原创 Product Oriented Recurrence （矩阵快速幂+欧拉扩展公式）

2019-06-13 21:27:03 _-Y-_-Y-_ 阅读数 79 更多 编辑

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/weixin_44410512/article/details/91897500

题目链接

https://cn.vjudge.net/problem/CodeForces-1182E
题目大意

$$f_x = c^{2x-6} \cdot f_{x-1} \cdot f_{x-2} \cdot f_{x-3} \ \text{for}\ x \ge 4.$$

输入 n, f_1, f_2, f_3, c (4≤n≤1018, 1≤f1, f2, f3, c≤109).
输出 $f_n \% (10^9+7)$.

思路

```
我们可以将fn拆乘 (x个f1相乘)乘以（y个f2相乘）乘以（z个f3相乘）乘以 pow（c,k）

   f3  f2  f1
1   0   0   1
2   0   1   0
3   1   0   0
4   1   1   1
5   2   2   1
6   4   3   2
7   7   6   4

从第四行开始
第n行的f1系数等于 第(n-1)行f1系数+第(n-2)行f1系数+第(n-3)行f1系数
第n行的f2系数等于 第(n-1)行f2系数+第(n-2)行f2系数+第(n-3)行f2系数
第n行的f3系数等于 第(n-1)行f3系数+第(n-2)行f3系数+第(n-3)行f3系数

以f1为例
矩阵可以写成
1 1 1     4 0 0
1 0 0  *  2 0 0
0 1 1     1 0 0

接下来就是求 pow(c,k);

1   1
2   1
3   1
4   pow(c,2)
5   pow(c,4)*pow(c,2)
6   pow(c,6)*pow(c,4)*pow(c,2)*pow(c,2)
7   pow(c,8)*pow(c,4)*pow(c,2)*pow(c,2) * pow(c,4)*pow(c,2) * pow(c,2)

从第四行开始
第n行为 第(n-1)行结果*第(n-2)行结果*第(n-3)行结果*pow(c,2*n-6)

矩阵可以写成
 1  1  1  2 -6       14 0 0 0 0
 1  0  0  0  0        6 0 0 0 0
 0  1  0  0  0   *    2 0 0 0 0
 0  0  0  1  1        7 0 0 0 0
 0  0  0  0  1        1 0 0 0 0
```

AC code

编程语言大PK，你选谁？
关闭

```
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <iostream>
using namespace std;
#define ll long long
const ll MOD = 1e9 + 7;
struct matrix{//矩阵快速幂
    ll m[5][5];
};
ll pow(ll x,ll n,ll mod){//快速幂
    ll res=1;
    while(n>0){
       if(n%2==1){
         res=res*x;
         res=res%mod;
       }
       x=x*x;
       x=x%mod;
       n>>=1;
    }
    return res;
}

matrix matrix_multi(matrix a,matrix b){//矩阵相乘
    matrix tmp;
    for(int i=0;i<5;i++)
    for(int j=0;j<5;j++){
        tmp.m[i][j]=0;
        for(int k=0;k<5;k++)
            tmp.m[i][j]=((tmp.m[i][j])% (MOD-1) + (a.m[i][k]*b.m[k][j]+MOD-1)% (MOD-1)) % (MOD-1);
    }
    return tmp;
}

matrix matrix_pow(matrix a,matrix b,ll n){//矩阵快快速幂
    while(n>0){
        if(n&1) b=matrix_multi(a,b);
        a=matrix_multi(a,a);
        n>>=1;
    }
    return b;
}

ll ff3(ll f3, ll n){
    if(n==4) return pow(f3,1,MOD);
    if(n==5) return pow(f3,2,MOD);
    if(n==6) return pow(f3,4,MOD);
  matrix a,b;
    memset(a.m,0,sizeof a.m);
    memset(b.m,0,sizeof b.m);
    a.m[0][0]=1; a.m[0][1]=1; a.m[0][2]=1;
    a.m[1][0]=1; a.m[1][1]=0; a.m[1][2]=0;
    a.m[2][0]=0; a.m[2][1]=1; a.m[2][2]=0;

    b.m[0][0]=4; b.m[0][1]=0; b.m[0][2]=0;
    b.m[1][0]=2; b.m[1][1]=0; b.m[1][2]=0;
    b.m[2][0]=1; b.m[2][1]=0; b.m[2][2]=0;

    matrix ans=matrix_pow(a,b,n-6);
    return pow(f3,ans.m[0][0],MOD);
}

ll ff2(ll f2,ll n){
    if(n==4) return pow(f2,1,MOD);
    if(n==5) return pow(f2,2,MOD);
    if(n==6) return pow(f2,3,MOD);
    matrix a,b;
    memset(a.m,0,sizeof a.m);
    memset(b.m,0,sizeof b.m);
```

编程语言大PK，你选谁？
关闭

```
    a.m[0][0]=1; a.m[0][1]=1; a.m[0][2]=1;
    a.m[1][0]=1; a.m[1][1]=0; a.m[1][2]=0;
    a.m[2][0]=0; a.m[2][1]=1; a.m[2][2]=0;

    b.m[0][0]=3; b.m[0][1]=0; b.m[0][2]=0;
  b.m[1][0]=2; b.m[1][1]=0; b.m[1][2]=0;
  b.m[2][0]=1; b.m[2][1]=0; b.m[2][2]=0;

    matrix ans=matrix_pow(a,b,n-6);
    return pow(f2,ans.m[0][0],MOD);
}

ll ff1(ll f1,ll n){
    if(n==4) return pow(f1,1,MOD);
    if(n==5) return pow(f1,1,MOD);
    if(n==6) return pow(f1,2,MOD);
    matrix a,b;
    memset(a.m,0,sizeof a.m);
    memset(b.m,0,sizeof b.m);
    a.m[0][0]=1; a.m[0][1]=1; a.m[0][2]=1;
    a.m[1][0]=1; a.m[1][1]=0; a.m[1][2]=0;
    a.m[2][0]=0; a.m[2][1]=1; a.m[2][2]=0;

    b.m[0][0]=2; b.m[0][1]=0; b.m[0][2]=0;
    b.m[1][0]=1; b.m[1][1]=0; b.m[1][2]=0;
    b.m[2][0]=1; b.m[2][1]=0; b.m[2][2]=0;

    matrix ans=matrix_pow(a,b,n-6);
    return pow(f1,ans.m[0][0],MOD);
}

ll cc(ll c,ll n){

  if(n==4) return  pow(c,2,MOD);
  if(n==5) return  pow(c,6,MOD);
  if(n==6) return  pow(c,14,MOD);
  matrix a,b;
    memset(a.m,0,sizeof a.m);
    memset(b.m,0,sizeof b.m);
  b.m[0][0]=14;
  b.m[1][0]=6;
  b.m[2][0]=2;
  b.m[3][0]=7;
  b.m[4][0]=1;

  a.m[0][0]=1; a.m[0][1]=1; a.m[0][2]=1; a.m[0][3]=2; a.m[0][4]=-6;
  a.m[1][0]=1; a.m[1][1]=0; a.m[1][2]=0; a.m[1][3]=0; a.m[1][4]=0;
  a.m[2][0]=0; a.m[2][1]=1; a.m[2][2]=0; a.m[2][3]=0; a.m[2][4]=0;
  a.m[3][0]=0; a.m[3][1]=0; a.m[3][2]=0; a.m[3][3]=1; a.m[3][4]=1;
  a.m[4][0]=0; a.m[4][1]=0; a.m[4][2]=0; a.m[4][3]=0; a.m[4][4]=1;

  matrix ans=matrix_pow(a,b,n-6);
    return pow(c,ans.m[0][0],MOD);
}

void solve(){
    ll n,f1,f2,f3,c;
    scanf("%lld %lld %lld %lld %lld", &n, &f1, &f2, &f3, &c);
    if(n==1){
    printf("%lld\n", f1);
    return ;
  }
  if(n==2){
    printf("%lld\n", f2);
    return ;
  }
  if(n==3){
    printf("%lld\n", f3);
    return ;
  }
    ll anss=1;
```

编程语言大PK，你选谁？
关闭

```
        anss=( anss * ff1(f1,n))%MOD;
        anss=( anss * ff2(f2,n))%MOD;
        anss=( anss * ff3(f3,n))%MOD;
        anss=( anss * cc(c,n))%MOD;
    printf("%lld\n", anss);
      return ;
}

int main(){
    solve();
  return 0;
}

```

为什么要在矩阵相乘里面要MOD-1呢?
因为矩阵算出来的 x, y, z, k 意思是 f_1^x, f_2^y, f_3^z, c^k
又因欧拉定理的推论:

若正整数a，n互质，那么对于任意正整数b，有$a^b \equiv a^{b \bmod \varphi(n)} \pmod n$

所以最终结果是

$$(f_1^{(x\%MOD-1)} \times f_2^{y\%MOD-1} \times f_1^{z\%MOD-1} \times c^{k\%MOD-1})\%MOD$$

有 0 个人打赏

文章最后发布于: 2019-06-13 21:27:03

©2019 CSDN 皮肤主题: 大白 设计师: CSDN官方博客

编程语言大PK，你选谁？
关闭