0. 模运算和乘法逆元

- 由于计数问题经常涉及到非常大的数，以至于按位输出这些数的复杂度都是不可接受的，于是采用模运算将结果限定在一个有限的范围内。
- 模运算的性质
  - 如果$a=b$，那么$a\%p=b\%p$
  - $(a+b)\%p=a\%p+b\%p$
  - $(a-b)\%p=(a\%p-b\%p+p)\%p(a>b)$
  - $(a\times b)\%p=a\%p\times b\%p$
- 乘法逆元
  - 为了将在模意义下表示除法，必须找到一个和某数相乘得1的数，这个数就是乘法逆元
  - 如果$(x\times\hat{x})\%p=1$，则称$\hat{x}$是x的乘法逆元，那么一个数除以x可以代替为乘$\hat{x}$
  - 如果p是素数，那么每个数存在逆元且唯一
  - 逆元求法

```
LL inv(LL x, LL p)
{
    if (x==1) return 1;
    else return (p-p/x)*inv(p%x, p)%p;
}
```

  - 费马小定理
    - $a^{p-1} \mod p=1$
  - 快速幂代码

```
LL exp(LL a, LL b, LL p)
{
    a%=p;
    LL tmp=1;
    while (b)
    {
        if (b&1) tmp=(tmp*a)%p;
        a=a*a%p;
        b=b/2;
    }
    return tmp;
}
//逆元:
rev[x]=exp(x, p-2, p);
```

- 最大公约数和最小公倍数

```
LL gcd(LL a,LL b)
{
    return (b==0?a:gcd(b, a%b));
}
LL lcm(LL a, LL b)
{
    return a*b/gcd(a,b);
}
```

- 筛法求素数

```
void get_prime(int n)
{
    memset(np,0,sizeof(np));
    for (int i=2;i<=n;i++)
    {
        if (!np[i])
        {
            prime.push_back(i);
        }
        for (int j=0;j<prime.size()&&prime[j]*i<=n;j++)
        {
            np[prime[j]*i]=1;
            if (i%prime[j]==0) break;
        }
    }
}
```

1. 计数原理

- 加法原理
  - 如果做一件事的全部方法可以分成互不相关（互相独立）的k类，其中属于第i类的方法有a_i种，那么做这件事的方案共有$\sum_{i=1}^{k} a_i$种
- 乘法原理
  - 如果做一件事要经过k个步骤，并且在i-1步完成后，做第i步的方法有a_i种，那么做这件事的方案共有$\prod_{i=1}^{k} a_i$种
- 等价原理
  - 设A, B是两个有限集，如果存在A到B的一一映射，则$|A| = |B|$

2. 计数公式

- 排列
  - 从n个元素中取出m个元素并指定这m个元素的顺序的方案数
  - 排列$P(n,m) = \frac{n!}{(n-m)!}$

        - 环形排列$\hat{P}(n,m)=\frac{P(n,m)}{m}$
    - 多重排列$P_r(n,m)=n^m$
        - 环形多重排列$\hat{P}_r(n,m)=\frac{\sum_{d|m}\phi(d)n^{\frac{m}{d}}}{m}$
- 组合
    - 从n个元素中取出m个元素的方案数
    - 组合$C(n,m)=\frac{n!}{(n-m)!m!}$
    - 多重组合$C_r(n,m)=(n+m-1,n)$
        - 等价于不定方程$\sum_{i=1}^{m}X_i=n$的非负整数解的个数
- 排列组合求法
    - 递推预处理阶乘和阶乘的逆元

```
frac[0] = 1;
for (int i=1;i<=n;i++)
    frac[i] = frac[i-1]*i%p
invf[n] = inv(frac[n],p)
for (int i=n-1;i>=0;i--)
    invf[i] = invf[i+1]*(i+1)%p
```

    - 求组合和排列复杂度$O(n)$

```
LL C(LL n, LL m)
{
    if (n<m) return 0;
    return frac[n]*invf[n-m]%p*invf[m]%p;
}
LL P(LL n, LL m)
{
    if (n<m) return 0;
    return frac[n]*invf[n-m]%p;
}
```

    - 卢卡斯定理
        - 当n和m很大时，可以将n和m分解成p进制数，对每一位求组合数$C(n,m)=\prod C(n_i,m_i)$，
          其中$n=\sum n_i\times p^i$，$m=\sum m_i\times p^i$
        - 复杂度$O(p\times logp)$

```
//$p<=1e5$
LL com(LL n,LL m,LL p)
{
    if (m>n) return 0;
    LL ans=1;
    for (int i=1;i<=m;i++)
    {
        ans=ans*(n-i+1)%p;
        ans=ans*exp(i,p-2,p)%p;
    }
    return ans;
}

LL lucas(LL n,LL m,LL p)
{
    if (m==0) return 1;
    return (com(n%p,m%p,p)*lucas(n/p,m/p,p))%p;
}
```

3. 递推数列

- 斐波那契数列
  - $1, 1, 2, 3, 5, 8, 13, 21...$
  - 递推式 $f_n = f_{n-1} + f_{n-2}$
  - 通项式 $f_n = \frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^n - \frac{1}{\sqrt{5}}(\frac{1-\sqrt{5}}{2})^n$
  - 等价问题
    - 集合$\{1, 2, ..., n-2\}$的不含相邻两数的子集个数
    - 用1×2的骨牌完全覆盖$2 \times n - 1$格子的方案数
    - 用1步或2步登上$n-1$阶台阶的方案数
  - 转移矩阵
    - $S_0 = \begin{pmatrix} f_0 & f_1 \end{pmatrix}$，$T = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$，$S_i = S_0 T^i$
    - 矩阵快速幂代码

```cpp
const int ms=5;
struct matrix
{
    LL e[ms][ms];
    matrix()
    {
        memset(e,0,sizeof(e));
    }
    matrix(int x)
    {
        memset(e,0,sizeof(e));
        for (int i=0;i<ms;i++) e[i][i]=x;
    }
    matrix operator *(const matrix &b)const{
        matrix c;
        for (int k=0;k<ms;k++)
            for (int i=0;i<ms;i++) if(e[i][k])
                for (int j=0;j<ms;j++) if(b.e[k][j])
                    c.e[i][j]+=e[i][k]*b.e[k][j];
        return c;
    }
    friend matrix operator ^(matrix e, LL k)
    {
        matrix tmp=matrix(1);
        while (k)
        {
            if (k&1) tmp=tmp*e;
            k=k>>1;
            e=e*e;
        }
        return tmp;
    }
};
```

- 卡特兰数
  - $1,1,2,5,14,42,132...$
  - 递推式$C_n=\sum_{i=1}^{n-1}C_iC_{n-i}$
  - 通项式$C_n=\frac{1}{n}C(2n-2,n-1)$
  - 等价问题
    - 三角剖分：凸$n+1$边形的三角剖分数
    - 乘法结合：给定顺序的n个元素乘法不同结合方案数
    - 不交连弦：圆周上$2(n-1)$个点的不想交连弦的方案数
    - 01序列：任意前缀0的个数大于1个数的长度为$2(n-1)$的01序列的个数
- 斯特林数
  - 递推式

$$S_1(n,k)=S_1(n-1,k-1)-(n-1)S_1(n-1,k)$$

$$S_2(n,k)=S_2(n-1,k-1)+kS_2(n-1,k)$$

   - 等价问题
      - $|S_1(n,k)|$是n次对称群S_n中恰由k个轮换构成的置换个数
      - $S_2(n,k)$是n元集无序拆分成k个非空子集的方案数
- 贝尔数
   - $1,1,2,5,15,52\dots$
   - 递推式 $B_n=\sum_{i=0}^{n} C(n,i)B_i$
   - $B_n=\sum_{i=1}^{n} S_2(n,i)$
   - 等价问题
      - n元集合的无序拆分程若干非空子集的方案数
- 球盒公式
  n个球放入m个盒子

球标号	盒标号	盒可空	公式	等价形式
✓	✓	✓	m^n	m元集合的n多重排列
✓	×	×	$S_2(n,m)$	n元集合无序分拆成m个非空子集
✓	×	✓	$\sum_{i=1}^{m} S_2(n,m)$	n元集合无序分拆成m个子集
✓	✓	×	$m!S_2(n,m)$	n元集合有序分拆成m个非空子集
×	✓	✓	$C(m+n-1,n)$	$x_1+\dots+x_m=n$的非负整数解个数
×	✓	×	$C(n-1,n-m)$	$x_1+\dots+x_m=n$的正整数解个数
×	×	×	$p_m(n)$	整数n无序拆分成m个部分
×	×	✓	$\sum_{i=1}^{m} p_i(n)$	整数n无序拆分成最多m个部分

4. 容斥原理

- 定义：若$S=A_1\cup A_2\cup\dots\cup A_n$，则

$$|S|=\sum_{i}^{n}|A_i|-\sum_{i,j}^{n}|A_i\cap A_j|+\sum_{i,j,k}^{n}|A_i\cap A_j\cap A_k|-\dots+(-1)^{n-1}|A_1\cap\dots\cap A_n|$$

- 解题一般过程：设有元素集A和性质集P，设$W(p_{i_1},p_{i_2},\dots,p_{i_r})$为集合$A$中具有$P$里$p_{i_1},p_{i_2},\dots,p_{i_r}$性质的元素个数，$W(r)=\sum_{i_1,i_2,\dots,i_r} W(p_{i_1},p_{i_2},\dots,p_{i_r})$，$E(r)$为集合$A$中恰好具有$r$种性质的元素个数。于是有

$$E(r)=\sum_{k=r}^{n}(-1)^{k-r}C(k,r)W(k)$$

特别地，

$$E(0)=\sum_{k=0}^{n}(-1)^{k}W(k)$$

- 注意：在$W(p_{i_1},p_{i_2},...,p_{i_r})$中，没有元素被重复计数，但在$W(r)$中，那些具有$k$个性质($k\geq r$)的元素被重复计数$C(k,r)$次
- 有上限的不定方程

$$x_1+x_2+...+x_m=n$$

其中$0\leq x_i<n_i$
令性质$p_i=[x_i\geq n_i]$，$0\leq z_i=x_i-n_i$

$$W(p_{i_1}p_{i_2}...p_{i_r})=C(n-\sum_{k=1}^{r}n_{i_k}+m-1,n-\sum_{k=1}^{r}n_{i_k})$$

$$W(r)=\sum_{i_1i_2...i_r}C(n-\sum_{k=1}^{r}n_{i_k}+m-1,n-\sum_{k=1}^{r}n_{i_k})$$

$$E(0)=C(n+m-1,n)+\sum_{r=1}^{m}(-1)^{r}W(r)$$

- 错位排列
  - n个数的排列，其中每个数都不能在自己原来的位置（1不能在第1个，2不能在第2个...）有多少种方案。
  - 递推式：
    - $D_n=(n-1)*(D_{n-1}+D_{n-2})$
  - 容斥：
    - $D_n=n!-\binom{n}{1}*(n-1)!+\binom{n}{2}*(n-2)!-\binom{n}{3}*(n-3)!+...$
    - $=n!-\frac{n!}{1!}+\frac{n!}{2!}-\frac{n!}{3!}...=n!(1-\frac{1}{1!}+\frac{1}{2!}-\frac{1}{3!}+...+\frac{1}{n!})$

5. 生成函数

- 什么是生成函数
  - 假设我们有一个具有组合意义的数列$\{a_n\}$，我们可以将其写成幂级数的形式，记为
    $$G(x)=a_0+a_1x+a_2x^2+a_3x^3+...+a_nx^n$$
    为数列a_n
  - 在这里，很多初学者会不理解这里x的含义，只是简单地把它理解成为一个变量，通常他们会想，“我把x带入不同的数字会得到什么结果”。这里注意，生成函数中的x通常来说不是一个数字，而生成函数的数值也往往没有什么意义。这里的x而是代表一个基本的组合对象，同时也可以理解成一种抽象的占位符（类似数组的第i项），而x的指数代表这一组合对象出现的“次数”。
  - 举例来说，在$G(x)$中，设x表示一个字符（0或1），那么x^n的系数就是长度为n的01字符串的个数。特别地，x^0的系数是1，表示空串。
- 生成函数到底有什么用
  - 这是初学者最大的疑问，生成函数到底是干什么用的。我觉得这里通常有一个误区是把“生成函数”当做某种“算法”，认为它是用来解决某一类问题的。的确，生成函数可以很方便地处理某几类组合问题，但是我更愿意把生成函数看做一种“数学语言”，是用来描述“组合问题”的一种便捷的工具。生成函数把抽象的组合问题表达成数学语言（组合对象抽象为变量，组合方法抽象为运算，组合的技术抽象为系数），从而用公式的推导来解决组合问题。
  - 具体来说，比如斐波那契数列，我们可以用数列$1,1,2,3,5,8,13...$来表示，也可以用递推$f_i=f_{i-1}+f_{i-2},f_0=1$表示，那么同样可以用生成函数
    $$F(x)=1+x+2x^2+3x^3+5x^4+...=\frac{1}{1-x-x^2}$$
- 生成函数的组合意义
  生成函数中包含的基本运算有加法和乘法（乘幂可以看做若干乘法的叠加），这里的加法和乘法并不仅仅是通常意义的加法和乘法，我们如果将它们理解为“加法原理”和“乘法原理”，则会对理解生成函数有很大的帮助。
  - 加法原理
    A_1+A_2表示选择取A_1或者选择取A_2（类比概率中的A_1发生或者A_2发生）。
  - 乘法原理
    $A_1\times A_2$表示选取A_1并且选取A_2（类别概率中的A_1发生并且A_2发生）。
  - 例子
    A代表数字0或数字1，即$A=0+1=2x$，那么$A^2=A\times A=(0+1)\times(0+1)=00+01+10+11=0^2+01+10+1^2=4x^2$就代表“所有长度为2的01串”这个组合问题。若要求“所有不同长度的01串”，则可以用生成函数$G(A)=1+A+A^2+...=1+(0+1)+(0^2+01+10+1^2)+...=1+2x+4x^2+...$来表示。这三个都是表示这

同一组合问题的生成函数，但是“A”，“x”，“0”，“1”，“01”，“10”...却是不同的基本组合对象，所以它们前面的系数是不同的。

- 普通型生成函数
  - $OGF(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + ... + a_nx^n$
  - 组合对象：x^i，表示i个x的"集合"
  - 组合方法：$x^i \times x^j = x^{i+j}$（i个元素的集合和j个元素的集合合并）
  - 组合意义：计数有关“i个x的集合”(无标号对象)出现的次数
  - 收敛形式：

收敛式	级数式	数列
$\frac{1}{1-x}$	$1 + x + x^2 + ...$	$1, 1, 1, 1, ...$
$\frac{k}{1-x}$	$k + kx + kx^2 + ...$	$k, k, k, k, ...$
$\frac{1}{1-kx}$	$1 + kx + k^2x^2 + ...$	$1, k, k^2, ...$
$\frac{1}{1-x^k}$	$1 + x^k + x^{2k} + ...$	$1, 0...0, 1, 0...0, 1, ...$
$\frac{x^k}{1-x}$	$x^k + x^{k+1} + ...$	$0...0, 1, 1, ...$
$\frac{1}{(1-x)^2}$	$1 + 2x + 3x^2 + ...$	$1, 2, 3, ...$

  - 例子1：多米诺骨牌拼2n的地砖
  - 例子2：换零钱
    - $P = 1 + ① + ①^2 + ①^3 + ...$
    - $N = P(1 + ⑤ + ⑤^2 + ⑤^3 + ...)$
    - $D = N(1 + ⑩ + ⑩^2 + ⑩^3 + ...)$
    - 收敛式：$\frac{1}{1-z}\frac{1}{1-z^5}\frac{1}{1-z^{10}}$
- 指数型生成函数
  - $EGF(x) = a_0 + a_1\frac{x}{1!} + a_2\frac{x^2}{2!} + a_3\frac{x^3}{3!} + ... + a_n\frac{x^n}{n!}$
  - 组合对象：$\frac{x^i}{i!}$，表示i个x的“序列”（有标号对象，这里注意分母上的阶乘并没有数值含义，仅仅是为了将组合对象从“集合”变成“序列”，所以不必去想$x^i/i!$的意义。）
  - 组合方法：$\frac{x^i}{i!} \times \frac{x^j}{j!} = \frac{x^{i+j}}{i!j!} = \binom{i+j}{i}\frac{x^{i+j}}{(i+j)!}$（i个元素的序列和j个元素的序列合并，保持两个序列中的元素相对位置不变。abc和de合并，保持相对顺序，即从_ _ _ _ _中选择3个位置先放abc，剩下的位置就是de）
  - 组合意义：计数有关“i个x的序列”出现的次数
  - 收敛式

收敛式	级数式	数列
e^x	$1 + \frac{x}{1!} + \frac{x^2}{2!} + ...$	$1, 1, 1, 1, ...$

收敛式	级数式	数列
e^{-x}	$1-\frac{x}{1!}+\frac{x^2}{2!}-...$	$1,-1,1,-1,...$
$\frac{e^x+e^{-x}}{2}$	$1+\frac{x^2}{2!}+\frac{x^4}{4!}+...$	$1,0,1,0,...$
$\frac{e^x-e^{-x}}{2}$	$\frac{x}{1!}+\frac{x^3}{3!}+\frac{x^5}{5!}+...$	$0,1,0,1,...$
e^{kx}	$1+k\frac{x}{1!}+k^2\frac{x^2}{2!}+...$	$1,k,k^2,k^3,...$

- 例子1：构数问题
  1个1，2个2，3个3,4个4构成哪些5位数
  - $(1+x)(1+x+\frac{x^2}{2!})(1+x+\frac{x^2}{2!}+\frac{x^3}{3!})(1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}) \to a_5\frac{x^5}{5!}$
- 例子2：染色问题
  红黄蓝三种颜色染n个格子，要求红色为奇数，蓝色为偶数
  - $(x+\frac{x^3}{3!}+\frac{x^5}{5!}+...)(1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}+...)(1+\frac{x^2}{2!}+\frac{x^4}{4!}+...)=\frac{1}{2}(e^x+e^{-x})e^x\frac{1}{2}(e^x-e^{-x})=\frac{1}{4}(e^{3x}-e^{-x}) \to a_n=\frac{3^n-(-1)^n}{4}$