B. inversions

时限：1 s　内存：512 MB　文件：inversions.cpp

问题描述

给出一个长度为 2^n 的整数序列 $a_{1..2^n}$，m 次操作，每次操作给出一整数 q_i。

操作含义：把该序列分成 2^{n-q_i} 个连续的长度均为 2^{q_i} 的段，把每段分别翻转。查询每次翻转后的逆序对的数量。

为了避免昂贵的输入输出代价，输入采用 xorShift128Plus 算法生成，包含参数 $n, m, threshold, k1, k2$。实现如下：

```
typedef unsigned long long ull;
ull k1, k2;
ull xorShift128Plus() {
    ull k3 = k1, k4 = k2;
    k1 = k4;
    k3 ^= (k3 << 23);
    k2 = k3 ^ k4 ^ (k3 >> 17) ^ (k4 >> 26);
    return k2 + k4;
}
void gen(int n, int m, int threshold, ull _k1, ull _k2) {
     k1 = _k1, k2 =_k2;
     for (int i = 1; i ≤ (1 << n); i++) a[i] = xorShift128Plus() %
threshold + 1;
     for (int i = 1; i ≤ m; i++) q[i] = xorShift128Plus() % (n + 1);
}
```

设第 i 次翻转后的答案为 ans_i，你只需要输出 $(ans_1 \times 1) \oplus (ans_2 \times 2) \oplus \cdots \oplus (ans_m \times m)$ 的值（其中 $\oplus$ 代表异或运算）。

保证标准程序不会依赖于特殊的输入输出方式实现。

输入格式

第一行五个整数 $n, m, threshold, k1, k2$。

输出格式

输出一行一个整数表示 $(ans_1 \times 1) \oplus (ans_2 \times 2) \oplus \cdots \oplus (ans_m \times m)$。

样例1 输入

```
3 3 3 9982 44353
```

样例1 输出

```
48
```

样例1 解释

初始序列 $\{a_i\}$ 为 $\{2, 3, 2, 1, 2, 1, 3, 1\}$，操作序列 $\{q_i\}$ 为 $\{1, 2, 2\}$。

第一次操作后序列变为 $\{3,2,1,2,1,2,1,3\}$，此时逆序对个数为 12。

第二次操作后序列变为 $\{2,1,2,3,3,1,2,1\}$，此时逆序对个数为 12。

第二次操作后序列变为 $\{3,2,1,2,1,2,1,3\}$，此时逆序对个数为 12。

数据范围及约定

对于 10% 的数据，$m=0$。

对于另 30% 的数据，$n \le 5$。

对于 100% 的数据，$0 \le n \le 20,\ 0 \le m \le 10^6, 2 \le threshold \le 10^9, 1 \le k1, k2 \le 10^{18}$。