《算法竞赛进阶指南》2022 年新版（大象出版社）· 勘误

算法竞赛进阶指南于 2022 年更换了出版社，重新进行了排版设计，并对第 0x00 章的内容进行了一定程度的更新，其他章节也有少量修订。请认准当前唯一正版，大象出版社出版的算法竞赛进阶指南。

勘误与光盘下载——勘误 PDF 已上传至 GitHub，同时 GitHub 也可以下载到最新的配套光盘内容。有任何疑问或建议，可以开一个 Issue 或 Pull request。

视频讲解——本书部分章节新增了视频讲解，详细剖析全部知识点、例题和程序实现，不定期直播讲解习题。动态规划：20 小时，图论：30 小时，平均每小时仅收费 5~6 元，视频可永久回放，评论区有答疑。地址：https://www.acwing.com/activity/content/35/ 和 https://www.acwing.com/activity/content/41/。

在线评测——本书例题与习题上传至了 AcWing，大家可以在线提交。

【0x12 队列】【例题 AcWing133 蚯蚓】

对单调性证明的修正：

如果 $q=0$，即蚯蚓不会变长，那么本题相当于维护一个集合，支持查询最大值、删除最大值、插入新的值。这是二叉堆（第 0x17 节）或 STL set（第 0x71 节）能够完成的基本操作，时间复杂度约为 $O(m\log n)$。

当 $q>0$ 时，除了最大值拆成的两个数之外，集合中的其他数都会增加 q。设最大值为 x。我们不妨认为产生了两个大小为 $\lfloor px\rfloor-q$ 和 $x-\lfloor px\rfloor-q$ 的新数，然后再把整个集合都加上 q。这与之前的操作是等价的。

于是，我们可以维护一个变量 $delta$ 表示整个集合的“偏移量”，集合中的数加上 $delta$ 是它的真实数值。起初，$delta=0$。对于每一秒：

1. 取出集合中的最大值 x，令 $x=x+delta$；
2. 把 $\lfloor px\rfloor-delta-q$ 和 $x-\lfloor px\rfloor-delta-q$ 插入集合；
3. 令 $delta=delta+q$。

重复上述步骤 m 轮，即可得到最终集合中所有数的值。然而，本题中 m 的范围过大，我们需要一种线性算法来更快地求解。一般情况下，维护集合的最大值不可能做到线性，我们不得不大胆猜想：集合中新加入的数值必然满足某种规律，从而让我们有更高效的计算方法。这启发我们对切割蚯蚓新产生的几个算式进行比较。

先从 $q=0$ 的情况入手。设 x_1,x_2 为非负整数且 $x_1\ge x_2$。

注意到 $0<p<1$ 是非负常数，显然有 $\lfloor px_1\rfloor\ge\lfloor px_2\rfloor$。如果我们能进一步证明 $x_1-\lfloor px_1\rfloor\ge x_2-\lfloor px_2\rfloor$，那么不仅从集合中取出的数是单调递减的，新产生的两类数值也分别随着时间单调递

减。这种单调性就为更快地维护集合最大值创造了条件。我们来尝试证明这个命题。

由 $x_1 - x_2 \geq p(x_1 - x_2)$ 移项得 $(x_1 - x_2) + px_2 \geq px_1$。两边各加上取整符号，不等式仍成立，即 $\lfloor (x_1 - x_2) + px_2 \rfloor \geq \lfloor px_1 \rfloor$。在加法算式中，非负整数可以自由移入、移出取整符号而不改变式子的值，故有 $(x_1 - x_2) + \lfloor px_2 \rfloor \geq \lfloor px_1 \rfloor$。再次移项即可得到 $x_1 - \lfloor px_1 \rfloor \geq x_2 - \lfloor px_2 \rfloor$。命题得证。

拓展到 $q > 0$ 的情况。当 $x_1 \geq x_2$ 且 $0 < p < 1$ 时，直接比较大小关系容易发现 $px_1 + q \geq px_2 + pq$ 以及 $px_2 \leq p(x_2 + q)$，从而推出：

$$\lfloor px_1 \rfloor + q = \lfloor px_1 + q \rfloor \geq \lfloor px_2 + pq \rfloor = \lfloor p(x_2 + q) \rfloor$$

$$x_1 - \lfloor px_1 \rfloor + q \geq x_2 - \lfloor px_2 \rfloor + q \geq x_2 + q - \lfloor p(x_2 + q) \rfloor$$

上面两个不等式的意义是：若 x_1 在 x_2 之前被取出集合，则在一秒以后，x_1 分成的两个数 $\lfloor px_1 \rfloor + q$ 和 $x_1 - \lfloor px_1 \rfloor + q$ 分别不小于 $x_2 + q$ 分成的两个数 $\lfloor p(x_2 + q) \rfloor$ 和 $x_2 + q - \lfloor p(x_2 + q) \rfloor$。换言之，不仅从集合中取出的数是单调递减的，新产生的两类数值也分别随着时间单调递减。

我们可以建立三个队列 A, B, C，共同构成需要维护的集合。队列 A 保存初始的 n 个数，从大到小排序。队列 B 保存每秒新产生的 $\lfloor px \rfloor$ 那一段数值。队列 C 保存每秒新产生的 $x - \lfloor px \rfloor$ 那一段数值。起初队列 B, C 为空，新产生的数从队尾插入。根据之前的结论，B, C 单调递减。因此，每个时刻集合中最大的数就是队列 A, B, C 的三个队首之一。再配合集合的偏移量 $delta$，整个算法的时间复杂度为 $O(m + n \log n)$。