**波动数列**

问题描述

　　观察这个数列：  
　　1 3 0 2 -1 1 -2 ...  
  
　　这个数列中后一项总是比前一项增加2或者减少3。  
  
　　栋栋对这种数列很好奇，他想知道长度为 n 和为 s 而且后一项总是比前一项增加a或者减少b的整数数列可能有多少种呢？

输入格式

　　输入的第一行包含四个整数 n s a b，含义如前面说述。

输出格式

　　输出一行，包含一个整数，表示满足条件的方案数。由于这个数很大，请输出方案数除以100000007的余数。

样例输入

4 10 2 3

样例输出

2

样例说明

　　这两个数列分别是2 4 1 3和7 4 1 -2。

数据规模和约定

　　对于10%的数据，1<=n<=5，0<=s<=5，1<=a,b<=5；  
　　对于30%的数据，1<=n<=30，0<=s<=30，1<=a,b<=30；  
　　对于50%的数据，1<=n<=50，0<=s<=50，1<=a,b<=50；  
　　对于70%的数据，1<=n<=100，0<=s<=500，1<=a, b<=50；  
　　对于100%的数据，1<=n<=1000，-1,000,000,000<=s<=1,000,000,000，1<=a, b<=1,000,000。

本题的C++参考代码如下：

#include<stdio.h>

#include<iostream>

using namespace std;

const int mod=100000007;

int a,b,n,s,dp[1005][1005],i,j,jt;

void cut(int &a)

{

while(a>=mod){

a-=mod;

}

return;

}

int main(){

scanf("%d%d%d%d",&n,&s,&a,&b);

b%=n;

b\*=-1;

while(b<0){

b+=n;

}

a%=n;

s%=n;

while(s<0){

s+=n;

}

for(i=0;i<n;++i){

for(j=0;j<n;++j){

dp[i][j]=0;

}

}

dp[1][a]=dp[1][b]=1;

for(i=1;i<n-1;++i){

for(j=0;j<n;++j){

jt=(j+a\*(i+1))%n;

dp[i+1][jt]+=dp[i][j];

cut(dp[i+1][jt]);

jt=(j+b\*(i+1))%n;

if(jt>=n){

jt-=n;

}

dp[i+1][jt]+=dp[i][j];

cut(dp[i+1][jt]);

}

}

printf("%d\n",dp[n-1][s]);

return 0;

}

本题的C参考代码如下：

#define mod 100000007

int n,s,a,b,x[1001][1001],i,j,t;

fun()

{

while(x[i+1][t]>=mod)x[i+1][t]-=mod;

}

int main()

{

scanf("%d%d%d%d",&n,&s,&a,&b);

b%=n;

b\*=-1;

while(b<0)b+=n;

a%=n;

s%=n;

while(s<0)s+=n;

for(i=0;i<n;i++)

for(j=0;j<n;j++)

x[i][j]=0;

x[1][a]=x[1][b]=1;

for(i=1;i<n-1;i++)

for(j=0;j<n;j++)

{

t=(j+a\*(i+1))%n;

x[i+1][t]+=x[i][j];

fun();

t=(j+b\*(i+1))%n;

if(t>=n)

t-=n;

x[i+1][t]+=x[i][j];

fun();

}

printf("%d\n",x[n-1][s]);

return 0;

}

本题的Java参考代码如下：

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

int mod = 100000007;

int n, s, a, b, i, j, t;

int x[][] = new int[1001][1001];

Scanner sc = new Scanner(System.in);

n = sc.nextInt();

s = sc.nextInt();

a = sc.nextInt();

b = sc.nextInt();

b %= n;

b \*= -1;

while (b < 0)

b += n;

a %= n;

s %= n;

while (s < 0)

s += n;

for (i = 0; i < n; i++)

for (j = 0; j < n; j++)

x[i][j] = 0;

x[1][a] = x[1][b] = 1;

for (i = 1; i < n - 1; i++)

for (j = 0; j < n; j++) {

t = (j + a \* (i + 1)) % n;

x[i + 1][t] += x[i][j];

x[i + 1][t] %= mod;

t = (j + b \* (i + 1)) % n;

t %= n;

x[i + 1][t] += x[i][j];

x[i + 1][t] %= mod;

}

System.out.printf("%d\n", x[n - 1][s]);

}

}