Hàm phi Euler và ứng dụng

Trần Việt Khoa
tvkhoa.husc@gmail.com,
Khoa Công nghệ thông tin, Đại học Khoa học Huế

Ngày 15 tháng 3 năm 2017

1 Hàm phi Euler

Định nghĩa: hàm phi Euler, ký hiệu $\phi(n)$ được tính bằng số các số nguyên từ 1 đến n mà nguyên tố cùng nhau với n. Hai số nguyên tố cùng nhau nếu ước số chung lớn nhất của chúng bằng 1. Xem [1]

Ví dụ: $\phi(5) = 4$ vì gcd(1, 5)=1, gcd(2, 5)=1, gcd(3, 5)=1 và gcd(4, 5)=1. Trong đó hàm gdc(a,b) tính ước số chung lớn nhất của hai số a và b.
Bảng các giá trị của $\phi(n)$ của vài số nguyên dương đầu tiên:

N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$\phi(n)$	1	1	2	2	4	2	6	4	6	4	10	4	12	6	8	8	16	6	18

Các thuộc tính của hàm phi Euler:

• Nếu p là một số nguyên tố thì $\phi(p) = p - 1$, dễ thấy vì $gcd(p, q) = 1\ \forall q :\ 1 \leq q \leq p$.

• Nếu p là một số nguyên tố và $k \geq 1$ thì $\phi(p^k) = p^k - p^{k-1}$.

• Nếu a và b nguyên tố cùng nhau thì $\phi(a \times b) = \phi(a) \times \phi(b)$ theo lý thuyết của định lý số dư Trung hoa.

Như vậy ta có thể tính được hàm phi Euler của n thông qua phân tích n thành các thừa số nguyên tố. Nếu $n = p_1^{a_1} \times p_2^{a_2} \times ... \times p_k^{a_k}$ (với các số p_i là các thừa số nguyên tố của n) thì $\phi(n) = \phi(p_1^{a_1}) \times \phi(p_2^{a_2}) \times ... \times \phi(p_k^{a_k}) = (p_1^{a_1} - p_1^{a_1-1}) \times (p_2^{a_2} - p_2^{a_2-1}) \times ... \times (p_k^{a_k} - p_k^{a_k-1}) = n \times (1 - \frac{1}{p_1}) \times (1 - \frac{1}{p_2}) \times ... \times (1 - \frac{1}{p_k})$.

Ví dụ: Tính $\phi(616)$, ta có $616 = 2^3 \times 7 \times 11$, áp dụng công thức trên ta có:

$\phi(616) = 616 \times (1 - \frac{1}{2}) \times (1 - \frac{1}{7}) \times (1 - \frac{1}{11}) = 616 \times \frac{1}{2} \times \frac{6}{7} \times \frac{10}{11} = 240$.

Bản cài đặt bằng C++ với độ phức tạp phụ thuộc vào phân tích thừa số nguyên tố $O(\sqrt{n})$:

```
int phi(int n) {
    int result = n;
    for(int i = 2; i * i <= n; ++i)
        if(n % i == 0) {
            while(n % i == 0)
                n /= i;
            result -= result / i;
        }
```

```
    if(n > 1)
        result -= result / n;
    return result;
}
```

Bài tập 1: tính hàm phi Euler, địa chỉ nộp bài http://www.spoj.com/problems/ETF/

2 Nghịch đảo modular

Cho một số nguyên dương m và một số nguyên a nguyên tố cùng nhau với m. Số a^{-1} được gọi là phần tử nghịch đảo modular nếu thỏa: $a \times a^{-1} \equiv 1 \ (mod\ m)$. Xem [2]

Ví dụ: a = 3, m = 11, hai số này nguyên tố cùng nhau, dễ thấy có $a^{-1} = 4$ vì $3 \times 4 \equiv 1 \ (mod\ 11)$, ngoài ra ta cũng tìm được $a^{-1} = 15$ thỏa $15 \times 3 \equiv 1 \ (mod\ 11)$ nhưng phần tử này không thuộc trong vành cần tìm {0, 1, 2, ... 10} nên không phải là phần tử cần tìm.

Ví dụ trên dễ dàng nhẩm tính được phần tử nghịch đảo, tuy nhiên với dữ liệu lớn thì cần có thuật toán xác định phần tử này. Có nhiều thuật toán để giải quyết bài toán này, tuy nhiên thuật toán hiệu quả nhất là sử dụng định lý Euler và định lý Fermat nhỏ.

Định lý Euler: $a^{\phi(m)} \equiv 1(mod\ m)$ nếu a và m nguyên tố cùng nhau.

Định lý Fermat nhỏ: trong trường hợp m là số nguyên tố thì từ định lý Euler ta có: $a^{m-1} \equiv 1(mod\ m)$.

Nhân hai vế với a^{-1} cho cả hai công thức của hai định lý trên ta có:

- với modulo cho m bất kỳ: $a^{\phi(m)-1} \equiv a^{-1}(mod\ m)$ và,

- với modulo cho m là số nguyên tố: $a^{m-2} \equiv a^{-1}(mod\ m)$.

Bài toán quy về bài lũy thừa nhị phân, thuật toán cài đặt như sau:

```
long long modInverse(long long val, int mod) {
   return powMod(val, mod - 2, mod);
}
```

BÀI TOÁN (Tổ hợp chập k modulo n)[1]

Phát biểu

Tính tổ hợp chập k từ n phần tử, ký hiệu C_n^k , kết quả modolu cho m = 1000003.

Dữ liệu vào

- dòng thứ nhất là T ($T \leq 2000$) là số testcase.

- mỗi testcase là hai số nguyên n ($1 \leq n \leq 10^6$) và k ($0 \leq k \leq n$) cách nhau ký tự trắng.

Kết quả

- in ra kết quả cần tìm cho mỗi testcase, kết quả modolu cho m=1000003.

[1] http://www.lightoj.com/volume_showproblem.php?problem=1067&language=english&type=pdf

Ví dụ

Dữ liệu vào	Kết quả
3 4 2 5 0 6 4	Case 1: 6 Case 2: 1 Case 3: 15

Lời giải Đây là bài toán quen thuộc, nhưng các bạn không thể giải theo phương pháp quy hoạch động dựa trên công thức đệ quy $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$ do T, n quá lớn.

Nhận thấy $m = 10^6 + 3$ là số nguyên tố, ngoài ra công thức tính tổ hợp $C_n^k = \frac{n!}{(n-k)! \times k!}$.

Nếu đặt $A = (n-k)! \times k!$ thì A nguyên tố cùng nhau với m (vì m là số nguyên tố), theo định lý Fermat nghịch đảo của A là A^{m-2}, vậy $C_n^k \ mod \ m = (n! \times A^{m-2}) \ mod \ m$.

BÀI TOÁN (Lũy thừa tháp)[2]

Phát biểu

Cho a, b, c. Thực hiện các phép tính sau:

$result1 = b^c$,

$result2 = a^{result1}$.

Vì thực hiện phép lũy thừa nên kết quả là số rất lớn, để giải quyết khó khăn đó sao cho vẫn xử lý được khi lập trình người ta đưa ra ý tưởng là modulo kết quả cho một số nguyên tố và số phù hợp là $m = 10^9 + 7$. Bạn hãy giải quyết vấn đề trên.

Chú ý: giả thiết rằng $0^0 = 1$

Dữ liệu vào

- gồm nhiều testcase, mỗi testcase là một bộ ba số a, b, c cách nhau ký tự trắng $0 \leq a, b, c \leq 2^{31} - 1$.

- bộ a=b=c=-1 dùng để kết thúc việc input.

Kết quả

- in ra kết quả cần tìm cho mỗi testcase, kết quả modolu cho m.

Ví dụ

Dữ liệu vào	Kết quả
2 2 2 3 4 5 -1 -1 -1	16 763327764

Lời giải

Bài toán được giải bằng công thức sau: $a^{b^c} \ (mod \ M)$, và được chia thành hai bước:

Bước 1: tính $b^c \ (mod \ M)$, trong trường hợp này do b và M nguyên tố cùng nhau(M là số nguyên tố), nên ta có thể áp dụng Fermat nhỏ với hàm $\phi(M) = M - 1$.

[2]http://www.spoj.com/problems/POWERUP/

result1 = powMod(b,c, M-1);

Bước 2: đơn giản hơn nhiều, ta có: result2 = powMod(a, result1, M);

```
while(true){
   long long a , b , c;
   cin >> a >> b >> c;
   if(a == -1 && b == -1 && c == -1) break;
   if(c == 0) cout << (a % M) <<endl;
   else
      if(b == 0) cout <<"1" << endl;
      else
         if(a % M == 0) cout <<"0"<<endl;
         else {
            long long result1 = powMod( b , c , M - 1);
            long long result2 = powMod(a , result1 , M);
            cout << result1 << endl;
         }
}
```

Bài tập 2: Difficult math (ACM Vòng Miền Bắc 2016), bạn nộp bài tại địa chỉ oj.hueuni.edu.vn.

3 Định lý số dư Trung Hoa

Định lý số dư Trung hoa bắt đầu bằng bài toán thú vị sau: Một cô gái đem một giỏ trứng đi chợ bán, không may bị một thanh niên đi ngựa là đổ hỏng hết trứng. Anh thanh niên phải đền tiền cho cô giá và hỏi rằng rổ trứng có bao nhiêu quả để đền tiền, cô gái bảo không nhớ chính xác chỉ nhớ rằng nếu chia cho 3 thì dư 2 quả, chia cho 5 thì dư 3 quả và chia cho 7 thì dư 2 quả. Nhờ bạn tính giúp.

Gọi x là số cần tìm, ta có hệ phương trình sau:

$$\begin{cases} x \equiv 2(\ mod\ 3) \\ x \equiv 3(\ mod\ 5) \\ x \equiv 2(\ mod\ 7) \end{cases}$$

Hệ phương trình trên được tổng quát hóa như sau: cho $n_1, n_2, \ldots, n_r$ là các số nguyên dương thỏa: $n_i \perp n_j, \forall i \neq j$ [3]. Hệ phương trình đồng dư tuyến tính:

$$\begin{cases} x \equiv b_1(\ mod\ n_1) \\ x \equiv b_2(\ mod\ n_2) \\ \ldots\ldots \\ x \equiv b_r(\ mod\ n_r) \end{cases}$$

Giải:

Bước 1. Tính $N = \prod_{i=1}^{r} n_i$ và $N_k = \dfrac{N}{n_k}\ for\ k = 1, 2, \ldots, r.$

Bước 2. Nghiệm $x \equiv b_1 c_1 \dfrac{N}{n_1} + \cdots + b_r c_r \dfrac{N}{n_r} \pmod{N}$.

[3] ký hiệu $n_i \perp n_j$ dùng cho hai số nguyên tố cùng nhau

Trong đó c_i được xác định từ công thức $c_i \frac{N}{n_i} \equiv 1 \pmod{n_i}$. Hay c_i là nghịch đảo modulo n_i của $\frac{N}{n_i}$. Sử dụng hàm modInverse để tính, ví dụ: modInverse(35,3) = 2

Xét ví dụ trên, ta có: $N = 3 \cdot 5 \cdot 7 = 105, \frac{N}{n_1} = 35, \frac{N}{n_2} = 21, \frac{N}{n_3} = 15.$

$c_1 = 2$ vì $35 \cdot 2 \equiv 2 \cdot 2 \equiv 1 \pmod{3}$, tương tự ta có $c_2 = 1, c_3 = 1$

$x \equiv b_1 c_1 \frac{N}{n_1} + \cdots + b_r c_r \frac{N}{n_r} \pmod{N} = 2 \cdot 35 \cdot 2 + 3 \cdot 21 \cdot 1 + 2 \cdot 15 \cdot 1 = 233 \equiv 23 \pmod{105}.$

Vậy x = 23 quả trứng.

Bài tập 3: http://www.spoj.com/problems/MMOD29/vn/

Tài liệu

[1] http://www.geeksforgeeks.org/eulers-totient-function/

[2] https://e-maxx-eng.appspot.com/algebra/module-inverse.html