二次剩余

nq较小的情况，可以$O(nq)$遍历每个二次函数判断是否删除

对于m较小的情况，可以对于每个不同的m用一个堆按照k递增的顺序维护每个二次函数

每次弹掉的部分一定是最小的几个，每次遍历m个位置，一共会弹掉最多$n+q$个二次函数

因此总复杂度为均摊的$O((n+q)\log n+qm)$

对于m较大的情况，依然考虑用堆维护这些二次函数

根据$f(x)=(x-m)^2+k\le t$，发现最多只有x两边$\sqrt{t}$个位置的值可能存在需要弹掉的二次函数

因此用相同的方法维护因此复杂度为$O((n+q)\log n+q\sqrt{t})$

倍数区间

一个非常浅显的思路是枚举每个数a_i，找到两边最长的合法位置L_i, R_i

容易想到可以二分L_i, R_i，但是并不好直接检查是否合法，考虑将倍数转化一下

考虑$\forall [L, R], x|a_i \Leftrightarrow x|\gcd(a_L, \cdots, a_R)$

因此倍数问题可以转化为gcd问题

用倍增维护二分，预处理复杂度$O(n\log^2 n)$，依次查询复杂度为$O(n\log n)$

用线段树维护二分，预处理和查询复杂度均为$O(n\log n)$

然而最优的做法是用栈维护L_i, R_i

依次考虑每个数a_i，用一个栈维护前面出现的，且互相之间分离的每个$[L_j, j]$

插入时依次考虑栈内每个元素a_j，若$a_i|a_j$，则显然$\forall k \in [L_j, j], a_i|a_k$，因此可以弹掉这个元素，同时扩展$L_i$为$L_j$

同理地处理R_i，复杂度为$O(n)$

Tips:输出时注意不要输出重复

C.飞翔的鸟

暴力做法就是直接dp然后两边累和，预处理和最后计数的复杂度为$O(nk)$

因为k很小，考虑用矩阵乘法优化

设无障碍的转移矩阵为A，有障碍的转移矩阵为B

则答案矩阵应为$Ans=\sum_{i=2}^{x-1}A^{i-1}\cdot B\cdot A^{n-i}$，我们需要知道$Ans[\frac{k}{2}][\frac{k}{2}]$，最后再乘上$n-2$的逆元

求出这个矩阵有两种做法

Solution1

考虑用一个矩阵维护两部分的答案

$$X=A^i,Y=\sum_{j=2}^{i}A^{j-1}BA^{i-j}$$

$$X'=X\cdot A,Y=Y\cdot A+X\cdot B$$

两部分分别转移按照递归关系转移即可，复杂度为$O((2k)^3\log n)$

(由于矩阵乘法的实现，可能实际复杂度还需要乘2，所以应当是$O(16k\log n)$，但是只要在矩阵乘法中合理优化，就能避免大部分空余复杂度)

Solution2

考虑倍增答案，每次倍增转移上面提到的两个矩阵，设其为X,Y

则扩展一倍之后，按照转移的式子应有$X'=XY+YX,Y'=Y\cdot Y$，这一步需要3次乘法

对于扩展之后仍然剩余的一个可以额外处理，这一步需要两次乘法

每次转移有$3-5$次乘法，复杂度上限为$O(5\cdot k^3\log n)$

选择不同的算法，根据常数优化能得到不同的分数

实际只需要优化取模次数，用unsigned long long 存储，每$16-18$次进行依次取模运算即可

ZYT的答案

对于n, m较小的情况，可以考虑枚举路径起点，带回撤地遍历树就能得到所有情况的答案，复杂度为$O(n^2)$

对于$x_i = y_i$的情况，等价于求权值最大的路径，可以直接$O(n)dp$求出直径

对于$x_i = 1$的情况，方案必然过1号节点，实际上直接从1号点开始累路径前缀和即可

一般的情况，考虑将每份答案的贡献描述为：当选择的路径包含x_i到达y_i的子段时，加上该权值

考虑一种比较简单的情况，$LCA(x_i, y_i) \neq x_i, y_i$，那么就是所有从$x_i$的子树到达$y_i$的子树中的路径都要加上权值

如果用dfs序描述一颗子树为一段连续的区间，那么这个问题可以转化为一个二维区间加法和查询全局最大值的问题

同理的x_i, y_i为祖先关系的情况也可以得到类似的处理

处理出所有询问后，依次扫描起始点的dfs序，用线段树维护二维加法和全局最大值，复杂度为$O(m \log n)$