

## 题目链接

### 题目大意

这题给你n个1和m个0，组成一个字符串，并且对任意一个前缀，1都不能比0少。

### 分析



这是一道比较典型的卡特兰数题。我们可以将问题看作是求解(n+m,n-m)有多少种01字符串组合方式,n+m是字符总数，n-m是1比0多的个数。

从(0,0)开始，记将1放进集合记作操作1，将0放进集合记作操作2。

- 进行操作1，集合变成(x+1,y+1),及向右上角走。
- 进行操作2，集合变成(x+1,y-1)，即往右下角走

最后目标是走到(n+m,n-m)。而如果y碰到y=-1这条线，那么就是非法的。

于是问题就变成了从(0,0)状态走到(n+m,n-m)状态一共有多少种合法的走法。(走：即向空字符串从头到尾填充0和1，不同的走法就实现了0和1的不同组合)

- 如果考虑所有状况，则很明显是 $C(n+m,n)$  (1要走n步，从n+m步中选出n步的种类，即长度为n+m的字符串中有n个为1有多少种)
- 所有合法种类我们不好求，但是非法可求，则用合法的减去非法的种类。
  - 非法的情况即碰过了y=-1的情况如图。 
  - 将与y=-1的最后一个交点之前的部分以y=-1为轴对折，可以发现这种情况对应了一种从(0,-2)到(n+m,n-m)的情况。实际在我们这种对折规矩下，所有从(0,0)到(n+m,n-m)的非法情况都和由从(0,-2)到(n+m,n-m)的所有情况一一对应，则所有非法情况种类可以通过求(0,-2)到(n+m,n-m)的所有情况求出。(合法情况是不会对应从(0,-2)到(n+m,n-m)的，因为合法情况没有与y=-1的交点，所以无法与我们规定的这种翻折对应。) 
  - 非法情况是从(0,-2)到(n+m,n-m)，那么路径就要变。但是总的操作数不能变，因为横坐标还是0到n+m;
    - 设操作1变为a种，操作2变为b种则 
$$\begin{cases} a+b=n+m \\ a-b=n-m+2 \end{cases}$$
 解得 
$$\begin{cases} a=n+1 \\ b=m-1 \end{cases}$$
 则非法情况总数为 $C(n+m,n+1)$

最后ans= $C(n+m,n)-C(n+m,n+1)$

### 组合数求解

$C(a \bmod p) \bmod p = \frac{n!}{m!(n-m)!} \bmod p$  可以看出，如果直接求解组合数然后再取模，那么分子或者分母会溢出。而取模分配率对加减乘成立，如对乘法有

$$(a * b) \% p = [(a \% p) * (b \% p)] \% p$$

但是对除法却不存在

$$(a / b) \% p = [(a \% p) / (b \% p)] \% p \text{ (错误)}$$

那怎么办呢，于是科学家们扔出了逆元这个玩意。#### 逆元 定理：若在mod p意义下，对于一个整数a，有  $a * x \equiv 1 \bmod p$  那么这个整数x即为a的乘法逆元，同时a也为x的乘法逆元。

充要条件：a存在模p的乘法逆元的充要条件是 $\gcd(a,p) = 1$ ，即a与p互质。

应用：求 $\frac{a}{b} \% p$ 等同于求 $a * (b \text{的逆元}) \% p$

- 证明：

我们假设  $\frac{a}{b} \% p = m$  (a和b满足 $a \% b = 0$ ) 由乘法逆元符合分配律对(1)式两边乘以b有  $a \% p = (m(b \% p)) \% p$  即  $a \equiv mb \bmod p$  假设b的逆元是x，则(3)式两边乘以一个x有

$a x \equiv m b x \pmod p$  由逆元的定理有  $b^{-1} x \equiv 1 \pmod p$  则  $a^{-1} x \equiv m \pmod p$  证毕

于是我们就可以把除法的求模运算转换为乘法的求模运算，从而可以利用分配律。

类比：其实逆元可以类比于普通乘法中的倒数，即当  $b \cdot x = 1$  时  $\frac{1}{b} = x$ ，但是在求模运算中， $x$  与  $\frac{1}{b}$  只是等同，而并非就是取倒数。

## 费马小定理

那么逆元怎么求解呢，可不是像普通乘法那样直接是倒数。则可以通过定义  $a \cdot x \equiv 1 \pmod p$  来求解逆元。由于在ACM中大多数时候  $p$  是质数，否则容易找到余数的规律，所以用费马小定理即可求解  $a$  的逆元。

定理：假如  $a$  是一个整数， $p$  是一个质数，那么

1. 如果  $a$  是  $p$  的倍数， $a^p \equiv a \pmod p$
2. 如果  $a$  不是  $p$  的倍数， $a^{p-1} \equiv 1 \pmod p$

由乘法逆元的充要条件是  $\gcd(a, p) = 1$ ，所以  $a$  不会是  $p$  的倍数，所以用第二条。将第二条转化以下有  $a \cdot a^{p-2} \equiv 1 \pmod p$  则  $a^{p-2}$  就是  $a$  的逆元，证明自己去百度。

最后利用快速幂求解即可。

## 快速幂

快速幂即求解幂的快速算法。考察  $3^{10}$ ，有  $3^{10} = 9^5 = 981^2 = 96561^1 = (9 \cdot 6561) \cdot 1^0$  设幂为  $p$ ，底数为  $b$ ，则其算法思路为

1. 如果  $p$  是偶数， $b = b \cdot b$ ； $p > 1$
2. 如果  $p$  是奇数，那么  $ans = ans \cdot b$ ； $b = b \cdot b$

这样最后得到的结果即在  $ans$  中

```
#include<iostream>
using namespace std;
const long long mod =20100403;
int n , m;

// 求阶乘
long long fac(long long a){
    long long ans=1;
    for(long long i = a;i>=1;--i){
        ans=(ans*i%mod)%mod;
        ans=ans%mod;
    }
    return ans;
}

// 矩阵快速幂求逆元
long long Fpow(long long b,long long p){
    long long ans=1;
    while(p>0){
        if(p%2==0){
            b=(b*b)%mod;
            p=p/2;
        }else{
            ans=(ans*b)%mod;
            p=p-1;
        }
    }
    return ans;
}
```

```

        ans=(ans*b)%mod;
        b=(b*b)%mod;
        p=p/2;
    }
}
return ans;
}
// 求组合数
long long C(int a,int b){
    long long a1=fac(a); // 求分子
    long long b1=(fac(b)*fac(a-b))%mod; //分母
    //求b的逆元
    b1=Fpow(b1,mod-2);
    return ((a1%mod)*(b1%mod))%mod;
}
int main(){
    freopen("test/E.txt","r",stdin);
    cin>>n>>m;
    long long ans;
    ans=(C(n+m,n)-C(n+m,n+1)+mod)%mod; // 可能会求得附属，所以先加上模再取模，因为计算结果
    是分别取模之后的，所以小的可能本来就比mod小
    cout<<ans<<endl;
}

```