# 组合数取模







组合数取模在**ACM**竞赛中是一个很重要的问题，很多选手因为数据太大而束手无策，今天就来详细讲解它。

组合数取模就是求 $C_n^m\%p$ 的值，当然根据 $n$， $m$ 和 $p$ 的取值范围不同，采取的方法也不一样。

接下来，我们来学习一些常见的取值情况

（**1**） $1 \leq m \leq n \leq 1000$ 和 $1 \leq p \leq 10^9$

这个问题比较简单，组合数的计算可以靠杨辉三角，那么由于 $n$ 和 $m$ 的范围小，直接两层循环即可。

（**2**） $1 \leq m \leq n \leq 10^{18}$ 和 $2 \leq p \leq 10^5$，并且 $p$ 是素数

这个问题有个叫做**Lucas**的定理，定理描述是，如果

$$n = n_k p^k + n_{k-1} p^{k-1} + ... + n_1 p + n_0$$
$$m = m_k p^k + m_{k-1} p^{k-1} + ... + m_1 p + m_0$$

那么得到

$$C_n^m = \prod_{i=0}^{k} C_{n_i}^{m_i} (mod\ p)$$

这样然后分别求，采用逆元计算即可。

题目：http://acm.fzu.edu.cn/problem.php?pid=2020

题意：求 $C_n^m\%p$，其中$1 \le p \le 10^9$，并且 $p$ 是素数。

代码：

[cpp]

```cpp
#include <iostream>
#include <string.h>
#include <stdio.h>

using namespace std;
typedef long long LL;

LL n,m,p;

LL quick_mod(LL a, LL b)
{
    LL ans = 1;
    a %= p;
    while(b)
    {
        if(b & 1)
        {
            ans = ans * a % p;
            b--;
        }
        b >>= 1;
        a = a * a % p;
    }
    return ans;
}

LL C(LL n, LL m)
{
    if(m > n) return 0;
    LL ans = 1;
    for(int i=1; i<=m; i++)
    {
        LL a = (n + i - m) % p;
        LL b = i % p;
        ans = ans * (a * quick_mod(b, p-2) % p) % p;
    }
    return ans;
}

LL Lucas(LL n, LL m)
{
    if(m == 0) return 1;
    return C(n % p, m % p) * Lucas(n / p, m / p) % p;
}

```

```cpp
int main()
{
    int T;
    scanf("%d", &T);
    while(T--)
    {
        scanf("%I64d%I64d%I64d", &n, &m, &p);
        printf("%I64d\n", Lucas(n,m));
    }
    return 0;
}
```

由于上题的 $p$ 比较大，所以组合数只能一个一个计算，如果 $p$ 的范围小点，那么就可以进行阶乘预处理计算了。

（3） $1 \le m \le n \le 10^6$ 和 $1 \le p \le 10^5$，并且 $p$ 可能为合数

这样的话先采取暴力分解，然后快速幂即可。

题目：**http://acm.nefu.edu.cn/JudgeOnline/problemshow.php?problem_id=628**

代码：

[cpp]

```cpp
#include <iostream>
#include <string.h>
#include <stdio.h>

using namespace std;
typedef long long LL;
const int N = 200005;

bool prime[N];
int p[N];
int cnt;

void isprime()
{
    cnt = 0;
    memset(prime,true,sizeof(prime));
    for(int i=2; i<N; i++)
    {
        if(prime[i])
        {
            p[cnt++] = i;
            for(int j=i+i; j<N; j+=i)
```

```
                prime[j] = false;
        }
    }
}

LL quick_mod(LL a,LL b,LL m)
{
    LL ans = 1;
    a %= m;
    while(b)
    {
        if(b & 1)
        {
            ans = ans * a % m;
            b--;
        }
        b >>= 1;
        a = a * a % m;
    }
    return ans;
}

LL Work(LL n,LL p)
{
    LL ans = 0;
    while(n)
    {
        ans += n / p;
        n /= p;
    }
    return ans;
}

LL Solve(LL n,LL m,LL P)
{
    LL ans = 1;
    for(int i=0; i<cnt && p[i]<=n; i++)
    {
        LL x = Work(n, p[i]);
        LL y = Work(n - m, p[i]);
        LL z = Work(m, p[i]);
        x -= (y + z);
        ans *= quick_mod(p[i],x,P);
        ans %= P;
    }
    return ans;
}

int main()
{
    int T;
    isprime();
    cin>>T;
    while(T--)
    {
```

```
        LL n,m,P;
        cin>>n>>m>>P;
        n += m - 2;
        m--;
        cout<<Solve(n,m,P)<<endl;
    }
    return 0;
}
```

接下来看一些关于组合数取模的典型题目。

题目：**http://acm.hdu.edu.cn/showproblem.PHP?pid=3944**

分析：组合数取模的典型题目，用**Lucas**定理，注意要阶乘预处理，否则会**TLE**的。

题目：**http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemId=4536**

题意：给一个集合，一共 $n$ 个元素，从中选取 $m$ 个元素，选出的元素中没有相邻的元素的选法一共有多少种？

分析：典型的隔板法，最终答案就是 $C_{n-m+1}^{m} mod\ p$ 。然后用**Lucas**定理处理即可。

题目：**http://acm.hdu.edu.cn/showproblem.php?pid=4373**

题意： $m$ 个**for**循环嵌套，有两种形式，第一类从**1**开始到 $n$ ，第二类从上一层循环当前数开始到 $n$ ，第一层一定

是第一种类型，求总的循环的次数对**364875103**取余的结果。

分析：首先可以看出，每一个第一类循环都是一个新的开始，与前面的状态无关，所以可以把 $m$ 个嵌套分为几个不

同的部分，每一个部分由第一类循环开始，最终结果相乘即可。剩下的就是第二类循环的问题，假设一个 $m$

层循环，最大到 $n$ ，分析一下得到如下结果

（1）只有一层，则循环次数为 $n$

（2）只有前两层，则循环次数为

$$\begin{aligned} num &= n + (n-1) + \dots + 1 \\ &= \frac{n(n+1)}{2} \\ &= C_{n+1}^{2} \end{aligned}$$

（3）只有前三层，则循环次数为

$$\begin{aligned} num &= \frac{n(n+1)}{2} + \frac{(n-1)n}{2} + \dots + 1 \\ &= \frac{1+2+\dots+n}{2} + \frac{1+2^2+3^2+\dots+n^2}{2} \\ &= \frac{n(n+1)}{4} + \frac{n(n+1)(2n+1)}{12} \\ &= \frac{n(n+1)(n+2)}{6} \\ &= C_{n+2}^{3} \end{aligned}$$

由此得到结论：第 $m$ 的循环次数为 $C_{n+m-1}^{m}$

代码：

载:

[cpp]

```cpp
#include <iostream>
#include <string.h>
#include <stdio.h>

using namespace std;
typedef long long LL;

const int N = 25;
const int MOD1 = 97;
const int MOD2 = 3761599;
const int MOD = MOD1 * MOD2;

int m,n,k;
int a[N];
LL fac1[MOD1+10];
LL fac2[MOD2+10];
```

```
LL inv1,inv2;

LL quick_mod(LL a,LL b,LL m)
{
    LL ans = 1;
    a %= m;
    while(b)
    {
        if(b & 1)
        {
            ans = ans * a % m;
            b--;
        }
        b >>= 1;
        a = a * a % m;
    }
    return ans;
}

LL C(LL n,LL m,LL p,LL fac[])
{
    if(n < m) return 0;
    return fac[n] * quick_mod(fac[m] * fac[n-m], p - 2, p) % p;
}

LL Lucas(LL n,LL m,LL p,LL fac[])
{
    if(m == 0) return 1;
    return C(n % p, m % p, p, fac) * Lucas(n / p, m / p, p, fac);
}

void Init()
{
    fac1[0] = fac2[0] = 1;
    for(int i=1; i<MOD1; i++)
        fac1[i] = (fac1[i-1] * i) % MOD1;
    for(int i=1; i<MOD2; i++)
        fac2[i] = (fac2[i-1] * i) % MOD2;
    inv1 = MOD2 * quick_mod(MOD2, MOD1-2, MOD1);
    inv2 = MOD1 * quick_mod(MOD1, MOD2-2, MOD2);
}

int main()
{
    Init();
    int T, tt = 1;
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d%d%d",&n,&m,&k);
        for(int i=0; i<k; i++)
            scanf("%d",&a[i]);
        a[k] = m;
        LL ans = 1;
        for(int i=0; i<k; i++)
```

```cpp
        {
            LL m1 = Lucas(a[i+1] - a[i] + n - 1, a[i+1] - a[i], MOD1, fac1);
            LL m2 = Lucas(a[i+1] - a[i] + n - 1, a[i+1] - a[i], MOD2, fac2);
            LL mm = (m1 * inv1 + m2 * inv2) % MOD;
            ans = ans * mm % MOD;
        }
        printf("Case #%d: ",tt++);
        cout<<ans<<endl;
    }
    return 0;
}
```

题目：**http://acm.hdu.edu.cn/showproblem.php?pid=4349**

题意：$C_n^0, C_n^1, C_n^2, ..., C_n^n$ 中有多少个奇数，其中 $1 \le n \le 10^8$ 。

分析：其实组合数判断奇偶性有一个优美的结论

如果 $(n\&m) == m$ ，那么 $C_n^m$ 为奇数，否则为偶数

当然本题要判断的组合数很多，所以不能用上述结论，只能另辟蹊径。由于是判断奇偶性，那么就是判断 $C_n^m mod\ 2$ 是否为1，利用**Lucas**定理，先把 $n$ 和 $m$ 化为二进制，这样它们都是01序列了。我们又知道 $C_0^1 = 0, C_0^0 = 1, C_1^0 = 1, C_1^1 = 1$ 。这样 $n$ 中为0的地方对应的 $m$ 中的位置只有一种可能，那就是0。

这样我们可以不用管 $n$ 中为0的地方，只考虑 $n$ 中为1的位置，可以看出，$n$ 中为1的位置对应的 $m$ 中为0或1，其结果都是1，这样答案就是：1<<( $n$ 二进制表示中1的个数)

代码：

```cpp
#include <iostream>
#include <string.h>
#include <stdio.h>

using namespace std;

```

```
int main()
{
    int n;
    while(scanf("%d",&n)!=EOF)
    {
        int cnt = 0;
        while (n)
        {
            if (n & 1) cnt++;
            n >>= 1;
        }
        printf("%d\n",1<<cnt);
    }
    return 0;
}
```

题目：**http://61.187.179.132/JudgeOnline/problem.php?id=1951**

题意：给定两个正整数 $G$ 和 $n$，其中 $1 \leq n \leq 10^9$，求下面表达式的值

$$ans = G^{\sum_{i|n} C_n^i} mod\ 999911659$$

分析：由于**999911659**是素数，用费马小定理降幂得到

$$ans = G^{\sum_{i|n} C_n^i mod\ 999911658} mod\ 999911659$$

现在关键是求

$$\sum_{i|n} C_n^i mod\ 999911658$$

那么我们枚举 $i$ 分别计算，但是模的是合数，所以对**999911658**进行分解得到

$999911658 = 2 \times 3 \times 4679 \times 35617$，那么分别求 $A_1, A_2, A_3, A_4$，即

$$A_1 = \sum_{i|n} C_n^i \, mod \; 2$$

$$A_2 = \sum_{i|n} C_n^i \, mod \; 3$$

$$A_3 = \sum_{i|n} C_n^i \, mod \; 4679$$

$$A_4 = \sum_{i|n} C_n^i \, mod \; 35617$$

然后进一步得到同余方程组为

$$x \equiv A_1 (mod \; 2)$$
$$x \equiv A_2 (mod \; 3)$$
$$x \equiv A_3 (mod \; 4679)$$
$$x \equiv A_4 (mod \; 35617)$$



再通过中国剩余定理（**CRT**）可以求得最终答案 $x$。

代码：

[cpp]

```cpp
#include <iostream>
#include <string.h>
#include <stdio.h>

using namespace std;
typedef long long LL;

const int P = 999911659;

LL a[5] = {0, 0, 0, 0};
LL m[5] = {2, 3, 4679, 35617};
LL fac[5][36010];
LL N, G;

void Init()
{
    for(int i=0; i<4; i++)
    {
        fac[i][0] = 1;
        for(int j=1; j<36010; j++)
            fac[i][j] = fac[i][j-1] * j % m[i];
    }
```

```
}

LL quick_mod(LL a,LL b,LL m)
{
    LL ans = 1;
    a %= m;
    while(b)
    {
        if(b & 1)
        {
            ans = ans * a % m;
            b--;
        }
        b >>= 1;
        a = a * a % m;
    }
    return ans;
}

LL C(LL n,LL k,int cur)
{
    LL p = m[cur];
    if(k > n) return 0;
    return fac[cur][n] * quick_mod(fac[cur][k] * fac[cur][n-k], p - 2, p) % p;
}

LL Lucas(LL n,LL k,int cur)
{
    LL p = m[cur];
    if(k == 0)  return 1;
    return C(n % p, k % p, cur) * Lucas(n / p, k / p, cur) % p;
}

void extend_Euclid(LL a, LL b, LL &x, LL &y)
{
    if(b == 0)
    {
        x = 1;
        y = 0;
        return;
    }
    extend_Euclid(b, a % b,x, y);
    LL tmp = x;
    x = y;
    y = tmp - a / b * y;
}

LL RemindChina(LL a[],LL m[],int k)
{
    LL M = 1;
    LL ans = 0;
    for(int i=0; i<k; i++)
        M *= m[i];
    for(int i=0; i<k; i++)
    {
```

```
        LL x, y;
        LL Mi = M / m[i];
        extend_Euclid(Mi, m[i], x, y);
        ans = (ans + Mi * x * a[i]) % M;
    }
    if(ans < 0)
        ans += M;
    return ans;
}

int main()
{
    Init();
    while(cin>>N>>G)
    {
        a[0] = a[1] = 0;
        a[2] = a[3] = 0;
        if(G == P)
        {
            cout<<"0"<<endl;
            continue;
        }
        G %= P;
        for(int i=1; i*i <= N; i++)
        {
            if(N % i == 0)
            {
                LL x = i;
                a[0] = (a[0] + Lucas(N, x, 0)) % m[0];
                a[1] = (a[1] + Lucas(N, x, 1)) % m[1];
                a[2] = (a[2] + Lucas(N, x, 2)) % m[2];
                a[3] = (a[3] + Lucas(N, x, 3)) % m[3];
                x = N / i;
                if(i * i != N)
                {
                    a[0] = (a[0] + Lucas(N, x, 0)) % m[0];
                    a[1] = (a[1] + Lucas(N, x, 1)) % m[1];
                    a[2] = (a[2] + Lucas(N, x, 2)) % m[2];
                    a[3] = (a[3] + Lucas(N, x, 3)) % m[3];
                }
            }
        }
        LL ans = quick_mod(G, RemindChina(a, m, 4), P);
        cout<<ans<<endl;
    }
    return 0;
}
```

题目：已知有如下表达式

$$(1+x+x^2)^n=\sum_{i=0}^{2n}C_ix^i$$

给定 $n$ 和 $k$，求 $C_k mod\ 3$。

分析：如果直接二项式展开，这样会很麻烦，而且不容易求出，本题有技巧。做如下变换

$$\begin{aligned}(1+x+x^2)^n &\equiv (1+x+x^2-3x)^n mod\ 3\\ &\equiv (1-x)^{2n} mod\ 3\end{aligned}$$

所以问题变为求 $(-1)^k C_{2n}^k mod\ 3$ 的值。