J

可以用找规律的方式得到答案。

结论是：对 $1 \sim 2^n$（$n \in \mathbb{N}^+$），一定可以将其分为两组，使得每组的 $0 \sim n-1$ 次幂的和均相等（0 次幂和相等说明了每组都只有 2^{n-1} 个数）。

考虑使用数学归纳法证明这个结论并给出构造。

当 $n = 1$ 时显然正确。

假设 $1 \sim 2^n$ 被分为了两份：

$a_1, a_2, \cdots, a_{2^{n-1}}$，$b_1, b_2, \cdots, b_{2^{n-1}}$，且满足 $0 \sim n-1$ 次幂和均相等。

令第一个集合为：

$$a_1, a_2, \cdots, a_{2^{n-1}}, b_1 + 2^n, b_2 + 2^n, \cdots, b_{2^{n-1}} + 2^n$$

令第二个集合为：

$$b_1, b_2, \cdots, b_{2^{n-1}}, a_1 + 2^n, a_2 + 2^n, \cdots, a_{2^{n-1}} + 2^n$$

要证明这两个集合的 k 次幂相等，只需对形如 $(a + 2^n)^k$ 进行二项式展开。其中 $0 \sim k-1$ 次幂已经证明，对于 k 次幂，前面的不带 2^n 的项会与后面形成互补（由 a 与 b 的 $0 \sim n-1$ 次幂和相等），因此也相等。

有了这个结论，这题就变成了输入一个数，输出一些数了。

Q

相当于选一些边，使得这些边形成两个完全图，那么每个点要么属于连通块 1，要么属于连通块 2。

那么考虑没选的那些边，一定是连接连通块 1 和连通块 2 的两个点的。

也就是说，没被选的那些边一定形成了一张二分图。

考虑从大到小删边，并用并查集维护那些删掉的边形成的连通块。我们需要用并查集来判断两点之间的距离的奇偶性。

对每个点，记录它到并查集上的父亲结点的距离的奇偶性。

每次删掉一条边（相当于往并查集里加入一条边）的时候，判断一下这条边是否会形成奇环。如果形成了奇环，说明这条边加入以后就不是二分图了，那么说明答案就是这条边的长度。

时间复杂度为 $O(n^2 \log n)$。

K

首先，对 a 进行排序。

假设存在一个 $a = 0$。

我们要使这些数的异或和为 0，那么从最高位开始考虑。我们每次考虑进行一些操作，去掉当前异或和的最高位的 1。假设最高位为 P。

一个显然的结论是，若 $a_i \le a_j$，则进行若干次操作以后，a_i 与 a_j 之间仍然满足 $a_i \le a_j$。因为若 $a_i > a_j$，则必然在某次操作之后 $a_i = a_j$，此时对 a_j 进行操作显然是等价的。

这意味着，我们可以找一个满足第 P 位为 0 的，且后 $P-1$ 位最大的数，让它恰好加到第 P 位为 1 为止，从而消掉这位的 1。这样操作不会改变相对的大小顺序，且花费了最少的步数，所以这个贪心是优的。

由于存在 0，因此我们每次操作都可以找到这样的数。这样的时间复杂度是 $O(n \log V)$ 的（V 为值域，下同）。

那么现在不存在 0 了，就可能导致一开始找不到这样的数（显然，第一次找到以后，就会一直存在后面的位为 0 的数）。那么，我们需要人为的造一个 0 出来。对第 X 位，我们找两个数，它们的 X 位为 0，且后面 $X-1$ 位尽可能大。然后我们将它们加到 X 位均为 1。这样 X 位的异或和没有被改变，同时我们花费最小的代价得到了两个 0 。然后做上面存在 $a=0$ 时的步骤即可。

我们需要从大到小枚举这个 X，然后在这一位上造两个 0（如果有必要），然后再进行上述操作。因此时间复杂度为 $O(n \log^2 A)$。

观察到，我们取出来用来造 0 的两个数的过程，和贪心的时候取出来操作的数的过程是一样的。事实上，我们只要对每个 X，选择两个（尽可能找两个）满足第 X 位为 0，后面 $X-1$ 位尽可能大的，**且前面没被选**的数，然后对这些选上的数做贪心的步骤即可。

这样的时间复杂度就是 $O(n \log A)$。

A

以下称一个素数的正整数次幂为素数幂。

能够作为答案的数只有素数幂。

原因是，若有一个数满足 $p \times q$ 是无法表示出来的数（p, q 互质），那么 p 和 q 肯定至少有一个无法被表示出来。这样最终肯定会得到一个素数幂。

由于 n 只有 10^5，因此答案最大不超过第 10^5+1 个质数，大概 1.3×10^6。可以认为这是 $n \log n$ 级别的数。而 1.3×10^6 中的素数幂的总数是 $O(n)$ 级别的数。

这可以推出，如果 a_i 不是素数幂，或者 a_i 很大，那么这个 a_i 就没用了。

我们考虑从小到大枚举区间的右端点 r，并维护 d_i。其中 d_i 表示，左端点 l 落在 $1 \sim d_i$ 时，区间 $[l, r]$ 能表示 i。这个 d_i 显然应该是符合条件的最大的。

考虑新加入了一个 p^k。那么对每个 p^a 满足 $a \ge k$ 的，我们可以用类似 dp 的方式使用 $d[p^{a-k}]$ 来更新 $d[p^a]$。

查询的时候，我们实际上是想找到最小的 i，满足 $d_i < l$。那么可以使用线段树维护这个 d 数组，查询的时候直接在线段树上二分即可。修改的时候只需要单点修改。

由于很多数都不是素数幂，因此可以把 d_i 的 i 的定义改成第 i 小的素数幂，这样线段树的大小就只有 $O(n)$ 级别。

由于每个 r 需要进行 $\log$ 次的单点修改，因此时间复杂度为 $O(n \log^2 n + m \log n)$。