

THUẬT TOÁN TRONG AN TOÀN THÔNG TIN

Information Security Algorithms

HỌC VIỆN KỸ THUẬT MÃ KHÓA
ACADEMY OF CRYPTOGRAPHY TECHNIQUES

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 1

MỤC TIÊU

Trang bị kiến thức về một số thuật toán để thực hiện các tính toán hiệu quả ứng dụng trong an toàn thông tin, đặc biệt là trong mật mã khóa công khai và trong phát hiện tấn công, mã độc.

GIỚI THIỆU HỌC PHẦN

$$\begin{aligned} (x+1)^3 &= \left(\frac{x^2-1}{x-1}\right) + (x(x-1)) + \binom{3(x-1)}{2} \\ &= \left(\frac{x-1}{x-1}\right) + x(x-1) + \binom{x-1}{2} \\ &= 1 + x(x-1) + \frac{x(x-1)(x-2)}{2!} \\ &= 1 + (x+6)x - 9x^2 + \frac{x(x+6)(x-2)}{2!} \\ &= 1 + (x+6)x - 9x^2 + 27x^3 - 27x^4 + 27x^5 \\ &\quad - 9x + \sqrt{3} \cdot \sqrt{x^2 + 27x^3 - 27x^4 + 27x^5} \\ &\quad + (x+8x^2)^2 - (x+8x^2)^2 \\ &\quad + (x+8x^2)^2 (x+7x^3+8x^4+4x^5) \end{aligned}$$

THỜI LƯỢNG: 3tc

- 18 tiết lý thuyết
 - 24 tiết thực hành
- ĐÁNH GIÁ KẾT QUẢ HỌC TẬP**
- Điểm chuyên cần
 - Di học đầy đủ, đúng giờ
 - Tham gia xây dựng bài
 - Kiểm tra giữa kỳ: thi viết
 - Thi kết thúc học phần: Thực hành lập trình trên máy

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 3

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 2

GIỚI THIỆU HỌC PHẦN



Guide to Elliptic Curve Cryptography, Springer, 2004 (Chapter 2. Finite Field Arithmetic)



Darrel Hankerson, Alfred Menezes and Scott Vanstone,



Prime Numbers - A Computational Perspective (2nd edition), Springer, 2005



Richard Crandall and Carl Pomerance,



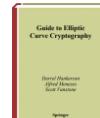
Algorithms and Theory of Computation Handbook: General Concepts and Techniques, CRC Press, 2010 (Chapter 13. Pattern Matching in Strings)

Mikhail J. Atallah and Marina Blanton



Và tài liệu khác

Google



16 May 2021 | Page 4

NỘI DUNG

01. TÍNH TOÁN TRÊN SỐ NGUYÊN LỚN
TRONG TRƯỜNG F_p
02. MỘT SỐ THUẬT TOÁN VỀ SỐ
NGUYÊN TỐ
03. ĐỔI SÁNH MẪU TRÊN CHUỖI



16 May 2021 | Page 5

HỌC VIỆN KỸ THUẬT MÃ KHÓA
ACADEMY OF CRYPTOGRAPHY TECHNIQUES

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

CHƯƠNG 01

TÍNH TOÁN TRÊN SỐ NGUYÊN LỚN TRONG TRƯỜNG F_p

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 6

HỌC VIỆN KỸ THUẬT MÃ KHÓA
ACADEMY OF CRYPTOGRAPHY TECHNIQUES

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 6



BÀI 01 - MỤC TIÊU

- Nắm được, cài đặt được các phép tính toán hiệu quả trên số nguyên lớn



BÀI 01 - MỤC TIÊU

- VD: Thời gian cần thiết để phân tích số nguyên n ra thừa số nguyên tố bằng thuật toán nhanh nhất hiện nay:

Số chữ số thập phân	Số phép tính bit	Thời gian
50	$1.4 \cdot 10^{10}$	3,9 giờ
75	$9 \cdot 10^{12}$	104 ngày
100	$2,3 \cdot 10^{15}$	74 năm
200	$1,2 \cdot 10^{23}$	$3,8 \cdot 10^9$ năm
300	$1,5 \cdot 10^{29}$	$4,9 \cdot 10^{15}$ năm
500	$1,3 \cdot 10^{39}$	$4,2 \cdot 10^{25}$ năm



TÍNH TOÁN TRÊN SỐ NGUYÊN LỚN TRONG TRƯỜNG F_p



- Giới thiệu về các trường hữu hạn
- Phép tính cộng và trừ
- Phép tính nhân
- Phép tính bình phương
- Phép lấy modulo
- Phép lũy thừa
- Phép tính nghịch đảo



Tính toán với modulo là số nguyên tố đặc biệt của NIST



Giới thiệu về các trường hữu hạn

- Trường là một tập hợp với 2 phép toán $(+, \cdot)$ thỏa mãn các tính chất số học thông thường:
 - $(F, +)$ là nhóm Abel với phép cộng
 - $(F \setminus \{0\}, \cdot)$ là nhóm abel với phép nhân
 - Tính phân phối: $(a+b)c = a.c + b.c \quad \forall a, b, c \in F$
- Trường hữu hạn (còn gọi là trường Galois) là những trường có hữu hạn số phần tử, số này gọi là bậc của trường đó.



Giới thiệu về các trường hữu hạn

- ...
- Các phép toán trên trường hữu hạn:
 - Có thể nói là có các phép toán cộng, trừ, nhân, chia số khác 0
 - Phép trừ được coi như là cộng với số đối của phép cộng $a - b = a + (-b)$
 - Phép chia là nhân với số đối của phép nhân $a/b = a \cdot b^{-1}$



Giới thiệu về các trường hữu hạn

- Số lượng phần tử của một trường hữu hạn được gọi là cấp hoặc bậc của nó.
- Trường hữu hạn F cấp q nếu và chỉ nếu q là lũy thừa nguyên tố p^m (trong đó p là số nguyên tố, m là số nguyên dương). Nếu $m = 1$ thì F được gọi là trường nguyên tố, nếu $m \geq 2$ được gọi là trường mở rộng.



Giới thiệu về các trường hữu hạn

- Trường nguyên tố $F_p = \{0, 1, \dots, p-1\}$ với các phép toán $(+, \cdot)$ thực hiện theo modulo p .

- Ví dụ:

- $F_{29} = \{0, 1, 2, \dots, 28\}$
 - Phép toán cộng: $17 + 20 = 8$ vì $37 \bmod 29 = 8$
 - Phép trừ: $17 - 20 = 26$ vì $-3 \bmod 29 = 26$
 - Phép nhân: $17 \cdot 20 = 21$ vì $340 \bmod 29 = 21$
 - Phép lấy nghịch đảo: $17^{-1} = 12$ vì $17 \cdot 12 \bmod 29 = 1$.



Giới thiệu về các trường hữu hạn

- Trường nhị phân $F_{2^m} = \{a_{m-1}z^{m-1} + a_{m-2}z^{m-2} + \dots + a_1z + a_0 \mid a_i \in \{0, 1\}\}$



Giới thiệu về các trường hữu hạn

- Ví dụ: F_{2^4} gồm 16 phần tử là các đa thức nhị phân có bậc cao nhất là 3

0	z^2	z^3	$z^3 + z^2$
1	$z^2 + 1$	$z^3 + 1$	$z^3 + z^2 + 1$
z	$z^2 + z$	$z^3 + z$	$z^3 + z^2 + z$
$z + 1$	$z^2 + z + 1$	$z^3 + z + 1$	$z^3 + z^2 + z + 1$



Giới thiệu về các trường hữu hạn

- Ví dụ một số phép toán trong F_{2^4} với đa thức rút gọn $f(z) = z^4 + z + 1$.
 - Phép cộng: $(z^3 + z^2 + 1) + (z^2 + z + 1) = z^3 + z$
 - Phép trừ: $(z^3 + z^2 + 1) - (z^2 + z + 1) = z^3 + z$. Lưu ý, vì $-1 = 1$ trong F_2 ($a - a = a$ với mọi $a \in F_{2^m}$).
 - Phép nhân: $(z^3 + z^2 + 1) \cdot (z^2 + z + 1) = ?$
 $\because (z^3 + z^2 + 1) \cdot (z^2 + z + 1) = z^5 + z + 1$ và $z^5 + z + 1 \bmod z^4 + z + 1 = z^2 + 1$
 - Nghịch đảo: $(z^3 + z^2 + 1)^{-1} = z^2$ vì $(z^3 + z^2 + 1) \cdot z^2 \bmod z^4 + z + 1 = 1$



Giới thiệu về các trường hữu hạn

- Nhóm nhân của trường hữu hạn:
 - Các phần tử khác 0 của trường hữu hạn F_q , KH: F_q^* là một nhóm cyclic với phép nhân, do đó tồn tại một phần tử sinh $b \in F_q^*$ sao cho: $F_q^* = \{b^i : 0 \leq i \leq q-2\}$
 - Cấp của phần tử $a \in F_q^*$ là số nguyên dương nhỏ nhất t sao cho $a^t = 1$. Vì F_q^* là nhóm cyclic nên t là ước của $q-1$.



TÍNH TOÁN TRÊN SỐ NGUYÊN LỚN TRONG TRƯỜNG F_p



Giới thiệu về các trường hữu hạn



Phép tính cộng và trừ



Phép tính nhân



Phép tính bình phương



Phép lấy modulo



Phép lũy thừa



Phép tính nghịch đảo



Tính toán với modulo là số nguyên tố đặc biệt của NIST



Phép tính cộng và trừ

- Các thuật toán cộng, trừ, nhân, chia,.. giới thiệu trong chương này phù hợp với triển khai phần mềm
- Ta giả thiết nền tảng triển khai có kiến trúc W – bit trong đó W là bội số của 8 (phổ biến là 64 – 32 bit), các hệ thống máy có công suất thấp có thể có W nhỏ hơn, VD: hệ thống nhúng W = 16 bit, thẻ thông minh W = 8 bit.
- Các bit của một W-bit là từ U được đánh số từ phải qua trái bắt đầu từ 0 đến W-1.

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 19



Phép tính cộng và trừ

- Ta có $F_p = \{0, \dots, p-1\}$.
- Tính $m = \lceil \log_2 p \rceil$ là độ dài bit của p và $t = \lceil m/W \rceil$ là độ dài từ của p
- Biểu diễn của phần tử a được lưu trữ trong một mảng $A = (A[t-1], \dots, A[2], A[1], A[0])$ của t các từ W bit, trong đó bit ngoài cùng bên phải của $A[0]$ là bit có trọng số thấp nhất.

A[t-1]	...	A[2]	A[1]	A[0]
--------	-----	------	------	------

- Biểu diễn $a \in F_p$ như một mảng A của các từ W-bit:

$$a = 2^{(t-1)W}A[t-1] + \dots + 2^{2W}A[2] + 2^WA[1] + A[0]$$

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 20



Phép tính cộng và trừ

- Ví dụ: cho $W=8$, xét $F_{2147483647}$, hãy biểu diễn số $a = 23456789$ dưới dạng mảng
 - Ta có $m = \lceil \log_2 p \rceil = \lceil \log_2 2147483647 \rceil = 31$, $t = \lceil m/W \rceil = \lceil 31/8 \rceil = 4$
 - Biểu diễn a dưới dạng mảng $(A[3], A[2], A[1], A[0])$
 - $$\begin{aligned} a &= 2^{(t-1)W}A[t-1] + \dots + 2^{2W}A[2] + 2^WA[1] + A[0] \\ &= 2^{(4-1) \cdot 8}A[4-1] + 2^{2 \cdot 8}A[2] + 2^8A[1] + A[0] \\ &= 2^{24}A[3] + 2^{16}A[2] + 2^8A[1] + A[0] \\ &= 2^{24} \cdot 1 + 2^{16} \cdot 101 + 2^8 \cdot 236 + 21 \end{aligned}$$

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 21

Vậy a được biểu diễn qua
mảng A: (1, 101, 236, 21)



Phép tính cộng và trừ

Bài tập áp dụng:

- Cho $W=8$, xét $F_{2147483647}$, hãy biểu diễn a dưới dạng mảng
 - $a = 765432 \rightarrow (0, 11, 173, 248)$
 - $a = 123456 \rightarrow (0, 1, 226, 64)$

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 22



Phép tính cộng và trừ

- Thuật toán cộng và trừ trên trường hữu hạn được đưa ra dưới dạng các thuật toán tương ứng cho các số nguyên w. Phép gán dạng “(ϵ, z) $\leftarrow w$ ” được định nghĩa như sau:
 - $z \leftarrow w \bmod 2^w$ và
 - $\epsilon \leftarrow 0$ nếu $w \in [0, 2^w]$, ngược lại $\epsilon \leftarrow 1$
 - Nếu $w = x + y + \epsilon'$ với $x, y \in [0, 2^w]$ và $\epsilon' \in \{0, 1\}$, thì $w = \epsilon 2^w + z$ và ϵ được gọi là “bit nhớ” (carry bit) cho phép cộng mỗi một từ đơn ($\epsilon = 1$ nếu và chỉ nếu $z < x + \epsilon'$)

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 23



Phép tính cộng và trừ

- Thuật toán cộng chính xác bội:

Algorithm 1. Multiprecision addition

- Input:** số nguyên $a, b \in [0, 2^{Wt}]$
Output: (ϵ, c) với $c = a + b \bmod 2^{Wt}$ và ϵ là bit nhớ
- $(\epsilon, C[0]) \leftarrow A[0] + B[0]$
 - For i from 1 to $t-1$ do
 - $(\epsilon, C[i]) \leftarrow A[i] + B[i] + \epsilon$
 - Return (ϵ, c)

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 24



Phép tính cộng và trừ

- ❖ Ví dụ minh họa thuật toán cộng chính xác bội:

- Cho $a = (0, 11, 173, 248)$; $b = (0, 1, 226, 64)$, với $w = 8$, $t = 4$.
- Áp dụng thuật toán 1 tìm $c = a + b \text{ mod } 2^{wt}$



Phép tính cộng và trừ

- ❖ $(\varepsilon, C[0]) \leftarrow A[0] + B[0] = 248 + 64 = 312 \text{ mod } 2^8 = 56$ (gán $\varepsilon = 1$)
- ❖ $i = 1: (\varepsilon, C[1]) \leftarrow A[1] + B[1] + \varepsilon = 173 + 226 + 1 \text{ mod } 2^8 = 400$
 $\text{mod } 2^8 = 144$ (gán $\varepsilon = 1$)
- ❖ $i = 2: (\varepsilon, C[2]) \leftarrow A[2] + B[2] + \varepsilon = 11 + 1 + 1 \text{ mod } 2^8 = 13$ (gán $\varepsilon = 0$)
- ❖ $i = 3: (\varepsilon, C[3]) \leftarrow A[3] + B[3] + \varepsilon = 0 + 0 + 0 \text{ mod } 2^8 = 0$ (gán $\varepsilon = 0$)
- ❖ Return $(0, (0, 13, 144, 56))$



Phép tính cộng và trừ

Bài tập:

- Cho $W = 8$, $t = 4$. Áp dụng thuật toán 1 tìm $c = a + b \text{ mod } 2^{wt}$ với $a = (57, 169, 36, 27)$; $b = (0, 98, 34, 62)$



Phép tính cộng và trừ

- ❖ Thuật toán trừ chính xác bội:

Algorithm 2. Multiprecision subtraction

Input: số nguyên $a, b \in [0, 2^{wt}]$

Output: (ε, c) với $c = a - b \text{ mod } 2^{wt}$ và ε là bit mượn

1. $(\varepsilon, C[0]) \leftarrow A[0] - B[0]$
2. For i from 1 to $t - 1$ do
 - 2.1 $(\varepsilon, C[i]) \leftarrow A[i] - B[i] - \varepsilon$
3. Return (ε, c)



Phép tính cộng và trừ

- ❖ Ví dụ minh họa thuật toán cộng chính xác bội:

- Cho $a = (0, 11, 173, 248)$; $b = (0, 1, 226, 64)$, với $w = 8$, $t = 4$.
- Áp dụng thuật toán 1 tìm $c = a - b \text{ mod } 2^{wt}$



Phép tính cộng và trừ

- ❖ $(\varepsilon, C[0]) \leftarrow A[0] - B[0] = 248 - 64 = 184 \text{ mod } 2^8$ (gán $\varepsilon = 0$)
- ❖ $i = 1: (\varepsilon, C[1]) \leftarrow A[1] - B[1] - \varepsilon = 173 - 226 - 0 \text{ mod } 2^8 = -53 \text{ mod } 2^8 = 203$ (gán $\varepsilon = 1$)
- ❖ $i = 2: (\varepsilon, C[2]) \leftarrow A[2] - B[2] - \varepsilon = 11 - 1 - 1 \text{ mod } 2^8 = 9$ (gán $\varepsilon = 0$)
- ❖ $i = 3: (\varepsilon, C[3]) \leftarrow A[3] + B[3] - \varepsilon = 0 - 0 - 0 \text{ mod } 2^8 = 0$ (gán $\varepsilon = 0$)
- ❖ Return $(0, (0, 9, 203, 184))$



Phép tính cộng và trừ

Bài tập:

- Cho $W = 8$, $t = 4$. Áp dụng thuật toán 2 tính $c = a - b \bmod 2^{Wt}$ với $a = (57, 169, 36, 27)$; $b = (0, 98, 34, 62)$



Phép tính cộng và trừ

Thuật toán cộng trên F_p :

Algorithm 3. Addition in F_p

Input: số modulo p , số nguyên $a, b \in [0, p - 1]$

Output: $c = a + b \bmod p$

1. Dùng thuật toán Algorithm 1 để thu được (ε, c) với $c = a + b \bmod 2^{Wt}$ và ε là bit nhớ.

2. Nếu $\varepsilon = 1$ thì trừ p từ $c = (C[t-1], \dots, C[2], C[1], C[0])$;

Ngược lại nếu $c \geq p$ thì $c \leftarrow c - p$

3. Return (c)



Phép tính cộng và trừ

Ví dụ minh họa thuật toán 3:

- Cho $p = 2.147.483.647$, $W = 8$; ta có $m = \lceil \log_2 p \rceil = 31$; $t = \lceil m/W \rceil = 4$
- $a = (0, 11, 173, 248)$; $b = (0, 1, 226, 64)$.
- Áp dụng thuật toán 3 tìm $c = a + b \bmod p$
 - Áp dụng thuật toán 1 ta có $(\varepsilon, c) = (0, (0, 13, 144, 56))$
 - Vì $c < p \Rightarrow$ Return $(0, 13, 144, 56)$



Phép tính cộng và trừ

Bài tập:

- Cho $p = 2.147.483.647$, $W = 8$; ta có $m = \lceil \log_2 p \rceil = 31$; $t = \lceil m/W \rceil = 4$
- $a = (157, 0, 173, 23)$; $b = (169, 1, 0, 64)$.
- Áp dụng thuật toán 3 tìm $c = a + b \bmod p$



Phép tính cộng và trừ

$$\diamond a = (157, 0, 173, 23); b = (169, 1, 0, 64)$$

Áp dụng tt 1 tìm (ε, c) :

- $(\varepsilon, C[0]) \leftarrow A[0] + B[0] = 23 + 64 = 87 \bmod 2^8 = 87$ (gán $\varepsilon = 0$)
 - $i = 1: (\varepsilon, C[1]) \leftarrow A[1] + B[1] + \varepsilon = 173 + 0 + 0 \bmod 2^8 = 173$ (gán $\varepsilon = 0$)
 - $i = 2: (\varepsilon, C[2]) \leftarrow A[2] + B[2] + \varepsilon = 0 + 1 + 0 \bmod 2^8 = 1$ (gán $\varepsilon = 0$)
 - $i = 3: (\varepsilon, C[3]) \leftarrow A[3] + B[3] + \varepsilon = 157 + 169 + 0 \bmod 2^8 = 326 \bmod 2^8 = 70$ (gán $\varepsilon = 1$)
 - Vậy $(\varepsilon, c) = (1, (70, 1, 173, 87))$

Ta có $\varepsilon = 1$



Phép tính cộng và trừ

- Ta có $p = 2.147.483.647$ được biểu diễn dưới dạng mảng $(127, 255, 255, 255)$
- Áp dụng tt2 tính $c = p \bmod 2^{32}$ (với $c = (70, 1, 173, 87)$)
 - $(\varepsilon, C[0]) \leftarrow c[0] - p[0] = 87 - 255 = -168 \bmod 2^8 = 88$ (gán $\varepsilon = 1$)
 - $i = 1: (\varepsilon, C[1]) \leftarrow c[1] - p[1] - \varepsilon = 173 - 255 - 1 \bmod 2^8 = -83 \bmod 2^8 = 173$ (gán $\varepsilon = 1$)
 - $i = 2: (\varepsilon, C[2]) \leftarrow c[2] - p[2] - \varepsilon = 1 - 255 - 1 \bmod 2^8 = -255 \bmod 2^8 = 1$ (gán $\varepsilon = 1$)
 - $i = 3: (\varepsilon, C[3]) \leftarrow c[3] - p[3] - \varepsilon = 70 - 127 - 1 \bmod 2^8 = -58 \bmod 2^8 = 198$ (gán $\varepsilon = 1$)
 - Return $(1, (198, 1, 173, 88))$



Phép tính cộng và trừ

- Thuật toán trừ trên \mathbb{F}_q :

Algorithm 4. Subtraction in \mathbb{F}_q

Input: số modulo p, số nguyên a, b $\in [0, p - 1]$

Output: c = a - b mod p

- Dùng thuật toán Algorithm 2 để thu được (ε, c) với $c = a - b \text{ mod } 2^W$ và ε là bit dư.
- Nếu $\varepsilon = 1$ thì thêm p từ c = ($C[t-1], \dots, C[2], C[1], C[0]$);
- Return (c)



Phép tính cộng và trừ

- BT áp dụng:

Cho $p = 2.147.483.647$, $W = 8$; ta có $m = \lceil \log_2 p \rceil = 31$; $t = \lceil m/W \rceil = 4$

a = (0, 11, 173, 248); b = (0, 1, 226, 64).

Áp dụng thuật toán 4 tìm c = a - b mod p



TÍNH TOÁN TRÊN SỐ NGUYÊN LỚN TRONG TRƯỜNG \mathbb{F}_p



- Giới thiệu về các trường hữu hạn
- Phép tính cộng và trừ
- Phép tính nhân
- Phép tính bình phương
- Phép lấy modulo
- Phép lũy thừa
- Phép tính nghịch đảo



Tính toán với modulo là số nguyên tố đặc biệt của NIST



Phép tính nhân

- Thuật toán nhân

Trong đó UV biểu thị cho 2W bit được nối bởi W bit của từ U với W bit từ V

Algorithm 4. Integer multiprecision (operand scanning form)

Input: số nguyên a, b $\in [0, p - 1]$

Output: c = a . b

- | | |
|--|---|
| 1. For i from 0 to t - 1 do
1.1 C[i] $\leftarrow 0$ | (UV) $\leftarrow C[i + j] + A[i]. B[j] + U$ |
| 2. For i from 0 to t - 1 do
2.1 U $\leftarrow 0$ | C[i + j] $\leftarrow V$ |
| 2.2 For j from 0 to t - 1 do | 2.3. C[i + t] $\leftarrow U$ |
| 2.2.1 For k from 0 to 2t - 2 do | 3. Return(c). |



Phép tính nhân

- Ví dụ:

- Cho $p = 2.147.483.647$, $W = 8$; ta có $m = \lceil \log_2 p \rceil = 31$; $t = \lceil m/W \rceil = 4$
- a = (0, 11, 173, 248); b = (0, 1, 226, 64).
- Tính c = a.b



Phép tính nhân

Algorithm 4. Integer multiprecision (product scanning form)

Input: số nguyên a, b $\in [0, p - 1]$

Output: c = a . b

- | | |
|--|---|
| 1. R ₀ $\leftarrow 0$, R ₁ $\leftarrow 0$, R ₂ $\leftarrow 0$. | (ε, R ₀) $\leftarrow R_0 + V$
(ε, R ₁) $\leftarrow R_1 + U + ε$
R ₂ $\leftarrow R_2 + ε$ |
| 2. For k from 0 to 2t - 2 do | 2.2 C[k] $\leftarrow R_0$, R ₀ $\leftarrow R_1$, R ₁ $\leftarrow R_2$, R ₂ $\leftarrow 0$ |
| 2.1 For (i, j) $\in \{(i, j) \mid i + j = k, 0 \leq i, j \leq t - 1\}$ do | 3. C[2t - 1] $\leftarrow R_0$ |
| (U, V) $\leftarrow A[i]. B[j]$ | 4. Return(c). |



TÍNH TOÁN TRÊN SỐ NGUYỄN LỚN TRONG TRƯỜNG F_p



- Giới thiệu về các trường hữu hạn
- Phép tính cộng và trừ
- Phép tính nhân
- Phép tính bình phương
- Phép lấy modulo
- Phép lũy thừa
- Phép tính nghịch đảo



Tính toán với modulo là số nguyên tố đặc biệt của NIST



Phép tính bình phương

Algorithm 5. Integer squaring

Input: số nguyên $a \in [0, p - 1]$
Output: $c = a^2$

- | | |
|---|--|
| <ol style="list-style-type: none"> $R_0 \leftarrow 0, R_1 \leftarrow 0, R_2 \leftarrow 0.$ For k from 0 to $2t - 2$ do <ol style="list-style-type: none"> For $(i, j) \in \{(i, j) \mid i + j = k, 0 \leq i, j \leq t - 1\}$ do $(U, V) \leftarrow A[i], A[j]$ | <ol style="list-style-type: none"> If $(i < j)$ then do: $(\varepsilon, UV) \leftarrow R_0 + V$
 $(\varepsilon, R_1) \leftarrow R_1 + U + \varepsilon$
 $R_2 \leftarrow R_2 + \varepsilon$ $C[2t - 1] \leftarrow R_0, R_0 \leftarrow R_1, R_1 \leftarrow R_2, R_2 \leftarrow 0$ $C[2t - 1] \leftarrow R_0, R_0 \leftarrow R_1, R_1 \leftarrow R_2, R_2 \leftarrow 0$ |
| | <ol style="list-style-type: none"> Return(c). |



TÍNH TOÁN TRÊN SỐ NGUYỄN LỚN TRONG TRƯỜNG F_p



- Giới thiệu về các trường hữu hạn
- Phép tính cộng và trừ
- Phép tính nhân
- Phép tính bình phương
- Phép lấy modulo
- Phép lũy thừa
- Phép tính nghịch đảo



Tính toán với modulo là số nguyên tố đặc biệt của NIST



Phép lấy modulo

Algorithm 6. Barrett reduction

Input: $p, b \geq 3, k = \lfloor \log_b p \rfloor + 1, 0 \leq z < b^{2k},$ và $\mu = \lfloor b^{2k} / p \rfloor$
Output: $z \bmod p$

- $\hat{q} \leftarrow \lfloor z/b^{k-1} \cdot \mu/b^{k+1} \rfloor$
- $r \leftarrow (z \bmod b^{k-1}) - (\hat{q} \cdot p \bmod b^{k-1})$
- If $r < 0$ then $r \leftarrow r + b^{k-1}$
- While $r \geq p$ do $r \leftarrow r - p$
- Return (r)



BÀI 02 - MỤC TIÊU

- Nắm được, cài đặt được các phép tính toán hiệu quả trên số nguyên lớn



TÍNH TOÁN TRÊN SỐ NGUYỄN LỚN TRONG TRƯỜNG F_p

- Giới thiệu về các trường hữu hạn
- Phép tính cộng và trừ
- Phép tính nhân
- Phép tính bình phương
- Phép lấy modulo
- Phép lũy thừa
- Phép tính nghịch đảo



Tính toán với modulo là số nguyên tố đặc biệt của NIST



Phép lũy thừa

- ❖ Lấy $R > p$ với $\gcd(R, p) = 1$. Tính $zR^{-1} \bmod p$ với $z < pR$
- ❖ Xét p là số lẻ, $R = 2^{wt}$. Nếu $p' = -p^{-1} \bmod R$ thì $c = zR^{-1} \bmod p$ có thể gồm:
 - $c \leftarrow (z + (zp' \bmod R)p)/R$
 - Nếu $c \geq p$ thì $c \square c - p$
- ❖ Với $t(t+1)$ phép nhân chính xác đơn
- ❖ Cho $x \in [0, p]$, $\tilde{x} \leftarrow xR \bmod p$. Chú ý: $(\tilde{x}\tilde{y})R^{-1} \bmod p = (xy)R \bmod p$
- ❖ Ta định nghĩa tích của \tilde{x} và \tilde{y} :
 - $\text{Mont}(\tilde{x}, \tilde{y}) = \tilde{x}\tilde{y}R^{-1} \bmod p = xyR \bmod p$

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 49



Phép lũy thừa

Algorithm 7. Montgomery exponentiation (basic)

Input: số nguyên lẻ p , $R = 2^{wt}$, $p' = -p^{-1} \bmod R$, $x \in [0, p)$, $e = (e_l, e_{l-1}, \dots, e_0)_2$
Output: $x^e \bmod p$

1. $\tilde{x} \leftarrow xR \bmod p$, $A \leftarrow R \bmod p$
2. For i form l down to 0 do
 - 2.1. $A \leftarrow \text{Mont}(A, A)$
 - 2.2. If $e_l = 1$ then $A \leftarrow \text{Mont}(A, \tilde{x})$
3. Return ($\text{Mont}(A, 1)$)

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 50



TÍNH TOÁN TRÊN SỐ NGUYỄN LỚN TRONG TRƯỜNG F_p



- ✓ Giới thiệu về các trường hữu hạn
- ✓ Phép tính cộng và trừ
- ✓ Phép tính nhân
- ✓ Phép tính bình phương
- ✓ Phép lấy modulo
- ✓ Phép lũy thừa
- ✓ Phép tính nghịch đảo

Tính toán với modulo là số nguyên tố đặc biệt của NIST

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 51



Phép tính nghịch đảo

Thuật toán Euclidean mở rộng:

- Nếu $\gcd(a, b) = d$ thì phương trình bất định $ax + by = d$ có nghiệm nguyên (x, y) và một nghiệm nguyên (x, y) như vậy có thể được tính bằng thuật toán Euclidean mở rộng.
- Điều cần và đủ để có nghịch đảo là $d = 1$ và khi đó:
 - x là nghịch đảo của $a \bmod b$ và y là nghịch đảo của $b \bmod a$
- Ta mở rộng thuật toán Euclidean:
 - Tìm ước chung lớn nhất của a và b ,
 - Tính nghịch đảo trong trường hợp $\text{GCD}(a, b) = 1$.

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 52



Phép tính nghịch đảo

Thuật toán 8: Euclidean mở rộng

Input: Hai số nguyên dương a, b ($a \geq b$)

Output: $d = \gcd(a, b)$ và số nguyên x, y thỏa mãn $ax + by = d$

1. If $b = 0$ then $d \leftarrow a$, $x \leftarrow 1$, $y \leftarrow 0$ and Return(d, x, y).
2. $x_2 \leftarrow 1$, $x_1 \leftarrow 0$, $y_2 \leftarrow 0$, $y_1 \leftarrow 1$
3. While $b > 0$ do
 - 3.1. $q \leftarrow \lfloor a/b \rfloor$, $r \leftarrow a - qb$, $x \leftarrow x_2 - qx_1$, $y \leftarrow y_2 - qy_1$
 - 3.2. $a \leftarrow b$, $b \leftarrow r$, $x_2 \leftarrow x_1$, $x_1 \leftarrow x$, $y_2 \leftarrow y_1$, $y_1 \leftarrow y$
4. $d \leftarrow a$, $x \leftarrow x_2$, $y \leftarrow y_2$
5. Return(d, x, y)

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 53



Phép tính nghịch đảo

- Áp dụng thuật toán trên với các đầu vào:

- 1) $a = 1759$, $b = 550$
- 2) $a = 3458$, $b = 4864$

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 54



Phép tính nghịch đảo

q	r	x	y	a	b	x ₂	x ₁	y ₂	y ₁	⋮
-	-	-	-	1759	550	1	0	0	1	
3	109	1	-3	550	109	0	1	1	-3	
5	5	-5	16	109	5	1	-5	-3	16	
21	4	106	-339	5	4	-5	106	16	-339	
1	1	-111	335	4	1	106	-111	-339	355	
4	0	550	-1759	1	0	-111	550	355	-1759	

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 55



Phép tính nghịch đảo

Algorithm 9 Inversion in \mathbb{F}_p using the extended Euclidean algorithm

INPUT: Prime p and $a \in [1, p - 1]$.OUTPUT: $a^{-1} \bmod p$.

1. $u \leftarrow a, v \leftarrow p$.
2. $x_1 \leftarrow 1, x_2 \leftarrow 0$.
3. While $u \neq 1$ do
 - 3.1 $q \leftarrow \lfloor v/u \rfloor, r \leftarrow v - qu, x \leftarrow x_2 - qx_1$.
 - 3.2 $v \leftarrow u, u \leftarrow r, x_2 \leftarrow x_1, x_1 \leftarrow x$.
4. Return($x_1 \bmod p$).

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 56



Phép tính nghịch đảo

- ❖ Áp dụng thuật toán trên với các đầu vào:

$$\square a = 127, p = 319$$

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 57



Phép tính nghịch đảo

Algorithm 10 Binary gcd algorithm

INPUT: Positive integers a and b .OUTPUT: $\gcd(a, b)$.

1. $u \leftarrow a, v \leftarrow b, e \leftarrow 1$.
2. While both u and v are even do: $u \leftarrow u/2, v \leftarrow v/2, e \leftarrow 2e$.
3. While $u \neq 0$ do
 - 3.1 While u is even do: $u \leftarrow u/2$.
 - 3.2 While v is even do: $v \leftarrow v/2$.
 - 3.3 If $u \geq v$ then $u \leftarrow u - v$; else $v \leftarrow v - u$.
4. Return($e \cdot v$).

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 58



Phép tính nghịch đảo

Algorithm 11 Binary algorithm for inversion in \mathbb{F}_p

INPUT: Prime p and $a \in [1, p - 1]$.OUTPUT: $a^{-1} \bmod p$.

1. $u \leftarrow a, v \leftarrow p$.
2. $x_1 \leftarrow 1, x_2 \leftarrow 0$.
3. While ($u \neq 1$ and $v \neq 1$) do
 - 3.1 While u is even do
 - $u \leftarrow u/2$.
 - If x_1 is even then $x_1 \leftarrow x_1/2$; else $x_1 \leftarrow (x_1 + p)/2$.
 - 3.2 While v is even do
 - $v \leftarrow v/2$.
 - If x_2 is even then $x_2 \leftarrow x_2/2$; else $x_2 \leftarrow (x_2 + p)/2$.
 - 3.3 If $u \geq v$ then: $u \leftarrow u - v, x_1 \leftarrow x_1 - x_2$;

Else: $v \leftarrow v - u, x_2 \leftarrow x_2 - x_1$.
4. If $u = 1$ then return($x_1 \bmod p$); else return($x_2 \bmod p$).

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 59



Phép tính nghịch đảo

Algorithm 12 Partial Montgomery inversion in \mathbb{F}_p

INPUT: Odd integer $p > 2$, $a \in [1, p - 1]$, and $n = \lceil \log_2 p \rceil$.OUTPUT: Either "not invertible" or (x, k) where $n \leq k \leq 2n$ and $x = a^{-1}2^k \bmod p$.

1. $u \leftarrow a, v \leftarrow p, x_1 \leftarrow 1, x_2 \leftarrow 0, k \leftarrow 0$.
2. While $v > 0$ do
 - 2.1 If v is even then $v \leftarrow v/2, x_1 \leftarrow 2x_1$;

else if u is even then $u \leftarrow u/2, x_2 \leftarrow 2x_2$;

else if $v \geq u$ then $v \leftarrow (v - u)/2, x_2 \leftarrow x_2 + x_1, x_1 \leftarrow 2x_1$;

else $u \leftarrow (u - v)/2, x_1 \leftarrow x_2 + x_1, x_2 \leftarrow 2x_2$.
 - 2.2 $k \leftarrow k + 1$.
3. If $u \neq 1$ then return("not invertible").
4. If $x_1 > p$ then $x_1 \leftarrow x_1 - p$.
5. Return(x_1, k).

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 60



Phép tính nghịch đảo

Algorithm 13 Montgomery inversion in \mathbb{F}_p

INPUT: Odd integer $p > 2$, $n = \lceil \log_2 p \rceil$, $R^2 \bmod p$, and $\tilde{a} = aR \bmod p$ with $\gcd(a, p) = 1$.
OUTPUT: $a^{-1}R \bmod p$.

1. Use Algorithm 2.23 to find (x, k) where $x = \tilde{a}^{-1}2^k \bmod p$ and $n \leq k \leq 2n$.
2. If $k < Wt$ then
 - 2.1 $x \leftarrow \text{Mont}(x, R^2) = a^{-1}2^k \bmod p$.
 - 2.2 $k \leftarrow k + Wt$. [Now, $k > Wt$.]
3. $x \leftarrow \text{Mont}(x, R^2) = a^{-1}2^k \bmod p$.
4. $x \leftarrow \text{Mont}(x, 2^{2Wt-k}) = a^{-1}R \bmod p$.
5. Return(x).



Phép tính nghịch đảo

Algorithm 14 Simultaneous inversion

INPUT: Prime p and nonzero elements a_1, \dots, a_k in \mathbb{F}_p
OUTPUT: Field elements $a_1^{-1}, \dots, a_k^{-1}$, where $a_i a_i^{-1} \equiv 1 \pmod{p}$.

1. $c_1 \leftarrow a_1$.
2. For i from 2 to k do: $c_i \leftarrow c_{i-1} a_i \bmod p$.
3. $u \leftarrow c_k^{-1} \bmod p$.
4. For i from k down to 2 do
 - 4.1 $a_i^{-1} \leftarrow u c_{i-1} \bmod p$.
 - 4.2 $u \leftarrow u a_i \bmod p$.
5. $a_1^{-1} \leftarrow u$.
6. Return($a_1^{-1}, \dots, a_k^{-1}$).



TÍNH TOÁN TRÊN SỐ NGUYỄN LỚN TRONG TRƯỜNG \mathbb{F}_p



- Giới thiệu về các trường hữu hạn
- Phép tính cộng và trừ
- Phép tính nhân
- Phép tính bình phương
- Phép lấy modulo
- Phép lũy thừa
- Phép tính nghịch đảo

Tính toán với modulo là số nguyên tố đặc biệt của NIST



Tính toán với modulo là số nguyên tố đặc biệt của NIST

Algorithm 15 Fast reduction modulo $p_{192} = 2^{192} - 2^{64} - 1$

INPUT: An integer $c = (c_5, c_4, c_3, c_2, c_1, c_0)$ in base 2^{64} with $0 \leq c < p_{192}^2$.
OUTPUT: $c \bmod p_{192}$.

1. Define 192-bit integers:
 $s_1 = (c_5, c_1, c_0)$, $s_2 = (0, c_3, c_4)$,
 $s_3 = (c_4, c_4, 0)$, $s_4 = (c_5, c_5, c_5)$.
2. Return($s_1 + s_2 + s_3 + s_4 \bmod p_{192}$).

Algorithm 16 Fast reduction modulo $p_{224} = 2^{224} - 2^{96} + 1$

INPUT: An integer $c = (c_1, \dots, c_2, c_1, c_0)$ in base 2^{32} with $0 \leq c < p_{224}^2$.
OUTPUT: $c \bmod p_{224}$.

1. Define 224-bit integers:
 $s_1 = (c_9, c_5, c_4, c_3, c_2, c_1, c_0)$, $s_2 = (c_{10}, c_9, c_8, c_7, 0, 0, 0)$,
 $s_3 = (0, c_{13}, c_{12}, c_{11}, 0, 0, 0)$, $s_4 = (c_{13}, c_{12}, c_{11}, c_{10}, c_9, c_8, c_7)$,
 $s_5 = (0, 0, 0, 0, c_{13}, c_{12}, c_{11})$.
2. Return($s_1 + s_2 + s_3 + s_4 + s_5 \bmod p_{224}$).



Tính toán với modulo là số nguyên tố đặc biệt của NIST

Algorithm 17 Fast reduction modulo $p_{256} = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$

INPUT: An integer $c = (c_{15}, \dots, c_2, c_1, c_0)$ in base 2^{32} with $0 \leq c < p_{256}^2$.
OUTPUT: $c \bmod p_{256}$.

1. Define 256-bit integers:
 - $s_1 = (c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0)$,
 - $s_2 = (c_{15}, c_{14}, c_{13}, c_{12}, c_{11}, 0, 0, 0)$,
 - $s_3 = (0, c_{15}, c_{14}, c_{13}, c_{12}, 0, 0, 0)$,
 - $s_4 = (c_{15}, c_{14}, 0, 0, 0, c_{10}, c_9, c_8)$,
 - $s_5 = (c_8, c_{13}, c_{15}, c_{14}, c_{13}, c_{11}, c_{10}, c_9)$,
 - $s_6 = (c_{10}, c_9, 0, 0, 0, c_{13}, c_{12}, c_1)$,
 - $s_7 = (c_{11}, c_9, 0, 0, c_{15}, c_{14}, c_{13}, c_{12})$,
 - $s_8 = (c_{12}, 0, c_{10}, c_9, c_8, c_{15}, c_{14}, c_{13})$,
 - $s_9 = (c_{13}, 0, c_{11}, c_{10}, c_9, 0, c_{15}, c_{14})$.
2. Return($s_1 + s_2 + 2s_3 + s_4 + s_5 - s_6 - s_7 - s_8 - s_9 \bmod p_{256}$).



Tính toán với modulo là số nguyên tố đặc biệt của NIST

Algorithm 18 Fast reduction modulo $p_{384} = 2^{384} - 2^{138} - 2^{96} + 2^{32} - 1$

INPUT: An integer $c = (c_{23}, \dots, c_1, c_0)$ in base 2^{32} with $0 \leq c < p_{384}^2$.
OUTPUT: $c \bmod p_{384}$.

1. Define 384-bit integers:
 - $s_1 = (c_{11}, c_{10}, c_9, c_8, c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0)$,
 - $s_2 = (0, 0, 0, 0, 0, c_{23}, c_{22}, c_{21}, 0, 0, 0, 0, 0)$,
 - $s_3 = (c_{21}, c_{22}, c_{21}, c_{20}, c_{19}, c_{18}, c_{17}, c_{16}, c_{15}, c_{14}, c_{13}, c_{12})$,
 - $s_4 = (c_{20}, c_{19}, c_{18}, c_{17}, c_{16}, c_{15}, c_{14}, c_{13}, c_{12}, c_{23}, c_{22}, c_{21})$,
 - $s_5 = (c_{19}, c_{18}, c_{17}, c_{16}, c_{15}, c_{14}, c_{13}, c_{12}, c_{20}, 0, c_{23}, 0)$,
 - $s_6 = (0, 0, 0, 0, c_{23}, c_{22}, c_{21}, c_{20}, 0, 0, 0, 0)$,
 - $s_7 = (0, 0, 0, 0, 0, 0, c_{23}, c_{22}, c_{21}, 0, 0, c_{20})$,
 - $s_8 = (c_{22}, c_{21}, c_{20}, c_{19}, c_{18}, c_{17}, c_{16}, c_{15}, c_{14}, c_{13}, c_{12}, c_{23})$,
 - $s_9 = (0, 0, 0, 0, 0, 0, 0, c_{23}, c_{22}, c_{21}, c_{20}, 0, 0, 0)$,
 - $s_{10} = (0, 0, 0, 0, 0, 0, 0, 0, c_{23}, c_{22}, c_{21}, 0, 0, 0)$.
2. Return($s_1 + s_2 + s_3 + s_4 + s_5 + s_6 + s_7 - s_8 - s_9 - s_{10} \bmod p_{384}$).



Tính toán với modulo là số nguyên tố đặc biệt của NIST

HỌC VIỆN KỸ THUẬT MÃ
ACADEMY OF CRYPTOGRAPHY TECHNIQUES

Algorithm 19 Fast reduction modulo $p_{521} = 2^{521} - 1$

INPUT: An integer $c = (c_{104}, \dots, c_2, c_1, c_0)$ in base 2 with $0 \leq c < p_{521}^2$.

OUTPUT: $c \bmod p_{521}$.

1. Define 521-bit integers:
 $s_1 = (c_{104}, \dots, c_{523}, c_{522}, c_{521})$,
 $s_2 = (c_{520}, \dots, c_2, c_1, c_0)$.
2. Return($s_1 + s_2 \bmod p_{521}$).

NỘI DUNG



- 01. TÍNH TOÁN TRÊN SỐ NGUYÊN LỚN TRONG TRƯỜNG F_p**
- 02. MỘT SỐ THUẬT TOÁN VỀ SỐ NGUYÊN TỐ**
- 03. ĐỔI SÁNH MẪU TRÊN CHUỖI**

CHƯƠNG 02

MỘT SỐ THUẬT TOÁN VỀ SỐ NGUYÊN TỐ



Chương 2 – Mục tiêu

- ❖ Nắm được các kiến thức cơ bản về số nguyên tố, số giả nguyên tố
- ❖ Hiểu và lập trình được một số thuật toán:
 - Sàng nguyên tố
 - Phân tích một số nguyên thành thừa số nguyên tố
 - Kiểm tra một số có là nguyên tố không
 - Sinh số nguyên tố



Bài 01 – Mục tiêu

- ❖ Nắm được khái niệm số nguyên tố
- ❖ Nắm được kiến thức cơ bản về sàng số nguyên tố Eratosthenes nguyên thủy và sàng phân đoạn
- ❖ Nắm được bài toán phân tích một số nguyên ra thừa số nguyên tố
- ❖ Hiểu và lập trình được hai thuật toán về sàng Eratosthenes và sàng phân đoạn
- ❖ Hiểu và lập trình được thuật toán phân tích một số ra thừa số nguyên tố Pollard's Rho

Bài 01 – Nội dung

- ❖ Kiến thức chung
- ❖ Sàng Eratosthenes nguyên thủy
- ❖ Sàng Eratosthenes phân đoạn
- ❖ Thuật toán Pollard's Rho



MỘT SỐ THUẬT TOÁN VỀ SỐ NGUYÊN TỐ



Sàng Eratosthenes

- Phân tích ra thừa số nguyên tố
- Kiểm tra tính nguyên tố
- Sinh số nguyên tố



Kiến thức chung

- ❖ Số nguyên tố:
 - Là số tự nhiên có đúng 2 ước số tự nhiên là 1 và chính nó
- ❖ Sàng Eratosthenes
 - Là một thuật toán cổ đại để tìm tất cả các số nguyên tố nhỏ hơn hoặc bằng một số nguyên cho trước



Sàng Eratosthenes nguyên thủy

- ❖ Ý tưởng:
 - Bắt đầu với số nguyên tố đầu tiên là 2
 - Sinh tất cả các bội của số nguyên tố đã cho (nhỏ hơn số nguyên cho trước) với hiệu số cố định giữa các số bằng số nguyên tố đó
 - Đánh dấu tất cả các bội của mỗi số nguyên tố là hợp số
 - Các số còn lại đánh dấu là số nguyên tố



Sàng Eratosthenes nguyên thủy

- ❖ Các bước thực hiện:
 - Bước 1: Liệt kê các số nguyên liên tiếp từ 2 tới n ($2, 3, 4, \dots, n$)
 - Bước 2: Khởi tạo $p = 2$
 - Bước 3: Liệt kê các bội số của p bằng cách đếm các số giả của p từ $2p, 3p, 4p, \dots$ tới n ; đánh dấu là hợp số



Sàng Eratosthenes nguyên thủy

- ❖ Các bước thực hiện (..)
 - Bước 4: Tìm số nhỏ nhất trong danh sách, lớn hơn p mà không bị đánh dấu
 - Nếu không có số nào thì dừng lại
 - Ngược lại, gán p bằng số vừa tìm được và lặp lại bước 3
 - Bước 5: Kết thúc thuật toán, các số còn lại trong danh sách không bị đánh dấu là tất cả các số nguyên tố nhỏ hơn hoặc bằng n



Sàng Eratosthenes nguyên thủy

- ❖ VD: Tìm các số nguyên tố ≤ 30
 - ❑ Liệt kê các số nguyên từ 2 tới 30

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
 - ❑ $p = 2$, loại bỏ các bội của 2

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
 - ❑ $p = 3$, loại bỏ các bội của 3

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 79



Sàng Eratosthenes nguyên thủy

- ❖ VD: Tìm các số nguyên tố ≤ 30 (..)
 - ❑ $p = 5$, loại bỏ các bội của 5

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
 - ❑ $p = 7$, loại bỏ các bội của 7, tuy nhiên các bội đó (14, 21, 28) đều đã bị loại

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
 - ❑
 - ❑ Tất cả các số còn lại đều là nguyên tố

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 80



Sàng Eratosthenes nguyên thủy

- ❖ Nhận xét:
 - ❑ Thuật toán duyệt toàn bộ mảng chứa chuỗi các số không lớn hơn n mà không hiển thị vị trí tham chiếu.
 - ❑ Yêu cầu bộ nhớ lớn
 - Với n lớn, bộ nhớ có thể không đáp ứng đủ dãy số nguyên tố
 - Với n vừa phải, việc sử dụng bộ nhớ cache của nó là không tối ưu
 - ❑ => Sàng phân đoạn ra đờি

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 81



Sàng Eratosthenes phân đoạn

- ❖ Được biết đến từ những năm 1970
- ❖ Mỗi lần chỉ sàng các phần trong phạm vi



Sàng Eratosthenes phân đoạn

- ❖ Thuật toán:
 - ❑ Bước 1: Chia phạm vi từ 2 tới n thành các đoạn có kích cỡ Δ nào đó, với $\Delta \leq \sqrt{n}$
 - ❑ Bước 2: Sử dụng sàng Eratosthenes để tìm các số nguyên tố trong đoạn đầu tiên

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 83



Sàng Eratosthenes phân đoạn

- ❖ Thuật toán: (..)
 - ❑ Bước 3: Theo thứ tự tăng dần, với mỗi đoạn tiếp theo tìm các số nguyên tố như sau, trong đó m là giá trị lớn nhất của đoạn
 - Bước 3.1: Thiết lập một mảng Boolean có kích thước là Δ

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 84



Sàng Eratosthenes phân đoạn

❖ Thuật toán: (..)

- Bước 3.2: Đánh dấu các vị trí trong mảng không là nguyên tố ứng với các bội của mỗi số nguyên tố $p \leq \sqrt{m}$ đã tìm được bằng cách tính bội nhỏ nhất của p trong khoảng $m - \Delta$ và m , và liệt kê các bội của nó theo các bước của p như bình thường. Các vị trí còn lại tương ứng với các số nguyên tố trong đoạn vi binh phương của một số nguyên tố trong đoạn không thuộc đoạn đó (với $k \geq 1$, ta có $(k\Delta + 1)^2 > (k + 1)\Delta$)



MỘT SỐ THUẬT TOÁN VỀ SỐ NGUYÊN TỐ



Sàng Eratosthenes

Phân tích ra thừa số nguyên tố

Kiểm tra tính nguyên tố

Sinh số nguyên tố



Kiến thức chung

❖ Bài toán

- Cho số nguyên dương n, hãy phân tích n ra thừa số nguyên tố, tức là $n = p_1^{e_1} \cdot p_2^{e_2} \dots p_k^{e_k}$, trong đó p_i là các số nguyên tố đôi một khác nhau và $e_i \geq 1$



Kiến thức chung

- Ví dụ: Cho $N = 408.508.091$, tìm số nguyên tố p, q:

$$p \times q = 408.508.091$$

- Với máy tính cầm tay \Rightarrow mất bao lâu để có được p, q?
 - Kiểm tra mỗi số nguyên tố xem có là ước của N hay không? Ví dụ: 3, 5, ..., cho tới $p = 18.313$ (số nguyên tố thứ 2000) thì thấy 18.313 thực sự là thừa số của $408.508.091$, như vậy dễ dàng xác định được số $q = 22.307$.
 - Một máy tính kiểm tra 4 số nguyên tố/1 phút \Rightarrow mất 500 phút \Leftrightarrow **hơn 8 giờ** để tìm ra p, q
- Nếu biết trước giá trị $p = 18.313$ và $q = 22.307 \Rightarrow$ mất chưa tới **10s** để tính ra N



Kiến thức chung

❖ Lưu ý:

- Bài toán quyết định liệu một số nguyên là hợp số hay là nguyên tố nói chung dễ hơn nhiều so với bài toán phân tích ra thừa số
- \Rightarrow Trước khi phân tích n ra thừa số, nên kiểm tra để đảm bảo n thực sự là hợp số



Kiến thức chung

- Định nghĩa 2.2.1:** (Thừa số không tầm thường)

Phân tích ra thừa số không tầm thường của n là một dạng $n=a.b$, trong đó $1 < a, b < n$. Khi đó a, b được gọi là các thừa số không tầm thường.



Kiến thức chung

Nhận xét:

- Trong định nghĩa 2.2.1, a,b không nhất thiết phải là số nguyên tố
- Khi phân tích n ra thừa số nguyên tố, có thể thực hiện:
 - Bước 1: Sử dụng các thuật toán phân tích n thành thừa số
 - Bước 2: Kiểm tra tính nguyên tố của a và b
 - Bước 3: Nếu a hoặc b là hợp số thì tiếp tục áp dụng thuật toán phân tích ra thừa số.



Kiến thức chung

- VD: Áp dụng thuật toán trên phân tích n = 4.337.800 thành tích của các thừa số nguyên tố
 - $4337800 = 2 \times 2168900$ (a = 2, b = 2168900 có là nguyên tố?)
 - $4337800 = 2 \times (2 \times 1084450)$ (kiểm tra 2, 1084450 có là nguyên tố?)
 - $4337800 = 2 \times (2 \times (2 \times 542225))$ (kiểm tra 2, 542225 có là nguyên tố?)
 - $4337800 = 2 \times (2 \times (2 \times (5 \times 108445)))$ (kiểm tra 5, 108445 có là nguyên tố?)
 -

Vậy n được phân tích thành tích:
 $4337800 = 2^3 \times 5^2 \times 23^2 \times 41$



Kiến thức chung

Định nghĩa 2.2.2: Lũy thừa hoàn hảo

- Số nguyên n được gọi là một lũy thừa hoàn hảo nếu $n = x^k$ với x, k là các số nguyên thỏa mãn $x \geq 2, k \geq 2$
- **Chú ý:** Kiểm tra lũy thừa hoàn hảo (perfect power)
 - Nếu $n \geq 2$ có thể kiểm tra xem liệu n có là một lũy thừa hoàn hảo không.
 - Bài toán phân tích n ra thừa số nguyên tố luôn giả thiết n không phải là một lũy thừa hoàn hảo, tức là có ít nhất 2 thừa số nguyên tố khác nhau



Kiến thức chung

Phép chia thử:

- Với n là hợp số, trước khi áp dụng các thuật toán phân tích n thành thừa số nguyên tố nên thực hiện phép chia thử n cho tất cả các số nguyên tố “nhỏ”



Thuật toán Pollard's Rho

- Cho $f: S \rightarrow S$ là một hàm ngẫu nhiên, trong đó S là một tập hữu hạn các ứng cử n.
- Cho x_0 là một phần tử ngẫu nhiên của S, xét chuỗi x_0, x_1, x_2, \dots được xác định bởi $x_{i+1} = f(x_i)$, với $i \geq 0$
 - Vì S là một tập hữu hạn nên chuỗi phải có chu kỳ
 - => Tồn tại các chỉ số $i \neq j$, sao cho $x_i = x_j$ (gọi là xảy ra xung đột)



Thuật toán Pollard's Rho

- Vấn đề chuỗi có chu kỳ liên quan tới một số tần công thâm mă, bao gồm phân tích số nguyên ra thừa số là tìm các chỉ số $i \neq j$, sao cho $x_i = x_j$



Thuật toán Pollard's Rho

- ❖ **Thuật toán Floyd:** (Tim chu kì)

- **Bước 1:** Bắt đầu với cặp (x_1, x_2)

- **Bước 2:** Tính lặp đi lặp lại (x_i, x_{2i}) từ cặp (x_{i-1}, x_{2i-2}) cho tới khi $x_m = x_{2m}$, với giá trị m nào đó



Thuật toán Pollard's Rho

- ❖ Cho p là thừa số nguyên tố của hợp số nguyên n . Thuật toán Pollard's Rho phân tích n thành thừa số nguyên tố sẽ tìm sự lặp lại trong chuỗi các số nguyên x_0, x_1, x_2, \dots được xác định bởi $x_0 = 2$, $x_{i+1} = f(x_i) = x_i^2 + 1 \text{ mod } p$ với $i \geq 0$
- Sử dụng thuật toán Floyd tìm x_m và x_{2m} sao cho $x_m \equiv x_{2m} \pmod{p}$



Thuật toán Pollard's Rho

- Vì n chia hết cho p nhưng chưa biết p nên tính $x_i \text{ mod } n$ và kiểm tra liệu $\gcd(x_m - x_{2m}, n) > 1$ không. Nếu lại có $\gcd(x_m - x_{2m}, n) < n$ thì thu được thừa số không tầm thường của n .



Thuật toán Pollard's Rho

- ❖ **Mục tiêu:** Tìm các thừa số nhỏ của một hợp số
- ❖ **Đầu vào:** n là hợp số nhưng n không phải là lũy thừa của một số nguyên tố



Thuật toán Pollard's Rho

- ❖ **Bước 1:** Đặt $a=2$; $b=2$

- ❖ **Bước 2:** For $i=1, 2, \dots$ do

- **Bước 2.1:** Tính $a = a^2 + 1 \text{ mod } n$; $b = b^2 + 1 \text{ mod } n$; $b = b^2 + 1 \text{ mod } n$
 - **Bước 2.2:** Tính $d = \gcd(a - b, n)$
 - **Bước 2.3:** If $1 < d < n$ then return(d) và kết thúc với thành công
 - **Bước 2.4:** If $d = n$ then kết thúc với thất bại



Thuật toán Pollard's Rho

- ❖ ...
- ❖ Thuật toán tìm $\gcd(a, b)$:
- Input: a, b
 - Ouput: $\gcd(a, b)$
 - 1. $A=a$, $B=b$
 - 2. while $B>0$
 - $R = A \text{ mod } B$
 - $A = B$, $B = R$
 - 3. Return A



Thuật toán Pollard's Rho

- ❖ VD: Tính GCD(1970, 1066)?
 - ❖ $1970 = 1 \times 1066 + 904 \quad \text{gcd}(1970, 1066)$
 - ❖ $1066 = 1 \times 904 + 162 \quad \text{gcd}(904, 162)$
 - ❖ $904 = 5 \times 162 + 94 \quad \text{gcd}(162, 94)$
 - ❖ $162 = 1 \times 94 + 68 \quad \text{gcd}(94, 68)$
 - ❖ $94 = 1 \times 68 + 26 \quad \text{gcd}(68, 26)$
 - ❖ $68 = 2 \times 26 + 16 \quad \text{gcd}(26, 16)$
 - ❖ $26 = 1 \times 16 + 10 \quad \text{gcd}(16, 10)$
 - ❖ $16 = 1 \times 10 + 6 \quad \text{gcd}(10, 6)$
 - ❖ $10 = 1 \times 6 + 4 \quad \text{gcd}(6, 4)$
 - ❖ $6 = 1 \times 4 + 2 \quad \text{gcd}(4, 2)$
 - ❖ $4 = 2 \times 2 + 0$
- gcd(1970, 1066) = 2**

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 103



Thuật toán Pollard's Rho

- ❖ Ví dụ: Tìm thừa số không tầm thường của $n = 455459$ sử dụng thuật toán Pollard's Rho

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 104



Thuật toán Pollard's Rho

- ❖ Chú ý:
- ❑ Thuật toán kết thúc thất bại thì có thể thử lại với hàm khác thay vì $f(x) = x^2 + 1$
- ❑ Chẳng hạn có thể chọn $f(x) = x^2 + c$ với $c \neq 0; -2$

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 105



Bài 02 – Mục tiêu

- ❖ Hiểu và lập trình được một số thuật toán kiểm tra một số có là nguyên tố hay không
- ❑ Thuật toán kiểm tra Fermat
- ❑ Thuật toán kiểm tra xác suất Miller–Rabin
- ❖ Hiểu và lập trình được thuật toán sinh số nguyên tố

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 106



Bài 02 – Nội dung

- ❖ Kiến thức chung
- ❖ Thuật toán kiểm tra Fermat
- ❖ Thuật toán kiểm tra xác suất Miller–Rabin
- ❖ Thuật toán sinh số nguyên tố

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 107



MỘT SỐ THUẬT TOÁN VỀ SỐ NGUYÊN TỐ



- ✓ Sàng Eratosthenes

- ✓ Phân tích ra thừa số nguyên tố

- ✓ Kiểm tra tính nguyên tố

- ✓ Sinh số nguyên tố

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 108



Kiến thức chung

- Thuật toán kiểm tra Fermat không thực sự là kiểm tra số nguyên tố dựa trên xác suất vì nó thường thất bại trong việc phân biệt số nguyên tố và một loại hợp số đặc biệt được gọi là số Carmichael

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 109



Kiến thức chung

- Các kiểm tra tính nguyên tố theo xác suất có khuôn mẫu sau:
 - Với mỗi số nguyên dương lẻ n , tập $W(n) \subset Z_n$ được xác định thỏa mãn những tính chất sau:
 - Với $a \in Z_n$, có thể kiểm tra xem liệu $a \in W(n)$ không với thời gian đa thức
 - Nếu n là nguyên tố, thì $W(n) = \emptyset$
 - Nếu n là hợp số, thì $\#W(n) \geq \frac{n}{2}$

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 110

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 111

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 112



Kiến thức chung

- Nếu thực sự $a \in W(n)$ thì n được gọi là **thất bại** với kiểm tra tính nguyên tố đối với cơ sở a
 - \Rightarrow chắc chắn n là hợp số
- Nếu $a \notin W(n)$ thì n được gọi là **qua** với kiểm tra tính nguyên tố đối với cơ sở a
 - \Rightarrow chưa thể khẳng định chắc chắn n là nguyên tố

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 113

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 114



Kiến thức chung

- Định nghĩa 2.3.2:** Một số nguyên n được tin là số nguyên tố theo kiểm tra tính nguyên tố dựa trên xác suất được gọi là số nguyên tố có thể có (có thể là nguyên tố)



Thuật toán kiểm tra Fermat

- ❖ **Định lí Fermat:** Cho n là một số nguyên tố
 - Nếu $\gcd(a, n) = 1$ thì $a^{n-1} \equiv 1 \pmod{n}$
- ❖ **Định nghĩa 2.3.3:** Cho n là một hợp số nguyên lẻ. Một số nguyên a , $1 \leq a \leq n - 1$ thỏa mãn $a^{n-1} \not\equiv 1 \pmod{n}$ được gọi là bằng chứng Fermat chứng tỏ n là hợp số.



Thuật toán kiểm tra Fermat

- ❖ **Định nghĩa 2.3.4:** Cho n là một hợp số nguyên lẻ và cho a là một số nguyên, $1 \leq a \leq n - 1$. Thì n được gọi là một số giả nguyên tố với cơ sở a nếu $a^{n-1} \equiv 1 \pmod{n}$. Số nguyên a được gọi là một giá trị đánh lừa cho tính nguyên tố của n .
 - VD: $n = 341 (= 11 \times 31)$ là một số giả nguyên tố đối với cơ sở $a = 2$ vì $2^{340} \equiv 1 \pmod{341}$



Thuật toán kiểm tra Fermat

- ❖ **Thuật toán FERMAT(n, t):** Kiểm tra xem liệu n có là số nguyên tố?
 - Đầu vào: n là số nguyên lẻ, $n \geq 3$ và tham số an toàn $t \geq 1$
 - Đầu ra: Hoặc “nguyên tố” hoặc “hợp số”



Thuật toán kiểm tra Fermat

- ❖ **Thuật toán FERMAT(n, t):** Kiểm tra xem liệu n có là số nguyên tố?
 - **Bước 1:** For $i = 1$ to t do
 - **Bước 1.1:** Chọn ngẫu nhiên số nguyên a , $2 \leq a \leq n - 2$
 - **Bước 1.2:** Sử dụng thuật toán nhân bình phương có lặp tính $r = a^{n-1} \pmod{n}$
 - **Bước 1.3:** Nếu $r \neq 1$ thi return (“Hợp số”)
 - **Bước 2:** Return (“Nguyên tố”)



Thuật toán kiểm tra Fermat

- ❖ **Nhận xét:**
 - Kết quả của thuật toán FERMAT(n, t) là hợp số thì chắc chắn n là hợp số
 - Ngược lại, không có bằng chứng chứng tỏ n thật sự là nguyên tố



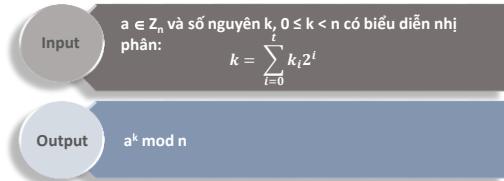
Thuật toán kiểm tra Fermat

- ❖ **Nhận xét: (..)**
 - Tuy nhiên, vì các số giả nguyên tố đối với cơ sở a được biết là rất hiếm nên kiểm tra Fermat cung cấp câu trả lời đúng cho hầu hết các đầu vào
 - Nhưng không có nghĩa là cung cấp câu trả lời đúng cho hầu hết các lần trên mọi đầu vào



Thuật toán kiểm tra Fermat

- Thuật toán nhân bình phương có lặp

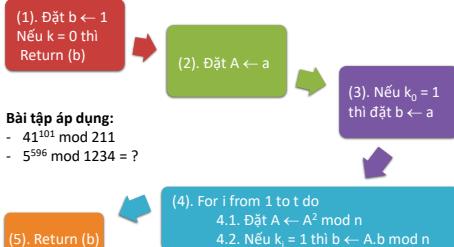


Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 121



Thuật toán nhân bình phương có lặp



Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 122



Thuật toán nhân bình phương có lặp

- Giải:

- Ta phân tích $101 = 2^6 + 2^5 + 2^2 + 2^0$. Áp dụng phương pháp nhân và bình phương có lặp ta có bảng giá trị sau:

i	0	1	2	3	4	5	6
k_i	1	0	1	0	0	1	1
A	41	204	49	80	70	47	99
b	41	404	110	110	110	106	155

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 123



Thuật toán kiểm tra Fermat

- BT áp dụng thuật toán Fermat kiểm tra số $n = 383$ có là số nguyên tố hay ko? (cho $t = 2$)



Thuật toán kiểm tra Fermat

- Định nghĩa 2.3.5:** Số Carmichael

- Số Carmichael n là một hợp số nguyên thỏa mãn $a^{n-1} \equiv 1 \pmod{n}$ với tất cả các số nguyên a thỏa mãn $\gcd(a, n) = 1$

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 125



Thuật toán kiểm tra Fermat

- Nếu n là số Carmichael thì bằng chứng Fermat duy nhất cho n là các số nguyên a , $1 \leq a \leq n - 1$, mà $\gcd(a, n) > 1$
- Do vậy nếu các thừa số nguyên tố của n đều lớn thì kiểm tra Fermat trả về kết quả n là nguyên tố với xác suất cao ngay cả khi số lần lặp t là lớn
- Kiểm tra xác suất Solovay-Strassen và Miller-Rabin khắc phục được nhược điểm này của kiểm tra Fermat

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 126



Thuật toán kiểm tra Fermat

- ❖ **Khẳng định 2.3.1:** Điều kiện cần và đủ của số Carmichael
 - Một hợp số n là số Carmichael khi và chỉ khi 2 điều kiện sau thỏa mãn:
 - (i) n không là bình phương của một số, chẳng hạn n không chia hết cho bình phương của bất kỳ số nguyên tố nào
 - (ii) $n - 1$ chia hết cho $p - 1$ với mọi ước p của n
- ❖ **Hệ quả:** Mọi số Carmichael đều là tích của ít nhất 3 số nguyên tố khác nhau



Thuật toán kiểm tra xác suất Miller–Rabin

- ❖ **Khẳng định 2.3.2:** Cho x, y và n là các số nguyên. Nếu $x^2 \equiv y^2 \pmod{n}$ nhưng $x \not\equiv \pm y \pmod{n}$ thì $\gcd(x - y, n)$ là thừa số không tần thường của n .



Thuật toán kiểm tra xác suất Miller–Rabin

- ❖ **Khẳng định 2.3.3:** Cho n là một số nguyên tố lẻ, và cho $n - 1 = 2^s r$, trong đó r là một số lẻ. Cho a là một số nguyên bất kì thỏa mãn $\gcd(a, n) = 1$. Thị với j nào đó, $0 \leq j \leq s - 1$:
 - Hoặc $a^r \equiv 1 \pmod{n}$
 - Hoặc $a^{2^j r} \equiv -1 \pmod{n}$



Thuật toán kiểm tra xác suất Miller–Rabin

- ❖ **Định nghĩa 2.3.6:** Cho n là một hợp số nguyên lẻ và cho $n - 1 = 2^s r$, trong đó r là lẻ. Cho a là một số nguyên trong đoạn $[1, n - 1]$.
 - (i) Nếu $a^r \not\equiv 1 \pmod{n}$ và nếu $a^{2^j r} \not\equiv -1 \pmod{n}$ với tất cả j , $0 \leq j \leq s - 1$ thì a được gọi là bằng chứng mạnh chứng tỏ n là hợp số.



Thuật toán kiểm tra xác suất Miller–Rabin

- ❖ **Định nghĩa 2.3.6: (..)**
 - (ii) Ngược lại, chẳng hạn nếu hoặc $a^r \equiv 1 \pmod{n}$ hoặc $a^{2^j r} \equiv -1 \pmod{n}$ với j nào đó, $0 \leq j \leq s - 1$, thì n được gọi là số giả nguyên tố mạnh đối với cơ sở a (Tức là n hoạt động như một số nguyên tố). Số nguyên a được gọi là giá trị đánh lừa mạnh cho tính nguyên tố của n .



Thuật toán kiểm tra xác suất Miller–Rabin

- ❖ VD: Số giả nguyên tố mạnh
 - Xét hợp số nguyên $n = 91 (= 7 \times 13)$. Vì $91 - 1 = 90 = 2 \times 45$ $\Rightarrow s = 1$ và $r = 45$.
 - Vì $9^r = 9^{45} \equiv 1 \pmod{91}$ nên 91 là một số giả nguyên tố mạnh đối với cơ sở 9
 - Tập tất cả các giá trị đánh lừa mạnh của 91 là: $\{1, 9, 10, 12, 16, 17, 22, 29, 38, 53, 62, 69, 74, 75, 79, 81, 82, 90\}$



Thuật toán kiểm tra xác suất Miller–Rabin

- ❖ **Thuật toán MILLER-RABIN(n,t):** Kiểm tra xem liệu n có là số nguyên tố?
- ❑ Đầu vào: Một số nguyên lẻ $n \geq 3$ và tham số an toàn $t \geq 1$
- ❑ Đầu ra: Hoặc “nguyên tố” hoặc “hợp số”



Thuật toán kiểm tra xác suất Miller–Rabin

- ❖ **Thuật toán MILLER-RABIN(n,t):** Kiểm tra xem liệu n có là số nguyên tố?
 - ❑ **Bước 1:** Viết $n - 1 = 2^s r$ để r là lẻ
 - ❑ **Bước 2:** For $i = 1$ to t do
 - **Bước 2.1:** Chọn ngẫu nhiên một số nguyên a , $2 \leq a \leq n - 2$
 - **Bước 2.2:** Sử dụng thuật toán nhân bình phương có lặp tinh $y = a^r \bmod n$



Thuật toán kiểm tra xác suất Miller–Rabin

- ❖ **Thuật toán MILLER-RABIN(n,t):** Kiểm tra xem liệu n có là số nguyên tố?
 - **Bước 2.3:** Nếu $y \neq 1$ và $y \neq n - 1$ thì
 - $j = 1$
 - While $j \leq s - 1$ và $y \neq n - 1$ do
 - Tính $y = y^2 \bmod n$
 - Nếu $y = 1$ thi return(“hợp số”)
 - $j = j + 1$
 - Nếu $y \neq n - 1$ thi return(“hợp số”)
 - ❑ **Bước 3:** Return (“nguyên tố”)



Thuật toán kiểm tra xác suất Miller–Rabin

- ❖ **Nhận xét:**
 - ❑ Thuật toán MILLER-RABIN(n,t) kiểm tra xem liệu với mỗi cơ sở a có thỏa mãn định nghĩa 2.3.6 không?
 - ❑ Dòng lệnh 5 của bước 2.3 nếu $y = 1$ thì $a^{2^j r} \equiv 1 \pmod{n}$. Vì đây cũng là trường hợp mà $a^{2^{j-1} r} \not\equiv \pm 1 \pmod{n}$ nên nó tuân theo Khẳng định 2.3.2 mà n là hợp số (thực tế là $\gcd(a^{2^j r} - 1, n)$ là thừa số không tầm thường của n)



Thuật toán kiểm tra xác suất Miller–Rabin

- ❖ **Nhận xét:** (..)
 - ❑ Dòng lệnh thứ 7 của bước 2.3, nếu $y \neq n - 1$ thì a là bằng chứng mạnh đối với n .
 - ❑ Nếu thuật toán MILLER-RABIN(n,t) trả về kết quả là “hợp số” thì n chắc chắn là hợp số vì các số nguyên tố không được vi phạm Khẳng định 2.3.3
 - ❑ Nếu n **thực sự** là “nguyên tố” thì thuật toán luôn trả lại kết quả là “nguyên tố”



Thuật toán kiểm tra xác suất Miller–Rabin

- ❖ **Nhận xét:** (..)
 - ❑ ...
 - ❑ Người ta chứng minh được rằng xác suất để số giả nguyên tố đó không là số nguyên tố là $\frac{1}{4}$. Suy ra nếu lặp t phép thử với các lựa chọn ngẫu nhiên khác nhau của số a , thì khi đó xác suất để số n sau t phép thử là số nguyên tố là: $1 - (1/4)^t$
 - **Ví dụ:** Sau 10 bước, $t = 10$, mà số đã cho n đều có thể là nguyên tố, thì xác suất để n là số nguyên tố là $1 - (1/4)^{10} > 0.99999$.



Thuật toán kiểm tra xác suất Miller–Rabin

- VD: áp dụng thuật toán Miller Rabin kiểm tra $n = 383$ có là nguyên tố hay không?



Thuật toán kiểm tra xác suất Miller–Rabin

- BTVN: Áp dụng thuật toán Miller – Rabin kiểm tra $n = 57$ có là nguyên tố?



MỘT SỐ THUẬT TOÁN VỀ SỐ NGUYÊN TỐ



Sàng Eratosthenes

Phân tích ra thừa số nguyên tố

Kiểm tra tính nguyên tố

Sinh số nguyên tố



Kiến thức chung

- Bài toán sinh số nguyên tố khác với bài toán kiểm tra tính nguyên tố của một số, nhưng thường liên quan đến bài toán kiểm tra tính nguyên tố
- Bài toán sinh số nguyên tố cho phép xây dựng các ứng cử của một dạng cố định (được sửa – fixed form), dạng này có thể dẫn tới kiểm tra hiệu quả cao so với các ứng cử ngẫu nhiên



Kiến thức chung

- Định lí 2.4.1:** (Định lí số nguyên tố)

Cho $\pi(x)$ kí hiệu là số các số nguyên tố $\leq x$, thì $\lim_{x \rightarrow \infty} \frac{\pi(x)}{x/\ln x} = 1$

- Nhận xét:

- Từ định lí 2.4.1 suy ra tỉ lệ số nguyên dương $\leq x$ là nguyên tố xấp xỉ là $1/\ln x$
 - Vì một nửa các số nguyên $\leq x$ là chẵn nên tỉ lệ các số nguyên lẻ $\leq x$ là nguyên tố xấp xỉ là $2/\ln x$



Kiến thức chung

- => Chiến lược chọn một số nguyên tố có thể k-bit ngẫu nhiên là chọn lặp đi lặp lại một cách ngẫu nhiên các số nguyên lẻ k-bit n cho đến khi được một số n mà thuật toán MILLER-RABIN(n,t) cho kết quả là nguyên tố với một tham số an toàn t thích hợp.



Kiến thức chung

- ❖ Vì xác xuất một số nguyên ngẫu nhiên n có một ước nguyên tố nhỏ là tương đối lớn nên trước khi áp dụng kiểm tra Miller-Rabin nên chia thử n với các số nguyên tố nhỏ hơn một giới hạn B được xác định trước, có thể thực hiện:
 - ❑ Chia n cho tất cả các số nguyên tố nhỏ hơn B
 - ❑ Hoặc tính ước chung lớn nhất của n và các tích của một vài các số nguyên tố $\leq B$

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 145



Thuật toán sinh số nguyên tố

- ❖ **Thuật toán RANDOM-SEARCH(k, t):** Tim kiếm ngẫu nhiên một số nguyên tố sử dụng kiểm tra Miller-Rabin
- ❑ **Đầu vào:** Một số nguyên k , tham số an toàn t
- ❑ **Đầu ra:** Một số nguyên tố có thể k -bit

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 146



Thuật toán sinh số nguyên tố

- ❖ **Thuật toán RANDOM-SEARCH(k, t):** (..)
- ❑ **Bước 1:** Sinh ngẫu nhiên một số nguyên n k -bit
- ❑ **Bước 2:** Sử dụng phép chia thử để xác định liệu n có chia hết cho một số nguyên tố bất kì $\leq B$ không.
 - Nếu n chia hết thì quay lại Bước 1
- ❑ **Bước 3:** Nếu MILLER-RABIN(n, t) trả về kết quả “nguyên tố” thì return(n). Ngược lại, quay lại Bước 1

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 147



NỘI DUNG

- 01.** TÍNH TOÁN TRÊN SỐ NGUYÊN LỚN TRONG TRƯỜNG F_p
- 02.** MỘT SỐ THUẬT TOÁN VỀ SỐ NGUYÊN TỐ
- 03.** ĐỔI SÁNH MẪU TRÊN CHUỖI



Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

HỌC VIỆN KỸ THUẬT MÃ MÃ
ACADEMY OF CRYPTOGRAPHY TECHNIQUES

CHƯƠNG 03

ĐỔI SÁNH MẪU TRÊN CHUỖI

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 149



ĐỔI SÁNH MẪU TRÊN CHUỖI



Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 150

- ❑ Bài toán đổi sánh mẫu
- ❑ Thuật toán đổi sánh vét cạn
- ❑ Thuật toán Boyer-Moore
- ❑ Thuật toán Knuth-Morris-Pratt
- ❑ Thuật toán Wu-Manber
- ❑ Thuật toán Aho-Corasick



Đối sánh mẫu là gì?

- ❖ Định nghĩa: Cho một văn bản T và một mẫu chuỗi ký tự cần tìm P, hãy tìm kiếm sự xuất hiện của mẫu P trong văn bản T
- ❖ $T = T_0T_1T_2\dots T_{n-1}$, $|T|=n$ (độ dài của văn bản)
- ❖ $P = P_0P_1P_2\dots P_{m-1}$, $|P|=m$ (độ dài của mẫu đối sánh)
- ❖ Σ – bảng chữ cái với $|\Sigma| = c$

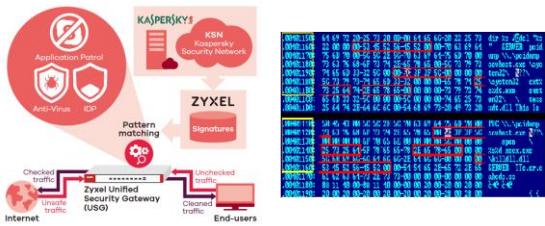


Ví dụ

- ❖ Ví dụ:
 - T: “the rain in Viet Nam stays mainly on the plain”
 - P: “n th”
- ❖ Ứng dụng:
 - Các trình text editors, dịch vụ tìm kiếm (e.g. Google),...



Vai trò của đối sánh mẫu trong ATTT



String và Substring

- ❖ S là một chuỗi ký tự có kích thước là n.
- ❖ Một **chuỗi con (substring)** $S[i \dots j]$ của S là một phần của chuỗi ký tự S bắt đầu từ chỉ số i và kết thúc tại j, $0 \leq i \leq j \leq n - 1$
- ❖ với $i \in [0, n - 1]$ ta có:
 - Tiền tố (prefix) của S là chuỗi con $S[0 \dots i]$
 - Hậu tố (suffix) của S là chuỗi con $S[i \dots n-1]$
- ❖ Ví dụ:

S

a	n	t	o	a	n
0					5



String và Substring (Ví dụ)

S

a	n	t	o	a	n
0					5

- ❖ Substring $S[1..3] == "nto"$
- ❖ Tất cả các tiền tố có thể có của S:
 - “antoan”, “antoa”, “anto”, “ant”, “an”, “a”
- ❖ Tất cả các hậu tố có thể có của S:
 - “antoan”, “ntoan”, “toan”, “oan”, “an”, “n”



Các thuật toán đối sánh mẫu

- | | |
|---|---|
| <p>Đơn mẫu:</p> <ul style="list-style-type: none"> ❖ Vét cạn ❖ Knuth–Morris–Pratt (KMP) ❖ Karp–Rabin ❖ Boyer–Moore ❖ Horspool ❖ Shift-OR, Shift-AND ❖ Factor searches | <p>Đa mẫu:</p> <ul style="list-style-type: none"> ❖ Wu – Manber ❖ Commentz – Walter ❖ Aho–Corasick <p>Indexing:</p> <ul style="list-style-type: none"> ❖ Trie (và suffix trie) ❖ Suffix tree |
|---|---|



ĐỔI SÁNH MẪU TRÊN CHUỖI

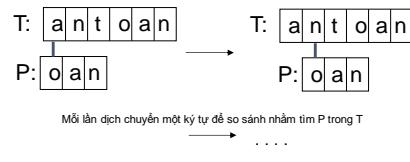


- Bài toán đổi sánh mẫu
- Thuật toán đổi sánh vét cạn
- Thuật toán Boyer-Moore
- Thuật toán Knuth-Morris-Pratt
- Thuật toán Wu-Manber
- Thuật toán Aho-Corasick



Thuật toán vét cạn

- ❖ Kiểm tra mỗi vị trí của đoạn văn bản T để tìm sự xuất hiện đầu tiên của mẫu P tại vị trí đó



Phân tích độ phức tạp tính toán

- ❖ Độ phức tạp thời gian O(mn) trong trường hợp tệ nhất.
- ❖ Tuy nhiên trong trường hợp trung bình độ phức tạp: O(m+n).
- ❖ Hiệu quả trong trường hợp bảng chữ cái lớn:
 - Ví dụ. A..Z, a..z, 1..9, ...
- ❖ Không hiệu quả trong trường hợp bảng chữ cái nhỏ
 - Ví dụ. 0, 1 (chẳng hạn file nhị phân, file ảnh...)



Phân tích độ phức tạp tính toán

- ❖ Ví dụ về trường hợp tồi nhất:
 - T: "aaaaaaaaaaaaaaaaaaaaaaaaah"
 - P: "aaah"
- ❖ Ví dụ về trường hợp trung bình:
 - T: "a string searching example is standard"
 - P: "example"



ĐỔI SÁNH MẪU TRÊN CHUỖI

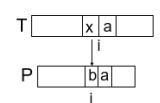


- Bài toán đổi sánh mẫu
- Thuật toán đổi sánh vét cạn
- Thuật toán Boyer-Moore
- Thuật toán Knuth-Morris-Pratt
- Thuật toán Wu-Manber
- Thuật toán Aho-Corasick



Thuật toán Boyer - Moore

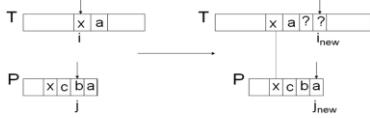
- ❖ Thuật toán đổi sánh mẫu Boyer – Moore dựa trên hai kỹ thuật chính:
 1. Kỹ thuật **looking-glass** (**đổi sánh cuối trước**)
 - Tim mẫu P trong T bằng cách tìm từ cuối về đầu của mẫu P
 - i, j là chỉ số của các ký tự đang được so sánh T[i] với P[j]
 - Nếu T[i] = P[j] thì $i = i - 1, j = j - 1$
 - Ngược lại thực hiện nhảy cách
 2. Kỹ thuật **character-jump** (**nhảy cách**)
 - Khi ký tự đổi sánh không khớp tại vị trí $T[i] = x$
 - Tức là $T[i] \neq P[j]$





Thuật toán Boyer – Moore (Nhảy cách-Case 1)

- ♦ $T[i] = x \neq P[j]$
 - Nếu P chứa x ở một vị trí t nào đó, $t < j$, thì dịch chuyển P sang phải sao vị trí t trong P thẳng với vị trí i hiện tại của T . Sau đó i chuyển sang vị trí i_{new} j chuyển về cuối của P (j_{new})



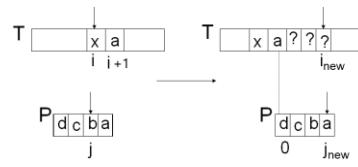
Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 163



Thuật toán Boyer – Moore (Nhảy cách-Case 2)

- ♦ $T[i] = x \neq P[j]$
 - Ngược lại, dịch P sao cho $P[0]$ thẳng với $T[i+1]$, sau đó i chuyển sang vị trí i_{new} và j chuyển về cuối của P (j_{new})



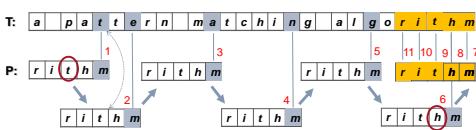
Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 164



Thuật toán Boyer – Moore (Ví dụ 1)

- ♦ Bài tập áp dụng:



Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 165



Tiền xử lý (hàm last occurrence)

- ♦ Thuật toán Boyer-Moore tiền xử lý mẫu P và bảng chữ cái Σ để xây dựng một hàm last occurrence $L()$
 - $L()$ ánh xạ tất cả các ký tự trong bảng chữ cái Σ thành số nguyên
- ♦ $L(x)$ được định nghĩa là:
 - Chỉ số i lớn nhất sao cho $P[i] == x$, hoặc
 - -1 nếu không tồn tại chỉ số i đó
- ♦ $L()$ được tính khi mẫu P được đưa vào.
- ♦ Giá trị của $L()$ được lưu trữ dưới dạng một mảng

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 167



Tiền xử lý (hàm last occurrence) – Ví dụ

P	a	b	a	c	a	b
	0	1	2	3	4	5

- $A = \{a, b, c, d\}$
- $P: "abacab"$

x	a	b	c	d
$L(x)$	4	5	3	-1

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 168

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 169

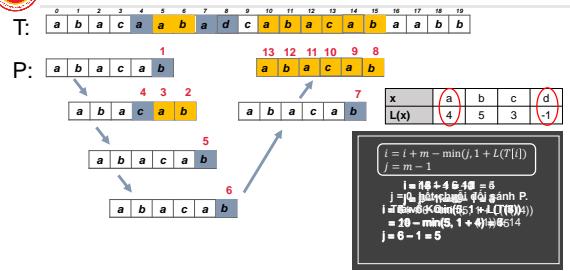


Tiền xử lý (hàm last occurrence)

- ♦ Khi thực hiện đối sánh gấp kí tự không khớp $T[i] \neq P[j]$ thì cần sử dụng $L()$ để nhảy cách như sau:
 - $i_{\text{new}} = i + m - \min(j, 1 + L(T[i]))$
 - $j_{\text{new}} = m - 1$
 - Trong đó m là độ dài của mẫu P



Ví dụ minh họa



Độ phức tạp

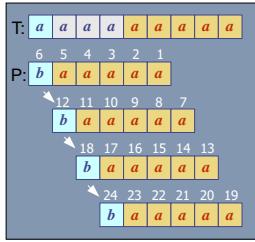
- Độ phức tạp thời gian của thuật toán Boyer-Moore trong trường hợp tồi nhất là O($nm + \Sigma$)
- Boyer-Moore thực hiện tốt nếu bảng chữ Σ lớn, chậm khi bảng chữ Σ nhỏ.
- Tốt cho các văn bản ngôn ngữ thông thường, kém cho tệp nhị phân
- Boyer-Moore **nhanh hơn nhiều lần** so với vét cạn.



Độ phức tạp (trường hợp tệ nhất)

- T: "aaaaaaaa"
- P: "baaaaa"

Mất bao nhiêu bước thực hiện đổi sánh?



ĐỔI SÁNH MẪU TRÊN CHUỖI



- Bài toán đổi sánh mẫu
- Thuật toán đổi sánh vét cạn
- Thuật toán Boyer-Moore
- Thuật toán Knuth-Morris-Pratt
- Thuật toán Wu-Manber
- Thuật toán Aho-Corasick



Knuth-Morris-Pratt (KMP)

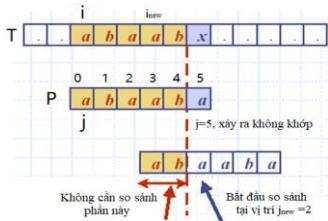
- Thuật toán Knuth-Morris-Pratt tìm kiếm mẫu P trong chuỗi T theo thứ tự từ trái sang phải (tương tự vét cạn).
- Dịch chuyển mẫu thông minh hơn so với vét cạn
- Nếu xuất hiện sự không trùng khớp giữa mẫu P tại $P[j]$, bước dịch chuyển dài nhất có thể thực hiện là bao nhiêu?
 - Trả lời:* Độ dài của tiền tố dài nhất của $P[0 .. j-1]$ sao cho nó là hậu tố của $P[1 .. j-1]$



Knuth-Morris-Pratt (KMP)

- Khi thực hiện so sánh, luôn cần sử dụng tới 2 giá trị sau:
 - i: vị trí bắt đầu của chuỗi con trong T đang so sánh với mẫu P, tức là $T[i]$ **thẳng** với $P[0]$
 - j: vị trí của kí tự trong P đang được so sánh (so sánh $T[i+j]$ với $P[j]$)
- Gặp kí tự khớp thì tăng j lên 1
- Ngược lại, gặp kí tự không khớp, cần xác định i_{new} và j_{new}

Ví dụ



Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 176

Ví dụ

- ❖ Giả sử xuất hiện kí tự không giống nhau tại j , tức là $T[i + j] \neq P[j]$, khi đó:
 - Tìm tiền tố dài nhất của $P[0..j-1] = "a b a a b"$ sao cho tiền tố đó cũng là hậu tố của $P[1..j-1] = "b a a b"$
 - => Đó là: "a b"
 - Không cần phải so sánh lại chuỗi con "a b" nữa nên đặt $j_{new} = 2$

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 177



Hàm failure function

- ❖ Là hàm tiền xử lý mẫu của KMP
- Để tìm kiếm sự trùng khớp của các tiền tố trong $P[0..j-1]$ với các hậu tố trong $P[1..j-1]$, với j là vị trí tại đó $T[i + j] \neq P[j]$
- Được sử dụng để xác định i_{new} và j_{new}

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 178



Hàm failure function

- ❖ Hàm **failure function** $F(j)$ được định nghĩa là kích thước của tiền tố dài nhất của $P[0..j-1]$ sao cho nó cũng là hậu tố của $P[1..j-1]$.
- Mặc định:
 - $F(0) = -1$
 - Độ dài của xâu rỗng là 0

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 179



Ví dụ: Tính hàm failure function

- ❖ P: "abacab", $j = [0..5]$
- $j=0 \Rightarrow F(0)=-1$
- $j=1$
 - Các tiền tố của $P[0..j-1] = "a"$ là {a}
 - Các hậu tố của $P[1..j-1] = \emptyset$ là {}
 - => $F(1)=0$

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 180



Ví dụ: Tính hàm failure function

- ❖ P: " abacab", $j = [0..5]$
- $j=2$
 - Các tiền tố của $P[0..j-1] = "ab"$ là {a,ab}
 - Các hậu tố của $P[1..j-1] = "b"$ là {b}
 - => $F(2)=0$

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 181



Ví dụ: Tính hàm failure function

❖ P: " abacab", j = [0..5]

▫ j=3

- Các tiền tố của P[0..j-1] = "aba" là {a, ab, aba}
- Các hậu tố của P[1..j-1] = "ba" là {ba, a}
- => F(3)=1



Ví dụ: Tính hàm failure function

❖ P: " abacab", j = [0..5]

▫ j=4

- Các tiền tố của P[0..j-1] = "abac" là {a, ab, aba, abac}
- Các hậu tố của P[1..j-1] = "bac" là {bac, ac, c}
- => F(4)=0



Ví dụ: Tính hàm failure function

❖ P: " abacab", j = [0..5]

▫ j=5

- Các tiền tố của P[0..j-1] = "abaca" là {a, ab, aba, abac, abaca}
- Các hậu tố của P[1..j-1] = "baca" là {baca, aca, ca, a}
- => F(5)=1

j	0	1	2	3	4	5
F(j)	-1	0	0	1	0	1



Knuth-Morris-Pratt

❖ Giả sử:

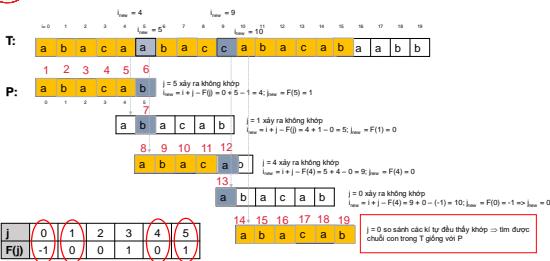
- i là vị trí bắt đầu của chuỗi con trong T mà đang so sánh với mẫu P, tức là $T[i]$ thắng với $P[0]$
- j: vị trí của kí tự trong P đang được so sánh và $T[i + j] \neq P[j]$

❖ => Xác định i_{new} và j_{new} như sau:

- $i_{new} = i + j - F[i]$
- $j_{new} = F[j]$, nếu $F[j] = -1$ thì $j_{new} = 0$



Ví dụ



Ưu nhược điểm của KMP

- ❖ KMP có độ phức tạp tối ưu: $O(m+n) \rightarrow$ rất nhanh
- ❖ Thuật toán không cần di chuyển ngược lại trong văn bản T
 - Điều này làm cho nó hiệu quả khi xử lý trên các tệp tin rất lớn được đọc vào từ thiết bị ngoại vi hoặc trong luồng mạng
- ❖ KMP không làm việc tốt trên bộ ký tự đối sánh có kích thước lớn
 - Xác suất không khớp lớn hơn
 - Việc không khớp xuất hiện sớm trong mẫu, nhưng KMP lại nhanh hơn khi việc không khớp xảy ra ở phía sau của mẫu



ĐỔI SÁNH MẪU TRÊN CHUỖI



- Bài toán đổi sánh mẫu
- Thuật toán đổi sánh vét cạn
- Thuật toán Boyer-Moore
- Thuật toán Knuth-Morris-Pratt
- Thuật toán Wu-Manber
- Thuật toán Aho-Corasick



Bài toán đổi sánh mẫu

Văn bản N K A N M A N K **K B A A**
Mẫu K B A A

- Bài toán đổi sánh đơn mẫu: Cho trước một chuỗi ký tự gọi là văn bản và một chuỗi ký tự gọi là mẫu, xác định xem văn bản có chứa mẫu hay không
- Bài toán đổi sánh đa mẫu: Cho trước một tập hợp các mẫu (cũng là các chuỗi ký tự), xác định xem văn bản có chứa mẫu nào hay không
- Hai thuật toán phổ biến nhất để giải quyết bài toán đổi sánh mẫu
 - Boyer-Moore
 - Aho-Corasick



Thuật toán Boyer-Moore (BM)

Văn bản (T) N K A N M A N K **K B A A**
Mẫu (P) K B A A

- Giải quyết bài toán đổi sánh đơn mẫu
- Chuỗi ký tự cần tìm kiếm gọi là mẫu, ký hiệu P và có độ dài m
- Chuỗi ký tự được tìm kiếm trong đó gọi là văn bản, ký hiệu T và có độ dài l'
- Bài toán: Xác định mẫu có xuất hiện trong văn bản không và nếu có xuất hiện bao nhiêu lần và ở vị trí nào
- Thuật giải duyệt toàn bộ
 - Đặt P và T sao cho ký tự đầu tiên của 2 chuỗi thẳng hàng
 - Dịch chuyển P sang bên phải từng ký tự một, so sánh P với các ký tự thẳng hàng tương ứng của T xem có trùng khớp không
 - Quá trình kết thúc khi ký tự cuối cùng của P thẳng hàng với ký tự cuối cùng của T



Phương pháp duyệt toàn bộ

①	Văn bản Mẫu	N K A N M A N K K B A A	⑤	Văn bản Mẫu	N K A N M A N K K B A A
②	Văn bản Mẫu	N K A N M A N K K B A A	⑥	Văn bản Mẫu	N K A N M A N K K B A A
③	Văn bản Mẫu	N K A N M A N K K B A A	⑦	Văn bản Mẫu	N K A N M A N K K B A A
④	Văn bản Mẫu	N K A N M A N K K B A A	⑧	Văn bản Mẫu	N K A N M A N K K B A A
⑨	Văn bản Mẫu	N K A N M A N K K B A A			



Quy tắc dịch chuyển của BM (1)

- Với mỗi bước so sánh P với T, nếu ký tự cuối cùng của mẫu không trùng với ký tự tương ứng của văn bản giả sử là σ, xét 2 trường hợp sau

1. σ không xuất hiện trong P. Dịch chuyển P sao cho toàn bộ P nằm ngay bên phải σ
2. Nếu σ có xuất hiện trong P, xác định N nằm ngoài cùng bên phải của P. Dịch chuyển P sao cho σ của T thẳng hàng với σ của P

1 2 3 4 5 6 7 8 9 1 1 1	0 1 2
N K A N M A N K K B A A	
K B A A	
qua	[K B A A]
	[K B A A]
	[K B A A]
	[K B A A]

1) σ = N không xuất hiện trong mẫu

N P A N M A N **K** K B A A

Bổ	K B A A
qua	[K B A A]
	[K B A A]
	[K B A A]

N P A N M A N **K** K B A A



Quy tắc dịch chuyển của BM (2)

- Nếu ký tự cuối cùng của mẫu trùng với ký tự tương ứng của văn bản

- So sánh từng ký tự của mẫu với ký tự tương ứng của văn bản. Có 2 trường hợp sau:

1. Nếu đến một ký tự nào đó không có sự trùng khớp, dịch mẫu sang phải 1 ký tự để thực hiện bước tiếp theo
2. Nếu toàn bộ ký tự của mẫu trùng với các ký tự tương ứng của văn bản, thông báo tìm thấy mẫu trong văn bản

1 2 3 4 5 6 7 8 9 1 1 1	0 1 2
N K A N M A N P K B A A	
K B A A	

1 2 3 4 5 6 7 8 9 1 1 1	0 1 2
N K A N M A N P K B A A	
K B A A	

1 2 3 4 5 6 7 8 9 1 1 1	0 1 2
N K A N M A N P K B A A	
K B A A	



Xây dựng bảng delta

- Xây dựng bảng delta[σ] cho mỗi $\sigma \in \Sigma$. delta[σ] cho biết số ký tự sẽ được dịch chuyển khi gặp σ trong văn bản
- Nếu σ không xuất hiện trong mẫu P, delta[σ] = độ dài mẫu
- Nếu σ xuất hiện trong mẫu P, delta[σ] = độ dài mẫu - j, với j là vị trí ngoài cùng bên phải σ xuất hiện trong P

- $\Sigma = \{A, B, K, M, N\}$
- P = KBAA
- delta[A] = 0
- delta[B] = 2
- delta[K] = 3
- delta[M] = 4
- delta[N] = 4

	Delta[σ]
A	0
B	2
K	3
M	4
N	4



Ví dụ thuật toán Boyer-Moore

①	1 2 3 4 5 6 7 8 9 10 11 12	$\Sigma = \{A, B, K, M, N\}$
	N K A N M A N K B A A	
	K B A A	
②	1 2 3 4 5 6 7 8 9 10 11 12	
	N K A N M A N K K B A A	
	K B A A	
③	1 2 3 4 5 6 7 8 9 10 11 12	
	N K A N M A N K B A A	
	K B A A	
④	1 2 3 4 5 6 7 8 9 10 11 12	
	N K A N M A N K B A A	
	K B A A	

A	0
B	2
K	3
M	4
N	4

Bảng Delta

delta[A] = 0 kiểm tra các ký tự tiếp theo



Thuật toán Wu-Manber

Thuật toán Boyer-Moore được mở rộng bởi Wu và Manber cho phép đối sánh nhiều mẫu đồng thời

P = {p₁, p₂, ..., p_m} là một tập hợp các mẫu

Mỗi mẫu là một chuỗi ký tự với các ký tự lấy từ bảng chữ cái Σ

T = t₁, t₂, ..., t_n là chuỗi ký tự để tìm kiếm các mẫu trong đó

m là độ dài nhỏ nhất của các mẫu

Thay vì chỉ xét một ký tự như trong thuật toán Boyer-Moore, một block B ký tự sẽ được so sánh cùng một lúc



Quy tắc dịch chuyển

Văn bản	B M A N P A N A M A N A P	B M A N P A N A M A N A P
Mẫu 1	N A M A N A	N A M A N A
Mẫu 2	A M A N A B	A M A N A B
Mẫu 3	A B M A N A	A B M A N A
	N A M A N A	N A M A N A
	A M A N A B	A M A N A B
	A B M A N A	A B M A N A

1) **NPA** không xuất hiện trong bất cứ mẫu nào. Cả 3 mẫu cùng dịch 4 ký tự

2) **AMA** xuất hiện trong 3 mẫu. Mẫu 1 được phép dịch 2 ký tự, mẫu 2 được phép dịch 3 ký tự, mẫu 3 được phép dịch 4 ký tự, cả 3 mẫu cùng dịch 2 ký tự



Xây dựng bảng SHIFT

- Cho trước một chuỗi ký tự bắt đầu có độ dài B, bảng SHIFT[σ] cho biết số ký tự sẽ được dịch chuyển khi gặp chuỗi ký tự đó trong văn bản T
- Giá trị ban đầu SHIFT[X₁...X_B] = m-B+1
- Xét từng mẫu p_i = a₁a₂...a_m. Xét các chuỗi ký tự con có độ dài B của p_i = a_{i+B-1}...a_i với B ≤ m. Ta có SHIFT[p_i] = min(SHIFT[p_i], m-j)

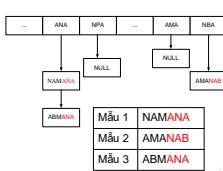
- Xét 3 mẫu
- Mẫu 1 N A M A N A
- Mẫu 2 A M A N A B
- Mẫu 3 A B M A N A
- Tính SHIFT[AMA]
- Ban đầu SHIFT[AMA] = 6-3+1 = 4
- Xét N A M A N A → SHIFT[AMA] = 2
- Xét A M A N A B → SHIFT[AMA] = 2
- Xét A B M A N A → SHIFT[AMA] = 2
- SHIFT[AMA] = 2



Xây dựng bảng HASH

- Khi SHIFT[X₁...X_B] = 0 tức là tồn tại các mẫu có B ký tự cuối cùng trùng với B ký tự đang xét của T
- Bảng HASH cho biết các mẫu đó là mẫu nào
- HASH[X₁...X_B] chứa trả về danh sách các mẫu với B ký tự cuối cùng là X₁...X_B.
- Xét từng mẫu p_i = a₁a₂...a_m.
- Đưa đưa p_i vào danh sách liên kết mà HASH[a_{m-B+1}...a_m] trả về

B	M	A	N	P	A	N	A	M	A	N	A



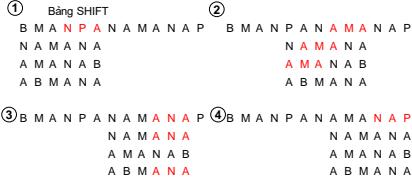


Ví dụ thuật toán Wu-Manber

NPA	4
AMA	2
ANA	0
NAP	4
NAB	0

NAB	→AMANAB
ANA	→NAMANA→ABMANA

Bảng HASH



Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 200



ĐỔI SÁNH MẪU TRÊN CHUỖI



- ✓ Bài toán đổi sánh mẫu
- ✓ Thuật toán đổi sánh vét cạn
- ✓ Thuật toán Boyer-Moore
- ✓ Thuật toán Knuth-Morris-Pratt
- ✓ Thuật toán Wu-Manber
- ✓ Thuật toán Aho-Corasick

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 201



$P = \{p_1, p_2, \dots, p_k\}$ là một tập hợp các mẫu, mỗi mẫu là một chuỗi ký tự $T = t_1, t_2, \dots, t_n$ là chuỗi ký tự để tìm kiếm các mẫu trong đó và được gọi là văn bản

Các ký tự của mẫu và văn bản được lấy ra từ bảng chữ cái Σ .
Bài toán: Xác định các mẫu trên có xuất hiện trong văn bản và nếu có thì xuất hiện bao nhiêu lần và ở vị trí nào

Thuật toán Aho-Corasick (AC) đưa ra lời giải cho bài toán trên với độ phức tạp $O(n+m+z)$ với z là số lần xuất hiện của các mẫu và n là tổng độ dài của các mẫu

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 202

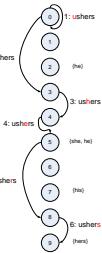


Automaton của Aho-Corasick

- Chu trình đầu tiên: Bắt đầu từ trạng thái 0, đọc ký tự đầu tiên của văn bản và chuyển sang trạng thái tiếp theo căn cứ trên ký tự này và trạng thái hiện thời
- Automaton của thuật toán Aho-Corasick là một máy hữu hạn trạng thái (finite-state machine) được xây dựng từ tập mẫu P .
 - Mỗi trạng thái được đại diện bằng một con số.
 - Trạng thái ban đầu thường được đại diện bởi số 0.
 - Hệm chuyển trạng thái $t = \delta(s, a)$ cho biết chuyển đến trạng thái nào từ trạng thái hiện thời và ký tự đầu vào
 - Mỗi trạng thái được liên kết với một tập con của mẫu (cố thể là rỗng) cho biết các mẫu của tập con này đã được tìm thấy
- Ví dụ: automaton cho tập mẫu $P = \{\text{he, she, his, hers}\}$; Văn bản $T = \text{ushers}$

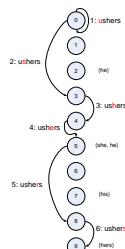
Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 203



Hoạt động của automaton Aho-Corasick

- Hoạt động của automaton dựa trên các chu trình
 - Chu trình đầu tiên: Bắt đầu từ trạng thái 0, đọc ký tự đầu tiên của văn bản và chuyển sang trạng thái tiếp theo căn cứ trên ký tự này và trạng thái hiện thời
 - Chu trình thứ hai: Đọc ký tự thứ hai của văn bản và chuyển sang trạng thái tiếp theo dựa trên ký tự này và trạng thái hiện thời
 - Lần lượt lặp như trên cho đến chu trình cuối cùng: đọc ký tự cuối cùng của văn bản, chuyển đến trạng thái cuối cùng và ngừng hoạt động
 - Khi đến trạng thái kết với một tập con khác rỗng của P , in các mẫu của tập con đó ra. Đó là các mẫu xuất hiện trong văn bản
- Ví dụ: automaton cho tập mẫu $P = \{\text{he, she, his, hers}\}$; Văn bản $T = \text{ushers}$



Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 204



Hàm chuyển trạng thái δ

Trạng thái hiện thời	Ký tự đầu vào	Trạng thái tiếp theo	Trạng thái hiện thời	Ký tự đầu vào	Trạng thái tiếp theo
0	h	1	9, 7, 3	h	4
	s	3		s	3
	.	0		.	0
1	e	2	5, 2	r	8
	i	6		h	1
	h	1		s	3
	s	3		.	0
4	0	0	6	s	7
				h	1
				.	0
				s	9
8	5	6		h	1
	6	1		.	0
	3	3		s	9
	0	0		h	1
				.	0

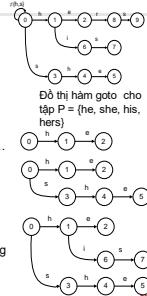
Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

16 May 2021 | Page 205



Hàm goto

- Hàm chuyển trạng thái được xây dựng dựa trên hàm goto và hàm failure
- Hàm goto
 - $g(\text{trạng thái}, \text{ký tự đầu vào}) = \text{trạng thái}$
 - hoặc $g(\text{trạng thái}, \text{ký tự đầu vào}) = fail$.
- Ví dụ: đồ thị hàm goto cho tập mẫu P = {he, she, his, hers}.
- $g(1, e) = 2; g(1, i) = 6;$
- $g(1, o) = fail$ với $o \neq (e, i)$ do đó $g(1, h) = fail; g(1, r) = fail\dots$
- Riêng $g(0, o) \neq fail$ với mọi o
- Xây dựng đồ thị hàm goto:
 - Các node tương ứng với các trạng thái, các cạnh tương ứng với các ký tự
 - Bắt đầu với trạng thái 0
 - Lần lượt thêm các mẫu vào đồ thị
 - Mỗi mẫu được thêm bằng cách thêm một đường đi có hướng từ trạng thái đầu tiên sao cho các cạnh của đường đi đó tương ứng với mẫu đó.



May 2021 | Page 206

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin



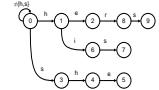
Hàm failure

- Hàm failure $f(\text{trạng thái}) = \text{trạng thái}$
- Hàm failure được xây dựng dựa trên hàm goto
- $f(s) = 0$ cho tất cả các trạng thái kè với trạng thái 0. Ví dụ: $f(1) = 0, f(3) = 0$.
- Bắt đầu từ node gốc 0, các node được thăm theo phương pháp tìm kiếm theo chiều rộng: các node ở gần gốc hơn được thăm trước
- Giả sử ta muốn tìm giá trị hàm failure cho trạng thái s. Tìm trạng thái r và ký tự a sao cho $g(r, a) = s$:

 - $D(x) u_i = f(r), u_1 = f(u_1), u_2 = f(u_2) \dots$ cho đến khi $g(u_n, a) \neq fail$
 - $D(x) f(s) = g(u_n, a)$

- Tính $f(2)$: ta có $g(1, e) = 2$ nên đặt $u_1 = f(1) = 0$; vì $g(0, e) = 0 \neq fail$ nên $f(2) = 0$
- Tính $f(5)$: ta có $g(4, e) = 5$ nên đặt $u_1 = f(4) = 1$; vì $g(1, e) = 2 \neq fail$ nên $f(5) = 2$

i	1	2	3	4	5	6	7	8	9
0	0	0	1	2	0	3	0	3	



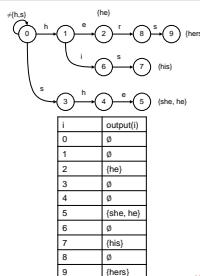
May 2021 | Page 207

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin



Hàm output

- Hàm output liên kết một trạng thái với một tập con của tập mẫu P (có thể là rỗng) cho biết đã tìm thấy tập con đó trong văn bản
- Hàm output được xây dựng đồng thời với hàm goto. Mẫu kết thúc tại trạng thái nào thì liên kết với trạng thái đó
- Hàm output còn được cập nhật bởi hàm failure. Ví dụ, $f(5) = 2$ mà $\text{output}(2) = \text{he}$ nên $\text{output}(5) = \text{output}(5) \cup \text{output}(2) = \{\text{she, he}\}$



May 2021 | Page 208

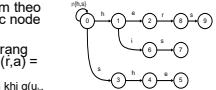
Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin



Xây dựng hàm chuyển trạng thái

- Xây dựng hàm chuyển trạng thái ô từ hàm goto và hàm failure
- Bắt đầu từ node gốc 0, các node được thăm theo phương pháp tìm kiếm theo chiều rộng
- Nếu $g(s, a) \neq fail$ $\delta(s, a) = g(s, a)$
- Nếu không $\delta(s, a) = \delta(f(s), a)$
- $\delta(4, e) = \delta(4, e) = 5$
- $\delta(4, i) = \delta(f(4), i) = \delta(1, i) = 6$
- $\delta(4, h) = \delta(f(4), h) = \delta(1, h) = 1$
- $\delta(4, s) = \delta(f(4), s) = \delta(1, s) = 3$
- $\delta(4, .) = \delta(f(4), .) = \delta(1, .) = 0$

i	1	2	3	4	5	6	7	8	9
0	0	0	0	1	2	0	3	0	3



May 2021 | Page 209

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin



So sánh Wu-Manber và Aho-Corasick

	Wu-Manber	Aho-Corasick
Cấu trúc dữ liệu của chữ ký mã độc	Đơn giản, chủ yếu là danh sách liên kết	Phức tạp, chủ yếu là cây tiền tố (trie)
Quá trình tiền xử lý	Không hiệu quả bằng Aho-Corasick	Tối ưu cho việc quét nhiều mẫu đồng thời
Format chữ ký mã độc	Thích hợp với các chữ ký dài và cố định	Thích hợp với các chữ ký dài và có định (wildcard, có dạng aabbcc*bbaacc)

Bảng so sánh một số tính chất của Wu-Manber và Aho-Corasick

Bộ môn Khoa Học An Toàn Thông Tin – Khoa An Toàn Thông Tin

May 2021 | Page 210