**递推求值**

问题描述

　　已知递推公式：  
  
　　F(n, 1)=F(n-1, 2) + 2F(n-3, 1) + 5,  
  
　　F(n, 2)=F(n-1, 1) + 3F(n-3, 1) + 2F(n-3, 2) + 3.  
  
　　初始值为：F(1, 1)=2, F(1, 2)=3, F(2, 1)=1, F(2, 2)=4, F(3, 1)=6, F(3, 2)=5。  
　　输入n，输出F(n, 1)和F(n, 2)，由于答案可能很大，你只需要输出答案除以99999999的余数。

输入格式

　　输入第一行包含一个整数n。

输出格式

　　输出两行，第一行为F(n, 1)除以99999999的余数，第二行为F(n, 2)除以99999999的余数。

样例输入

4

样例输出

14  
  
21

数据规模和约定

　　1<=n<=10^18。

C++代码：

// 递推求值.cpp : 定义控制台应用程序的入口点。

//

//#include "stdafx.h"

#include <iostream>

#include <cstdio>

#include <vector>

using namespace std;

typedef long long ll;

typedef vector<ll> vec;

typedef vector<vec> mat;

const int mod = 99999999;

mat mul(mat & a, mat & b)

{

mat c(a.size(), vec(b[0].size()));

for (int i = 0; i < a.size(); i++)

{

for (int k = 0; k < b.size(); k++)

{

for (int j = 0; j < b[0].size(); j++)

{

c[i][j] = (c[i][j] + a[i][k] \* b[k][j]) % mod;

}

}

}

return c;

}

mat pow(mat a, ll n)

{

mat b(a.size(), vec(a.size()));

for (int i = 0; i < a.size(); i++)

{

b[i][i] = 1;

}

while (n > 0)

{

if (n & 1)

b = mul(b, a);

a = mul(a, a);

n >>= 1;

}

return b;

}

int main()

{

mat F(7, vec(1));

F[0][0] = 6;

F[1][0] = 1;

F[2][0] = 2;

F[3][0] = 5;

F[4][0] = 4;

F[5][0] = 3;

F[6][0] = 1;

ll n;

cin >> n;

if (n <= 3)

{

printf("%d\n", F[3-n][0]);

printf("%d\n", F[5-(n-1)][0]);

//system("pause");

return 0;

}

mat x(7, vec(7));

for (int i = 0; i < 7; i++)

{

for (int j = 0; j < 7; j++)

{

x[i][j] = 0;

}

}

x[0][2] = 2;

x[0][3] = 1;

x[0][6] = 5;

x[1][0] = 1;

x[2][1] = 1;

x[3][0] = 1;

x[3][2] = 3;

x[3][5] = 2;

x[3][6] = 3;

x[4][3] = 1;

x[5][4] = 1;

x[6][6] = 1;

mat res(7, vec(7));

res = pow(x, n - 3);

mat fuck(7, vec(1));

fuck = mul(res, F);

/\*

for (int i = 0; i < 7; i++)

{

printf("%d\n", fuck[i][0]);

}

\*/

printf("%d\n", fuck[0][0]);

printf("%d\n", fuck[3][0]);

//system("pause");

return 0;

}

JAVA代码：

import java.util.Scanner;

public class Main {

final static int mod = 99999999;

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

long n = sc.nextLong();

if (n == 1) {

System.out.println(2);

System.out.println(3);

}else if(n == 2){

System.out.println(1);

System.out.println(4);

}else if(n == 3){

System.out.println(6);

System.out.println(5);

}else{

long[][]a = {{0,1,0,0,2,0,5},

{1,0,0,0,3,2,3},

{1,0,0,0,0,0,0},

{0,1,0,0,0,0,0},

{0,0,1,0,0,0,0},

{0,0,0,1,0,0,0},

{0,0,0,0,0,0,1}};

long[][]b = {{6},{5},{1},{4},{2},{3},{1}};

long[][]x = new long[5][1];

x = Multiply\_Matrix(Multiply\_ksm(a, n-3), b);

long result1 = x[0][0]%mod;

long result2 = x[1][0]%mod;

System.out.println(result1);

System.out.println(result2);

}

sc.close();

}

public static long[][] Multiply\_Matrix(long[][] a, long[][] b) {

long[][] c = new long[a.length][b[0].length];

for (int i = 0; i < a.length; i++) {

for (int j = 0; j < b[0].length; j++) {

long temp = 0;

for (int k = 0; k < b.length; k++) {

temp = (temp + ((a[i][k] % mod) \* (b[k][j] % mod)) % mod) % mod;

}

c[i][j] = temp;

}

}

return c;

}

public static long[][] Multiply\_ksm(long[][] a, long k) {

long[][] d = new long[a.length][a[0].length];

if (k == 1) {

return a;

} else if (k == 2) {

return Multiply\_Matrix(a, a);

} else if (k % 2 == 0) {

d = Multiply\_ksm(Multiply\_Matrix(a, a), k / 2);

return d;

} else {

d = Multiply\_ksm(Multiply\_Matrix(a, a), k / 2);

return Multiply\_Matrix(d, a);

}

}

}