# 前言：

再看这篇文章之前，需要理解FFT和NTT，对代码细节也要很清楚。

还是有这么一个问题模型：

A数组是a1 a2 a3 a4 a5….an

B数组是b1 b2 b3 b4 b5….bm

它们代表多项式系数，求多项式乘法后结果多项式系数

**系数对***p***取模**，且**不保证** p可以分解成 p =a\*2^k+1的形式。

1<=lena,lenb<1e5,0<=ai,bi<1e9,2<=p<=1e0+7

这个问题看似和之前普通多项式乘法一样，但其实比较麻烦，因为0<=ai,bi<1e9，之前系数都是100以内，在1e9以内意味着用纯FFT可能导致变换后的数组超double(远超),FFT不行;P不保证 p =a\*2^k+1,NTT不行。

两种方法都是正解：(1)把NTT推广到任意模数的形式。(2)通过某种方法使得精度符合要求。HEX代表ai和bi最大值，len代表lena和lenb最大值

# NTT任意模数：

这种方法有些地方叫三数NTT，三数NTT顾名思义,找三个费马质数，使得他们的乘积超过 HEX\*HEX\*len,选三个满足NTT性质且乘起来  大于1e23的模数分别NTT，最后中国剩余定理合并。有以下三个等式：

ans%p1=a1 ans%p2=a2 ans%p3=a3

合并前2个得到ans%P=A

# 分部FFT：

这种方法有些地方叫MTT，