Inverse and Modular Exponentiation

Sachin Tripathi

IIT(ISM),
Dhanbad

Euclid Algorithm

```
EUCLID(a, b)
if b = 0
    then return a
    else return EUCLID(b, a mod b)
```

As an example of the running of EUCLID, consider the computation of gcd(30, 21):

$$
\begin{aligned}
\text{EUCLID}(30, 21) &= \text{EUCLID}(21, 9) \\
&= \text{EUCLID}(9, 3) \\
&= \text{EUCLID}(3, 0) \\
&= 3 .
\end{aligned}
$$

Extended Euclid

- Extend the algorithm to compute integer coefficient x, y such that:

  $d = \gcd(a,b) = ax + by$

  Note that x,y may be zero or negative.

Algorithm:

EXTENDED-EUCLID takes as input a pair of nonnegative integers and returns a triple of the form (d, x, y) that satisfies equation

```
EXTENDED-EUCLID(a, b)
if b = 0
    then return (a, 1, 0)
(d', x', y') ← EXTENDED-EUCLID(b, a mod b)
(d, x, y) ← (d', y', x' − ⌊a/b⌋ y')
return (d, x, y)
```

Continued

The EXTENDED-EUCLID procedure is a variation of the EUCLID procedure. Line 1 is equivalent to the test "$b = 0$" in line 1 of EUCLID. If $b = 0$, then EXTENDED-EUCLID returns not only $d = a$ in line 2, but also the coefficients $x = 1$ and $y = 0$, so that $a = ax + by$. If $b \neq 0$, EXTENDED-EUCLID first computes (d', x', y') such that $d' = \gcd(b, a \bmod b)$ and

$$d' = bx' + (a \bmod b)y' .$$

As for EUCLID, we have in this case $d = \gcd(a, b) = d' = \gcd(b, a \bmod b)$. To obtain x and y such that $d = ax + by$, we start by rewriting equation

$$\begin{aligned} d &= bx' + (a - \lfloor a/b \rfloor b)y' \\ &= ay' + b(x' - \lfloor a/b \rfloor y') . \end{aligned}$$

Since

$$a \bmod n = a - \lfloor a/n \rfloor n$$

Example

a	b	$\lfloor a/b \rfloor$	d	x	y
99	78	1	3	−11	14
78	21	3	3	3	−11
21	15	1	3	−2	3
15	6	2	3	1	−2
6	3	2	3	0	1
3	0	—	3	1	0

An example of the operation of EXTENDED-EUCLID on the inputs 99 and 78. Each line shows one level of the recursion: the values of the inputs a and b, the computed value $\lfloor a/b \rfloor$, and the values d, x, and y returned. The triple (d, x, y) returned becomes the triple (d', x', y') used in the computation at the next higher level of recursion. The call EXTENDED-EUCLID(99, 78) returns $(3, -11, 14)$, so $\gcd(99, 78) = 3$ and $\gcd(99, 78) = 3 = 99 \cdot (-11) + 78 \cdot 14$.

Repeated Squaring

Let $\langle b_k, b_{k-1}, \ldots, b_1, b_0 \rangle$ be the binary representation of b. (That is, the binary representation is $k+1$ bits long, b_k is the most significant bit, and b_0 is the least significant bit.) The following procedure computes $a^c \bmod n$ as c is increased by doublings and incrementations from 0 to b.

```
MODULAR-EXPONENTIATION(a, b, n)
c ← 0
d ← 1
let ⟨b_k, b_{k-1}, ..., b_0⟩ be the binary representation of b
for i ← k downto 0
     do c ← 2c
        d ← (d · d) mod n
        if b_i = 1
          then c ← c + 1
               d ← (d · a) mod n
return d
```

Example

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

As an example, for $a = 7$, $b = 560$, and $n = 561$, the algorithm computes the sequence of values modulo 561

The variable c is not really needed by the algorithm but is included for explanatory purposes;