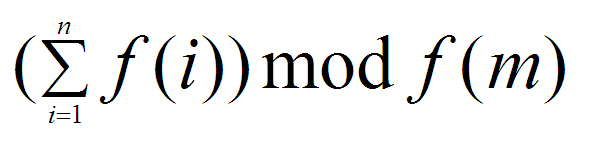
**斐波那契**

问题描述

　　斐波那契数列大家都非常熟悉。它的定义是：  
  
　　f(x) = 1 .... (x=1,2)  
　　f(x) = f(x-1) + f(x-2) .... (x>2)  
  
　　对于给定的整数 n 和 m，我们希望求出：  
　　f(1) + f(2) + ... + f(n) 的值。但这个值可能非常大，所以我们把它对 f(m) 取模。  
　　公式如下  
  
  
　　但这个数字依然很大，所以需要再对 p 求模。

输入格式

　　输入为一行用空格分开的整数 n m p (0 < n, m, p < 10^18)

输出格式

　　输出为1个整数，表示答案

样例输入

2 3 5

样例输出

0

样例输入

15 11 29

样例输出

25

本题的C++参考代码如下：

#include <map>

#include <set>

#include <list>

#include <cmath>

#include <ctime>

#include <deque>

#include <queue>

#include <stack>

#include <bitset>

#include <cctype>

#include <cstdio>

#include <string>

#include <vector>

#include <cstdlib>

#include <cstring>

#include <iomanip>

#include <sstream>

#include <iostream>

#include <algorithm>

using namespace std;

#define PB push\_back

#define MP make\_pair

#define AA first

#define BB second

#define OP begin()

#define ED end()

#define SZ size()

#define SORT(x) sort(x.OP,x.ED)

#define SQ(x) ((x)\*(x))

#define SSP system("pause")

#define cmin(x,y) x=min(x,y)

#define cmax(x,y) x=max(x,y)

typedef long long LL;

typedef pair<int, int> PII;

const double eps=1e-8;

const double INF=1e20;

const double PI=acos( -1. );

const int MXN = 50;

const LL MOD = 1000000007;

LL llmul( LL a,LL b,LL mod ) {

a%=mod;a+=mod;a%=mod;

b%=mod;b+=mod;b%=mod;

if ( a<b )swap( a,b );

LL ret=0;

while ( b ) {

if ( b&1 )ret=( ret+a )%mod;

a=( a<<1 )%mod;

b/=2;

}

return ret;

}

struct matrix {

LL x[3][3];

matrix() {memset( x,0,sizeof x );}

};

matrix mmul( matrix &A,matrix &B,LL mod ) {

matrix ret;

for ( int i=1; i<=2; i++ )

for ( int j=1; j<=2; j++ )

for ( int k=1; k<=2; k++ )

ret.x[i][j]=( ret.x[i][j]+llmul( A.x[i][k],B.x[k][j],mod ) )%mod;

return ret;

}

本题的Java参考代码如下：

import java.math.BigInteger;

import java.util.Scanner;

public class Main{

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

long n,m;

n=sc.nextLong();

m=sc.nextLong();

BigInteger p=sc.nextBigInteger(),fn,fm;

if(n+2>m)

{

fm=think(m,null);

fn=think(n+2,fm).subtract(new BigInteger("1"));

System.out.println(fn.remainder(fm).remainder(p));

}

else

{

fn=think(n+2,p).subtract(new BigInteger("1"));

System.out.println(fn.remainder(p));

}

}

private static BigInteger think(long m,BigInteger mod) {

// TODO Auto-generated method stub

BigInteger a1=new BigInteger("1"),a2=new BigInteger("1"),x[][];

if(m==1)return a1;

else if(m==2)return a2;

else

{

x=new BigInteger[2][2];

x[0][0]=new BigInteger("1");

x[0][1]=new BigInteger("1");

x[1][0]=new BigInteger("1");

x[1][1]=new BigInteger("0");

x=doublex(x,m-2,mod);

return x[0][0].add(x[0][1]);

}

}

private static BigInteger[][] doublex(BigInteger[][] x, long n,BigInteger mod) {

// TODO Auto-generated method stub

BigInteger x2[][];

x2=new BigInteger[2][2];

if(n==1)return x;

else

{

if(n%2==1)return cheng(doublex(cheng(x,x,mod),n/2,mod),x,mod);

else return doublex(cheng(x,x,mod),n/2,mod);

}

}

private static BigInteger[][] cheng(BigInteger[][] x, BigInteger[][] y,BigInteger mod) {

// TODO Auto-generated method stub

BigInteger z[][];

z=new BigInteger[2][2];

if(mod!=null)

{

z[0][0]=x[0][0].multiply(y[0][0]).add(x[1][0].multiply(y[0][1])).remainder(mod);

z[0][1]=x[0][0].multiply(y[0][1]).add(x[0][1].multiply(y[1][1])).remainder(mod);

z[1][0]=x[1][0].multiply(y[0][0]).add(x[1][1].multiply(y[1][0])).remainder(mod);

z[1][1]=x[1][0].multiply(y[0][1]).add(x[1][1].multiply(y[1][1])).remainder(mod);

return z;

}

z[0][0]=x[0][0].multiply(y[0][0]).add(x[1][0].multiply(y[0][1]));

z[0][1]=x[0][0].multiply(y[0][1]).add(x[0][1].multiply(y[1][1]));

z[1][0]=x[1][0].multiply(y[0][0]).add(x[1][1].multiply(y[1][0]));

z[1][1]=x[1][0].multiply(y[0][1]).add(x[1][1].multiply(y[1][1]));

return z;

}

}