Development of New Algorithm for Finding Inverse of Modular Multiplication

Dr. J. Thirumaran¹, S. Raja²

¹Dean, Rathinam College of Arts & Science, Coimbatore-642021, India
²Assistant Professor, Rathinam College of Arts & Science, Coimbatore-642021, India

Abstract: The output of division of two integers in most of the cases is not integer in traditional arithmetic. However, in modular arithmetic, \((c/d) \mod p\) is either integer if \(d\) and \(p\) are relatively prime. Basic Arrays and their Properties are analyzed first, The available algorithms are analyzed and MMI algorithm is proposed. The comparative Analysis of NEA vs. XEA are made and complexity Analysis of MMI Algorithm is also made.

Keywords: Modular multiplication, NMI, NEA, XEA, Algorithms

1. Introduction: Division of Two Integers

The output of division of two integers in most of the cases is not integer in traditional arithmetic. However, in modular arithmetic, \((c/d) \mod p\) is either integer if \(d\) and \(p\) are relatively prime, or it does not exist if \(d\) and \(p\) are not co-prime, i.e., if \(\gcd(d, p) > 1\). Analogous case exists in traditional arithmetic: For instance, if \(dy = 1\) and \(d = 0\), then there is no unique \(y\) that satisfies \(0y = 1\).

In this paper, we provide an Enhanced-Euclid algorithm (NEA) that finds for two relatively prime integers \(p_0\) and \(p_1\) an integer number \(x\), satisfying the equation \(p_0x \mod p_0 = 1\). This integer \(x\) is called a multiplicative inverse of \(p_1\) modulo \(p_0\) or, for short, a Modular Multiplicative Inverse (MMI). The Extended- Euclid algorithm (XEA) (Knuth, 1997) also finds a MMI of \(p_0\) and \(p_1\) if \(\gcd(p_0, p_1) = 1\). Otherwise, the XEA finds \(\gcd(p_0, p_1)\).

In this paper, we prove a validity of the NEA and provide its analysis. The analysis demonstrates that the NEA is faster than the XEA.

2. Basic Arrays and their Properties

Let us consider five finite integer arrays:

\[
\{p_i\}, \{c_i\}, \{t_k\}, \{w_k\}, \{z_k\}. \quad (2.1)
\]

**Definition 2.1.** Let \(\{p_i\}\) and \(\{c_i\}\) be integer arrays defined according to the following generating rules:

given two relatively prime integers \(p_0\) and \(p_1\) such that \(p_0 > p_1\),

for \(i \geq 1\) while \(p_i \geq 2\),
do \(p_{i+1} := p_i - 1\) mod \(p_1\) and \(c_i := x^{p_i-1} / p_i\). \( (2.2)\)

**Definition 2.2.** Let for every \(k \geq 1\) \(\{t_k\}\) be an arbitrary array; let \(\{w_k\}\) and \(\{z_k\}\) be defined by the following generating rules:

if \(w_0, w_1, z_0\) and \(z_1\) are initially specified,

then for every \(k \geq 2\),

\(w_k := w_{k-1} t_{k-1} + w_{k-2}\) and \(z_k := z_{k-1} t_{k-1} + z_{k-2}\). \( (2.3)\)

**Proposition 2.3.** Let us consider a sequence of determinants \(D_k := z_k z_{k-1}^{-1}\), then for every \(k \geq 1\),

\(D_k = (-1)^{k-1} D_1 \cdot (2.4)\)

Consider \(D_k\) and substitute in the left column the values of \(w_k\) and \(z_k\) defined in (2.3).

After simplifications, it follows that \(D_k = -D_{k-1}\), then this relation, if applied telescopically, implies (2.4).

**Proposition 2.4.** Let all three arrays \(\{t_k\}, \{w_k\}\) and \(\{z_k\}\) be integer, and \(w_0 := 1, z_0 := 0, |z_1| := 1\).

Proposition 2.3 implies that for every \(w_1 (-1)^{k-1} z_1 z_k\) is a multiplicative inverse of \(w_k - 1\) modulo \(w_k\). Indeed, since \(D_1 = -z_1\) then (2.4) implies that \(-w_k z_k \mod (1-1) z \mod (2.4)

**Proposition 2.5.** If for every \(0 \leq k \leq r\), \(t_k := cr-k\), then \(w_k := pr-k\), i.e., \(\{w_k\} = \{p_i\} R\) and \(\{t_k\} = \{c_i\} R\).

where the superscript \(R\) in (2.6) means that the arrays \(\{c_i\}\) and \(\{p_i\}\) are written in reverse.

Thus \(p_0\) and \(p_1\) are seeds that generate the arrays \(\{p_i\}, \{c_i\}\) where \(\{c_i\} = \{cr-k\}\) and \(\{w_k\} := \{pr-k\}\).

**Theorem 2.6.** For every \(k = 1, \ldots, r\), \((1-k) z_1 z_k\) is the multiplicative inverse of \(pr-k+1\) modulo \(pr-k\), i.e., if \((k\) is odd and \(z_1 = 1\) or \((k\) is even and \(z_1 = -1\)), then \(x := z_k\) else \(x := pr-k - z_k\); if \(k\) is odd and \(z_1 = -1\) then \(x := z_r\), i.e., \(p_1 z_1 = 1 \mod p_0\).

Proof follows from Propositions 2.3–2.5.

3. NEA for MMI

The proposed algorithm uses stack as a data structure. It solves Eq. (1.1).

vars: \(r, L, M, S, t\): all integer numbers, \(b\): boolean,

proc FORWARD:

assign \(L := p_0, M := p_1, b := 0\),

if \(r := 0\), height of the stack, \(r\) is used only for the analysis of the algorithm),

repeat \(t := xL/M \ast, S := L - M t, b := 1 - b, \{r := r + 1\}, (3.1)\)

push \(t\) (onto the top of the stack), \(L := M, M := S, (3.2)\) until \(S = 1\), (if \(S = 0\), then \(\gcd(p_0, p_1) = t\); no MMI)

consisting of first s + 1 terms of an array x0 , x1 ,...,
indeed, 1777 × 1822 mod 1973 = 1.

Table 3.1: NEA in progress

<table>
<thead>
<tr>
<th>( p1 = 1973 )</th>
<th>( p0 = 1777 )</th>
<th>( 196 )</th>
<th>( 11 )</th>
<th>( 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>151</td>
<td>136</td>
<td>15</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2: NEA algorithm with even number of columns

<table>
<thead>
<tr>
<th>2013</th>
<th>1976</th>
<th>37</th>
<th>15</th>
<th>7</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack</td>
<td>1</td>
<td>53</td>
<td>2</td>
<td>2</td>
<td>—</td>
</tr>
<tr>
<td>272</td>
<td>267</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Example 3.1. Let \( p0 \) and \( p1 \) be relatively prime integers; let us find an integer number \( x \) that is a MMI, i.e., satisfying the equation \( p0 \times x \mod p1 = 1 \) (1.1). Suppose that \( p0 = 1777 \) and \( p1 = 1973 \). Table 3.1 shows the algorithm in progress. Since the right-most element in the first row is equal one, hence the MMI exists. The second row stores the stack and the left-most element in the third row is equal to either \( x \), if the number of columns is even, or it is equal to \( p1 - x \) if the number of columns is odd. Thus, in this example \( x = 1973 - 151 = 1822 \). Indeed, 1777 × 1822 mod 1973 = 1.

Example 3.2. Let now \( p0 = 1976 \) and \( p1 = 2013 \), let us determine an integer \( x \) that satisfies Eq. (1.1). Table 3.2 shows the algorithm in progress. Since the number of columns is even, hence \( x = 272 \). Indeed, 1976 × 272 mod 2013 = 1.

Notice that the lengths of the stacks are very short in both examples: we need to store only three and four elements, respectively.

4. Complexity Analysis of MMI Algorithm

Let us consider four integer non-negative arrays: \{pi\} and \{ci\} as they defined in (2.2), and \{dk\} and \{qk\} defined in accordance with the rules:

\[
\begin{align*}
&dk = dt , \quad pi = ci , \\
&ci = dk , \quad qk = pi -1 \quad (4.1) \\
&pi + 1 = qk + 1 = qk + 1 \quad (4.2)
\end{align*}
\]

Here \{ci\} and \{dk\} are quotients; \{pi\} and \{qk\} are remainders. It is clear from (2.2) and (4.2) that \( pi + 1 = ci - 1 - pici = pi - 1 \mod pi \). Hence, both arrays \{pi\} and \{qk\} are strictly decreasing and all terms of the corresponding arrays \{ci\} and \{dk\} are positive integers.

Definition 4.1. \( \{xj \} \) is a \( (s + 1) \)-dimensional vector, consisting of first \( s + 1 \) terms of an array \( x0 , x1 ,..., xj , xj +1 , xj +2 \), i.e., \( \{xj \} = \{x0 , x1 ,..., xj \} \).

Theorem 4.2. Consider \( \{ci\} \geq 1 \), \( \{pi\} , \{dk\} , \{qk\} \geq 1 \) and \( \{pi\} \geq 1 \). Let \( p0 = q0 \), \( \{ci\} \leq \{dk\} \) i.e., for every \( j \geq 1 \), there is at least one \( j-1 \) such that \( ci < dl \); then for every \( 1 \leq j \leq s \) the following inequalities hold:

\[
\text{if } 1 \leq j \leq 1 - 1, \text{ then } pj \geq qj \text{ else } pj > qj \quad (4.3)
\]

Proof. Assuming that the statement (4.3) holds for every \( i \leq j-1 \), let us demonstrate by induction that it also holds for \( i = j \).

Consider \( ti = dj - cj = xqj - l/pj \), \( x \times pj - l/pj \). \( (4.4) \) If \( j \leq 1 - 1 \), then \( ti = 0 \) else \( ti > 0 \). Hence, \( (4.4) \) implies that if \( j \leq 1 - 1 \), then \( pj \geq qj \text{ else } pj > qj \). Since \( p0 = q0 \), therefore, (4.3) holds for every \( j \).

Consider a pair of relatively prime seeds \( p0 \) and \( p1 \) that generates an array \( \{ci\} \), \( \{ci\} \). Let us also consider another pair of relatively prime seeds \( p0 \) and \( q1 \) that generates an array \( \{dk\} \), \( \{dk\} \).

Corollary 4.3. A pair of seeds that is required for a given \( p0 \) is the maximum number of steps for computation of a MMI, generates an unary array of quotients, where every component in \( \{ci\} \) is 1. Thus, as it follows from (2.2) and (4.3), this pair of seeds must generate the following array of integer numbers: \( p2 := p0 - p1 \), \( p3 := p1 - p2 \),..., \( pr := pr -2 - pr -1 \). For instance, the array of the Fibonacci numbers \( \{Fr+2 , Fr+1 ,... , F4 , F3 , F2 \} \) generates the former array where for every \( i = 0,\... ,r \) \( pi := Fr+2 -i \).

Remark 4.4. The pair \( p0 = Fr+2 , p1 = Fr+1 \) is not the only one that generates (a) an unary array of quotients; (b) a decreasing integer array with the rth remainder equal to one.

The following pairs of seeds have the same property (for all non-negative integer numbers \( t \) and \( u \)):

1. \( p0 = Fr+2 + t*Fr , p1 = Fr+1 + t*Fr-1 , \) for \( t = 1, \{pi\} = \{L1, L2 ,..., Lr+1\} \) is a sequence of the Lucas numbers 1, 3, 4, 7, 11, 18,...
2. \( p0 = Fr+2 + (1 - t)*Fr-1 , t \geq 1 \), \( t \leq 1 \), \( p1 = Fr+1 + (1 - t)*Fr-2 \).
3. \( p0 = Fr+1 + t*Fr , t \geq 1 \), \( p1 = Fr + t*Fr-1 \).
4. \( p0 = (1 + t)*Fr+2 + t*Fr-2 + u*Fr , p1 = (1 + t)*Fr+1 + t*Fr-3 + u*Fr-1 \).

Here the Fibonacci numbers with zero and negative indices are computed \( m-1 \) with the formula: \( Fm = (-1)^m \). For all pairs, listed above, exactly \( r \) steps are required to find the MMI. However, all these pairs are special cases of a pair of seeds where \( p0 = bFr + Fr1 , p1 = bFr - Fr1 \). Then for all \( 0 \leq i \leq r \), \( pi = bFr+i - Fr+i-1 \), \( pr -1 = b \) and \( pr = 1 \). Let \( v = (1 - \sqrt{5})/2 \) and \( w = (1 + \sqrt{5})/2 \). Using a z-transform approach we deduce that for all \( 0 \leq k \leq r \) \( pr - k = (b - v)w^k + (w - b)v^k \sqrt{5} \) (4.5)

Therefore, for a large \( r \) \( pr - k \sqrt{5} < (b - v)w^k + (w - b)v^k \sqrt{5} \) \( \rightarrow 0 \), since \( |v| < 1 \) (4.6)

The relation (4.6) implies that for a large \( r \), \( p0 = [w(b - v)/5] + o(w) \). (4.7)

Let \( z := \max b \geq 2 \) \( r(p0 , b) \).

Then
6. Conclusion

If both seeds p0 and p1 are chosen randomly, then the probability that \( \gcd(p_0, p_1) = 1 \) is equal to \( 6/\pi^2 = 0.608 \) (Chesaro, 1881). Let us consider the following notations:

- \( \text{wxea} \) - Worst-case specific complexity (per step) of XEA;
- \( \text{wnea} \) - Worst-case specific complexity of NEA; \( \text{axea} \) - Average-case specific complexity of XEA; \( \text{anea} \) - Average-case specific complexity of NEA;

\[ \text{td}, \text{tm}, \text{ts}, \text{tst} \text{-time complexities of the operations of division, multiplication, addition, assignment and stack operations push and pop, respectively.} \]

Then, notice that

\[ \text{wxea} = \text{td} + 3\text{tm} + 3\text{ta} + 10\text{ts}, \quad (7.1) \]
\[ \text{wnea} = \text{td} + 2\text{tm} + 8\text{ts} + 2\text{tst}, \quad (7.2) \]
\[ \text{anea} = (\text{td} + 2\text{tm} + 8\text{ts} + 2\text{tst}) \times 6/\pi^2 \]
\[ + (\text{td} + \text{tm} + 8\text{ts} + 8\text{tst}) \times 1 - 6/\pi^2, \quad (7.3) \]
\[ \text{axea} = \text{wxea} \quad \text{and} \quad \text{tm} \approx \text{ts} \approx \text{tst}. \quad (7.4) \]

Therefore, the execution of the NEA requires significantly less time than the execution of the XEA.

References