求解思想

传统方法实现多项式相乘

- 给定两个次数界为n的多项式系数<a0，a1，a2，...，an>与<b0，b1，b2，...，bn>，如果使用传统的多项式相乘，那么时间复杂度将达到O(n^2)

利用FFT实现多项式相乘

- 由于插值多项式的唯一性，因而可以使用多项式的点值形式来唯一确定一个多项式，所以可以取两个多项式在n个相同点处的取值<y0，y1，y2，...，yn>与<z0，z1，z2，...，zn>作乘法，那么它们的乘积就是所求多项式的点值表达即<y0*z0，y1*z1，y2*z2，...，yn*zn>, 该操作的时间复杂度为O(n)

- 如果任意取点值对，那么从多项式到点值对以及从点值对转换为多项式的时间复杂度均为O(n^2), 时间复杂度并没有降低。这时可以把n个点取在n个n次单位复数根处,由于n次单位复数根具有一些特殊的性质（消去引理、折半引理），因而可以使得转换与还原的时间复杂度均降为O(nlgn)。

- 根据以上两点，fft实现两个多项式相乘的总时间复杂度为O(nlgn)。

利用FFT实现大整数乘法

- 首先需要将输入的两个数字字符串转换为两个多项式，数字字符串中的每一位都为多项式的一个系数。这里选取两个多项式次数界较大者记为t，找到一个非负整数k，使得2^(k-1) < t <= 2^k (k >= 0),记n = 2^k。通过在字符串之前添加有限个字符'0'使得次数界变为2×n，之后将数字字符串中的每一位作为多项式的一个系数(使用int类型存储，注意顺序从后往前)，例如31234 转换为<4，3，2，1，3，0，0，0，0，0，0，0，0，0，0，0>(n=8)

- 转换后的两个多项式可以用上面提到的fft实现相乘，再经过fft的逆过程还原为最终得到的结果多项式的系数<a0，a1，a2，a3，....>，可以通过a0 + a1*10 + a2 * 10^2 + a3 * 10 ^3.....计算出最终的结果，我使用ai % 10作为本位系数、ai / 10加到高位(i从0到2n)(具体细节见代码)实现在O(n)的时间复杂度下计算出最终的结果并存放到字符串中。

存在的问题

1. 最大的问题应该是精度的问题，代码中有几个步骤是从浮点数转化为整数，更多的步骤则是做浮点数的乘法，我不能够确定什么时候会出错，起初我尝试使用long double 与long long，用python生成最高2 ^20位的两个数字相乘进行测试，测试竟然都能够通过。

2. 代码写的很差，算法的执行时间比较长。在leetcode平台上的第一次成功提交用时100ms左右。由于要求是100位以下的数据，我把类型改为double 与 int，同时传递参数的方式改为引用传参，测试几次，最好的情况下用时32ms。

3. 尚未发现的其它错误

关于提交文件

1. commit.cpp : 主文件

2. generatetestdata.py 测试数据生成代码

3. test.data 提供的一份测试数据

4. 运行结果截图