Discrete Mathematics

complexity of arithmetic operations, complexity of arithmetic operations modulo N, square-and-multiply

Liangfeng Zhang
School of Information Science and Technology
ShanghaiTech University

Addition

Bit Length of Integer: $\ell(a) = \begin{cases} \lfloor \log_2(|a|) \rfloor + 1 & a \neq 0 \\ 1 & a = 0 \end{cases}$

Binary Representation: a 0-1 sequence

- $a = (a_{k-1} \dots a_1 a_0)_2 \Leftrightarrow a = a_{k-1} 2^{k-1} + \cdots + a_1 \cdot 2^1 + a_0 \cdot 2^0$

Algorithm for Addition:

- **Input**: $a = (a_{k-1} \cdots a_1 a_0)_2, b = (b_{k-1} \cdots b_1 b_0)_2$
- **Output**: $c = a + b = (c_k c_{k-1} \cdots c_1 c_0)_2$
  - $carry \leftarrow 0$
  - for $i \leftarrow 0$ to $k - 1$ do
    - $t \leftarrow a_i + b_i + carry$;
    - set c_i and $carry$ such that $t = 2 \cdot carry + c_i$
  - $c_k \leftarrow carry$
- **Complexity**: $O(k)$ bit operations

Subtraction

Algorithm for Subtraction:

- **Input**: $a = (a_{k-1} \cdots a_1 a_0)_2, b = (b_{k-1} \cdots b_1 b_0)_2, a \geq b$
- **Output**: $c = a - b = (c_{k-1} \cdots c_1 c_0)_2$
  - $carry \leftarrow 0$
  - for $i \leftarrow 0$ to $k - 1$ do
    - $t \leftarrow a_i - b_i + carry$;
    - set c_i and $carry$ such that $t = 2 \cdot carry + c_i$
- **Complexity**: $O(k)$ bit operations

Multiplication

Algorithm for Multiplication:

- **Input**: $a = (a_{k-1} \cdots a_0)_2, b = (b_{k-1} \cdots b_0)_2$
- **Output**: $c = ab = (c_{2k-1} \cdots c_0)_2$
  - $c \leftarrow 0; x \leftarrow a$
  - for $i \leftarrow 0$ to $k - 1$ do
    - if $b_i = 1$, then $c \leftarrow c + x$;
    - $x \leftarrow x + x$;
- **Complexity**: $O(k^2)$ bit operations

Division

Algorithm for Division:

- **Input**: $a = (a_{k-1} \cdots a_0)_2, b = (b_{l-1} \cdots b_0)_2, a \geq b, a_{k-1} = b_{l-1} = 1$
- **Output**: $q = (q_{k-l} \cdots q_0)_2$ and $r = (r_{l-1} \cdots r_0)_2$ s.t. $a = bq + r, 0 \leq r < b$
  - $(r_k r_{k-1} \cdots r_0)_2 \leftarrow (0a_{k-1} \cdots a_0)_2$
  - for $i \leftarrow k - l$ down to 0 do
    - $q_i \leftarrow 2r_{i+l} + r_{i+l-1}$;
    - If $q_i \geq 2$, then $q_i \leftarrow 1$;
    - $(r_{i+l} \cdots r_i)_2 \leftarrow (r_{i+l} \cdots r_i)_2 - q_i \cdot b$;
    - while $(r_{i+l} \cdots r_i)_2 < 0$ do
      - $(r_{i+l} \cdots r_i)_2 \leftarrow (r_{i+l} \cdots r_i)_2 + b$;
      - $q_i \leftarrow q_i - 1$;
  - output $q = (q_{k-l} \cdots q_0)_2$ and $r = (r_{l-1} \cdots r_0)$;
- **Complexity**: $O((k - l + 1) \cdot l)$ bit operations

Arithmetic Modulo N

THEOREM: Let $a, b \in \{0,1,\dots,N-1\}$. Then

- $(a \pm b) \bmod N$ can be computed in $O(\ell(N))$ bit operations
  - $\ell(a), \ell(b) \leq \ell(N)$
    - $a \pm b$ are computable in $O(\ell(N))$ bit operations
  - $0 \leq |a+b|, |a-b| < 2N$
    - $(a \pm b) \bmod N$ are computable in $O((\ell(2N) - \ell(N) + 1)\ell(N)) = O(\ell(N))$ bit operations
- $(ab) \bmod N$ can be computed in $O(\ell(N)^2)$ bit operations
  - $\ell(a), \ell(b) \leq \ell(N)$
    - ab is computable in $O(\ell(N)^2)$ bit operations
  - $0 \leq |ab| < N^2$
    - $(ab) \bmod N$ is computable in $O\big((\ell(N^2) - \ell(N) + 1)\ell(N)\big) = O(\ell(N)^2)$ bit operations.

Arithmetic Modulo N

Modulo exponentiation: For $0 \leq a < N, e \in \mathbb{N}$, $a^e \bmod N =?$

- Complexity? How to compute efficiently?

EXAMPLE: modulo exponentiation

- m =1437339113920498981637906207424471163445460406448981415203760376263650078098 9961566579389511210479437355107978772736352915127780140263030574243344234098335 8787394193855033926469913603762712163723160462115649025
- e =4631001162549482394344687394431824369029736722768833120796257387139181880015 6614404181253994785434292576255362553884181998492463297303466464428022018327564 72381022836757671552531962337198345690506439249417 6785
- N =2452466449002782119765176635730880184670267876783327597434144517150616008300 3858721695220839933207154910362682719167986407977672324300560059203563124656121 84658179041001318592996199338170121493350348758705510 67
- $\phi(N)$ =2452466449002782119765176635730880184670267876783327597434144517150616008 3003858721695220839933207154910262832286162703918422049427031393870390628339228 84877243942517668927868176971783437997584812286480916 67216
- $m^e \bmod N =?$

Arithmetic Modulo N

Modulo exponentiation: For $0 \le a < N, e \in \mathbb{N}, a^e \bmod N = ?$

- Complexity? How to compute efficiently?

EXAMPLE: modulo exponentiation

- $xy \bmod N = \big((x \bmod N) \cdot y\big) \bmod N = \big(x \cdot (y \bmod N)\big) \bmod N$
  $= \big((x \bmod N) \cdot (y \bmod N)\big) \bmod N$
- $a_1 = a$ $\quad = a^1 \bmod N$
- $a_2 = (a_1 \cdot a) \bmod N$ $\quad = a^2 \bmod N$
- $a_3 = (a_2 \cdot a) \bmod N$ $\quad = a^3 \bmod N$
- $\vdots$ $\quad \vdots$
- $a_e = (a_{e-1} \cdot a) \bmod N$ $\quad = a^e \bmod N$
- **Complexity**: $O(e)$ multiplications modulo N
  - $e = 2^{2048}$? --very slow

Square-and-Multiply

ALGORITHM: (200 BC) compute $a^e \bmod N$ in polynomial time

- **Input:** $a \in \{0,1,\dots,N-1\}$; $e = (e_{k-1} \cdots e_0)_2$ $\quad //k = \ell(e)$
  - $e = e_{k-1} \cdot 2^{k-1} + \cdots + e_1 \cdot 2^1 + e_0 \cdot 2^0$
- **Output:** $a^e \bmod N$
  - **Square**: this step requires $O(k)$ multiplications modulo N
    - $x_0 = a$
    - $x_1 = (x_0^2 \bmod N) = (a^2 \bmod N)$
    - $x_2 = (x_1^2 \bmod N) = (a^{2^2} \bmod N)$
    - …
    - $x_{k-1} = (x_{k-2}^2 \bmod N) = (a^{2^{k-1}} \bmod N)$
  - **Multiply**: this step requires $O(k)$ multiplications modulo N
    - $(a^e \bmod N) = (x_0^{e_0} \cdot x_1^{e_1} \cdots x_{k-1}^{e_{k-1}} \bmod N)$
- **Complexity**: $O(k)$ multiplications modulo N --fast

Square-and-Multiply

EXAMPLE: Compute 2^{123} mod 35 using square-and-multiply.

- **Input**: $a = 2; N = 35;\ e = 123 = (1\ 1\ 1\ 1\ 0\ 1\ 1)_2;\ k = 7$
- **Square:** $k - 1$ multiplications modulo N will be done
  - $x_0 = a = 2;$
  - $x_1 = x_0^2 \text{ mod } N = 4$
  - $x_2 = x_1^2 \text{ mod } N = 16$
  - $x_3 = x_2^2 \text{ mod } N = 11$
  - $x_4 = x_3^2 \text{ mod } N = 16$
  - $x_5 = x_4^2 \text{ mod } N = 11$
  - $x_6 = x_5^2 \text{ mod } N = 16$
- **Multiply:** at most $k - 1$ multiplications modulo N will be done
  - $a^e = x_0 x_1 x_3 x_4 x_5 x_6 = 2 \times 4 \times 11 \times 16 \times 11 \times 16 \equiv 8 \pmod{35}$
  - $(2^{123} \text{ mod } 35) = 8$