Modular Arithmetic Presentation Summary – Richard Starfield

Extended Euclidean Algorithm

Step No.	q	r	u (coeff. of 43)	v (coeff. Of 29)
-	-	43	1	0
-	-	29	0	1
1	1	14	1	-1
2	2	1	-2	3
3	14	0	-	-

For each step k, $q_k = r_{k-2}$ div r_{k-1}
Then
$r_k = r_{k-2} - q_k r_{k-1}$
$u_k = u_{k-2} - q_k u_{k-1}$
$v_k = v_{k-2} - q_k v_{k-1}$

Last nonzero r is the gcd. Obviously, when implementing the algorithm the entire table doesn't have to be stored.

Chinese Remainder Theorem

Given $x = a_k \pmod{m_k}$
for k = 1, 2, …; and all mods are relatively prime
$N = \bullet\ m_k = 2*3*5 = 30$
$n_k = N / m_k$
$y_k = n_k^{-1} \pmod{m_k}$
$x = (a_1 n_1 y_1 + a_2 n_2 y_2 + \ldots) \bmod N$

Under any m_k, the k^{th} term evaluates to a_k while the other terms evaluate to 0.
If the m_k's are not relatively prime, find the gcd and split each equation into components. Eg: 6 and 10 have gcd 2, so split 6 into 2 and 3, 10 into 2 and 5. If the two mod 2 equations contradict one another, there is no solution. Otherwise recombine the mod 2, mod 3 and mod 5 equations using the Chinese Remainder Theorem as above.

Example:

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

$n_1 = 30 / 2 = 15$; $n_2 = 10$; $n_3 = 6$
$y_1 = 15^{-1} \pmod 2 = 1^{-1} \pmod 2 = 1$; etc…
=> $x = 23 \bmod 30$

Simultaneous Linear Mod Equations

1) Prime mod:
Every number except 0 has an inverse, so multiply pivot row by inverse of pivot.

2) Compound mod:
Split into relatively prime components, solve separately and recombine using the Chinese Remainder theorem.

3) Prime power mod:
Find the smallest power of the prime for which there is a pivot, which is not divisible by this power of the prime. Use extended Euclid to calculate 'inverse' for the pivot with regard to this power. I.e. instead of solving ax = 1 (mod p) solve ax = 9 (mod 27). Then multiply the pivot row by this inverse (which will be relatively prime regarding the mod).

Binary Manipulation

English	Sets	Pascal	C
And (1)	Intersection	And	&
Or	Union	Or	|
Toggle/xor (2)	Union\intersection	Xor	^
Left shift (3)	-	Shl	<<
Right shift (3)	-	shr	>>

(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

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.)

References: (i.e. useful sites!)

http://wikibooks.org/wiki/Discrete_mathematics:number_theory
http://www.cut-the-knot.org/blue/Modulo.shtml
http://www.campusprogram.com/reference/en/wikipedia/m/mo/modular_arithmetic.html

Thanks to Bruce Merry for some very useful advice regarding mod Gaussian elimination.