哈斯克尔

考虑使用 bitset 维护答案。

将序列中的 0 看成 -1，则我们只需要看区间的和是否大于 0 即可，对数列求前缀和后只需要看 a'_r 和 a'_{l-1} 的大小关系。

对每个 a'_i，用一个 bitset 记录 $a'_j > a'_i$ 的位置（这个可以将 a' 排序后维护），将这个 bitset 右移 i 位，就是以 i 为左端点，且 f 值为 1 的**区间长度**，要 f 值为 0 的直接翻转 bitset 即可。然后将答案数组与该 bitset 取按位与。

由于 b_i 表示以 i 为右端点的限制，我们将上面反着维护即可。

时间复杂度 $O(\frac{n^2}{w})$。

方可特尔

算法一

令 $f_{i,j}$ 表示前 i 个数选 j 个的最大价值，从 $f_{i-1,j}$ 和 $f_{i-1,j-1}$ 转移得到。

时间复杂度 $O(n^2)$。

算法二

对于 x 较小的，x^k 很快就会非常接近 0，后面的贡献可以忽略，也可以使用算法一的 dp，只需要把第二维设得小一点即可。

算法三

本题有一个结论是：对于选 k 个的最优方案，可以从选 $k-1$ 个的最优方案中，添加 1 个得到。这个结论并不难猜到，下面给出结论的证明。

以下用 p 表示一种方案选择的下标序列，满足 $1 \le p_1 < p_2 < \cdots < p_k \le n$。

首先有一个引理，对于一个选 k 个元素的最优方案 p 和任意一个区间 $[L,R]$，假设 p_i 是 $[L,R]$ 中最左边的被选定的位置，p_j 是 $[L,R]$ 中最右边的被选定的位置，那么 $p_i, p_{i+1}, \ldots, p_j$ 是从 $[L,R]$ 中选 $j-i+1$ 个的最优方案（之一）。这个引理的证明很简单：若不是最优方案，则用最优方案替换掉当前方案中的这些元素，显然能得到更优的全局解。

我们回到最开始。首先 $k=1$ 显然是成立。

假设 $k=k_0-1$ 时结论成立，考虑 $k=k_0$ 的情况。我们首先可以任意给出一个解，然后考虑将 $k=1 \sim k_0-1$ 的所有新增点按顺序插入。

当一个位置 x 被插入时，若 x 本身已经在答案中，则跳过。否则，我们找到 x 前面（或者后面）第一个**不是被插入的**位置，将这个位置删掉，并将 x 插入。根据引理，这样做一定不会使答案变差。

这样就将 $k=k_0-1$ 时选的点全部插入到当前答案中了。

由数学归纳法，结论成立。

那么我们只需要知道每次加哪个点最优就好了。使用线段树维护选择每个位置能够增加的价值即可。需要支持查询全局最大值，单点修改，区间加，区间乘。

时间复杂度 $O(n\log n+m\log n)$。

阿坡李尅铁夫

对每个结点，我们求出它上方放**最少**的结点时，上方结点最小距离和 l 和最大距离和 r，以及下方结点的最小距离和 L 和最大距离和 R（直接将 Z 直接加到 l 和 r 中，之后不用考虑 Z）。

同样地，我们求出它上方放**最多**的结点时，上方结点最小距离和 l' 和最大距离和 r'，以及下方结点的最小距离和 L' 和最大距离和 R'。

由于边权为 1，所以我们可以通过调整，使得对于最小值为 x，最大值为 y，区间 $[x,y]$ 中的距离和都能取到。 若 $[l,r]$ 与 $[L,R]$ 有交，或者 $[l',r']$ 与 $[L',R']$ 有交，则答案为 0。

否则，由于多选择一个结点，距离和会变大，对于 $r<L$ 且 $r'>L'$ 的情况，即两个区间的大小关系发生了变化，我们可以说明，在绝大部分情况下，两个区间在某一时刻有交。如果两个区间的大小关系自始至终没变化，则选择两个区间距离最近的时候的答案即可。

考虑某一时刻 $r_0<L_0$，且下一时刻 $r_1>L_1$，由于 r 代表最大值，所以此时一定能够选择一个距离为 1 的结点，也就是说 $r_0+1\sim r_1$ 都是可以得到的。同理，$L_1\sim L_0-1$ 也都是可以得到的。那么仅当 $r_0=L_0-1$ 且 $l_1=R_1+1$ 的长度都是 1 时，两个区间没有交（此时答案为 1），其他情况两个区间都有交，答案为 0。这种情况只会出现在上方选 0 个或全选的情况，特判即可。

一个结点上方的结点到它的距离为 1 开始的连续数列，它的上下界可以直接计算。

我们需要维护，一个结点子树内取 k 个数的最大值以及最小值。使用线段树合并维护结点深度，然后用线段树上二分来求前 k 小/大数的和即可。

时间复杂度 $O(n\log n)$。

魔纳德

算法一

直接枚举每个整数并判断它是否是魔数。

算法二

采用分段打表的方法。设一个值 S，对于每个 S 的倍数位置，求出它之前有多少，并魔数打表存储。没有统计到的部分使用暴力。应该能做 10^9。

算法三

通过观察样例等方式可以发现，魔数的个数并不多，考虑把它们全部求出来，这样回答询问时只需要使用二分查找。

考虑折半，也就是将一个整数拆成 $a\times10^6+b$ 的形式。可以枚举 b 的每一种情况，并用一个状态 (x,y) 表示它，其中 x 是在 b 中出现而 $2b$ 中没出现的数位集合，y 则反之。这个状态可以用两个 long long 来压缩。可以使用 `map<pair<long long,long long>,vector<int> >` 等方式存储 b 的状态。

对每个 b 的 (x,y)，它 a 的状态必须得是 (y,x)。那么我们再枚举 a 的状态，然后找到能与 a 配对的所有 b，并把它加入答案。

注意到 $2b$ 可能会产生进位，要对进位和不进位两种情况分别处理。

设范围内一共有 k 个魔数，那么上面算法的复杂度为 $O(\sqrt{r}\log r + k)$。在 10^{12} 内共有 19024860 个魔数，因此可以跑出所有的数。

至于回答询问，则对所有魔数排序以后二分查找即可。实际上，如果枚举的顺序合理，则并不需要排序。