A、反转矩阵（reverse）

假设第 a 列和第 b 列中 1 的个数之和超过 2，则无论怎么反转都无法满足条件。如果 1 的个数之和为 0 或者 1，则怎么操作都是满足条件的。

如果 1 的个数之和为 2，且分别位于第 x 行和第 y 行，则分两种情况：若两个 1 都位于第 a 列或者都位于第 b ，则第 x 行和第 y 行的反转情况是相反的；否则这两行的反转情况是相同的。

统计这种方案数也是一个经典的问题。转化为图上问题，点 x 表示第 x 行不反转，点 $x+n$ 表示第 x 行反转，那么如果 x,y 的反转情况相同，则连边 $(x,y),(x+n,y+n)$，反转情况不相同则连边 $(x,y+n),(x+n),y$ 。最后如果 u 和 v 在同一连通块，则无解；否则若连通块数量为 $2k$，答案就是 2^k 。

时间复杂度 $O(n+m)$。

B、彩虹子数组（subarray）

首先将所有 a_i 都减去 i，问题就变成最长能把多长的子数组变成相同数。这是一个经典的问题，令所有数都变成子数组的中位数所需的操作次数是最少的。同时固定了子数组的左端点后，子数组的右端点具有单调性，所以可以枚举左端点后二分右端点，加上主席树求区间中位数以及求出区间所需的操作次数，复杂度为 $O(n\, log^2\, n)$。把二分右端点换成双指针就可以做到 $O(n\, log\, n)$，还可以把主席树换成普通的线段树。

但是这样做还是比较麻烦。仍然考虑使用双指针，同时使用一个大根堆和一个小根堆来维护中位数。设当前区间长度为 k，则大根堆维护区间前 $\lfloor \frac{k}{2} \rfloor$ 小的数，小根堆维护区间前 $\lceil \frac{k}{2} \rceil$ 大的数，同时维护每个堆中所有元素的和，这样就能很方便地求出中位数以及当前区间所需的操作次数。复杂度 $O(n\, log\, n)$。

C、树划分（partition）

若 $k \le \sqrt{n}$，则可以使用简单的树上背包解决这个问题。

否则，假设子树内切出去了 i 块大小为 k 的部分，j 块大小为 $k+1$ 的部分，那么包含子树根的剩余部分大小就可以求得为 $siz_u - ik - j(k+1) = siz_u(i+j)k - j$。由于剩余部分的大小肯定不超过 $k+1$ ，所以 $i+j$ 的值只可能是 $\frac{siz_u}{k}$ 或者 $\frac{siz_u}{k} - 1$。而 j 肯定是不超过 $\sqrt{n}$ 的，所以如果把剩余部分的大小计入状态，则状态数是 $O(\sqrt{n})$ 的。
可以发现当 $k \le \sqrt{n}$ 时，剩余部分的大小也不超过 $\sqrt{n}$，所以两种情况都可以用这种 dp 方法，即第二维是剩余部分大小的书上背包。如果使用 map 来存状态，可以做到 $O(n\sqrt{n}logn)$ 的复杂度。

D、循环序列（cyc）

一个序列开头为 x 等价于 $t \equiv b\ (mod\ k)$ 形式的方程，一段开头都是 x 可以用裴蜀定理判断是否有解（只考虑两个方程 $t \equiv b1\ (mod\ k1), t \equiv b2\ (mod\ k2)$，等价于 $k_1x_1 - k_2x_2 = b_2 - b_1$ 是否有解），由单调性，双指针对每个 l 求出最远的 r 使得 $[l,r]$ 合法，但需要处理这里的判断不能删除的问题，一个暴力的每次扩展之后遍历 $[l,r]$ 判断，但复杂度不对，考虑到 k 很小，实际上对于每个 k 只能有至多一个不同的 b，据此，每次扩展都只需要进行 $O(k)$ 次合并方程，则总复杂度 $O(k \times \sum k_i)$，根据去重实现可能多一个 log 因子。