**算法训练 矩阵乘方**

时间限制：1.0s   内存限制：512.0MB

问题描述

　　给定一个矩阵A,一个非负整数b和一个正整数m，求A的b次方除m的余数。  
　　其中一个nxn的矩阵除m的余数得到的仍是一个nxn的矩阵，这个矩阵的每一个元素是原矩阵对应位置上的数除m的余数。  
　　要计算这个问题，可以将A连乘b次，每次都对m求余，但这种方法特别慢，当b较大时无法使用。下面给出一种较快的算法(用A^b表示A的b次方)：  
　　若b=0，则A^b%m=I%m。其中I表示单位矩阵。  
　　若b为偶数，则A^b%m=(A^(b/2)%m)^2%m，即先把A乘b/2次方对m求余，然后再平方后对m求余。  
　　若b为奇数，则A^b%m=(A^(b-1)%m)\*a%m，即先求A乘b-1次方对m求余，然后再乘A后对m求余。  
　　这种方法速度较快，请使用这种方法计算A^b%m，其中A是一个2x2的矩阵，m不大于10000。

输入格式

　　输入第一行包含两个整数b, m，第二行和第三行每行两个整数，为矩阵A。

输出格式

　　输出两行，每行两个整数，表示A^b%m的值。

样例输入

2 2  
1 1  
0 1

样例输出

1 0  
0 1

本题的C++参考代码如下：

#include <iostream>

using namespace std;

// 坐标或向量的结构

struct pointer {

int x;

int y;

};

// 将A坐标按B向量平移，结果保存在C坐标中。

void move(pointer \*C, pointer \*A, pointer \*B)

{

C->x=A->x+B->x;

C->y=A->y+B->y;

}

int main()

{

pointer A, B, C;

cin >> A.x >> A.y;

cin >> B.x >> B.y;

move(&C, &A, &B);

cout << C.x << " " << C.y;

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int b, m;

int a[2][2], ans[2][2], temp[2][2] = {1,1,1,1};

void play()

{

int cnt, cnt2;

for(cnt = 0; cnt < 2; ++cnt)

{

for(cnt2 = 0; cnt2 < 2; ++cnt2)

{

printf("%d ", ans[cnt][cnt2]);

}

printf("\n");

}

}

void cp(int arr1[][2], int arr2[][2])

{

int cnt, cnt2;

for(cnt = 0; cnt < 2 ;++cnt)

for(cnt2 = 0; cnt2 < 2; ++cnt2)

arr1[cnt][cnt2] = arr2[cnt][cnt2];

}

void mod(int arr[][2])

{

int cnt, cnt2;

for(cnt = 0; cnt < 2; ++cnt)

for(cnt2 = 0; cnt2 < 2; ++cnt2)

arr[cnt][cnt2] %= m;

}

void fun2(int a[][2], int b[][2])

{

int cnt, cnt2;

temp[0][0] = a[0][0]\*b[0][0]+a[0][1]\*b[1][0];

temp[0][1] = a[0][0]\*b[0][1]+a[0][1]\*b[1][1];

temp[1][0] = a[1][0]\*b[0][0]+a[1][1]\*b[1][0];

temp[1][1] = a[1][0]\*b[0][1]+a[1][1]\*b[1][1];

}

void fun(int arr[][2], int k)

{

int cnt;

if(k == 0)

{

mod(temp);

cp(ans, temp);

return;

}

if(k == 1)

{

mod(ans);

return;

}

if(k == 2)

{

fun2(a, a);

cp(ans, temp);

// printf("2\n");

// play();

mod(ans);

return;

}

if(k%2 == 0)

{

fun(arr, k/2);

fun2(ans, ans);

cp(ans, temp);

//printf("=0\n");

//play();

mod(ans);

return;

}

if(k%2 != 0)

{

fun(arr, k-1);

fun2(ans, arr);

cp(ans, temp);

//printf("!=0\n");

// play();

mod(ans);

return;

}

}

int main()

{

int cnt, cnt2;

scanf("%d%d", &b, &m);

for(cnt = 0; cnt < 2; ++cnt)

for(cnt2 = 0; cnt2 < 2; ++cnt2)

{

scanf("%d", &a[cnt][cnt2]);

ans[cnt][cnt2] = a[cnt][cnt2];

}

fun(a, b);

play();

return 0;

}

本题的Java参考代码如下：

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main{

static int m;

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String[] str = br.readLine().split(" ");

int b = Integer.parseInt(str[0]);

m = Integer.parseInt(str[1]);

int[][] arr = new int[2][2];

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

String[] tag = br.readLine().split(" ");

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

arr[i][j] = Integer.parseInt(tag[j]);

}

}

int[][] tak = new int[2][2];

tak = mulpow(arr, b);

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 2; j++) {

System.out.print(tak[i][j] % m + " ");

}

System.out.println();

}

}

public static int[][] mul(int[][] x, int[][] y) {

int[][] temp = new int[2][2];

for (int a = 0; a < 2; a++) {

for (int b = 0; b < 2; b++) {

temp[a][b] = 0;

for (int c = 0; c < 2; c++) {

temp[a][b] += x[a][c] \* y[c][b];

temp[a][b] %= m;

}

}

}

return temp;

}

public static int[][] mulpow(int[][] arr, int b) {

if (b == 0) {

int[][] temp = new int[2][2];

for (int x = 0; x < 2; x++) {

for (int y = 0; y < 2; y++) {

if (x == y)

temp[x][y] = 1;

}

}

return temp;

}

if (b == 1)

return arr;

int[][] rep = mulpow(arr, b / 2);

if (b % 2 == 0)

return mul(rep, rep);

else

return mul(mul(rep, rep), arr);

}

}