

Số học - thuật toán

Lý thuyết số, hay số học là lĩnh vực nghiên cứu về các số nguyên. Trong tài liệu này, chúng ta sẽ đề cập đến một số kiến thức và các thuật toán về số học thường gặp, bao gồm từ những vấn đề cơ bản.

1. Số nguyên tố

1.1. Nếu p là ước nguyên tố bé nhất của n , thì $p^2 \leq n$

($n=pq$, mà $p \leq q$, do đó $p^2 \leq pq = n$)

1.2. Thuật toán kiểm tra tính nguyên tố

Từ 1.1. ta có thuật toán kiểm tra tính nguyên tố của một số n , chạy trong thời gian $O(n^{1/2})$:

```
function isprime(n) : boolean;
begin
    isprime:=false;
    if (n<=2) then exit;
    i:=2;
    while (i*i<=n) do
    begin
        if n mod i = 0 then
            exit;
        inc(i);
    end;
    isprime:=true;
end;
```

1.3. Phân tích ra thừa số nguyên tố

Cũng từ 1.1., ta có thuật toán phân tích một số n ra thừa số nguyên tố:

```
procedure factor(n);
begin
    i:=2;
    while (i*i<=n) do
    begin
        if (n mod i = 0) then
        begin
            a:=0;
            while (n mod i = 0) do
            begin
                n:=n div i;
                inc(a);
            end;
        end;
    end;
```

```

        inc(m);
        prime[m]:=i;
        power[m]:=a;
    end;
end;
if (n>1) then
begin
    inc(m);
    prime[m]:=n;
    power[m]:=1;
end;
end;

```

Thông tin được trả về trong mảng prime và power: prime[i], power[i] tương ứng cho biết thừa số và số mũ thứ i trong phép phân tích n ra thừa số nguyên tố.

Lưu ý những dòng màu đỏ: sau khi thực hiện các phép chia, n có thể còn là một thừa số nguyên tố độc lập.

Đây là phương pháp phân tích đơn giản nhất, được gọi là phép *thử chia*. Trong trường hợp xấu nhất, n là số nguyên tố, thuật toán chạy trong thời gian $O(n^{1/2})$

1.4. Sàng số nguyên tố

Khi cần biết các số nguyên tố đến một phạm vi nào đó, ví dụ từ 2 đến 10^8 , sử dụng sàng số nguyên tố Eratosthenes sẽ hiệu quả hơn về thời gian.

Thuật sau tạo sàng số nguyên tố từ 2 đến N:

```

procedure sieve(n);
begin
    fillchar(p, sizeof(p), true);
    for i:=2 to n do
        if (p[i]) then
            begin
                j:=i+i;
                while (j<=n) do
                    begin
                        p[j]:=false;
                        j:=j+i;
                    end;
            end;
    end;
end;

```

Thông tin trả về trong mảng p: p[i] = true nếu và chỉ nếu i là số nguyên tố.

Bạn có thể ước lượng thời gian chạy của thuật toán sàng Eratosthenes là $O(n \log n)$, ta sẽ đề cập đến một số ước lượng cần thiết ở những phần sau.

1.5. Ước lượng số số nguyên tố

Kí hiệu $\pi(n)$ là số số nguyên tố bé hơn n . Đây là một ước lượng tương đối tốt và ngắn gọn cho $\pi(n)$:

$$\pi(n) \approx n / \ln(n)$$

Ví dụ, $\pi(10^6) \approx 10^6 / \ln(10^6) \approx 72382$

Con số này sẽ giúp cho việc ước lượng thời gian tính toán cho các bài toán liên quan đến số nguyên tố

1.6. Bài tập

Cho một dãy số nguyên dương n phần tử: a_1, a_2, \dots, a_n

Dãy con của dãy số là dãy nhận được sau khi xóa đi một số phần tử nào đó

Yêu cầu: Tìm dãy con dài nhất, sao cho tổng của hai số liên tiếp là số nguyên tố

Input:

NT1.IN

Dòng 1: n

Dòng 2: n số nguyên dương, a_1, a_2, \dots, a_n

Output:

NT1.OUT

Dòng 1: độ dài của dãy con tìm được

Giới hạn:

- $n \leq 10^4$
- $a_i \leq 10^4$

2. USCLN, thuật toán Euclid

2.1. $(a, b) = (b, a \bmod b)$

Thật vậy, từ đẳng thức

$$a = bq + r.$$

với $r = a \bmod b$

Ta thấy mọi ước chung d của a, b cũng là ước chung của b, r . Do đó $(a, b) = (b, r)$.

2.2. Thuật toán tìm USCLN

Từ 2.1, ta có thuật toán Euclid tìm USCLN của a và b , viết dưới dạng đệ quy:

```
function gcd(a, b);
begin
    if (a < b) then
        gcd := gcd(b, a)
    else if (b = 0) then
```

```

        gcd:=a
    else
        gcd:=gcd(b,  a mod b);
end;

```

Với $a, b \leq n$, bạn có thể ước lượng thời gian thực hiện thuật toán vào khoảng $O(\log_{10}n)$, tức là tỉ lệ với số chữ số của n .

2.3.

Nếu $(a, b)=d$, thì tồn tại hai số nguyên x, y sao cho

$$ax + by = d$$

2.4. Thuật toán Euclid mở rộng

Thuật toán Euclid mở rộng sẽ tìm USCLN d của a và b , đồng thời tìm được cả hai số nguyên x, y trong phần 2.3

Thuật toán Euclid mở rộng có thể diễn đạt bằng đệ quy như sau:

```

procedure ee(a, b, var x, var y);
var
    x2, y2;
begin
    if (a<b) then
        ee(b, a, x, y)
    else if (b=0) then
        begin
            x:=1;
            y:=0;
        end else
        begin
            ee(b, a mod b, x2, y2);
            x:=y2;
            y:=x2-(a div b)*y2;
        end;
end;

```

Giải thích:

Từ 2.1, ta đã biết $(a, b) = (b, r) = d$

$ee(a, b, \text{var } x, \text{var } y)$ trả về giá trị x, y sao cho $ax + by = d$

Dòng lệnh màu đỏ chạy thủ tục đệ quy: tìm x_2, y_2 sao cho:

$$bx_2 + ry_2 = d$$

Mặt khác:

$$r = a - bq$$

Với

$$r = a \bmod b$$

$$q = a \div b$$

Do đó

$$\begin{aligned}bx_2 + (a-bq)y_2 &= d \\ ay_2 + b(x_2 - qy_2) &= d\end{aligned}$$

Vậy

$$\begin{aligned}x &= y_2 \\ y &= x_2 - qy_2\end{aligned}$$

Cấu trúc đệ quy của thuật toán Euclid mở rộng cũng tương tự như thuật toán Euclid.

2.5. Một số tính chất

Giả sử

$$\begin{aligned}a &= p_1^{a_1} p_2^{a_2} \dots p_k^{a_k} \\ b &= p_1^{b_1} p_2^{b_2} \dots p_k^{b_k}\end{aligned}$$

Định nghĩa:

$$\begin{aligned}\text{USCLN: } (a, b) &= p_1^{\min(a_1, b_1)} p_2^{\min(a_2, b_2)} \dots p_k^{\min(a_k, b_k)} \\ \text{BSCNN: } [a, b] &= p_1^{\max(a_1, b_1)} p_2^{\max(a_2, b_2)} \dots p_k^{\max(a_k, b_k)}\end{aligned}$$

Tính chất:

- $(a, b) \times [a, b] = ab$
- $(a, b, c) = ((a, b), c) = (a, (b, c))$
- $[a, b, c] = [[a, b], c] = [a, [b, c]]$

Chú ý:

- Không có đẳng thức $(a, b, c) [a, b, c] = abc$

2.6. Bài tập

Cho dãy số nguyên dương n phần tử a_1, a_2, \dots, a_n .

Yêu cầu:

Tìm dãy con liên tiếp dài nhất có USCLN > 1

Input:

NT2.INP

Dòng 1: n

Dòng 2: n số nguyên dương, a_1, a_2, \dots, a_n

Output:

NT2.OUT

Dòng 1: độ dài của dãy con tìm được

Giới hạn:

- $n \leq 30000$
- $0 < a_i \leq 32767$

3. PT, HPT đồng dư

3.1. Nghịch đảo

Trở lại với thuật toán Euclid mở rộng:

Giả sử ta thực hiện `ee(a, m, var x, var y)`

Trong trường hợp $(a, m) = 1$, ta thu được giá trị x, y sao cho:

$$ax + my = 1$$

Hay

$$ax \equiv 1 \pmod{m}$$

$(a, m) = 1 \Leftrightarrow \exists x, ax \equiv 1 \pmod{m}$
 x được gọi là nghịch đảo của a theo modulo m , ký hiệu a^{-1}
 Để tìm x , ta sử dụng thuật toán Euclid mở rộng

3.2. Phương trình đồng dư bậc nhất

$$ax \equiv b \pmod{m} \quad (3.2)$$

3.2.1. Trường hợp $(a, m) = 1$

Theo 3.1. $\exists a^{-1}, aa^{-1} \equiv 1 \pmod{m}$

Do đó $aa^{-1}b \equiv b \pmod{m}$

Đặt $x = (a^{-1}b)$ thì x là một nghiệm của (3.2)

Giả sử tồn tại x' , sao cho

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

Suy ra $ax \equiv ax' \pmod{m}$, mà $(a, m) = 1$

Suy ra $x \equiv x' \pmod{m}$

Vậy $x = (a^{-1}b)$ là nghiệm duy nhất của (3.2) theo modulo m

3.2.2. Trường hợp $(a, m) = d$

Nếu d không là ước của b , hiển nhiên (3.2) vô nghiệm

Nếu $b \mid d$, xét phương trình:

$$(a/d)y \equiv (b/d) \pmod{(m/d)}$$

Ta có $(a/d, m/d) = 1$, do đó theo 3.2.1.

$$y \equiv (a/d)^{-1} (b/d) \pmod{(m/d)}$$

Đặt

$$y_0 = (a/d)^{-1} (b/d)$$

(3.2) có d nghiệm:

$$x_t = y_0 + t(m/d) \quad \text{với } t = 0, 1, \dots, d-1$$

theo modulo m

3.3. Định lý phần dư Trung Hoa

Nếu

$$x \equiv a_1 \pmod{m_1}$$

$$x \equiv a_2 \pmod{m_2}$$

...

$$x \equiv a_n \pmod{m_n}$$

và m_1, m_2, \dots, m_n đôi một nguyên tố cùng nhau thì x được xác định duy nhất theo modulo $M = m_1 m_2 \dots m_n$:

$$x \equiv a_1 b_1 c_1 + a_2 b_2 c_2 + \dots + a_n b_n c_n \pmod{M} \quad (3.3)$$

Trong đó

$$c_i = M / a_i \\ b_i = c_i^{-1} \pmod{a_i}$$

Phương pháp để tìm công thức (3.3):

Xét trường hợp $a_2 = a_3 = \dots = a_n = 0$

Cần tìm x_1 :

$$x_1 \equiv a_1 \pmod{m_1}$$

$$x_1 \equiv 0 \pmod{m_2}$$

...

$$x_1 \equiv 0 \pmod{m_n}$$

Ta sẽ tìm được

$$x_1 \equiv a_1 b_1 c_1 \pmod{M}$$

Tương tự, lại xét trường hợp $a_1 = a_3 = \dots = a_n = 0$

$$x_2 \equiv a_2 b_2 c_2 \pmod{M}$$

...

$$x_n \equiv a_n b_n c_n \pmod{M}$$

Tổ hợp các kết quả lại ta thu được công thức (3.3), phương pháp này được gọi là phương pháp chồng.

4. Phép chia hết

4.1.

Số các số nguyên dương không vượt quá n chia hết cho d :

$$\left\lfloor \frac{n}{d} \right\rfloor$$

hay

$$n \text{ div } d$$

4.2. Ước số

Giả sử

$$n = p_1^{a_1} p_2^{a_2} \dots p_k^{a_k}$$

4.2.1. Số ước

$$\nu(n) = (a_1 + 1)(a_2 + 1) \dots (a_k + 1)$$

4.2.2. Tích các ước

$$\prod d = n^{\nu(n)/2}$$

Thật vậy, viết n dưới dạng tích hai thừa số:

$$n = \begin{cases} d_1 d'_1 \\ d_2 d'_2 \\ \dots \\ d_v d'_v \end{cases}$$

Do đó

$$n^{v(n)} = \left(\prod d\right)^2$$

hay

$$\prod d = n^{v(n)/2}$$

4.2.3. Tổng các ước

$$\sigma(n) = \prod_i \frac{p_i^{a_i+1} - 1}{p_i - 1} \quad (4.2.3)$$

Thật vậy: nếu $(a, b) = 1$ ta cm được

$$\sigma(ab) = \sigma(a)\sigma(b)$$

Do đó

$$\sigma(n) = \prod_i \sigma(p_i^{a_i})$$

Mà

$$\sigma(p_i^{a_i}) = 1 + p_i + p_i^2 + \dots + p_i^{a_i} = \frac{p_i^{a_i+1} - 1}{p_i - 1}$$

Từ đó thu được công thức (4.2.3)

5. Số Fibonacci

5.1. Cách tính nhanh F_n

Viết dưới dạng tích hai ma trận:

$$\begin{vmatrix} 0 & 1 \\ 1 & 1 \end{vmatrix}^n X \begin{vmatrix} F_{n-1} \\ F_n \end{vmatrix} = \begin{vmatrix} F_n \\ F_{n+1} \end{vmatrix}$$

Đặt

$$A = \begin{vmatrix} 0 & 1 \\ 1 & 1 \end{vmatrix} \quad v = \begin{vmatrix} 1 \\ 1 \end{vmatrix} = \begin{vmatrix} F_1 \\ F_2 \end{vmatrix}$$

Ta có công thức:

$$A^{n-1}v = \begin{vmatrix} F_n \\ F_{n+1} \end{vmatrix} \quad (5.1)$$

Mà A^{n-1} có thể tính trong thời gian $O(\log n)$, do đó công thức (5.1.) cho phép ta tính F_n trong thời gian $O(\log n)$

5.2. Một số kết quả thú vị

5.2.1. UCLN của F_m, F_n

Công thức Lucas:

$$(F_m, F_n) = F_{(m, n)}$$

5.2.2. Xác định một số có phải là số Fibonacci

Gessel (1972):

n là số Fibonacci nếu và chỉ nếu $5n^2 + 4$ hoặc $5n^2 - 4$ là số chính phương

5.3. Biểu diễn Zeckendorf

5.3.1. Định lý Zeckendorf:

Mọi số nguyên dương đều được biểu diễn duy nhất dưới dạng tổng các số Fibonacci, trong đó không có hai số Fibonacci liên tiếp, nghĩa là dưới dạng:

$$n = \sum_k a_k F_k$$

với

$$a_k = 0 \text{ hoặc } a_k = 1$$

và

$$a_k a_{k+1} = 0$$

5.3.2. Ví dụ:

$$100 = 89 + 8 + 3$$

5.3.3. Thuật toán

Tìm biểu diễn Zeckendorf, hay còn gọi là biểu diễn dưới dạng cơ số Fibonacci của n :

```
while (n>0) do
begin
    f là số fibonacci lớn nhất không vượt quá n;
    chọn f vào biểu diễn;
    n:=n-f;
end;
```

hay ta có thể cài đặt như sau:

```
for i:=max downto 1 do
    while (Fi <= n) do
        begin
            chọn Fi;
            n:=n-Fi;
        end;
```

với max là chỉ số lớn nhất của số Fibonacci trong giới hạn làm việc

Tính đúng đắn:

Thuật toán tham 5.3.3. sẽ không bao giờ chọn hai số Fibonacci liên tiếp, thật vậy, giả sử thuật toán chọn F_{n-1} , F_{n-2} vào tổng, thì do ta đi qua danh sách số Fibonacci theo thứ tự giảm dần, thuật toán ắt đã chọn $F_n = F_{n-1} + F_{n-2}$ thay vì F_{n-1} , F_{n-2}

5.4. Bài tập:

5.4.1. <http://acm.uva.es/p/v9/948.html>

6. Tham khảo

Trên đây chỉ là một số vấn đề về số học - thuật toán thôi, các bạn nên tìm hiểu thêm, từ bất kỳ nguồn nào, internet, sách vở. Nếu có những vấn đề, những bài tập hay, hãy đóng góp lại cho mọi người!

Một số nguồn để các bạn tham khảo thêm:

Concerte Mathematics – A Foundation for Computer Science

Mathworld

Wikipedia

Thuật ngữ, ghi chú

1.

Số nguyên tố Kiểm tra tính nguyên tố Phân tích ra thừa số nguyên tố Sàng	PRIME PRIMALITY TEST PRIME FACTORIZATION SIEVE
---	---

Có rất nhiều thuật toán kiểm tra & phân tích ra thừa số nguyên tố hiệu quả hơn, tuy nhiên những gì chúng ta trình bày là những thuật toán đơn giản nhất, sẽ sử dụng trong các bài tập và kì thi.

1.5. Xem PRIME NUMBER THEOREM

2.

USCLN Thuật toán Euclid Thuật toán Euclid mở rộng BSCNN	Greatest Common Divisor (GCD) Euclidean Algorithm Extended Euclidean Algorithm Least Common Multiple (LCM)
--	---

2.1. Về thời gian chạy của thuật toán Euclid, xem thêm LAMÉ'S THEOREM

2.3. Xem thêm BÉZOUT'S LEMMA

3.

Phương trình đồng dư Định lý phần dư Trung Hoa	Congruence Equation Chinese Remainder Theorem
---	--

5.

Biểu diễn Zeckendorf	Zeckendorf Representation
----------------------	---------------------------

