20250930 题解

序列(array)

枚举 h，统计贡献，此时数列每一项都必须 $\le h$，每一项的 $h-a_i+1$ 是个连续段，简单等差数列求和，然后把每一项的和乘起来

但是这里会误算一些贡献，误算的是所有项都没取到 h 的，也就是所有项都 $\le h-1$ 的，这样的情况每一项取法也是个连续段，所以也是等差数列求和后乘起来（注意计算这个结果时序列最大值还是按 h 计算，这样一减就是把误算的都删了）

也可以设 $dp[i][0/1]$ 表示考虑了前 i 项，是否有至少一项取到了 h 的结果，$\mathcal{O}(1)$ 转移

最终时间复杂度均为 $\mathcal{O}(nv)$，其中 v 表示值域

账本(book)

算法 1

注意到操作 1 和操作 2 的顺序不影响答案，于是可以 $\mathcal{O}(2^n)$ 枚举对哪些位置进行操作 1，再 $\mathcal{O}(n)$ 枚举进行多少次操作 2，最后 $\mathcal{O}(n)$ 检查该方案是否合法并更新答案。

时间复杂度为 $\mathcal{O}(2^n n^2)$

算法 2

先枚举操作 2 的次数 ，下面考虑贪心地进行操作 1。

记原来的账本为 $a_0,a_1,\ldots,a_{n-1}$，其中 $+$ 对应 $a_i=1$， $-$ 对应 $a_i=-1$。

那么进行了 k 次操作 2 后得到的账本是 $a_{(0+k)\mod n},a_{(1+k)\mod n},\ldots,a_{(n-1+k)\mod n}$

求出账本的前缀和 $s_i=\sum_{j=0}^{i}a_{(j+k)\mod n}$

为保证修改后任意一个前缀和非负，只需保证当前的最小前缀和在修改后非负，如果当前的最小前缀和是负数，则先将该前缀中最前的一定数量的 $-$ 改为 $+$，计算出这种操作的次数，并记录对 s_{n-1} 的影响。

再调整 $\frac{abs(q-(p+s_{n-1}))}{2}$ 次，使账本满足结束时是 q 元的要求。

时间复杂度 $\mathcal{O}(n^2)$

算法 3

注意到操作的代价可以 $\mathcal{O}(1)$ 计算，时间瓶颈在于找到前缀和最小的位置。

为方便可以将 $a_0,\ldots,a_{n-1}$复制一遍接在后面得到一条长为 $2n$ 的链 $a_0,\ldots,a_{n-1},a_n,\ldots,a_{2n-1}$。

对扩展后的数组求前缀和 t_i，则进行 k 次操作 2 时前缀和最小的位置即为 t 在区间 $[k,k+n-1]$ 上的最小值的位置。

因为 $[k,k+n-1]$ 的左右端点都是随 k 单调递增的，所以可以用单调队列求出最小值的位置。

也可以看成每次取原数组 a 的一个后缀拼上一个前缀，设 a 的前缀和数组为 t，那么我们想要的最小值就是 t 的后缀最小值或者 t_{n-1} 加 t 的前缀最小值里

时间复杂度 $\mathcal{O}(n)$

删数(delete)

算法 1

有能删的数的时候一定删除最后边的，这样不会让前边本来能删的变成不能删，模拟时间复杂度不超过 $\mathcal{O}(qn^2)$

算法 2

对于 $l=1$ 的问题，我们枚举 r 从 $1\sim n$，如果新添加的 $a_r\le i$ 且 $a_r\ge i-ans$，那么在操作的过程中一定有某一时刻能删除 a_r 且不影响其它任何数，所以 ans 就加 1

这个做法同时能以 $\mathcal{O}(qn)$ 的时间复杂度过掉 $n,q\le 3000$ 的测试点

算法 3

枚举 r 从 $1\sim n$，可以发现 l 越小，能删除的一定越多，用一个数据结构维护 l 取每一个地方的时候的答案，可以发现当 r 从 $i-1$ 变到 i 的时候是该数据结构上一个前缀加 1

每次我们就线段树二分找到 $ans_l\ge i-a_i$ 的最后位置 pos，然后对 $[1,pos]$ 区间加 1

时间复杂度 $\mathcal{O}(n\log n)$

选数(select)

一个数 x 可以写成 $2^i\times 3^j\times k$ 的形式

对于每个 k（k 中不含质因子 $2,3$），构造一个二维数组，把 k 写在左上角，同行从左往右乘 3 递推，同列从上往下乘 2 递推，超过 n 的就不能选

问题转换为求选择不相邻的若干数的方案数，因为行列数都是 $\log$ 级别，所以可以状压 dp

用轮廓线 dp 的方法进行状压 dp 复杂度会较低，设 $dp[i][j][s]$ 表示依次考虑每个格，考虑完 $[i][j]$ 这个格以后第 i 行一个截止到 j 的前缀拼上第 $i-1$ 行一个从 $j+1$ 开始的后缀的状态为 s 的答案

枚举下一格选还是不选进行转移

应该可以直接过起码 50 分

考虑优化，可以发现把 k 写在左上角，把 n 作为大小限制，其实跟把 1 写在左上角，把 n/k 作为大小限制是一样的

因此可以使用数论分块优化枚举 k 的过程

进一步，如果这次的大小限制算出的有效行数和每一行的有效列数都和上一次一样的话，可以不重新推，直接用上一次的结果即可

另外 dp 的状态数也可以优化，可以发现有效状态中最多有 1 处相邻的 1（出现在第 i 行和第 $i-1$ 行拼接的那个地方），只保留这些状态进行 dp

把这些优化都做了就可以通过 $n\le 10^{10}$ 的数据了