2018年全国高校密码挑战赛决赛答辩论文

赛题一:FCSR序列的有理逼近

PKU-NSS：张样攀、董佶圣

指导老师：徐茂智

单位：北京大学数学科学学院

Abstract

我们在LINUX平台上使用GMP大数库[3]实现了文献[1]中提出的有理逼近算法，并成功计算出赛题所给序列的有理逼近.之后我们又对算法进行了改进和代码优化，使得算法复杂度由$O(N^2\log(N))$优化至$O(N^2)$级别，仅需28 秒左右就能够计算出最优逼近.最终算出的逼近结果过于庞大，列于附件*ans.txt*中.

1 问题描述

设$a(n)=(a_0,a_1,a_2,\cdots,a_{n-1})$是一条有限二元序列，即$a_i\in\{0,1\}$,$0\leqslant i\leqslant n-1$.若有理分数$\frac{p}{q}$满足$q$是正奇数，$\gcd((p,q))=1$，并且

$$p\equiv q(a_0+a_1\cdot 2+\cdots+a_{n-1}\cdot 2^{n-1}) \bmod 2^n,$$

则称$\frac{p}{q}$是序列$a(n)$的有理分数表示.

附件“*sequence.txt*”中是长度$n=1966000$的有限二元序列$a(n)=(a_0,a_1,a_2,\cdots,a_{n-1})$，对$1\leqslant k\leqslant n$,求有限序列$a(k)=(a_0,a_1,a_2,\cdots,a_{k-1})$的有理分数表示的序列越长（即$k$越大），得分越高；在$k$ 值相等的情况下，$\Phi(p,q)=\max\{|p|,|q|\}$越小，得分越高.

2 有理逼近算法

带进位反馈移位寄存器（简称FCSR）由美国学者Klapper 和Goresky 于1993 年提出.与传统的二元域上线性反馈移位寄存器相比，FCSR 通过引入若干进位寄存器，实现了有理分数2-adic 展开序列的快速生成.文[1]是关于FCSR 序列的一个比较全面的综述，给出并证明了计算序列有理分数表示的有效方法——有理逼近方法(Rational Approximation Algorithm).

该算法的证明用到了$p-adic$数以及格基理论的一些性质，这里我们就不再展开叙述.算法的优势在于其复杂度约为$O(N^2\log(N))$，其中N为所求序列的长度.

2.1 算法框架

为算法中方便说明，我们以整数环上的二元数组为基本操作单位，并有定义如下：

定义 1. *1. 若*$f=<f_1,f_2>\in \mathbf{Z}\times\mathbf{Z}$，*则*$\Phi(f)=\max(|f_1|,|f_2|)$.

2. 若$f=<f_1,f_2>\in \mathbf{Z}\times\mathbf{Z}$，$d\in\mathbf{Z}$，*则*$d\cdot f=<d\cdot f_1,d\cdot f_2>$.

3. 若$f=<f_1,f_2>$,$g=<g_1,g_2>\in \mathbf{Z}\times\mathbf{Z}$，*则*$f+g=<f_1+g_1,f_2+g_2>$.

以下是算法实现的伪代码：

算法 1 Rational Approximation

输入: 有限二元序列$a(n)=(a_0,a_1,a_2,\cdots,a_{n-1})$

输出: $a(n)$的有理分数表示

读入前$k-1$个比特$a_0,a_1,a_2,\cdots,a_{k-2}$直到找到第一个非零比特$a_{k-1}$
$\alpha=a_{k-1}\cdot 2^{k-1}$
$f=<0,2>$
$g=<2^{k-1},1>$
while 还有更多比特未处理 **do**
    输入一个新比特a_k
    $\alpha=\alpha+a_k\cdot 2^k$
    if $\alpha\cdot g_2-g_1\equiv 0 \pmod{2^{k+1}}$ **then**
        $f=2\cdot f$
    else if $\Phi(g)<\Phi(f)$ **then**
        找到使得$\Phi(f+d\cdot g)$最小的奇数d
        $\langle g,f\rangle\leftarrow\langle f+d\cdot g,2\cdot g\rangle$
    else
        找到使得$\Phi(g+d\cdot f)$最小的奇数d
        $\langle g,f\rangle\leftarrow\langle g+d\cdot f,2\cdot f\rangle$
    end if
    $k=k+1$
end while
return $g=<g_1,g_2>$

最终输出结果$g=<g_1,g_2>$，g_2是奇数，并且有$a(n)\equiv g_1/g_2 \mod 2^n$.由文献可知，我们有$\log_2(\phi(g))<n/2$.

可以证明下述定理：

定理 1. 对于一长度为n的有限二元序列(n)，若存在两个互质整数p与q（q为奇数），使得$a(n) \equiv p/q \mod 2^n$，并且$\max(|p|,|q|) < 2^{(n-1)/2}$，那么这样的整数对唯一.

Proof. 这个定理的证明是简单的，假设有两组这样的整数$< p_1, q_1 >$与$< p_2, q_2 >$满足这个条件，那么有：

$$a(n) \equiv p_1/q_1 \equiv p_2/q_2 \mod 2^n \tag{1}$$

由于q_1与q_2都是奇数，所以由上式推出

$$p_1 \cdot q_2 \equiv p_2 \cdot q_1 \mod 2^n,$$

在$\max(|p_i|,|q_i|) < 2^{(n-1)/2}, i = 1, 2$ 的假设下，又有

$$-2^{n-1} < p_1 \cdot q_2, p_2 \cdot q_1 < 2^{n-1}.$$

结合两个乘积在模2^n 下相等可以知道，$p_1 \cdot q_2 = p_2 \cdot q_1$.

于是，这两组整数对应的分数相等，即$p_1/q_1 = p_2/q_2$.结合互质性质我们可以知道必然有$p_1 = p_2$，$q_1 = q_2$，即满足定义的整数存在时一定只有唯一一组. □

故对于Rational Approximation算法关于序列$a(n)$有理逼近结果的两个整数p, q（根据算法原理两个整数自然互质，并且是序列$a(n)$的有理表示），如果其另外再满足上述定理的极大性限制，那么其一定是最优有理逼近.

2.2 算法效率分析

按照文献中算法复杂度分析部分的解释，在本算法中，当读入第$k+1$个比特时，如果$\alpha \cdot g_2 - g_1 \equiv 0 \pmod{2^{k+1}}$,那么意味着我们对$a(n)$前$k$位的有理逼近结果也是前$k+1$比特的逼近结果.此时我们不需要纠错，单轮的复杂度为$O(k)$(对整数f_1与f_2分别乘以2).

如果$\alpha \cdot g_2 - g_1 \not\equiv 0 \pmod{2^{k+1}}$，算法则需要根据上一轮的结果进行纠错来达到最优有理逼近，文章中提到此时的单轮复杂度主要部分在于两个$O(k)$级别的大整数做除法，在使用文献[2]提出的算法优化乘法的时候，其复杂度为$O(k \log(k))$.

而关于判断部分，计算$\alpha \cdot g_2 - g_1$需要一个开销为$O(k \log(k))$级别的乘法，但文章中提到，这个乘法可以通过一定的方式优化为几个k位整数的加法操作.故而其忽略了这一部分的复杂度.

我们假设序列足够随机，使得以上两种情形发生概率相等，算法主要复杂度在于需要纠错的情形，其整体复杂度应当为$O(N^2 \log(N))$.

而在下面的算法优化过程中，我们将实现判断部分的优化，从而尽量减少时间开销.但最后发现文献所分析的复杂度的过程并不正确，虽然按其实现的未优化算法的复杂度的确为$O(N^2 \log(N))$.

2.3 算法实现与实际效果

我们于linux平台上使用C语言下的GMP大整数库实现了有理逼近算法，并对赛题一给定序列成功进行了有理逼近.

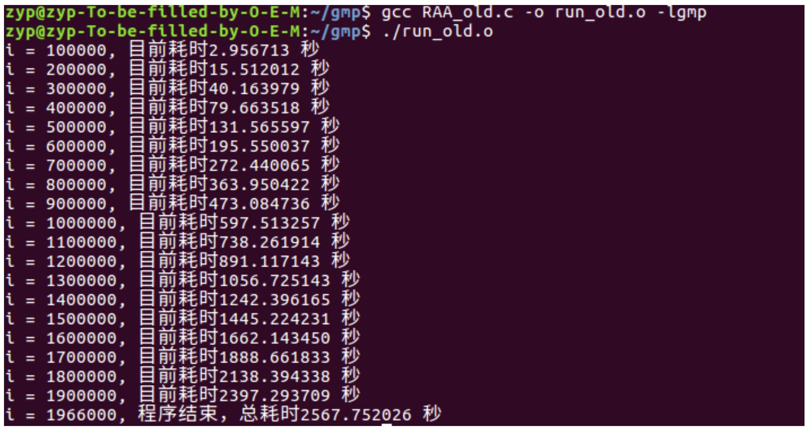

Figure 1: 程序输出结果

这是一个串行算法，在个人pc平台（配置为：i5-6500 3.20 GHz）上共耗时约2500余秒. 在前期，算法的复杂度基本按照$O(N^2\log(N))$的曲线增加.而当算法使用的变量长度大于一百万位左右时，算法的效率明显加快，与估计不符. 经检验，是原始随机数的随机性不够所导致，从第1310716位开始没有继续进行纠错过程，而之前的计算过程中每一位的纠错概率大概为50%，这说明原始随机数的131万余位后的生成方法不当，故后期的时间开销不符合我们的预设.

最后输出整数对$g=<g_1,g_2>$，满足g_2为奇数，且g_1，g_2互质.这里由于长度太大，我们将g_1,g_2的详细结果放在了附件之中.详见文档$ans_old.txt$.而算法实现的c代码也在附件中，名称为$RAA_old.c$.

因为g_1与g_2的大小满足定理1的限制，所以我们得到的这两个整数的确是赛题所给序列的最优有理逼近.

而我们验证计算的结果也证明了其的确为一有效的有理表示.

3 改进有理逼近算法

我们在上述有理逼近算法的基础上，为缩减时间复杂度对算法进行了改进，改进后的算法如下：

算法 2 Improved Rational Approximation

输入: 有限二元序列$a(n) = (a_0, a_1, a_2, \cdots, a_{n-1})$

输出: $a(n)$的有理分数表示

读入前$k-1$个比特$a_0, a_1, a_2, \cdots, a_{k-2}$直到找到第一个非零比特$a_{k-1}$
$\alpha = a_{k-1} \cdot 2^{k-1}$
$f = < 0, 2 >$
$g = < 2^{k-1}, 1 >$
$F = \frac{\alpha \cdot f_2 - f_1}{2^k}$
$G = \frac{\alpha \cdot g_2 - g_1}{2^k}$
while 还有更多比特未处理 **do**
    输入一个新比特a_k
    $\alpha = \alpha + a_k \cdot 2^k$(这一行实际上并不需要)
    if $a_k = 1$ **then**
        $F = F + f_2$
        $G = G + g_2$
    end if
    if $G \equiv 0 \pmod 2$ **then**
        $f = 2 \cdot f$
        $\langle G, F \rangle \leftarrow \left\langle \frac{G}{2}, F \right\rangle$
    else if $\Phi(g) < \Phi(f)$ **then**
        找到使得$\Phi(f + d \cdot g)$最小的奇数d
        $\langle G, F \rangle \leftarrow \left\langle \frac{F + d \cdot G}{2}, G \right\rangle$
        $\langle g, f \rangle \leftarrow \langle f + d \cdot g, 2 \cdot g \rangle$
    else
        找到使得$\Phi(g + d \cdot f)$最小的奇数d
        $\langle G, F \rangle \leftarrow \left\langle \frac{G + d \cdot F}{2}, F \right\rangle$
        $\langle g, f \rangle \leftarrow \langle g + d \cdot f, 2 \cdot f \rangle$
    end if
    $k = k + 1$
end while
return $g = < g_1, g_2 >$

改进的主要思想是将判断过程的大整数乘法$\alpha \cdot g_2$省去，这是原始算法中开销最大的部分.在说明算法前，我们需要一个小引理.

引理 1. *对于Rational Approximation算法中使用到的两个整数对f与g来说，当算法对第k位比特纠*

错完成后，我们有：

$$\alpha \cdot f_2 - f_1 \equiv \alpha \cdot g_2 - g_1 \equiv 0 \mod 2^k$$

.

这个引理的证明是依赖于Rational Approximation算法的有效性，当Rational Approximation算法有效时，这个引理是显然的.

3.1 算法效率分析

由上面的引理，注意到$\alpha \cdot f_2 - f_1$与$\alpha \cdot g_2 - g_1$之间，具有格的可加性，于是我们考虑用以下的两个参数代替$\alpha \cdot g_2 - g_1$，令：

$$F = \frac{\alpha \cdot f_2 - f_1}{2^k},\quad G = \frac{\alpha \cdot g_2 - g_1}{2^k}$$

于是当$G \equiv 0 \mod 2$时，也就意味着$\alpha \cdot g_2 - g_1 \equiv 0 \mod 2^{k+1}$.这也就是我们算法中的判断条件.而我们在辅助参数$F, G$的帮助下，计算$G$每轮最多在原有基础上增加两个与$d$的乘法，两个除以2的除法，以及若干个加法.关于$d$的乘法我们之后再统一讨论其复杂度，除此之外的复杂度是$O(k)$级别的.

这样一来，我们将$\alpha \cdot g_2$的大额乘法开销，换成了几个小额的加法和移位除法的开销.

之后我们考虑那些与d相乘的乘法部分开销.接下来我们试图证明这部分的总开销实际上应当是$O(N^2)$级别.

我们需要一个关键的引理：

引理 2. $\sum(\log|d|) \sim O(N)$.

其中d遍历算法的每一次纠错过程.

首先，为了证明这个引理，我们需要对$|d|$的大小进行估计.

当$\Phi(f) > \Phi(g)$时，d是所有奇数中使得$\Phi(f + d \cdot g)$最小的那个（当$\Phi(f) \leqslant \Phi(g)$时是使得$\Phi(g + d \cdot f)$最小），不妨以前者为例，即$d$最小化$\max\{|f_1 + d \cdot g_1|, |f_2 + d \cdot g_2|\}$. 考虑两个函数方程

$$y_1 = |f_1 + g_1 \cdot x|, y_2 = |f_2 + g_2 \cdot x|,$$

我们先求令$y(x) = \max\{y_1, y_2\}$最小的横坐标取值x_0.由于y_1, y_2两个函数均为先减后增的线性函数，不难得出$y(x)$同样为先减后增的函数，且最小值取在两函数图像交点处.因此，最小化$y(d)$的奇数d在x_0附近,且$|d - x_0| < 2$.

若$|g_1| = |g_2|$，则为函数图像仅有一个交点或两图像完全重合的平凡情况，下述讨论$|g_1| \neq |g_2|$的情况，此时函数图像交点的横坐标为$x_1 = \frac{f_1 - f_2}{g_2 - g_1}$和$x_2 = -\frac{f_1 + f_2}{g_1 + g_2}$.

当g_1, g_2异号时，有$|g_2 - g_1| > |g_2 + g_1|$,故

$$\left|\frac{-f_2 g_1 + g_2 f_1}{g_2 - g_1}\right| < \left|\frac{f_2 g_1 - g_2 f_1}{g_2 + g_1}\right|$$

即

$$\left|\frac{f_2g_2-f_2g_1+g_2f_1-g_2f_2}{g_2-g_1}\right|<\left|\frac{f_2g_2+f_2g_1-g_2f_1-g_2f_2}{g_2+g_1}\right|,$$

$$\left|f_2+g_2\cdot\frac{f_1-f_2}{g_2-g_1}\right|<\left|f_2-g_2\cdot\frac{f_1+f_2}{g_2+g_1}\right|,$$

也即

$$|y_2(x_1)|<|y_2(x_2)|$$

因此，最小点值为$x_1=\frac{f_1-f_2}{g_2-g_1}$.同理，当$g_1,g_2$同号时，最小点值为$x_2=-\frac{f_1+f_2}{g_1+g_2}$.

由于d为奇数，故当g_1,g_2异号时，有

$$\frac{f_1-f_2}{g_2-g_1}-2<d<\frac{f_1-f_2}{g_2-g_1}+2$$

当g_1,g_2同号时，

$$-\frac{f_1+f_2}{g_1+g_2}-2<d<-\frac{f_1+f_2}{g_1+g_2}+2$$

综上，有

$$|d|<\left|\frac{f_1+\varepsilon f_2}{g_1+\varepsilon g_2}\right|+2.$$

其中，$\varepsilon=\dfrac{g_1g_2}{|g_1g_2|}$，本文之后的所有分数中的$\varepsilon$都是其分母两个加项符号的乘积，不再赘述.

接下来，将通过估计$\left|\dfrac{f_1+\varepsilon f_2}{g_1+\varepsilon g_2}\right|$来证明引理.

在第k步开始时，此时$\langle f,g\rangle$未更新，记为$\left\langle f^k,g^k\right\rangle$，令

$$B^k=\begin{cases}f^k,\Phi(f^k)>\Phi(g^k)\\ g^k,else\end{cases}\quad S^k=\begin{cases}f^k,\Phi(f^k)\leqslant\Phi(g^k)\\ g^k,else\end{cases}$$

于是可记$D_k=\left|\dfrac{B_1^k+\varepsilon B_2^k}{S_1^k+\varepsilon S_2^k}\right|$，则根据上述证明，若第$k$轮进行纠错，则有：

$$|d_k|<D_k+2,$$

其中d_k是第k轮进行纠错时的最小化奇数，设整个序列中所有发生了纠错的位置为$1\le k_1<k_2<k_3<\cdots$, 只需要证明

$$\sum_{i\geq 1}\log(D_{k_i})\sim O(N)\quad .$$

注意到，在$k_i<k<k_{i+1}(i\geq 1)$时，序列不进行纠错，故$\left\langle f^{k+1},g^{k+1}\right\rangle=\left\langle 2f^k,g^k\right\rangle$, 于是显然有$D_{k+1}\leqslant 2\cdot D_k$.于是有:

$$D_{k_{i+1}}\leqslant 2^{k_{i+1}-k_i-1}\cdot D_{k_i+1},$$

即

$$\log(D_{k_{i+1}}) \leqslant k_{i+1} - k_i - 1 + \log(D_{k_i+1}).$$

故我们有

$$\sum_{i\geq 1}\log(D_{k_i}) \leqslant \log(D_{k_1}) + \sum_{i\geq 1}[\log(D_{k_i+1}) + k_{i+1} - k_i - 1]$$
$$\sim \sum_{i\geq 1}[\log(D_{k_i+1}) - 1] + O(N)$$

结合这些，往证：对$\forall i$，D_{k_i+1}有界.

这里我们需要一个显然的结论：$\Phi(B^{k_{i-1}} + d^{k_{i-1}} \cdot S^{k_{i-1}}) \leq \Phi(B^{k_i} + d^{k_i} \cdot S^{k_i})$，即：$\Phi(g^{k_{i-1}+1}) \leq \Phi(g^{k_i+1})$.这个性质可以利用有理逼近算法的有效性证明，即随着序列位数的输入，当前最佳逼近的结果在范数Φ 下只会越来越大（最多不变）.

回到引理的证明，因为D_{k_1+1}有限，故只考虑$i > 1$的情形，当算法在第k_{i-1}位及第k_i位完成纠错后，有两种情况：

1. 当$\Phi(f^{k_i+1}) > \Phi(g^{k_i+1})$时：

$$D_{k_i+1} = \left|\frac{f_1^{k_i+1} + \varepsilon f_2^{k_i+1}}{g_1^{k_i+1} + \varepsilon g_2^{k_i+1}}\right|$$

   这之下分两种情形：

   - $\Phi(f^{k_i}) > \Phi(g^{k_i})$时，因为$|g_1^{k_i+1} + \varepsilon g_2^{k_i+1}| > \Phi(g^{k_i+1})$，$f^{k_i+1} = 2 \cdot g^{k_i} = 2 \cdot g^{k_{i-1}+1}$，所以

$$\begin{aligned} D_{k_i+1} &= \left|\frac{f_1^{k_i+1} + \varepsilon f_2^{k_i+1}}{g_1^{k_i+1} + \varepsilon g_2^{k_i+1}}\right| \\ &= 2 \cdot \left|\frac{g_1^{k_{i-1}+1} + \varepsilon g_2^{k_{i-1}+1}}{g_1^{k_i+1} + \varepsilon g_2^{k_i+1}}\right| \\ &< 2 \cdot \frac{2\Phi(g^{k_{i-1}+1})}{\Phi(g^{k_i+1})} \\ &< 4 \end{aligned}$$

   - $\Phi(f^{k_i}) \leqslant \Phi(g^{k_i})$时，因为$|g_1^{k_i+1} + \varepsilon g_2^{k_i+1}| > \Phi(g^{k_i+1})$, $g^{k_i} = g^{k_{i-1}+1}$，$f^{k_i+1} = 2 \cdot f^{k_i}$，

所以

$$
\begin{aligned}
D_{k_i+1} &= \left| \frac{f_1^{k_i+1} + \varepsilon f_2^{k_i+1}}{g_1^{k_i+1} + \varepsilon g_2^{k_i+1}} \right| \\
&= 2 \cdot \left| \frac{f_1^{k_i} + \varepsilon f_2^{k_i}}{g_1^{k_i+1} + \varepsilon g_2^{k_i+1}} \right| \\
&< 2 \cdot \frac{2\Phi(f^{k_i})}{\Phi(g^{k_i+1})} \\
&\leqslant 4 \cdot \frac{\Phi(g^{k_i})}{\Phi(g^{k_i+1})} \\
&= 4 \cdot \frac{\Phi(g^{k_{i-1}+1})}{\Phi(g^{k_i+1})} \\
&< 4
\end{aligned}
$$

综合这两种情况，可以知道此时$D_{k_i+1} < 4$.

2. 当$\Phi(f^{k_i+1}) \leqslant \Phi(g^{k_i+1})$时：

$$
D_{k_i+1} = \left| \frac{g_1^{k_i+1} + \varepsilon g_2^{k_i+1}}{f_1^{k_i+1} + \varepsilon f_2^{k_i+1}} \right|
$$

不妨假设$\Phi(f^{k_i}) > \Phi(g^{k_i})$,于是$f^{k_i+1} = 2g^{k_i}$考虑下列函数的最低点：

$$
y(x) = \max\{|f_1^{k_i} + x \cdot g_1^{k_i}|, |f_2^{k_i} + x \cdot g_2^{k_i}|\}
$$

易知d_{k_i}是所有奇数中使得$y(x)$最小的那个数.于是再考虑:

$$
\begin{aligned}
\overline{y}(x) &= \max\{|g_1^{k_i+1} + x \cdot f_1^{k_i+1}|, |g_2^{k_{i+1}} + x \cdot f_2^{k_i+1}|\} \\
&= \max\{|f_1^{k_i} + d_{k_i} \cdot g_1^{k_i} + x \cdot f_1^{k_i+1}|, |f_2^{k_i} + d_{k_i} \cdot g_2^{k_i} + x \cdot f_2^{k_i+1}|\} \\
&= \max\{|f_1^{k_i} + (d_{k_i} + 2x) \cdot g_1^{k_i}|, |f_2^{k_i} + (d_{k_i} + 2x) \cdot g_2^{k_i}|\} \\
&= y(d_{k_i} + 2x)
\end{aligned}
$$

由之前的证明可知，此时D_{k_i+1}是使得$\overline{y}(x)$最小的实数的绝对值，同时有$d_{k_i}+2x$是使得$y(x)$最小的实数，即有：$D_{k_i} = |d_{k_i} \pm 2D_{k_i+1}|$.由于$d_{k_i}$的最小化特性以及函数$y(x)$ 是单谷函数可知，$D_{k_i+1} < 1$.

当$\Phi(f^{k_i}) \leqslant \Phi(g^{k_i})$时，同理可证$D_{k_i+1} < 1$，于是此时$D_{k_i+1} < 1$.

综合以上两种情形，对$\forall i > 1$，D_{k_i+1}有界，故$\sum_{i\geq 1}[\log(D_{k_i+1}) - 1] \sim O(N)$，于是引理得证.

这个引理使得我们发现，$\log|d|$的平均大小不可能很大，经实际运算检验，绝大部分的d的绝对值为1.考虑算法所有循环中和d相关的乘法与除法，我们可以知道GMP大数库中基础的除法的复杂度为$O(QM)$，其中Q是被除数的长度，N 是除数的长度，而$M = Q - N$是最后商的长度.根据我们实现的寻找d最小化$\Phi(f + d \cdot g)$的算法，我们可以知道$M \sim \log|d| + 1$.于

是每一轮中与d相关的乘法除法复杂度为$O(k\log|d|)$，于是整个算法中与d相关的乘法复杂度为$O(\sum_{i=1}k_i\log|d_{k_i}|)\sim O(N)\cdot O(\sum\log|d|)=O(N^2)$.

而我们的改进算法中，所有乘除法都是与d相关的（不包括乘除2这种移位操作），而剩下的加减法的综合复杂度也是$O(N^2)$，故改进算法的整体复杂度为$O(N^2)$.

3.2 算法实现与实际效果

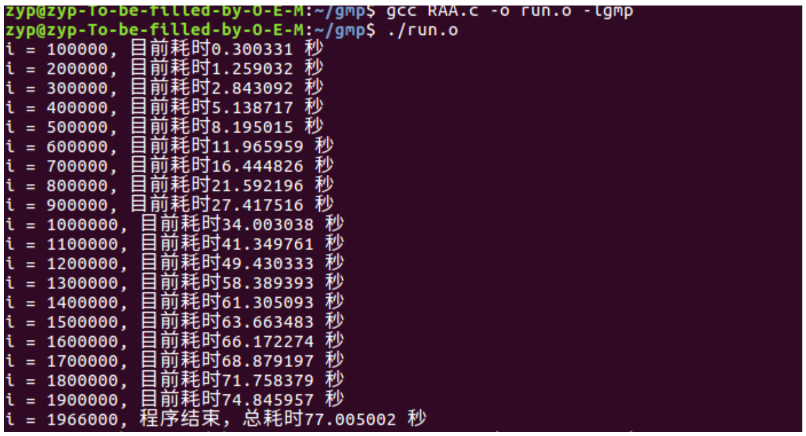

Figure 2: 改进后程序输出结果

在同一台电脑同一个环境下，优化算法运行的时间开销为70余秒.并且其时间开销曲线在前一百三十万比特下符合$O(N^2)$的结论，当序列更长时，由于所解序列的随机性不够使得算法不需要纠错，故而算法速度提高.

最后输出的结果与改进前算法结果一致，且经过验算，输出的两个数的确是所求的最佳有理逼近结果，说明改进后算法的有效性.

4 效率优化

在改进有理逼近算法的基础上，为了进一步缩短时间复杂度，我们进行了一系列尝试.

我们发现在是否进行纠偏判断时，判断条件为$G\equiv 0 \pmod 2$，其判断标准只需依赖G的低位比特，因此可以简化大数G 的计算耗时；而进行d 的求解时，由上述过程可知d的大小仅与被除数和除数的前若干位相关，也即与f和g的前若干位.这样一来，通过使用高位有限位比特和低位有限位比特，就可以精确地计算出d的大小，以及是否需要纠偏.

这减少了f 和g等大整数的计算耗时，从而实现优化算法效率.

另一方面，我们注意到当出现连续两轮都纠偏时，第二轮纠偏的参数d几乎都是±1.这是因为连续两轮纠偏会使得第二轮的参数$D<4$，故由$|d|<D+2$可知此时d被限制在有限几个小奇数之

中.于是$d=\pm 1 or \pm 3$的概率极大.因此我们可以在连续两轮都纠偏时先行判断一次d是否等于± 1其一，这个判断的耗时低于通用的d求解函数，可以起到一定的加速效果.我们在代码中实现了这一优化，可以提高大约3%的速度.这种优化方法原理简单，故下面不再详细描述，我们主要说明利用高低位估计的优化方法.

4.1 优化方式

根据上述思想，我们引入几个新的变量f_low、f_high、g_low、g_high、F_low、G_low，分别代表原始相应变量的有限高位比特和低位比特，长度分别以$highlevel$、$lowlevel$ 控制.其中

$$X\_low \equiv X \mod 2^{lowlevel},\ \ X\_high = \lfloor X/2^{highlevel} \rfloor$$

$lowlevel$由我们设定，$highlevel$由算法动态管理，使得X_high的长度在一定范围之内（大约在lowlevel长度的一倍到两倍之间）.

优化后的数据结构包含三个部分，第一部分称之为记录点，记录上一次精确计算出的f、g、F、G这几个数据；第二部分称之为预测位置，由f_low、f_high、g_low、g_high、F_low、G_low等数据组成；第三部分称之为路径，记录了从上一个记录点至今的操作过程，即与每一轮是否纠偏，纠偏的系数d等相关的一个系数矩阵.

我们从记录点出发，使用预测位置来判断是否需要纠偏，计算纠偏参数d，并将计算结果按矩阵乘法方式存入路径.同时，依照纠偏参数和上一轮的预测位置计算出新的预测位置.当预测位置的估计误差值到达一个合理的容忍上界后，再根据记录点和路径还原出当前的准确位置，纠正预测位置，最后初始化路径系数矩阵后进入新一轮计算.

预测位置的误差按照乘法速度扩散，因此其误差位数的增长速度和两个因素成线性关系，一个是$\log|d|$，另一个是距离上一个记录点的距离.由这两个参数的累加和便可以计算出估计误差，一般而言，实际误差小于估计误差.

通常设置容忍上界为$lowlevel$的一半以下，取三分之一最为稳妥.这是因为当误差小于高位比特长度的一半时，可以正确计算出纠偏参数d，当误差长度小于低位比特的长度时，是否纠偏可以正确判断.

我们可以看出，当控制住误差范围之后，我们只需要通过预测位置这些较小的整数运算，就能够得出纠偏参数，只有当误差变得足够大才对大整数进行运算.这种方式可以减少一些大整数的运算.

4.2 实际效果

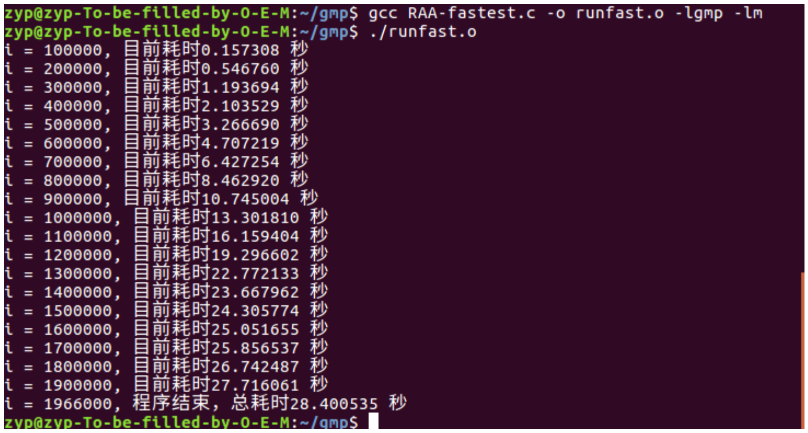

Figure 3: 优化后程序输出结果

在同一台电脑同一个环境下，我们分别使用高低位各一千位左右的比特进行估计运算，此时优化算法运行的时间开销为28秒左右，并且其时间开销曲线符合之前相同的结论.

最后输出的结果与优化前算法一致，说明了优化没有改变算法的有效性.

5 总结

本文实现了文献中提供的有理逼近算法，并成功求出了完整序列的最佳有理逼近.同时在原作者提供的优化思路上完成了算法的改进和进一步，由原算法的2500余秒提高至70余秒，进一步提高至28秒.同时，我们完成了改进后算法的复杂度证明，从理论上证明了新的算法远优异于旧有算法.

6 附录及相关说明

- $sequence.txt$:该文档为赛题一提供的原始序列.
- $RAA\_old.c$:这是未优化前的算法C语言源代码，编译前请安装好GMP大数库，运行程序时需要将$sequence.txt$文件放置在同一文件夹下.
- $RAA.c$:这是优化后的算法C语言源代码，已标明注释，编译前请安装好GMP大数库，运行程序时需要将$sequence.txt$文件放置在同一文件夹下.
- $RAA\_fastest.c$:这是初赛后进一步效率优化的算法C语言源代码，编译前请安装好GMP大数库，运行程序时需要将$sequence.txt$文件放置在同一文件夹下.
- $ans.txt$:这是$RAA.c$编译后程序的输出结果，其中第三行p/q为验证的结果，因二进制数表达顺序，其实际为$sequence.txt$的逆序排列（开头有一个零被忽略了）.
- $ans\_old.txt$:这是$RAA_old.c$编译后程序的输出结果，数据实际上与$ans.txt$一样.
- $ans\_ex.txt$:这是$RAA_fastest.c$编译后程序的输出结果，数据实际上与$ans.txt$一样.

References

[1] Klapper A, Goresky M. Feedback shift registers, 2-adic span, and combiners with memory[M]. Springer-Verlag New York, Inc. 1997.

[2] Schönhage A, Strassen V. Schnelle Multiplikation grosser Zahlen[J]. Computing, 1971, 7(3-4):281-292.

[3] https://gmplib.org/