实验报告

针对大整数运算，在不利用直接的乘除操作之后，是可以通过位运算进行实现。从计算机的本质来说，数字在计算机中是二进制的储存方式，因此可以将其转化为二进制数组进行表示，通过依次的位运算和按位与 1 操作可以建立二进制数组，利用之前的多项式乘法的知识可知两个数字相乘的位数最多是两个数字的位数相加，所以不妨建立一个长度为 64*2 的数组来储存乘积，由于是数组的缘故，所以不一定需要考虑进位操作，这一操作会在之后的取余操作中进行说明。相乘操作还是从多项式的乘法得到启发，直接对于相应大数组的 index=两个小数组的 index 相加，只要依次叠加即可。考虑到会越界的问题，所以可以移动位数之后与元数组进行比较就可以，在取余操作之中，可以从高位依次向低位取余，因为高位取余直接是可以倍数的缘故，所以只要及时处理越界问题即可。这样就可以从本质上处理时间复杂度。

实验中遇到的 bug：起初是利用了一个 mask 去依次移位来与操作，后来从计算机的本质来说，储存方式就是二进制，所以选择改进储存的数组的方式。之后处理后来的数组 index 越界问题的时候，此处在杨斯凡的帮助下修改完成。