生成函数的运算与组合计数问题

杭州学军中学 金策

摘 要

本文介绍了处理形式幂级数的一些高效算法，并在生成函数的运算过程中加以应用，从而解决一系列组合计数问题。

目 录

1 引言

组合计数问题是信息学竞赛中常见的一类问题，而生成函数往往是解决这类问题的重要工具。近年来，信息学竞赛中出现了这样一类计数问题，不仅需要选手根据题意对生成函数进行分析与推导，还要求使用高效的算法完成各类多项式运算，才能优化求解的时间复杂度。本文将对这一类问题进行进一步分析与总结。

本文第2,3节回顾了一些需要用到的基本知识。其中第2节提到了几种熟知的多项式乘法算法，以及算法实现时需要注意的地方；第3节引入了组合对象的生成函数的概念，并分析了生成函数加法、乘法的组合意义。

第4节介绍了求解形式幂级数乘法逆元的牛顿迭代法，这一算法是后文许多算法的基础。第5节中的例题对该算法进行了简单应用。

第6节介绍了形式幂级数的对数、指数函数的求解算法。

第7,8节分析了集合、环这两类常见组合模型的计数方法，并对第6节中的算法进行了应用。

第9节简单介绍与复合运算相关的算法和定理。

第10节介绍了运用二元生成函数解决问题的技巧。

2 多项式与形式幂级数

2.1 多项式

多项式是我们熟知的数学概念。一个关于x的多项式可以写成

$$A(x) = \sum_{i=0}^{n-1} a_i x^i \tag{2.1}$$

的形式，其中系数a_i均为某个环R中[1]的元素。这些多项式组成多项式环$R[x]$。

非零多项式$A(x)$的次数定义为其最高次项的次数，记作$\deg A(x)$。

2.2 多项式的基本运算

设参与运算的多项式的最高次数为n。那么多项式的加法、减法显然可以在$O(n)$时间内计算。

[1]一般指可交换环，可以是复数域$\mathbb{C}$、实数域$\mathbb{R}$、整数环$\mathbb{Z}$、剩余类环$\mathbb{Z}/n\mathbb{Z}$等

我们关心的是两个多项式的乘积。朴素的计算方法需要$O(n^2)$时间，并不够优秀。

一种优化方法是分治乘法[2]，它的原理是利用

$$\begin{aligned}&(Ax^m+B)(Cx^m+D)\\=&ACx^{2m}+\big((A+B)(C+D)-AC-BD\big)x^m+BD\end{aligned}$$

减少乘法次数进行递归。复杂度为$O(n^{\log_2 3})=O(n^{1.585})$。

分治乘法的劣势在于复杂度较高，但它不涉及除法，所以对环R没有特别的要求。

此外，当运算在模2意义下进行时，我们也可以利用位运算加速，使得算法常数减少到原来的1/32。

另一种做法是使用FFT[3]。它利用单位根的性质，实现了多项式的系数表示与点值表示之间的快速转化。时间复杂度是$O(n\log n)$。

为了使用FFT，R需具有2^k次单位根($2^k\geq 2n$)，且存在2的乘法逆元。当系数在复数域内时，单位根总能找到，但计算时容易出现精度问题。

需要注意到的是，信息学竞赛中涉及的计数问题往往要求答案模一个大质数p(如10^9+7，或$998244353=7\times 17\times 2^{23}+1$等等)后输出。这样的好处包括：每次算术运算的时间可以视作$O(1)$，不存在精度问题，且大多数情况下可以支持除法(只要在问题规模范围内，除数都远小于p，存在乘法逆元)。因此本文中假设所有运算都在$\mathbb{F}_p$下进行。

在模p意义下进行FFT时，若满足$2^k\big|\varphi(p)=p-1$，则可取p的原根g，并用$g^{\frac{p-1}{2^k}}$作为单位根；否则，将系数视作$\mathbb{Z}$的元素进行运算后再对p取模。这时，相乘得到的系数大小不超过np^2，只要取若干个便于FFT的大质数分别进行运算，再用中国剩余定理还原系数即可。

2.3 形式幂级数

一个多项式仅有有限项的系数是非零的。若去掉这一限制，可将其推广为形式幂级数

$$A(x)=\sum_{i\geq 0}a_ix^i, \tag{2.2}$$

[2]详见http://en.wikipedia.org/wiki/Karatsuba%5falgorithm

[3]详见http://en.wikipedia.org/wiki/Discrete%5fFourier%5ftransform

它们组成了形式幂级数环$R[[x]]$。

此定义中，x仅作为一个符号，而不用具体的数值代入运算，故不必考虑与幂级数敛散性有关的问题。

我们用$[x^n]A(x)$表示$A(x)$的n次项系数a_n。

形式幂级数的加减法与乘法也可与多项式运算类似定义:

$$A(x)=\sum_{i\geq 0} a_i x^i, B(x)=\sum_{i\geq 0} b_i x^i, \tag{2.3}$$

$$A(x)\pm B(x)=\sum_{i\geq 0}(a_i \pm b_i)x^i, \tag{2.4}$$

$$A(x)B(x)=\sum_{k\geq 0}\Big(\sum_{i+j=k} a_i b_j\Big)x^k. \tag{2.5}$$

实际运算时，通常只需保留次数不超过$n-1$的项进行计算，并将多余的项舍去(即在模x^n意义下计算)。因此加减法的复杂度为$O(n)$，乘法的复杂度为$O(n\log n)$。

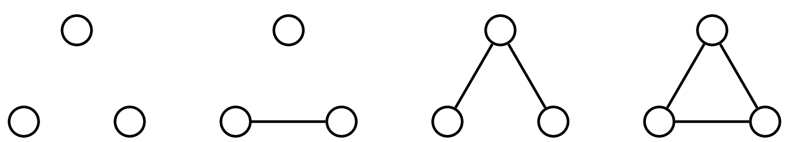
3 组合计数问题

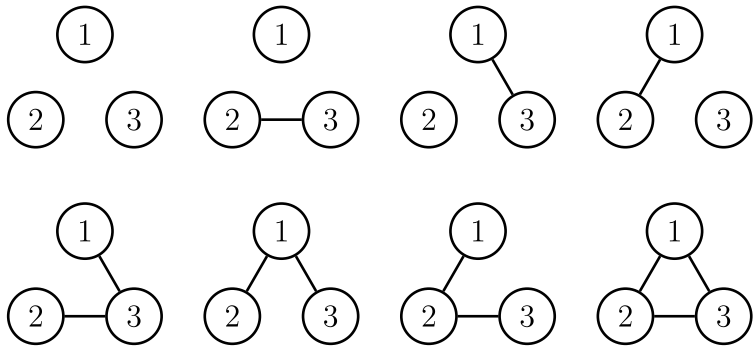
3.1 组合对象

组合计数问题是一类常见的问题。这类问题中一般定义了一类组合对象A，它可能是满足某一性质的树、图、串等对象的集合；其中每个对象$a\in A$都被定义了大小$size(a)\in\mathbb{N}$，它可能代表结点数量、序列长度等。对于某个固定的n，满足$size(a)=n$的对象a的数量是有限的，记作A_n。我们的任务通常为求出A_n的数值。

根据不同的问题要求，组合对象可以分成无标号和有标号两类。下面简单以无向图为例解释它们的区别：

n个点的标号图中，每个顶点都被赋予了$1,2,\cdots,n$中的唯一标号；然而在无标号图中，每个顶点的地位是没有区别的。如图所示，$n=3$时，无标号的简单无向图共有4种，而有标号的简单无向图共有8种。





3.2 普通生成函数

数列$A_0, A_1, \cdots$的普通生成函数(Ordinary Generating Function, OGF)定义为形式幂级数

$$A(x) = \sum_{i \ge 0} A_i x^i. \tag{3.1}$$

A是一类无标号对象，则A的普通生成函数$A(x)$定义为数列$A_0, A_1, \cdots$的普通生成函数，其中A_n为满足$size(a) = n$的对象$a \in A$的数量。

对于两类无标号对象A, B，定义一类新的对象$C = A \cup B$，若A, B交集为空，则C的生成函数即为

$$C(x) = A(x) + B(x). \tag{3.2}$$

再来考虑A, B的笛卡尔积$D = A \times B$，其中D的每个元素d都是一个二元组(a, b)，其中$a \in A, b \in B$，并定义$size(d) = size(a) + size(b)$。则有

$$D_k = \sum_{i+j=k} A_i B_j, \tag{3.3}$$

于是D的OGF即为

$$D(x) = A(x)B(x). \tag{3.4}$$

这一操作实现了A中元素和B中元素的拼接。

3.3 指数生成函数

数列$A_0, A_1, \cdots$的指数生成函数(Exponential Generating Function, EGF)定义为形式幂级数

$$A(x) = \sum_{i \ge 0} A_i \frac{x^i}{i!}. \tag{3.5}$$

指数生成函数在处理有标号问题时更加便捷。对于一类有标号对象A，A的指数生成函数$A(x)$定义为数列$A_0, A_1, \cdots$的指数生成函数，其中A_n为满足$size(a) = n$的对象$a \in A$的数量。

给定两类有标号对象A, B，对于它们的并集$C = A \cup B$，同样有

$$C(x) = A(x) + B(x). \tag{3.6}$$

现在考虑有标号对象的拼接。给定两个对象a, b，设$size(a) = n, size(b) = m$，它们分别带有$1, 2, \cdots, n$和$1, 2, \cdots, m$的标号。为将a, b拼接得到c，需给c分配$1, 2, \cdots, n+m$的标号。规定重新分配时需要保持标号的原有相对顺序，则有

$$\binom{n+m}{n} = \frac{(n+m)!}{n!\,m!} \tag{3.7}$$

种方法。

因此，若将两类带标号对象A, B拼接得到D，则有

$$D_k = \sum_{i+j=k} A_i B_j \frac{k!}{i!j!}, \tag{3.8}$$

从而D的EGF也具有

$$D(x) = A(x)B(x) \tag{3.9}$$

的形式。

4 乘法逆元

当$A(x)B(x) = 1$时，称$A(x), B(x)$互为乘法逆元，可以写作$A(x) = B(x)^{-1} = 1/B(x)$。除以一个形式幂级数，就相当于乘上它的乘法逆元。

例如，

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + \cdots. \tag{4.1}$$

根据这一等式可得，若数列$a_0, a_1, \cdots$的普通生成函数为$A(x)$，令$s_n = \sum_{i=0}^{n} a_i$，那么$s_0, s_1, \cdots$的普通生成函数为$A(x)/(1-x)$。

再结合二项式系数的递推性质，可得

$$\frac{1}{(1-x)^{n+1}} = \sum_{i \geq 0} \binom{n+i}{n} x^i. \tag{4.2}$$

接下来介绍，在给定$A(x)$时，如何求出$A(x)$的乘法逆元$B(x)$。

可以证明，$A(x)$存在乘法逆元的充要条件是$A(x)$的常数项存在乘法逆元。必要性由$\big([x^0]A(x)\big)\big([x^0]B(x)\big) = 1$可得。而当$A(x)$的常数项可逆时，即可根据乘法的定义(2.5)按顺序求出$[x^1]B(x)$, $[x^2]B(x)$, $\cdots$的值，于是此时乘法逆元存在且唯一。同时，我们得到了一个求出乘法逆元前n项的$O(n^2)$朴素算法。

下面介绍一个用$O(n \log n)$时间计算乘法逆元的算法，它的本质是牛顿迭代法。

首先求出$A(x)$的常数项的逆元b，并令$B(x)$的初始值为b。

假设已求得满足

$$A(x)B(x) \equiv 1 \pmod{x^n} \tag{4.3}$$

的$B(x)$，则

$$\begin{aligned} A(x)B(x) - 1 &\equiv 0 \pmod{x^n}, \\ \big(A(x)B(x) - 1\big)^2 &\equiv 0 \pmod{x^{2n}}, \\ A(x)\big(2B(x) - B(x)^2A(x)\big) &\equiv 1 \pmod{x^{2n}}. \end{aligned} \tag{4.4}$$

我们用$O(n \log n)$时间计算出$2B(x) - B(x)^2A(x)$，并将它赋值给$B(x)$进行下一轮迭代。每迭代一次，$B(x)$的有效项数n都会增加一倍。于是该算法的时间复杂度为

$$T(n) = T(n/2) + O(n \log n) = O(n \log n).$$

5 乘法逆元的应用

例题1 (序列计数). *你有若干种颜色不同的骨牌，其中大小为$1 \times i$的骨牌共有a_i种。每种骨牌都可以无限量使用。用骨牌不重叠地铺满一排$1 \times n$的方格，共有几种方法?*$(a_i, n \le 10^5)$

我们枚举使用的骨牌数量k。设$A(x) = \sum_{i\ge 0} a_i x^i$，则根据生成函数的乘法的意义，容易知道此时的答案为$[x^n]A(x)^k$。所以总方法数目为

$$[x^n]\sum_{k\ge 0} A(x)^k = [x^n]\frac{1}{1 - A(x)}. \tag{5.1}$$

于是只需计算$1 - A(x)$的乘法逆元即可，时间复杂度是$O(n \log n)$。

一般地，对于一类组合对象A，由A的元素组成的序列定义一类新的组合对象B，则B的生成函数为

$$B(x) = \frac{1}{1 - A(x)}. \tag{5.2}$$

这一结论对于无标号(OGF)、有标号(EGF)的情况都成立。

例题2 (预处理Bernoulli数[4]). 对于所有$0 \le i \le n-1$求出B_i。($n \le 10^5$)

Bernoulli数$B_0, B_1, \cdots$的指数生成函数为

$$\begin{aligned} B(x) &= \sum_{i \ge 0} B_i \frac{x^i}{i!} \\ &= \frac{x}{e^x - 1} \\ &= \Big(\sum_{i \ge 0} \frac{x^i}{(i+1)!} \Big)^{-1}. \end{aligned} \tag{5.3}$$

这样，只要求一次乘法逆元即可。

例题3. 字符集大小为m。给定一个长为k的字符串s，求出所有长为n的串中，不包含子串s的共有几个。($n, m, k \le 10^5$)

假定串的下标都从1开始。

考虑DP，用f_i表示有多少种方法填入前i个字符，使得第一次匹配上串s的位置是$[i-k+1, i]$。当$i < k$时，$f_i = 0$；当$i \ge k$时，

$$f_i = m^{i-k} - \sum_{0 \le j \le i-k} f_j m^{i-k-j} - \sum_{d > 0, d \in C} f_{i-d}, \tag{5.4}$$

其中集合C定义为$\big\{d \ge 0 \big| s[d+1, k] = s[1, k-d]\big\}$。

这个DP方程的含义是，末k位字符与串s相同，前$i-k$位字符可以任意确定，但为了保证$i-k+1$是第一次匹配上的位置，需要从中减去之前已经匹配过的情况。减去的第一项是这次匹配与上一次没有重叠的情况，第二项是与上次出现重叠的情况，为此需要先用KMP算法求出C集合。

为了优化这个DP，考虑f的生成函数$f(x) = \sum_{i \ge 0} f_i x^i$。令$C(x) = \sum_{d \in C} x^d$，则可由DP方程写出

$$f(x) = \frac{x^k}{1 - mx} - \frac{x^k f(x)}{1 - mx} - f(x)\big(c(x) - 1\big),$$

[4]定义见`http://en.wikipedia.org/wiki/Bernoulli%5fnumber`

即

$$f(x)=\frac{x^k}{x^k+(1-mx)c(x)}.$$

最后的答案即为$g_n=m^n-\sum_{i\geq 0}f_im^{i-j}$，而相应的生成函数为

$$g(x)=\frac{c(x)}{x^k+(1-mx)c(x)}. \tag{5.5}$$

时间复杂度是$O(n\log n)$。

6 对数与指数运算

6.1 复合运算

首先引入形式幂级数的复合运算。

设$A(w)=\sum_{i\geq 0}a_iw^i, B(x)=\sum_{i\geq 1}b_ix^i$，则$B(x)$与$A(w)$的复合为

$$C(x)=A\circ B(x)=A\big(B(x)\big)=\sum_{i\geq 0}a_i\big(B(x)\big)^i, \tag{6.1}$$

可以将其整理为$C(x)=\sum_{i\geq 0}c_ix^i$ 的形式。

由于已假定$B(x)$没有常数项，因此即使A有无限多个非零项，$B(x)$与$A(w)$的复合仍然可以定义。

6.2 形式导数

对于$A(x)=\sum_{i\geq 0}a_ix^i$，定义$A(x)$的形式导数为

$$A'(x)=\sum_{i\geq 1}ia_ix^{i-1}. \tag{6.2}$$

容易验证

$$\big(cA(x)\big)'=cA'(x), \tag{6.3}$$

$$\big(A(x)\pm B(x)\big)'=A'(x)\pm B'(x), \tag{6.4}$$

$$\big(A(x)B(x)\big)'=A'(x)B(x)+A(x)B'(x), \tag{6.5}$$

$$\left(\frac{1}{A(x)}\right)'=-\frac{A'(x)}{A(x)^2}, \tag{6.6}$$

$$\Big(A\big(B(x)\big)\Big)'=A'\big(B(x)\big)B'(x) \tag{6.7}$$

这些基本求导法则对于形式导数依然成立。

6.3 对数函数与指数函数

我们可以定义形式幂级数的对数与指数函数，这相当于将给定的级数$A(x)=\sum_{i\geq 1} a_i x^i$与对应的麦克劳林级数复合，即

$$\ln\big(1-A(x)\big) = -\sum_{i\geq 1}\frac{A(x)^i}{i}, \tag{6.8}$$

$$\exp\big(A(x)\big) = \sum_{i\geq 0}\frac{A(x)^i}{i!}. \tag{6.9}$$

容易验证，对数和指数函数的大多数性质在这里仍然成立。

6.3.1 对数函数的计算

给定$A(x)=1+\sum_{i\geq 1} a_i x^i$，令

$$B(x) = \ln\big(A(x)\big),$$

则

$$B'(x) = \frac{A'(x)}{A(x)}, \tag{6.10}$$

于是只需求出$A(x)$的乘法逆元，就可求出$\ln\big(A(x)\big)$，时间复杂度是$O(n\log n)$。

6.3.2 指数函数的计算

给定$A(x)=\sum_{i\geq 1} a_i x^i$，令

$$B(x) = \exp\big(A(x)\big) = \sum_{i\geq 0} b_i x^i,$$

则

$$B'(x) = B(x)A'(x), \tag{6.11}$$

比较系数得

$$\begin{aligned} b_0 &= 1, \\ b_i &= \frac{1}{i}\sum_{1\leq k\leq i} k a_k b_{i-k} \quad (i\geq 1). \end{aligned} \tag{6.12}$$

用分治法计算所有b_i，每次用FFT计算左半边的b_i对右半边所产生的贡献，时间复杂度是$O(n\log^2 n)$。

下一节的牛顿迭代法可以在$O(n\log n)$时间计算$\exp\big(A(x)\big)$。

6.4 牛顿迭代法

令$f(x)=e^{A(x)}$，可得到一个关于$f(x)$的方程

$$g\big(f(x)\big)=\ln\big(f(x)\big)-A(x)=0. \tag{6.13}$$

考虑用牛顿迭代法解这一方程[5]。首先$f(x)$的常数项是容易确定的。设已求得$f(x)$的前n项$f_0(x)$，即

$$f(x)\equiv f_0(x) \pmod{x^n}, \tag{6.14}$$

作泰勒展开，得

$$\begin{aligned}0&=g\big(f(x)\big)\\&=g\big(f_0(x)\big)+g'\big(f_0(x)\big)\big(f(x)-f_0(x)\big)+\frac{g''\big(f_0(x)\big)}{2}\big(f(x)-f_0(x)\big)^2+\cdots\\&\equiv g\big(f_0(x)\big)+g'\big(f_0(x)\big)\big(f(x)-f_0(x)\big) \pmod{x^{2n}},\end{aligned}$$

即

$$f(x)\equiv f_0(x)-\frac{g\big(f_0(x)\big)}{g'\big(f_0(x)\big)} \pmod{x^{2n}}, \tag{6.15}$$

将(6.13)代入，得

$$\begin{aligned}f(x)&=f_0(x)-\frac{\ln\big(f_0(x)\big)-A(x)}{1/f_0(x)}\\&=f_0(x)\Big(1-\ln\big(f_0(x)\big)+A(x)\Big).\end{aligned} \tag{6.16}$$

按此迭代，复杂度$T(n)=T(n/2)+O(n\log n)=O(n\log n)$。

[5]一般情况下，方程$g\big(f(x)\big)$有解需要g满足一定条件。但在这里并没有问题，因此不深入讨论。

6.5 k次幂的计算

例题4 (k次幂的计算). 给定多项式$f(x)$和正整数k，求出$f(x)^k$的前n项系数。$(n, k \le 10^5)$

普通的做法是利用倍增快速幂。每次用FFT将两个多项式相乘，并舍去多余项，时间是$O(n \log n)$。共做$O(\log k)$次，总时间$O(n \log n \log k)$。

根据指数函数和对数函数的性质，当$f(x)$的常数项为1时，有

$$f(x)^k = \exp\big(k \ln f(x)\big). \tag{6.17}$$

按此式计算，时间复杂度是$O(n \log n)$。

如果$f(x)$的常数项不为1，设$f(x)$的最低次项为ax^d，则

$$f(x)^k = a^k x^{kd} \left(\frac{f(x)}{ax^d}\right)^k,$$

于是转化成了常数项为1的情况，可以按照(6.17)计算。

我们也可以计算$f(x)$的k次方根。这时若$f(x)$的最低次项为ax^d，则需要保证$k|d$，且a存在k次方根。然后即可将常数项调整为1后按照上面的方法计算。

7 集合的计数

7.1 有标号集合的计数

例题1中已讨论了无标号对象所组成的序列的计数，且得到的结论对有标号对象也成立。下面考虑有标号对象组成的集合的计数。

集合与序列的区别在于集合内的元素没有顺序关系，需要将重复的统计去除。因此，A的元素组成的集合的EGF为

$$B(x) = \sum_{i \ge 0} \frac{A(x)^i}{i!} = e^{A(x)}. \tag{7.1}$$

例题5 (连通图计数). 求出n个顶点的连通无向图的个数。顶点有标号，不允许重边和自环。$(n \le 10^5)$

设G是所有简单无向图，则G的EGF为

$$G(x) = \sum_{n \ge 0} 2^{\binom{n}{2}} \frac{x^n}{n!}. \tag{7.2}$$

令C为所有简单连通图。由于一个简单无向图可以看做若干个连通分量组成的集合，于是

$$G(x) = e^{C(x)},$$

从而

$$C(x) = \ln G(x). \tag{7.3}$$

只要求出$G(x)$的对数即可求得答案。时间复杂度为$O(n \log n)$。

例题6 (有限制的置换计数). 给定集合S和正整数n，计算有多少个n阶置换p，满足p分解后的每一个轮换的大小都在S内。($n \le 10^5$)

一个置换是若干个轮换组成的集合，每个轮换相当于一个带标号的环。根据圆排列的计数，大小为k的轮换的数量有$(k-1)!$个，对应的EGF为

$$\frac{(k-1)!}{k!}x^k = \frac{x^k}{k}, \tag{7.4}$$

因此答案的EGF即为

$$\exp\Big(\sum_{k \in S} \frac{x^k}{k}\Big). \tag{7.5}$$

7.2 无标号集合的计数

例题7 (完全背包计数). 你有若干种不同的物品，其中体积为i的物品有a_i种。每种物品有无限个。求选取物品恰好装满总体积为n的背包的方案数。($a_i, n \le 10^5$)

在本题中，无标号集合里可能存在重复元素，因此不能简单地除以$k!$来去重。

令$A(x) = \sum_{i \ge 1} a_i x^i$，考虑对每一种体积$i$写出生成函数，并相乘，进一步

用对数函数化乘为加，由此得到OGF为

$$\begin{aligned}&\prod_{1\le i\le n}(1+x^i+x^{2i}+\cdots)^{a_i}\\ =&\prod_{1\le i\le n}(\frac{1}{1-x^i})^{a_i}\\ =&\exp\Big(-\sum_{1\le i\le n}a_i\ln(1-x^i)\Big)\\ =&\exp\Big(\sum_{1\le i\le n}a_i\sum_{j\ge1}\frac{x^{ij}}{j}\Big)\\ =&\exp\Big(\sum_{j\ge1}\frac{1}{j}A(x^j)\Big),\end{aligned}\tag{7.6}$$

注意到$A(x^j)$中只有n/j项是有用的，故可用

$$n+\frac{n}{2}+\frac{n}{3}+\cdots+1=O(n\log n)$$

的时间对所有$A(x^j)$求和。

最后再求一次exp，总复杂度为$O(n\log n)$。[6]

例题8 (01背包计数). *你有若干种不同的物品，其中体积为i的物品有a_i种。每种物品仅有1个。求选取物品恰好装满总体积为n的背包的方案数。*$(a_i,n\le 10^5)$

本题和上题的处理方法相同，这里不再赘述。

8 环的计数

这一节中考虑环的计数。环的特别之处在于它可以旋转，两个旋转后相同的环即被认为是同一个环。

8.1 有标号环的计数

比较容易处理的是有标号环的计数。根据圆排列公式，k元环的数量有$(k-1)!$个。因此，对于一类组合对象A，它的元素所组成的环(可旋转而不可

[6] 当a_i均为1时，本题即为整数无序拆分问题，可以使用欧拉五边形数定理更方便地解决，详见http://en.wikipedia.org/wiki/Pentagonal%5fnumber%5ftheorem

翻转)的EGF为

$$\sum_{k\geq 1}\frac{A(x)^k}{k} = -\ln\big(1-A(x)\big). \tag{8.1}$$

例题9. *求包含n个顶点，n条边的连通无向图有几个。顶点有标号。不允许重边和自环。*$(n\leq 10^5)$

由基本图论知识可知，这样的图包含一个长度不小于3的环，且环上的每个结点是一棵有根树的根。

先来解决树的问题。这里考虑的树是带标号有根树，且结点的儿子没有顺序。设有根树的EGF为$T(x)$。由Cayley定理[7] 可知，n个结点的有根树数量为n^{n-1}，因此[8]

$$T(x) = \sum_{n\geq 1}\frac{n^{n-1}x^n}{n!}. \tag{8.2}$$

再来考虑环。需要注意本题中的环是没有方向的(可以翻转)，于是由树组成的不小于3的环的EGF 为

$$\begin{aligned}&\frac{1}{2}\sum_{k\geq 3}\frac{T(x)^k}{k}\\ =&-\frac{1}{2}\ln\big(1-T(x)\big)-\frac{T(x)}{2}-\frac{T(x)^2}{4}.\end{aligned} \tag{8.3}$$

8.2 无标号环的计数

例题10 (无标号环的计数)**.** *你有若干种颜色不同的珠子，其中重量为i的珠子共有a_i种。每种珠子都可以无限量使用。用珠子串成一串重量为n的项链，共有几种方法？项链允许旋转，但不允许翻转。*$(a_i, n\leq 10^5)$

无标号环的处理稍微复杂一些，因为环中允许出现相同元素，可能产生循环节。我们容易想到利用Pólya计数公式[9] 来解决这一问题。

---

[7]详见`http://en.wikipedia.org/wiki/Cayley%27s%5fformula`

[8]也可以根据树的定义得到$T(x) = xe^{T(x)}$，再由拉格朗日反演推出这一结论

[9]详见`http://en.wikipedia.org/wiki/P%C3%B3lya%5fenumeration%5ftheorem`

给定一类组合对象A，由k个A中元素组成的环的OGF应当是

$$\begin{aligned}&\frac{1}{k}\sum_{0\le i<k}A(x^{k/\gcd(i,k)})^{\gcd(i,k)}\\=&\frac{1}{k}\sum_{d|k}\varphi(d)A(x^d)^{k/d},\end{aligned}$$

于是A中元素组成的环的OGF为

$$\begin{aligned}&\sum_{k\ge1}\frac{1}{k}\sum_{d|k}\varphi(d)A(x^d)^{k/d}\\=&\sum_{d\ge1}\varphi(d)\sum_{k\ge1}\frac{1}{kd}A(x^d)^k\\=&-\sum_{d\ge1}\frac{\varphi(d)}{d}\ln\big(1-A(x^d)\big).\end{aligned}\tag{8.4}$$

与例题7类似，$\ln\big(1-A(x^d)\big)$中只有n/d项是有用的，因此只需求出$\ln\big(1-A(x)\big)$后，再用$O(n\log n)$时间求和即可。总时间为$O(n\log n)$。

9 复合运算

9.1 复合与复合逆

在6.1节中，我们介绍了两个形式幂级数的复合运算。

设$f(w)=\sum_{i\ge1}f_iw^i, g(x)=\sum_{i\ge1}g_ix^i$，如果$f\big(g(x)\big)=x$，那么称$f(w),g(x)$互为复合逆(反函数)。同时我们可以得到$f_1g_1=1, g\big(f(w)\big)=w$。

设$f(w),g(x)$互为复合逆。对于$f(w)$，如果存在一个算法，对任何给定的$p(x)$都高效地求出$f\big(p(x)\big)$，则可利用6.4节中的牛顿迭代法，对于任何给定的$q(w)$求出$g\big(q(w)\big)$。若取$q(w)=w$，即可求出$g(x)$。

给定$f(w),g(x)$，根据定义直接利用FFT计算$f\big(g(x)\big)$的前n项系数，时间复杂度为$O(n^2\log n)$。下面介绍一个用$O(n^2)$时间计算$f\big(g(x)\big)$的算法，它的核心是小步大步思想：

令$d=\lceil\sqrt{n}\,\rceil$，先求出$g(x)^2,g(x)^3,\cdots,g(x)^d$，复杂度为$O(dn\log n)$。再利用已求出的$g(x)^d$，算出$g(x)^{2d},\cdots,g(x)^{(d-1)d}$，复杂度也是$O(dn\log n)$。

接下来，根据

$$f\big(g(x)\big)=\sum_{0\le i<d} g(x)^{id}\sum_{0\le j<d} f_{id+j}g(x)^j, \tag{9.1}$$

并代入之前求得的结果，即可计算。总时间为$O(n\sqrt{n}\log n+n^2)=O(n^2)$。

参考文献[1]中给出了一个时间复杂度为$O\left((n\log n)^{3/2}\right)$的算法计算$f\big(g(x)\big)$，感兴趣的读者可自行参看。

9.2 Lagrange反演

Lagrange反演公式[10]：若$f(w)$是$g(x)$的复合逆，则

$$[x^n]g(x)=\frac{1}{n}[w^{n-1}]\Big(\frac{w}{f(w)}\Big)^n. \tag{9.2}$$

它具有一个推广形式

$$[x^n]h\big(g(x)\big)=\frac{1}{n}[w^{n-1}]h'(w)\Big(\frac{w}{f(w)}\Big)^n, \tag{9.3}$$

其中取$h(w)=w$即为(9.2)式。

求出$g(x)$的复合逆的复杂度较高，但根据(9.2)式与例题4中的方法，可用$O(n\log n)$时间求出复合逆中的某一项系数。

例题11. 给定整数n和集合$S(1\notin S)$，求出含有n个结点，且每个非叶结点u的儿子数量$\deg(u)\in S$的有根多叉树的数量。树的结点无标号，结点的孩子有顺序。$(n\le 10^5)$

令这些树的OGF为$T(x)$。根据定义，一棵树可以是单个叶子结点，或者是一个非叶节点拼上$s(s\in S)$棵子树组成的序列，于是

$$T(x)=x+\sum_{s\in S}T(x)^s, \tag{9.4}$$

可以将其写成$x=f\big(T(x)\big)$的形式，其中$f(w)$是一个一次项系数为1的多项式。

令$g(x)$为$f(w)$的复合逆，则有$T(x)=g(x)$，于是用Lagrange反演求出其n次项系数即可。

[10]证明可参考`http://www.math.msu.edu/%7Emagyar/Math880/Lagrange.pdf`

例题12 (带标号的边双连通图计数). 求出n阶边双连通图[11]的个数。顶点有标号，不允许重边和自环。($n \le 10^5$)

令$C(x)$为所有连通无向图的EGF，$B(x)$为其中所有边双连通图的EGF。例题5中已求出了$C(x)$，现在的任务是计算$B(x)$的x^n项系数。

考虑这样的二元组(G, v)，其中G是一个图，v是G中的某个具有特殊地位的顶点，不妨将此二元组称作有根图。一个图G_0共可产生$|G_0|$个有根图，从而所有连通图产生的有根图的EGF为$xC'(x)$。

给定一个有根图(G_C, v)，可将G_C缩成一棵BCC树T，树上每个结点代表原图中的一个BCC(边双连通分量)。记根结点v所在的BCC为B_v，将B_v定为树T的根。若B_v包含的顶点数量为n，则有根图(B_v, v)的EGF为$nb_nx^n/n!$。

在T中，根B_v可能有若干个(或0个)子树。设T_1是其中某个子树，这个子树也是某个连通图G_1缩成的BCC树。连通图G_1中的某点u与根B_v中的某一点w有连边。有根图(G_1, u)的EGF为$xC'(x)$，又由于点w有$|B_v| = n$种选择，所以一个子树的EGF是$nxC'(x)$，从而若干个子树的EGF就是$e^{nxC'(x)}$。

枚举B_v的大小n，有

$$xC'(x) = \sum_{n \ge 1} \frac{b_n}{(n-1)!} x^n e^{nxC'(x)}. \tag{9.5}$$

简记$C_1(x) = xC'(x), B_1(w) = wB'(w)$，则有

$$C_1(x) = B_1(xe^{C_1(x)}). \tag{9.6}$$

其中，$C_1(x), xe^{C_1(x)}$都是容易计算的。需要计算的是$B_1(w)$的某一项系数。于是问题转化为，已知$P(x), Q(x)$，求出满足$P(x) = R\big(Q(x)\big)$的$R(w)$的n次项系数$[w^n]R(w)$。用$Q^{(-1)}(w)$表示$Q(x)$的复合逆，则有$R(w) = P\big(Q^{(-1)}(w)\big)$。根据Lagrange反演公式的推广形式，得

$$[w^n]R(w) = \frac{1}{n}[x^{n-1}]P'(x)\Big(\frac{x}{Q(x)}\Big)^n,$$

故可用$O(n \log n)$时间计算答案。

[11]删去任意一条边后仍然连通的无向图

10 二元生成函数

有时，我们会在生成函数中引入第二个变量，并用它做一些附加标记。

例题13 (连通图计数II). 求出含有n个顶点，m条边的连通无向图的个数。顶点有标号，不允许重边和自环。$(n, m \le 1000)$

引入字母y来标记边，即用$x^n y^m/n!$表示一个n个点，m条边的图的EGF，则所有无向图的EGF为

$$G(x, y) = \sum_{n \ge 0} \frac{x^n}{n!}(1+y)^{\binom{n}{2}}. \tag{10.1}$$

与例题5类似，需求出$G(x, y)$的对数。这里涉及到了二元多项式的乘法和乘法逆元。

为了将两个二元多项式相乘，可将系数排成矩阵后，对每一行做DFT，再对每一列做DFT，这样便能得到二元多项式的点值表示，然后将点值相乘再转回系数即可。求乘法逆元的过程与一元情况大同小异。总复杂度是$O\big(nm(\log n + \log m)\big)$。

例题14. 给定一个多项式$f(x)$，和一个数k。对于所有$1 \le i \le n$，求出$[x^k]f(x)^i$。$(n, k \le 3000)$

假设$k = O(n)$。利用一个与9.1节中类似的小步大步算法，可以做到$O(n^2)$的复杂度。

现从另一个角度考虑。引入字母u，则所求的答案序列为下式中的x^k项系数：

$$1 + uf(x) + u^2 f(x)^2 + \cdots = \frac{1}{1 - uf(x)}, \tag{10.2}$$

这是一个关于u的幂级数。

为了提取x^k项系数，考虑Lagrange反演。最简单的情况是当$[x^0]f(x) = 0, [x^1]f(x) \neq 0$时，$f(x)$存在复合逆$g(w)$，于是

$$[x^k]\frac{1}{1 - uf(x)} = \frac{1}{k}[w^{k-1}]\frac{u}{(1-uw)^2}\left(\frac{w}{g(w)}\right)^k. \tag{10.3}$$

分母可以用(4.2)展开。由此提示了这个问题与求解$f(x)$的复合逆可在$O(n \log n)$时间内相互转化。

另外，如果$f(x)$不存在复合逆，则还需做一些处理：

- 如果$f(x)$常数项为0，但最低次项$f_t x^t$的次数$t \geq 2$，那么对$f(x)/f_t$开t次方根表示成$f(x) = f_t \cdot \big(p(x)\big)^t$，那么$p(x)$的最低次项次数为1，可以使用上述方法。

- 如果$f(x)$的常数项是$c \neq 0$，则可将其写成$f(x) = c + q(x)$。由此可先求出所有$[x^k]q(x)^i$，再根据

$$\begin{aligned}[x^k]\big(c+q(x)\big)^i &= [x^k]\sum_{0\le j\le i}\binom{i}{j}c^j q(x)^{i-j}\\ &= i!\sum_{0\le j\le i}\frac{c^j}{j!}\frac{[x^k]q(x)^{i-j}}{(i-j)!},\end{aligned}$$

  用FFT求一次卷积即可。

只要能求出$f(x)$的复合逆，就可以再用$O(n\log n)$时间解决本题。一般情况下，求复合逆较难达到优秀的复杂度。但在某些特殊情况下，$f(x)$的复合逆较易求得，此时便可使用这一方法。

例题15 (预处理第一类Stirling数[12]). *给定n，求出所有*$\begin{bmatrix}n\\k\end{bmatrix}, 0 \le k \le n$。($n \le 10^5$)

传统做法是利用

$$\sum_{k=0}^{n}\begin{bmatrix}n\\k\end{bmatrix}x^k = x(x+1)(x+2)\cdots(x+n-1) \tag{10.4}$$

进行分治，每次将因式均等的分成两半分别计算后用FFT乘起来，复杂度是$O(n\log^2 n)$。

另外一个做法利用了

$$\begin{bmatrix}n\\k\end{bmatrix} = \frac{n!}{k!}[x^n]\big(-\ln(1-x)\big)^k. \tag{10.5}$$

根据此式套用刚才的方法即可，复杂度$O(n\log n)$。然而该算法的常数略大。

[12] $\begin{bmatrix}n\\k\end{bmatrix}$表示将$n$个元素划分成$k$个环的方案数量。

11 结语

本文介绍了形式幂级数的乘法逆元、对数、指数等运算的$O(n\log n)$算法，并以串、树、图、集合等各类组合模型为例，在计数问题中加以应用。对于复合逆的求解，本文也介绍了用$O(n\log n)$时间计算其某一项系数的算法。一些原本需要$O(n^2)$或$O(n\log^2 n)$解决的计数问题，可以用本文中的方法优化至$O(n\log n)$，但有时算法的常数较大，在实现时需要注意。

可以预见到，这一类组合计数问题会在今后的信息学竞赛中继续出现。但仍有许多有趣的问题并未在本文中涉及到，还有待我们进一步研究和探索。

感谢

感谢中国计算机学会提供学习和交流的平台。

感谢教练徐先友老师对我的指导。

感谢彭雨翔、吕凯风、毛啸等同学给我的启发。

感谢徐寅展同学为本文审稿。

参考文献

[1] Brent R P, Kung H T. Fast algorithms for manipulating formal power series[J]. Journal of the ACM (JACM), 1978, 25(4): 581-595.

[2] Flajolet P, Sedgewick R. Analytic combinatorics[M]. Cambridge University Press, 2009.

[3] 彭雨翔, Inverse Element of Polynomial.

[4] 彭雨翔, Fast Fourier Transform Modulo Prime.

[5] 刘汝佳, 黄亮, 《算法艺术与信息学竞赛》, 清华大学出版社。