



BCPC 2025 The 20th Beihang University
Collegiate Programming Contest

第二十届北京航空航天大学程序设计竞赛

决赛

2025 年 12 月 20 日 10:00 — 14:00

题目概况

题号	题目名	时间限制	空间限制
A	签到	1 秒	2 GiB
B	排列之差	1 秒	2 GiB
C	Music In My Mind	1 秒	2 GiB
D	平衡矩阵	1 秒	2 GiB
E	MAX — MIN	2 秒	2 GiB
F	Floral CatastrophE	4 秒	2 GiB
G	很短的题面	2 秒	2 GiB
H	Purple Haze	4 秒	2 GiB
I	Mystic Bounds	2 秒	2 GiB
J	Nine Circles	0.999 秒	999 MiB





时间限制: 1 秒 输入文件: 标准输入
空间限制: 2 GiB 输出文件: 标准输出

T0xel 需要填 n 个签到表, 其中每个签到表有若干个不同的项目 (例如姓名, 学号等), 每种项目用一个正整数标记。T0xel 讨厌重复的填写相同的内容。初始 T0xel 填写每个签到表的怒气值为 1, 在填写完一个签到表后, 其他未填写的签到表每有一个与该签到表相同的项目, 填写该未填写的签到表的怒气值会永久增加 1 (换句话说, 如果有 x 个相同的项目, 则怒气值增加 x)。T0xel 想规划一个填写签到表的顺序, 使得填写完所有签到表的总怒气值之和最小。

输入格式

输入共 $n + 1$ 行。

第一行, 两个整数 n, m ($1 \leq n, m \leq 10^5$), 表示签到表的数量和不同项目的数量。

接下来 n 行, 每行首先是一个整数 x ($1 \leq x \leq m$), 表示该签到表共有 x 个项目, 接下来是 x 个不同的正整数 a_i ($1 \leq a_i \leq m$), 表示每个项目的编号。

保证测试用例中 x 的总和不超过 3×10^5 。

输出格式

输出一个整数, 表示最小的总怒气值。

样例

标准输入	标准输出
3 2 1 1 2 1 2 1 2	5
5 4 3 1 3 4 2 2 4 3 1 2 4 2 1 3 3 2 3 4	20

提示

对于第一个样例, 依次填写签到表 1, 3, 2, 怒气值总和为 $1 + 1 + 3 = 5$ 。



排列之差

时间限制: 1 秒 输入文件: 标准输入
空间限制: 2 GiB 输出文件: 标准输出

给出一个正整数 n , 你需要构造一个 n 的排列 $P = (p_1, p_2, \dots, p_n)$, 满足:

- $\max_{1 \leq i \leq n} \{|p_i - p_{i \bmod n+1}|\} - \min_{1 \leq i \leq n} \{|p_i - p_{i \bmod n+1}|\}$ 的值是所有 n 的排列 P 中最小的。

请你输出任意一个满足条件的 n 的排列。

一个 n 的排列是一个包含从 1 到 n 的每个整数恰好一次的数组, 顺序任意。

输入格式

第一行, 一个正整数 n ($2 \leq n \leq 10^5$)。

输出格式

一行, 一个 n 的排列, 表示答案。

样例

标准输入	标准输出
3	1 3 2
2	1 2

提示

对于样例一的输出, $|p_1 - p_2| = |1 - 3| = 2$, $|p_2 - p_3| = |3 - 2| = 1$, $|p_3 - p_1| = |2 - 1| = 1$, 最大值与最小值之差为 $2 - 1 = 1$ 。

可以证明不存在使得最大值与最小值之差为 0 的排列。



时间限制: 1 秒 输入文件: 标准输入
空间限制: 2 GiB 输出文件: 标准输出

Nerovix 正在用随机播放功能听一个歌单, 歌单里有 n 首歌, 第 i 首歌的时长是 a_i 秒。常用音乐软件的随机播放功能会把当前歌单内的所有歌打乱顺序, 然后 **按照这个固定的乱序不断循环** 播放这 n 首歌。

Nerovix 开始随机播放后, 从不暂停, 从不切歌, 从不点进度条跳着听, 且只听这个歌单。请问他第 k 秒 (下标从 1 开始, 也就是说第 1 秒正在听第 1 首歌的第 1 秒) 可能正在听哪些歌?

输入格式

第一行, 两个整数 n, k ($1 \leq n \leq 20, 1 \leq k \leq 2 \times 10^9$), 表示歌单中的歌曲数量和需要询问的时间点 k (从开始随机播放起的第 k 秒)。

第二行, n 个整数 $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq 10^8$), 第 i 个整数表示第 i 首歌的长度。

输出格式

输出一行, 包含若干个升序排列的整数, 表示所有第 k 秒可能正在播放的歌曲。

样例

标准输入	标准输出
5 37 3 6 2 7 1	1 2 3 4

提示

在样例中, 若随机播放器按照, 例如, 第 2, 4, 5, 3, 1 首歌的顺序, 依次循环播放这 5 首歌, 那么第 37 秒正在播放第 1 首歌 (的第 2 秒)。同理, 第 37 秒也有可能正在播放第 2, 3, 4 首歌。但无论播放顺序如何, 此时都不可能正在播放第 5 首歌。



平衡矩阵

时间限制: 1 秒 输入文件: 标准输入
空间限制: 2 GiB 输出文件: 标准输出

对于一个 r 行 c 列的非负整数矩阵, 如果其中所有整数之和恰好为 $r \times c$, 那么 Serval 称这个矩阵是平衡的。

现在给定一个 n 行 m 列的非负整数矩阵 A , 其中第 i 行第 j 列的整数记作 $A_{i,j}$ 。Serval 想知道, 给定的矩阵 A 中有多少个子矩阵是平衡的。

A 的子矩阵是从 A 中选取连续的若干行和连续的若干列, 这些行和列相交处的元素按照在 A 中的顺序构成的矩阵。对于两个子矩阵, 如果得到它们所选取的行和列不同, 则认为这两个子矩阵是不同的。

例如, 矩阵 $A = \begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix}$ 共有 9 个子矩阵:

$$\begin{bmatrix} 1 \end{bmatrix}, \begin{bmatrix} 0 \end{bmatrix}, \begin{bmatrix} 1 \end{bmatrix}, \begin{bmatrix} 2 \end{bmatrix}, \begin{bmatrix} 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix}$$

其中以红色标记的 5 个子矩阵是平衡的。

输入格式

第一行, 两个正整数 n, m ($1 \leq n, m \leq 500$), 分别表示矩阵的行数与列数。

接下来 n 行中的第 i 行 ($1 \leq i \leq n$) 包含 m 个非负整数 $A_{i,1}, A_{i,2}, \dots, A_{i,m}$ ($0 \leq A_{i,j} \leq 4$), 表示给定的矩阵。

输出格式

输出共一行, 一个整数, 表示给定矩阵 A 中平衡子矩阵的数量。

样例

标准输入	标准输出
2 2 1 0 1 2	5
4 7 0 1 0 3 0 0 3 3 0 1 1 1 1 0 0 1 2 1 0 1 1 0 0 0 0 4 0 0	71



MAX — MIN

时间限制: 2 秒 输入文件: 标准输入
空间限制: 2 GiB 输出文件: 标准输出

小 C 有一个长度为 n 的数组 $a = [a_1, a_2, \dots, a_n]$, 其中的元素均是正整数。

小 C 会对数组 a 重复进行减法操作, 直到数组 a 为空。在一次减法操作中, 小 C 会选择数组 a 中的最大值 (如果有多个, 则选择下标最大的), 并将其减去 a 中的最小值。如果相减得到 0, 则将 0 从数组中移除。

例如, 数组 $a = [5, 2]$ 需要 5 次操作变成空数组 $[]$:

$$[5, 2] \xrightarrow{-2} [3, 2] \xrightarrow{-2} [1, 2] \xrightarrow{-1} [1, 1] \xrightarrow{-1} [1] \xrightarrow{-1} []$$

其中每次操作选择的最大值以下划线表示。

小 C 想知道需要至少多少次操作才能将 a 变为空。可以证明, 有限次操作后 a 必可变为空。

输入格式

第一行, 一个正整数 n ($1 \leq n \leq 10^6$), 表示数组 a 的长度。

第二行, n 个正整数 $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq 10^6$), 表示数组 a 中的元素。

输出格式

一行, 一个正整数, 表示将 a 变为空所需要的操作次数。

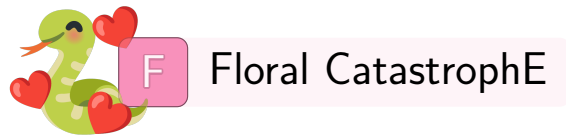
样例

标准输入	标准输出
2 5 2	5
6 1 1 4 5 1 4	16

提示

样例 1 即是题目描述中给出的例子。

对于样例 2, 可以发现每次操作时的最小值均是 1, 因此所需操作次数为 $1 + 1 + 4 + 5 + 1 + 4 = 16$ 。



时间限制: 4 秒 输入文件: 标准输入
空间限制: 2 GiB 输出文件: 标准输出

Nerovix 给了你 n 个节点, 第 i 个节点有属性 (a_i, b_i) 。你可以给任意两个节点之间建立无向边, 连接第 i 和第 j 个点会产生一条边权为 $\frac{|a_i - a_j|}{b_i + b_j}$ 的无向边。

Nerovix 希望你建立一些边使得这 n 个节点互相连通, 同时最小化图中的最大边权。你需要告诉他这个最大边权的最小值。

输入格式

第一行, 一个整数 n ($2 \leq n \leq 10^5$), 表示节点个数。

接下来 n 行, 第 i 行包含两个整数 a_i, b_i ($1 \leq a_i, b_i \leq 10^6$), 表示第 i 个节点的属性。

输出格式

输出一行, 一个浮点数, 表示最大边权的最小值。

你的答案与标准答案的相对误差或绝对误差小于等于 10^{-9} 时被认为是正确的。也就是说, 如果标准答案是 a , 你的答案是 b , 那么当 $\frac{|a-b|}{\max(1, |a|)} \leq 10^{-9}$ 时, 你的答案将被视为正确。

样例

标准输入	标准输出
3 1 1 7 5 10 10	0.8181818181818181111111114514

提示

在样例中, 建立第 1 个点和第 3 个点以及第 2 个点和第 3 个点之间的边是最优的。你可以验证样例中给出的结果满足误差限制。



时间限制: 2 秒 输入文件: 标准输入
空间限制: 2 GiB 输出文件: 标准输出

给定一个正整数 n , 和 m 个正整数 $a_1, \dots, a_m$, 求最大正整数的 x 使得 $\frac{n!}{(\prod_{i=1}^m a_i!)^x}$ 为整数。

输入格式

输入共 2 行。

第一行为两个正整数 n, m ($1 \leq n \leq 10^{18}, 1 \leq m \leq 100$)。

第二行为 m 个正整数 a_i ($1 \leq a_i \leq 10^6$)。

保证 $\sum_{i=1}^m a_i \leq n$ 。

输出格式

输出一个整数 x 。

样例

标准输入	标准输出
5 2 2 2	3
11451 4 19 19 8 10	11401

提示

对于样例 1, 当 $x = 3$ 时, $\frac{5!}{2!2!3!} = \frac{120}{24} = 5$ 。当 $x = 4$ 时, $\frac{5!}{2!2!4!} = \frac{120}{96} = 1.25$, 因此答案为 3。



H Purple Haze

时间限制: 4 秒 输入文件: 标准输入
空间限制: 1 GiB 输出文件: 标准输出

在失落的世界里, 回忆化为了无数的音符残片。演奏每首乐曲都会产生残片, 它们散落在一条贯穿正负无穷的整数时间轴上。听一曲 **Purple Haze**, 其旋律就隐藏在这些残片之中。

Insight, 全知全能的神, 将在 **Purple Haze** 的演奏中随意点击 n 次, 产生 n 块残片。残片将精确地出现在 n 个给定的整数时间戳 $\{t_1, t_2, \dots, t_n\}$ 上。

在 **Purple Haze** 中, 世界以固定的整数 m 为长度, 将无限的时间轴划分为无数个首尾相接、长度为 m 的“判定区间”:

$$\{\dots, [-m, 0), [0, m), [m, 2m), \dots\}$$

Insight, 无所不能的神, 可以自由设定一个整数“全局偏移” offset , 该偏移会平移所有的判定区间。 offset 可以是任意正数、负数或零。如果一个 offset 被选定, 那么时间轴将被划分为如下的无穷区间集合:

$$\{\dots, [\text{offset} - m, \text{offset}), [\text{offset}, \text{offset} + m), [\text{offset} + m, \text{offset} + 2m), \dots\}$$

即, 形如 $[\text{offset} + m \cdot k, \text{offset} + m \cdot (k + 1))$ 的全体区间, 其中 k 为任意整数。

演奏开始后, 世界会按时间顺序, 对上述的每一个判定区间依次进行判定:

- 判定成功 (**Pure**): 如果一个判定区间 $[\text{offset} + m \cdot k, \text{offset} + m \cdot (k + 1))$ 包含了至少一个给定的残片 t_i , 即存在某个 i 使得 $\text{offset} + m \cdot k \leq t_i < \text{offset} + m \cdot (k + 1)$, 则该区间被视为判定成功。
- 判定失败 (**Lost**): 如果区间内不包含任何 t_i , 则判定失败。

Insight 的目标是调整 offset , 让连续判定成功的区间数达到最大值。

不幸的是, 你被 **Insight** 盯上了。现在祂命令你立刻找出最佳的 offset , 使得按时间顺序连续判定成功的最大次数 (连击次数) 尽量大。你只需要输出可能的最大连击次数。

输入格式

第一行包含两个整数 n 和 m ($1 \leq n \leq 2 \times 10^5, 1 \leq m \leq 10^{18}$), 分别表示残片的数量和判定区间的长度。

第二行包含 n 个整数 $t_1, t_2, \dots, t_n$ ($-10^{18} \leq t_i \leq 10^{18}$), 表示 **Insight** 产生的 n 块残片各自出现的时间戳。 t_i 升序排列, 可能相同。

输出格式

输出一个整数, 表示在所有可能的 offset 中, 可以达到的最大连续判定成功次数。

样例

标准输入	标准输出
5 5 1 6 7 12 19	5
6 16 3 10 15 27 28 34	3



时间限制: 2 秒 输入文件: 标准输入
空间限制: 2 GiB 输出文件: 标准输出

Nerovix 给了你一个平面上的圆，圆上等间距地排布着 n 个点，依次标号为 $1, 2, \dots, n$ 。每个点都被染上了一种颜色，其中第 i 个点的颜色是 c_i 。你可以在这个平面上添加直线，但添加的直线不可以经过这 n 个点中的任何一个。

Nerovix 希望任意两个相同颜色的点都在某条直线的两侧。那么，你最少添加多少条直线就能满足他的条件？

输入格式

第一行，一个整数 n ($1 \leq n \leq 913947$)，表示圆上的点数。

第二行， n 个整数 $c_1, c_2, \dots, c_n$ ($1 \leq c_i \leq 10^9$)，第 i 个整数表示第 i 个点的颜色。

输出格式

输出一行，一个整数，表示为满足条件所需要添加的最小直线条数。

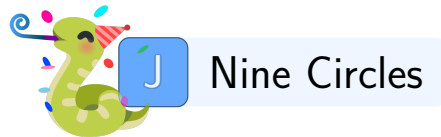
样例

标准输入	标准输出
6 3 1 2 3 1 3	2
3 1 2 3	0

提示

对于第一个样例，只需要两条直线就可以满足条件。一种可行的方案是：第一条直线分割第 5, 6 个点与其他点，第二条直线分割第 3, 4, 5 个点与其他点。若只使用一条直线，则颜色为 3 的三个点至少有两个点在直线同侧，无论如何也无法满足条件。

对于第二个样例，不需要添加任何直线，因为不存在某两个点颜色相同。



时间限制: 0.999 秒 输入文件: 标准输入
空间限制: 999 MiB 输出文件: 标准输出

Nerovix 给了你二维平面上的 n 个圆, 第 i 个圆的圆心是 (x_i, y_i) , 半径是 r_i , 可能存在某些圆的圆心、甚至圆心和半径都相同。你有一个放缩装置, 允许你选择任意非负实数 k , 并把每个圆的半径都变为原来的 k 倍, 同时保持每个圆的圆心坐标不变。注意, 你可以选择 $k = 0$, 这会导致所有圆退化为点。

Nerovix 希望存在一条直线与所有圆有交, 有交的定义为圆与直线相交或相切。如果圆退化为点, 则有交定义为直线经过点。为了使用一次放缩装置来满足他的要求, 你选取的 k 的最小值是多少?

输入格式

第一行, 一个整数 n ($n \in \{9, 99, 999, 1999\}$), 表示圆的个数。

接下来 n 行, 第 i 行包含三个整数 x_i, y_i, r_i ($0 \leq x_i, y_i \leq 9999, 1 \leq r_i \leq 9999$), 表示第 i 个圆的圆心和半径。

输出格式

输出一行, 一个非负实数, 表示为满足条件最小的 k 。

你的答案与标准答案的相对误差或绝对误差小于等于 10^{-9} 时被认为是正确的。也就是说, 如果标准的答案是 a , 你的答案是 b , 那么当 $\frac{|a-b|}{\max(1, |a|)} \leq 10^{-9}$ 时, 你的答案将被视为正确。

样例

标准输入	标准输出
9 0 2 1 1 0 1 2 2 1 3 0 1 4 2 1 5 0 1 6 2 1 7 0 1 8 2 1	0.999999999

提示

在样例中, 选择直线 $y = 1$ 时可以取到最小的 $k = 1$ 。你可以验证样例中给出的结果满足误差限制。