

1 Chinese Remainder Theorem

Given the moduli set m_i for $i = 1, 2, \dots, n$ such that

$$\gcd(m_i, m_j) = 1 \text{ for } i \neq j ,$$

there exists a unique integer u in the range $[0, M - 1]$ where $M = m_1 m_2 \cdots m_n$ with the property

$$u = u_i \pmod{m_i}$$

for $i = 1, 2, \dots, n$.

2 Single-Radix Conversion Algorithm

Step 1. Compute $M = m_1 m_2 \cdots m_n$ and $m_1 m_2 \cdots m_{i-1} m_{i+1} \cdots m_n = \frac{M}{m_i}$ using multi-precision arithmetic.

Step 2. Compute the multiplicative inverses of $\frac{M}{m_i}$ modulo m_i for $1 \leq i \leq n$, i.e., compute the constants c_i such that

$$\frac{M}{m_i} \cdot c_i = 1 \pmod{m_i} \text{ for } 1 \leq i \leq n .$$

Step 3. Compute u by performing the sum

$$u = \frac{M}{m_1} c_1 u_1 + \frac{M}{m_2} c_2 u_2 + \cdots + \frac{M}{m_n} c_n u_n \pmod{M} ,$$

in multi-precision arithmetic.

Theorem 1 Given the moduli m_1, m_2, \dots, m_n and the remainders u_0, u_1, \dots, u_n such that $m_i \leq W$ for $0 \leq i \leq n$, the number u can be computed in $O(n^2)$ arithmetic steps with the single-radix conversion algorithm.

3 Mixed-Radix Conversion Algorithm

Step 1. Compute constants c_{ij} for $1 \leq i < j \leq n$ where

$$c_{ij} \cdot m_i = 1 \pmod{m_j} .$$

Step 2. Compute

$$\begin{aligned} v_1 &= u_1 \pmod{m_1} , \\ v_2 &= (u_2 - v_1) c_{12} \pmod{m_2} , \\ v_3 &= ((u_3 - v_1) c_{13} - v_2) c_{23} \pmod{m_3} , \\ &\vdots \\ v_n &= (\cdots ((u_n - v_1) c_{1n} - v_2) c_{2n} - \cdots - v_{n-1}) c_{n-1,n} \pmod{m_n} . \end{aligned}$$

Once the mixed-radix digits have been obtained, u is written in terms of these digits and the moduli as

$$u = v_1 + v_2 m_1 + v_3 m_1 m_2 + \cdots + v_n m_1 m_2 \cdots m_{n-1} .$$

Computation of u using the above formula also requires $O(n^2)$ arithmetic operations. We now define V_{ij} for $0 \leq i < j \leq n$ such that $V_{0i} = u_i$ for $1 \leq i \leq n$ and $V_{i-1,i} = v_i$ for $1 \leq i \leq n$. These V_{ij} for $0 \leq i < j \leq n$ are the temporary values of v_j resulting from the operations in Step 2 of the mixed-radix conversion algorithm. This way, we build a triangular table of values with diagonal entries $v_i = V_{i-1,i}$ for $0 \leq i \leq n$. The entries of this table are named *multiplied differences*. For $n = 4$, it can be given as follows:

$$\begin{aligned} V_{01} &= u_1 [m_1] \\ V_{02} &= u_2 [m_2] & V_{12} &= (V_{02} - V_{01})c_{12} [m_2] \\ V_{03} &= u_3 [m_3] & V_{13} &= (V_{03} - V_{01})c_{13} [m_3] & V_{23} &= (V_{13} - V_{12})c_{23} [m_3] \\ V_{04} &= u_4 [m_4] & V_{14} &= (V_{04} - V_{01})c_{14} [m_4] & V_{24} &= (V_{14} - V_{12})c_{24} [m_4] & V_{34} &= (V_{24} - V_{23})c_{34} [m_4] \end{aligned}$$

Here $[m_i]$ stands for modulo m_i . The mixed-radix conversion algorithm computes the terms $V_{i,j}$ for $1 \leq i < j \leq n$ by performing the following operations on single-precision integer operands:

$$\begin{aligned} c_{ij} &= \text{INVERSE}(m_i, m_j) , \\ V_{ij} &= (V_{i-1,j} - V_{i-1,i})c_{ij} \pmod{m_j} . \end{aligned}$$

Theorem 2 *Given the moduli m_1, m_2, \dots, m_n and the remainders u_1, u_2, \dots, u_n such that $m_i \leq W$ for $1 \leq i \leq n$, the mixed-radix number representation (v_1, v_2, \dots, v_n) of u can be computed in $O(n^2)$ arithmetic steps with the mixed-radix conversion algorithm.*

We note that the above theorems are true for the *preconditioned* Chinese remaindering as well. In this case the constants c_i and c_{ij} are precomputed for the single-radix and the mixed-radix conversion algorithms, respectively.

4 Computation of the Inverse

The inverse $x = a^{-1} \pmod{m}$ is computed with Euclid's extended algorithm (EEA).

Input: $a, m \in D$, not both zero, D is an Euclidean domain.

Output: g, s, t such that $g = s \cdot a + t \cdot m$ and $g = \gcd(a, m)$.

```

procedure EEA( $a, m, g, s, t$ )
begin
  ( $g_0, g_1$ ) = ( $a, m$ )
  ( $s_0, s_1$ ) = ( $1, 0$ )
  ( $t_0, t_1$ ) = ( $0, 1$ )
  while  $g_1 \neq 0$  do
    begin
       $q = g_0 \text{ div } g_1$ 
      ( $g_0, g_1$ ) = ( $g_1, g_0 - g_1 \cdot q$ )
      ( $s_0, s_1$ ) = ( $s_1, s_0 - s_1 \cdot q$ )
      ( $t_0, t_1$ ) = ( $t_1, t_0 - t_1 \cdot q$ )
    end
  end

```

```

    end
    g = g0 ; s = s0 ; t = t0
end procedure

```

The following procedure uses EEA to compute the inverse.

Input: $a, m \in D$.

Output: If $\gcd(a, m) = 1$ then $x = a^{-1} \pmod{m}$.

```

procedure INVERSE(a, m, x)
begin
  EEA(a, m, g, s, t)
  if g = 1 then x = s
    else PRINT('inverse does not exist')
  end procedure

```

An Example

We execute $\text{INVERSE}(a, m, x)$ for $a = 16$ and $m = 21$ in order to compute $16^{-1} \pmod{21}$, i.e., to solve for x in

$$16 \cdot x = 1 \pmod{21} .$$

Procedure EEA computes the following tableau

iteration	g	g_0	g_1	s_0	s_1	t_0	t_1
0	-	16	21	1	0	0	1
1	0	21	16	0	1	1	0
2	1	16	5	1	-1	0	1
3	3	5	1	-1	4	1	-3
4	5	1	0	4	-21	-3	16

EEA thus returns $g = 1$, $s = 4$, and $t = -3$, with the property that

$$g = s \cdot a + t \cdot m .$$

Thus, we have

$$1 = 4 \cdot 16 + (-3) \cdot 21 .$$

Thus, INVERSE computes

$$x = 16^{-1} = s = 4 \pmod{21} .$$

Verify:

$$16 \cdot 4 = 64 = 1 \pmod{21} .$$