

CSCI 124/297

Discrete Structures II: Two Algorithms in Modular Arithmetic

Poorvi L. Vora

November 30, 2006

1 Generalized Euclidean Algorithm to Determine Inverse in Z_m

Recall the euclidean algorithm. For example, recall its use to determine $\gcd(79, 551)$.

$$\begin{aligned}(a, b) &= (551, 79) \\(a, b) &= (79, 77) \\(a, b) &= (77, 2) \\(a, b) &= (2, 1) \\(a, b) &= (1, 0) \\ &\text{return}(1)\end{aligned}$$

As the gcd of 79 and 551 is 1, 79 is invertible *modulo* 551. In fact, the euclidean algorithm can be reversed as follows to determine the inverse, by keeping a record of the quotient.

| i | a | b | q_i |
|-----|-----|-----|-------|
| 0 | 551 | 79 | 6 |
| 1 | 79 | 77 | 1 |
| 2 | 77 | 2 | 38 |
| 3 | 2 | 1 | 2 |
| 4 | 1 | 0 | |

Now reverse direction and compute an extra pair, $(s, t) := (t, s - t * q_i)$.

| i | a | b | q_i | s | t |
|-----|-----|-----|-------|-----|--------------------|
| 0 | 551 | 79 | 6 | 39 | $-38-(39)6 = -272$ |
| 1 | 79 | 77 | 1 | -38 | $1 - (-38)(1)=39$ |
| 2 | 77 | 2 | 38 | 1 | $0-(1)(38)=-38$ |
| 3 | 2 | 1 | 2 | 0 | 1 |
| 4 | 1 | 0 | | | |

What you've found is the numbers s and t such that $sa + tb = 1$. Thus $39 \times 551 + (-272) \times 79 = 1$, and $(-272) \times 79 = 1 \pmod{551}$, or $279 \times 79 = 1 \pmod{551}$, and $79^{-1} \pmod{551} = 279$.

```

gcd_and_inverse(m, n) /* m > n */
(a, b) := (m, n) /* Initialize */
i = 0; /* Keep track of all quotients */
while (b ≠ 0)
{
  qi := ⌊ $\frac{a}{b}$ ⌋ /* Remember  $i^{th}$  quotient */
  (a, b) := (b, a mod b) /* Reduce problem */
  i := i+1 /* Increment count */
}
i := i-1; /* Now i reflects the number of q's because the last value of i does not correspond to a q */
/* Now go back up loop to determine inverse */
(s, t) := (0, 1) /* Initialize */
i := i-1 /* Update i */
while (i > 0)
{
  (s, t) := (t, s - t * qi)
  i := i-1
}
return(a, (s,t))

```

Example: Use the generalized euclidean algorithm to determine the inverse of $28 \bmod 75$.

| i | a | b | q_i |
|-----|-----|-----|-------|
| 0 | 75 | 28 | 2 |
| 1 | 28 | 19 | 1 |
| 2 | 19 | 9 | 2 |
| 3 | 9 | 1 | 9 |
| 4 | 1 | 0 | |

Now reverse direction.

| i | a | b | q_i | s | t |
|-----|-----|-----|-------|-----|-------------------|
| 0 | 75 | 28 | 2 | 3 | $-2-(3)(2) = -8$ |
| 1 | 28 | 19 | 1 | -2 | $1 - (-2)(1) = 3$ |
| 2 | 19 | 9 | 2 | 1 | $0-(1)(2) = -2$ |
| 3 | 9 | 1 | 9 | 0 | 1 |
| 4 | 1 | 0 | | | |

Thus $3 \times 75 + (-8) \times 28 = 1$, and $(-8) \times 28 = 1 \bmod 75$, or $67 \times 28 = 1 \bmod 75$, and $28^{-1} \bmod 75 = 67$.

2 Fast Modular Exponentiation

Consider how one might compute $x^k \bmod m$ for large values of x , k and m . If one actually performed $k - 1$ multiplications: $x \times x \times \dots \times x$, that could be a large number of multiplications. On the other hand, one might use the square-and-multiply rule which is considerably more efficient.

Let's consider an example with small enough numbers to compute results by hand: $5^{13} \bmod 17$.

$$5^{13} \bmod 17 = ((5^2)^2)^2 \times (5^2)^2 \times 5$$

which involves only 5 multiplications instead of 12. Or another example:

$$5^{15} \bmod 17 = ((5^2)^2)^2 \times (5^2)^2 \times 5^2 \times 5$$

which involves 6 multiplications.

This can be formalized as follows:

exponentiation(x, b, m) / x^b mod m */*

z := 1

/ b_i = ith bit in binary representation of b */*

for i = l - 1 downto 0

z := z² mod n

if b_i = 1 z := z × x mod n

endfor

return(z)

Example: $5^{51} \bmod 7$.

$51 = 110011$

| i | b_i | z_{first} | z_{second} |
|-----|-------|------------------|------------------|
| 5 | 1 | 1 | 5 |
| 4 | 1 | $25 \bmod 7 = 4$ | $20 \bmod 7 = 6$ |
| 3 | 0 | 1 | 1 |
| 2 | 0 | 1 | 1 |
| 1 | 1 | 1 | 5 |
| 0 | 1 | 4 | 6 |

Example: $7^{29} \bmod 11$

$29 = 11101$

| i | b_i | z_{first} | z_{second} |
|-----|-------|-------------|--------------|
| 4 | 1 | 1 | 7 |
| 3 | 1 | 5 | 2 |
| 2 | 1 | 4 | 6 |
| 1 | 0 | 3 | 3 |
| 0 | 1 | 9 | 8 |