BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

ĐỒ ÁN TỐT NGHIỆP

NGÀNH : CÔNG NGHỆ THÔNG TIN

Sinh viên : Đỗ Hoàng Anh

Giảng viên hướng dẫn : TS Hồ Văn Canh

HẢI PHÒNG – 2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

PHƯƠNG PHÁP NHẬN BIẾT SỐ NGUYÊN TỐ 2^N-1

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh viên : Đỗ Hoàng Anh

Giảng viên hướng dẫn : TS Hồ Văn Canh

HẢI PHÒNG – 2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

NHIỆM VỤ ĐỀ TÀI TỐT NGHIỆP

Sinh viên: Đỗ Hoàng Anh **Mã SV:** 1812101004

Lớp : CT2201C

Ngành : Công nghệ thông tin

Tên đề tài: Phương pháp nhận biết số nguyên tố dạng 2^n-1

NHIỆM VỤ ĐỀ TÀI

1. Mô tả tóm tắt đề tài:

- Tìm hiểu về vai trò của số nguyên tố trong an toàn bảo mật thông tin

- Tìm hiểu các phương pháp nhận biết số nguyên tố dạng 2^n-1

- Cài đặt chương trình thử nghiệm.

2.Các tài liệu cần thiết

[1] Hồ Văn Canh, Lê Danh Cường (2018): Mật mã và an toàn thông tin: Lý thuyết và ứng dụng. NXB: Thông tin và Truyền thông – 8/2018.

[2] Mennezes, Paul C. Van Oorschot, S cott A. Vanstone (1999): Handbook of Applied Cryptography. CRC Press: Boca Raton, New York, London, Tokyo.

[3] Neal Koblitz (2000): A Course in Number Theory and Cryptography.

Springer-Verlag Press: New York, Berlin Heidelberg, London, Pái, and Tokyo (2000).

[4] Phan Đình Diệu (2002): Mật mã và an toàn thông tin. NXB Đại học Quốc gia Hà Nội năm 2002

.[5] Trịnh Nhật Tiến (2003): Mật mã và an toàn CSDL. NXB ĐHQG Hà Nội năm 2003.

3.Địa điểm thực tập tốt nghiệp

Công ty cổ phần đầu tư tài chính và công nghệ Datatech

CÁN BỘ HƯỚNG DẪN ĐỀ TÀI TỐT NGHIỆP

Họ và tên : Hồ Văn Canh

Học hàm, học vị : Đại Tá-Tiến Sĩ

Cơ quan công tác : Học Viện Kỹ Thuật Mật Mã

Nội dung hướng dẫn:

Nội dung dự kiến

- Một số khái niệm cơ bản trong số học, đại số.
- Một số phương pháp kiểm tra số nguyên tố.
- Ứng dụng của số nguyên tố và thử nghiệm chương trình..

Đề tài tốt nghiệp được giao ngày 17 tháng 4 năm 2023

Yêu cầu phải hoàn thành xong trước ngày 17 tháng 6 năm 2023

Đã nhận nhiệm vụ ĐTTN	**Đã giao nhiệm vụ ĐTTN**
Sinh viên	Giảng viên hướng dẫn
Đỗ Hoàng Anh	**TS. Hồ Văn Canh**

Hải Phòng, ngày 8 tháng 6 năm 2023

TRƯỞNG KHOA

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN TỐT NGHIỆP

Họ và tên giảng viên: Hồ Văn Canh

Đơn vị công tác: Cục Kỹ thuật Nghiệp Vụ-CBA

Họ và tên sinh viên: Đỗ Hoàng Anh Ngành: Công nghệ thông tin

Đề tài tốt nghiệp: Phương pháp nhận biết số nguyên tố dạng 2^n-1.

1. Tinh thần thái độ của sinh viên trong quá trình làm đề tài tốt nghiệp

……

……

……

……

2. Đánh giá chất lượng của đồ án/khóa luận(so với nội dung yêu cầu đã đề ra trong nhiệm vụ Đ.T. T.N trên các lý luận, thực tiễn, tính toán số liệu…)

……

……

……

……

3. Ý kiến của giảng viên hướng dẫn tốt nghiệp

Đạt ☐ Không đạt ☐ Điểm:…….

Hải Phòng, ngày 8 tháng 6 năm 2023

Giảng viên hướng dẫn

(Ký và ghi rõ họ tên)

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN CHẤM PHẢN BIỆN

Họ và tên giảng viên:

Đơn vị công tác:

Họ và tên sinh viên: Đỗ Hoàng Anh Ngành: Công nghệ thông tin

Đề tài tốt nghiệp:**.**

1. Phần nhận xét của giảng viên chấm phản biện

……

……

……

……

2. Những mặt còn hạn chế

……

……

……

……

3. Ý kiến của giảng viên chấm phản biện

Được bảo vệ Không được bảo vệ Điểm……

Hải Phòng, ngày 13 tháng 6 năm 2023

Giảng viên chấm phản biện

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Lời đầu tiên, em xin gửi lời cảm ơn chân thành đến các Thầy Cô trong Khoa Công Nghệ Thông, Trường Đại học Quản Lý và Công Nghệ Hải Phòng đã giảng dạy, chỉ bảo cho em kiến thức và kinh nghiệm quý báu trong suốt 4 năm học tại trường để em có thể thực hiện đồ án tốt nghiệp này. Đặc biệt em xin gửi lời cảm ơn sâu sắc tới Thầy Hồ Văn Canh người đã trực tiếp hướng dẫn và tận tình giúp đỡ em hoàn thành tốt đồ án tốt nghiệp của mình. Em cũng xin cảm ơn Cô Nguyễn Thị Xuân Hương – Lãnh đạo Khoa Công Nghệ Thông Tin đã luôn tạo điều kiện cho em và các bạn trong suốt quá trình học cũng như thực hiện các công tác tốt nghiệp.

Em xin trân trọng cảm ơn Ban lãnh đạo, các Thầy Cô ở các phòng ban của Trường ĐH Quản Lý và Công Nghệ Hải Phòng đã cho em môi trường học tập tốt nhất có thể từ khi em bắt đầu đặt chân vào giảng đường và cho đến khi hoàn thành đồ án tốt nghiệp quan trọng nhất trong cuộc đời sinh viên.

Trong quá trình thực tập, cũng như là trong quá trình làm đồ án tốt nghiệp em không tránh khỏi những thiếu sót về trình độ lý luận cũng như kinh nghiệm thực tiễn nên em rất mong sự đóng góp ý kiến và chỉ bảo từ Thầy, Cô để em tiến bộ hơn và có thêm nhiều kinh nghiệm và kiến thức để có thể góp ích cho những công việc sau này.

Em xin chân thành cảm ơn!

LỜI CAM ĐOAN

Em xin cam đoan rằng đề tài này được tiến hành một cách minh bạch, công khai. Mọi thứ được dựa trên sự cố gắng cũng như sự nỗ lực của bản thân cùng với sự giúp đỡ của thầy Hồ Văn Canh.

Các số liệu và kết quả nghiên cứu được đưa ra trong đồ án là trung thực và không sao chép hay sử dụng kết quả của bất kỳ đề tài nghiên cứu nào tương tự. Nếu như phát hiện rằng có sự sao chép kết quả nghiên cứu đề những đề tài khác bản thân em xin chịu hoàn toàn trách nhiệm.

Hải Phòng, ngày 13 tháng 6 năm 2022

Sinh viên

(Ký và ghi rõ họ tên)

MỤC LỤC

DANH MỤC HÌNH

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Tiếng Anh	Tiếng Việt
Ged	Greatest Common Divisor	Ước số chung lớn nhất
Lcm	Least Common Multiple	Bội số chung nhỏ nhất
Mod	Modulo	

MỞ ĐẦU

Ta biết rằng số nguyên tố và đặt biệt là số nguyên tố lớn đóng vai trò rất quan trọng trong nhiều lĩnh vực về an toàn – bảo mật thông tin như: Trong hạ tần cơ sở khóa công khai, trong các hệ mật mã RSA, Elgamal, trong xác thực và chữ ký điện tử, vv. Tuy nhiên một vấn đề đặt ra là làm thế nào để khẳng định mốt số nguyên dương N nào đó là số nguyên tố hay hợp số?

Ta biết rằng số nguyên tố là một số chia hết cho một và chính nó. Như vậy số nguyên tố là một số nguyên (dương) lẻ? Vì nếu nó là số chẵn thì nó sẽ chia hết cho 2 và như vậy, nó không phải là một số nguyên tố. Vậy số nguyên tố (trừ số 2) là một số lẻ. Nhưng số lẻ chưa hẳn đã là số nguyên tố. ví dụ:

Số 9 là số lẻ nhưng không phải là số nguyên tố.

Số 15 là số lẻ nhưng cũng không phải là số nguyên tố. Vì (15 = 3.5) là hợp số.

Số 2047 cũng không phải là số nguyên tố vì (2047 = 23.89)

Vậy cho một số nguyên lẻ n bất kỳ làm thể nào để nhận biết được n là số nguyên tố (chỉ chia hết cho 1 và chính nó)? Đến nay đã có nhiều tài liệu nghiên cứu về vấn đề này chẳng hạn trong Knut(2004):" The Artof Programming Computer" (Tập II- 2004): " chẳng hạn thuật toán sàng bình phương " (Quadratic Sieve algerithin) [1: Alfred J. Menezes, Pau C.van Oorschot and Scott A.Vanstone (1998) CRC press: Boca Raton. Newyork, London and Tokyo(1998)].

Chương 1. CÁC KHÁI NIỆM CƠ BẢN

1.1. MỘT SỐ KHÁI NIỆM TRONG SỐ HỌC, ĐẠI SỐ

1.1.1. Khái niệm trong số học

1). Ký hiệu chia hết

Cho a và b là hai số nguyên dương.

Số a chia hết cho số b ký hiệu là a ⋮ b ⇔ Tồn tại n ∈ N sao cho:

$$a = b * n$$

Khi đó người ta nói b là ước của a và ký hiệu: b | a.

2). Ước số chung lớn nhất

Cho a và b là hai số nguyên dương.

Ước số chung lớn nhất của a và b là số tự nhiên m lớn nhất sao cho m | a và m | b. Khi đó ký hiệu là gcd(a, b) = m.

3). Hai số nguyên tố cùng nhau

Cho a và b là hai số nguyên dương.

Số a và số b được gọi là 2 nguyên tố cùng nhau ⇔ gcd(a, b) = 1.

4). Đồng dư modulo

Cho n ∈ N, n ≠ 0 và a, b ∈ * Zn .

Ký hiệu a ≡ b (mod n) nghĩa là a đồng dư với b theo mod n

⇔ Tồn tại số nguyên k ∈ * Zn sao cho a = b + k * n

Tức là (a - b) = k * n, như vậy n | (a - b).

5). Một số tính chất của đồng dư modulo

(a ± b) (mod n) ≡ [(a mod n) ± (b mod n)] (mod n)

(a * b) (mod n) ≡ [(a mod n) * (b mod n)] (mod n)

1.1.2. Khái niệm trong đại số

1). Khái niệm nhóm

Nhóm là một cặp (G, *), trong đó G là tập hợp khác rỗng, * là phép toán hai ngôi trên G thoả mãn ba điều kiện sau:

1. Phép toán có tính kết hợp:

$(x * y) * z = x * (y * z)$ với mọi $x, y, z \in G$.

2. Có phần tử phần tử trung lập $e \in G$:

$x * e = e * x = x$ với mọi $x \in G$.

3. Với mọi $x \in G$, có phần tử nghịch đảo $x' \in G$:

$x * x' = x' * x = e$

2). Nhóm con

Cho G là một Nhóm, cho $S \subset G$ và $S \neq \phi$. S được gọi là Nhóm con của G nếu:

1/. Phần tử trung lập e của G nằm trong S.

2/. S khép kín đối với luật hợp thành trong G (tức là $x * y \in S$ với mọi $x, y \in S$).

3/. S khép kín đối với phép lấy nghịch đảo trong G (tức $x^{-1} \in S$ với mọi $x \in S$).

3). Nhóm Cyclic

a). Khái niệm Nhóm Cyclic

Nhóm (G, *) được gọi là Nhóm Cyclic nếu nó được sinh ra bởi một trong các phần tử của nó.

Tức là có phần tử $g \in G$ mà với mỗi $a \in G$, đều tồn tại số $n \in N$ để $g^n = g * g * \ldots * g = a$ (Chú ý: $g * g * \ldots * g$ là $g * g$ với n lần).

Khi đó g được gọi là phần tử sinh hay phần tử nguyên thuỷ của nhóm G.

Nói cách khác: G được gọi là Nhóm Cyclic nếu tồn tại g ∈ G sao cho mọi phần tử trong G đều là một luỹ thừa nguyên nào đó của g.

Ví dụ: Nhóm (Z + , +) gồm các số nguyên dương là Cyclic với phần tử sinh g = 1.

b). Cấp của Nhóm Cyclic:

Cho (G, *) là Nhóm Cyclic với phần tử sinh g và phần tử trung lập e. Nếu tồn tại số tự nhiên nhỏ nhất n mà $g^n = e$, thì G sẽ chỉ gồm có n phần tử khác nhau: e, g, g^2 , g^3 , . . . , g^{n-1} . Khi đó G được gọi là nhóm Cyclic hữu hạn cấp n.

Nếu không tồn tại số tự nhiên n để $g^n = e$, thì G có cấp ∞.

Ví dụ: (Z^+ , +) gồm các số nguyên dương là Cyclic với phần tử sinh g = 1, e = 0. Đó là Nhóm Cyclic vô hạn, vì không tồn tại số tự nhiên n để $g^n = e$,

c). Cấp của một phần tử trong Nhóm Cyclic:

Phần tử α ∈ G được gọi là có cấp d, nếu d là số nguyên dương nhỏ nhất sao cho $\alpha^d = e$, trong đó e là phần tử trung lập của G.

Như vậy phần tử α có cấp 1, nếu α = e.

4). Tập Z_n và Z^*

$Z_n = \{0, 1, 2, .. . , n - 1\}$. Tức Z_n là tập các số nguyên không âm < n.

Tập này cùng với phép cộng lập thành Nhóm Cyclic có phần tử sinh là 1. Đó là Nhóm hữu hạn có cấp n.

Zn * = $\{e \in Z_n$, e là nguyên tố cùng nhau với n$\}$. Tức là e # 0.

Đó là tập các số nguyên dương < n, nhưng nguyên tố cùng nhau với n. được gọi là tập Thặng dư thu gọn theo mod n, lập thành một Nhóm với phép nhân mod n.

ϕ(n) là số các phần tử của tập z_n^* .

5). Một số kết quả

Những kết quả sau đã được chứng minh, nhắc lại để sử dụng:

* Định lý Lagrange: Cho G là nhóm Cấp n và g∈ G. Khi đó cấp của g là ước của n.

* Hệ quả: Giả sử g∈ z_n^* có Cấp m thì m là ước của ϕ (n).

Nếu b ∈ * z_n^* thì $b^{\phi(n)} \equiv 1 \pmod n$.

Nếu p là số nguyên tố thì ϕ (p) = p -1.

Do đó với mọi b ∈ * z_p^* (tức b nguyên tố với p) thì $b^{\phi(p)} \equiv 1 \pmod n$ hay $b^{p-1} \equiv 1 \pmod n$.

* Định lý: Nếu p là số nguyên tố thì z_p^* là Nhóm Cyclic.

Chú ý: Phần tử α ∈ z_n^* có cấp d nếu d là số nguyên dương nhỏ nhất sao cho α^d = e trong z_n^* , tức là $\alpha^d \equiv 1 \pmod n$.

6). Khái niệm Logarit rời rạc

Cho p là số nguyên tố, α là phần tử nguyên thuỷ của Z_p, β∈ z_n^* .

Logarit rời rạc chính là việc giải phương trình $x = \log_\alpha \beta \pmod p$ với ẩn x.

Hay phải tìm số x duy nhất sao cho: $\alpha x \equiv \beta \pmod p$.

- Bổ đề: Nếu (a, n) = 1 thì tồn tại $a^{-1} \in Z_n$ thoả mãn $a * a^{-1} \equiv 1 \pmod n$.

- Định lý (Euler tổng quát): Nếu (a, n) = 1 thì $a^{\phi(n)} \bmod n = 1$.

- Hệ quả: Với p là một số nguyên tố và (a, p) = 1 thì $a^{p-1} \pmod p = 1$.

1.2. MỘT SỐ THUẬT TOÁN

1.2.1. Thuật toán tính ước chung lớn nhất

Ký hiệu Z là tập hợp các số nguyên, Z = { ..., -2, -1, 0, 1, 2, ...}, và Z^+ là tập hợp các số nguyên không âm, Z^+ = {0, 1, 2, ...}. Trong mục này sẽ nhắc lại một số kiến thức về số học của các số nguyên cần cho việc trình bày lý thuyết mật mã. Vì để luận văn không quá dài dòng, các kiến thức sẽ được nhắc đến chủ yếu là các khái niệm, các mệnh đề sẽ được sử dụng, v.v..., còn các phần chứng minh sẽ được lược bỏ.

Tập hợp Z là đóng kín đối với các phép cộng, trừ và nhân, nhưng không đóng kín đối với phép chia: chia một số nguyên cho một số nguyên không phải bao giờ cũng được kết quả là một số nguyên! Vì vậy, trường hợp chia hết, tức khi chia số nguyên a cho số nguyên b được thương là một số nguyên q, a = b.q, có một ý nghĩa đặc biệt. Khi đó, nói a chia hết cho b, a là bội số của b, b là ước số của a, và ký hiệu là b|a. Dễ thấy ngay rằng số 1 là ước số của mọi số nguyên bất kỳ, số 0 là bội số của mọi số nguyên bất kỳ, mọi số nguyên a đều là ước số, đồng thời là bội số của chính nó.

Định lý 1.2

Nếu b > 0 và b|a thì gcd(a,b) = b.

Nếu a = bq +r thì gcd(a,b) = gcd(b,r).

Một số nguyên m được gọi là bội số chung của a và b nếu a|m và b|m. Số m được gọi là bội số chung bé nhất của a và b và được ký hiệu là lcm(a,b), nếu m>0, m là bội số chung của a và b, và mọi bội số chung của a và b đều là bội của m. Ví dụ lcm(14,21) = 42.

Với hai số nguyên dương a và b bất kỳ ta có quan hệ lcm(a,b).gcd(a,b) = a.b.

Từ định lý 1.2 ta suy ra thuật toán sau đây thực hiện việc tìm ước số chung lớn nhất của hai số nguyên bất kỳ:

Thuật toán Euclide tìm ước số chung lớn nhất:

INPUT: hai số nguyên không âm a và b, với a≥b.

OUTPUT: ước số chung lớn nhất của a và b

1. Trong khi còn b>0, thực hiện:

1.1. đặt r ← a mod b,a ← b, b ← r.

2. Cho ra kết quả (a).

Ví dụ: Dùng thuật toán Euclide tìm gcd(4864, 3458), lần lượt được các giá trị gán cho các biến a, b và r như sau:

Bảng 1.2.1. Mô tả quá trình tính toán của thuật toán Euclid

	a	b	c
4864 = 1.3458 + 1406	4864	3458	
3458 = 2.1406 + 646	3458	1406	1406
1406 = 2.646 + 114	1406	646	646
646 = 5.114 + 76	646	114	114
114 = 1.76 +38	114	76	76
76 = 2.38 +0	76	38	38
	38	0	0

Cho hai số nguyên bất kỳ a và b, b > 1. Thực hiện phép chia a cho b ta sẽ được hai số q và r sao cho

a = b.q + r, 0 ≤ r <b.

Số q được gọi là số thương của phép chia a cho b, ký hiệu a div b và số r được gọi là số dư của phép chia a cho b, ký hiệu a mod b. Ví dụ: 25 div 7 = 3 và 25 mod 7 = 4, -25 div 7 = -4 và – 25 mod 7 = 3.

Một số nguyên d được gọi là ước số chung của hai số nguyên a và b nếu d/a và d/b. Số nguyên d được gọi là ước số chung lớn nhất của a và b nếu d>0, d là ước số chung của a và b, và mọi ước chung của a và b đều là ước số của d. Ký hiệu ước số chung lớn nhất của a và b là gcd (a,b). Ví dụ gcd (12, 18) = 6, gcd (-18, 27) = 3.

Dễ thấy rằng với mọi số nguyên dương a ta có gcd (a, 0) = a, ta cũng sẽ qui ước xem rằng gcd (0,0) = 0.

Một số nguyên a > 1 được gọi là số nguyên tố, nếu a không có ước số nào ngoài 1 và chính a; và được gọi là hợp số, nếu không phải là nguyên tố. Ví dụ các số 2, 3, 5, 7 là số nguyên tố; các số 6, 8, 10, 12, 14, 15 là hợp số. Hai số a và b được gọi là nguyên tố với nhau, nếu chúng không có ước số chung nào khác 1, tức là nếu gcd (a,b) = 1. Một số nguyên n > 1 bất kỳ đều có thể viết dưới dạng:

$$n = P_1^{a_1}.P_2^{a_2} \dots P_k^{a_k}$$

Trong đó p_1, p_2, ...,p_k là các số nguyên tố khác nhau, a_1, a_2, ...a_k là các số mũ nguyên dương. Nếu không kể thứ tự các thừa số nguyên tố thì dạng biểu diễn đó là duy nhất, ta gọi đó là dạng khai triển chính tắc của n. Ví dụ dạng khai triển chính tắc của 1800 là $2^3 3^2 5^2$.

Các số nguyên tố và các vấn đề về số nguyên tố có một vai trò quan trọng trong số học và trong ứng dụng vào lý thuyết mã hóa, sẽ xét riêng trong chương sau.

Thuật toán cho kết quả: gcd (4864, 3458) = 38.

Biết rằng nếu gcd (a,b) = d, thì phương trình bất định

a.x + b.y = d

có nghiệm nguyên (x,y), và một nghiệm nguyên (x,y) như vậy có thể tìm được bởi thuật toán Euclide mở rộng như sau:

Thuật toán Euclide mở rộng:

INPUT: hai số nguyên không âm a và b với a ≥b.

OUTPUT: d = gcd (a,b) và hai số x, y sao cho a.x +b.y = d

1. Nếu b = 0 thì đặt d ← a, x ← 1, y ← 0 và cho ra (d, x, y).

2. Đặt x2 = 1, x1 = 0, y2 = 0, y1 = 1.

3. Trong khi còn b > 0 thực hiện:

3.1 q ← a div b, r ← a mod b, x ← $x_2 – qx_1$, y ← $y_2 – qy_1$.

3.2 a ← b, b ← r, x_2 ← x_1, x_1 ← x, y_2 ← y_1 và y_1 ← y.

4. Đặt d ← a, x ← x2, y ← y2, và cho ra kết quả (d, x, y)

Ví dụ: Dùng thuật toán Euclide mở rộng cho các số a = 4864 và b = 3458, lần lượt được các giá trị sau đây cho các biến a, b, q, r, x, y, x_1, x_2, y_1, y_2 (sau mỗi chu trình thực hiện hai lệnh 3.1 và 3.2).

Bảng 1.2.2. Mô tả quá trình tính toán của thuật toán Euclid mở rộng

a	b	q	r	x	y	x1	x2	y1	y2
4864	3458					0	1	1	0
3458	1406	1	1406	1	-1	1	0	-1	1
1406	646	2	646	-2	3	-2	1	3	-1
646	114	2	114	5	-7	5	-2	-7	3
114	76	5	76	-27	38	-27	5	38	-7
76	38	1	38	32	-45	32	-27	-45	38
38	0	2	0	-91	128	-91	32	128	-45

Dễ thử lại rằng sau mỗi lần thực hiện chu trình gồm hai lệnh 3.1 và 3.2 các giá trị x, y, r thu được luôn thỏa mãn 4864. x + 3458.y = r, và do đó khi kết thúc các vòng lặp (ứng với giá trị b = 0), thực hiện tiếp lệnh 4 ta được kết quả d = 38 và y = -45, cặp số (32, -45) thỏa mãn: 4864.32 + 3458. (-45) = 38.

1.2.2. Thuật toán tính phần tử nghịch đảo theo Modulo

*** Định nghĩa:**

Cho a ∈ Z_n , nếu tồn tại b ∈ Z_n sao cho a.b ≡ 1 (mod n), ta nói b là phần tử nghịch đảo của a trong Z_n và ký hiệu a^{-1} .

Một phần tử có phần tử nghịch đảo, gọi là khả nghịch.

*** Định lý:** gcd (a, n) = 1 ⇔ Phần tử a ∈ Z_n có phần tử nghịch đảo.

Chứng minh:

Nếu $a.a^{-1} \equiv 1 \pmod n$ thì $a.a^{-1} = 1 + kn \leftrightarrow a.a^{-1} - kn = 1 \rightarrow (a, n) = 1$.

Nếu (a, n) = 1, ta có $a.a^{-1} + kn = 1 \rightarrow a.a^{-1} = 1 + kn$, do đó $a.a^{-1} \equiv 1 \pmod n$.

*** Hệ quả:** Mọi phần tử trong Z_n^* đều có phần tử nghịch đảo.

*** Tìm phần tử nghịch đảo bằng Thuật toán Euclid mở rộng.**

Input: $a \in Zn$, n

Output: Phần tử nghịch đảo của a.

```
Procedure Invert(a, n);
Begin
    g0:=n; g1:=a; u0:=1; u1:=0; v0:=0; v1:=1;
    i:=1;
    while gi ≠ 0 do
    begin
        y:= gi-1 div gi; gi+1 := gi+1 - y.gi;
        ui+1 := ui+1 - y.ui; vi+1 := vi+1 - y.vi;
        i:= i+1;
        end;
        t := vi+1;
        if t > 0 then a-1 := t else a-1:=t + n;
End;
```

Ví dụ: Tìm phần tử nghịch đảo của 3 trong Z_7

Tức là phải giải phương trình $3.x \equiv 1 \pmod 7$, x sẽ là phần tử nghịch đảo của 3.

I	g_i	u_i	v_i	y
1	7	1	0	
1	3	0	1	2
2	1	1	-2	3
3	0			

Vì t = v2 = -2 < 0 do đó $x = a^{-1} := t + n = -2 + 7 = 5$.

Vậy 5 là phần tử nghịch đảo của 3 trong Z_7

Chú ý

Định lý (Euler tổng quát): Nếu (a, n) = 1 thì $a^{\phi(n)} \bmod n = 1$

Hệ quả: Nếu p là số nguyên tố và (a, p) = 1, thì $a^{p-1} \pmod p = 1$

1.2.3. Thuật toán phân tích một số ra các thừa số nguyên tố

Đặt vấn đề: Có một khối lượng khổng lồ các tài liệu về các thuật toán phân tích thừa số. Tuy vậy, trong phần này chỉ đưa ra một cái nhìn khái quát bao gồm việc thảo luận sơ lược về các thuật toán phân tích thừa số tốt nhất hiện thời và cách sử dụng chúng trong thực tế. Ba thuật toán hiệu quả nhất trên các số thật lớn là: sàng bậc hai, thuật toán đường cong Elliptic và sàng trường số. Các thuật toán nổi tiếng khác bao gồm: phương pháp p và thuật toán (p – 1) của Pollard, thuật toán (p + 1) của Williams, thuật toán liên phân số và dĩ nhiên là cả phép chia thử.

1.2.3.1. Phương pháp Pollard

Thuật toán (p – 1) của Pollard (đưa ra vào năm 1975) là một ví dụ về một thuật toán đơn giản khi được áp dụng đối với các số nguyên lớn.

Inputs: n, số nguyên cần phân tích; và f(x), hàm tạo số giả ngẫu nhiên modulo n.

Output: một nhân tử không tầm thường (khác 1 và n) của n, hoặc không thực hiện được.

$x \leftarrow 2, y \leftarrow 2; d \leftarrow 1$
While d = 1:
$x \leftarrow f(x)$
$y \leftarrow f(f(y))$
$d \leftarrow GCD(\|x - y\|, n)$
If d = n, return không thực hiện được
Else return d.

Hình 1. 2: Thuật toán phân tích thừa số p - 1

Chú ý rằng thuật toán có thể không tìm thấy nhân tử và trả về kết quả không thực hiện được với một hợp số n. Trong trường hợp này sử dụng hàm f(x) khác và thử lại. Thuật toán cũng không làm việc khi n là số nguyên tố, trong trường hợp này d sẽ luôn là 1.

Đối với hàm f, chúng ta chọn đa thức với hệ số nguyên. một trong những dạng chung nhất đó là: $f(x) = x^2 + c \bmod n$, $c \neq 0, -2$.

Ví dụ: Cho n = 8051 và f(x) = x2 + 1 mod 8051

i	xi	yi	$\gcd(\|x_i - y_i\|, 8051)$
1	5	26	1
2	26	7474	1
3	677	871	97

97 là một nhân tử không tầm thường của 8051. Nhân tử còn lại là thương của phép chia n cho 97 bằng 83.

1.2.3.2. Thuật toán đường cong Elliptic

Thuật toán đường cong Elliptic mạnh hơn (được Lenstra xây dựng vào những năm 80) trên thực tế là sự tổng quát hóa của phương pháp p - 1. Ta sẽ không thảo luận về mặt lý thuyết ở đây mà chỉ nhấn mạnh rằng, thành công của phương pháp đường cong Elliptic tuỳ thuộc vào một tình huống tương tự: một số nguyên "gần

với" p chỉ có các thừa số nguyên tố bộ. Trong khi phương pháp p - 1 phụ thuộc vào quan hệ trong Z_p thì phương pháp đường cong Elliptic phụ thuộc vào các Nhóm xác định trên các đường cong Elliptic theo modulo n.

1.2.3.3. Thuật toán kiểu Las Vegas

Thuật toán được xây dựng trên cơ sở một số nguyên tố nhất định liên quan tới các căn bậc hai của 1 theo modulo n, trong đó n = p * q là tích của hai số nguyên tố lẻ phân biệt.

1. Chọn w là ngẫu nhiên và $1 \leq w \leq n - 1$

2. Tính x = gcd (w, n)

3. Nếu $1 < x < n$ thì thoát (thành công: x = p hoặc x = q)

4. Tính a = A (b)

5. Viết $ab - 1 = 2^s r$, r lẻ

6. Tính $v = w^r \bmod n$

7. Nếu $v = 1 \pmod n$ thì thoát (không thành công)

8. Trong khi $v \neq 1 \pmod n$ thì thực hiện

9. $v_0 = v$

10. $v = v^2 \bmod n$

11. Nếu $v_0 \equiv -1 \pmod n$ thì thoát (không thành công)

    ngược lại

    tính $x = \gcd (v_0 + 1, n)$

    (thành công: x = p hoặc x = q)

Hình 1. 3: Thuật toán phân tích thừa số (cho trước số mũ giải mã

1.2.3.4. Thuật toán phân tích cổ điển

Hiện nay người ta chưa có cách nào tính trực tiếp p, q hữu hiệu từ n trừ khi biết $\phi(n)$. Vì khi biết $\phi(n)$, ta có:

$$p * q = n$$

$$\phi(n) = (p - 1) * (q - 1)$$

$$\Rightarrow p + q = n - \phi(n) + 1$$

Dựa vào định lý Vi-ét p và q là nghiệm của phương trình:

$$x^2 - (n - \phi(n) + 1) * x + n = 0$$

Giải phương trình này ta dễ dàng tìm được p và q.

Ví dụ: n = 84773093, $\phi(n) = 84754668 \Rightarrow$ q = 9539, p = 8887.

Ngoài ra theo cách cổ điển, sử dụng thuật toán :

```
// Input: n
// Output:     p thoả mãn p | n
//
//                  0 trong trường hợp ngược lại
For (int i = 3; i <= sqrt (n); i+ = 2)
if (n % i) return i;
return 0;
```

Hình 1. 4: Thuật toán phân tích cổ điển

Trong thuật toán trên vòng lặp là $(n^{1/2}/ 2)$. Nếu n có 512 bit, giá trị lớn nhất của n là 2^{512}. Nếu 1 máy tính thực hiện 10^6 chỉ lệnh trong 1 giây thì thời gian thực hiện là:

$T = n^{1/2} / 2 \leq 2^{1/2 * 512} / 2 = 2^{256} / 2 = 2^{255}$(giây) $\cong 2^{238}$ (ngày) $\cong 2^{230}$ năm.

(1 ngày = 60 * 60 * 24 = 86400 giây $\cong 2^{17}$ giây.

1 năm = 30 * 12 * 86400 giây = 31104000 giây ≅ 2^{25} giây.)

Nếu kẻ giả mạo muốn tìm p, q theo cách này thì đây là điều không tưởng.

1.3. ĐỘ PHỨC TẠP TÍNH TOÁN

1.3.1. Khái niệm về độ phức tạp tính toán

Lý thuyết thuật toán và các hàm số tính được ra đời từ những năm 30 của thế kỷ 20 đã đặt nền móng cho việc nghiên cứu các vấn đề "tính được", "giải được" trong toán học, đưa đến nhiều kết quả rất quan trọng và lý thú. Nhưng từ cái "tính được" một cách trừu tượng, hiểu theo nghĩa tiềm năng, đến việc tính được trong thực tế của khoa học tính toán bằng máy tính điện tử, là cả một khoảng cách rất lớn. Vấn đề là do ở chỗ những đòi hỏi về không gian vật chất và về thời gian để thực hiện các tiến trình tính toán nhiều khi vượt quá xa những khả năng thực tế. Từ đó, vào khoảng giữa những năm 60 (của thế kỷ trước), một lý thuyết về độ phức tạp tính toán bắt đầu được hình thành và phát triển nhanh chóng, cung cấp cho chúng ta nhiều hiểu biết sâu sắc về bản chất phức tạp của các thuật toán và các bài toán, cả những bài toán thuần túy lý thuyết đến những bài toán thường gặp trong thực tế. Sau đây giới thiệu sơ lược một số khái niệm cơ bản và vài kết quả sẽ được dùng đến của lý thuyết đó.

Trước hết, hiểu độ phức tạp tính toán (về không gian hay về thời gian) của một tiến trình tính toán là số ô nhớ được dùng hay số các phép toán sơ cấp được thực hiện trong tiến trình tính toán đó.

Dữ liệu đầu vào đối với một thuật toán thường được biểu diễn qua các từ trong một bảng ký tự nào đó. Độ dài của một từ là số ký tự trong từ đó.

Cho một thuật toán A trên bảng ký tự $\sum$ (tức có đầu vào là các từ trong $\sum$). Độ phức tạp tính toán của thuật toán A được hiểu là một hàm số $f_A(n)$ sao cho với mỗi số n, $f_A(n)$ là số ô nhớ, hay số phép toán sơ cấp tối đa mà A cần để thực hiện tiến trình tính toán của mình trên các dữ liệu vào có độ dài ≤ n. Ta nói thuật toán A có độ phức tạp thời gian đa thức, nếu có một đa thức P(n) sao cho với mọi n đủ lớn ta có $f_A(n) \leq P(n)$, trong đó $f_A(n)$ là độ phức tạp tính toán theo thời gian của A.

Về sau khi nói đến các bài toán, ta hiểu đó là các bài toán quyết định, mỗi bài toán P như vậy được xác định bởi:

- Một tập các dữ liệu vào I (trong một bảng ký tự $\sum$ nào đó).

- Một câu hỏi Q trên các dữ liệu vào, sao cho với mỗi dữ liệu vào $x \in I$, câu hỏi Q có một trả lời đúng hoặc sai.

Bài toán quyết định P là giải được, nếu có thuật toán để giải nó, tức là thuật toán làm việc có kết thúc trên mọi dữ liệu vào các bài toán, và cho kết quả đúng hoặc sai tùy theo câu hỏi Q trên dữ liệu đó có trả lời đúng hoặc sai. Bài toán P là giải được trong thời gian đa thức, nếu có thuật toán giải nó với độ phức tạp thời gian đa thức. Sau đây là vài ví dụ về các bài toán quyết định:

Bài toán SATISFIABILYTY (viết tắt là SAT):

- Mỗi dữ liệu vào là một công thức F của logic mệnh đề, được viết dưới dạng hội chuẩn tắc, tức dạng hội của một số các "clause".

- Câu hỏi là: công thức F có thỏa được hay không?

Bài toán CLIQUE:

- Mỗi dữ liệu vào là một graph G và một số nguyên k.

- Mỗi câu hỏi là: Graph G có một clique với $\geq$ k đỉnh hay không? (một clique của G là một graph con đầy đủ của G).

Bài toán KNAPSACK:

- Mỗi dữ liệu là một bộ n + 1 số nguyên dương $I = (s_1, ...s_n; T)$.

- Câu hỏi là: có hay không một vectơ Boole $(x_1, ...,x_n)$ sao cho

$\sum_{i=1}^{n} xi\text{-}si = T$?

(vectơ Boole là vectơ có các thành phần là 0 hoặc 1)

Bài toán thặng dư bậc hai:

- Mỗi dữ liệu gồm hai số nguyên dương (a, n).

- Câu hỏi là: a có là thặng dư bậc hai theo mod n hay không?

Bài toán hợp số:

- Mỗi dữ liệu là một số nguyên dương N.

- Câu hỏi: N là hợp số không? Tức có hay không hai số m, n >1 sao cho N = m.n?

Tương tự, nếu đặt câu hỏi là "N là số nguyên tố hay không?" thì ta được **bài toán số nguyên tố.**

Đối với tất cả các bài toán kể trên, trừ bài toán hợp số và số nguyên tố, cho đến nay người ta đều chưa tìm được thuật toán giải chúng trong thời gian đa thức.

1.3.2. Lớp phức tạp

Xét một vài lớp các bài toán được xác định theo độ phức tạp tính toán của chúng. Trước hết, định nghĩa P là lớp tất cả các bài toán có thể giải được bởi thuật toán đơn định trong thời gian đa thức.

Giả sử cho hai bài toán A và B với các tập dữ liệu trong hai bảng ký tự tương ứng là $\sum_1$ và $\sum_2$. Một thuật toán f: $\sum^*_1 \rightarrow \sum^*_2$ được gọi là một phép quy dẫn bài toán A về bài toán B, nếu nó biến mỗi dữ liệu x của bài toán A thành một dữ liệu f(x) của bài toán B, và sao cho câu hỏi của A trên x có trả lời đúng khi và chỉ khi câu hỏi của B trên f(x) cũng có trả lời đúng. Ta nói bài toán A quy dẫn được về bài toán B trong thời gian đa thức, và ký hiệu A α B, nếu có thuật toán f với độ phức tạp thời gian đa thức qui dẫn bài toán A về bài toán B. Dễ thấy rằng, nếu A α B và B $\in$ P, thì cũng có A $\in$ P.

Một lớp quan trọng các bài toán đã được nghiên cứu nhiều là lớp các bài toán khá thường gặp trong thực tế nhưng cho đến nay chưa có khả năng nào chứng tỏ là chúng có thể giải được trong thời gian đa thức. Đó là lớp các bài toán NP-đầy đủ được định nghĩa sau đây:

Cùng với khái niệm thuật toán tất định thông thường (có thể mô tả chính xác chẳng hạn bởi máy Turing tất định), xét khái niệm thuật toán không đơn định với một ít thay đổi như sau: nếu đối với máy Turing tất định, khi máy đang ở một trạng thái q và đang đọc một ký tự a thì cặp (q, a) xác định duy nhất một hành động kế tiếp của máy, còn đối với máy Turing không đơn định, qui ước rằng (q,

a) xác định không phải duy nhất mà là một tập hữu hạn các hành động kế tiếp; máy có thể thực hiện trong bước kế tiếp một trong các hành động đó.

Như vậy, đối với một dữ liệu vào x, một thuật toán không đơn định được (được xác định chẳng hạn bởi một máy Turing không đơn định) không phải chỉ có một tiến trình tính toán duy nhất, mà có thể có một số hữu hạn những tiến trình tính toán khác nhau.

Ta nói thuật toán không đơn định A chấp nhận dữ liệu x, nếu với dữ liệu vào chấp nhận (tức với kết quả đúng).

Một bài toán P được gọi là giải được bởi thuật toán không đơn định trong thời gian đa thức nếu có một thuật toán không đơn định A và một đa thức p(n) sao cho với mọi dữ liệu vào x có độ dài n, x $\in$ P (tức câu hỏi của P có trả lời đúng trên x) khi và chỉ khi thuật toán A chấp nhận x bởi một tiến trình tính toán có độ phức tạp thời gian $\leq$ p(n). Ta ký hiệu lớp tất cả với các bài toán giải được bởi thuật toán không đơn định trong thời gian đa thức là NP.

Người ta đã chứng tỏ được rằng tất cả những bài toán trong các ví dụ kể trên và rất nhiều các bài toán tổ hợp thường gặp khác đều thuộc lớp NP, dù rằng hầu hết chúng đều chưa được chứng tỏ là thuộc P. Một bài toán P được gọi là NP-đầy đủ, nếu P $\in$ NP và với mọi Q $\in$ NP đều có Q α P.

Lớp NP có một số tính chất sau đây:

1) P $\subseteq$ NP

2) Nếu A α Bvà B $\in$ NP, thì A $\in$ NP

3) Nếu A, B $\in$ NP, A α B, và A là NP- đầy đủ, thì B cũng là NP-đầy đủ

4) Nếu có A sao cho A là NP-đầy đủ và A $\in$ P, thì P=NP

Từ các tính chất đó có thể xem rằng trong lớp NP, P là lớp con các bài toán "dễ" nhất, còn các bài toán NP đầy đủ là các bài toán "khó" nhất; nếu có ít nhất một bài toán NP đầy đủ được chứng minh là thuộc P, thì lập tức suy ra P = NP, dù rằng cho đến nay tuy đã có rất nhiều cố gắng nhưng toán học vẫn chưa tìm được con được nào hy vọng đi đến giải quyết vấn đề [P = NP?], thậm chí vấn đề

đó còn được xem là một trong 7 vấn đề khó nhất của toán học trong thiên niên kỷ mới!

1.3.3. Hàm một phía và cửa sập một phía

Khái niệm độ phức tạp tính toán cung cấp một cách tiếp cận mới đối với vấn đề bí mật trong các vấn đề bảo mật và an toàn thông tin. Dù ngày nay đã có những máy tính điện tử có tốc độ tính toán cỡ hàng tỷ phép tính một giây, nhưng với những thuật toán có độ phức tạp tính toán cỡ $f(n) = 2^n$, thì ngay với những dữ liệu có độ dài khoảng n = 1000, việc thực hiện các thuật toán đó đã không thể xem là khả thi, vì nó đòi hỏi thực hiện khoảng 10^{300} phép tính! Như vậy, một giải pháp mã hóa chẳng hạn có thể xem là có độ bảo mật cao, nếu để giải mã cần phải thực hiện một tiến trình tính toán có độ phức tạp rất lớn. Do đó, việc phát hiện và sử dụng các hàm số có độ phức tạp tính toán rất lớn là có ý nghĩa hết sức quan trọng đối với việc xây dựng các giải pháp về mã hóa và an toàn thông tin.

Hàm số số học y = f(x) được gọi là hàm một phía (one-way function), nếu việc tính thuận từ x ra y là "dễ", nhưng việc tính ngược lại từ y tìm lại x là rất "khó", ở đây các tính từ "dễ" và "khó" không có các định nghĩa chính xác mà được hiểu một cách thực hành, có thể hiểu chẳng hạn dễ là tính được trong thời gian đa thức (với đa thức bậc thấp), còn khó là không tính được thời gian đa thức! Thực tế thì cho đến hiện nay, việc tìm và chứng minh một hàm số nào đó là không tính được trong thời gian đa thức còn là việc rất khó, cho nến "khó" thường khi chỉ được hiểu một cách đơn giản chưa tìm được thuật toán tính nó trong thời gian đa thức! Với cách hiểu tương đối như vậy về "dễ" và "khó", người ta đã đưa ra một số thí dụ sau đây về các hàm một phía.

Ví dụ 1:

Cho p là một số nguyên tố, và a là một phần tử nguyên thủy mod p. Hàm số $y = a^x \bmod p$ (từ Z^*_p vào Z^*_p) là một hàm một phía, vì hàm ngược của nó, tính từ y tìm x mà ta ký hiệu $x = \log_a(y)$, là một hàm có độ phức tạp tính toán rất lớn.

Ví dụ 2:

Cho n=p.q là tích của hai số nguyên tố lớn. Hàm số $y=x^2 \bmod n$ (từ Z_n vào Z_n) cũng được xem là một hàm một phía.

Ví dụ 3:

Cho n = p.q là tích của hai số nguyên tố lớn, và a là một số nguyên sao cho gcd (a, $\phi(n)$) = 1. Hàm số $y = x^a \bmod n$ (từ Z_n vào Z_n) cũng là một hàm một phía, nếu giả thiết là biết n nhưng không biết p, q.

Hàm y = f(x) được gọi là hàm cửa sập một phía (trapdoor one-way function), nếu việc tính thuận từ x ra y là "dễ", việc tính ngược từ y tìm lại x là rất "khó", nhưng có một cửa sập z để với sự trợ giúp của cửa sập z thì việc tính x từ y và z lại trở thành dễ.

Ví dụ 4 (tiếp tục ví dụ 3):

Hàm số $y = x^a \bmod n$ khi biết p và q là hàm cửa sập một phía. Từ x tính y là dễ, từ y tìm x (nếu chỉ biết n,a) là rất khó, nhưng vì biết p và q nên biết $\phi(n) = (p1)(q-1)$, và dùng thuật toán Euclide mở rộng, tìm được b sao cho $a.b \equiv 1 \pmod{\alpha(n)}$, từ đó dễ tính được $x = y^b \bmod n$. Ở đây có thể xem b là cửa sập.

1.4 Kết luận chương

Chương 1 này tìm hiểu tông quan các khái niệm trong số học và đại số, Tìm hiểu một số thuật toán, có thể liệt kê thuật toán tính ước chung lớn nhất, thuật toán nghịch đảo theo Modulo, Thuật toán phân tích một số ra các thừa số nguyên tố, nhưng thuật toán này góp phần xây dựng chương trình vào chương tiếp theo. Tìm hiểu về các lớp phức tạp trong tính toán, các khái niệm phức tạp tính toán, lớp phức tạp, hàm một phía cửa sập một phía.

CHƯƠNG II: MỘT SỐ PHƯƠNG PHÁP KIỂM TRA SỐ NGUYÊN TỐ

2.1. SỐ NGUYÊN TỐ

2.1.1. Khái niệm về số nguyên tố

Số nguyên tố: là số tự nhiên chỉ có ước số là 1 và chính nó.

Ví dụ: Các số nguyên tố 79, 83, 89, 97

2.1.2 Tính chất của số nguyên tố

Ký hiệu "b|a" nghĩa là b là ước của a, ký hiệu a ⋮ b nghĩa là a chia hết cho b

1). <u>Ước tự nhiên khác 1 nhỏ nhất của một số tự nhiên là số nguyên tố .</u>

Chứng minh: Giả sử d|a; d nhỏ nhất; d ≠1. Nếu d không nguyên tố ⇒ d = d1.d2; d1, d2 > 1 ⇒ d1|a với d1 < d mâu thuẫn với d nhỏ nhất. Vậy d là nguyên tố.

2). Cho p là số nguyên tố; a∈N; a ≠ 0. Khi đó

$$\gcd(a,p) = p \Leftrightarrow (a \vdots p)$$

$$\gcd(a,p) = 1 \Leftrightarrow (a \not\vdots p)$$

3). Nếu tích của nhiều số chia hết cho một số nguyên tố p, thì có ít nhất một thừa số chia hết cho p.

$$\prod_{i=1}^{N} a_i \vdots p \Rightarrow \exists a_i \vdots p$$

4). Ước số dương bé nhất khác 1 của hợp số a là số nguyên tố không vượt quá √a.

5). Số 2 là số nguyên tố nhỏ nhất và cũng là số nguyên tố chẵn duy nhất.

6). Tập hợp các số nguyên tố là vô hạn (không có số nguyên tố lớn nhất).

Chứng minh: Giả sử có pr là số nguyên tố lớn nhất và các số nguyên tố được ký hiệu là: p1 = 2, p2 = 3, p3 = 5, … , pr

Đặt: P= p1. p2. p3…pr

Ta có: ∀ pi , i=1,2,3,…,r

(P + 1) mod pi = (p1. p2. p3…pr + 1) mod pi = 1

Vậy P + 1 là số nguyên tố. Nhưng P+1 > pr mâu thuẫn với giả thiết pr là số nguyên tố lớn nhất. Vậy tập hợp các số nguyên tố là vô hạn.

2.1.3. Định lý cơ bản của số học

Định lý 1.1

Mọi số tự nhiên lớn hơn 1 đều phân tích được thành tích những thừa số nguyên tố, và sự phân tích này là duy nhất nếu không kể đến thứ tự của các thừa số.

Từ đó có dạng phân tích tiêu chuẩn của một số tự nhiên bất kỳ là:

$$n = p_1^{k_1} p_2^{k_2} \dots p_m^{k_m}$$

Trong đó $p_1, p_2, \dots, p_m$ là các số nguyên tố đôi một khác nhau. Ta có n chia hết cho $(k_1+1)(k_2+1)\dots(k_m+1)$ số tự nhiên.

Ví dụ: 300 chia hết cho (2+1)(2+1)(1+1) =18 số tự nhiên. $300 = 2^2 .5^2 .3$

Tuy nhiên, vì tập hợp số nguyên tố là tập con của số tự nhiên, mà tập hợp số tự nhiên là đếm được nên tập hợp các số nguyên tố là đếm được. Lưu ý khái niệm đếm được trong toán học khác với ngôn ngữ đời thường, một tập hợp có vô hạn phần tử vẫn có khả năng đếm được.

Định lý 1.2

Mọi hợp số n đều có ước nguyên tố nhỏ hơn hoặc bằng n . Thật vậy mọi hợp số n ta có thể phân tích thành tích tiêu chuẩn sau:

Mọi hợp số n đều có ước nguyên tố nhỏ hơn hoặc bằng n .

Thật vậy mọi hợp số n ta có thể phân tích thành tích tiêu chuẩn sau:

$$n = p_1^{k_1} p_2^{k_2} \dots p_m^{k_m}$$

Trong đó $p_1, p_2, \dots, p_m$ là các số nguyên tố đôi một khác nhau.

Mặt khác hợp số chẵn bé nhất 4 có ước nguyên tố là $2 \leq \sqrt{4}$ và hợp số lẻ bé nhất 9 có ước nguyên tố là $3 \leq \sqrt{9}$. Do vậy mọi hợp số n đều có ước nguyên tố nhỏ hơn hoặc bằng $\sqrt{n}$.

2.1.4. Sự phân bố số nguyên tố

Định nghĩa 1.1

Hàm π(n) được định nghĩa là số các số nguyên tố nhỏ hơn hay bằng n.

Ví dụ π(5) = 3 bởi vì 2, 3 và 5 là số nguyên tố.

Bảng liệt kê dưới đây liệt kể một vài giá trị của π.

n	2	3	4	5	6	7	8	9	10	100	1000	10000
π(n)	1	2	2	3	3	4	4	4	4	25	168	1229

Định lý 2.3. (Định lý số học)

$$\pi(n) \sim \frac{n}{\ln n}$$

Phát biểu của định lý: π(n) có giá trị xấp xỉ bằng n n ln . Có nghĩa là khi n→∞, π(n) có giá trị gần bằn $\frac{n}{\ln n}$. Một cách hình thức, điều đó có nghĩa tỉ số của hai hàm xấp xỉ bằng 1 khi n đủ lớn hoặc $\lim_{x\to\infty} \frac{\pi(n)}{n/\ln n} = 1$

Bảng dưới đây so sánh giá trị gần đúng và giá trị chính xác của π(n) mà ta biết được.

n	2	3	4	5	6	7	8	9	10	100	1000	10000
π(n)	1	2	2	3	3	4	4	4	4	25	168	1229
n/ln(n)	2.9	2.7	2.9	3.1	3.3	3.6	3.8	4.1	4.3	22	145	1086

Có thể sử dụng định lý số học (Prime Number Theorem) để ước lượng số các số nguyên tố trong một khoảng bằng cách trừ hai giá trị của π. Ví dụ để biết trong khoảng 5 tỉ đến 6 tỉ có bao nhiêu số nguyên tố:

$\pi(6.10^9) - \pi(5.10^9) \approx 2{,}66.10^8 - 2{,}24.10^8 = 4{,}26.10^8$

Vậy có:

- $4{,}26.10^8$ số nguyên tố trong khoảng 5 tỉ đến 6 tỉ.

- Khoảng 4,3% các số trong khoảng này là số nguyên tố, như vậy trung bình cứ trong 23 số thì có một số là nguyên tố.

Điều này rất có ích cho việc xây dựng các ứng dụng tìm số nguyên tố. Ví dụ, một phần mềm mã hóa cần sinh ngẫu nhiên một số nguyên tố trong khoảng 5 tỉ đến 6 tỉ thì trung bình nó cần kiểm tra 23 số ngẫu nhiên trước khi tìm ra một số. Để tìm ra một số nguyên tố trong khoảng n cho trước cần kiểm tra khoảng ln n số nguyên chọn ngẫu nhiên. Con số này giảm đi một nửa nếu chỉ chọn ngẫu nhiên các số lẻ trong khoảng n.

2.2 SỐ NGUYÊN TỐ CÓ DẠNG ĐẶT BIỆT

2.2.1. Số nguyên tố dạng Mersenne

2.2.1.1 Khái niệm số nguyên tố Mersenne

Số nguyên tố Mersenne là một số Mersenne có dạng:

$$M_n=2^n-1$$

Một số định nghĩa yêu cầu lũy thừa (n) phải là số nguyên tố và là một số nguyên tố. Ví dụ 31 là số nguyên tố Mersenne vì $31 = 2^5 − 1$ và 31 là số nguyên tố.

Điều kiện cần để M_n là số nguyên tố là q phải là số nguyên tố. Tuy nhiên thực tế cho thấy: mặc dù q là số nguyên tố nhưng M_n có thể là một hợp số. Chẳng hạn: $2^{11}-1 = 2047$ là một hợp số, vì $2047 = 23.89$.

Bảng dưới đây liệt kê một vài Mersenne và cho biết số nào là số nguyên tố.

n	2	3	4	5	6	7	8	9	10	11
M_n	3	7	15	31	63	127	255	511	1023	2047
Prime?	Y	Y	N	Y	N	Y	N	N	N	N

Định lý 1.4

Nếu $M_n = 2^n -1$ là số nguyên tố thì n là số nguyên tố.

Chứng minh. Thật vậy, nếu n là một hợp số, chẳng hạn n = p.r trong đó p và r là những số nguyên dương nào đó, p,r > 1. Thế thì theo Hệ quả (Corollary) trong [3], ta có; $2^n – 1 = 2^{pr} – 1 = (2^p-1)(2^{p(r-1)} + 2^{p(r-2)}+…+ 2^p + 1)$. Vậy $2^n – 1$ phải là hợp số. Định lý 1.4 được chứng minh.

Định lý 2.5

Mọi số nguyên tố n > 2, mọi phân tích nguyên tố của M_n tương đương với 1 mod n và ± 1 mod 8.

Hiện nay, các số nguyên tố lớn nhất được tìm thấy thường là số nguyên tố Mersenne. Các số nguyên tố Mersenne có quan hệ chặt chẽ với các số hoàn thiện (là số nguyên dương có tổng các ước số nguyên dương bé hơn nó bằng chính nó), nghĩa là các số bằng tổng các ước chân chính của nó. Trong lịch sử, việc nghiên cứu các số nguyên tố Mersenne đã từng bị thay đổi do các liên quan này. Vào thế kỷ 4 42 TCN, Euclid phát biểu rằng nếu M là số nguyên tố Mersenne thì M(M+1)/2 là số hoàn thiên.

Vào thế kỷ 18, Leonhard Euler chứng minh rằng tất cả các số hoàn thiện chẵn đều có dạng này. Không một số hoàn thiện lẻ nào được biết, và người ta nghi ngờ rằng chúng không tồn tại.

2.2.1.2 Tìm các số nguyên tố Mersenne

Phần này sẽ được áp dụng trong chương trình code tính số nguyên tố Mersenne 2^{n-1}.

Đẳng thức: $2^{ab} - 1 = (2^a - 1).(1 + 2^a + 2^{2a} + 2^{3a} + \ldots 2^{(b-1)a})$

Cho biết rằng M_n có thể là số nguyên tố chỉ nếu chính n=ab là số nguyên tố, điều đó làm giản lược bớt việc tìm các số nguyên tố Mersenne. Mệnh đề đảo, nói rằng M_n là số nguyên tố nếu n là số nguyên tố là chưa hoàn toàn đúng.

Ví dụ $2^{11} - 1 = 23 \times 89$ là hợp số.

Đã có các thuật toán nhanh để tìm số nguyên tố Mersenne, do đó hiện nay đã biết các số nguyên tố Mersenne rất lớn. Chẳng hạn, ngày nay, người ta đã xác định được rằng : các số nguyên dạng $2^n - 1$ là những số nguyên tố nếu p∈{ 2,3,5,7,13,17,19,31,61,89,107, 521, 607,1279,2203,2281,3217,4253, 4423, 9689, 9941,11213,19937, 21701, 23209,}. Theo Wikipedia

Bốn số nguyên tố Mersenne đầu tiên $M_2 = 3$, $M_3 = 7$, $M_5 = 31$ và $M_7 = 127$ đã được biết từ cổ xưa. Số thứ năm, $M_{13} = 8191$, được tìm thấy vào trước năm 1461; hai số tiếp theo (M_{17} và M_{19}) tìm thấy bởi Cataldi vào năm 1588. Sau hơn một thế kỷ M_{31} được kiểm tra bởi Euler vào năm 1750. Số tiếp theo (trong lịch sử, không theo thứ tự số) là M_{127}, do Lucas tìm thấy vào năm 1876, sau đó M_{61} do Pervushin

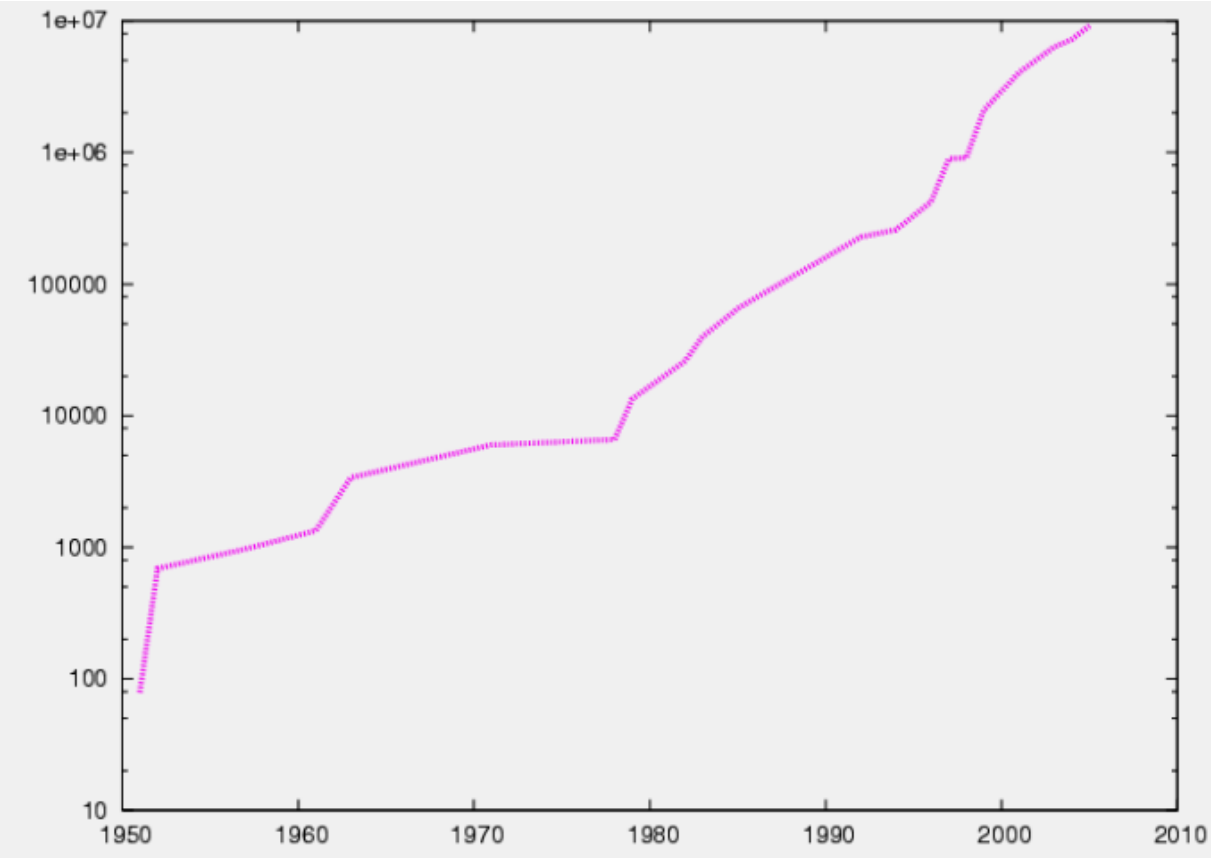
tìm vào năm 1883. Hai số nữa (M_{89} và M_{107}) được tìm thấy vào thế kỷ 20, bởi Powers vào năm 1911 và 1914.

Từ thế kỷ 17, các số này được mang tên nhà toán học Pháp Marin Mersenne, người đã chứng minh một loạt các số nguyên tố Mersenne với số mũ lên tới 257. Danh sách của ông đã mắc một số sai lầm, như bao gồm cả M_{67}, M_{257}, và bỏ quên M_{61}, M_{89} và M_{107}.

Phương pháp tốt nhất để kiểm tra tính nguyên tố của các số Mersenne được dựa vào sự tính toán một dãy tuần hoàn, được phát biểu đầu tiên bởi Lucas năm 1878 và chứng minh bởi Lehmer vào những năm 1930. Hiện nay nó được gọi là kiểm tra Lucas - Lehmer với số nguyên tố Mersenne. Đặc biệt, ta có thể chứng minh rằng (với $n > 2$) $M_n = 2^n - 1$ là số nguyên tố nếu và chỉ nếu M_n chia hết cho S_{n-2}, trong đó $S_0 = 4$ và với $k > 0$, 2 $S_k = S^2{}_{k-1} - 2$. trong đó $S_{n-2} \equiv 0 \bmod(n^{n-1})$.

Việc tìm các số nguyên tố Mersenne thực sự được cách mạng bởi các máy tính điện tử số. Thành công đầu tiên của tư tưởng này thuộc về số nguyên tố Mersenne M_{521}, nhờ nỗ lực khéo léo vào lúc 10:00 P.M ngày 30/01/1952 khi sử dụng máy tính tự động Western U.S. National Bureau of Standards (SWAC) tại Institute for Numerical Analysis thuộc Đại học California tại Los Angeles, dưới sự điều khiển trực tiếp của Lehmer, sử dụng chương trình viết và chạy bởi GS R.M. Robinson. M_{521} là số nguyên tố Mersenne đầu tiên tìm thấy sau 38 năm; số tiếp theo, M_{607}, đã được tìm thấy do computer này sau gần hai giờ chạy máy. Ba số tiếp theo M_{1279}, M_{2203}, M_{2281} đã được tìm thấy với cùng chương trình trên sau nhiều tháng nữa. M_{4253} là số nguyên tố Mersenne đầu tiên là số nguyên tố siêu lớn (trên 1000 chữ số thập phân-titanic). M_{44497} là số nguyên tố đầu tiên có trên 10.000 chữ số thập phân (gigantic).

Đến tháng 9 năm 2008, chỉ mới biết 46 số nguyên tố Mersenne; số lớn nhất đã biết là số $(2^{43112609} - 1)$. Cũng như nhiều số nguyên tố Mersenne trước đó, nó được tìm ra nhờ dự án tính toán phân tán trên Internet, được biết với tên gọi Tìm kiếm số nguyên tố Mersenne khổng lồ trên Internet (Great Internet Mersenne Prime Search - GIMPS)



Hình 2.2.1.2 Đồ thị biểu diễn số các chữ số của số nguyên tố Mersenne lớn nhất đã biết theo từng năm của kỷ nguyên điện tử

Danh sách các số nguyên tố Mersenne đã biết:

TT	n	Mn	Số chữ số trong M_n	Ngày tìm được	Người tìm
1	2	3	1	cổ đại	Hy Lạp cổ đại
2	3	7	1	cổ đại	Hy Lạp cổ đại
3	5	31	2	cổ đại	Hy Lạp cổ đại
4	7	127	3	cổ đại	Hy Lạp cổ đại
5	13	8191	4	1456	Khuyết danh
6	17	131071	6	1588	Cataldi
7	19	524287	6	1588	Cataldi
8	31	2147483647	10	1750	Euler

9	61	2305843009213693951	19	1883	Pervushin
10	89	618970019…449562111	27	1911	Powers
11	107	162259276…010288127	33	1914	Powers
12	127	170141183…884105727	39	1876	Lucas
13	521	686479766…115057151	157	30/1/1952	Robinson
14	607	531137992…031728127	183	30/1/1952	Robinson
15	1 279	104079321…168729087	386	25/6/1952	Robinson
16	2 203	147597991…697771007	664	7/10/1952	Robinson
17	2 281	446087557…132836351	687	9/10/1952	Robinson
18	3 217	259117086…909315071	969	8/9/1957	Riesel
19	4 253	190797007…350484991	1281	3/11/1961	Hurwitz
20	4 423	285542542…608580607	1332	3/11/1961	Hurwitz
21	9 689	478220278…225754111	2917	11/5/1963	Gillies
22	9 941	346088282…789463551	2993	16/5/1963	Gillies
23	11 213	281411201…696392191	3376	2/6/1963	Gillies
24	19 937	431542479…968041471	6002	4/3/1971	Tuckerman
25	21 701	448679166…511882751	6533	30/10/1978	Noll & Nickel
26	23 209	402874115…779264511	6987	9/2/1979	Noll
27	44 497	854509824…011228671	13395	8/4/1979	Nelson & Slowinski
28	86 243	536927995…433438207	25 962	25/9/1982	Slowinski
29	110 503	521928313…465515007	33 265	28/1/1988	Colquitt & Welsh
30	132 049	512740276…730061311	39 751	20/9/1983	Slowinski
31	216 091	746093103…815528447	65 050	6/9/1985	Slowinski

32	756 839	174135906…544677887	227 832	19/2/1992	Slowinski & Gage trên Cray-2
33	859 433	129498125…500142591	258 716	10/1/1994	Slowinski & Gage
34	1 257 787	412245773…089366527	378 632	3/9/1996	Slowinski & Gage
35	1 398 269	814717564…451315711	420 921	13/11/1996	GIMPS / Joel Armengaud
36	2 976 221	623340076…729201151	895 932	24/8/1997	GIMPS / Gordon Spence
37	3 021 377	127411683…024694271	909 526	27/1/1998	GIMPS / Roland Clarkson
38	6 972 593	437075744…924193791	2 098 960	1/6/1999	GIMPS / Nayan Hajratwala
39	13 466 917	924947738…256259071	2 098 960	14/11/2001	GIMPS / Michael Cameron
40*	20 996 011	125976895…855682047	6 320 430	17/11/2003	GIMPS / Michael Shafer
41*	24 036 583	299410429…733969407	7 235 733	15/5/2004	GIMPS / Josh Findley
42*	25 964 951	122164630…577077247	7 816 230	18/2/2005	GIMPS / Martin Nowak
43*	30 402 457	315416475…652943871	9 152 052	15/12/2005	GIMPS / Curtis Cooper& Steven Boone

44*	32 582 657	124575026…053967871	9 808 358	4/9/2006	GIMPS / Curtis Cooper & Steven Boone
45*	37 156 667	202254406…308220927	11 185 272	6/9/2008	GIMPS / Hans-Michael Elvenich
46*	43 112 609	316470269…697152511	12 978 189	23/8/2008	GIMPS / Edson Smith

Chưa khẳng định được có số nguyên tố Mersenne nào nằm giữa số thứ 39 ($M_{13\,466\,917}$) và 46 ($M_{43\,112\,609}$) trong bảng mà chưa được phát hiện hay không, do đó thứ tự các số đó là tạm thời. Một ví dụ là số thứ 29 được phát hiện ra sau số thứ 30 và 31, số thứ 46 cũng được công bố trước số 45 2 tuần.

Để hình dung độ lớn của số nguyên tố lớn nhất được tìm thấy (số thứ 46), cần có 3461 trang giấy để biểu diễn số đó với các chữ số trong hệ cơ số 10, 75 chữ số một dòng và 50 dòng một trang.

Ưu điểm:

Đưa ra được các số rất lớn: $2^{19937}-1$

Pass rất nhiều các bài kiểm tra về tính ngẫu nhiên, có thể nói Mersenne là một thuật toán vô cùng tốt.

Nhược điểm: Mersenne có nhược điểm cơ bản về hiệu suất, có thể coi là chậm và tốn bộ nhớ

Số nguyên tố Mersenne sẽ được áp dụng vào chương 3.

2.2.2. Số nguyên tố Lucas-Lehmer

Khái niệm:

Trong số học cho máy tính (hay số học thuật toán), kiểm tra lucas-lehmer là phép kiểm tra tính số nguyên tố đối với số tự nhiên n, nó đòi hỏi rằng có một thừa số nguyên tố của n – 1 là đã biết.

Nếu tồn tại số a nhỏ hơn n và lớn hơn 1 là số thoả mãn

$$a^{n-1} \equiv 1 \pmod{n} \text{ và } a^{n-1/q} \not\equiv 1 \text{ (mode n)}$$

Với mọi ước nguyên tố q của n − 1, thì n là số nguyên tố. Nếu không tìm thấy số a như vậy thì n là hợp số.

Chẳng hạn, với n = 71, n − 1 = 70 = (2)*(5)*(7).Lấy a = 11 trước hết:

$$11^{70} \equiv 1 \text{ (mode 71)}$$

Điều này cho thấy bậc của 11 mod 71 là 70 vì ước của 70 chỉ có thể như trên. Nhưng kiểm tra với các ước của 70 ta có:

$$1135 \equiv 70 \not\equiv 1 \text{ (mode 71)}$$

$$1114 \equiv 54 \not\equiv 1 \text{ (mode 71)}$$

$$1110 \equiv 32 \not\equiv 1 \text{ (mode 71)}$$

Do đó bậc của 11 mod 71 là 70, và như vậy 71 là số nguyên tố.

Ứng dụng :

Dãy Lucas được sử dụng trong kiểm tra xác suất giả nguyên tố Lucas nằm trong Kiểm tra tính nguyên tố Baillie-PSW thường dùng.

Dãy Lucas được dùng trong một số phương pháp chứng minh tính nguyên tố, bao gồm Kiểm tra Lucas-Lehmer-Riesel và các phương pháp khác trong Brillhart-Lehmer-Selfridge 1975[4]

LUC là hệ thống mật mã khóa công khai dựa trên dãy Lucas[5] thực hiện hệ analog ElGamal (LUCELG), Diffie – Hellman (LUCDIF) và RSA (LUCRSA). Việc mã hóa thông điệp trong LUC được tính như một phần tử của dãy Lucas nhất định, thay vì lũy thừa mô-đun như trong RSA hoặc Diffie – Hellman. Tuy nhiên, bài viết của Bleichenbacher và cộng sự[6] cho thấy nhiều lợi thế bảo mật của LUC là không chính xác hoặc không đáng kể khi so sánh với các hệ thống dựa trên lũy thừa mô đun. Kiểm tra lucas-lehmer sẽ được áp dụng vào chương 3

Theo Wikipedia

2.2.3. Số nguyên tố Fermat

Số Fermat là một khái niệm trong toán học, mang tên nhà toán học Pháp Pierre de Fermat, người đầu tiên đưa ra khái niệm này. Nó là một số nguyên dương có dạng $F_n = F_n = 2^{2^n} + 1$

Rất nhiều số Fermat là số nguyên tố nên một thời người ta cho rằng tất cả các số có dạng đó đều là số nguyên tố. Với n là số không âm. Các số Fermat đầu tiên bao gồm:

$F_0 = 2^1 + 1 = 3$ $F_1 = 2^2 + 1 = 5$

$F_2 = 2^4 + 1 = 17$ $F_3 = 2^8 + 1 = 257$

$F_4 = 2^{16} + 1 = 65.537$ $F_5 = 2^{32} + 1 = 4.294.967.297$

$F_6 = 2^{64} + 1 = 18.446.744.073.709.551.617$

$F_7 = 2^{128} + 1 = 340.282.366.920.938.463.463.374.607.431.768.211.457$

$F_8 = 2^{256} + 1 =$ 115.792.089.237.316.195.423.570.985.008.687.907.853.269.984.665.640.564.039.457.584.007.913.129.639.937

Công thức thiết lập số Fermat

$F_n = (F_{n-1} - 1)^2 + 1$ $F_n = F_{n-1} + 2^{2^{n-1}} F0 \dots F_{n-2}$

$F_n = F^2_{n=1} - 2(F_{n-2} = 1)^2$

$F_n = F_0 \dots F_{n-1} + 2$

Với $n \geq 2$, các hệ thức trên có thể chứng minh bằng cách quy nạp toán học. Ta có thể tính gần đúng số chữ số của chúng bằng hệ thức gần đúng:

$D(n,b) = \lfloor \log_b (2^{2^n} + 1) + 1 \rfloor \approx \lfloor 2^n \log_b 2 + 1 \rfloor$

Fermat đưa ra dự đoán vào năm 1637 rằng các số dạng này là số nguyên tố chỉ ra sự đúng đắn của n từ 0 đến 4. Tuy nhiên, tất cả các số Fermat được kiểm sau đó đều cho kết quả là hợp số.

Định lý 2.6 Nếu $p = 2^m + 1$ là số nguyên tố lẻ thì m là lũy thừa của 2.

Định lý này nói rằng bất kỳ số nguyên tố nào lớn hơn là lũy thừa của 2 cộng với 1 thì là số nguyên tố Fermat.

Định lý 2.7 Cho $n \geq 2$, mọi thừa số nguyên tố p của F_n có dạng $p \equiv 1 \bmod 2^{n+2}$.

Định lý này nói rằng cả định lý Euler phân tích ra tích các thừa số nguyên tố đều được kiểm tra khi kiểm thử F_n . Các phương pháp khác nhau được sử dụng để biểu diễn F_n là hợp số cho n từ 5 đến 32 là phương pháp Trial Division và kiểm thử Pepin.

2.3. MỘT SỐ PHƯƠNG PHÁP KIỂM TRA SỐ NGUYÊN TỐ

2.3.1. Phương pháp cổ điển

2.3.1.1. Phương pháp Trial Division

Nếu n không có bất kỳ ước số a nào nằm trong khoảng 1< a <√n , thì n là số nguyên tố.

Sau đây là thuật toán sơ khai kiểm tra nguyên tố với số tự nhiên n:

```
Function Prime(n : Integer) : Boolean; Var
i : Integer;
Begin Prime := False;
For i := 2 to n-1 do
if n mod i = 0 then Break;
Prime := True;
End;
```

Như ta thấy thuật toán ở trên khá đơn giản, tuy nhiên cũng dễ nhận ra là thuật toán này đi theo hướng vét cạn do đó hoàn toàn không tối ưu. Chi phí trong trường hợp xấu nhất lên tới O(n).

Phương pháp Trial Division dựa vào nhận xét: Nếu n là hợp số thì nó phải có một ước nhỏ hơn √n , và do đó ta chỉ cần xét các ước không quá √n của nó. Thật may mắn khi chúng ta có thể chứng minh rằng thay vì chia thử cho toàn bộ các số từ 2 đến (n-1) chúng ta chỉ cần kiểm tra các ước số ≤ √n . Thuật toán cải tiến sẽ được mô tả như sau :

```
Function Prime(n : Integer) : Boolean;
    Var
    I,m : Integer;
    Begin
    Prime := False; m:= Trunc(Sqrt(n));
    For i := 2 to m do
    if n mod i = 0 then Break;
    Prime := True;
    End;
```

Thuật toán này thực hiện tối đa n phép toán để kiểm tra $\sqrt{n}$ là số nguyên tố. Nếu $\sqrt{n}=2^t$ thì thời gian thực hiện của thuật toán là $O(\sqrt{n}^1) = O(2^{t/2})$ là hàm mũ của t. Tuy nhiên ta cũng cần chú ý rằng thuật toán cải tiến chỉ khác thuật toán ban đầu khi n là số nguyên tố, còn với n là hợp số thì hai thuật toán kết thúc sau cùng một số phép tính.

2.3.1.2. Phương pháp sàng Eratosthenes

Thuật toán này sẽ áp dụng vào chương trình code phần: Liệt kê các số nguyên tố trong một phạm vi.

Sàng Eratosthenes: Là một thuật giải toán cổ xưa để tìm các số nguyên tố nhỏ hơn 100. Thuật toán này do nhà toán học cổ Hy Lạp là Eratosthenes (Ơ-ra-tôxten) phát minh ra.

Ban đầu, nhà toán học Eratosthenes sau khi tìm ra thuật toán, đã lấy lá cọ và ghi tất cả các số từ 2 cho đến 100. Ông đã chọc thủng các hợp số và giữ nguyên các số nguyên tố. Bảng số nguyên tố còn lại trông rất giống một cái sàng. Do đó, nó có tên là sàng Eratosthenes.

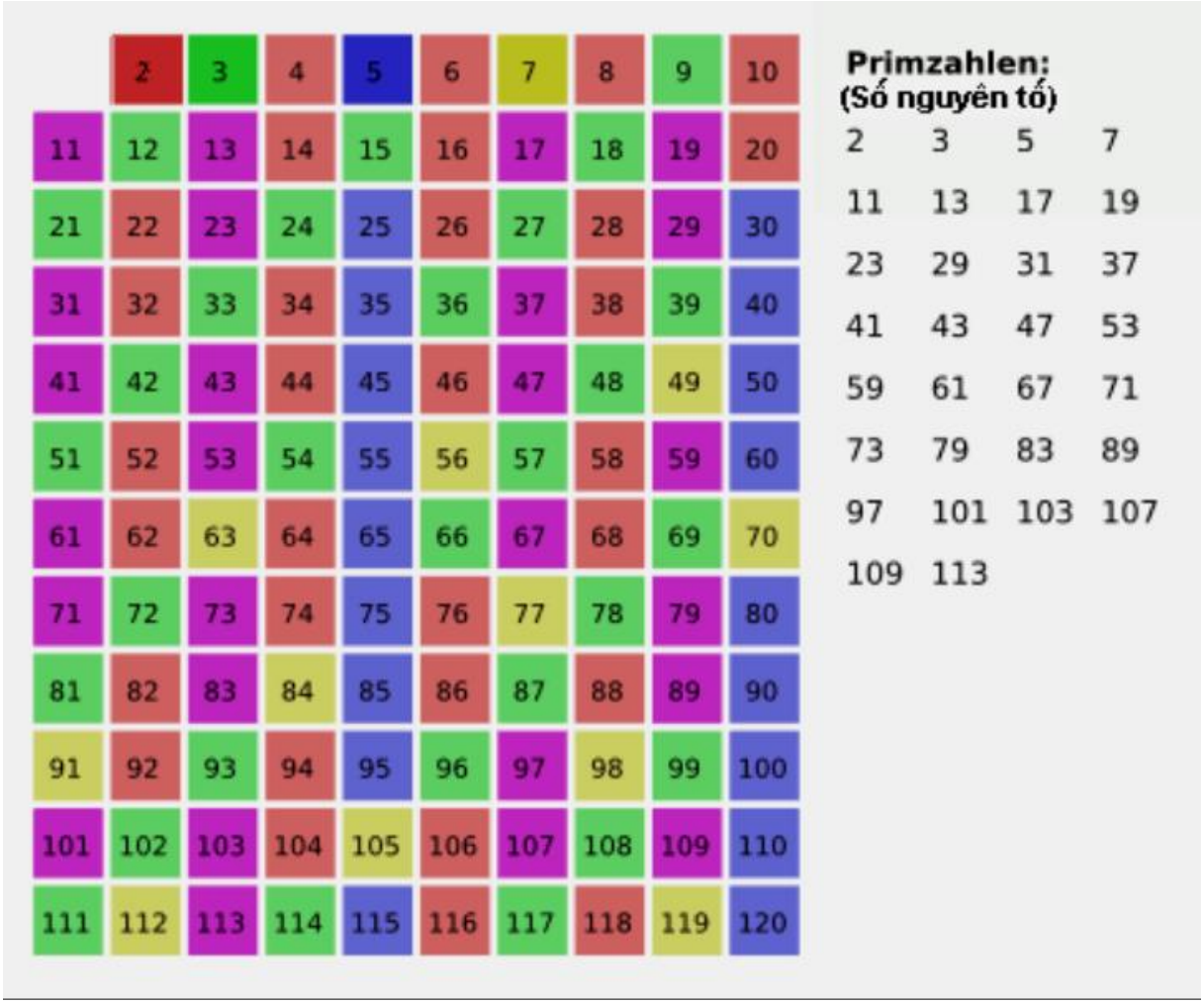
Chú ý: Sàng Eratosthenes chỉ ghi các số từ 2 đến 100 mà không ghi hai số 0 và 1, cả hợp số lẫn số nguyên tố đều lớn hơn 0 và 1.

Ta có thể diễn đạt giải thuật sàng Eratosthene như sau:

```
Procedure Eratosthene(var Prime[1..n]: boolean);
Var i,j,m:integer;
Begin
for i:=1 to n do Prime[i]:=True;
Prime[1]:=false; m:= Trunc(Sqrt(n));
For i:=2 to m do
if (Prime[i])then Prime[Boi cua i]:= False;
End;
```

Đầu tiên xóa số 1 ra khỏi tập các số nguyên tố. Tiếp theo số 1 là số 2, là số nguyên tố giữ lại. Tiếp thep xoá tất cả các bội của 2 ra khỏi bảng. Số đầu tiên không bị xoá là số 3 (số nguyên tố). Tiếp theo xoá các bội của 3...

Giải thuật tiếp tục cho đến khi gặp số nguyên tố lớn hơn hoặc bằng sqrt(n) thì dừng lại. Tất cả các số không bị xoá là số nguyên tố.



Với mỗi số nguyên tố p nhỏ hơn √n , vòng lặp bên trong sẽ thực hiện n/p lần.

Do đó thời gian thực hiện thuật toán tương đương với $\sum p \leq n\frac{n}{p}$, mà $\sum p \leq n\frac{n}{p} =$ O(n log log n)

vì vậy độ phức tạp là O(n log log n). Tuy nhiên nó phải tìm tất các số nguyên tố nhỏ hơn n để kiểm tra n là số nguyên tố, để kiểm tra số nguyên tố rất lớn thì điều này là không khả thi.

2.3.2. Phương pháp xác suất

2.3.2.1. Thuật toán Fermat

1). Cơ sở lý thuyết của giải thuật: $a^{p-1} = 1(\text{mod } p)$

Định lý 2.8. (Định lý nhỏ Fermat)

Nếu p là số nguyên tố và $1 \leq a \leq p$ thì

Hệ quả

Nếu p là số nguyên tố, a là số nguyên, thì $a^p \equiv a \text{ (mod p)}$

Ví dụ: a = 2 thì

Với p= 3: $a^{3-1} = 4 = 1 \text{ mod } 3$ Với p = 4: $a^{4-1} = 8 = 0 \text{ mod } 4$

Với p= 5: $a^{5-1} = 16 = 1 \text{ mod } 5$ Với p = 6: $a^{6-1} = 32 = 2 \text{ mod } 6$

Với p= 7: $a^{7-1} = 64 = 1 \text{ mod } 7$ Với p = 8: $a^{8-1} = 128 = 0 \text{ mod}$

Với p= 9: $a^{9-1} = 256 = 1 \text{ mod } 9$

Kết quả trên xác nhận khi p là số nguyên tố thì $2^{p-1} \equiv 1 \text{ (mod p)}$, và cũng cho một phỏng đoán rằng p không phải là số nguyên tố khi $2^{p-1} \neq 1(\text{mod p})$. Tuy nhiên phỏng đoán này lại không đúng, chẳng hạn như n = 341 không phải là số nguyên tố nhưng nó vẫn thỏa $2^{341} - 1 \equiv 1 \text{ (mod 341)}$.

Các số mà thỏa mãn định lý Fermat mà không phải là số nguyên tố gọi là số giả nguyên tố và được định nghĩa như sau:

Định nghĩa 1.2

Một số giả nguyên tố cơ sở a là một hợp số nguyên n thỏa mãn $a^{n-1} \equiv 1(\text{mod n})$.

2). Giải thuật:

Nếu số giả nguyên tố cơ sở a không tồn tại thì định lý Fermat cho một cách rất đơn giản để kiểm tra số nguyên tố. Đáng tiếc số giả nguyên tố cơ sở a lại tồn tại với mọi cơ sở vì vậy định lý Fermat chỉ cho một cách kiểm tra thiên về hợp số và thuật toán như sau:

```
Boolean pseudoprime (n, b)
    {
        If (b^(n-1) % n == 1)return        true;
        Else return false;
    }
```

Thuật toán Fermat được xây dựng trên cơ sở thuật toán kiểm tra thiên về hợp số ở trên. Nó sẽ kiểm tra một số n là giả nguyên tố với số lần kiểm tra là k, kết luận n là số nguyên tố với xác suất nào đó nếu nó vượt qua k lần kiểm tra. Thuật toán như sau:

Thuật toán pseudoprime có độ phức tạp là O(log n). Thuật toán kiểm tra Fermat thực hiện pseudoprime k lần vì vậy độ phức tạp sẽ là O(k.log n). Do vậy nó là thuật toán xác suất kiểm tra số nguyên tố hiệu quả.

```
Boolean fermat (n, k)
    {
        For (i from 1 to k)
        {       b = random(2, n-1); //inclusive
                If  (!pseudoprime(n,b))  return false;
        }
        return true;  //probably prime
    }
```

2.3.2.2. Thuật toán Miller - Rabin

1). Cơ sở lý thuyết của giải thuật:

Cho p là một số nguyên tố, a là số nguyên không chia hết cho p, ta có :

$a^{p-1} \equiv 1 \pmod p$ (1)

$a^2 \equiv 1 \pmod p \Leftrightarrow a \equiv 1 \pmod p$ hoặc $a \equiv -1 \pmod p$. (2).

Hai tính chất trên khá quen thuộc trong số nguyên tố, ta có thể dễ dàng chứng minh.

Với p-1 = q.2^t trong đó q là một số nguyên dương lẻ. Theo (1) và (2) mọi số a không chia hết cho p ta đều có :

$a^q \equiv 1 \pmod p$ hoặc

$a^{q.2^t} \equiv -1 \pmod p$ với một số nguyên r nào đó thoả $0 \le r < t$. (*)

Như vậy, một số n là nguyên tố thì nó phải thoả mãn điều kiện trên với mọi $1 \le a \le n-1$, ngược lại nếu tồn tại một a không thoả mãn tính chất trên thì chắc chắn n không phải là số nguyên tố.

Mặt khác, ta có thể chứng minh được rằng, nếu n là một hợp số lẻ > 3 thì có không quá $\frac{n-1}{4}$ giá trị a trong khoảng 1 đến n-1 thoả mãn (*). Do đó, với một số a chọn ngẫu nhiên trong khoảng 1 đến n-1, thì xác suất a thoả mãn (*) không quá $\frac{1}{4}$, và nếu k lần chọn thì xác suất để mọi lần chọn đều thoả mãn là không quá k $\frac{1}{4}$.

Từ đó, người ta đã đưa ra một thuật toán xác suất cho phép kiểm tra một số n có phải nguyên tố hay không trong thời gian đa thức với độ chính xác rất cao thuật toán (Miller - Rabin).

Thuật toán Miller – Rabin thực hiện về bản chất nó tính $a^r \bmod n$ bằng phương pháp bình phương liên tiếp. Thuật toán Miller – Rabin thực hiện k lần tính $a^r \bmod n$ với xác suất phân loại sai là $\le \frac{1}{4^k}$.

Phần quan trọng nhất là kiểm tra a có thoả mãn tính chất (*) hay không. Trước hết, ta phải tính $a^q \bmod n$, nếu ta tính bằng thực hiện tuần tự các phép nhân q lần thì chi phí sẽ rất lớn (q có thể tỉ lệ với n). Vì ta chỉ cần tính luỹ thừa modulo n nên ta có một phương pháp rất hay : Phương pháp bình phương liên tiếp.

Ý tưởng cơ bản của nó như sau : phân tích q dạng nhị phân, q sẽ có dạng tổng của các luỹ thừa của 2.

Ta xây dựng dãy {u} như sau :

- $u_0 = a$
- $u_{k+1} = u_k^{\,2} \bmod n \ \forall\ k \in N$.

Khi đó, ta dễ dàng chứng minh $u_k = a^{2^k} \bmod n$.

Khi đó, $a_q \bmod n$ sẽ được tính bằng tích theo modulo n của các u_k tương ứng với các luỹ thừa của 2 trong phân tích nhị phân của q.

2). Giải thuật:

Thuật toán được trình bày như sau:

Algorithm for the Miller-Rabin Probabilistic Primality Test

Miller-Rabin(n,t)

INPUT: An odd integer n>1 and a positive

security parameter t

OUTPUT: the answer “COMPOSITE” or “PRIME”

Write n-1 = $2^s r$ such that r is odd

Repeat from 1 to t

Choose a random integer a which satisfies 2<a<n=1

Compute $y=a^r \bmod n$

If y<>1 and y<>n-1 then DO

    j:=1

    while j<s and y<>n-1 then DO

        y:= $y^2 \bmod n$

        if y=1 then return(“COMPOSITE”)

        j:=j+1

    if y<>n-1 then return(“COMPOSITE”)

return(“PRIME”)

2.3.2.3. Thuật toán AKS

1). Cơ sở lý thuyết của giải thuật:

Tháng 8 năm 2002, ba tác giả Manindra Agrawal, Neeraj Kayal va Nitin Xaxena (Viện công nghệ Kanpur Ấn Độ) công bố thuật toán kiểm tra tính nguyên tố với độ phức tạp đa thức (thường gọi là thuật toán AKS). Nó thoả mãn ba tính chất:

- Xác định: Luôn cho câu trả lời chính xác.

- Không điều kiện: Không bác bỏ bất cứ giả thiết không được chứng minh nào chẳng hạn như giả thiết Riemann.

- Thời gian đa thức: Độ phức tạp thời gian của thuật toán là đa thức.

Thuật toán xuất phát từ ý tưởng sau:

Số nguyên tố p là nguyên tố khi và chỉ khi đẳng thức sau đúng với một số nguyên a nào đó mà nguyên tố cùng nhau với p:

$(x-a)^p \equiv x^p - a \pmod{p}$ (*)

Như việc kiểm tra đẳng thức trên không phải đơn giản (khi p đủ lớn), cho rút gọn hai vế của đẳng thức trên theo modulo đa thức xr -1 (với r là một số nguyên có tính chất "đặc biệt"), sau đó lại rút gọn các hệ số của kết quả thu được theo modulo p. Tức là hệ thức sau:

$(x-a)^p \equiv x^p - a \pmod{x^r - 1, p}$ (**)

Biểu thức (**) có thể xảy ra trong một số trường hợp p là hợp số, nhưng người ta đã chứng minh được rằng không hợp số p nào thoả mãn (**) với mọi a nằm trong vùng $1< a < \sqrt{r} \log p$. Như vậy việc kiểm tra (**) cho các số a nằm trong vùng này sẽ tương đương với việc kiểm tra tính nguyên tố của p và thuật toán có độ phức tạp là đa thức.

2). Giải thuật:

Thuật toán được trình bày như sau:

Input: integer n>1;

1. If (n is of the form ab, b>1) output COMPOSITE;
2. r=2;
3. while (r<n){
4. If (gcd(n,r) ≠ 1) output COMPOSITE;
5. If (r is prime)
6. Let q be the largest prime factor of r-1;
7. If (q≥ 4√r log n) and (n^((r-1)/2) ≡/ 1 (mod r)) break;
8. r← r + 1;
9. }
10. For a= 1 to 2 √r log n
11. if ((x-a)n ≡/ (xn – a)(mod xr – 1, n)) output COMPOSITE;
12. Output PRIME;

Thuật toán AKS trên đây được chứng tỏ là có độ phức tạp thời gian là đa thức cỡ $O(\log^{12}n)$ khi thử trên số n bất kỳ, nhưng nếu thử với số nguyên có dạng Sophie German (dạng 2^p+1) thì độ phức tạp sẽ chỉ cỡ $O(\log^{12}n)$

2.4 Kết luận chương:

Chương 2 đã nêu ra khái niệm của một số nguyên tố, tính chất và định lý cơ bản của số nguyên tố, sự phân bố của số nguyên tố trong số học. Đồng thời chương này giúp em tìm hiểu được các số nguyên tố có dạng dặc biệt như Mersenne. Fermat, Trong mersenne chương nêu ra được danh sách số nguyên tố đã biết và số liệu thông kê của người tìm ra và ưu-nhược điểm của Mersenne. Cuối chương nêu ra một số phương pháp kiểm tra số nguyên tố, kiểm tra lucas-lehmer. ta có thể biết qua vài phương pháp như, phương pháp cổ điển và phương pháp xác xuất. các định lý và khái niệm ở chương 2 là tiền đề để tiếp tục xây dựng chương trình ở chương tiếp theo

CHƯƠNG 3: ỨNG DỤNG CỦA SỐ NGUYÊN TỐ VÀ THỬ NGHIỆM CHƯƠNG TRÌNH

3.1. THỬ NGHIỆM CHƯƠNG TRÌNH

3.1.1. Cấu hình hệ thống.

- Yêu cầu phần cứng: Phần cứng tối thiểu đề nghị cho tất cả máy dự cài đặt và sử dụng chương trình Demo

+ Tối thiểu là i3 3210.

+ Dung lượng RAM đề nghị 8gb.

+ Dung lượng cần để cài đặt chương trình 15GB.

- Yêu cầu phần mềm : Chương trình chạy trên nền tảng hệ điều hành Windows, Vsual Studio, cần cài đặt ASP.NET MVC, Desktop Devellogment with C++, C#.

3.1.2. Chức năng chính.

Chương trình cho phép thực hiện các chức năng sau :

- Kiểm tra một số nguyên tố.

- Tính số nguyên tố dạng Mersenne 2^N-1.

- Liệt kê các số nguyên tố trong một phạm vi.

- Phân tích thừa số nguyên tố.

***Lưu ý**: Chương trình em viết các số nguyên tố bằng kiểu int(32bit)(-2,147,483,648 tới 2,147,483,647). Nên sẽ có một hạn chế đó là các số lớn hơn kiểu int, thì chương trình sẽ không kiểm tra được số nguyên tố

3.1.3 Kiểm tra một số nguyên tố:

Nhập các dãy số từ 2 – 5000 vào kiểm tra chương trình

Tìm số nguyên tố trong phạm vi 1-5000

Qua kiểm tra chương trình ta có :

Kết quả thu về 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101…4999 đây là dãy số nguyên tố từ 1- 5000.

Hình 3.1.3 Kiểm tra số nguyên tố 2,5000

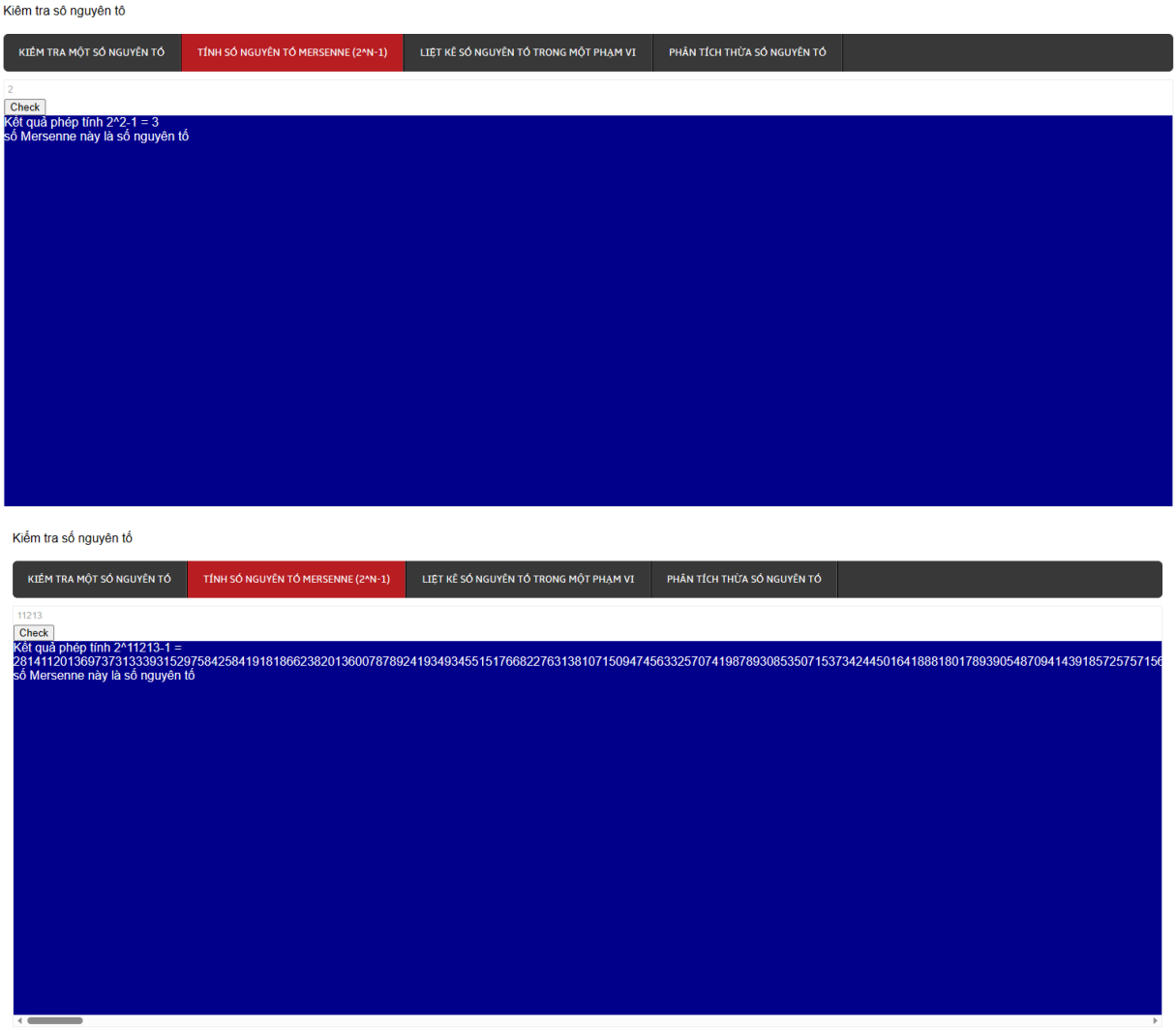
3.1.4 Tính số nguyên tố dạng Mersenne (2^n-1)

Theo số nguyên tố dạng Mersenne phát biểu trong chương (**2.2.1.2**) các số nguyên dạng 2^n-1 là những số nguyên tố nếu:

p∈{ 2,3,5,7,13,17,19,31,61,89,107, 521, 607,1279,2203,2281,3217,4253, 4423, 9689, 9941,11213,19937, 21701, 23209, 44497, 86243,}. Theo Wikipedia

Sử dụng kiểm tra Lucas-Lehmer vào thuật toán.

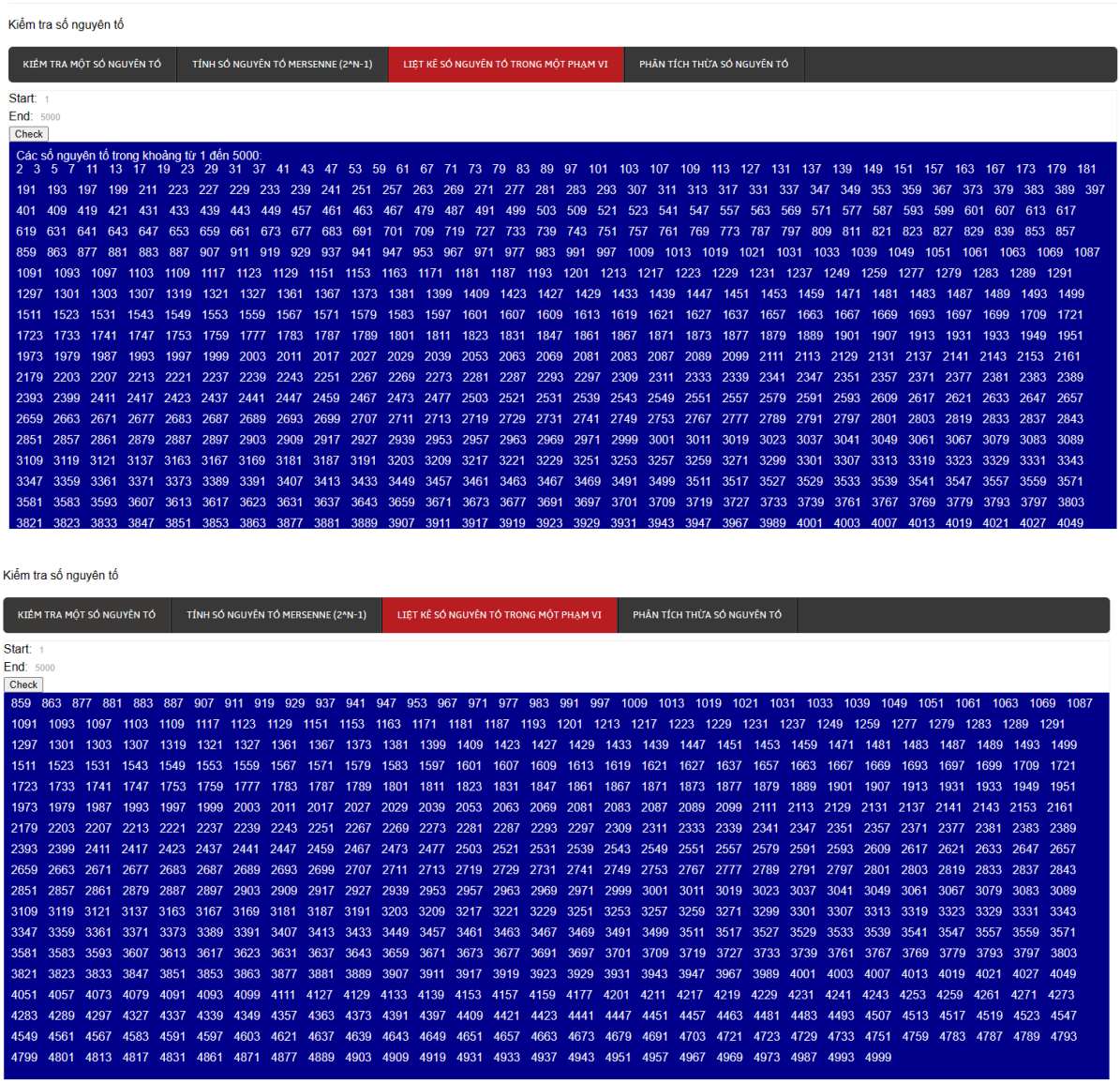
Tiến hành kiểm tra dãy số:



Hình 3.1.4. Kết quả trả về của chương trình tính số nguyên tố 2^n-1của 2, 11213.

3.1.5. Liệt kê số nguyên tố trong 1 phạm vi:

Kiêm tra trong khoảng số từ 1->5000:

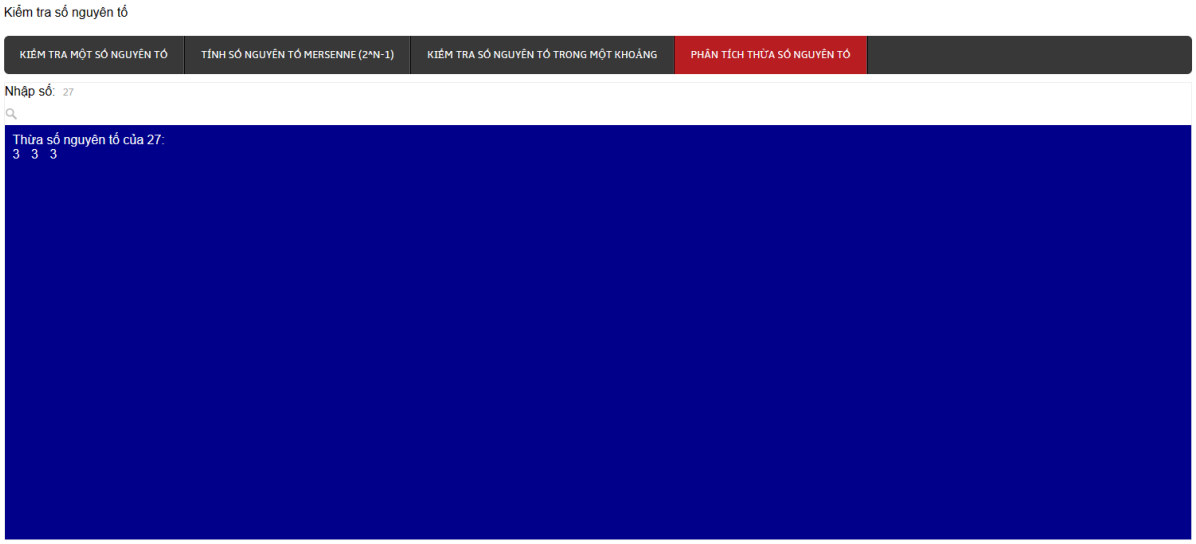


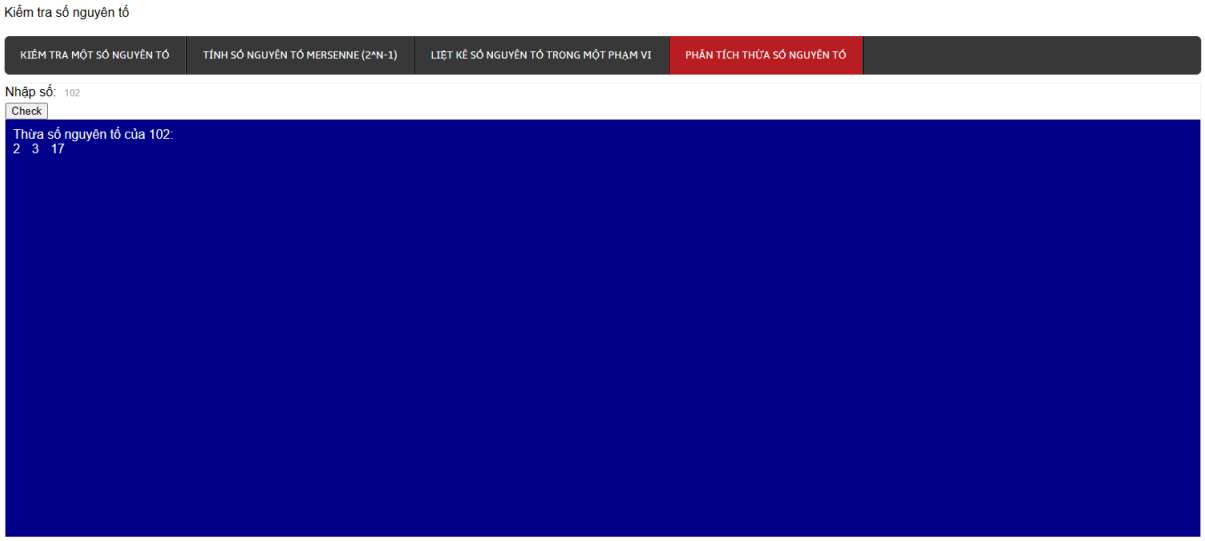
Hình 3.1.5. Kết quả nhận được sau khi liệt kê từ 1 -> 5000.

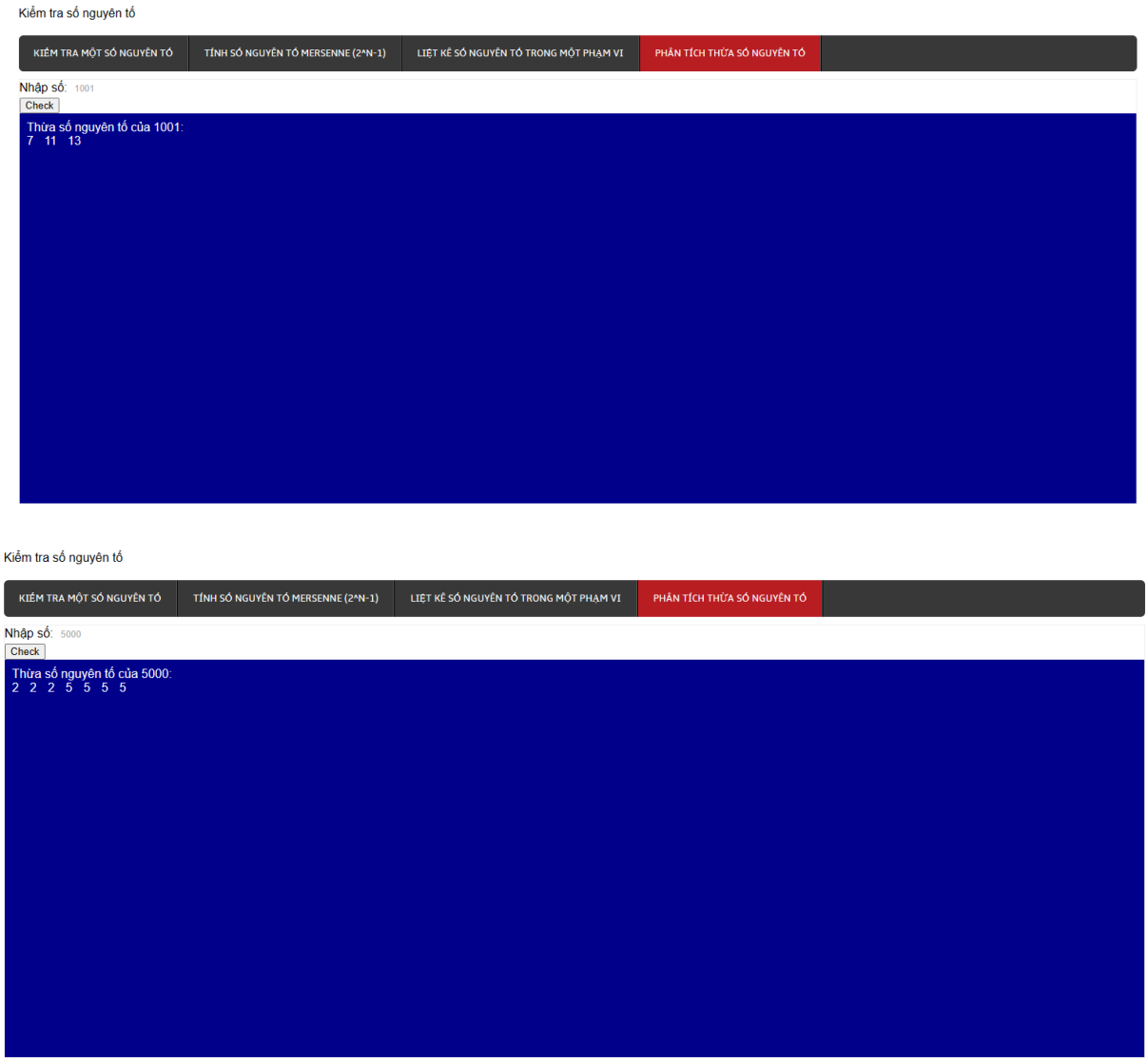
3.1.6. Phân tích thừa số nguyên tố:

Xét các thừa số nguyên tố: 27,102,1001,5000:

Kết quả thu được: 27 = 3x3x3, 102 = 2x3x17, 1001 = 7x11x13 .







Hình 3.1.6 Phân tích thừa số nguyên tố 27,102,1001,5000.

3.2 Kết luận chương

Chương 3 chạy thử nghiệm chương trình Kiểm tra các số nguyên tố dạng Mersenne (2^n-1). Chương trình phân tích ra các thừa số nguyên tố và kiểm tra một số có phải số nguyên tố hay không, Tính số nguyên tố dạng Mersenne 2^n-1 theo kiểm tra lucas-lehmer. Liệt kê số nguyên tố trong một phạm vi bằng phương pháp sàng eratosthenes. Phân tích thừa số ra các thừa số nguyên tố.

KẾT LUẬN

Em đã trình bày ý nghĩa vai trỏ của số nguyên tố trong an toàn - bảo mật thông tin. Trên sơ đồ em đã khảo sát một số thuật toán phân tích số nguyên tố lớn. Có rất nhiều thuật toán phân tích số nguyên tố tuy nhiên các thuật toán đó không đáp ứng được yêu cầu khi số nguyên tố đủ lớn. Ví dụ thuật toán Sàng Bình Phương là thuật toán mạnh nhất hiện nay nhưng chúng chỉ xác định được một số nguyên tố là số nguyên tố khi nó nhỏ hơn nhỏ hơn hoặc bằng 1 hàm số thập phân còn nếu một số nguyên tố đủ lớn hơn 100 chữ số thập phân thì chương trình chạy với thuật toán đó không khả thi đối với máy tính thông thường hiện nay. Đồ án của em đã cho phép xác định một số là số nguyên tố hay không có độ rất lớn với thời gian chạy thực tế cho kết quả đúng nếu só dó có dạng 2^{n-1}. Trong đó n là số nguyên tố.

*Hạn chế:

Mặc dù em rất cố gắng nhưng do thời gian bị hạn chế và sự hiểu biết của em có hạn nên chỉ dừng lại ở tìm hiểu các dạng số nguyên tố đã được phát triển và hình thành từ trước. nên có thể trong đồ án em còn một số hạn chế. Em rất mọng nhận được ý kiến đóng góp đến từ thây cô giáo. Em xin chân thành cảm ơn các thầy cô

CPTÀI LIỆU THAM KHẢO

[1.]Alfred J. Menezes. Paul C. van Ooschot, Scott A. Vanstone: “Handbook of applied Cryptography”, CPC press: Boca Raton-Newyork-london-Tokyo, 1998.

[2.]L.MAdleman(1994): “the function field Sieve:(1994),” Number Theory(LNCS 877). 108-121(1994).

[3.]Neal Koblitz(2000): “ Cơ sở của lý thuyết số và mật mã(bản dịch anh-việt)”. Springer-Veilog: Newyork, Bedtine, Heiderberg London, Paris, Tokyo (2000).

12.04 LTS Install Guide, Symmetrix Technologies.

[4.]Hồ Văn Canh (chủ biên), Lê Danh Cường (2018) : ‘’ Mật mã và an toàn thông tin’’. Nhà xuất bản thông tin – Truyền Thông ; 2018

[5.]Học viện kỹ thuật mật mã (2008), “Bộ giao thức TCP/IP”, Học viện kỹ thuật mật mã, Hà Nội[8.]https://fossies.org/linux/snort/configure.in

[6.]https://vi.wikipedia.org/wiki/S%E1%BB%91_Lucas#:~:text=S%E1%BB%91%20nguy%C3%AAn%20t%E1%BB%91%20Lucas%20l%C3%A0,s%E1%BB%91%20A005479%20trong%20b%E1%BA%A3ng%20OEIS)

[7.]https://www.snort.org

[8.]https://seclists.org/snort/

[9.]https://fossies.org/linux/snort/configure.in

[10.]https://www.academia.edu/4302986/Cai_d%E1%BA%B7t_Snort_Barnyard_BASE_tren_Cent_OS_5_2