BJFUOJ 1238解题报告——fudq的迷宫

欢迎AC：<http://acm.bjfu.edu.cn/acmhome/problemdetail.do?&method=showdetail&id=1238>

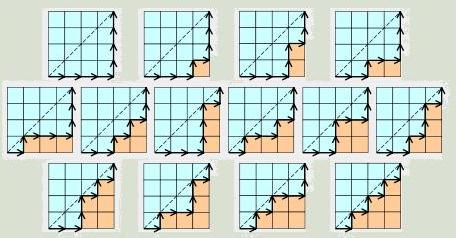
一、题目再现：

**fudq的迷宫**

时间限制(普通/Java):2000MS/3000MS 运行内存限制:65536KByte  
总提交:96 测试通过:13

**描述**

fudq发明了一个n\*n的迷宫，起点在迷宫的左下角格点上，宝物在迷宫的右上角格点上。从起点开始寻找宝物，fudq规定每一步只能向上或向右走，但是不能越过对角线。  
很快fudq就发现这样的走法有好多种，多得数不清楚了，你能告诉fudq一共有多少种么？  
如下图为n=4的情况，一共有14种走法。



**输入**

输入有多组测试数据，每组数据输入两个正整数n和m，中间用一个空格隔开。2<=n<=500000,1<=m<=45000。

**输出**

对应每组数据输出结果D，因为D可能会很大，只需输出D%m的结果即可，每组输出占一行。

**样例输入**

2 10  
4 10

**样例输出**

2  
4

**组合数论：Catalan数**

二、递推公式与通项公式：

卡特兰数Catalan number，是组合数学中一个常出现在各种计数问题中出现的数列。

其前几项为 : 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452, ...

h(0)=1,h(1)=1，

（1）定义式：如果一个序列满足，这样的序列成为卡特兰数Catalan数。这是卡特兰数的定义式，一般情况下的结论都是基于此定义得出。

（2）递推公式： ()

（3）通项公式：

（4）通项公式的变式：

三、卡特兰数的应用：

（1）卡特兰数的渐进增长为

（2）定义dyck word为一个有n个X，n个Y组成的字串，满足所有的**前缀字串**皆满足X的个数大于等于Y的个数。

前缀字串是指原字串中从某一字符开头满足前缀条件的子串，这里的前缀条件就是以X开始。

如：以长度为6的dyck word串为例：

有以下几种可能：

XXXYYY、XXYYXY、XYXYXY、XYXXYY、XXYXYY

那么结论：长度为2n的dyck word串种数为n阶卡特兰数。

（3）括号匹配算法中，所以括号匹配情形种数：满足’(’的数量大于等于’)’的数量。

类比（2）的结论可以得出下面的结论：

括号匹配算法中，满足括号匹配规则的情况种数是一个n阶卡特兰数。

（4）入栈出栈种数：将n个数据入栈再出栈，可以得到的出栈序列总数是一个卡特兰数。

（5）电影院找零问题：有2n个人来电影院看电影，电影票价5元，其中有n个人只有5元钞票，另外n个人只有10元钞票，那么为了保证售票处总能给持有10元的钞票的客人找零，这2n个客人的排列方式有几种：

结论：是一个n阶的卡特兰数。

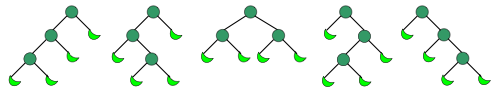
证明：可以把持有5元的客人买票视作5元进栈，持有10元的客人视作5元出栈，这样就转化为（4）的问题。

（6）含有n个节点的二叉树种类：是一个卡特兰数。

证明：设含有n个节点的二叉树种类为Cn，那么除了根节点外，还有n-1个节点，假如说其中有k个节点构成了这棵树的左子树，则必有n-k-1个节点构成了这棵树的右子树。所以：

这正是卡特兰数的定义。

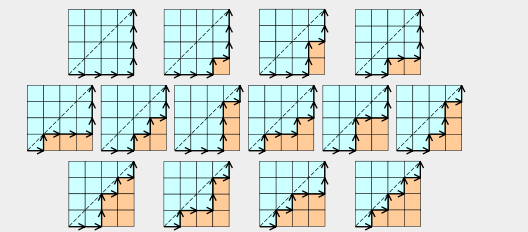
如：n=3的二叉树种数：（原点表示节点，月牙表示空节点）



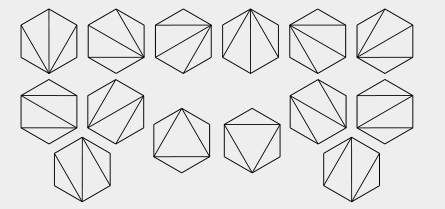
（7）定义：一个单调路径为从一个n\*n的格点左下角出发，到右上角结束，满足①每一步只能向上或向右走 ②不能越过主对角线。

对于一个n\*n的格点中，单调路径种数为一个n阶卡特兰数。

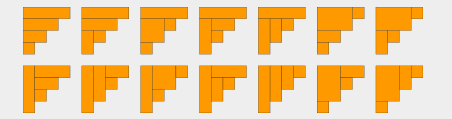
如：n=4时：单调路径共有以下14种。



（8）通过连接顶点而将n+2边的凸多边形分成三角形的方法种数为一个n阶卡特兰数。



（9）用n个长方形填充一个高度为n的阶梯状图形的方法总数为一个卡特兰数。



四、题目分析：

分析题目可知，需要求n阶卡特兰数对m取余的值。2<=n<=500000，1<=m<=45000。

五、解题策略：

首先，根据基础知识可知，答案是一个卡特兰数。所以问题就转化成了如何求一个卡特兰数。

卡特兰数有三种计算方式：

（1）利用递推公式Cn=(4n-2)/(n+1)\*Cn-1循环求解：

这样，已知C0=1，那么即可以得出任意Ck。

伪程序如下：

#define MAXN 500000

int c[MAXN+10];

c[0]=1;

for(int i=1;i<=500000;i++)

c[i]=(4n-2)\*c[i-1]\*(n+1);

这样，理论上就能够得出任意一项卡特兰数。

解题策略分析：根据卡特兰序列的前几项：1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452, ...

我们可以看出卡特兰数的增长率还是非常大的。

如果对m取余：c[i]=(((4n-2)\*c[i-1])%m)/n+1，这样，当4n-2乘以c[i-1]的时候就有可能产生溢出。

（2）利用 进行计算。

这样需要用质因数分解法求出，然后再除以n+1。但是我们可以想象这个确实是很难解决的。

（3）利用 求解。

这样做需要分别将(2n)！，(n+1)！，n！进行质因数分解，分解成质数的指数乘积形式，然后将所有的指数相乘取余即可得出答案。

这里就涉及到三个问题：

①质因数分解分为普通的质因数分解和阶乘的质因数分解，由于阶乘的质因数分解的时间复杂度比普通的质因数分解要小的多，所以我们考虑采用阶乘的质因数分解法。

②将所有的指数相乘的时候用快速幂取余来优化算法。

③质因数分解的时候可以先用素数筛选法进行素数打表，可以提高时间复杂度和空间复杂度。

代码如下：

#include <iostream>

#include <cstdio>

#include <cstring>

using namespace std;

const int N=1000000; //定义常量N为500000的2倍，需要筛选的最大素数。

bool isPrime[N/2]; //素数筛选法的判定数组

int prime[N/12]; //素数筛选法的素数表

int zhi[N/12]; //质因数分解的对应指数数组

inline int min(int a,int b) {return a<b?a:b;} //最小值

inline int max(int a,int b) {return a>b?a:b;} //最大值

//快速素数筛选法主算法

int Peratosthenes()

{

int i,j,tot=1;

memset(isPrime,true,sizeof(isPrime));

prime[0]=2;

for(i=3;i\*i<=N;i+=2) if(isPrime[(i-1)/2])

{

prime[tot++]=i;

for(j=i\*i;j<=N;j+=2\*i) isPrime[(j-1)/2]=false;

}

for(;i<=N;i+=2) if(isPrime[(i-1)/2])

prime[tot++]=i;

return tot;

}

//阶乘的质因数分主算法，当cmd为1时代表分子，cmd为0时代表分母

int division(int n,int cmd)

{

int i=0,b;

if(cmd==0) cmd=-1;

for(i=0;prime[i]<=n;i++)

{

b=n;

while(b)

{

zhi[i]+=b\*cmd/prime[i];

b/=prime[i];

}

}

return i;

}

//快速幂取余主算法

int pow\_mod(int a,int b,int m)

{

int ans=1;

a=a%m;

while(b)

{

if(b&1) ans=(ans\*a)%m;

a=(a\*a)%m;

b>>=1;

}

return ans;

}

//主函数中len进行了压缩，是所能达到的最大素数

int main()

{

int n,m,i,ans,j,len,pos,t;

len=Peratosthenes();

while(~scanf("%d%d",&n,&m))

{

pos=0;

memset(zhi,0,sizeof(zhi));

ans=1;

pos=max(pos,division(2\*n,1));

pos=max(pos,division(n+1,0));

pos=max(pos,division(n,0));

t=len;

len=min(pos,len);

for(i=0;i<len;i++) if(zhi[i])

{

int yu=pow\_mod(prime[i],zhi[i],m);

ans=(ans\*yu)%m;

}

len=t;

printf("%d\n",ans);

}

return 0;

}