共同好友

使用 `vis` 标记，求2个集合 $s1, s2$ 的交集大小可以在 $O(max(|s1|, |s2|))$ 时间内完成，这样能通过前2个子任务和随机数据。

使用`bitset`优化，写得好可以通过子任务3。

注意到这个问题形式非常简单，应该是经典问题（大概率没有 $log(n)$）做法。

考虑根号分治，设置一个阈值 `B`，对于度数 $> B$ 的点，预处理其和所有点的答案（点数不超过 $k = 2m/B$），这部分采用暴力标记的方法，时间 $O(k(n+m))$。

询问时，如果 (u, v) 中有预处理过的点则 $O(1)$ 回答；否则暴力，暴力部分复杂度 $O(qB)$。

总复杂度 $O(k(n+m)+qB)$，B 取大第一点（~ 4000）会比较快。

操作序列

记第 i 个区间的乘法次数之和为 S_i。

对于某次加法操作 p，其最终贡献取决于它之后的所有乘法操作次数 cnt，可以将 cnt 看成由2部分组成：

1. 对于所有包含 p 的区间 $[l_i, r_i]$，之后所有区间的乘法次数之和 $\sum_i \sum_{j>i} S_j$
2. 对于所有包含 p 的区间 $[l_i, r_i]$，区间内 $[p+1, r_i]$ 中的乘法次数之和

从 $q \to 1$ 遍历所有区间，维护一个倒序差分数组 d，对于第 i 个区间：

1. 在 $d[r_i]$ 处加上 $2^{\sum_{j>i} S_j}$；
2. 在 $d[l_i - 1]$ 处减去 $2^{S_i + \sum_{j>i} S_j}$；

从 $m \to 1$ 遍历所有操作，维护一个变量 $now = 2^{cnt}$，由差分数组 d 可以求出第一部分 $2^{\sum_i \sum_{j>i} S_j}$；每次遇到乘法操作让 now 乘2就可以正确处理第2部分区间内的贡献。

AND和XOR

首先有 $O(na^2)$ 的暴力DP，滚动之后空间是 $O(a^2)$，可以得到60分。

观察到每位只有3个状态：全为 1，或者有奇数/偶数个 1。

可以预处理 $s \to$ 其三进制表示。

记DP状态 $dp[i][0/1][s] =$ 前 i 个数已经选了奇数/偶数个数，每位的状态是 s 的方案数，枚举 $a[i]$ 选或不选进行转移，复杂度$O(n3^{12})$，滚动之后空间 $O(3^{12})$。

子任务4

考察我们实际上求了什么：设 $f_{S,j,0/1}$ 表示在所有 $x\&S=S$ 的 x 中选出一个大小为奇数/偶数的子集，异或和为 j 的方案数，我们最后要求的是 $\sum\limits_S f_{S,S,1}+(-1)^{popcount(S)}(f_{S,0,0}-1)$。

先枚举 S，设 $a_1,\dots,a_k$ 表示所有 $x\&S=S$ 的 x 组成的序列，布尔数组 $c_1,\dots,c_k$ 表示所有每个 x 有没有被选，发现选奇数/偶数个和异或和的都限制可以被一个异或方程组刻画。

枚举每一位 h，设 $x_{(h)}$ 表示 x 的第 h 位，则 $f_{S,S,1}$ 的限制为：

$$\begin{cases} c_1\oplus c_2\oplus\cdots\oplus c_k=1\\ a_{1(0)}c_1\oplus a_{2(0)}c_2\oplus\cdots\oplus a_{k(0)}c_k=S_{(0)}\\ a_{1(1)}c_1\oplus a_{2(1)}c_2\oplus\cdots\oplus a_{k(1)}c_k=S_{(1)}\\ \cdots\\ \\ a_{1(15)}c_1\oplus a_{2(15)}c_2\oplus\cdots\oplus a_{k(15)}c_k=S_{(15)} \end{cases}$$

$f_{S,0,0}$ 和其本质相同。

对其做高斯消元，若无解方案数就是 0，否则就是 2^{k-cnt}，cnt 为消完后的矩阵非 0 行的个数。

直接做是 $O(2^m m^2 n)$ 的，期望获得 55 分。

子任务 5&6

发现由于矩阵只有 $0/1$，所以可以把每行存到一个 bitset 里优化高斯消元，复杂度除以一个 w，可以获得 70-80 分不等。

到这时候你会发现你把线性基给忘了，而线性基的经典应用——求 n 个数选出异或和为 x 的子集的方案数，和上面的高斯消元是等价的。

具体地，我们先把每一个 a_i 增加一个恒为 1 的位，这里使用了 $a_i\times 2+1\to a_i$，这样如果我们要求选奇数个异或和为 S，就变成了选异或和为 $2S+1$ 的子集。于是这个问题就直接可以用线性基解决。

具体地，将修改后的每个 a_i 插入线性基，设 tot 为插入失败的 a_i 个数（即 a_i 能被已经插入的数表示出来）。则 $f_{S,0,0}=2^{tot}$；若 $2S+1$ 能被表示，$f_{S,S,1}=2^{tot}$，否则为 0（简单来说就是 tot 个数可以随便取或不取，最后再用插入成功的那些数唯一地组合出 $2S+1$），具体证明可以自行上网查找。

复杂度 $O(2^m nm)$，期望获得 80 分。

子任务 7

每次枚举 S 挨个加入每个数太蠢了，考虑 S 的线性基里面是 S 的超集形成的线性基，所以我们直接可以用类似高维前缀和的方式进行 $O(2^m m)$ 次合并，每次合并是 $O(m^2)$ 的，所以加上每个值插入的复杂度，总复杂度是 $O(nm+2^m m^3)$，常数很小。

朝日

来源

NEERC2016L(CodeForces Gym 101190L)

30~50分

各种暴力。

一种 50 分的做法是: DFS所有的序列，参数中先枚举序列和、再从小到大 DFS 序列中的每一个元素。输出够了 r 个字符就退出。稍微剪枝即可。注意你很容易判断一个状态最后能不能到达某一个合法状态，不合法的直接停掉，这样复杂度还是很有理有据的与 r 是线性的数量关系。

70分

考虑令 $f[i,j,k]$ 为第 i 个素数以后构成的、序列和为 j 、长度为 k 的素数导出序列个数。

$g[i,j,k]$ 为第 i 个素数以后构成的、序列和为 j、长度为 k 的素数导出序列对应的字符串的总长度。

显然我们可以递推 f,g，这是一个基础背包。

考虑复杂度，根据计算发现，第 10^{18} 个字符大约是在和 $s=735$ 的时候取到，所以总状态数有 $\pi(s)s^2/2\approx 3.5\times 10^7$ 个，可以通过。

那么我们每次想知道一个字符，只需要像 50分一样，大力 DFS，根据 f,g 判断下面的分支的大小，进入适当的分支，就可以了。

100分

稍微优化一下，我们发现我们可以不一个个字符DFS，而是可以一次 DFS求出所有字符，就可以过了（意思就是保证每次往下DFS时必然有输出，复杂度就可以做到关于输出长度线性）。