ỨNG DỤNG NHÂN MA TRẬN TRONG CÁC BÀI TOÁN TIN HỌC

Contents

I. Lý thuyết ma trận

1.1 Định nghĩa ma trận.

Ma trận là một mảng chữ nhật chứa các số hoặc những đối tượng toán học khác, mà có thể định nghĩa một số phép toán như cộng hoặc nhân trên các ma trận. Hay gặp nhất đó là ma trận trên một trường F là một mảng chữ nhật chứa các đại lượng vô hướng của F. Bài viết này đề cập đến các ma trận thực, tức là ma trận mà các phần tử của nó là các số thực.

$$A= \begin{Bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{Bmatrix}$$

Trong đó A_{ij} €F ($1 \le i \le n$; $1 \le j \le m$), được gọi là một ma trận m hàng n cột với các phần tử trong F. Véc tơ hàng ($\begin{pmatrix} A_{i1} & A_{i2} & .... & A_{in} \end{pmatrix}$ được gọi là hàng thứ i của ma trận A.

Véc tơ cột ($\begin{pmatrix} A_{1j} & A_{2j} & .... & A_{mj} \end{pmatrix}$ được gọi là cột thứ j của ma trận A.

Ma trận trên được gọi là ma trận A=(Aij)$_{m\times n}$. Tập hợp các ma trận m hàng, n cột với các phần tử trong F được kí hiệu là M(m×n,F).

1.2 Phép cộng và phép nhân hai ma trận

Ta định nghĩa phép cộng và phép nhân vô hướng trên M(m×n,F) như sau:

+ Phép cộng hai ma trận A và B:

$$\begin{Bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{Bmatrix} + \begin{Bmatrix} B_{11} & B_{12} & \cdots & B_{1n} \\ B_{21} & B_{22} & \cdots & B_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1} & B_{m2} & \cdots & B_{mn} \end{Bmatrix}$$

$$= \begin{Bmatrix} A_{11}+B_{11} & A_{12}+B_{12} & \cdots & A_{1n}+B_{1n} \\ A_{21}+B_{21} & A_{22}+B_{22} & \cdots & A_{2n}+B_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1}+B_{m1} & A_{m2}+B_{m2} & \cdots & A_{mn}+B_{mn} \end{Bmatrix}$$

Cài đặt trong pascal:

```
for i:=1 to M do
for j:=1 to N do
      C[i,j]:= A[i,j] + B[i,j];
```

Ví dụ: $\begin{pmatrix} 5 & -3.5 & 6 \\ 4 & -2 & 3 \\ 1 & 0 & 7 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 6 & -3.5 & 6 \\ 4 & -1 & 3 \\ 1 & 0 & 8 \end{pmatrix}$

+Phép nhân hai ma trận:

Cho hai ma trận A=(A$_{ij}$) € M (m × n,F), B = (B$_{jk}$) € M(n × p,F)

Tích AB của ma trận A và ma trận B là ma trận C =(C$_{ik}$) € M(m × p,F) với các phần tử được xác định như sau:

$C_{ik} = \sum_{j=1}^{n} A_{ij} B_{jk}$, (1≤ i ≤m, 1≤ k ≤p)

Trong Pascal có thể cài đặt như sau:

```
for i:=1 to M do
for k:=1 to P do
```

```
begin
      C[i,k]:=0;
      for j:=1 to N do
            C[i,k]:=C[i,k]+A[i,j] * B[j,k];
end
```

Ví dụ:

$$\begin{pmatrix} 1 & 1 & 3 \\ 2 & 1 & 0 \\ 2 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 2 \\ 0 & 2 \\ 1 & 3 \end{pmatrix} = \begin{pmatrix} 4 & 13 \\ 2 & 6 \\ 3 & 7 \end{pmatrix}$$

Tích của một số a và ma trận A€M(m × n,F) là một ma trận B€M(m × n,F).

Ví dụ: $a\times\begin{Bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{Bmatrix} = \begin{Bmatrix} aA_{11} & aA_{12} & \cdots & aA_{1n} \\ aA_{21} & aA_{22} & \cdots & aA_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ aA_{m1} & aA_{m2} & \cdots & aA_{mn} \end{Bmatrix}$

1.3 Ma trận vuông

Ma trận vuông là ma trận có số hàng và số cột bằng nhau. Ma trận n × n còn gọi là ma trận vuông bậc n. Bất kì hai ma trận vuông cùng bậc đều có thể thực hiện được phép cộng và phép nhân với nhau. Các phần tử A_{ii} tạo thành đường chéo chính của ma trận vuông.

+ Các loại thường gặp:

Nếu mọi phần tử của A ở bên dưới đường chéo chính bằng 0, thì A được gọi là ma trận tam giác trên. Tương tự, nếu mọi phần tử của A nằm phía trên đường chéo chính bằng 0, thì A được gọi là ma trận tam giác dưới. Nếu mọi phần tử nằm bên ngoài đường chéo chính đều bằng không thì A được gọi là ma trận chéo.

+ Ví dụ ma trận tam giác trên: $\begin{pmatrix} 2 & 1 & 5 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix}$

+ Ví dụ ma trận tam giác dưới: $\begin{pmatrix} 2 & 0 & 0 \\ 3 & 1 & 0 \\ 4 & 6 & 1 \end{pmatrix}$

+ Ví dụ ma trận chéo: $\begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{pmatrix}$

+ Ma trận đơn vị I_n:

Có số chiều là n là một ma trận n × n trong đó mọi phần tử trên đường chéo chính bằng 1 và tất cả các phần tử khác bằng 0, ví dụ:

$$I_1=(1),\ I_2=\begin{pmatrix}1&0\\0&1\end{pmatrix},\ I_3=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix},\ \ldots,\ I_n=\begin{pmatrix}1&0&\cdots&0\\0&1&\ldots&0\\\vdots&\vdots&\ddots&\vdots\\0&0&\cdots&1\end{pmatrix}$$

Nó là một ma trận vuông bậc *n*, và cũng là trường hợp đặc biệt của ma trận chéo. Nó là ma trận đơn vị bởi vì khi thực hiện nhân một ma trận với nó thì vẫn thu được ma trận đó:

$\mathbf{AI}_n = \mathbf{I}_m\mathbf{A} = \mathbf{A}$ với ma trận **A** bất kỳ *mxn*.

1.4 Ma trận đối xứng

+ Khái niệm về ma trận chuyển vị:

Nếu A là một ma trận có kích thước m×n với các giá trị a_{ij} tại hàng i và cột j thì ma trận chuyển vị $B = A^T$ là ma trận có kích thước n × m với các giá trị: $b_{ji}= a_{ij}$.

$$\text{Ví dụ: } A=\begin{pmatrix}1&2\\3&4\\5&6\end{pmatrix} \text{ thì } B=A^T=\begin{pmatrix}1&3&5\\2&4&6\end{pmatrix}$$

+ Ma trận đối xứng:

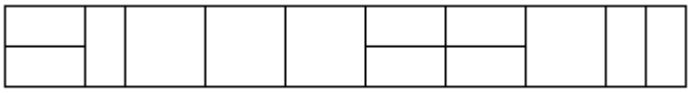
Ma trận vuông A bằng ma trận chuyển vị của nó, tức là $A=A^T$, thì A là ma trận đối xứng.

$$\text{Ví dụ: } A=\begin{pmatrix}1&2&1&4\\2&3&5&2\\1&5&4&1\\4&2&1&2\end{pmatrix}=A^T$$

II. Bài tập ứng dụng

Bài 1. **LÁT VIỀN**

Đường viền trang trí ở nền nhà có kích thước $2 \times N$ được lát bằng 2 loại gạch: loại kích thước 1×2 và loại 2×2. Hãy xác định số cách lát khác nhau có thể thực hiện.



Dữ liệu: Vào từ file văn bản TILE.INP.

- Dòng đầu chứa số nguyên dương T ($T \leq 10$) – số lượng bộ dữ liệu
- T dòng tiếp, dòng thứ i ghi một số nguyên dương N là kích thước nền nhà trong bộ dữ liệu thứ i.

Kết quả: Đưa ra file văn bản TILE.OUT T dòng tương ứng kết quả tìm được với từng bộ dữ liệu theo thứ tự. Đưa ra theo số dư cho $10^9 + 7$

Ví dụ:

TILE.INP	TILE.OUT
3 2 8 12	3 171 2731

Hướng dẫn: Gọi F_i là số cách tạo đường viền kích thước $2 \times i$. Dễ thấy:

$$F_i = F_{i-1} + 2F_{i-2}$$

Khi đó ta có:

$$\begin{matrix} F_n \\ F_{n-1} \end{matrix} = \begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix} \times \begin{matrix} F_{n-1} \\ F_{n-2} \end{matrix}$$

$$\begin{matrix} F_n \\ F_{n-1} \end{matrix} = T^{n-1} \times \begin{matrix} F_1 \\ F_0 \end{matrix} \text{ với } T = \begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix}.$$

Do đó, để tính F_n ta có thể dùng phương pháp nhân ma trận giải quyết bài toán trong thời gian $O(\log_2 n \times sizeof(T)^3)$

Code mẫu:

```
using namespace std;

typedef long long ll;

struct matrix
{
    int val[maxn][maxn];
    matrix()
    {
        memset(val, 0, sizeof(val));
    }
```

```
    matrix operator * ( const matrix & x)
    {
        matrix res;
        for (int u = 0; u < 2; u++)
            for (int v = 0; v < 2; v++)
                for (int i = 0; i < 2; i++)
                    res.val[u][v] = ((ll)val[u][i] *
x.val[i][v] + res.val[u][v] ) % MOD;
        return res;
    }


};

matrix A, res;
long long n;
long long F[10000001];

matrix POWW(matrix A, long long b)
{
    matrix C;
    for (int i = 0; i < 2; i++) C.val[i][i] = 1;
    for ( ; b; b /= 2, A = A * A)
        if (b % 2) C = C * A;
    return C;
}

int main()
{
     ios_base::sync_with_stdio(0);
    freopen(Task".inp", "r", stdin);
    freopen(Task".out", "w", stdout);
    int T;
    cin >> T;
    while (T--){
        cin >> n;
        A.val[0][0] = 1;
        A.val[0][1] = 2;
        A.val[1][0] = 1;
        res= POWW(A,n+1);
        cout << res.val[1][0] << endl;
    }
    return 0;
}
```

Bài 2. Trò chơi lò cò *Nguồn: Duyên Hải 2015*

Carnaval Hạ Long 2015 với chủ đề "Hội tụ tinh hoa - Lan tỏa nụ cười", điểm mới của lễ hội là sự song hành giữa biểu diễn nghệ thuật "Nơi tinh hoa hội tụ" và diễu hành đường

phố "Nụ cười Hạ Long" với sự góp mặt của hơn 2000 diễn viên quần chúng. Có rất nhiều chương trình vui chơi được tổ chức, một trong những trò chơi thu hút được nhiều du khách tham gia đó là trò chơi nhảy lò cò, cụ thể: người chơi cần vượt qua một đoạn đường dài n mét, mỗi bước, người chơi có ba cách nhảy với độ dài bước nhảy tương ứng là 1 mét, 2 mét, 3 mét. Một cách đi chuyển đúng là dãy các bước nhảy có tổng đúng bằng n.

Yêu cầu: Cho n và M, gọi K là số cách đi chuyển đúng khác nhau để đi hết đoạn đường n mét, hãy tính phần dư của K chia M.

Dữ liệu: Vào từ file văn bản LOCO.INP: gồm một dòng chứa hai số nguyên dương n, M ($M \leq 2015$);

Kết quả: Đưa ra file văn bản LOCO.OUT một số nguyên là phần dư của K chia M.

Ví dụ:

LOCO.INP	LOCO.OUT
5 100	13

Ghi chú:

- Có 20% số test ứng với 20% số điểm có $n \leq 20$;
- Có 40% số test ứng với 40% số điểm có $n \leq 10^6$;
- Có 40% số test còn lại ứng với 40% số điểm có $n \leq 10^{15}$.

Hướng dẫn giải:

```
using namespace std;

int  M;
long long n;
int F[3] = {1,1,2};

struct matrix
{
    int val[maxn][maxn];
    matrix()
    {
        memset(val, 0, sizeof(val));
    }

    matrix operator * ( const matrix & x)
    {
        matrix res;
        for (int u = 1; u <= 3; u++)
            for (int v = 1; v <= 3; v++)
                for (int i = 1; i <= 3; i++)
                    res.val[u][v] = ((ll)val[u][i] *
x.val[i][v] + res.val[u][v] ) % M;
```

```
        return res;
    }
};

matrix A;

matrix POWW(matrix A, long long b)
{
    matrix C;
    C.val[1][1] = C.val[2][2] = C.val[3][3] = 1;
    for ( ; b; b /= 2, A = A * A)
        if (b % 2) C = C * A;
    return C;
}

int main()
{
     ios_base::sync_with_stdio(0);
    freopen(Task".inp", "r", stdin);
    freopen(Task".out", "w", stdout);
    cin >> n >> M;
    if (n < 3)
    {
        cout << F[n] % M;
        return 0;
    }
    A.val[1][1] = A.val[1][2] = A.val[1][3] = A.val[2][1] =
A.val[3][2] = 1;
    matrix B = POWW(A, n-2);
    cout << (B.val[1][1] * 2 + B.val[1][2] + B.val[1][3]) %
M;
}
```

Bài 3. Tổng FIBOS

Xét dãy số Fibonacci $\{\boldsymbol{F_n}\}$ theo định nghĩa: $\begin{cases} F_0 = F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \ \forall n > 1 \end{cases}$

Cho số $\boldsymbol{n}$, hãy tính tổng $S = F_0 + F_1 + F_2 + \cdots + F_n$ và đưa ra số dư của $\boldsymbol{S}$ chia cho (10^9+7).

Dữ liệu: vào từ file văn bản FIBOS.INP gồm 1 dòng ghi số nguyên dương $\boldsymbol{n}$ ($n \leq 10^{15}$).

Kết quả: ghi ra file văn bản FIBOS.OUT một số nguyên – số dư tìm được.

Ví dụ:

FIBOS.INP	FIBOS.OUT
3	7
5	20

Hướng dẫn giải:

```
using namespace std;
```

```
typedef long long ll;

struct matrix
{
    int val[maxn][maxn];
    matrix()
    {
        memset(val, 0, sizeof(val));
    }

    matrix operator * ( const matrix & x)
    {
        matrix res;
        for (int u = 0; u < 2; u++)
            for (int v = 0; v < 2; v++)
                for (int i = 0; i < 2; i++)
                    res.val[u][v] = ((ll)val[u][i] *
x.val[i][v] + res.val[u][v] ) % MOD;
        return res;
    }


};

matrix A, res;
long long n;
long long F[10000001];

matrix POWW(matrix A, long long b)
{
    matrix C;
    for (int i = 0; i < 2; i++) C.val[i][i] = 1;
    for ( ; b; b /= 2, A = A * A)
        if (b % 2) C = C * A;
    return C;
}

int main()
{
     ios_base::sync_with_stdio(0);
    freopen(Task".inp", "r", stdin);
    freopen(Task".out", "w", stdout);
    int T;
    F[0] = F[1] = 1;
    for (int i = 2; i <= 1000000; i++) F[i] = (F[i-1] + F[i-
2]) % MOD;
    cin >> n;
    //cout << (F[n+2] + 1000000006) % MOD << endl;
    A.val[0][0] = 1;
    A.val[0][1] = 1;
    A.val[1][0] = 1;
    res= POWW(A,n+3);
    cout << ((res.val[1][0] + MOD - 1) % MOD) << endl;
    return 0;
```

```
}
```