NCKU Programming Contest Training Course
2016/03/30

Fast Matrix Exponentiation

Jingfei Yang

http://myweb.ncku.edu.tw/~e84016184/FastMatrixPower.pdf

Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

一般冪運算

- 將一個數自乘 n 次： $A^n = \prod_{i=1}^{n} A, n \geq 0$
- 一般運算：直接相乘

```
int pow(int A, int n) {
    int res = 1;
    for( int i = 0; i < n; ++i ) res *= A;
    return res;
}
```

  - 時間複雜度： $O(n)$
  - 問題：當指數很大的時候 (e.g. 1,000,000,000)
    1. 用 int 存會 overflow
    2. 計算量變很大，運算時間也變很久

快速冪運算

- 將一個數自乘 n 次： $A^n = \prod_{i=1}^{n} A, n \geq 0$
- 二分法運算：利用結合律
  - 舉例：運算 A^8
    - $A^8 = A \times A \times A \times A \times A \times A \times A \times A$
      7 次乘法運算
    - $A^8 = (A \times A \times A \times A) \times (A \times A \times A \times A)$
      $= (A \times A \times A \times A)^2$
      4 次乘法運算
      $= [(A \times A) \times (A \times A)]^2$
      $= [(A \times A)^2]^2$
      3 次乘法運算
  - 時間複雜度： $O(\log n / \log 2)$
    e.g. 指數 1,000,000,000 僅運算 30 次

快速冪運算

- 將一個數自乘 n 次： $A^n = \prod_{i=1}^{n} A, n \geq 0$

  二分法運算：利用結合律

```
int fast(int A, int n) {
    int res = 1;
    while(n) {
        if(n & 1) res *= A;  // if it is odd
        A *= A;
        n >>= 1;             // divided by 2
    }
    return res;
}
```

made by Jingfei

Fibonacci 數列

- 1, 1, 2, 3, 5, …
- $f(n) = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ f(n-1) + f(n-2), & n > 1 \end{cases}$
- 一般解法：Dynamic Programming
  - Top Down
  - Bottom Up
- 線性代數解法 (利用矩陣)：

$$\begin{bmatrix} F_2 \\ F_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_1 \\ F_0 \end{bmatrix} = \begin{bmatrix} F_1 + F_0 \\ F_1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \begin{bmatrix} F_1 \\ F_0 \end{bmatrix}$$

Fibonacci 數列

- 1, 1, 2, 3, 5, …
- $f(n) = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ f(n-1) + f(n-2), & n > 1 \end{cases}$
- 一般解法：Dynamic Programming
  - Top Down
  - Bottom Up
- 線性代數解法 (利用矩陣)：

$$\begin{bmatrix} F_2 \\ F_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_1 \\ F_0 \end{bmatrix} = \begin{bmatrix} F_1 + F_0 \\ F_1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix} = \boxed{\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1}} \begin{bmatrix} F_1 \\ F_0 \end{bmatrix}$$

快速冪

made by Jingfei

k 階遞迴數列

- $x_n = a_0 x_{n-1} + a_1 x_{n-2} + \cdots + a_{k-1} x_{n-k}$
  - E.g. 二階：$f(n) = 2 \cdot f(n-1) + 3 \cdot f(n-2)$

- $x_n = \begin{bmatrix} a_0 & \cdots & a_{k-1} \end{bmatrix} \begin{bmatrix} x_{n-1} \\ \vdots \\ x_{n-k} \end{bmatrix}$

$$\Rightarrow \begin{bmatrix} a_0 & a_1 & \cdots & a_{k-2} & a_{k-1} \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{n-1} \\ x_{n-2} \\ x_{n-3} \\ \vdots \\ x_{n-k} \end{bmatrix} = \begin{bmatrix} x_n \\ x_{n-1} \\ x_{n-2} \\ \vdots \\ x_{n-k+1} \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} x_n \\ x_{n-1} \\ x_{n-2} \\ \vdots \\ x_{n-k+1} \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & \cdots & a_{k-2} & a_{k-1} \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}^{n-k+1} \begin{bmatrix} x_{k-1} \\ x_{k-2} \\ x_{k-3} \\ \vdots \\ x_0 \end{bmatrix}$$

k 階遞迴數列

- $x_n = a_0 x_{n-1} + a_1 x_{n-2} + \cdots + a_{k-1} x_{n-k}$
  - E.g. 二階：$f(n) = 2 \cdot f(n-1) + 3 \cdot f(n-2)$

- $x_n = \begin{bmatrix} a_0 & \cdots & a_{k-1} \end{bmatrix} \begin{bmatrix} x_0 \\ \vdots \\ x_{n-k} \end{bmatrix}$

$$\Rightarrow \begin{bmatrix} a_0 & a_1 & \cdots & a_{k-2} & a_{k-1} \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{n-1} \\ x_{n-2} \\ x_{n-3} \\ \vdots \\ x_{n-k} \end{bmatrix} = \begin{bmatrix} x_n \\ x_{n-1} \\ x_{n-2} \\ \vdots \\ x_{n-k+1} \end{bmatrix}$$

快速冪

$$\Rightarrow \begin{bmatrix} x_n \\ x_{n-1} \\ x_{n-2} \\ \vdots \\ x_{n-k+1} \end{bmatrix} = \boxed{\begin{bmatrix} a_0 & a_1 & \cdots & a_{k-2} & a_{k-1} \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}^{n-k+1}} \begin{bmatrix} x_{k-1} \\ x_{k-2} \\ x_{k-3} \\ \vdots \\ x_0 \end{bmatrix}$$

Example

- 二階：$f(n) = \begin{cases} 2, & n = 0 \\ 1, & n = 1 \\ 2 \cdot f(n-1) + 3 \cdot f(n-2), & n > 1 \end{cases}$
- $\begin{bmatrix} F_2 \\ F_1 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_1 \\ F_0 \end{bmatrix} = \begin{bmatrix} 2F_1 + 3F_0 \\ F_1 \end{bmatrix}$

  $\Rightarrow \begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix}^{n-1} \begin{bmatrix} F_1 \\ F_0 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix}^{n-1} \begin{bmatrix} 1 \\ 2 \end{bmatrix}$
- $E.g: n = 3$
  - $f(2) = 2f(1) + 3f(0) = 8, f(3) = 2f(2) + 3f(1) = 19$
  - $\begin{bmatrix} F_3 \\ F_2 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix}^2 \begin{bmatrix} F_1 \\ F_0 \end{bmatrix} = \begin{bmatrix} 7 & 6 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 19 \\ 8 \end{bmatrix}$

Matrix struct & operation

```
#define MOD 1000000007
typedef long long int LL;

LL add(LL a, LL b){return (a+b)%MOD;};
LL mul(LL a, LL b){return a*b%MOD;};

struct Mat{
    LL x[2][2];
    Mat(LL a=0,LL b=0,LL c=0,LL d=0){
        x[0][0]=a; x[0][1]=b; x[1][0]=c; x[1][1]=d;
    }
    Mat operator *(const Mat &A)const{
        Mat res;
        for(int i=0; i<2; ++i)
            for(int j=0; j<2; ++j)
                for(int k=0; k<2; ++k)
                    res.x[i][j]=add(res.x[i][j],mul(x[i][k], A.x[k][j]));
        return res;
    }
};
```

Fast Matrix Exponentiation

- 與一般 (非矩陣) 快速冪相同，僅變數型態改變

```
Mat fast(Mat A, int n){
    Mat res= Mat(1,0,0,1);
    while(n){
        if(n & 1) res *= A;
        A *= A;
        n>>=1;
    }
    return res;
}
```

main function

- 二階：$f(n) = \begin{cases} 2, & n = 0 \\ 1, & n = 1 \\ 2 \cdot f(n-1) + 3 \cdot f(n-2), & n > 1 \end{cases}$
- $\begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix}^{n-1} \begin{bmatrix} F_1 \\ F_0 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix}^{n-1} \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

```
int main(){
    LL F0,F1,n;
    F0=2; F1=1;
    scanf("%lld",&n);
    Mat ans=fast(Mat(2,3,1,0),n-1);
    LL A = add(mul(ans.x[0][0],F1),mul(ans.x[0][1],F0));
    if(A<0) A+=MOD;
    printf("%lld\n",A);
    return 0;
}
```

Practice

- [Uva 10229 Modular Fibonacci]
- [POJ 3070 Fibonacci]

Reference

- 快速幂运算 - Blueve 湛蓝
  http://blueve.me/archives/660
- Wiki 遞迴關係式
  https://zh.wikipedia.org/wiki/遞迴關係式