6 The Chinese Remainder Algorithm

Let R be a Euclidean Domain and $m_0, m_1, m_2, \ldots, m_{r-1} \in \mathsf{R}$ with $\gcd(m_i, m_j) = 1$ for $i \neq j$. Then let $m = m_0 m_1 \cdots m_{r-1}$.

Fact 6.1 (The Chinese Remainder Theorem)**.**

$$\frac{\mathsf{R}}{(m)} \cong \frac{\mathsf{R}}{(m_0)} \times \frac{\mathsf{R}}{(m_1)} \times \cdots \times \frac{\mathsf{R}}{(m_{r-1})}$$

Example 6.2. *For* $\mathsf{R} = \mathbb{Z}$*, suppose* $m_0 = 7$*,* $m_1 = 11$ *and* $m_2 = 13$*, so* $m = 1001$*, and*

$$\frac{\mathbb{Z}}{(1001)} \cong \frac{\mathbb{Z}}{(7)} \times \frac{\mathbb{Z}}{(11)} \times \frac{\mathbb{Z}}{(13)}.$$

Consider the representation of $a = 233 \bmod m$*.*

$$233 \mapsto (2, 2, 12)$$

If we had $b = 365 \bmod m$*, then* $b \mapsto (1, 2, 1)$*. If we want to compute* $a + b \bmod m$*, we could compute*

$$(2,2,12) + (1,2,1) = (3,4,0) \ \mapsto\ 598 \bmod 1001.$$

Similarly $a \cdot b$ *can be computed by component-wise product:*

$$(2,2,12) * (1,2,1) = (2,4,12) \ \mapsto\ 961 \bmod 1001.$$

What about $1234 \bmod m$*?*

$$1234 \mapsto (2,2,12)$$

The mapping is only defined modulo m, so 233 *and* 1234 *have the same representation. If we know that* $a \in \mathbb{Z}$ *is between in* $\{0, \ldots, m-1\}$ *then we, then we recover it uniquely from its image in* $\mathbb{Z}_7 \times \mathbb{Z}_{11} \times \mathbb{Z}_{13}$*. This is the basis of many so-called "modular" algorithms.*

The fact that the Chinese Remainder Theorem provides an isomorphism means that for any sequence of residues like $(2,2,12)$ *there exists a unique element in* $\mathbb{Z}_m$*. How do we find this?*

The isomorphism given by the Chinese remainder theorem can be implemented by efficient algorithms in both directions.

One direction is "easy": given a (and $m_0, \ldots, m_{r-1}$), compute

$$a \mapsto (a \operatorname{rem} m_0, a \operatorname{rem} m_1, \ldots, a \operatorname{rem} m_{r-1}).$$

This maps a to a "small" residue in each of $\mathbb{Z}_{m_0}, \ldots, \mathbb{Z}_{m_{r-1}}$. We saw that in the case $\mathsf{R} = \mathbb{Z}$ that this was particularly efficient, at least in the naive cost model: when $0 \le a < m$ we could compute this with $O((\log m)^2)$ word operations.

We now consider the other direction: Given $v_0, v_1, \ldots, v_{r-1} \in \mathsf{R}$, find a such that

$$a \equiv v_0 \bmod m_0, \ \ a \equiv v_1 \bmod m_1, \ \ \ldots, \ a \equiv v_{r-1} \bmod m_{r-1}.$$

The existence of a is guaranteed by the Chinese Remainder Theorem.

We do something very similar to Lagrange interpolation. Find $L_0,\dots,L_{r-1}$ such that $L_i \equiv 0 \bmod m_j$ for $i \neq j$ and $L_i \equiv 1 \bmod m_i$. Then

$$a = v_0 L_0 + v_1 L_1 + \cdots v_{r-1} L_{r-1} \in \mathsf{R},$$

which has the desired properties. But how do we find $L_0,\dots,L_{r-1}$?

We assume that $\gcd(m_i,m_j) = 1$ for $i \neq j$ (we say the $m_0,\dots,m_{r-1}$ are pairwise relatively prime). This implies $\gcd(m/m_i,m_i) = 1$ for $0 \le i < r$. Thus, by the extended Euclidean algorithm, there exists s_i,t_i such that $s_i \cdot m/m_i + t_i m_i = 1$. In other words, $L_i = s_i \cdot (m/m_i) \equiv 1 \bmod m_i$. Also, since $m/m_i = m_0 m_1 \cdots m_{i-1} m_{i+1} \cdots m_{r-1}$, we know that $m/m_i \equiv 0 \bmod m_j$ for $i \neq j$, so $L_i \equiv 0 \bmod m_j$ for $i \neq i$.

Example 6.3. *Again with* $\mathsf{R} = \mathbb{Z}$ *and* $m_0 = 7$*,* $m_1 = 11$ *and* $m_2 = 13$*, suppose* $v_0 = 2$*,* $v_1 = 2$ *and* $v_2 = 12$*. Then*

$$\begin{array}{ll}
\gcd(11\cdot 13, 7) = 1 = -2\cdot(11\cdot 13) + 41\cdot 7 & \Longrightarrow L_0 = -2\cdot 11\cdot 13 = -286 \\
\gcd(7\cdot 13, 11) = 1 = 4\cdot(7\cdot 13) - 33\cdot 11 & \Longrightarrow L_1 = 4\cdot 7\cdot 13 = 364 \\
\gcd(7\cdot 11, 13) = 1 = -1\cdot 7\cdot 11 + 41\cdot 6\cdot 13 & \Longrightarrow L_2 = -1\cdot 7\cdot 11 = -77
\end{array}$$

Thus

$$a \equiv v_0 L_0 + v_1 L_1 + v_2 L_2 \equiv 2\cdot(-286) + 2\cdot 364 + 12\cdot(-77) \equiv -768 \equiv 233 \bmod 1001.$$

Now consider the cost of computing a. First, we need to compute $m = m_0 m_1 \cdots m_{r-1}$ in case this is not given as part of the input. Assuming that each $m_i \geq 2$ so that we can make the simplification $\lg m_i \le 1 + \log_2 m_i \le 2 \log m_i$, we simply compute $m_0 m_1$, $(m_0 m_1) m_2, \dots$ in succession for an overall cost of

$$\begin{array}{rcl}
c \sum_{i=1}^{r-1} (\lg m_0 m_1 \dots m_{i-1})(\lg m_i) & \le & c(\lg m) \sum_{i=1}^{r-1} (\lg m_i) \\
& \le & c2(\log_2 m) \sum_{i=1}^{r-1} 2(\log_2 m_i) \\
& < & c2(\log_2 m) \sum_{i=0}^{r-1} 2(\log_2 m_i) \\
& = & c4(\log_2 m)(\log_2 m)
\end{array}$$

word operations, for some constant c. The cost of computing m/m_i for $0 \le i \le r-1$ is bounded by $c\sum_{i=0}^{r-1} (\lg m/m_i)(\lg m_i) \le c(\lg m)\sum_{i=0}^{r-1}(\lg m_i)$, which can be simplified to $O((\log m)^2)$ word operations. Next we compute $\gcd(m/m_i, m_i) = s_i(m/m_i) + t_i m_i$ at a cost of $O((\lg m/m_i)(\lg m_i))$ for $0 \le i \le r-1$. This again simplifies to $O((\log m)^2)$ word operations. Finally, computing the products $L_i = s_i(m/m_i)$ and $v_i L_i$ can be shown to have cost $O((\log m)^2)$, using the fact that $|s_i| \le m_i$ and $\lg m/m_i < \lg m$.

In summary, both directions of the Chinese Remainder theorem can be computed with $O((\log m)^2)$ word operations.

6.1 Variants of Chinese Remaindering

There are a number of useful variants of the Chinese remainder theorem and algorithm. First we consider the mixed radix representation.

Mixed radix representation

Suppose $0 \le a \le M = m_0 m_1 \cdots m_r$, where $m_i \in \mathbb{Z}$ are all at least 2 (and are not necessarily relatively prime).

Claim: We can write a uniquely as

$$a = a_0 + a_1 m_0 + a_2 m_0 m_1 + \cdots + a_r m_0 ... m_{r-1}$$

with $0 \le a_i < m_i$ for all i. This is called the *mixed radix representation* of a.

For example, if $m_0 = 7$, $m_1 = 11$, $m_2 = 13$, then

$$233 = 2 + (0)(7) + (3)(7 \times 11)$$

We should prove that such a representation always exists. We use weak induction on r.

If $r = 0$ then $a = a_0$, which covers the base case.

Assume the inductive hypothesis that there is a mixed modulus representation of a for $m_0, \ldots, m_{r-1}$. Now show it for $m_0, \ldots, m_r$.

Let $\check{a} = a \text{ rem } m_0 m_1 \cdots m_{r-1}$. Then by induction we know we can write

$$\check{a} = a_0 + a_1 m_0 + a_2 m_0 m_1 + \cdots a_{r-1} m_0 \cdots m_{r-2}.$$

Define $a_r = (a - \check{a})/(m_0 m_1 ... m_{r-1})$. Then

$$a = a_0 + a_1 m_0 + \cdots + a_{r-1} m_0 m_1 ... m_{r-2} + a_r m_0 m_1 \cdots m_{r-1}.$$

We know a_r is unique since the other quantities are unique, and $0 \le a_r < m_r$ follows from the fact that $0 \le a < m_0 m_1 ... m_{r-1}$.

Incremental Chinese Remaindering

Incremental Chinese remaindering computes $\mathrm{rem}(a, m_0), \mathrm{rem}(a, m_0 m_1), \mathrm{rem}(a, m_0 m_1 m_2), \ldots$. More precisely, given two relatively prime moduli $M, m \in \mathbb{Z}_{>1}$, and two images $V, v \in \mathbb{Z}$ such that $0 \le V < M$ and $0 \le v < m$, our goal is to reconstruct an $a \in \mathbb{Z}$ such that $a \equiv V \bmod M$ and $a \equiv v \bmod m$. Here we think of M as big (for example, $M = m_0 m_1 \cdots m_{r-1}$) and m_r as small (for example, $m = m_r$). The obvious way to do this is to use the EEA to compute $s, t \in \mathbb{Z}$ such that $sM + tm = 1$, and then set $a = tVm + svM$. In assignment 2 you are asked to analyze this method, and then derive a better method based on the mixed-radix representation of a.

Incremental Chinese remaindering can be used for so-called "output sensitive" algorithms. Sometimes we don't know how big the (integer) output is in advance. Therefore we compute the result modulo more and more primes. When we recover the same a modulo a few prime, we "guess" that we have the correct integer result. For some problems it is possible to prove that the output is correct if the result does not change for a few primes. Often we just prove this is true with high probability for randomly chosen primes.