**BỘ GIÁO DỤC – ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN ĐHQGHCM**

KHOA CÔNG NGHỆ THÔNG TIN



**BÁO CÁO**

Môn học: Nhập môn mã hoá – mật mã

Bài tập nhóm

19127449 – Phùng Anh Khoa

19127525 – Nguyễn Thanh Quân

TP.HCM, ngày 06 tháng 12 năm 2021

Mục lục

# **Lý thuyết**

## **Giới thiệu**

Để hiểu rõ hơn về thuật toán kiểm tra số nguyên tố ta cần hiểu số nguyên tố là gì và tại sao lại có thuật toán kiểm tra số nguyên tố.

Số nguyên tố là số tự nhiên lớn hơn 1 và chỉ có hai ước số là 1 và chính nó, như vậy các số 2, 3, 5, 7, 11, … được coi là các số nguyên tố. Kiểm tra một số có phải là một số nguyên tố hay không là một bài toán đơn giản mà bất kỳ một lập trình viên nào cũng đã làm. Tuy nhiên, Để đáp ứng nhu cầu về bảo mật của ngành mật mã hoá, những thuật toán kiểm tra số nguyên tố phải xử lý được cái số có số bit rất lớn mà các thuật toán đơn giản không thể xử lý được hoặc xử lý rất lâu.

Vì nhu cầu trên nên các thuật toán kiểm tra số nguyên tố như Fermat, Miller-Rabin, Solovay-Strassen, Frobenius, baillie-PSW, eliptic curve, pocklington, … ra đời để giải quyết vấn đề về tốc độ và bảo mật.

Để hiểu rõ hơn nhóm sẽ phân tích và cài đặt thuật toán kiểm tra số nguyên tố **Miller-Rabin**.

**Miller-Rabin** là một thuật toán xác suất để kiểm tra tính nguyên tố của một số, Được đề xuất bởi [Gary L. Miller](https://www.uakron.edu/president/) và là thuật toán dựa trên giả thiết [Riemann tổng quát.](https://vi.wikipedia.org/wiki/Gi%E1%BA%A3_thuy%E1%BA%BFt_Riemann) Thuật toán dựa trên định lý quan trọng sau

* *Nếu n là số nguyên tố thì (n-1)! ≡ (n-1) (mod n)*
* *Với mỗi số nguyên tố, ∅(n) là các số nguyên tố cùng nhau với n nhưng nhỏ hơn n.*

*Khi đó, ∀x, x > 0, x∅(n) ≡ 1 (mod n)*

## **Phân tích**

Giống với Fermat và Solovay-Strassen, Miller-Rabin sử dụng nguyên lý [đồng dư](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%93ng_d%C6%B0) (Mô-đun) nhưng sử dụng thêm hệ thống các kết quả (System of congruence) để giữ các giá trị thử nghiệm.

Với một số n, ta có thể viết như sau:

n – 1 = 2s \* d, d là số lẻ

ví dụ: 11 = 20 \* 11 36 = 22 \* 9

Nếu như n là một số nguyên tố thì theo định lý fermat

an-1 ≡ 1 mod n với n-1 = 2s d, d là số lẻ

an-1 ≡ 1 (mod n) ⬄ a2^s \*d - 1 ≡ 0 mod n

⬄ (a2^(s-1) \* d + 1) (a2^(s-1) \* d -1) ≡ 0 mod n

⬄ (a2^(s-1) \* d + 1) (a2^(s-2) \* d + 1) (a2^(s-2) \* d - 1) ≡ 0 mod n

⋮

⬄ (a2^(s-1) \* d + 1) (a2^(s-2) \* d + 1) … (ad + 1) (ad - 1) ≡ 0 mod n

**(1)** Khi n là số nguyên tố thì chắc chắn một trong các hệ số này phải bằng 0 mod n:

ad ≡ 1 (mod n) hoặc a2^i \*d ≡ -1 (mod n) cho một số i ∈ {0, …, s}

Ví dụ:

nếu n = 13 => n – 1 = 22 \* 3

* s = 2, d = 3
* a3 ≡ 1 (mod n)
* a3 ≡ -1 (mod n)
* a6 ≡ -1 (mod n)
* cho a từ khoảng 2 tới 12

nếu n = 61 => n – 1 = 22 \* 15

* s = 2, d = 15
* a15 ≡ 1 (mod n)
* a15 ≡ -1 (mod n)
* a30 ≡ -1 (mod n)
* cho a từ khoảng 2 tới 60

**(2)** Với n lẻ > 1, n-1 = 2s \* d, d lẻ và chọn a trong khoảng 1 tới n-1

- Nếu toàn bộ các đồng dư trong (1) đều sai, Thì ta nói a là Miller-rabin witness

ad ≢ 1 (mod n) và a2^i \*d ≢ -1 (mod n) cho toàn bộ i trong khoảng 0 tới n – 1

- Nếu có một đồng dư trong (1) đúng, Thì ta nói a là Miller-rabin nonwitness

ad ≡ 1 (mod n) và a2^i \*d ≡ -1 (mod n) cho vài i trong khoảng 0 tới n – 1

Như thuật giải fermat và solovay-Strasen, Miller-rabin witness là định nghĩa để chỉ một số n là hợp số

Và một số nguyên tố thì sẽ không có Miller-rabin witness

Vậy để thực hiện thuật toán này ta làm như sau:

+ Với n là số cần kiểm tra

+ Ta kiểm tra n có phải là số lẻ hay không, nếu không phải số lẻ thì return false

+ Phân tích n – 1 thành 2s \* d

+ Chọn ngẫu nhiên 1 số a trong khoảng 0 tới n-1

+ Kiểm tra nếu ad ≡ 1 (mod n) thì return true

+ Nếu không thì cho i chạy từ 0 tới s

* kiểm tra nếu ai\*d ≡ -1 (mod n) thì return true

+ Nếu các trường hợp trên đều không đúng thì return false

Xác suất để thuật toán chạy sai: Nếu n là hợp số dương lẻ thì trong các số a ∊ {2, …, n-1} tồn tại không quá n-1/4 a để n là số giả nguyên tố mạnh fermat.

## **Cài đặt**

*Pseudocode*

* Input: số tự nhiên n
* Output: True / False

Miller\_Rabin(n):

**if** n % 2 == 0 **then** return False

a = random(2, n-1)

s, d = fermat(n-1)

**if** mod(ad, n) == 1 **then** return True

**For** i = 0 **to** s

**if** mod(ai\*d, n) == -1 return True

return False

Ref

<https://codelearn.io/sharing/thuat-toan-kiem-tra-tinh-nguyen-to>

<https://en.wikipedia.org/wiki/Primality_test>

<https://vi.wikipedia.org/wiki/Ki%E1%BB%83m_tra_Miller-Rabin>

<https://www.math.arizona.edu/~jtaylor/notes/crypto_talks/primality_testing_and_the_miller-rabin_algorithm.pdf>

<https://123docz.net/document/19067-xay-dung-chuong-trinh-kiem-tra-so-nguyen-to-bang-thuat-toan-miller-rabin-doc-doc.htm>

<https://kconrad.math.uconn.edu/blurbs/ugradnumthy/millerrabin.pdf>