任凭风浪起，稳坐钓鱼台

出题人：ilnil

题意

给出n,k,x，求：
$\sum_{i=1}^{n}\sum_{j=i+1}^{n}(a_i \oplus a_j)^x$

其中$\oplus$为异或，$x=3$。

$nk \le 2*10^6$

一个暴力

枚举两个数，然后计算它们异或值模998244353。

时间复杂度 $O(n^2\lceil\frac{k}{w}\rceil)$。

一个小优化

可以压63位从高位到低位计算只用一次模。

例如这样：

```
mul63=(unsigned long long)1<<63%998244353;
for(int l=k-1;l>=0;--l)
    v=(v*mul63+(a[i][l]^a[j][l]))%998244353;
```

$k \le 20$

异或fwt 。

时间复杂度 $O(2^k)$

另外一个暴力

考虑将这个式子$(a_i \oplus a_j)^3$拆开，就变为如果第(k_1,k_2,k_3)位异都都起来为1，答案就加上$2^{k_1+k_2+k_3}$

那么就枚举3个位，然后考虑计算有多少对的三个位异或起来为1。

枚举每一个数，将对应的三个位映射到最低三位，用一个2^3的桶记录下来，最后乘起来就能得到个数。

时间复杂度 $O(n\binom{k}{3})$

另外一个暴力（更快一点）

考虑使用$bitset$。

设$f[i][x][j]$表示第i位第j个数是否是x，那么$f[i][x]$就是一个n位的$bitset$

那么如果求的是第(k_1,k_2,k_3)位是(x_1,x_2,x_3)的个数，那么就是

$count(f[k_1][x_1]\&f[k_2][x_2]\&f[k_3][x_3])$

一点点的小优化

在枚举k_1和k_2的时候就存下$f[k_1][x_1]\&f[k_2][x_2]$这个$bitset$，枚举k_3的时候计算的时候就不用再and一遍。

时间复杂度$O(2^x\frac{n}{w}\binom{k}{3})$

另外第二个暴力

容易发现只用计算其中3个位异或起来都为1的个数，但是直接fwt全部位太浪费了，

于是出题人就想到了更快的方法：

设一个值a，将k个位为分为每a个位一组，不足a个自成一组。

那么对于每三组做一次fwt、一次高维前缀和，做完所有就能够得出对于每三个位异或起来都为1的个数。

时间复杂度 $O(\binom{a}{3}(n+3a*2^{3a}))$

能过的暴力合集

当k比较大的时候，就使用“一个暴力”。

当k比较小的时候，就使用“另外第二个暴力”。

实践证明k大约是150。

任凭风浪起，稳坐钓鱼台（续）

出题人：ilnil

题意

给出n，求：

$\sum_{i=1}^{n}\sum_{j=1}^{n}\mu(i*j)$

$n\le 10^9$

一个10分的方法

$\sum_{i=1}^{n}\sum_{j=1}^{n}\mu(ij)=\sum_{i=1}^{n}\sum_{j=1}^{n}\mu(i)\mu(j)[gcd(i,j)=1]=\sum_{k=1}^{n}\mu(k)(\sum_{i=1}^{n/k}\mu(ik))^2$

预处理μ，枚举k,i累加起来。

时间复杂度$O(n\log n)$

一个20分的方法

发现$\sum_{i=1}^{n/k}\mu(ik)$就是求一个高维前缀和。

时间复杂度$O(n\log\log n)$

接下来我们需要一个亚线性的方法（或者一个难以分析复杂度的方法？）解决这个问题。

一个难以分析复杂度的方法

设$f(n,a)=\sum_{i=1}^n\mu(i)[(i,a)=1]$

原式$=\sum_{i=1}^n\mu(i)\sum_{j=1}^{n/i}\mu(j)[(i,j)=1]\sum_{k=1}^{n/i}\mu(k)[(i,k)=1]$

然后设阈值B，就可以得到

原式$=\sum_{i=1}^B\mu(i)f(n/i,i)^2-f(n/B,i)^2+\sum_{j=1}^{n/B}(f(n/j,j)+2\mu(j)\sum_{k=1}^{j-1}\mu(k)f(n/j,jk)$

考虑如何快速计算$f(n,a)$

设$f(p)=\mu(p)[(a,p)=1]$

显然f是一个积性函数（因为$[(a,p)=1]$也是一个积性函数），可以使用$min25$筛处理

由于计算的是类似μ的前缀和，所以使用$min25$筛递归版会更快一些

对于递归中要预处理的$p2(o)=\sum_{i=1}^o[i\text{是质数}]f(i)$，其中$o\in[1,\sqrt{n}]\cup\{n/i|i\in[1,\sqrt{n}]\}$

可以用$min25$筛一开始预处理$p(o)=\sum_{i=1}^o[i\text{是质数}]$，其中$o\in[1,\sqrt{n}]\cup\{n/i|i\in[1,\sqrt{n}]\}$

然后计算$f(n,a)$的时候，将$p2$赋值为p的同时然后将a中有关质因数的位置更改。

那么计算一次$f(n,a)$的复杂度为$T(n)=w(a)+\sqrt{n}+\sum_{i=1}^n[i*big_i\leq n\&\mu(i)\neq 0]$

其中$w(a)$为a的不同质因数个数

其中big_i为i的最大质因子$(i>1)$，$big_1=\infty$

然后枚举$k<j<=n/B$，容易发现$[(i,j)=1][(i,k)=1]=[(i,jk)=1]=[(i,small(jk))=1]$（其中$small(x)=max\{a|[a|x]\wedge[\mu(a)^2=1]\}$），

对于一个j，有些$small(jk)$已经算过的，所以就记下已经计算过的答案，这样就能减少常数了。

一个能够通过所有数据的方法

现在的难点在于如何快速求出$f(n,a)$，

（如果$\mu(a)=0$，那么可以将a替换成$small(a)$，可以证明这样等价，那么这里设$\mu(a)^2=1$）

对$f(n,a)$继续莫比乌斯反演可以得到这样一个式子：$f(n,a)=\sum_{d|a}f(n/d,d)$

考虑求$f(n,a)$是直接使用上面的方法递归，

当$a=1$时，使用$min25$筛预处理所有的$f(n,1)$

直接跑能够在时限内跑出$5e8$。

我们可以发现我们可以在$O(A\log A)$的时间内预处理出对于所有的$n*a\le A$的$f(n,a)$。

这样递归到$n*a\le A$时可以使用预处理的值。

加上这个小优化之后就能在时限内跑出$1e9$了。

鱼和熊掌不可兼得

出题人： Cold_Chair

题意

这是一道交互题，交互库有一个排列A。

每次你可以询问一个排列B，交互库会返回$\sum_{i=1}^{n}[A_i=B_i]$。

$n\le 5000$

60~68分随机乱搞：

这个是在有了std之后出题人随便想出来的一个乱搞。

结果它秒杀了除了std以外的所有做法，是$O(n\ log\ n)$的，只不过常数较大。

考虑直接搜索非常慢，不如一开始随机一些错排出来，这样就能提前ban掉一些选择，然后继续搜索。

随机一个错排出来的期望次数是$O(e)$的，所以还是能随机出比较多的错排。

于是你发现只要随机个$3n\ log\ n$次就能ban掉所有的选择，

直接这样做能到60分，加一些小优化就可以68分。

47~60分的有理有据做法（By InFleaKing）：

先随机出一个错拍，这样起点就是一个$count=0$的排列了。

此时若交换两个数$p[i]$和$p[j]$，$count$发生了改变，说明要不$i=p[j]$，要不$j=p[i]$。

我们给i和j之间连一条无向边，那么整个图一定是若干个环。

找到每一个环，对每一个环再用一次询问确定方向，这样做询问次数$<=n^2/2$，有47分。

考虑找边的过程可以多线程优化，合理分块就可以做到$O(n\sqrt{n})$。

100分的睿智做法（By ilnil）：

找边的过程其实可以用分治优化。

假设n是偶数，那么一共有$n*(n-1)/2$的边，我们做$(n-1)$轮，每次拿出$n/2$条互不相交的边。

每一轮，把这些边同时交换，如果$count$变化了，那我们分治，直到找到所有有用的边。

显然的是，有用边的总数是$O(n)$，而一条边最多带来log次查询，所以这里的复杂度是$O(n\ log\ n)$。

那么问题在于如何做$(n-1)$轮，每次拿出$n/2$条互不相交的边，使得每条边被恰好拿一次。

这是一个简单的构造题:
把点从0到$n-1$标号，第$x+1$轮包含所有满足$(i+j)=x(mod\,n-1)$的边(i,j)和 $(p,n-1)$满足$2p=x(mod\,n-1)$。

如果n是奇数，那么就多加一个点，不管连向额外点的边。

一点点小优化：

当一个点已经找到了2条边的时候，后面关于这个点的边就不用加入了。

最后查询次数能在$n\leq 5000$控制在$10n$以内。