

组合数取模

2012-10-03 12:41

10918人阅读

评论(9)

收藏

举报

分类: 数论 (68) ▼

版权声明：本文为博主原创文章，未经博主允许不得转载。

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

$$\begin{aligned}n &= n_k p^k + n_{k-1} p^{k-1} + \dots + n_1 p + n_0 \\m &= m_k p^k + m_{k-1} p^{k-1} + \dots + m_1 p + m_0\end{aligned}$$

那么得到

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

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 $1 \leq p \leq 10^9$, 并且 p 是素数。

代码:

```
[cpp] C 8
01. #include <iostream>
02. #include <string.h>
03. #include <stdio.h>
04.
05. using namespace std;
06. typedef long long LL;
07.
08. LL n,m,p;
09.
10. LL quick_mod(LL a, LL b)
11. {
12.     LL ans = 1;
13.     a %= p;
14.     while(b)
15.     {
16.         if(b & 1)
17.         {
18.             ans = ans * a % p;
19.             b--;
20.         }
21.         b >>= 1;
22.         a = a * a % p;
23.     }
24.     return ans;
25. }
26.
27. LL C(LL n, LL m)
28. {
29.     if(m > n) return 0;
30.     LL ans = 1;
31.     for(int i=1; i<=m; i++)
32.     {
33.         LL a = (n + i - m) % p;
34.         LL b = i % p;
35.         ans = ans * (a * quick_mod(b, p-2) % p) % p;
36.     }
37.     return ans;
38. }
39.
40. LL Lucas(LL n, LL m)
41. {
42.     if(m == 0) return 1;
43.     return C(n % p, m % p) * Lucas(n / p, m / p) % p;
44. }
45.
```

```

46. int main()
47. {
48.     int T;
49.     scanf("%d", &T);
50.     while(T--)
51.     {
52.         scanf("%I64d%I64d%I64d", &n, &m, &p);
53.         printf("%I64d\n", Lucas(n,m));
54.     }
55.     return 0;
56. }

```

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) $1 \leq m \leq n \leq 10^6$ 和 $1 \leq p \leq 10^5$ ，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] C
01. #include <iostream>
02. #include <string.h>
03. #include <stdio.h>
04.
05. using namespace std;
06. typedef long long LL;
07. const int N = 200005;
08.
09. bool prime[N];
10. int p[N];
11. int cnt;
12.
13. void isprime()
14. {
15.     cnt = 0;
16.     memset(prime, true, sizeof(prime));
17.     for(int i=2; i<N; i++)
18.     {
19.         if(prime[i])
20.         {
21.             p[cnt++] = i;
22.             for(int j=i+i; j<N; j+=i)

```

```
23.         prime[j] = false;
24.     }
25. }
26. }
27.
28. LL quick_mod(LL a,LL b,LL m)
29. {
30.     LL ans = 1;
31.     a %= m;
32.     while(b)
33.     {
34.         if(b & 1)
35.         {
36.             ans = ans * a % m;
37.             b--;
38.         }
39.         b >>= 1;
40.         a = a * a % m;
41.     }
42.     return ans;
43. }
44.
45. LL Work(LL n,LL p)
46. {
47.     LL ans = 0;
48.     while(n)
49.     {
50.         ans += n / p;
51.         n /= p;
52.     }
53.     return ans;
54. }
55.
56. LL Solve(LL n,LL m,LL P)
57. {
58.     LL ans = 1;
59.     for(int i=0; i<cnt && p[i]<=n; i++)
60.     {
61.         LL x = Work(n, p[i]);
62.         LL y = Work(n - m, p[i]);
63.         LL z = Work(m, p[i]);
64.         x -= (y + z);
65.         ans *= quick_mod(p[i],x,P);
66.         ans %= P;
67.     }
68.     return ans;
69. }
70.
71. int main()
72. {
73.     int T;
74.     isprime();
75.     cin>>T;
76.     while(T-->0)
77.     {
```

```
78.         LL n,m,P;
79.         cin>>n>>m>>P;
80.         n += m - 2;
81.         m--;
82.         cout<<Solve(n,m,P)<<endl;
83.     }
84.     return 0;
85. }
```

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 $C_{n-m+1}^m \bmod p$ 。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：

下载：

```
[cpp] C 8
01. #include <iostream>
02. #include <string.h>
03. #include <stdio.h>
04.
05. using namespace std;
06. typedef long long LL;
07.
08. const int N = 25;
09. const int MOD1 = 97;
10. const int MOD2 = 3761599;
11. const int MOD = MOD1 * MOD2;
12.
13. int m,n,k;
14. int a[N];
15. LL fac1[MOD1+10];
16. LL fac2[MOD2+10];
```

```
17. LL inv1,inv2;
18.
19. LL quick_mod(LL a,LL b,LL m)
20. {
21.     LL ans = 1;
22.     a %= m;
23.     while(b)
24.     {
25.         if(b & 1)
26.         {
27.             ans = ans * a % m;
28.             b--;
29.         }
30.         b >>= 1;
31.         a = a * a % m;
32.     }
33.     return ans;
34. }
35.
36. LL C(LL n,LL m,LL p,LL fac[])
37. {
38.     if(n < m) return 0;
39.     return fac[n] * quick_mod(fac[m] * fac[n-m], p - 2, p) % p;
40. }
41.
42. LL Lucas(LL n,LL m,LL p,LL fac[])
43. {
44.     if(m == 0) return 1;
45.     return C(n % p, m % p, p, fac) * Lucas(n / p, m / p, p, fac);
46. }
47.
48. void Init()
49. {
50.     fac1[0] = fac2[0] = 1;
51.     for(int i=1; i<MOD1; i++)
52.         fac1[i] = (fac1[i-1] * i) % MOD1;
53.     for(int i=1; i<MOD2; i++)
54.         fac2[i] = (fac2[i-1] * i) % MOD2;
55.     inv1 = MOD2 * quick_mod(MOD2, MOD1-2, MOD1);
56.     inv2 = MOD1 * quick_mod(MOD1, MOD2-2, MOD2);
57. }
58.
59. int main()
60. {
61.     Init();
62.     int T, tt = 1;
63.     scanf("%d",&T);
64.     while(T--)
65.     {
66.         scanf("%d%d%d",&n,&m,&k);
67.         for(int i=0; i<k; i++)
68.             scanf("%d",&a[i]);
69.         a[k] = m;
70.         LL ans = 1;
71.         for(int i=0; i<k; i++)
```

```

72.     {
73.         LL m1 = Lucas(a[i+1] - a[i] + n - 1, a[i+1] - a[i], MOD1, fac1);
74.         LL m2 = Lucas(a[i+1] - a[i] + n - 1, a[i+1] - a[i], MOD2, fac2);
75.         LL mm = (m1 * inv1 + m2 * inv2) % MOD;
76.         ans = ans * mm % MOD;
77.     }
78.     printf("Case #d: ", tt++);
79.     cout<<ans<<endl;
80. }
81. return 0;
82. }

```

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

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

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

$C_n^m \bmod 2$ 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: $1 \ll (n \text{ 二进制表示中 } 1 \text{ 的个数})$

代码:

```

[cpp] C 8
01. #include <iostream>
02. #include <string.h>
03. #include <stdio.h>
04.
05. using namespace std;
06.

```



```

07.  int main()
08.  {
09.      int n;
10.      while(scanf("%d",&n)!=EOF)
11.      {
12.          int cnt = 0;
13.          while (n)
14.          {
15.              if (n & 1) cnt++;
16.              n >>= 1;
17.          }
18.          printf("%d\n",1<<cnt);
19.      }
20.      return 0;
21.  }

```

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

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

分析: 由于999911659是素数, 用费马小定理降幂得到

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

现在关键是求

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

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

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

即

$$A_1 = \sum_{i|n} C_n^i \bmod 2$$

$$A_2 = \sum_{i|n} C_n^i \bmod 3$$

$$A_3 = \sum_{i|n} C_n^i \bmod 4679$$

$$A_4 = \sum_{i|n} C_n^i \bmod 35617$$

然后进一步得到同余方程组为

$$\begin{aligned} x &\equiv A_1 \pmod{2} \\ x &\equiv A_2 \pmod{3} \\ x &\equiv A_3 \pmod{4679} \\ x &\equiv A_4 \pmod{35617} \end{aligned}$$

再通过中国剩余定理（**CRT**）可以求得最终答案 x 。

代码：

```
[cpp] C 8
01. #include <iostream>
02. #include <string.h>
03. #include <stdio.h>
04.
05. using namespace std;
06. typedef long long LL;
07.
08. const int P = 999911659;
09.
10. LL a[5] = {0, 0, 0, 0};
11. LL m[5] = {2, 3, 4679, 35617};
12. LL fac[5][36010];
13. LL N, G;
14.
15. void Init()
16. {
17.     for(int i=0; i<4; i++)
18.     {
19.         fac[i][0] = 1;
20.         for(int j=1; j<36010; j++)
21.             fac[i][j] = fac[i][j-1] * j % m[i];
22.     }
```

```
23. }
24.
25. LL quick_mod(LL a,LL b,LL m)
26. {
27.     LL ans = 1;
28.     a %= m;
29.     while(b)
30.     {
31.         if(b & 1)
32.         {
33.             ans = ans * a % m;
34.             b--;
35.         }
36.         b >>= 1;
37.         a = a * a % m;
38.     }
39.     return ans;
40. }
41.
42. LL C(LL n,LL k,int cur)
43. {
44.     LL p = m[cur];
45.     if(k > n) return 0;
46.     return fac[cur][n] * quick_mod(fac[cur][k] * fac[cur][n-k], p - 2, p) % p;
47. }
48.
49. LL Lucas(LL n,LL k,int cur)
50. {
51.     LL p = m[cur];
52.     if(k == 0) return 1;
53.     return C(n % p, k % p, cur) * Lucas(n / p, k / p, cur) % p;
54. }
55.
56. void extend_Euclid(LL a, LL b, LL &x, LL &y)
57. {
58.     if(b == 0)
59.     {
60.         x = 1;
61.         y = 0;
62.         return;
63.     }
64.     extend_Euclid(b, a % b,x, y);
65.     LL tmp = x;
66.     x = y;
67.     y = tmp - a / b * y;
68. }
69.
70. LL RemindChina(LL a[],LL m[],int k)
71. {
72.     LL M = 1;
73.     LL ans = 0;
74.     for(int i=0; i<k; i++)
75.         M *= m[i];
76.     for(int i=0; i<k; i++)
77.     {
```

```
78.         LL x, y;
79.         LL Mi = M / m[i];
80.         extend_Euclid(Mi, m[i], x, y);
81.         ans = (ans + Mi * x * a[i]) % M;
82.     }
83.     if(ans < 0)
84.         ans += M;
85.     return ans;
86. }
87.
88. int main()
89. {
90.     Init();
91.     while(cin>>N>>G)
92.     {
93.         a[0] = a[1] = 0;
94.         a[2] = a[3] = 0;
95.         if(G == P)
96.         {
97.             cout<<"0"<<endl;
98.             continue;
99.         }
100.        G %= P;
101.        for(int i=1; i*i <= N; i++)
102.        {
103.            if(N % i == 0)
104.            {
105.                LL x = i;
106.                a[0] = (a[0] + Lucas(N, x, 0)) % m[0];
107.                a[1] = (a[1] + Lucas(N, x, 1)) % m[1];
108.                a[2] = (a[2] + Lucas(N, x, 2)) % m[2];
109.                a[3] = (a[3] + Lucas(N, x, 3)) % m[3];
110.                x = N / i;
111.                if(i * i != N)
112.                {
113.                    a[0] = (a[0] + Lucas(N, x, 0)) % m[0];
114.                    a[1] = (a[1] + Lucas(N, x, 1)) % m[1];
115.                    a[2] = (a[2] + Lucas(N, x, 2)) % m[2];
116.                    a[3] = (a[3] + Lucas(N, x, 3)) % m[3];
117.                }
118.            }
119.        }
120.        LL ans = quick_mod(G, RemindChina(a, m, 4), P);
121.        cout<<ans<<endl;
122.    }
123.    return 0;
124. }
```

题目：已知有如下表达式

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

给定 n 和 k ，求 $C_k \bmod 3$ 。

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

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

所以问题变为求 $(-1)^k C_{2n}^k \bmod 3$ 的值。