**组合数C(n,m)的四种计算方法**

组合c(m,n)的计算方法

**问题：求解组合数C(n,m)，即从n个相同物品中取出m个的方案数，由于结果可能非常大，对结果模10007即可。**

共四种方案。ps：注意使用限制。

# 方案1:

**暴力求解，C(n,m)=n\*(n-1)\*...\*(n-m+1)/m!，**n<=15 ；

1. int Combination(int n, int m)
2. {
3. const int M = 10007;
4. int ans = 1;
5. for(int i=n; i>=(n-m+1); --i)
6. ans \*= i;
7. while(m)
8. ans /= m--;
9. return ans % M;
10. }

# 方案2:

打表，C(n,m)=C(n-1,m-1)+C(n-1,m)，**n<=10,000**

1. const int M = 10007;
2. const int MAXN = 1000;
3. int C[MAXN+1][MAXN+1];
4. void Initial()
5. {
6. int i,j;
7. for(i=0; i<=MAXN; ++i)
8. {
9. C[0][i] = 0;
10. C[i][0] = 1;
11. }
12. for(i=1; i<=MAXN; ++i)
13. {
14. for(j=1; j<=MAXN; ++j)
15. C[i][j] = (C[i-1][j] + C[i-1][j-1]) % M;
16. }
17. }
18. int Combination(int n, int m)
19. {
20. return C[n][m];
21. }

# 方案3:

质因数分解，C(n,m)=n!/(m!\*(n-m)!)，C(n,m)=p1a1-b1-c1p2a2-b2-c2…pkak-bk-ck,**n<=10,000,000**

1. #include <cstdio>
2. const int maxn=1000000;
3. #include <vector>
4. using namespace std;
5. bool arr[maxn+1]={false};
6. vector<int> produce\_prim\_number()
7. {
8. vector<int> prim;
9. prim.push\_back(2);
10. int i,j;
11. for(i=3;i\*i<=maxn;i+=2)
12. {
13. if(!arr[i])
14. {
15. prim.push\_back(i);
16. for(j=i\*i;j<=maxn;j+=i)
17. arr[j]=true;
18. }
19. }
20. while(i<maxn)
21. {
22. if(!arr[i])
23. prim.push\_back(i);
24. i+=2;
25. }
26. return prim;
27. }
28. //计算n!中素数因子p的指数
29. int cal(int x,int p)
30. {
31. int ans=0;
32. long long rec=p;
33. while(x>=rec)
34. {
35. ans+=x/rec;
36. rec\*=p;
37. }
38. return ans;
39. }
40. //计算n的k次方对m取模，二分法
41. int pow(long long n,int k,int M)
42. {
43. long long ans=1;
44. while(k)
45. {
46. if(k&1)
47. {
48. ans=(ans\*n)%M;
49. }
50. n=(n\*n)%M;
51. k>>=1;
52. }
53. return ans;
54. }
55. //计算C（n，m）
56. int combination(int n,int m)
57. {
58. const int M=10007;
59. vector<int> prim=produce\_prim\_number();
60. long long ans=1;
61. int num;
62. for(int i=0;i<prim.size()&&prim[i]<=n;++i)
63. {
64. num=cal(n,prim[i])-cal(m,prim[i])-cal(n-m,prim[i]);
65. ans=(ans\*pow(prim[i],num,M))%M;
66. }
67. return ans;
68. }
69. int main()
70. {
71. int m,n;
72. while(~scanf("%d%d",&m,&n),m&&n)
73. {
74. printf("%d\n",combination(m,n));
75. }
76. return 0;
77. }

# 方案4:

Lucas定理，将m,n化为p进制,有:C(n,m)=C(n0,m0)\*C(n1,m1)...(mod p)，算一个不是很大的C(n,m)%p,p为素数，化为线性同余方程,用扩展的欧几里德定理求解，**n在int范围内，修改一下可以满足long long范围内。**

1. #include <stdio.h>
2. const int M = 2013;
3. int ff[M+5]; //打表，记录n!，避免重复计算
5. //求最大公因数
6. int gcd(int a,int b)
7. {
8. if(b==0)
9. return a;
10. else
11. return gcd(b,a%b);
12. }
14. //解线性同余方程，扩展欧几里德定理
15. int x,y;
16. void Extended\_gcd(int a,int b)
17. {
18. if(b==0)
19. {
20. x=1;
21. y=0;
22. }
23. else
24. {
25. Extended\_gcd(b,a%b);
26. long t=x;
27. x=y;
28. y=t-(a/b)\*y;
29. }
30. }
32. //计算不大的C(n,m)
33. int C(int a,int b)
34. {
35. if(b>a)
36. return 0;
37. b=(ff[a-b]\*ff[b])%M;
38. a=ff[a];
39. int c=gcd(a,b);
40. a/=c;
41. b/=c;
42. Extended\_gcd(b,M);
43. x=(x+M)%M;
44. x=(x\*a)%M;
45. return x;
46. }
48. //Lucas定理
49. int Combination(int n, int m)
50. {
51. int ans=1;
52. int a,b;
53. while(m||n)
54. {
55. a=n%M;
56. b=m%M;
57. n/=M;
58. m/=M;
59. ans=(ans\*C(a,b))%M;
60. }
61. return ans;
62. }
64. int main()
65. {
66. int i,m,n;
67. ff[0]=1;
68. for(i=1; i<=M; i++) //预计算n!
69. ff[i]=(ff[i-1]\*i)%M;
70. while(~scanf("%d%d",&n, &m))
71. {
72. printf("%d\n",Combination(n,m));
73. }
74. return 0;
75. }