回忆

考虑一次操作，就是选中 $n-1$ 类回忆使其减少 1 而选中一类回忆使其增加 y。

那么一次操作之后，两类的差值要么不变，要么变化 $y+1$。而我们希望最后除了 k 类其余数量均相等，否则显然无解。

那么，当非 k 类回忆的数量不同余于 $y+1$ 时无解。

考虑最小操作次数，首先，除了 k 类外，最大的一类必然是固定的，不会因操作而发生转变，否则必然不优，这是显然的。同时，任何时候不可能以最大的一类为 i，这显然是不优的。因此得出结论，排除 k 类之后最大的一类会在每次操作之后 -1。

也就是说，操作次数实际就是排除 k 类之后最大的一类的回忆个数。

但这样不会有无解的情况吗？比如两个非 k 类点同时为 0 导致无法操作？不会，因为 x 的下界为 $n-1$，因此这种情况不会出现，读者自证不难。

然后考虑最后回忆段数，这是简单的，每次操作都会新增 $y-n+1$ 段回忆，直接算即可。

mex 序列

算法1

枚举所有的子序列检查即可，复杂度 $O(2^n \times n)$。

算法2

记录 $f_{i,j}$ 表示考虑了前 i 项，mex 为 j 的符合条件的子序列的方案数，简单转移即可，复杂度 $O(n^2)$。

算法3

当 $a_i > 0$ 时，mex 的值恒为 0。所以只能包含 1，直接计算即可。

算法4

同算法2，复杂度 $O(10 \times n)$。

算法5

记录当前子序列的最大值 x。发现满足条件的 mex 只能是 $x-1$ 或者 $x+1$。直接定义 $f_{i,x,0/1}$。 表示考虑了前 i 项，当前子序列的最大值为 x，当前子序列的 mex 为 $x-1$ 或 $x+1$ 的方案数。发现每次只改变 $O(1)$ 项的值，复杂度 $O(n)$。注意部分细节。

序列

算法1

求出每个区间的最大子段和然后做一个二维前缀和。期望得分 15。

算法2

对于 $a_i \geq 0$ 的部分，也就是求所有子区间的和的和，对于 i 的贡献为 $(r-i+1)\times(i-l+1)\times a_i$。 维护 $a_i, a_i\times i, a_i\times i^2$ 的区间和即可，期望得分 30。

算法3

对于 $q=1$，可以考虑分治。每次计算跨过 mid 和 $\mathrm{mid}+1$ 的贡献，对于一个区间 $[l,r]$ 最大子段和为 $[l,mid]$ 最大子段和， $[mid+1,r]$ 最大子段和, $[l,mid]$ 的最大后缀和加上 $[mid+1,r]$ 的最大前缀和。这三者的最大值。二维数点即可。但是因为已经有单调性，就可以直接指针扫，复杂度为 $O(n\log n)$。期望得分 50。

算法4

对于 $l=1$，相当于求出每个 r，求出 $[1,r]$ 的所有后缀的最大子段和的和。我们对于每个 l，求出 $[l,n]$ 的所有前缀的最大子段和的和，相当于翻转一下。

我们从大到小枚举 l，我们定义 f_i 表示区间 $[l,i]$ 的最大后缀和，转移为 $f_i=max(0,f_{i-1})+a_i$。区间 $[l,i]$ 的最大子段和就等于 f 区间 $[l,i]$ 的最大值。

当 l 向左移动一位时，$f_l=a_l$。需要更新 f 数组，相当于找到第一个 $f_i\leq 0$ 的位置，前面的进行区间加操作。如果 f_i 之后还 ≤ 0 就停止更新，否则继续找到下一个 $f_i\leq 0$ 的位置。

我们发现每找一个 $f_i\leq 0$ 的位置后，这个位置的 f 就一直会 >0 了。所以我们总共只会进行 $O(n)$ 次区间加的操作。就可以实时维护 f 数组。

而最大子段和是 f 的前缀最大值，维护一个单调栈。我们发现如果一个 f_i 被弹出了，之后也会被弹出。所以我们用一个 set 维护单调栈的下标，每次尝试弹出即可。同时更新每个位置的前缀最大值。期望得分 75。

算法5

在算法4的基础上套用一个支持区间加，区间历史版本和的数据结构即可。 期望得分 100。

词典

算法1

设计一个 $f_{i,j}$ 和 $g_{i,j}$ 表示大小为 i 的词典，当前 $0/1$ 的个数为 j 的最小代价。转移枚举接下来一位选择 k 个 0，$i-k$ 个 1。复杂度 $O(n^2\times\max(a,b))$，期望得分 20。

算法2

注意到代价函数是凸的，而 f 和 g 的转移是类似于合并两个凸包，所以 f,g 也是凸的。直接在原来的基础上比较一下斜率即可，复杂度 $O(n\times\max(a,b))$，期望得分 40。

算法3

这个限制相当于不能存在两个连续的 0，设 $g_i=g_{i-1}+\lfloor 1+log_2 i\rfloor$，$f_i$ 表示答案。
$f_i=\max\limits_{k=1}^{i-1}(f_k+g_k+g_k+f_{i-k}+g_{i-k})$。

我们发现 f_i-f_{i-1} 的值很小。我们考虑记录 F_i 表示斜率为 i 的有多少个。同理 G，那么 $G_i=2^{i-1}$。

我们考虑计算 F_i，就是相当于两个凸包的 F_i 之和。我们就枚举一个 j， 考虑从 g 斜率为 j 加上 f 斜率为 $i-j-j$ 的部分。相当于两个区间的交。当然由于最多只有一个 j 合法，也可以二分查找。发现在斜率为 1790 时就到 10^{15} 了。可以轻松通过，期望得分 70。

算法4

就是算法3的拓展，只需要暴力求出斜率 ≤ 8 的值后面就可以暴力计算了，这部分状态数为 3000 多。期望得分 100
。