# 高精度所有算法

## 总体概括：

一定要用面向对象的方式实现，尽量封装得严密，因为代码细节多，代码量大。用不用重载运算符？不要重载运算符，就用定义函数来完成，因为C++不会自动内存管理，重载运算符容易疏忽内存管理而内存泄漏。

比如BT\*a=BT(“4”) ；BT\*b=BT(“4”)；BT\*c=A\*b；那么a和b什么时候释放，什么时候不用释放，是需要分场合手动完成的，重载\*不如定义个mul方法 好

## 类成员：

一个数组指针number，一个sz表示当前大整数位数，一个sign表示符号，-1负数，0是0，1是正数

## 构造方法：

BigInteger(int up=MAX)无参构造方法，用来定义个变量大整数而不初始化值，up表示number数组长度，默认是MAX

BigInteger(char\*s)传入一个字符串，把字符转换大整数，number数组长度根据字符串长度而定

BigInteger(ll s,int up=MAX)把ll变成大整数，number数组长度根据up而定

~BigInteger()析构方法，释放number指针

## 加法减法：

没有高级算法，都是用o(n)的复杂度来运算，但是处理很多种情况

比较麻烦。。。要注意符号问题

## 乘法：

### 朴素相乘：

时间是o(n\*m),长度较小时快于其他算法。但数据量大时效率不高

### FFT/NTT(Schönhage-Strassen算法)：

大数乘法算法最快的，n\*logn,但FFT浮点数计算不能超过double，NTT受大素数P影响，HEX\*HEX\*sz不能大于P，意味着压位受影响。而且效率要在数据量大时候才有体现。是python高精度类库的算法

思路是把大数当作多项式，把多项式由常规的系数表示变为点值表示，而2个点值表示的多项式的乘法就是对应系数相乘再相加，最后再把结果来一次逆变换变回系数表示，久能得到结果。

### Toom-Cook：

基于分治算法，暂时不太懂等待填坑，复杂度忘了好像是n^(1.5)，也是java高精度类库的算法，可以满足一般需求,在巨大数量级下效率不如FFT，但是中等大小的数据效率较好。

## 除法：

### 试除法：

复杂度n\*m\*HEX，比较简单的暴力法

### 二分试除法：

复杂度n\*m\*logHEX，稍微复杂的暴力法

### NTT/FFT+牛顿迭代:

牛顿迭代具体可以看另一个模板说明。

对于大数a/b;牛顿迭代公式求1/b;求得迭代函数是x=2\*x-b\*x\*x=x\*(2.00-bx)

X初始化一个很小的浮点数，变成了大数乘法。复杂度a\*N\*logN ，其中N=changeN(n+m), changeN(n+m),代表比n+m大的第一个2的多少次幂数；a代表常数，这个算法有很大的常数。

那么X初始值怎么取，可以把b向上取，比它大的除了第一位其余都是0的数的倒数，比如b=128888888888888888,在不压位情况下，x=1/200000000000000000.

对于迭代倒数过程中，我们应该限制位数，x的有效数字应该是1 2 4 8 16 32…倍增进行，而不是每次都要很大的有效数字。牛顿迭代的性质就是每次精度倍增。

如果不利用这个容易写成nlogn^2的算法。

对于x的取值，更好的做法是，取x前几位a，用double计算a倒数保留前几位有效数字,并且把它当做x去迭代，51nod1029有人用250ms过了题

## 取模：

### 试除法/二分试除法：

暴力的算法，方法等同于大数除法，时间复杂度相同，只不过返回的数据不同

### 被除数 - 商\*除数:

适用于NTT/FFT+牛顿迭代的大数除法，有着更大得常数，因为要在除法基础上再进行1次乘法和1次减法

## 乘方：

快速幂+上面介绍的加法和乘法：

对于a^(b)

时间复杂度是：lenb\*logHEX\*N\*logN

N=changeN(2\*lena), 代表比2\*lena大的第一个2的多少次幂数；

## 开方：

不管开几次方，都可以用牛顿迭代法求得，但要涉及上面介绍的其他运算，时间及复杂度等同于这些运算总和。

根据牛顿迭代性质，精度是倍增的，一定控制精度，否则时间复杂度会很高，牛顿迭代求得的迭代公式是x=(a/x+x)/2,较快的思路是算好前1位的然后迭代loglena次，想要长度的精度倍增需要保证初始值前1位是正确的 ,注意要控制所有数长度