CTS2024 倾诉 (Confide)

ltst

2025 年 1 月 18 日

简要题意

给定一个长度为 n 的正整数序列 $a_1, a_2, \cdots, a_n$。进行不超过 k 次操作，每次操作选择 $1 \le p \le n-1$，将 a_{p+1} 赋值为 $a_{p+1} + \frac{1}{2}a_p$，然后将 a_p 赋值为 0。求操作后序列最大值的最小可能值。输出答案乘 2^n 的二进制表示。

$1 \le n \le 2 \times 10^4$，$1 \le a_i \le 10^6$，$1 \le k \le 10^9$。

吐槽

$\log^2$ 跑 2×10^4 要跑 2s 还不知道卡没卡满写没写对我请问呢。现在的数据经过了十几份代码的交叉验证应该是对的，但是标算还是可能有细节锅，例如数据可能有边界情况没有覆盖到，同时所有点的单个测试数据的 n 都是一样的，所以可能有一些清空的锅。如果发现了锅请联系我 qwq。

n^4

最大值最小，二分答案。注意到答案是 2^nV 级别的，所以需要 $n+\log V$ 次二分。给定 mid，我们需要判断是否能在 k 次操作内将序列所有元素的值减小到 mid 以下。

n^4

最大值最小，二分答案。注意到答案是 $2^n V$ 级别的，所以需要 $n+\log V$ 次二分。给定 mid，我们需要判断是否能在 k 次操作内将序列所有元素的值减小到 mid 以下。

分析操作的结构。注意到无论如何操作，最终序列中每个非 0 的数都一定是从一段区间传过来的，这一结论可以归纳证明。于是我们可以使用若干个三元组 $\{(l_i, r_i, t_i)\}_{i=1}^{k}$ 描述一个终态，其表示最终 t_i 位置的值是 $a_{l_i \sim r_i}$ 传过来的，即这个位置的值是 $\sum_{j=l_i}^{r_i} a_j 2^{j-t_i}$。

n^4

三元组集合确定了答案，故我们考虑固定一个三元组集合 $\{(l_i, r_i, t_i)\}_{i=1}^{k}$，判断其能否通过题设操作得到。
假设 t_i 是单调增的，可以发现以下所有条件构成了三元组集合能被操作得到的充要条件：

- $l_i \le r_i \le t_i (1 \le i \le k)$；
- $l_1 = 1$，$r_k = n$，$l_i = r_{i-1} + 1 (2 \le i \le k)$。

n^4

三元组集合确定了答案，故我们考虑固定一个三元组集合 $\{(l_i, r_i, t_i)\}_{i=1}^{k}$，判断其能否通过题设操作得到。
假设 t_i 是单调增的，可以发现以下所有条件构成了三元组集合能被操作得到的充要条件：

- $l_i \le r_i \le t_i (1 \le i \le k)$;
- $l_1 = 1$，$r_k = n$，$l_i = r_{i-1} + 1 (2 \le i \le k)$。

当以上所有条件都满足的时候，最少的操作步数为 $\sum_{i=1}^{k} t_i - l_i$，通过按照 t_i 从大到小、从 l_i 到 $t_i - 1$ 依次操作。

n^4

考虑按照 t_i 从大到小依次加入三元组。设 $f_{p,q}$ 表示当前加入的所有三元组中最靠前的三元组的 $l_i = p, t_i = q$ 且满足以上条件时，$\sum t_j - l_j$ 的最小值。转移枚举 $l_i \le t_{i-1} < t_i$ 以及 $l_{i-1} < l_i$。直接做是 n^5 的，因为还有一个 n 的大数比较。

n^4

考虑按照 t_i 从大到小依次加入三元组。设 $f_{p,q}$ 表示当前加入的所有三元组中最靠前的三元组的 $l_i = p, t_i = q$ 且满足以上条件时，$\sum t_j - l_j$ 的最小值。转移枚举 $l_i \le t_{i-1} < t_i$ 以及 $l_{i-1} < l_i$。
直接做是 n^5 的，因为还有一个 n 的大数比较。
首先注意到 t_{i-1} 相同的转移是相同的，可以先对 $f_{l_i,\star}$ 做一个后缀 min，再从每个 t_{i-1} 的位置转移，这样优化掉一个 n。
其次 l_{i-1} 的最远转移点可以二分得到，压位高精之后约等于少了一个 n。这样就做到了 n^3 单次 DP。

一个简单的优化

以上 DP 还有优化的空间。一个与正解有关的优化是，注意到 $r_i - t_i > \log V + 1$ 是不优的，这是因为把 t_i 减 1 后

$$\sum_{j=1}^{r_i} a_j 2^{j-t_i+1} \leq \sum_{j=\log V+1}^{n} 2^{-j} \times V < 1,$$

而 a_n 是正整数所以答案至少是 1。
每次选择 t_i 最小的 $r_i - t_i > \log V + 1$ 的三元组，其 t_i 总是能减 1，得到一个步数更小且最大值不会变大的方案。因此我们可以把 $f_{p,q}$ 的第二维 q 限制在 $[p, p + \log V]$ 内，单次 DP 复杂度变为 $n^2 \log V$。

压缩二分的 n

以上的做法的三个 n 分别来自二分、DP 状态和高精度比较。我们先说明如何把二分的 n 改成 $\log$。

压缩二分的 n

以上的做法的三个 n 分别来自二分、DP 状态和高精度比较。我们先说明如何把二分的 n 改成 $\log$。
注意到答案一定取到某个三元组 (l, r, t) 上，而三元组的数量是 n^3 的。如果我们可以直接在三元组上进行二分，就可以做到 $\log n$ 的二分了，但是我们不太能对三元组排序。

压缩二分的 n

以上的做法的三个 n 分别来自二分、DP 状态和高精度比较。我们先说明如何把二分的 n 改成 $\log$。

注意到答案一定取到某个三元组 (l, r, t) 上，而三元组的数量是 n^3 的。如果我们可以直接在三元组上进行二分，就可以做到 $\log n$ 的二分了，但是我们不太能对三元组排序。

于是我们使用随机二分的技巧：给定 L, R，我们在所有 $L < (l, r, t) < R$ 的三元组 (l, r, t) 中随机摇一个作为 mid。这样做期望的判断次数依然是 $\log n$ 级别的。

压缩二分的 n

以上的做法的三个 n 分别来自二分、DP 状态和高精度比较。我们先说明如何把二分的 n 改成 $\log$。

注意到答案一定取到某个三元组 (l,r,t) 上，而三元组的数量是 n^3 的。如果我们可以直接在三元组上进行二分，就可以做到 $\log n$ 的二分了，但是我们不太能对三元组排序。

于是我们使用随机二分的技巧：给定 L,R，我们在所有 $L<(l,r,t)<R$ 的三元组 (l,r,t) 中随机摇一个作为 mid。这样做期望的判断次数依然是 $\log n$ 级别的。

统计所有满足 $L<(l,r,t)<R$ 的三元组，可以先枚举 r 再枚举 t 然后二分满足条件的 l 的范围。注意到 $t-r>\log V$ 的三元组一定小于 L（因为 $L\geq 1$），所以直接统计的复杂度是 $O(n^2\log V)$ 的，合计复杂度 $O(n^2\log n\log V)$。

压缩高精度比较的 n

接下来我们再把高精度比较的 n 变成 polylog。

压缩高精度比较的 n

接下来我们再把高精度比较的 n 变成 polylog。
注意到我们所有用到高精度比较的问题都形如："给定一个三元组 (l_1, r_1, t_1)，找到最小的 l_2 满足 $(l_2, r_2, t_2) \le (l_1, r_1, t_1)$ 或 $(l_2, r_2, t_2) < (l_1, r_1, t_1)$"。
二分 l_2 之后，比较两个大数，本质上是在比较它们的二进制表示作为两个字符串的字典序。故考虑二分两个大数的 LCP，并使用哈希判断在 LCP 位置是否相等。

压缩高精度比较的 n

接下来我们再把高精度比较的 n 变成 polylog。
注意到我们所有用到高精度比较的问题都形如："给定一个三元组 (l_1, r_1, t_1)，找到最小的 l_2 满足 $(l_2, r_2, t_2) \leq (l_1, r_1, t_1)$ 或 $(l_2, r_2, t_2) < (l_1, r_1, t_1)$"。
二分 l_2 之后，比较两个大数，本质上是在比较它们的二进制表示作为两个字符串的字典序。故考虑二分两个大数的 LCP，并使用哈希判断在 LCP 位置是否相等。
假设我们有了一个 $O(1)$ 计算哈希值的程序，那么在二分出两个三元组的 LCP 位置 L 之后，查询两者哈希值在 $L+1$ 位置的截断的差。这个值要么是 1，要么是 -1，分别对应大于和小于的情况。

计算三元组在某一位上的截断

具体地，我们需要计算的是：给定 (l, r, t) 和 LCP 位置 x，计算 $\lfloor \sum_{j=l}^{r} a_j 2^{j-t+x} \rfloor$ 的值。由于是哈希当然要对大质数取模。

计算三元组在某一位上的截断

具体地，我们需要计算的是：给定 (l,r,t) 和 LCP 位置 x，计算 $\lfloor\sum_{j=l}^{r} a_j 2^{j-t+x}\rfloor$ 的值。由于是哈希当然要对大质数取模。
不妨假设 $l<t-x<r$，此时贡献分为两部分：

- $t-x\sim r$ 的贡献，其值为 $\sum_{j=t-x}^{r} a_j 2^{j-t+x}$，不需要下取整。直接维护 $a_j 2^j$ 的前缀和然后相减得到区间哈希值。

计算三元组在某一位上的截断

具体地，我们需要计算的是：给定 (l,r,t) 和 LCP 位置 x，计算 $\lfloor\sum_{j=l}^{r} a_j 2^{j-t+x}\rfloor$ 的值。由于是哈希当然要对大质数取模。

不妨假设 $l<t-x<r$，此时贡献分为两部分：

- $t-x\sim r$ 的贡献，其值为 $\sum_{j=t-x}^{r} a_j 2^{j-t+x}$，不需要下取整。直接维护 $a_j 2^j$ 的前缀和然后相减得到区间哈希值。
- $l\sim t-x-1$ 的贡献，这里需要下取整。考虑计算 $f_{p,q}=\lfloor\sum_{i=1}^{p-q} a_{p-i}2^{-i}\rfloor$，那么我们需要得到的是 $f_{t-x,l}$ 的值。

计算 f

直接计算 $f_{p,q}$ 是 $O(n^2)$ 的。但注意到 $q < p - \log V$ 时，$q \sim p - \log V - 1$ 的贡献为 $\sum_{i=\log V+1}^{p-q} a_{p-i} 2^{-i} \leq 1$。因此 $f_{p,1\sim p-\log V}$ 只有至多两种取值，且它们相差 1。

计算 f

直接计算 $f_{p,q}$ 是 $O(n^2)$ 的。但注意到 $q < p - \log V$ 时，$q \sim p - \log V - 1$ 的贡献为 $\sum_{i=\log V+1}^{p-q} a_{p-i}2^{-i} \le 1$。因此 $f_{p,1\sim p-\log V}$ 只有至多两种取值，且它们相差 1。
因此我们只需要计算出 $f_{p,p-\log V\sim p}$ 以及 g_p 表示 $f_{p,1\sim p-\log V}$ 的转折点即可。这容易通过 $f_{p-1,\star}$ 和 g_{p-1} 推出。
预处理 f 和 g 的复杂度为 $O(n\log V)$。在预处理之后，然后我们可以 $O(1)$ 得到三元组在某一位上的截断。

现在的复杂度是多少?

在摇 mid 和 DP 的过程中，我们都需要做 $O(n\log V)$ 次二分操作。每一次操作中，我们需要先二分 l，再二分 LCP。外层还有一个 $\log n$ 的二分次数，因此总复杂度为 $O(n\log^3 n\log V)$。以上算法的常数非常非常的大，四个 $\log$ 很可能跑不过 $n^2\log^2$。

现在的复杂度是多少?

在摇 mid 和 DP 的过程中，我们都需要做 $O(n \log V)$ 次二分操作。每一次操作中，我们需要先二分 l，再二分 LCP。外层还有一个 $\log n$ 的二分次数，因此总复杂度为 $O(n \log^3 n \log V)$。

以上算法的常数非常非常的大，四个 log 很可能跑不过 $n^2 \log^2$。

接下来我们更精细地实现以上所有过程，最终做到 $O(n \log n(\log n + \log V))$。

更精细的三元组二分

回忆我们的三元组二分问题："给定 (l_1, r_1, t_1)，找到最小的 l_2 满足 $(l_2, r_2, t_2) \le (l_1, r_1, t_1)$ 或 $(l_2, r_2, t_2) < (l_1, r_1, t_1)$"。
每一次先二分 l_2 再二分 LCP 显然有冗余判断。考虑一个更精细的二分方法。

更精细的三元组二分

回忆我们的三元组二分问题："给定 (l_1, r_1, t_1)，找到最小的 l_2 满足 $(l_2, r_2, t_2) \leq (l_1, r_1, t_1)$ 或 $(l_2, r_2, t_2) < (l_1, r_1, t_1)$"。
每一次先二分 l_2 再二分 LCP 显然有冗余判断。考虑一个更精细的二分方法。
首先，在之后的描述中，我们认为 (l, r, t) 是一个数的同时也是一个长度为 $\log V + 1 + (t - l)$ 的字符串，其最靠前的字符为 $\log V$ 位的权值，最靠后的字符为 2^{l-t} 位的权值。也就是说，我们不给更低位补 0。
另外，特判掉 $l_2 = 1$ 合法或 $l_2 = r_2$ 不合法的情况，此时 l_2 是一个非平凡的值。

二分 LCP 最大值

接下来我们二分 $\max_{l_2\leq r_2} LCP((l_1,r_1,t_1),(l_2,r_2,t_2))$ 对应的位权。不妨假设其为 2^x。我们只考虑 $x\leq 0$ 的情况，即二分右边界为 1。

二分 LCP 最大值

接下来我们二分 $\max_{l_2 \le r_2} LCP((l_1, r_1, t_1), (l_2, r_2, t_2))$ 对应的位权。不妨假设其为 2^x。我们只考虑 $x \le 0$ 的情况，即二分右边界为 1。
首先，由于长度要求，我们需要 $l_2 \le r_2 - x$。此时我们需要判断是否存在一个 l_2 满足 $\lfloor (l_2, r_2, t_2) \times 2^x \rfloor = \lfloor (l_1, r_1, t_1) \times 2^x \rfloor$，那么 $r_2 + x \sim r_2$ 的贡献总是存在且对应着不需要下取整的情况。
直接使用之前的哈希结果统计这一部分取值。

二分 LCP 最大值

接下来我们二分 $\max_{l_2 \le r_2} LCP((l_1, r_1, t_1), (l_2, r_2, t_2))$ 对应的位权。不妨假设其为 2^x。我们只考虑 $x \le 0$ 的情况，即二分右边界为 1。
首先，由于长度要求，我们需要 $l_2 \le r_2 - x$。此时我们需要判断是否存在一个 l_2 满足 $\lfloor (l_2, r_2, t_2) \times 2^x \rfloor = \lfloor (l_1, r_1, t_1) \times 2^x \rfloor$，那么 $r_2 + x \sim r_2$ 的贡献总是存在且对应着不需要下取整的情况。直接使用之前的哈希结果统计这一部分取值。
$l_2 \sim r_2 - x - 1$ 的部分对应 f 数组，也就是说我们此时希望找到一个 l_2 满足 f_{r_2+x,l_2} 等于某个固定的值。直接在 f 数组上二分是 $\log\log V$ 的。

二分 LCP 最大值

接下来我们二分 $\max_{l_2 \le r_2} LCP((l_1, r_1, t_1), (l_2, r_2, t_2))$ 对应的位权。不妨假设其为 2^x。我们只考虑 $x \le 0$ 的情况，即二分右边界为 1。
首先，由于长度要求，我们需要 $l_2 \le r_2 - x$。此时我们需要判断是否存在一个 l_2 满足 $\lfloor (l_2, r_2, t_2) \times 2^x \rfloor = \lfloor (l_1, r_1, t_1) \times 2^x \rfloor$，那么 $r_2 + x \sim r_2$ 的贡献总是存在且对应着不需要下取整的情况。直接使用之前的哈希结果统计这一部分取值。
$l_2 \sim r_2 - x - 1$ 的部分对应 f 数组，也就是说我们此时希望找到一个 l_2 满足 f_{r_2+x, l_2} 等于某个固定的值。直接在 f 数组上二分是 $\log \log V$ 的。
知道 x 之后，$x - 1$ 位置上就会存在一个从小于 (l_1, r_1, t_1) 到大于 (l_1, r_1, t_1) 的变化点。利用 f 数组找到它。

再精细一点?

然后我们把这个 $\log\log V$ 优化掉。
假设取到最长的 LCP 的最小位置为 l_2。当二分的 LCP 位置 $r_2+x\geq l_2+\log V$ 时，最后判定 x 合法所取到的 f 上的取值 f_{x,l_2} 满足 $x-l_2>\log V$。

再精细一点?

然后我们把这个 $\log\log V$ 优化掉。
假设取到最长的 LCP 的最小位置为 l_2。当二分的 LCP 位置 $r_2 + x \geq l_2 + \log V$ 时，最后判定 x 合法所取到的 f 上的取值 f_{x,l_2} 满足 $x - l_2 > \log V$。
在我们之间处理 f 的时候提到，$x - l_2 > \log V$ 时只有两种取值。
在这种情况下我们不需要二分。
于是我们先只判断两种取值，二分出一个 x'，此时最长的 LCP 一定落在 $[x' - \log V, x']$ 内，再在里面二分套二分找到真实的 LCP。

再精细一点?

然后我们把这个 $\log\log V$ 优化掉。
假设取到最长的 LCP 的最小位置为 l_2。当二分的 LCP 位置 $r_2 + x \geq l_2 + \log V$ 时，最后判定 x 合法所取到的 f 上的取值 f_{x,l_2} 满足 $x - l_2 > \log V$。
在我们之间处理 f 的时候提到，$x - l_2 > \log V$ 时只有两种取值。在这种情况下我们不需要二分。
于是我们先只判断两种取值，二分出一个 x'，此时最长的 LCP 一定落在 $[x' - \log V, x']$ 内，再在里面二分套二分找到真实的 LCP。
复杂度 $\log n + (\log\log V)^2$。在数据范围下 $(\log\log V)^2$ 约等于 $\log V$。

再再再精细一点

另外，实际上我们可以利用单调性把后面的二分套二分变成维护指针做到 $\log V$。
二分出 x' 之后，不断减小 x' 并判断当前位置是否可以作为 LCP。对于每个 x'，找到最大的 pos 满足 $f_{r+x',pos}$ 小于等于两个哈希值的差。注意这里差一定是在 $[0, V]$ 范围内的，否则要么 l_2 的答案是平凡值，否则 $(r+x', r, pos)$ 已经大于 (l_1, r_1, t_1) 了。
当 $f_{r+x',pos}$ 小于哈希值差的时候，当前位置不是 LCP，答案一定就是 pos；否则当前位置可以是 LCP。减小 x' 的同时，注意到下一次的新的 pos' 一定会大于等于 pos。因此直接维护 pos 就行。

再再再精细一点

另外，实际上我们可以利用单调性把后面的二分套二分变成维护指针做到 $\log V$。
二分出 x' 之后，不断减小 x' 并判断当前位置是否可以作为 LCP。对于每个 x'，找到最大的 pos 满足 $f_{r+x',pos}$ 小于等于两个哈希值的差。注意这里差一定是在 $[0, V]$ 范围内的，否则要么 l_2 的答案是平凡值，否则 $(r+x', r, pos)$ 已经大于 (l_1, r_1, t_1) 了。
当 $f_{r+x',pos}$ 小于哈希值差的时候，当前位置不是 LCP，答案一定就是 pos；否则当前位置可以是 LCP。减小 x' 的同时，注意到下一次的新的 pos' 一定会大于等于 pos。因此直接维护 pos 就行。
最终我们将二分的复杂度做到了 $O(\log n + \log V)$，总复杂度 $O(n \log n \log V(\log n + \log V))$。

更精细的摇 mid

再回忆一下我们怎么摇 mid 的：给定 L,R，需要枚举 r 和 $r \le t \le r+\log V$，然后找到 l 满足 $L<(l,r,t)<R$。
设 $p_{r,t}$ 表示小于 R 的最大的 l，$q_{r,t}$ 表示小于等于 L 的最大的 l。
只需要计算出 p 和 q 即可。以下考虑 p，q 类似。

更精细的摇 mid

再回忆一下我们怎么摇 mid 的：给定 L, R，需要枚举 r 和 $r \le t \le r + \log V$，然后找到 l 满足 $L < (l, r, t) < R$。
设 $p_{r,t}$ 表示小于 R 的最大的 l，$q_{r,t}$ 表示小于等于 L 的最大的 l。
只需要计算出 p 和 q 即可。以下考虑 p，q 类似。
首先注意到 $p_{r,t}$ 随着 t 的增大是单调不增的，所以考虑对 (l, t) 双指针。当 $t - l \le \log V$ 的时候，比较的两个数的其中一个数的最低位是 $\log V$ 级别的。我们在下一页里说明这里怎么做到 $O(1)$。

更精细的摇 mid

再回忆一下我们怎么摇 mid 的：给定 L, R，需要枚举 r 和 $r \le t \le r + \log V$，然后找到 l 满足 $L < (l, r, t) < R$。
设 $p_{r,t}$ 表示小于 R 的最大的 l，$q_{r,t}$ 表示小于等于 L 的最大的 l。只需要计算出 p 和 q 即可。以下考虑 p，q 类似。
首先注意到 $p_{r,t}$ 随着 t 的增大是单调不增的，所以考虑对 (l, t) 双指针。当 $t - l \le \log V$ 的时候，比较的两个数的其中一个数的最低位是 $\log V$ 级别的。我们在下一页里说明这里怎么做到 $O(1)$。
当 $t - l > \log V$ 时，$1 \sim l - 1$ 移到 t 至多贡献 1，因此 t 增大 1 之后 $p_{r,t}$ 全都会变成 1。只需要在当前位置二分一次即可。

比较一个长数和一个 $\log V$ 长度的数

现在的任务是给定两个三元组 $(l_1, r_1, t_1), (l_2, r_2, t_2)$，其中 $t_2 - l_2 \leq \log V$，$O(1)$ 判断大小。

比较一个长数和一个 $\log V$ 长度的数

现在的任务是给定两个三元组 $(l_1, r_1, t_1), (l_2, r_2, t_2)$，其中 $t_2 - l_2 \le \log V$，$O(1)$ 判断大小。
我们先计算它们乘 $2^{\log V}$ 下取整之后的**精确结果**。因为两个三元组都不会超过 $2V$，所以精确结果是可以用 long long 存下来的。
具体地，我们需要预处理 $h_{l,r}$ 表示 $\sum_{k=l}^{r} a_k 2^{k-l}$，用 h 替代前面的高位哈希，低位贡献继续使用 f。

比较一个长数和一个 $\log V$ 长度的数

现在的任务是给定两个三元组 $(l_1, r_1, t_1), (l_2, r_2, t_2)$，其中 $t_2 - l_2 \le \log V$，$O(1)$ 判断大小。
我们先计算它们乘 $2^{\log V}$ 下取整之后的**精确结果**。因为两个三元组都不会超过 $2V$，所以精确结果是可以用 long long 存下来的。
具体地，我们需要预处理 $h_{l,r}$ 表示 $\sum_{k=l}^{r} a_k 2^{k-l}$，用 h 替代前面的高位哈希，低位贡献继续使用 f。
当它们不等的时候，可以直接判断大小；当它们相等的时候，还需要判断 (l_1, r_1, t_1) 的低位是否有值。一个简单的方法是将它们乘 2^n 下取整的哈希值拿出来判断是否相等。

更精细的 DP

先回顾一下我们的 DP 是什么：设 $f_{p,q}(q \le p + \log V)$ 表示最靠前的三元组为 (p, r, q) 时最小的 $\sum t_i - l_i$ 的和。按照 p 从大到小转移：

- 首先对 $f_{p,q}$ 做后缀 min，此时 q 的状态的意义变为 $r_{i-1} + 1$。
- 之后对每个 q 枚举 $l_{i-1} < p$，若 $(l_{i-1}, p-1, q-1) \le mid$，有转移 $f_{l_{i-1},\min(l_{i-1}+\log V, q-1)} \leftarrow f_{p,q} + (q - 1 - l_{i-1})$。

更精细的 DP

先回顾一下我们的 DP 是什么：设 $f_{p,q}(q \le p + \log V)$ 表示最靠前的三元组为 (p, r, q) 时最小的 $\sum t_i - l_i$ 的和。按照 p 从大到小转移：

- 首先对 $f_{p,q}$ 做后缀 min，此时 q 的状态的意义变为 $r_{i-1} + 1$。
- 之后对每个 q 枚举 $l_{i-1} < p$，若 $(l_{i-1}, p-1, q-1) \le mid$，有转移 $f_{l_{i-1}, \min(l_{i-1} + \log V, q-1)} \leftarrow f_{p,q} + (q - 1 - l_{i-1})$。

首先注意到对于相同的 q，p 越小 $f_{p,q}$ 越大，这是因为在转移式子里 $f_{p,q}$ 越小越难被转移到且转移时 $q - p$ 的增量也越大。因此转移 $f_{p,q}$ 时可以直接拿 $f_{p+1,q}$ 的转移点过来继续转移。当 $l_{i-1} + \log V \ge q - 1$ 时判断能否转移需要做的是大数和 $\log V$ 长度的数的比较，使用之前一页的方法做到 $O(1)$。

更精细的 DP

当 $l_{i-1}+\log V<q-1$ 的时候是一个区间覆盖。注意到跟之前摇 mid 相同的事实：设 $q^\star$ 为最小的满足 $l_{i-1}+\log V<q-1$ 时进行了转移的 q，那么 $q^\star+1$ 就可以转移到任意左的点了。因为做过后缀 $\min$，所以 $q^\star+1$ 之后的转移点对 $\log V$ 以外的位置的影响就不需要考虑了。

更精细的 DP

当 $l_{i-1}+\log V<q-1$ 的时候是一个区间覆盖。注意到跟之前摇 mid 相同的事实：设 $q^\star$ 为最小的满足 $l_{i-1}+\log V<q-1$ 时进行了转移的 q，那么 $q^\star+1$ 就可以转移到任意左的点了。因为做过后缀 $\min$，所以 $q^\star+1$ 之后的转移点对 $\log V$ 以外的位置的影响就不需要考虑了。
因此我们只需要做一次二分找到 $q^\star$ 的转移点，对于 $q^\star$ 和 $q^\star+1$ 对应两次对 $f_{i,i+\log V}$ 数组的区间 min，使用你喜欢的数据结构（如优先队列、线段树）单独维护这个数组即可。单次复杂度 $O(n(\log n+\log V))$。

卡常

以上的玩意跑得巨慢无比。可以考虑以下卡常：

- 比较两个大数的时候，一个数总是 mid。可以预处理出 mid 有关的哈希信息。
- 每次摇 mid 的时候需要计算 L 和 R 的边界（即前文中的 p 和 q）。在检验完 mid 的时候，L 和 R 里只有一个会被更新为 mid，所以 p 和 q 里只需要更新一个。
- 把二分改成倍增也可能更快，因为可能其中很多部分的二分都不是很长。

祝大家新年快乐