**算法训练 K好数**

时间限制：1.0s   内存限制：256.0MB

问题描述

如果一个自然数N的K进制表示中任意的相邻的两位都不是相邻的数字，那么我们就说这个数是K好数。求L位K进制数中K好数的数目。例如K = 4，L = 2的时候，所有K好数为11、13、20、22、30、31、33 共7个。由于这个数目很大，请你输出它对1000000007取模后的值。

输入格式

输入包含两个正整数，K和L。

输出格式

输出一个整数，表示答案对1000000007取模后的值。

样例输入

4 2

样例输出

7

数据规模与约定

对于30%的数据，KL <= 106；

对于50%的数据，K <= 16， L <= 10；

对于100%的数据，1 <= K,L <= 100。

锦囊1

使用动态规划。

锦囊2

用F[i][j]表示长为i，最后一位数字是j的K好数的个数，则F[i][j]=\sum F[i-1][k]，其中|j-k|!=1。

本题的C++参考代码如下：

#include <iostream>

using namespace std;

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

int main(int argc, char\*\* argv) {

int k,l;

cin>>k>>l;

long long table[100][100];

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

{

table[0][i]=1ll;

}

table[0][0]=0ll;

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

{

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

{

long long x=0;

for(int y=0;y<k;y++)

{

if(y!=j+1 && y!=j-1)

{

x+=table[i-1][y];

}

}

table[i][j]=x%1000000007ll;

}

}

long long count=0;

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

{

count+=table[l-1][i];

}

cout<<(count%1000000007ll);

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

int main()

{

int i;

int k; //进制数

int l; //位数

long long ka[100]; //前

long long kb[100]; //当前

long long cont=0; //计数

scanf("%d%d",&k,&l);

kb[0]=ka[0]=0;

for(i=1;i<k;i++)

{

kb[i]=ka[i]=1;

}

for(i=2;i<=l;i++)

{

int j;

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

{

int m=0;

for(m=0;m<k;m++)

{

if(m<j-1 || m>j+1)

kb[j]+=ka[m];

}

}

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

{

ka[j]=kb[j];

ka[j]=kb[j]%1000000007;

}

}

while(k--)

{

cont+=ka[k];

cont=cont%1000000007;

}

printf("%I64d\n",cont);

return 0;

}

本题的Java参考代码如下：

import java.io.\*;

public class Main {

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

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

String s[] = bfr.readLine().split(" +");

int K = Integer.valueOf(s[0]);

int L = Integer.valueOf(s[1]);

int f[][] = new int[L][K];

int i,j,k,sum=0;

for(j=0;j<K;j++) f[0][j] = 1;

f[0][0]=0;

if(L>1)

{

for(i=1;i<L;i++)

{

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

{

for(k=0;k<K;k++)

if(k!=j-1 && k!=j+1)

{

f[i][j]+=f[i-1][k];

f[i][j]%=1000000007;

}

}

}

}

for(j=0;j<K;j++) { sum+=f[L-1][j]; sum%=1000000007; }

System.out.println(sum);

}

}