Solution

可怜

> powered by CJ

首先我们看见这个条件，第一步就是把r改成$l+1$，r'改成$l'+1$。即任意两个长度为2的子段均不等。

然后考虑去掉$[L,R]$的限制之后你会如何构造?

$$0,0,0,1,0,2,0,3,0,4$$

左端点的限制不会改变我们构造序列的方法，只考虑右端点。假设当前的限制为$[0,r]$。大的方向上，我们肯定是希望$0,0,0,1\ldots 0,r$之后跟着$1,1,1,2,\ldots,1,r$。

第i段内只有(i,j)与$(j,i)(j\geq i)$。拼接处则有$(r,i)(i\geq 1)$。

注意最后还可以出现一个$(r,0)$以及最后可以有r,r,r。即可，判断无解的充分条件是:

$$n>(r-l+2)(r-l+1)+1$$

二进制(gen)

传统根号分治。注意到求 $f(x,y)$ 式子比较冗长，比较麻烦，而答案要求的式子和图其实关系不大。考虑为什么要把这个问题放在无向图上。

不难发现原因：$\sum_{i=1}^{n} deg_i = 2m$。有了这么一个条件，我们就可以知道，本质不同的 deg_i 只有 $O(\sqrt{m})$ 种。所以我们只需要枚举这 $\sqrt{2m}^2 = 2m$ 种不同的 deg 组合计算，然后乘以出现次数即可。

由于题目中的式子是有序的，因此需要注意枚举方式，或者把最终的答案除以 2。

时间复杂度为 $O(n+m)$

聚众斗殴 (fight)

由于 n 是 2 的幂，所以我们可以用一颗二叉树来表示题目的过程。

如果我们只需要对 x 插入某**一个**位置求出获胜概率，我们可以记 $f_{i,j}$ 表示子树 i 中 j 点胜出的概率，转移时枚举左右子树分别的胜出者即可，时间复杂度为 $O(n^2)$。再枚举 x 点的位置，时间复杂度就变成了 $O(n^3)$。

有关 $f_{i,j}$ 的状态与计算用时：

$$T(n) = 2T(\frac{n}{2}) + \frac{n^2}{4} = O(n^2)$$

但是注意到一件事情，枚举 x 的位置之后，我们只需要知道 x 到 根路径上 "侧挂" 的子树 f_i 的值。而一个子树始终对应原序列的一个区间，可以发现不同的区间数量是 $O(n)$ 的。于是 $O(n^2)$ 求出这些区间的 DP 值即可。

具体来说，由于我们只需要知道 x 被留下的概率，所以 x 以及 x 的祖先所在的节点上，其他的元素被留下的概率是没有意义的。因此只考虑 x 向上的转移，其复杂度为：

$$T(n) = T(\frac{n}{2}) + O(n) = O(n)$$

对于不是 x 以及 x 祖先的节点，他所代表的区间只可能有两种，取决于 x 在它所代表的区间之前还是在它所代表的区间之后。

这些区间的总数量是 $O(n)$ 的，复杂度与上面的暴力分析相同，因此最终的总时间复杂度为 $O(n^2)$。

博弈(act)

以下记一局博弈的某个状态为 (a,b),记局面集合$\{(a_1,b_2),(a_2,b_2),...,(a_n,b_n)\}$ 为 X。

对于每个询问，先判掉状态是否在 X 中，这种情况是直接输的；然后判掉 (a,b) 是否和 X 中的某一个数相同，这种情况是直接赢的。

性质 1：显然的，若 (a,b) 必胜，必然存在一个 $(i,b)(i<a)$ 或 $(a,j)(j<b)$ 必败。

因此，当其中一个数不变时，必败态（除 X 外）必然只有 ≤ 1 个。因而可以得出：

性质 2：对于 a，求出最小的 b 满足不存在一个 $i<a$ 使得 (i,b) 必败，则 (a,b) 必败。

考虑 $n=0$，相当于一个只有两堆石头的 NIM 游戏，只有 (i,i) 必败。

考虑 $n=1$：

- 若 $a_1=b_1$，显然没有影响；
- 若 $a_1<b_1$，则因为 (a_1,b_1) 必败，故由性质 1 得 (b_1,b_1) 必胜，同时又由性质 2 推导得 (b_1,b_1+1) 为必败。进一步，对于任意的 $b\ge b_1$ 都能推得 $(b,b+1)$ 必败。也就是说，这个东西相当于使得 $b'=b-[b\ge b_1]$；
- 若 $a_1>b_1$，同理可得，这个东西相当于使得 $a'=a-[a\ge a_1]$。

更进一步的，考虑一般情况。

对于 $(a_i,b_i)\in X$，先得到当前的 a_i',b_i'：

- 若 $a_i'=b_i'$，没有影响；
- 若 $a_i'<b_i'$，则使得 $b'=b-[b\ge b_i']$；
- 若 $a_i'>b_i'$，则使得 $a'=a-[a\ge a_i']$；

对于局面 $(a,b)\notin X$，同样得到 a',b' 之后，必败当且仅当 $a'=b'$。

需要注意的是，对于同一个 a_i 或 b_i 只能减一次，原因显然。

所以只要模拟上述减的过程即可，然后就是简单数据结构了。

把 X 和询问全部离线，按 a 第一关键字，b 第二关键字排序来做。由于已经排好序，a 减多少已知，只要求出 b 减多少即可。这相当于一个支持单点插入、查询排名的数据结构，离散化+树状数组 或 平衡树均可。

时间复杂度 $O((n+q)\log(n+q))$。

重生（reborn）

给定长度为 n 序列 a，m 次询问，每次询问给出 l,r,x，求 $\max_{i\in[l,r]} a_i \text{ and } x$ 或 $\max_{i\notin[l,r]} a_i \text{ and } x$。

$a_i,x \le 2^{20}, n,m \le 3\times 10^5$。

考虑按照常规 xor 的操作放到 01Trie 上，但是发现如果 x 的某一位是 0，可以走 0 或者是 1。那么我们就暴力地类似线段树合并，从下往上把 1 儿子的信息暴力合并到 0 儿子上，这样就可以有 1 走 1，没 1 走 0了。

由于合并之后的树不支持快速插入，删除，所以考虑操作分块，每 S 个操作重构一次 Trie，多出的部分直接暴力扫。

于是我们就可以分析复杂度：（块长为 S）

$\mathcal{O}(\frac{n}{S} n\log^2 n + n\log n + n\cdot S)$

第一个是重构的复杂度，第二个是插入的复杂度，第三个是暴力扫的复杂度。

于是容易得到 $S=\sqrt{n\log^2 n}$ 的时候最优，最后的复杂度是 $\mathcal{O}(n\sqrt{n\log^2 n})=\mathcal{O}(n\sqrt{n}\log n)$。

特殊的，$a_i\notin[l,r]$ 的时候可以特殊处理，即不用重构，所以我们常数其实没有那么大。