

ECE 443/518 – Computer Cyber Security

Lecture 08 Euclidean Algorithm, Fermat's Little Theorem

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

September 10, 2025

Outline

Euclidean Algorithm

Fermat's Little Theorem

Reading Assignment

- ▶ This lecture: UC 6.3
- ▶ Next lecture: UC 6, 7, except 7.6

Outline

Euclidean Algorithm

Fermat's Little Theorem

Euclidean Algorithm

Input: two integers $a \geq b > 0$

1 $r_0 = a, r_1 = b, i = 1$

2 **Do:**

3 $i = i + 1$

4 $r_i = r_{i-2} \bmod r_{i-1}$

5 **While** $r_i \neq 0$

Output: $\gcd(a, b) = r_{i-1}$

- ▶ Example: $r_0 = 27, r_1 = 21, r_2 = 6, r_3 = 3, r_4 = 0$
 - ▶ In practice, there is no need to keep each r_k – we use them just for ease of presentation.
- ▶ For a proof of correctness
 - ▶ $r_{i-1} = \gcd(0, r_{i-1}) = \gcd(r_i, r_{i-1}) = \dots = \gcd(r_k, r_{k-1})$
 $= \gcd(r_{k-2} \bmod r_{k-1}, r_{k-1}) = \gcd(r_{k-2}, r_{k-1}) = \dots$
 $= \gcd(r_1, r_0) = \gcd(a, b)$
 - ▶ $r_1 > r_2 > \dots > r_{i-1} > r_i = 0$
- ▶ Is this algorithm better than the simple one?

Time Complexity of Euclidean Algorithm

- ▶ Let $q_{k-1} = \lfloor \frac{r_{k-2}}{r_{k-1}} \rfloor$. Since $r_{k-2} \geq r_{k-1}$, $q_{k-1} \geq 1$. So,

$$r_{k-2} = q_{k-1}r_{k-1} + r_k \geq r_{k-1} + r_k \geq 2r_k, \forall k = 2, 3, \dots, i.$$

- ▶ For i being odd, we have,

$$a = r_0 \geq 2r_2 \geq 2^2r_4 \geq \dots \geq 2^{\frac{i-1}{2}} r_{i-1} \geq 2^{\frac{i-1}{2}}.$$

- ▶ Similar for i being even.
- ▶ The loop iterates $O(\log a) = O(N)$ rounds.
 - ▶ Overall the time complexity is $O(N^3)$.
- ▶ GCD can be computed efficiently in polynomial time.
 - ▶ What is the complexity to obtain any divisor of a that is not 1 or a ? Or to prove that a is a prime number?

Extended Euclidean Algorithm (EEA)

Input: two integers $a \geq b > 0$
1 $r_0 = a, r_1 = b, s_0 = 1, t_0 = 0, s_1 = 0, t_1 = 1, i = 1$
2 **Do**:
3 $i = i + 1$
4 $r_i = r_{i-2} \bmod r_{i-1}, q_{i-1} = \lfloor \frac{r_{i-2}}{r_{i-1}} \rfloor$
5 $s_i = s_{i-2} - q_{i-1}s_{i-1}, t_i = t_{i-2} - q_{i-1}t_{i-1}$
6 **While** $r_i \neq 0$
Output: $\gcd(a, b) = r_{i-1}, s = s_{i-1}, t = t_{i-1}$

- ▶ Same time complexity as Euclidean Algorithm: $O(N^3)$
 - ▶ Same rounds of iterations. Additional calculations do not increase complexity.
- ▶ Anything special?

Extended Euclidean Algorithm (EEA Cont.)

k	0	1	2	3	4
r_k	27	21	6	3	0
$\begin{pmatrix} 1 \\ -q_{k-1} \end{pmatrix}$			$\begin{pmatrix} 1 \\ -1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ -3 \end{pmatrix}$	$\begin{pmatrix} 1 \\ -2 \end{pmatrix}$
$\begin{pmatrix} s_k \\ t_k \end{pmatrix} = \begin{pmatrix} s_{k-2} & s_{k-1} \\ t_{k-2} & t_{k-1} \end{pmatrix} \begin{pmatrix} 1 \\ -q_{k-1} \end{pmatrix}$	1 0	0 1	1 -1	-3 4	7 -9

- Starting with $(r_0 \ r_1) = (a \ b) \begin{pmatrix} s_0 & s_1 \\ t_0 & t_1 \end{pmatrix}$. We have,

$$\begin{aligned} (r_1 \ r_2) &= (r_0 \ r_1) \begin{pmatrix} 0 & 1 \\ 1 & -q_1 \end{pmatrix} = (a \ b) \begin{pmatrix} s_0 & s_1 \\ t_0 & t_1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -q_1 \end{pmatrix} \\ &= (a \ b) \begin{pmatrix} s_1 & s_2 \\ t_1 & t_2 \end{pmatrix} \end{aligned}$$

- So we can prove $(r_{i-1} \ r_i) = (a \ b) \begin{pmatrix} s_{i-1} & s_i \\ t_{i-1} & t_i \end{pmatrix}$.

- In other words, $as + bt = as_{i-1} + bt_{i-1} = r_{i-1} = \gcd(a, b)$

Solve Modular Algebra Equations

Assume $\gcd(a, m) = 1$. Solve

$$ax \equiv b \pmod{m}$$

- ▶ Apply EEA to find s and t such that $as + mt = 1$.
- ▶ Now we have $as \equiv 1 \pmod{m}$
- ▶ So $x \equiv 1 \cdot x \equiv asx \equiv (ax)s \equiv bs \pmod{m}$
- ▶ Equation solved!
 - ▶ Time complexity is $O(N^3)$, dominated by EEA.

Examples 1

- Solve $5x \equiv 1 \pmod{192}$.

k	0	1	2	3	4
r_k	192	5			
$\begin{pmatrix} 1 \\ -q_{k-1} \end{pmatrix}$					
$\begin{pmatrix} s_k \\ t_k \end{pmatrix}$	1	0			
	0	1			

- $192 \cdot (-2) + 5 \cdot 77 = 1$
► $x \equiv 77 \pmod{192}$

Solve System of Modular Algebra Equations

Assume $\gcd(m_1, m_2) = 1$. Solve

$$x \equiv a_1 \pmod{m_1}, \quad x \equiv a_2 \pmod{m_2}$$

- ▶ A.k.a. Chinese Remainder Theorem.
- ▶ Apply EEA to find s and t such that $m_1s + m_2t = 1$.
- ▶ Solution: $x \equiv a_1m_2t + a_2m_1s \pmod{m_1m_2}$
- ▶ Check: $x \equiv a_1m_2t \equiv a_1(1 - m_1s) \equiv a_1 \pmod{m_1}$.
 $x \equiv a_2m_1s \equiv a_2(1 - m_2t) \equiv a_2 \pmod{m_2}$.
- ▶ In particular, if $a_1 = a_2 = a$, the solution is
 $x \equiv am_2t + am_1s \equiv a(m_2t + m_1s) \equiv a \pmod{m_1m_2}$
- ▶ Time complexity is $O(N^3)$, dominated by EEA.

Examples 2

- Solve $x \equiv 6 \pmod{13}$, $x \equiv 11 \pmod{17}$

k	0	1	2	3	4
r_k	17	13			
$\begin{pmatrix} 1 \\ -q_{k-1} \end{pmatrix}$					
$\begin{pmatrix} s_k \\ t_k \end{pmatrix}$	1	0			
	0	1			

- $17 \cdot (-3) + 13 \cdot 4 = 1$
► $x \equiv 11 \cdot 13 \cdot 4 + 6 \cdot 17 \cdot (-3) \equiv 266 \equiv 45 \pmod{221}$

Outline

Euclidean Algorithm

Fermat's Little Theorem

Modular n -th Root

- ▶ What about modular n -th root?

$$x^n \equiv a \pmod{m}.$$

- ▶ Obviously you can solve it via brute-force in $O(2^N)$ time for a N -bit m . However, this is not what we are interested into.
- ▶ Consider the case when $m = p$ is a prime number first.

Fermat's Little Theorem

- ▶ Consider an integer x that is not a multiple of p .
- ▶ What does the sequence $kx \bmod p$ look like for $k = 1, 2, \dots, p-1$?
 - ▶ A permutation of $1, 2, \dots, p-1$ since
 - ▶ These $p-1$ remainders are all within $1, 2, \dots, p-1$.
 - ▶ They are all different since p is prime.
 - ▶ So $x \cdot (2x) \cdots ((p-1)x) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p}$.
- ▶ In other words, $(p-1)!x^{p-1} \equiv (p-1)! \pmod{p}$.
 - ▶ So $x^{p-1} \equiv 1 \pmod{p}$ since $\gcd((p-1)!, p) = 1$.
- ▶ Fermat's Little Theorem: $x^p \equiv x \pmod{p}$
 - ▶ Also include the case $x \equiv 0 \pmod{p}$.
- ▶ Example: $2^{13} \equiv 2 \pmod{13}$, $3^{13} \equiv 3 \pmod{13}$.

Summary

- ▶ EEA is essential for solving modular algebra equations.
 - ▶ In particular, if $\gcd(a, b) = 1$, we can apply EEA to find integers s and t such that $as + bt = 1$.
- ▶ EEA is efficient with a time complexity of $O(N^3)$ for N -bit inputs.