ĐỊNH LÝ THẶNG DƯ TRUNG HOA TRONG TIN HỌC

Ngày 3 tháng 11 năm 2022

Mục lục

1 Giới thiệu

Định lý thặng dư trung hoa hay còn biết đến là định lý CRT (Chinese Remainder Theorem) được tìm thấy bởi nhà toán học người Trung Quốc Sunzi để giải quyết các phương trình về đồng dư.

2 Phát biểu định lý

Gọi $p = p_1p_2p_3...p_n$, với p_i đôi một nguyên tố cùng nhau. Với số $a < p$, ta có các phương trình đồng dư như sau:

$$a \equiv a_1(modp_1)$$

$$a \equiv a_2(modp_2)$$

$$...$$

$$a \equiv a_n(modp_n)$$

Với a_i là các hằng số. Định lý CRT phát biểu rằng chỉ tồn tại duy nhất **một** giá trị a mod p thỏa mãn các điều kiện trên.

3 Hệ quả

Một hệ quả của định lý thặng dư trung hoa là:

$$x \equiv a(mod p)$$

Thì:

$$x \equiv a_1(mod p_1)$$
$$x \equiv a_2(mod p_2)$$
$$...$$
$$x \equiv a_n(mod p_n)$$

với p_i đôi một nguyên tố cùng nhau

4 Thuật toán Garner

Cho một mảng a_i gồm các số dư của a khi lấy modulo cho $p_1, p_2, ..., p_n$. Vậy ta có thể tìm được số a từ mảng đã cho hay không?

Câu trả lời là có bằng cách sử dụng thuật toán Garner để tìm được a. Thuật toán phát biểu như sau:

Chúng ta sẽ biểu diễn a dưới dạgn:

$$a = x_1 + x_2p_1 + ... + x_kp_1p_2...p_{k-1}$$

Với công thức trên, ta có thể dễ dàng suy ra:

$$a \equiv x_1(mod p_1)$$
$$a \equiv x_1 + x_2p_1(mod p_2)$$
$$...$$
$$a \equiv x_1 + x_2p_1 + ... + x_kp_1p_2...p_{k-2}(mod p_k)$$

Như vậy ta lại có thể suy ra như sau:

$$a_1 \equiv x_1(mod p_1)$$
$$a_2 \equiv x_1 + x_2p_1(mod p_2)$$
$$a_2 - x_1 \equiv x_2p_1(mod p_2)$$

Chia cả 2 vế cho p_1, ta được:

$$\frac{1}{p_1}(a_2 - x_1) \equiv x_2(mod p_2)$$

Gọi $r_{i,j}$ là modulo nghịch đảo của p_i đối với p_j.Thế $r_{1,2}$ vào phương trình trên, ta được:

$$r_{1,2}(a_2 - x_1) \equiv x_2(mod p_2)$$

Tương tự, ta có:

$$x_3 \equiv ((a_3 - x_1)r_{1,3} - x_2)r_{2,3}(modp_3)$$

Bằng quy nạp, ta chứng minh được:

$$x_n \equiv ((((a_n - a_1)r_{1,n} - x_2)r_{2,n} - x_3)...)r_{n-1,n}(modp_n)$$

Vì a có thể rất lớn nên khi tính a cần phải có số lớn.Nhưng ý tưởng là chạy thuật toán $O(n^2)$ để tính giá trị của a.Code mẫu trên C++ như sau:

```
//Code tim cac gia tri cua x
int a[N], x[N], p[N]
for (int i = 0; i < k; ++i) {
    x[i] = a[i];
    for (int j = 0; j < i; ++j) {
        x[i] = r[j][i] * (x[i] - x[j]);

        x[i] = x[i] % p[i];
        if (x[i] < 0)
            x[i] += p[i];
    }
}
```

Listing 1: Code mẫu thuật toán Garner trên C++

Để có thể code số lớn dễ dàng thì ta có thể chuyển qua Python vì ngôn ngữ này có hỗ trợ số lớn. Code tổng thể sẽ nhìn như sau:

```
def find_x(a, p, r):
    x = []
    k = len(a)
    for i in range(k):
        temp = a[i]
        for j in range(i):
            temp = ((temp - x[j])*r[i][j]) % p[i]
            if(temp < 0):
                temp += p[i]
        x.append(temp)
    return x

def find_a(a, p, r):
    x = find_x(a, p, r)
    sum = 0
    for i in range(len(x),0,-1):
        sum = sum*p[i] + x[i]
    return sum

def bipow(a, n, mod):
    if(n==0):
        return 1
    temp = bipow(a,n>>1,mod) % mod
    temp1 = (temp*temp) % mod
    if(n&1):
        return (temp1*a) % mod
    return temp1

```

```
29 def find_r(p):
30     r = []
31     for i in range(len(p)):
32         temp = []
33         for j in range(i):
34             temp.append(bipow(p[j],p[i]-2,p[i]));
35         r.append(temp)
36     return r
```

Listing 2: Code mẫu thuật toán Garner trên Python

5 Công thức khác

Ta còn có thể suy ra a bằng công thức sau:

Gọi $X_i = \frac{p_1 p_2 p_3 ... p_n}{p_i} = \frac{p}{p_i}$. Gọi x_m^{-1} là modulo nghịch đảo của x với m. Ta có:

$$a = a_1 X_1 X_{1p_1}^{-1} + a_2 X_2 X_{2p_2}^{-1} + ... + a_n X_n X_{np_n}^{-1}$$

Dễ thấy nếu xét p_i thì $a_j X_j X_{jp_j}^{-1}$ sẽ chia hết cho p_i với $j \neq i$. Cho nên $a \equiv a_i X_i X_{ipi}^{-1} (mod p_i)$.

Mà $X_{i,p_i}^{-1} \equiv \frac{1}{X_i} (mod p_i)$

Cho nên $X_i X_{ipi}^{-1} \equiv 1 (mod p_i)$ hay $a_i X_i X_{ipi}^{-1} \equiv a_i (mod p_i)$

Chứng minh tương tự thì ta sẽ rút về được phương trình đồng dư ban đầu.

6 Bài tập ví dụ

Codeforces Problem 687 B : The Remainder Game

Tóm tắt đề:

Hôm nay Pari và Arya đang chơi một trò chơi tên là Remainders. Pari chọn hai số nguyên dương x và k, và nói với Arya k chứ không phải x. Arya phải tìm ra giá trị $xmodk$. Có n số cổ $c_1, c_2, ..., c_n$ và Pari phải nói với Arya giá trị của $xmodc_i$ nếu Arya muốn. Với k và các giá trị cổ, hãy cho biết liệu Arya có chiến lược chiến thắng không dựa vào giá trị của x hay không. Nói cách khác, có đúng là Arya có thể biết giá trị của $xmodk$ bất kỳ số nguyên dương x nào không?

Giới hạn: $n, k, c_i <= 10^6$

Lời giải

Ta phân tích n thành nhân tử:

$$k = p_1^{y_1} p_2^{y_2} ... p_n^{y_n}$$

Với p_n là các số nguyên tố. Điều này đảm bảo các p_i đôi một nguyên tố cùng nhau.

Nếu ta biết được lần lượt $xmodp_i^{y_i}$ thì theo định lý thặng dư trung hoa, chỉ có duy nhất một giá trị $xmodk$. Nếu điều kiện đó không thỏa thì có thể có nhiều giá trị $xmodk$

Vậy nếu trong các c_i không có $p_i^{y_i}$ thì sao?

Gọi c_j là bội của $p_i^{y_i}$ nếu có.

Ta có:

$$x \equiv t(modc_j)$$

$$=> x - t \equiv 0(modc_j)$$

$$=> gcd(x - t, c_j) = c_j$$

Mà $c_j \equiv 0(modp_i^{y_i})$, nên :

$$gcd(x - t, p_i^{y_i}) = p_i^{y_i}$$

$$=> x - t \equiv 0(modp_i^{y_i})$$

$$=> x \equiv t(modp_i^{y_i})$$

Vậy là ta kết luận được rằng nếu ta biết được $xmodc_j$ thì ta cũng biết $xmodp_i^{y_i}$

Vậy ta sẽ code sao cho tiện?

Ta chứng minh được số lượng $p_i \leq 8$

Bằng cách phân tích thành nhân tử trong độ phức tạp $O(\sqrt{k})$. Ta có thể tìm được giá trị của $p_i^{y_i}$ và xét từng giá trị c_j để xem giá trị c nào là bội của $p_i^{y_i}$.

Vậy tổng độ phức tạp bài toán là $O(n + \sqrt{k})$

Code mãu:

```
#include<bits/stdc++.h>

using namespace std;
#define rep(i, x, y) for(int i = x; i <= y; ++i)
#define repi(i,x,y) for(int i = x; i >= y; --i)
#define ci(x) int x; cin>> x
#define TC(t) ci(t); while(t--)
#define fi first
#define se second
#define pb push_back
#define all(x) x.begin(), x.end()
#define cii(x, y) ci(x); ci(y)
#define ciii(x, y, z) ci(x); ci(y); ci(z)
#define mp make_pair
typedef long long ll;
typedef vector<int> vi;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
const int mod1 = 998244353;
const int pi = 31, pii = 29, piii = 41;
const int inf = 1e9 + 5;
```

```cpp
const int block = 330;
const int dx[4] = {0, 0, 1, -1};
const int dy[4] = {1, -1, 0, 0};

void readfile(){
    #ifdef ONLINE_JUDGE
    #else
        freopen("text.inp", "r", stdin);
    #endif // ONLINE_JUDGE
}

int n, k;
int a[N];
vector<int> store;
bool ok = 1;

void inp(){
    cin >> n >> k;
    rep(i,1,n)
        cin >> a[i];
}


void process(){
    int m = sqrt(k);
    rep(i,2,m){
        if(k%i==0){
            ll temp = 1;
            while(k%i==0){
                temp *= i;
                k /= i;
            }
            store.pb(temp);
        }
    }
    if(k > 1)
        store.pb(k);
    for(int u : store){
        bool t = 0;
        rep(i,1,n)
            if(a[i]%u==0){
                t = 1;
                break;
            }
        if(!t){
            ok = 0;
            break;
        }
    }
    if(ok) cout << "Yes";
    else cout << "No";
}

int main() {
    readfile();
    ios_base::sync_with_stdio(0); cin.tie(0);
    inp();
```

```
79      process();
80      return 0;
81  }
```

Listing 3: Code mẫu thuật trên C++