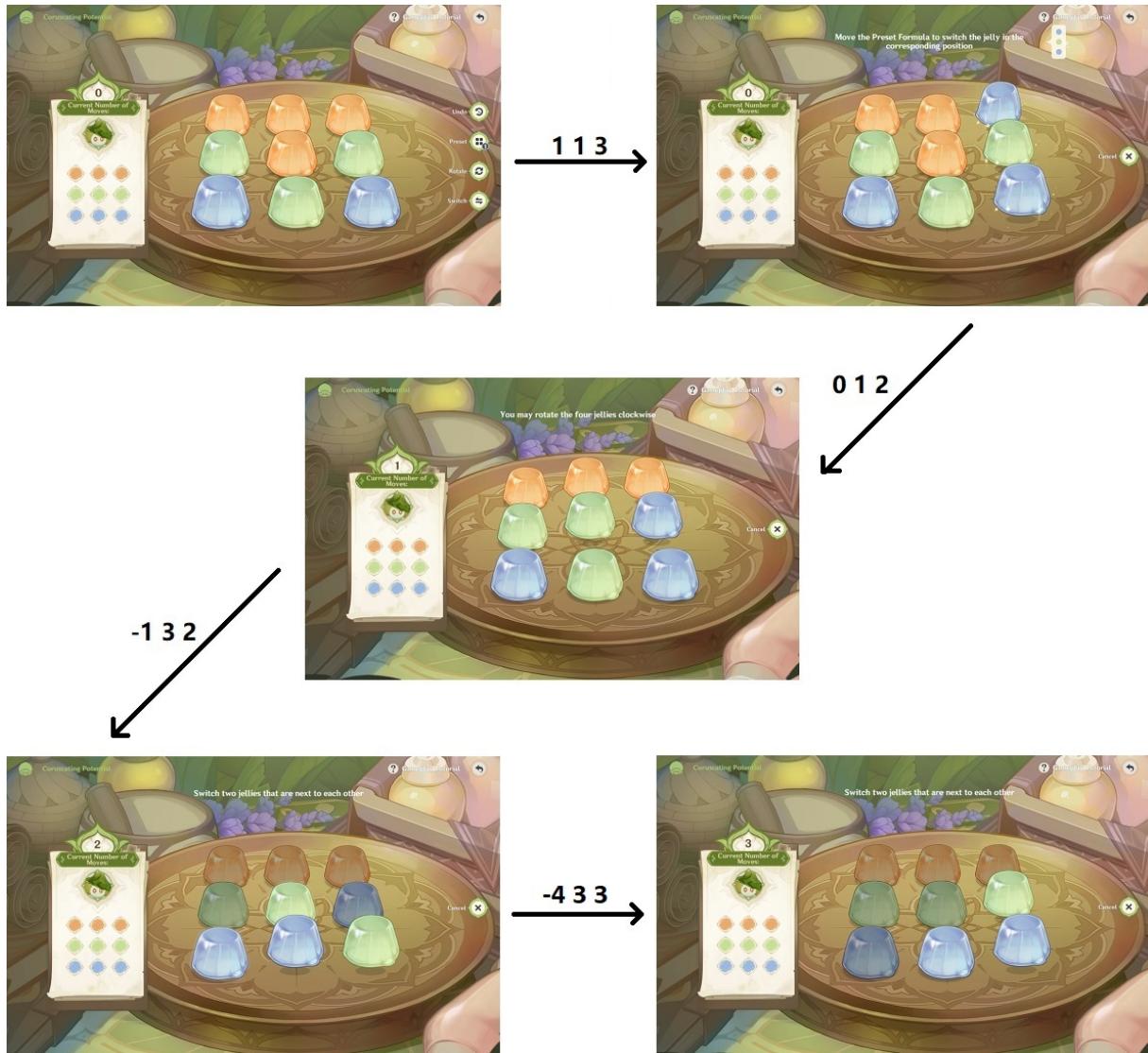
请注意，您并不需要最小化操作数量。若有多种方案，输出任意一种。

Examples

standard input	standard output
3 3 1 000 GOG BGB 000 GGG BBB 3 1 B G B	4 1 1 3 0 1 2 -1 3 2 -4 3 3
2 2 1 00 00 PP PP 1 2 OP	-1
4 8 4 11122222 33344444 55556666 77777777 NIxSHU0x DExDUIxx DANxSHIx YUANSHEN 2 3 NIy DEx 3 8 ZZZZZZZZ DANNSH9I YUA9SHEN 1 1 x 2 5 SH08y DUUI8	13 2 2 1 -3 3 4 -2 3 8 1 1 1 4 1 4 0 1 6 3 1 3 3 1 8 3 2 3 3 2 7 3 2 8 3 3 4 3 3 8 3 8 ZZZZZZZZ DANNSH9I YUA9SHEN 1 1 x 2 5 SH08y DUUI8

Note

对第一组样例数据解释如下。



Problem D. 聊天程序

您是国际聊天程序公司（International Chatting Program Company, ICPC）的研究员。今天，您在浏览研究数据时，发现了以下聊天记录。

SUA (2022/12/04 23:01:25)

我已经用光了程序设计竞赛题目的点子！请给我一道关于序列的题目。

机器人 (2022/12/04 23:01:27)

当然可以，下面是一道关于序列的程序设计竞赛的题目。

给定一个长度为 n 的整数序列 a_1, a_2, \dots, a_n ，同时给定另外四个整数 k, m, c 与 d ，您的目标是最大化序列中第 k 大的值。

为了达成这一目标，您可以进行以下操作至多一次：选择一个长度恰为 m 的连续子数组，并将一个长度为 m ，首项为 c ，公差为 d 的等差序列加到该连续子数组上。

更正式地，您可以选择一个整数 p 满足 $1 \leq p \leq n - m + 1$ ，而对于所有 $0 \leq i < m$ ，将 a_{p+i} 增加 $(c + di)$ 。

求至多一次操作之后，序列中第 k 大的值最大可能是多少。

序列中第 k 大的值，指的是将序列从大到小排序后，位于序列第 k 项的值。例如，序列 {5, 7, 1, 9} 中，第 3 大的值为 5；而序列 {9, 7, 5, 9} 中，第 3 大的值为 7。

SUA (2022/12/05 00:15:17)

这道题目好像很难！请告诉我它的解法。

机器人 (2022/12/05 00:15:30)

当然可以。首先，我们可以...

[数据删除]

遗憾的是，部分聊天记录由于硬盘损坏而丢失了。您为聊天程序可以创造程序设计竞赛的题目而感到非常惊奇。为了验证聊天程序是否能创造有效的题目，您决定尝试解决这道题目。

Input

每个测试文件仅有一组测试数据。

第一行输入五个整数 n, k, m, c 和 d ($1 \leq k, m \leq n \leq 2 \times 10^5$, $0 \leq c, d \leq 10^9$) 表示序列的长度，您的目标，等差序列的长度，首项和公差。

第二行输入 n 个整数 a_1, a_2, \dots, a_n ($0 \leq a_i \leq 10^9$) 表示给定的序列。

Output

输出一行一个整数，表示至多一次操作之后，序列中第 k 大的值最大可能是多少。

Examples

standard input	standard output
6 4 3 1 2 1 1 4 5 1 4	4
7 3 2 4 0 1 9 1 9 8 1 0	9
8 3 5 0 0 2 0 2 2 1 2 1 8	2

Note

对于第一组样例数据，可以选择 $p = 3$ 使序列变为 $\{1, 1, 5, 8, 6, 4\}$ 。序列中第 4 大的值为 4。

对于第二组样例数据，可以选择 $p = 5$ 使序列变为 $\{1, 9, 1, 9, 12, 5, 0\}$ 。序列中第 3 大的值为 9。

对于第三组样例数据，容易发现操作不改变序列的值，因此我们选择不进行操作。序列中第 3 大的值为 2。

能鼓励一下参加程序设计竞赛的选手们吗？



当然可以，下面是一些鼓励参加程序设计竞赛的选手的话语：



- 继续努力！你已经付出了很多努力来达到这一点，继续推动自己是很重要的。
- 不要害怕寻求帮助。如果你在一个问题上卡住了，不要犹豫，向其他人寻求指导。**仅限您的队友！**
- 保持积极态度。面对困难问题时感到沮丧是很自然的，但是尽量保持积极态度，继续前进。
- 必要时休息。保持专注，避免精疲力尽是很重要的。如果感觉不堪重负，就休息一下，稍后用新的心态再来思考问题。
- 记住，每个人都有自己的优势和劣势。不要将自己与他人比较，专注于自己的进步和提高。

总之，最重要的是要玩得开心，享受挑战。祝你好运！

OpenAI ChatGPT 正在鼓励选手

Problem E. 树的染色

有一棵 n 个节点的有根树。节点编号从 1 到 n （含两端），其中节点 1 是根节点。一开始所有 n 个节点都是白色的，您需要将所有节点染成黑色。

为了帮助您达成目标，我们提供 n 种操作，编号从 0 到 $(n - 1)$ （含两端）。操作 i ($0 \leq i \leq n - 1$) 需要您首先选择一个节点 u ，之后将所有满足以下条件的节点 v 染成黑色：

- 节点 v 在以 u 为根的子树里，也就是说 $u = v$ 或 u 是 v 的祖先节点。
- 节点 u 与 v 之间的距离恰为 i 。节点 u 与 v 之间的距离指的是从 u 走到 v 需要经过的最少边数。

执行一次操作 i 的代价是 a_i 。一个节点可以被多次染色，所有操作都可以被执行任意次。求将所有节点染成黑色的最小总代价。

Input

有多组测试数据。第一行输入一个整数 T 表示数据组数，对于每组测试数据：

第一行输入一个整数 n ($2 \leq n \leq 10^5$) 表示树的大小。

第二行输入 n 个整数 a_0, a_1, \dots, a_{n-1} ($1 \leq a_i \leq 10^9$)，其中 a_i 表示执行一次操作 i 的代价。

对于接下来 $(n - 1)$ 行，第 i 行输入两个整数 u_i 和 v_i ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$) 表示有一条边连接节点 u_i 和 v_i 。

保证所有数据 n 之和不超过 3×10^5 。

Output

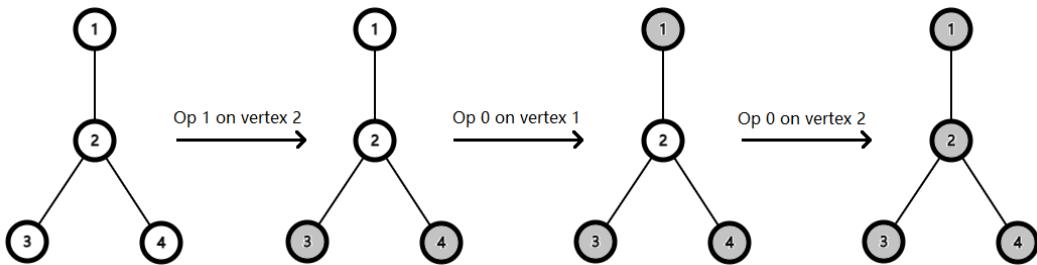
每组数据输出一行一个整数，表示染黑整棵树的最小总代价。

Example

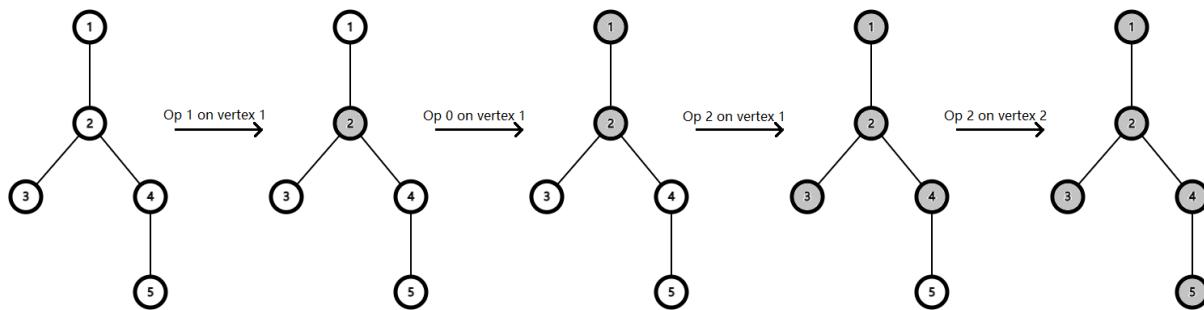
standard input	standard output
3	35
4	17
10 15 40 1	1218
1 2	
2 3	
2 4	
5	
10 5 1 100 1000	
1 2	
2 3	
2 4	
4 5	
4	
1000 200 10 8	
1 2	
2 3	
3 4	

Note

第一组样例数据如下所示。答案是 $15 + 10 + 10 = 35$ 。



第二组样例数据如下所示。答案是 $5 + 10 + 1 + 1 = 17$ 。



Problem F. 三角形

平面上有一个正方形。正方形左下角、左上角、右下角和右上角的坐标分别是 $(0, 0)$, $(0, 10^9)$, $(10^9, 0)$ 和 $(10^9, 10^9)$ 。

给定一个正整数 k , 您需要将该正方形分割成恰好 k 个锐角三角形。也就是说, 找到 k 个锐角三角形满足任意两个三角形不相交 (但它们可以有一个公共点或一条公共线段) 且所有三角形的并等于该正方形。

Input

每个测试文件仅有一组测试数据。

第一行输入一个整数 k ($1 \leq k \leq 50$) 。

Output

如果不存在合法分割, 输出 “No” (不输出引号) 。

否则首先输出一行 “Yes” (不输出引号) 。接下来 k 行每行输出六个由单个空格分割的数 x_1 , y_1 , x_2 , y_2 , x_3 和 y_3 表示 (x_1, y_1) , (x_2, y_2) 和 (x_3, y_3) 这三个点构成一个锐角三角形。请注意, 这 k 个锐角三角形必须是正方形的一个分割。

为了避免精度问题, 我们额外限制所有三角形顶点的坐标都必须是整数。可以证明, 对于本题所有可能的输入, 如果正方形可以被分割成 k 个锐角三角形, 则一定存在一个满足所有限制的分割。

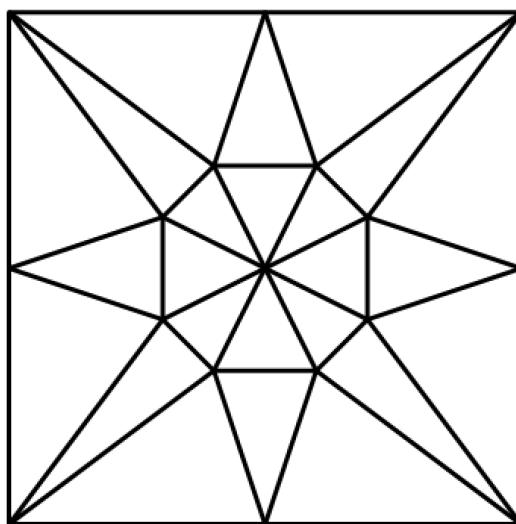
Examples

standard input
2
standard output
No

	standard input
24	standard output
Yes	
0 0 500000000 0 400000000 300000000	
1000000000 0 500000000 0 600000000 300000000	
0 0 0 500000000 300000000 400000000	
0 1000000000 0 500000000 300000000 600000000	
0 1000000000 500000000 1000000000 400000000 700000000	
1000000000 1000000000 500000000 1000000000 600000000 700000000	
1000000000 1000000000 1000000000 500000000 700000000 600000000	
1000000000 0 1000000000 500000000 700000000 400000000	
0 0 400000000 300000000 300000000 400000000	
0 500000000 300000000 400000000 300000000 600000000	
0 1000000000 300000000 600000000 400000000 700000000	
500000000 1000000000 400000000 700000000 600000000 700000000	
1000000000 1000000000 600000000 700000000 700000000 600000000	
1000000000 500000000 700000000 600000000 700000000 400000000	
1000000000 0 700000000 400000000 600000000 300000000	
500000000 0 400000000 300000000 600000000 300000000	
500000000 500000000 400000000 300000000 300000000 400000000	
500000000 500000000 300000000 400000000 300000000 600000000	
500000000 500000000 300000000 600000000 400000000 700000000	
500000000 500000000 400000000 700000000 600000000 700000000	
500000000 500000000 600000000 700000000 700000000 600000000	
500000000 500000000 700000000 600000000 700000000 400000000	
500000000 500000000 700000000 400000000 600000000 300000000	
500000000 500000000 600000000 300000000 400000000 300000000	

Note

下图展示了第二组样例数据中的分割方案。



Problem G. 邪恶铭刻

你迷失在森林深处，陪伴你的只有你的白鼬。它的初始攻击力为 1，是你唯一的初始野兽。

一条小道从眼前伸向远方。小道上有 n 个事件标志，每个标志属于以下事件之一：

- **卡牌选择**: 一位森林的住民加入了你的队伍。你获得了一只额外的野兽，它的初始攻击力为 1。
- **神秘石头**: 你被迫进行一次牺牲。你选择队伍中的两只野兽进行仪式：一只野兽将永远消失，而它的攻击力将加给另一只野兽。若你无法执行仪式，你将无法继续前进。
- **分岔路**: 你可以决定进行一次卡牌选择或神秘石头。你不能决定什么都不做。

当你穿过崎岖的路面时，事件将依次触发。在所有事件完成后，你的野兽的平均攻击力最大能达到多少？

Input

有多组测试数据。第一行输入一个整数 T 表示测试数据组数，对于每组测试数据：

第一行输入一个整数 n ($1 \leq n \leq 10^6$) 表示事件的数量。

第二行输入 n 个整数 a_1, a_2, \dots, a_n ($-1 \leq a_i \leq 1$)，其中 a_i 表示第 i 个事件标志的类型：1 表示卡牌选择，-1 表示神秘石头，0 表示分岔路。

保证所有数据 n 之和不超过 10^6 。

Output

每组数据输出一行。

如果无法完成所有事件，输出整数 -1。

否则可以证明答案是一个有理数 $\frac{p'}{q'}$ 。输出两个整数 p 和 q ，其中 $\frac{p}{q}$ 是 $\frac{p'}{q'}$ 的最简分数表示。

称 $\frac{p}{q}$ 是 $\frac{p'}{q'}$ 的最简分数表示，若 $\frac{p}{q} = \frac{p'}{q'}$ 且 p 和 q 的最大公约数为 1。

Example

standard input	standard output
6	3 2
7	3 1
1 1 1 -1 1 1 -1	-1
4	1 1
1 0 -1 0	2 1
4	-1
0 -1 -1 0	
1	
0	
2	
0 0	
1	
-1	

Note

对第一组样例数据解释如下：

事件	行动	野兽
1	获得额外的野兽	{1, 1}
1	获得额外的野兽	{1, 1, 1}
1	获得额外的野兽	{1, 1, 1, 1}
-1	选择攻击力为 1 和 1 的野兽	{2, 1, 1}
1	获得额外的野兽	{2, 1, 1, 1}
1	获得额外的野兽	{2, 1, 1, 1, 1}
-1	选择攻击力为 2 和 1 的野兽	{3, 1, 1, 1}

平均攻击力为 $\frac{3+1+1+1}{4} = \frac{6}{4} = \frac{3}{2}$ 。

对第二组样例数据解释如下：

事件	行动	野兽
1	获得额外的野兽	{1, 1}
0	进行卡牌选择并获得额外的野兽	{1, 1, 1}
-1	选择攻击力为 1 和 1 的野兽	{2, 1}
0	进行神秘石头并选择攻击力为 2 和 1 的野兽	{3}

平均攻击力为 $\frac{3}{1}$ 。

对第三组样例数据解释如下：

事件	行动	野兽
0	进行卡牌选择并获得额外的野兽	{1, 1}
-1	选择攻击力为 1 和 1 的野兽	{2}
-1	没有足够的野兽	失败

Problem H. 工厂重现

有一片由 n 座城市组成的王国。城市的编号从 1 到 n （含两端）且有 $(n - 1)$ 条道路连接各个城市。对于任意两座城市，居民们都可以沿着这些道路互相访问。

皇后最近决定建设 k 座新的工厂。为了防止污染，她规定每座城市最多只能建立一座工厂。

您作为皇家设计师，需要在规划建设的同时，求出两两工厂之间距离之和的最大值。

两座工厂之间的距离，即为两座工厂所在的两座城市之间的最短路径长度。路径的长度即为路径中所有边的长度之和。

Input

每个测试文件仅有一组测试数据。

第一行输入两个整数 n 和 k ($2 \leq n \leq 10^5$, $2 \leq k \leq n$) 表示城市的数量与新建工厂的数量。

对于接下来 $(n - 1)$ 行，第 i 行输入三个整数 u_i , v_i 和 w_i ($1 \leq u_i, v_i \leq n$, $u \neq v$, $1 \leq w_i \leq 10^4$) 表示有一条道路连接城市 u_i 与 v_i ，其长度为 w_i 。

Output

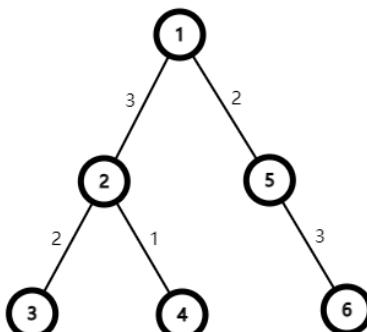
输出一行一个整数表示两两工厂之间距离之和的最大值。

Example

standard input	standard output
6 3 1 2 3 2 3 2 2 4 1 1 5 2 5 6 3	22

Note

样例数据解释如下。



可以选择在城市 3, 4 和 6 建立工厂。令 $d(i, j)$ 表示城市 i 与 j 之间的最短路径长度，则答案为 $d(3, 4) + d(3, 6) + d(4, 6) = 3 + 10 + 9 = 22$ 。

Problem I. 完美回文

给定长度为 n 的字符串 $S = s_0s_1 \cdots s_{n-1}$, 令 $f(S, d)$ 表示将 S 左移 d 次后获得的字符串。也就是说 $f(S, d) = s_{(d+0) \bmod n}s_{(d+1) \bmod n} \cdots s_{(d+n-1) \bmod n}$ 。称 S 为完美回文，若对于所有非负整数 d , $f(S, d)$ 都是回文串。

给定长度为 n 的仅由小写英文字母组成的字符串 $A = a_0a_1 \cdots a_{n-1}$, 您可以对 A 进行任意次以下操作（包括零次）：选择整数 i 满足 $0 \leq i < n$ 并将 a_i 改为任何小写英文字母。

求将 A 变为完美回文的最少操作次数。

称长度为 n 的字符串 $P = p_0p_1 \cdots p_{n-1}$ 是回文串，若对于所有 $0 \leq i < n$ 有 $p_i = p_{n-1-i}$ 。

Input

有多组测试数据。第一行输入一个整数 T 表示测试数据组数，对于每组测试数据：

第一行输入一个仅由小写英文字母构成的字符串 $a_0a_1 \cdots a_{n-1}$ ($1 \leq n \leq 10^5$)。

保证所有数据中字符串长度之和不超过 10^6 。

Output

每组数据输出一行一个整数表示将 A 变为完美回文的最少操作次数。

Example

standard input	standard output
2	2
abcb	0
xxx	

Note

对于第一组样例数据，可以将第一和第三个字符变为 ‘b’，这样字符串将变为 “bbbb”。容易发现对于所有非负整数 d , $f(\text{“bbbb”}, d) = \text{“bbbb”}$ 且 “bbbb” 是回文串，因此 “bbbb” 是完美回文。这些变化需要消耗 2 次操作，可以证明这是最少需要的操作次数。

对于第二组样例数据，“xxx” 已经是完美回文，因此无需任何操作。

Problem J. 完美匹配

给定一张包含 n 个顶点的无向图 (n 是偶数) 以及 n 个整数 a_1, a_2, \dots, a_n 。对于任意满足 $1 \leq i < j \leq n$ 的正整数 i 和 j , 若 $|i - j| = |a_i - a_j|$ ($|x|$ 表示 x 的绝对值) 则在无向图中顶点 i 和顶点 j 之间连一条无向边。显然, 这个无向图不包含自环和重边。

求该无向图的一个完美匹配, 或表明不存在完美匹配。

请回忆: 一张图的一个完美匹配为图中所有边的一个大小为 $\frac{n}{2}$ 的子集, 使得每一个顶点恰好作为这个子集中一条无向边的某个端点出现。

Input

有多组测试数据。第一行输入一个整数 T 代表测试数据组数。对于每组测试数据:

第一行输入一个偶数 n ($2 \leq n \leq 10^5$) 表示无向图顶点的数量。

第二行输入 n 个整数 a_1, a_2, \dots, a_n ($-10^9 \leq a_i \leq 10^9$)。

保证所有测试数据中 n 之和不超过 10^6 。

Output

对于每组数据, 若不存在完美匹配输出一行 “No” (不输出引号); 若存在完美匹配首先输出一行 “Yes” (不输出引号), 接下来输出 $\frac{n}{2}$ 行, 其中第 i 行输出两个由单个空格分隔的整数 u_i 和 v_i 表示完美匹配中的第 i 条边连接的两个顶点。若多种可行解, 输出任意一种。

Example

standard input	standard output
3	Yes
6	1 4
14 22 33 11 25 36	5 2
4	6 3
100 10 98 12	Yes
4	1 3
1 3 5 7	4 2
	No

Problem K. 堆里的 NaN

预备知识: NaN

NaN (Not a Number) 是 1985 年 IEEE 754 浮点数标准里引入的特殊浮点数。标准规定，当 **NaN** 与一个浮点数 x 进行比较时（ x 可以是正数，零，负数，甚至是 **NaN** 本身），应当返回以下结果。

比较	$\text{NaN} \geq x$	$\text{NaN} \leq x$	$\text{NaN} > x$	$\text{NaN} < x$	$\text{NaN} = x$	$\text{NaN} \neq x$
结果	False	False	False	False	False	True

预备知识: 堆

堆是一种数据结构，可以用具有特殊性质的序列表示。以下算法展示了如何将 n 个浮点数 a_1, a_2, \dots, a_n 按顺序加入小根堆 H 中，其中 H 是一个初始为空的序列。

以下算法中，用 h_i 表示 H 的第 i 个元素，用 $j/2$ 表示满足 $2x \leq j$ 的最大整数 x 。

Algorithm 1 Heapify

```

1: function HEAPIFY( $A$ )
2:   令  $H$  为一个空序列。
3:   for  $i \leftarrow 1$  to  $n$  do
4:     将  $a_i$  追加到  $H$  末尾。
5:      $j := i$ 
6:     while  $j > 1$  do
7:       if  $h_j < h_{j/2}$  then       $\triangleright$  请回忆，若  $h_j$  或  $h_{j/2}$  是 NaN，该表达式的结果为 false。
8:         交换  $h_j$  与  $h_{j/2}$ 。
9:          $j := j/2$ 
10:    else
11:      break
12:    end if
13:  end while
14: end for
15: return  $H$ 
16: end function

```

问题

给定一个整数 n ，考察这 n 个元素的排列：从 1 到 $(n - 1)$ 的所有整数（含两端），再加上一个额外的元素 **NaN**。称这 n 个元素的一个排列 P 是“堆序列”，若存在这 n 个元素的一个排列 Q 满足 $P = \text{HEAPIFY}(Q)$ 。

现在从这 n 个元素的所有排列中等概率随机选取一个（也就是说，一个特定的排列被选中的概率是 $\frac{1}{n!}$ ），求被选中的排列是堆序列的概率。

Input

有多组测试数据。第一行输入一个整数 T ($1 \leq T \leq 10^3$) 表示测试数据组数，对于每组测试数据：

第一行输入一个整数 n ($1 \leq n \leq 10^9$)。

Output

每组数据输出一行表示答案。

可以证明答案是一个有理数 $\frac{p}{q}$ 。为了避免精度问题，请输出整数 $(pq^{-1} \bmod M)$ 作为答案，其中 $M = 10^9 + 7$, q^{-1} 是满足 $qq^{-1} \equiv 1 \pmod{M}$ 的整数。

Example

standard input	standard output
5	1
1	666666672
3	55555556
7	596445110
10	3197361
20221218	

Note

对于第二组样例数据，有 4 个堆序列。

- $\{\text{NaN}, 1, 2\} = \text{HEAPIFY}(\{\text{NaN}, 1, 2\})$ 。
- $\{\text{NaN}, 2, 1\} = \text{HEAPIFY}(\{\text{NaN}, 2, 1\})$ 。
- $\{1, \text{NaN}, 2\} = \text{HEAPIFY}(\{1, \text{NaN}, 2\}) = \text{HEAPIFY}(\{2, \text{NaN}, 1\})$ 。
- $\{1, 2, \text{NaN}\} = \text{HEAPIFY}(\{1, 2, \text{NaN}\}) = \text{HEAPIFY}(\{2, 1, \text{NaN}\})$ 。

所以答案用有理数表示是 $\frac{4}{3!} = \frac{2}{3}$ 。因为 $3 \times 333333336 \equiv 1 \pmod{M}$ ，我们应该输出 $2 \times 333333336 \bmod M = 666666672$ 。

Problem L. 命题作文

有一张由 n 个点与 $(n - 1)$ 条边构成的无向连通图 G 。顶点编号从 1 到 n （含两端），第 i 条边连接顶点 i 与 $(i + 1)$ 。

接下来向图中依次加入额外 m 条边（加入后不删除）。每次加入一条额外边后，求从图中选择两条边 e 与 f 的方案数，满足如果边 e 与 f 同时被删除，图将变得不连通（即图中至少有两个连通块）。

请注意，先选择 e 再选择 f ，和先选择 f 再选择 e 被认为是同一种方案。

Input

有多组测试数据。第一行输入一个整数 T 表示数据组数，对于每组测试数据：

第一行输入两个整数 n 和 m ($1 \leq n, m \leq 2.5 \times 10^5$) 表示图的点数及增加的额外边数。

对于接下来 m 行，第 i 行输入两个整数 u_i 和 v_i ($1 \leq u_i, v_i \leq n$) 表示第 i 条额外边连接顶点 u_i 与 v_i 。

保证所有数据中 n 之和与 m 之和均不超过 2.5×10^5 。

Output

每组数据输出 m 行，第 i 行输出一个整数表示加入第 i 条额外边后的答案。

Example

standard input	standard output
3	6
4 3	5
2 4	6
4 2	21
3 3	24
7 3	10
3 4	15
1 2	12
1 7	3
6 4	2
1 3	
4 6	
2 5	
3 4	

Note

以下对第一组样例数据进行解释。

加入第一条额外边后，任意删除两条边都能将图变得不连通。因此答案为 6。

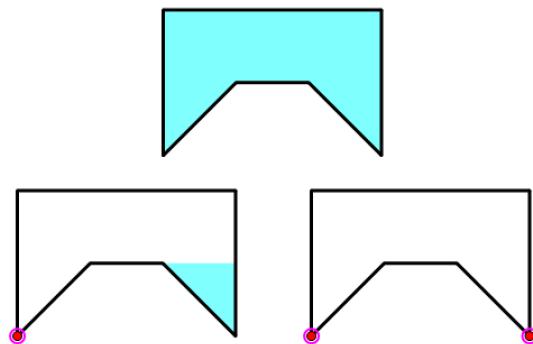
加入第二条额外边后，可以同时选择原始边 1 与任意另一条边，也可以同时选择原始边 2 与原始边 3。答案为 $4 + 1 = 5$ 。

加入第三条额外边后，可以同时选择原始边 1 与任意另一条边，也可以同时选择原始边 2 与原始边 3。答案为 $5 + 1 = 6$ 。

Problem M. 清空水箱

自来水厂最近建造了一种多边形水箱，水箱的厚度可以忽略不计。

为了启用水箱，工程师们准备在水箱上安装若干出水阀门。一个出水阀门可以被看作水箱上的一个点，当阀门打开时，水箱里的水会从阀门流出。



如上图所示，紫色的点代表阀门，而浅蓝色区域代表阀门全部打开后水箱内剩余的水。

作为总工程师的您需要知道，至少需要安装多少个出水阀门，才能在所有阀门同时打开后，让水箱里的水全部流出。

您可以认为水是一种理想流体且环境中不存在大气压，因此水总有流向更低处的趋势，即使位于水平面上也是如此。

Input

每个测试文件仅有一组测试数据。

第一行输入一个整数 n ($3 \leq n \leq 2 \times 10^3$) 表示多边形（也就是水箱的形状）的顶点数。

对于接下来 n 行，第 i 行输入两个整数 x_i 和 y_i ($|x_i|, |y_i| \leq 10^4$) 表示多边形第 i 个顶点的坐标。顶点按逆时针顺序给出。

给定的多边形是简单多边形。也就是说，多边形的顶点两两不同，且除了相邻边存在公共顶点外，不存在两条边有公共点。不保证相邻边不共线。

Output

输出一行一个整数表示清空水箱至少需要多少个出水阀门。

Examples

standard input	standard output
6 0 0 1 1 2 1 3 0 3 2 0 2	2
8 4 4 0 4 0 2 1 2 2 2 2 0 3 0 4 0	1
7 1 0 3 4 0 3 1 2 2 3 1 1 0 2	2

Note

对于第一组样例数据，在 $(0, 0)$ 与 $(3, 0)$ 安装两个出水阀门即可清空水箱。

对于第二组样例数据，在 $(3, 0)$ 安装一个出水阀门即可清空水箱。事实上，只要把该出水阀门安装在满足 $2 \leq x \leq 4$ 的 $(x, 0)$ 即可。

对于第三组样例数据，在 $(1, 0)$ 与 $(1, 2)$ 安装两个出水阀门即可清空水箱。

