Modular (Remainder) Arithmetic

$n = qk + r$ (for some k; $r < k$) $\Rightarrow$ $n \bmod k = r$

eg $37 = (2)(17) + 3$ $\Rightarrow$ $37 \bmod 17 = 3$

Divisibility notation: $17 \mid 37 - 3$

Sets of Remainders

x	x mod 5
-2 (5 – 2)	3
-1 (5 – 1) *	4
0	0
1	1
2	2
3	3
4	4
5	0
6	1
7	2
8	3

* Compilers may not handle this…

Congruences

mod 5:

$0 = 5 = 10 \Rightarrow (0 \bmod 5) = (5 \bmod 5) = (10 \bmod 5)$

$1 = 6 = 11\ldots$

$2 = 7 = 12\ldots$

$3 = 8 = 13\ldots$

$4 = 9 = 14\ (= -1 = -6)$

Operations

$$(x + y) \bmod n = (x \bmod n + y \bmod n) \bmod n$$
$$(x - y) \bmod n = (x \bmod n - y \bmod n) \bmod n$$

$$xy \bmod n = (x \bmod n)(y \bmod n) \bmod n$$

$$(a + k_1 n)(b + nk_2 n) = ab + n(bk_1 + ak_2 + n\, k_1\, k_2)$$

‘Shortcuts’ to prevent overflow

Example: last digit of n^{th} Fibonacci number

Example: 373! mod 997

= ((((1 mod 997)*2 mod 997)*3 mod 997)*4 mod 997)…

‘Division’

$ax = b \pmod{m}$

Eg: $5x = 7 \pmod{11}$

Solution: $5(8) = 40 = (11)(3) + 7$

Euclidean Algorithm

gcd(43, 29) = gcd(43 mod 29, 29)

43
29
14
1 ➡ relatively prime

Bezout's Identity:

gcd(a, b) = au + bv

➡ 1 = au + bv (if a, b relatively prime)

➡ ua = 1 - bv = 1 (mod b)

$u = a^{-1} \pmod{b}$

Extended Euclidean Algorithm

		43:	29:
q	r	u	v
-	43	1	0
-	29	0	1
1	14	1	-1
2	1	-2	3
14	0	-	-

$1 = -2(43) + 3(29)$ ➡ $(3)(29) = 1 \pmod{43}$

$a(29) = 5 \pmod{43}$

$(5)(29^{-1}) = 5*3 \bmod 43 = 15$

$(15)(29) \bmod 43 = 435 \bmod 43 = 5$

Non prime cases:

ax = b (mod m)   gcd(a, m) = d ? 1

if d | b problem has multiple solutions

Eg: 2x = 3 (mod 10)

gcd(2, 10) = 2

but 3 is not divisible by 2

no solution

Eg: 2x = 4 (mod 10)   divide through by gcd

x = 2 (mod 5)

$\Rightarrow$ x = 2 or 7 (mod 10)   (add multiples of 5)

Chinese Remainder Theorem

Given $x = a_k \pmod{m_k{*}}$
for $k = 1, 2, \ldots$

eg:
$x = 1 \bmod 2$
$x = 2 \bmod 3$
$x = 3 \bmod 5$

$N = ?$ $m_k = 2*3*5 = 30$
$n_k = N / m_k$

eg:
$n_1 = 30 / 2 = 15$ etc…

$y_k = n_k^{-1} \pmod{m_k}$

eg:
$y_1 = 15^{-1} \pmod{2}$
$= 1^{-1} \pmod{2} = 1$

$x = (a_1 n_1 y_1 + a_2 n_2 y_2 + \ldots) \bmod N$

eg $x = 23 \bmod 30$

*all relatively prime

Not relatively prime

eg:

x = 3 mod 6

x = 7 mod 10

gcd(6, 10) = 2

Split:

x = 1 mod 2 ←	don’t contradict →	x = 1 mod 2
x = 0 mod 3		x = 2 mod 5

thus recombine to give x = 27 mod 30

Matrices:

(solving linear equations with detached coefficients)

$$x + y + z = 5$$
$$x - y + 2z = 3$$
$$x + y - 3z = 0$$

$$\begin{pmatrix} 1 & 1 & 1 & : & 5 \\ 0 & -2 & 1 & : & -2 \\ 0 & 0 & -4 & : & -5 \end{pmatrix}$$

$$\begin{pmatrix} 1^{*} & 1 & 1 & : & 5 \\ 1 & -1 & 2 & : & 3 \\ 1 & 1 & -3 & : & 0 \end{pmatrix}$$

* pivot

Remainder Matrices

$$x + y + z = 5$$
$$x - y + 2z = 3$$
$$x + y - 3z = 0$$

taking mod 3:

$$\left(\begin{array}{ccccc} 1 & 1 & 1 & : & 2 \\ 1 & 2 & 2 & : & 0 \\ 1 & 1 & 0 & : & 0 \end{array}\right) \rightarrow \left(\begin{array}{ccccc} 1 & 1 & 1 & : & 2 \\ 0 & 1 & 1 & : & 1 \\ 0 & 0 & 2 & : & 1 \end{array}\right)$$

$2z = 1 \pmod 3$

$z = 2$

$y = 2$

$x = 1$

The easiest case: the prime case

Mod a prime, every number except 0 has an inverse

Thus, we can multiply a row by the inverse of the pivot

Non-prime mods

Use fundamental theorem of arithmetic and Chinese Remainder Theorem

Eg mod 6: (6 = 2*3)

$$\begin{pmatrix} 2 & 1 & : & 4 \\ 3 & 2 & : & 1 \end{pmatrix}$$

mod 3:

$$\begin{pmatrix} 2 & 1 & : & 1 \\ 0 & 2 & : & 1 \end{pmatrix}$$

$x_1 = 1, x_2 = 2$

mod 2:

$$\begin{pmatrix} 0 & 1 & : & 0 \\ 1 & 0 & : & 1 \end{pmatrix}$$

$x_1 = 1, x_2 = 0$

$x_1 = 1, x_2 = 2$

The prime power case

2 and 3 were relatively prime, but what if you were working mod 12? $12 = 3*2^2$.

Can't use Chinese remainder theorem since 2 and 2 are not relatively prime. Instead, work mod 2^2 and find pivot n such that:

$$n \bmod 2^k ? \ 0$$

for smallest possible k.

For example, if working mod 32 and the possible pivots are 12, 8 and 16, pivot around the 12 since 12 mod 8 ? 0.

In this case you cannot find 12x = 1 (mod 32), instead solve for 12x = 4 (mod 32). This can be done via extended Euclid since gcd(12, 32) = 4 and yields 12*3 = 4 (mod 32). Thus multiply the pivot row by 3.

Example

Mod 32:

$$\begin{pmatrix} 8 & 5 & 3 & : & 27 \\ 12 & 3 & 5 & : & 1 \\ 16 & 2 & 1 & : & 23 \end{pmatrix}$$

Move first row to top (current pivot row) and multiply by 3:

$$\begin{pmatrix} 4 & 9 & 15 & : & 3 \\ 8 & 5 & 3 & : & 27 \\ 16 & 2 & 1 & : & 23 \end{pmatrix}$$

$$\begin{pmatrix} 4 & 9 & 15 & : & 3 \\ 0 & 19 & 5 & : & 21 \\ 0 & 30 & 5 & : & 11 \end{pmatrix}$$

Now 19 mod 2 = 1 thus can find 19^{-1} mod 32 = 27

$$\begin{pmatrix} 4 & 9 & 15 & : & 3 \\ 0 & 1 & 7 & : & 23 \\ 0 & 0 & 19 & : & 25 \end{pmatrix}$$

Which yields $x_1 = 1$, $x_2 = 2$, $x_3 = 3$

Special case: mod 2

mod 2:

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 0$

equivalent to binary xor

So…

Unlike simultaneous ‘or’ equations which are NP-Complete, simultaneous ‘xor’ equations are solvable in polynomial (generally cubic) time via Gaussian elimination.

CEOI Example: X-Planet

X-Planet: given a set of lightbulbs, all initially off, and switches which each toggle the state of a given set of lightbulbs, determine any combination of switches which will turn on all the lightbulbs.

Equations: each lightbulb's state is affected by certain switches and in total an odd amount of these must be pressed.

IOI Example: Clocks

Given: a set of clocks in different positions, and controls which rotate a subset of clocks, determine the shortest sequence of moves to rotate all clocks to 12:00

• The original question limites the number of clocks to 9 and the possible positions on each clock to 4, so this is solvable (more easily) by brute force.

• Mod equations where you have more variables than equations will usually give multiple possible solutions. To determine an optimal solution you would still have to search within these; however the search would be reduced.

0011010101101010

Binary Manipulation

<table>
<tr><th>English</th><th>Sets</th><th>Pascal</th><th>C</th></tr>
<tr><td>and (1)</td><td>intersection</td><td>and</td><td>&</td></tr>
<tr><td>or</td><td>union</td><td>or</td><td>|</td></tr>
<tr><td>toggle/xor (2)</td><td>union\intersection</td><td>xor</td><td>^</td></tr>
<tr><td>left shift</td><td>-</td><td>shl</td><td><<</td></tr>
<tr><td>right shift (3)</td><td>-</td><td>shr</td><td>>></td></tr>
</table>

(1) can be equivalent to mod by powers of 2
(2) equivalent to adding bits mod 2
(3) equivalent to multiplying and (integer) dividing by powers of 2

Advantages

Depending on machine word size, these operations can work on 32 bits at once.

They are all small operations.

Eg: when working mod 2 with 31 or fewer variables, store as an integer. To add two rows, just xor them. To determine which row next to use as a pivot, just sort in descending order.

Binary Euclidean Algorithm

(1) If M, N even:

$$\gcd(M, N) = 2*\gcd(M/2, N/2)$$

(2) If M even while N is odd:

$$\gcd(M, N) = \gcd(M/2, N)$$

(3) If M, N odd:

$$\gcd(M, N) = \gcd(\min(M, N), |M - N|)$$

(replace larger with (larger – smaller); this will then be even and (1) can be applied.)

Binary Euclidean Algorithm

Disadvantage

In general, requires a few more steps

Advantage

Requires only binary shifts, binary ands, subtractions and if statements.

These operations are *much* faster than divisions and mods.

Extended Binary Euclid?

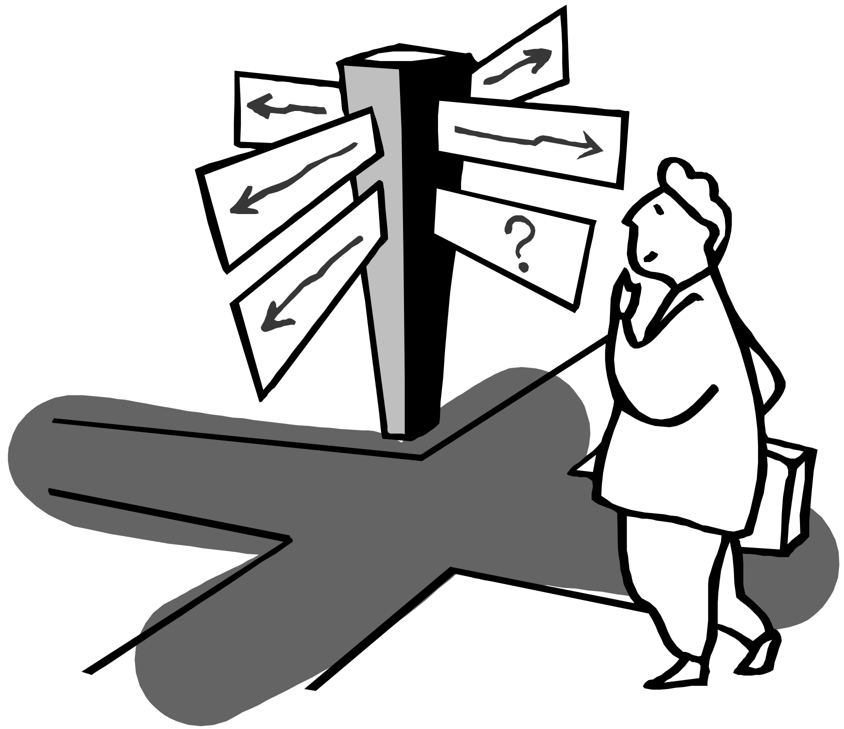
Possible, eg:

During halving stage:

gcd(7, 8) = gcd(7, 4)
Given 1 = -1*7 + 2*4, ie –1*7 = 1 (mod 4)
We can deduce –1*7 or (-1 + 4)*7 = 1 (mod 8)

gcd(7, 4) = gcd(7, 2)
Given 1 = 1*7 + -3*2, ie –3*2 = 1 (mod 7)
We can deduce $–3*4*2^{-1} = 1$ (mod 7)
2^{-1} can always be found quickly

Possible, but complicated



Summary

Type of problems to tackle with simultaneous mod equations:
Toggle/cyclic states affected linearly by different sets of stimuli, eg bulbs and button presses, eg 7th Guest puzzle.

Use mod theory with:
Anything numeric that could be thought of in terms of remainders.

Use binary operations:
When dealing with sets which can be stored in full and whose intersections/unions etc must be calculated quickly
For the binary Euclidean algorithm.