

“规律未来杯” 2026 年广东工业大学 ACM 程序设计竞赛

广东工业大学 ACM 集训队

2026 年 3 月 22 日



试题列表

A	魔法石法阵
B	凸包
C	消失的元素
D	收纳收纳袋
E	周长
F	瞭望塔
G	fufu 购买计划
H	死亡变速
I	凹包
J	最远的彼此
K	共享单车
L	PPIIIGG
M	微软大战代码

共 13 题

Problem A. 魔法石法阵

输入文件: standard input

输出文件: standard output

小 P 是一位魔法师学徒，他打算用 a 颗黑魔法石和 b 颗白魔法石打造出一个神奇的法阵。

他将每颗魔法石看作一个节点，每次施法可以在两颗魔法石之间连一条无向边。这个法阵需要满足：

- 所有魔法石构成的图是**连通**的；
- 对于每个节点，与其直接相连的黑魔法石数量和白魔法石数量**相等**。

由于施法会消耗魔力，小 P 希望在满足上述条件的前提下，使用的边数**尽可能少**。

请你判断是否存在这样的法阵，若存在，则输出最少边数并给出任意一种可行的连边方案；否则报告不可能。

输入格式

第一行包含一个整数 T ($1 \leq T \leq 10^4$)，表示测试数据的组数。

接下来 T 行，每行两个整数 a, b ($0 \leq a, b \leq 10^6$ ，且 $a + b \geq 1$)，分别表示黑魔法石和白魔法石的数量。

对于单个测试点，有 $\sum(a + b) \leq 10^6$ 。

输出格式

对于每组数据：

如果存在满足条件的法阵，则第一行输出一个整数 m ，表示最少需要的边数。接下来 m 行，每行输出两个整数 u, v ($1 \leq u, v \leq a + b$)，表示在编号为 u 和 v 的魔法石之间连一条边。其中，编号 1 到 a 为黑魔法石，编号 $a + 1$ 到 $a + b$ 为白魔法石。如果存在多种可行方案，输出任意一种即可。

如果不存在这样的法阵，则输出一行 -1。

注意：你给出的方案必须连通，且不能包含重边或自环。

样例

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

提示说明

对于第一组数据，仅有单颗黑魔法石，无需连边即可视为连通且平衡。

对于第三组数据，可以证明最少需要 4 条边，此时每颗魔法石均恰好与 1 颗黑魔法石和 1 颗白魔法石相邻。

Problem B. 凸包

输入文件: standard input

输出文件: standard output

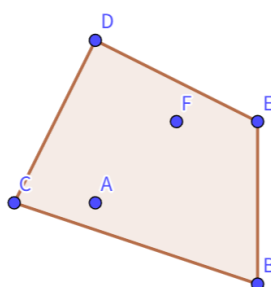
在二维平面上给定 n 个互不重合的整点 $P_i = (x_i, y_i)$ 。题目保证对于任意三个互不相同的点，它们均不共线。

你需要从这些点中选出两个非空且不相交的点集 A 和 B ，使得由点集 A 构成的凸包[†]与由点集 B 构成的凸包完全不相交（即两个凸多边形的内部及边界均无公共点）。

注意：你不需要使用所有的点。只要满足 $A \cap B = \emptyset$ 且两个凸包在几何上完全分离即可。本题保证在给定的条件下一定有解。如果存在多种方案，输出任意一种即可。

[†] 给定点集 S 的凸包是指包含 S 中所有点的最小凸多边形。直观地说，如果你在点集的所有点上钉上钉子，并用一根橡皮筋框住所有钉子，橡皮筋所围成的区域即为该点集的凸包。

注意：凸包至少要包含三条边，下图展示了一个点集对应的凸包：



点集 $\{A, B, C, D, E, F\}$ 对应的凸包为多边形 $BCDE$

输入格式

第一行一个整数 n ($6 \leq n \leq 500$)。

接下来 n 行，每行两个整数 x_i, y_i ($|x_i|, |y_i| \leq 10^9$)，表示编号为 i 的点 P_i 的坐标。

输出格式

如果存在多种方案，输出任意一种：

第一行输出两个整数 k, m ，表示集合 A, B 的大小。

第二行输出 k 个互不相同的整数，表示属于集合 A 的点的编号。

第三行输出 m 个互不相同的整数，表示属于集合 B 的点的编号。

样例

standard input	standard output
6	3 3
0 0	1 3 2
1 0	6 4 5
0 1	
2 2	
1 2	
2 1	

Problem C. 消失的元素

输入文件: standard input

输出文件: standard output

本题是一道多次运行的交互题（通信题）。你的提交会被运行两次，第一次运行的输出将被按规则输入给第二次运行。

Alice 和 Bob 正在合作玩一个数字猜谜游戏。每局游戏开始前，系统会公开两个整数 n 和 m ($n > m$)，并秘密生成一个包含 n 个互不相同正整数的集合 S ，其中所有元素都在 $[1, n + m]$ 范围内。

Alice 能够看到完整的集合 S ，但她只能通过一条窄带信道向 Bob 发送信息。由于带宽限制，她不能发送整个 S ，而只能从中删除恰好一个元素，然后将剩下的 $n - 1$ 个元素组成的集合 S' 发送给 Bob。通信协议要求任何集合在传输时都必须按元素严格递增的顺序排列成一个序列。

Bob 收到 S' （以递增序列形式）后，需要唯一确定 Alice 删除了哪个元素。

请你设计一种合作策略，使得无论 S 是什么，Bob 都能根据 S' 准确推断出被删除的元素。

交互协议

本题采用“运行两次”的评测模式。你的程序将被先后执行两次，每次执行对应一个角色。第一次扮演 Alice，第二次扮演 Bob。

第一次运行（扮演 Alice）

输入的第一行包含一个字符串 Alice。

第二行包含一个整数 T ($1 \leq T \leq 1000$)，表示测试用例的个数。

接下来对于每个测试用例：

1. 第一行读入两个整数 n, m ($1 \leq m < n \leq 10^4$)。
2. 第二行读入 n 个互不相同的正整数，表示集合 S 的所有元素。这些元素已经按严格递增顺序给出，且每个元素满足 $1 \leq x \leq n + m$ 。
3. 接下来，你应该立即输出一行，包含 $n - 1$ 个由空格分隔的正整数，表示你选择发送给 Bob 的集合 S' 的元素，并立刻刷新标准输出流。必须按严格递增的顺序输出。

第二次运行（扮演 Bob）

输入的第一行包含一个字符串 Bob。

第二行包含一个整数 T ($1 \leq T \leq 1000$)，表示测试用例的个数。

接下来对于每个测试用例：

1. 第一行读入两个整数 n, m ($1 \leq m < n \leq 10^4$)。
2. 第二行读入 $n - 1$ 个互不相同的正整数，表示 Alice 发送过来的集合 S' 的元素。这些元素已经按严格递增顺序给出，且每个元素满足 $1 \leq x \leq n + m$ 。
3. 接下来，你应该立即输出一行一个整数，表示你推断出的被 Alice 删除的元素，并立刻刷新标准输出流。

注意：由于网络波动，第二次运行时测试用例的到达顺序可能与第一次运行时 Alice 处理的顺序不同。你的程序应当按第二次运行接收到的顺序依次处理并输出答案。

如果输出的格式不符合要求（如元素个数不对、不是严格递增、元素并不能通过 S 删去一个数获得），你将获得 **Wrong Answer** 判题结果。

保证所有测试用例的 n 之和不超过 10^4 。

需要注意的是，每次输出完毕后，你都需要打印换行符并**刷新缓冲区**！你可以使用如下语句来刷新缓冲区：

- 对于 C/C++: `fflush(stdout);`
- 对于 C++: `std::cout << std::flush;` 或者使用 `std::endl`
- 对于 Java: `System.out.flush();`
- 对于 Python: `sys.stdout.flush();`

样例

standard input	standard output
Alice 2 3 1 1 2 3 4 2 1 3 4 5	 1 3 1 3 4
Bob 2 4 2 1 3 4 3 1 1 3	 5 2

提示说明

这两个示例旨在演示同一测试上的两次运行。

- 第一次运行有两个测试用例：
 - 第一个： $n = 3, m = 1, S = \{1, 2, 3\}$ 。假设 Alice 删除了 2，则输出 $S' = \{1, 3\}$ 。
 - 第二个： $n = 4, m = 2, S = \{1, 3, 4, 5\}$ 。假设 Alice 删除了 5，则输出 $S' = \{1, 3, 4\}$ 。
- 第二次运行接收到的测试用例顺序被打乱：
 - 第一个测试用例： $n = 4, m = 2, S' = \{1, 3, 4\}$ ，对应原第二个用例，被删除的元素是 5。
 - 第二个测试用例： $n = 3, m = 1, S' = \{1, 3\}$ ，对应原第一个用例，被删除的元素是 2。
- 因此第二次输出依次为 5 和 2。

注意： 样例中的删除方案只是一种可能，实际你需要设计一种策略使得 Bob 能唯一确定被删元素，这里仅示意格式。

Problem D. 收纳收纳袋

输入文件: standard input

输出文件: standard output

某个平行世界的 Minecraft 中，一个收纳袋有两个参数，大小 a 和容量 b ，而且收纳袋里可以装收纳袋！



各种各样的收纳袋

具体来说，一个收纳袋只能装一个收纳袋，而且被装入的收纳袋的实际大小必须不超过外层收纳袋的容量。若一个收纳袋装入了另一个收纳袋，则该收纳袋的实际大小会变为自身大小加上内部收纳袋的实际大小；否则，其实际大小就是自身大小。

某一天，史蒂夫在宝箱中发现了 n 个收纳袋，其中第 i 个收纳袋的大小为 a_i ，容量为 b_i 。

但坏消息是，史蒂夫突然发现自己只剩下一个空背包格子了！也就是说，所有收纳袋中只能有一个直接放在背包里，其余的收纳袋都必须通过嵌套的方式装进去！

史蒂夫有 q 个问题。每次给定一个区间 $[L, R]$ ，表示只考虑编号在该区间内的所有收纳袋。史蒂夫想知道：

1. 一个背包格子能否放下所有这些收纳袋。
2. 如果不能，史蒂夫决定把这些收纳袋的容量全部增加 x ，也就是对所有收纳袋执行 $b_i \leftarrow b_i + x$ 。他想知道，最少需要增加多少 x ，才能让这些收纳袋最终可以只占用一个背包格子。

注意，史蒂夫的询问并不会真的对收纳袋的容量做出修改。

输入格式

第一行输入一个整数 t ($1 \leq t \leq 1000$)，表示测试用例数量。

对于每个测试样例：

第一行输入两个整数 n, q ($1 \leq n, q \leq 2 \times 10^5$)。

第二行输入 n 个整数 $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq 10^9$), 表示每个收纳袋的大小。

第三行输入 n 个整数 $b_1, b_2, \dots, b_n$ ($1 \leq b_i \leq 10^9$), 表示每个收纳袋的容量。

接下来 q 行, 每行两个整数 L, R ($1 \leq L \leq R \leq n$), 表示一次询问的区间。

保证所有测试用例的 n 之和不超过 2×10^5 , q 之和不超过 2×10^5 。

输出格式

对于每个询问:

如果不需要增加容量即可嵌套, 则输出一行 **Yes**。

否则输出两行, 第一行输出 **No**, 第二行输出最小的 x 。

样例

standard input	standard output
1	No
5 3	3
3 2 5 1 2	No
4 3 2 4 1	3
1 3	Yes
2 5	
3 4	

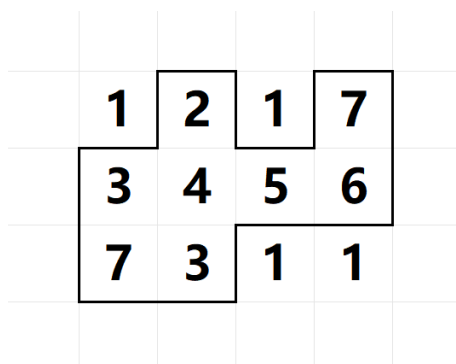
Problem E. 周长

输入文件: standard input

输出文件: standard output

塑料有一个大小为 $n \times m$ 的矩阵，每个格子上有一个正整数权值 $a_{i,j}$ 。他想要圈定一块封闭图形，要求其周长恰好为 C ，且封闭图形内部的格子权值之和尽量大，请你输出一个和的最大可能值。

封闭图形是在矩阵网格线上由若干条边首尾相连组成的一条不自交闭合折线，其内部恰好围住一块连通区域。



一个封闭图形

输入格式

第一行：一个正整数 T ($1 \leq T \leq 10^5$)，表示该测试点内测试样例的组数。接下来 T 组，每组格式如下：

- 第一行：三个正整数 n, m, C ，依次表示矩阵行数、列数，以及封闭图形的周长。满足 $1 \leq n, m \leq 3 \times 10^6$ ， $4 \leq C \leq 2(n + m)$ ，且 C 为偶数。
- 接下来 n 行：每行 m 个正整数，表示该矩阵的权值 $a_{i,j}$ ($1 \leq a_{i,j} \leq 10^7$)。

数据约束：每个测试点内，所有组的 n 之和、 m 之和、 $n \times m$ 之和均不超过 3×10^6 。

输出格式

对于每组测试样例，输出一行一个整数表示答案。

样例

standard input	standard output
2	7
2 2 4	13
5 2	
4 7	
3 3 8	
1 4 3	
5 1 5	
3 4 1	

Problem F. 瞭望塔

输入文件: standard input

输出文件: standard output

塔顶的怪物俯视大地，你抬头望去，只觉得它们不该站得这么高。



有 n 个瞭望塔，第 i 个瞭望塔有等级 k_i 和高度 h_i ，观测范围为 $k_i \cdot h_i^2$ 。

现在你可以不断发动攻击。每次攻击可以选择一个瞭望塔，使它的高度 h_i 减少 1（至多减少到 0）。

你的目标是让所有瞭望塔的观测范围之和不超过 X ，请你求出最少需要攻击多少次。

输入格式

第一行输入一个整数 T ($1 \leq T \leq 1000$)，表示数据组数。

对于每组数据：

- 第一行输入两个整数 n, X ($1 \leq n \leq 10^6$, $0 \leq X \leq 10^{18}$)。
- 接下来 n 行，每行输入两个整数 k_i, h_i ($1 \leq k_i, h_i$)，表示第 i 个瞭望塔的等级和高度。

保证每组数据均满足 $\sum_{i=1}^n k_i h_i^2 \leq 10^{18}$ 。

保证所有数据组中 n 的总和不超过 10^6 。

输出格式

对于每组数据，输出一行一个整数，表示最少的攻击次数。

样例

standard input	standard output
2	0
3 20	1
1 3	
2 2	
1 1	
4 50	
3 3	
1 4	
2 2	
5 1	

Problem G. fufu 购买计划

输入文件: standard input

输出文件: standard output

fufu 软乎乎地看着你，你的大脑已经停止思考，只剩下一个念头：买它！



你有价值 $1, 2, \dots, n$ 的硬币各一枚，想要购买价值为 m 的 fufu，你需要选出若干枚硬币，价值之和恰好为 m 。

可恶的老板给出了额外的 n 个要求，第 i 个要求是：

- 价值小于等于 i 的硬币至多使用 a_i 枚。

请你判断老板的要求能否满足，以及如何支付硬币才能满足老板的要求。

输入格式

第一行输入一个整数 T ($1 \leq T \leq 1000$)，表示数据组数。

对于每组数据：

- 第一行输入一个整数 n, m ($1 \leq n \leq 10^6, 1 \leq m \leq \frac{n(n+1)}{2}$)。
- 第二行包含 n 个整数 a_i ($0 \leq a_i \leq i$)。

保证所有数据组中 n 的总和不超过 10^6 。

输出格式

对于每组数据：

- 如果可以，先输出一行一个整数 k ，然后升序输出一行 k 个互不相同的硬币价值；
- 否则输出一行一个整数 -1 。

样例

standard input	standard output
1	2
4 5	2 3
0 2 2 3	

Problem H. 死亡变速

输入文件: standard input
输出文件: standard output

在音游的世界中，音乐的节奏忽快忽慢，不仅考验玩家的手速和反应力，还考验其对节拍与时间的精准把控。



小 L 正在开发一款音游。他希望引入这一机制，因此需要在 **谱面时间（拍）** 与 **物理时间（秒）** 之间进行高精度转换。

这两者之间的转换依赖于 **BPM**，若当前 BPM 为 b ，则每一拍的所经过的物理时间恒为 $\frac{60}{b}$ 秒。

为了实现复杂的变速效果，谱面文件中存储了一个 BPM 列表。列表包含 n 个时间点，每个时间点 (s_i, b_i) 表示从第 s_i 拍开始，BPM 变为 b_i ，直到下一个时间点为止（如果是最后一个时间点，则持续到无穷大）。

你需要处理 q 次查询：

- **B** x ：给定谱面时间 x （拍），计算并输出其对应的物理时间（秒）
- **S** x ：给定物理时间 x （秒），计算并输出其对应的谱面时间（拍）

输入格式

第一行输入一个整数 n ($1 \leq n \leq 5 \times 10^5$)。

接下来 n 行，每行包含两个整数 s_i, b_i ($0 \leq s_i \leq 10^9, 1 \leq b_i \leq 10^9$)。

保证：

- $s_1 = 0$
- 对任何 $1 \leq i < n$ ，有 $s_i < s_{i+1}$

接下来一行输入一个整数 q ($1 \leq q \leq 5 \times 10^5$)。

接下来 q 行，每行包含一个查询：

- $B\ x$ 或 $S\ x$ ，其中 x ($x \geq 0$) 为实数

输出格式

对于每次查询，输出一个实数表示转换结果。

你的答案将被判定为正确，当且仅当你的输出与标准答案的相对误差或绝对误差不超过 10^{-6} 。

样例

standard input	standard output
1	0.000000000000
0 120	60.000000000000
4	120.000000000000
B 0	145.000000000000
B 120	
S 60	
S 72.5	
3	41.454545454545
0 165	169.400000000000
180 147	232.650389610390
208 165	438.466178571429
6	0.014190000000
B 114	76.883076066790
S 61.6	
B 636.36	
S 160.689	
S 0.00516	
B 207.9999	

Problem I. 凹包

输入文件: standard input

输出文件: standard output

在平面上给定 n 个整点，其中第 n 个点为特殊点 $O = (0, 0)$ 。

你可以从这 n 个点中选出若干个点作为顶点，并按顺序连接形成一个多边形。

我们称一个由点列 $P_1, P_2, \dots, P_m$ ($m \geq 3$) 构成的多边形是**简单多边形**，当且仅当：

- 所有顶点两两不同；
- 边 $P_1P_2, P_2P_3, \dots, P_{m-1}P_m, P_mP_1$ 组成一条不自交的闭合折线；
- 任意两条非相邻边没有公共点；任意两条相邻边仅在公共端点处相交。

现在要求构造的简单多边形满足：

- 多边形的顶点均来自给定点集（可以选取点 O ）；
- 存在一条以 O 为端点的射线，使得该射线与多边形的**内部没有公共点**。该射线允许与多边形的边界相交、重合或经过顶点。

请你在所有满足条件的简单多边形中，求其**最大可能面积的两倍**，可以证明该值为整数。

输入格式

第一行包含一个整数 T ($1 \leq T \leq 1000$)，表示测试用例的组数。

对于每组测试数据：

- 第一行包含一个整数 n ($3 \leq n \leq 10^6$)；
- 接下来 $n - 1$ 行，每行包含两个整数 x_i, y_i ($-10^9 \leq x_i, y_i \leq 10^9$)，表示第 i 个点的坐标。

第 n 个点固定为 $O = (0, 0)$ 。

保证所有点两两不同，且所有点不全在一条直线上。

保证所有测试用例的 n 之和不超过 10^6 。

输出格式

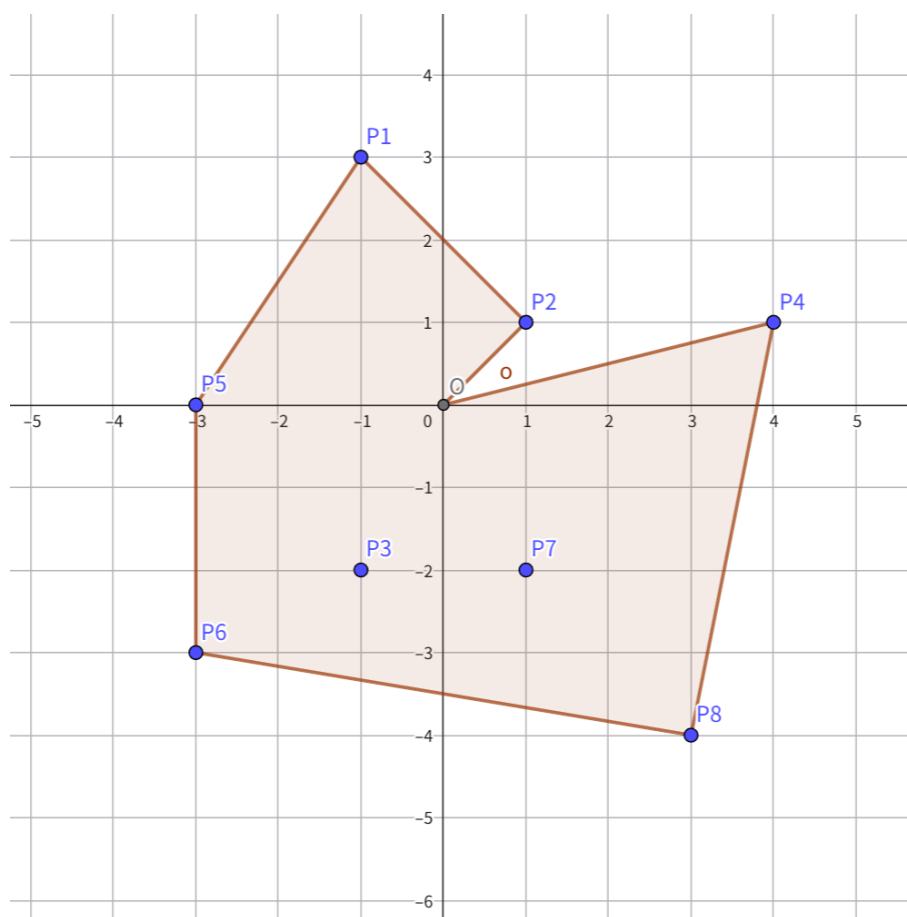
对于每组测试数据，输出一个整数，表示满足条件的简单多边形的最大可能面积的两倍。

样例

standard input	standard output
3	1
3	4
1 0	62
0 1	
4	
1 0	
0 2	
-1 -1	
9	
-1 3	
1 1	
-1 -2	
4 1	
-3 0	
-3 -3	
1 -2	
3 -4	

提示说明

对于第三个测试用例，满足条件的最大面积的简单多边形如下图所示：



Problem J. 最远的彼此

输入文件: standard input

输出文件: standard output

有些点，注定要站在棋盘的两端。

你看不见它们的位置，却能听见它们之间最遥远的回声。



本题是一道交互题。选手的程序可以向交互器进行若干询问，最后进行回答。具体交互方式请参考下方交互协议部分。

现在有一个 $n \times n$ 的未知的 01 矩阵 a ($a_{i,j} \in \{0,1\}$)，保证矩阵中至少存在一个 1。

你需要通过不超过 $8n$ 次询问，确定矩阵中两个值为 1 的位置之间的最大曼哈顿距离。形式化地，你需要输出：

$$\max_{a_{x_1,y_1}=1, a_{x_2,y_2}=1} (|x_1 - x_2| + |y_1 - y_2|)$$

每次询问的方式如下：

- 选择一个连续子矩阵，其左上角坐标为 (x_1, y_1) ，右下角坐标为 (x_2, y_2) ；
- 交互器会返回该子矩阵内所有元素的按位或结果，即：

$$\bigvee_{i=x_1}^{x_2} \bigvee_{j=y_1}^{y_2} a_{i,j}$$

其中 $\vee$ 表示按位或运算。

请注意，交互器是非自适应的。换句话说，矩阵 a 在任何查询之前就已经固定。

输入格式

每个测试包含多个测试用例。

第一行输入一个整数 t ($1 \leq t \leq 1000$)，表示测试用例数量。

每个测试用例包括一行一个整数 n ($1 \leq n \leq 10^4$)，表示矩阵大小。

保证 n 之和不超过 10^4 。

交互协议

每个测试用例的程序交互应从读取整数 n 开始。

要进行查询，请按以下格式输出一行：

? x_1 y_1 x_2 y_2

其中 $1 \leq x_1 \leq x_2 \leq n$, $1 \leq y_1 \leq y_2 \leq n$ ，表示询问子矩阵 $[x_1, x_2] \times [y_1, y_2]$ 的按位或。然后你会收到一个整数 v ：

- $v = 0$ 表示该子矩阵内所有元素均为 0；
- $v = 1$ 表示该子矩阵内至少存在一个 1。

要报告最终答案，请按以下格式输出一行：

! d

请注意，总查询次数不能超过 $8n$ 次，但报告最终答案不计入查询次数限制。在输出答案后，您的程序应继续处理下一个测试用例，若无更多测试用例则终止程序。

需要注意的是，每次输出完毕后，你都需要打印换行符并刷新缓冲区！你可以使用如下语句来刷新缓冲区：

- 对于 C/C++: `fflush(stdout);`
- 对于 C++: `std::cout << std::flush;` 或者使用 `std::endl`
- 对于 Java: `System.out.flush();`
- 对于 Python: `sys.stdout.flush();`

样例

standard input	standard output
2	
3	
	? 1 1 1 1
1	
	? 3 3 3 3
1	
	! 4
4	
	? 1 2 4 3
0	
	? 2 1 3 4
0	
	! 6

提示说明

对于样例，可能的矩阵如下

第一个测试用例（ $n = 3$ ）可以是：

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

此时两个 1 在 (1,1) 和 (3,3)，距离为 4。

第二个测试用例（ $n = 4$ ）可以是：

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

此时两个 1 在 (1,4) 和 (4,1)，距离为 6。

Problem K. 共享单车

输入文件: standard input
输出文件: standard output

塑料每天需要骑共享单车通勤。

共享单车由两家公司 A 和 B 运营，使用前需要购买对应公司的套餐：支付 K 元后，从购买当天起连续 L 天（含购买日）内可以无限次使用该公司的单车。两家公司的套餐互不通用，但价格相同。

已知未来 n 天每天可用的单车公司类型（每天一定有车可骑）：

- A: 只有 A 公司有车；
- B: 只有 B 公司有车；
- C: A 、 B 两家公司均有车。

塑料可以多次购买套餐（每次支付 K 元），使得每天都能骑上当天可用的车（即对于 A 天必须有 A 套餐， B 天必须有 B 套餐， C 天至少有一家公司的套餐）。

塑料想知道最少需要支付的总金额。

输入格式

第一行一个整数 T ($1 \leq T \leq 10^5$)，表示测试用例的组数。

每组测试用例包含两行：

- 第一行：三个整数 n, L, K ($1 \leq L \leq n \leq 2 \times 10^6$, $1 \leq K \leq 10^8$)，分别表示总天数、套餐有效天数、套餐价格。
- 第二行：一个长度为 n 的字符串 s ，由字符 A 、 B 、 C 组成，表示每天的车型。

所有测试用例的 n 总和不超过 2×10^6 。

输出格式

对于每组测试用例，输出一行一个整数，即最少需要支付的总金额。

样例

standard input	standard output
2	3
6 2 1	3
AABBAA	
7 3 1	
AACBCAC	

Problem L. PPIIIGG

输入文件: standard input
输出文件: standard output

在超级猪星上，超级小猪研究员发现了一个仅由字符 P、I、G 组成的字符串 S 。现在需要统计其中有多少个 **超级 PIG** 序列。



超级小猪

一个 **超级 PIG** 序列 需要满足以下条件：

- 它由 a 个 P、 b 个 I、 c 个 G 依次拼接而成，即形如 $\underbrace{PP \cdots P}_a \underbrace{II \cdots I}_b \underbrace{GG \cdots G}_c$ 。
- $a, b, c \geq 1$ 。
- 数量关系满足： $a + c - 1 \leq b$ 。

你需要计算在给定的字符串 S 中，有多少个子序列是 **超级 PIG** 序列。答案对 $10^9 + 7$ 取模。

对于一个序列，删除任意个任意位置的字符后得到的新非空序列是原序列的子序列。如 abcde 包含 abcde、ace 等子序列。

输入格式

第一行包含一个整数 T ($1 \leq T \leq 10^4$)，表示测试数据组数。

接下来每组数据：

- 第一行包含一个整数 n ($1 \leq n \leq 1000$)，表示字符串长度。
- 第二行包含一个长度为 n 的字符串 S ，仅由大写字母 P、I、G 组成。

保证所有测试数据的 n^2 之和不超过 10^7 。

输出格式

对于每组数据，输出一行一个整数，表示满足条件的子序列数量对 $10^9 + 7$ 取模后的结果。

样例

standard input	standard output
2	3
4	45
PIIG	
7	
PPIIIIGG	

提示说明

第一组样例中，PIIG 中所有可能的超级 PIG 子序列为 PIG（取位置 1,2,4）、PIG（取位置 1,3,4）和 PIIG（取全部 4 个字符），共 3 个。

Problem M. 微软大战代码

输入文件: standard input

输出文件: standard output

小 P 最近迷上了一款名为《微软大战代码》的游戏。这款游戏历史悠久，发展出了无数个版本，而每个版本又可能衍生出自己的后续版本。

为了理清这些版本的关系，小 P 将它们整理成了一棵版本树：

- 版本 1 是根节点（原版），没有前作。
- 对于 $i \geq 2$ ，版本 i 的直接前作为 x_i ($1 \leq x_i < i$)，且每条边 (x_i, i) 上有一个评价字母 ch_i ，为小写 a 到 z，字母越小表示改版越精良。

对于任意一个版本 u ，其**评价串**定义为从根节点（版本 1）到版本 u 的路径上所有边的字母按顺序拼接而成的字符串。根节点的评价串为空串。

小 P 决定根据以下规则对所有版本进行排序：

- 按评价串的**字典序升序**排序，也就是字典序越小的版本，排名越靠前。
- 若两个版本的评价串完全相同，则**版本序号较小**的排在前面。

注：字典序指通常的字符串比较方式：从左到右逐个字符比较，第一个不同字符的较小者字典序更小；若一个字符串是另一个的前缀，则较短的字符串字典序更小。

输入格式

第一行包含一个整数 T ($1 \leq T \leq 10^4$)，表示测试数据组数。

对于每组测试数据：

- 第一行包含一个整数 n ($1 \leq n \leq 10^5$)，表示版本总数。
- 接下来 $n-1$ 行，每行描述一个版本 i ($i = 2, 3, \dots, n$)，包含一个整数 x_i 和一个小写字母 ch_i ，表示版本 i 的前作为 x_i ，对应边的评价为 ch_i 。

保证 $x_i < i$ ，因此输入顺序与版本编号一致，且形成一棵以 1 为根的有根树。

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，输出一行 n 个整数，为按照上述规则排序后的版本序号，相邻序号间用一个空格隔开。

样例

standard input	standard output
2	1 2 3
3	1 3 2 5 6 4
1 b	
2 a	
6	
1 z	
1 a	
2 b	
2 a	
5 c	