A

题目的限制条件现相当于对于每个 $i \in [1,k]$，$a_i, a_{i+k}, a_{i+2k}, \ldots$ 是一个等差数列，每个 i 对应的等差数列的公差可能不一样。

考虑对于一个 k，存在 $2k+1 \le l < r$，且 $[1,r]$ 对应的数列在当前 k 下能够满足题目的限制条件，那么显然由于 $[1,l]$ 的数列更短且被 $[1,r]$ 所包含，所以 $[1,l]$ 对应的数列在当前 k 下也能够满足题目的限制条件。

换言之，每个 k 对应的合法区间是从 $2k+1$ 开始的连续一段，如果找到了这个区间，那么可以通过差分和前缀和轻松维护最终的答案。

对于每个 k 而言，满足条件的一段前缀一定满足 $[a_{k+1}, \ldots, a_{2k}] - [a_1, \ldots, a_k] = [a_{2k+1}, \ldots, a_{3k}] - [a_{k+1}, \ldots, a_{2k}] = \ldots = [d_1, \ldots d_k]$，其中这里的减号代表每一个位置的数分别相减。

如果可以快速求解两段长度相同的序列相减后得到的新序列并且可以快速比较新序列是否相同就可以对每个 k 找到最长的满足条件的序列。

考虑采用类似哈希的方式，记一个长度为 k 的序列 $b_1, b_2, \ldots, b_k$ 的哈希值为 $b_1 \times M^0 + b_2 \times M^1 + \ldots + b_k \times M^{k-1}$，那么将另一个长度也为 k 的序列 $\{c\}$ 的哈希值与其相减，得到的结果为 $M^0(b_1 - c_1) + M^1(b_2 - c_2) + \ldots + M^{k-1}(b_k - c_k)$

对于 $[a_{k+1}, \ldots, a_{2k}] - [a_1, \ldots, a_k], [a_{2k+1}, \ldots, a_{3k}] - [a_{k+1}, \ldots, a_{2k}]$，如果它们的哈希值相减是相同的，那么就代表对应位置可以成为一个等差数列，也就意味着 $[1,3k]$ 这个前缀是满足题意的。

类似的可以判断 $[1,4k], [1,5k], \ldots, [1,n]$ 的这些前缀是否满足题意，直到判断到某一个长度为 k 的整数倍的前缀不满足题意，就在其中二分找最后的一个满足题意的位置。

时间复杂度 $O(n + n/2 + n/3 + \ldots) = O(n \log n)$。

B

考虑一个序列 $\{a\}$ 中没有出现 x，那么它的 mex 至多为 x。

所以对于一个数 x，只需要枚举不包含它的极长连续子序列，并假定这个序列的 mex 是 x。统计其中出现的数字的种类数，再减去 x 的值就是当前子序列的答案。

上述情况必定能得出最优解，因为对于不包含 x 的极长连续子序列，如果其 mex 不为 x，那么在枚举到其 mex 时就可以获得当前区间对应的最优解。

mex 的枚举范围只需要从 1 枚举到第一个没出现过的数字即可，其它数字必然不可能成为 mex。

考虑 m 个相同的数字会将序列划分为至多 $m+1$ 段极长连续子序列，最多只需要统计 $2n$ 个区间中不同数字的数目，可以采用莫队，但是有超时的风险。

因为统计的是不同的数字，可以认为某个区间中所有相同的数字之中的最后一个才对答案有贡献。

于是将所有需要统计的区间按照右端点进行排序，每次移动右端点加入新数字时只需要把它当成目前的最后一个，将它的权值改为 1，并将本来权值是 1 的倒数第二个相同数字的权值改为 0，这样 $[l,r]$ 间的权值和就是 $[l,r]$ 区间的不同数字个数。

转化为单点加+区间求和问题，利用树状数组即可在 $O(n \log n)$ 的时间复杂度内求解。

C

$[l,r]$ 间的连续子序列中 $a_{i+1}-a_i=1$ 等价于对于所有 $i,j\in[l,r]$ 有 $a_i-i=a_j-j$，于是将每个数减去它的下标，问题转化为在至多 k 次加减操作后求最长的相等子序列。

- 将一个连续区间的所有数变为相等的最优解是全部变成中位数

如果一个区间 $[l,r]$ 可以在至多 k 次操作内变为全部相等，那么对于区间 $[l_1,r_1]\subseteq[l,r]$ 肯定也可以在 k 次操作内变为全部相等，于是可以采用双指针来对于每个右端点 r 找到最远的 l 满足 k 次操作之内可以将区间内的所有数变成相等的。

首先需要动态维护滑动窗口的中位数，可以采用两个 set，第一个 set S_1 放区间内最小的一半数，第二个 set S_2 放区间内最大的一半数，如果在加入或者删除一个数后两个 set 的大小之差大于 1，那么就从大小较大的那个 set 中把对应的数移动到另一个 set 中，最后如果 $|S_1|\geq|S_2|$，那么 S_1 中最大的那个数就是中位数。

假设中位数是 m，最终每个区间变成相同需要的操作数就是 $m|S_1|-sum_1+sum_2-m|S_2|$，其中 sum_1,sum_2 代表 S_1,S_2 中的元素和，在修改其中元素时动态维护即可。

D

对于每个格子 (i,j)，如果知道最早覆盖它的矩形是第 $min[i][j]$ 个，最晚覆盖它的矩形是第 $max[i][j]$ 个，那么这个格子在某个询问中没有被覆盖的条件就是 $s_i\leq min[i][j]\leq max[i][j]\leq t_i$。

考虑利用扫描线逐行确定每个格子的 min,max。将矩形分解为 (i,ya_i,xa_i,xb_i,Add) 与 (i,yb_i+1,xa_i,xb_i,Del)，再建立一棵线段树，线段树的叶子对应这一行的每个格子，对线段树的每个区间 $[l,r]$ 用两个优先队列维护覆盖当前行标号为 $[l,r]$ 的格子的矩形的最早和最晚编号。

对于Add操作，只需要在下放的过程中在对应线段树区间的优先队列中把矩形的编号加入即可。

对于Del操作，可以采用懒删除的方式，在每个区间中再用两个优先队列用相同方式维护删除的标号，只要维护 max,min 的优先队列队首不是需要删除的标号就可以不用去执行删除操作。

处理完每一行的操作后对线段树 dfs 到每个叶子节点，同时维护涉及到的区间的所有优先队列队首的值取 max 和 min 的值，到叶子节点的取值就是当前格子的 max,min。

假设行数和列数为 C，预处理时间复杂度为 $O(n\log n\log C+C^2)$

统计出考虑所有矩形的条件下被覆盖的格子个数，对于每个询问，原本被覆盖而在询问的条件下没有被覆盖的格子满足 $s_i\leq min[i][j]\leq max[i][j]\leq t_i$。

可以将所有格子对 $max[i][j]$ ，所有询问按照 t_i 排序，并统计每个询问中相对于不删除矩形的情况下减少的格子数量。

这些格子满足 $max[i][j]\leq t_i$，在排序后可以用双指针的方式加入满足这个条件的格子，同时将 $min[i][j]$ 位置的权值+1，那么 $[s_i,t_i]$ 之间的权值和就是 $s_i\leq min[i][j]\leq max[i][j]\leq t_i$ 的格子个数，用总数减去这个数量就是这个询问的答案。

单点加+区间求和，使用树状数组即可在 $O(q\log n+C^2\log C)$ 的复杂度内完成求解。

E

假设 $a_j=\max\{a_i\}$，对于 f_i 而言，在 $i\geq j$ 后 $f_i=a_j$ 并保持不变，对于 g_i 而言，在 $i\leq j$ 后 $f_i=a_j$ 并保持不变。

所以对于一个位置 i，必然有 $\max\{f_i,g_i\}=a_j$，$\min\{f_i,g_i\}=f_i+g_i-a_j$

所以 $\sum_{i=1}^n\min\{f_i,g_i\}-a_i=\sum_{i=1}^n f_i+\sum_{i=1}^n g_i-n\max\{a_i\}-\sum_{i=1}^n a_i$

$\max\{a_i\}$ 与 $\sum a_i$ 都好维护，问题来到了如何维护 f_i,g_i。

对于一个固定的序列 a，f_i 是逐段单调递增的，可以维护序列 a 的一个单调栈，栈内的每个元素控制的区间长度乘以这个元素的值就是 $\sum f_i$。

考虑修改操作，可以用 set 去维护这个单调栈，set 里面维护一个 pair 代表单调栈里面每个元素在原序列里面的位置和这个元素的值。修改操作就可以找到这个元素所在的区间，看这个元素加入 set 以后是否会改变单调栈的内容，如果会则将 set 的当前区间分裂为两个区间，并且看单调栈后面的元素是否比这个元素更小，如果是就还需要删除后面的元素。

g_i 的维护相当于将 a 序列倒过来用相同的方式维护。

每个元素在初始和修改的时候分别最多只会进入 set 一次，所以复杂度为 $O((n+q)\log(n+q))$