

数学 (2)

长郡中学 周书予

2019 年 11 月 12 日



多项式

由数或字母的积组成的代数式叫单项式。

有限个单项式的和叫多项式。

若不加特殊说明，本课件中的多项式均指一元多项式（只由数字和字母 x 组成）。

一般使用 $A(x) = \sum_{i=0}^n a_i x^i$ 的形式表示一个多项式，定义这个多项式的次数为 n 。

多项式的表示方法

系数表示法: $\sum_{i=0}^n a_i x^i$ 。

点值表示法: 给出 $n+1$ 对 (x_i, y_i) , 其中 x_i 两两不同, 满足 $A(x_i) = y_i$ 。

组合数学中有时会使用下降幂表示法: $\sum_{i=0}^n a_i x^{\underline{i}}$, 这里的 $x^{\underline{i}}$ 表示 $\prod_{j=0}^{i-1} (x - j)$ 。

多项式加法

系数表示法：对应项系数相加，复杂度 $O(n)$ 。

点值表示法：对应点值相加，复杂度 $O(n)$ 。

多项式乘法

系数表示法：设 $A(x) = \sum_{i=0}^n a_i x^i$, $B(x) = \sum_{i=0}^m b_i x^i$, $C(x) = \sum_{i=0}^{n+m} c_i x^i$, 且有 $C(x) = A(x)B(x)$, 那么

$$c_i = \sum_{j+k=i} a_j b_k$$

时间复杂度 $O(nm)$ 。

点值表示法：对应点值相乘，复杂度 $O(n)$ 。

多项式乘法常被称为卷积。

拉格朗日插值

给出 $n + 1$ 个点 (x_i, y_i) , 经过这 $n + 1$ 个点的 n 次多项式是唯一确定的, 为

$$\sum_{i=0}^n y_i \prod_{j=0, j \neq i}^n \frac{x - x_j}{x_i - x_j}$$

已知 $n + 1$ 个点值, 可以 $O(n)$ 求出任意一个其他点值, 也可以 $O(n^2)$ 还原出多项式。

一个有趣的问题

还记得中国剩余定理吗？

考虑 $A(x_i) = y_i$ ，其本质是 $A(x) \bmod (x - x_i) = y_i$ 。

所以，给出 $n + 1$ 对 (x_i, y_i) ，实际上确定的是这个多项式在模 $\prod_{i=0}^n (x - x_i)$ 意义下的结果，由于模的多项式的次数为 $n + 1$ 次，因而可以唯一确定这个 n 次多项式。

在接下来要讲到的 FFT 中，我们将 $\omega_n^0, \omega_n^1, \omega_n^2, \dots, \omega_n^{n-1}$ 作为点值代入，其本质是在模 $\prod_{i=0}^{n-1} (x - \omega_n^i) = x^n - 1$ 意义下做多项式乘法，因而有“FFT 的本质是循环卷积”一说。

多项式求导、求积分

给出 $A(x)$, 求 $A'(x)$, $\int A(x)dx$ 。

令 $A(x) = \sum_{i=0}^n a_i x^i$, 则

$$A'(x) = \sum_{i=1}^n i \times a_i x^{i-1}$$

$$\int A(x) dx = \sum_{i=0}^n \frac{a_i}{i+1} x^{i+1}$$

多项式牛顿迭代

给出 $G(x)$, 求 $F(x)$ 满足 $G(F(x)) = 0 \pmod{x^n}$ 。

考虑倍增, 已知 $G(F_t(x)) = 0 \pmod{x^{2^t}}$, 通过 $F_t(x)$ 求出 $F_{t+1}(x)$ 。

将 $G(F_{t+1}(x))$ 在 $F_t(x)$ 处泰勒展开, 得到

$$G(F_{t+1}(x)) = \sum_{i \geq 0} \frac{G^{(i)}(F_t(x))}{i!} (F_{t+1}(x) - F_t(x))^i$$

当 $i \geq 2$ 时 $(F_{t+1}(x) - F_t(x))^i = 0 \pmod{x^{2^{t+1}}}$, 那么

$$G(F_{t+1}(x)) = G(F_t(x)) + G(F_t(x))'(F_{t+1}(x) - F_t(x)) \pmod{x^{2^{t+1}}}$$

$$F_{t+1}(x) = F_t(x) - \frac{G(F_t(x))}{G(F_t(x))'} \pmod{x^{2^{t+1}}}$$

多项式求逆

给出 $A(x)$, 求 $B(x)$ 满足 $A(x)B(x) = 1 \pmod{x^n}$ 。

套用牛顿迭代

$$\begin{aligned} B_{t+1}(x) &= B_t(x) - \frac{A(x)B_t(x) - 1}{A(x)} \pmod{x^{2^{t+1}}} \\ &= B_t(x) - (A(x)B_t(x) - 1)B_t(x) \pmod{x^{2^{t+1}}} \\ &= B_t(x)(2 - A(x)B_t(x)) \pmod{x^{2^{t+1}}} \end{aligned}$$

时间复杂度 $T(n) = T(\frac{n}{2}) + O(n \log n) = O(n \log n)$ 。

多项式开方

给出 $A(x)$, 求 $B(x)$ 满足 $B^2(x) = A(x) \pmod{x^n}$ 。

套用牛顿迭代

$$\begin{aligned} B_{t+1}(x) &= B_t(x) - \frac{B_t^2(x) - A(x)}{2B_t(x)} \pmod{x^{2^{t+1}}} \\ &= \frac{1}{2} \left(\frac{A(x)}{B_t(x)} + B_t(x) \right) \pmod{x^{2^{t+1}}} \end{aligned}$$

需要实现多项式求逆, 时间复杂度 $T(n) = T(\frac{n}{2}) + O(n \log n) = O(n \log n)$ 。

多项式求对数函数

给出 $A(x)$, 求 $B(x)$ 满足 $\ln A(x) = B(x) \pmod{x^n}$ 。

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

两边同时对 x 求导

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

需要实现多项式求逆、求导、求积分, 时间复杂度 $O(n \log n)$ 。

多项式求指数函数

给出 $A(x)$, 求 $B(x)$ 满足 $e^{A(x)} = B(x) \pmod{x^n}$ 即 $\ln B(x) = A(x) \pmod{x^n}$ 。
套用牛顿迭代

$$\begin{aligned} B_{t+1}(x) &= B_t(x) - \frac{\ln B_t(x) - A(x)}{\frac{1}{B_t(x)}} \pmod{x^{2^{t+1}}} \\ &= B_t(x)(1 - \ln B_t(x) + A(x)) \pmod{x^{2^{t+1}}} \end{aligned}$$

需要实现多项式求对数函数, 时间复杂度 $T(n) = T(\frac{n}{2}) + O(n \log n) = O(n \log n)$ 。

多项式快速幂

给出 $A(x)$, k , 求 $B(x)$ 满足 $A^k(x) = B(x) \pmod{x^n}$ 。

$$A^k(x) = (e^{\ln A(x)})^k = e^{k \ln A(x)}$$

需要先将 $A(x)$ 化为常数项为 1。

多项式除法（取模）

给出 $A(x), B(x)$, 求 $C(x), D(x)$ 满足 $A(x) = B(x)C(x) + D(x)$, 且 $D(x)$ 的次数严格小于 $B(x)$ 。设 $A(x)$ 的次数为 n , $B(x)$ 的次数为 m , 那么 $C(x)$ 的次数显然为 $n - m$, $D(x)$ 的次数不超过 $m - 1$ 。

定义 $A^R(x) = x^n A(\frac{1}{x})$, $B^R(x) = x^m B(\frac{1}{x})$, $C^R(x) = x^{n-m} C(\frac{1}{x})$, $D^R(x) = x^{m-1} D(\frac{1}{x})$, 那么

$$x^n A(\frac{1}{x}) = x^m B(\frac{1}{x}) \times x^{n-m} C(\frac{1}{x}) + x^n D(x)$$

$$A^R(x) = B^R(x) C^R(x) + x^{n-m+1} D(x)$$

$$C^R(x) = \frac{A^R(x)}{B^R(x)} \mod x^{n-m+1}$$

求出 $C(x)$ 后 $D(x)$ 也就容易求了, 时间复杂度 $O(n \log n)$ 。

多项式多点求值

给出 $A(x)$ 和 m 个 x_i , 需要对每个 x_i 求出 $A(x_i)$ 。

求 $A(x)$ 在 x_i 处的点值即求 $A(x) \bmod (x - x_i)$, 考虑分治, 维护 $L(x) = \prod_{i=l}^{mid} (x - x_i)$ 和 $R(x) = \prod_{i=mid+1}^r (x - x_i)$ 并求出 $A(x) \bmod L(x)$ 和 $A(x) \bmod R(x)$, 向左右分别递归即可。

时间复杂度 $T(n) = 2T(\frac{n}{2}) + O(n \log n) = O(n \log^2 n)$ 。

多项式多点插值

给出 $n+1$ 个点 (x_i, y_i) , 求一个 n 次多项式 $A(x)$ 满足 $A(x_i) = y_i$ 。

利用拉格朗日插值, 这个多项式是 $\sum_{i=0}^n y_i \prod_{j=0, j \neq i}^n \frac{x-x_j}{x_i-x_j}$ 。

先考虑对每个 i 求出 $v_i = y_i \prod_{j=0, j \neq i}^n \frac{1}{x_i-x_j}$, 令 $P(x) = \prod_{j=0}^n (x-x_j)$, 那么

$\prod_{j=0, j \neq i}^n (x_i-x_j)$ 就是求 $\frac{M(x)}{x-x_i}$ 在 $x=x_i$ 处的点值, 根据洛必达法则, 这个东西是 $M'(x_i)$ 。

通过一次多点求值求出 v_i 后, 要求的是 $\sum_{i=0}^n v_i \prod_{j=0, j \neq i}^n (x-x_j)$ 。

分治, 设 $L(x) = \prod_{i=l}^{mid} (x-x_i) = \sum_{i=0}^{mid-l+1} l_i x^i$, $R(x) = \prod_{i=mid+1}^r (x-x_i) = \sum_{i=0}^{r-mid} r_i x^i$,

那么答案就是 $R(x)(\sum_{i=l}^{mid} v_i \prod_{j=l, j \neq i}^{mid} (x-x_j)) + L(x)(\sum_{i=mid+1}^r v_i \prod_{j=mid+1, j \neq i}^r (x-x_j))$,

将 v_i 数组对应与 $L(x), R(x)$ 做减法卷积后可以得到与原答案式相同的形式, 递归即可,

时间复杂度 $T(n) = 2T(\frac{n}{2}) + O(n \log n) = O(n \log^2 n)$ 。

单位根

方程 $x^n = 1$ 在复数域 \mathbb{C} 下的 n 个解 $\omega_n^0, \omega_n^1, \omega_n^2, \dots, \omega_n^{n-1}$ 称为 n 次单位根。

$\omega_n^k = \cos \frac{2k\pi}{n} + \sin \frac{2k\pi}{n} i$, n 个 n 次单位根分布在复平面的单位圆上, 并将其 n 等分。

单位根满足一些性质:

- $\omega_{nd}^{kd} = \omega_n^k$
- $\omega_n^{k+n/2} = -\omega_n^k$
- $\sum_{i=0}^{n-1} \omega_n^i = 0$
- $\sum_{i=0}^{n-1} \omega_n^{ik} = n \times [k \bmod n = 0]$

离散傅里叶变换

将多项式的系数表示转换成单位根点值表示的变换称为离散傅里叶变换 (DFT)。
与之相对的，将单位根点值表示转换成多项式系数表示的变换称为逆离散傅里叶变换 (IDFT)。
常使用快速傅里叶变换 (FFT) 来实现 DFT 以及 IDFT。

快速傅里叶变换

假设 n 是 2 的幂次, 有一个 $n-1$ 次多项式 $A(x) = \sum_{i=0}^{n-1} a_i x^i$, 取 $\omega_n^0, \omega_n^1, \dots, \omega_n^{n-1}$ 作为点值代入后得到其点值表示。

考虑将多项式 $A(x) = \sum_{i=0}^{n-1} a_i x^i$ 拆成两个多项式:

$$A_0(x) = \sum_{i=0}^{n/2-1} a_{2i} x^i, A_1(x) = \sum_{i=0}^{n/2-1} a_{2i+1} x^i$$

于是 $A(x) = A_0(x^2) + A_1(x^2)x$, 对 $A_0(x), A_1(x)$ 递归求 $\omega_{n/2}^0, \omega_{n/2}^1, \dots, \omega_{n/2}^{n/2-1}$ 的点值, 有:

$$A(\omega_n^k) = A_0(\omega_{n/2}^k) + \omega_n^k A_1(\omega_{n/2}^k)$$

$$A(\omega_n^{k+n/2}) = A_0(\omega_{n/2}^k) - \omega_n^k A_1(\omega_{n/2}^k)$$

时间复杂度 $T(n) = 2T(\frac{n}{2}) + O(n) = O(n \log n)$ 。

建议对快速傅里叶变换掌握其数学原理以及推导过程, 然后背熟模板。

逆离散傅里叶变换

DFT 的本质是给出了一个长度为 n 的数组 $\{a_0, a_1, \dots, a_{n-1}\}$, 求出了数组 $\{b_0, b_1, \dots, b_{n-1}\}$ 满足

$$b_i = \sum_{j=0}^{n-1} a_j \omega_n^{ij}$$

对数组 $\{b_0, b_1, \dots, b_{n-1}\}$ 再做一次 DFT, 得到数组 $\{c_0, c_1, \dots, c_{n-1}\}$, 满足

$$c_i = \sum_{j=0}^{n-1} b_j \omega_n^{ij} = \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} a_k \omega_n^{jk} \omega_n^{ij} = \sum_{k=0}^{n-1} a_k \sum_{j=0}^{n-1} \omega_n^{j(i+k)} = \sum_{k=0}^{n-1} a_k \times n \times [n | i+k] = na_{-i \bmod n}$$

因而只需要在做完 DFT 后执行 `reverse(a+1,a+n)`, 再全部除以 n 就实现了 IDFT。

快速数论变换

在模质数 p 意义下做 DFT，使用 p 的原根 g 代替单位根 ω_{p-1} ，可以发现原根仍满足单位根的所有性质。

FFT 使用的是 2 的幂次单位根，设 $p = 2^t \times r + 1$ ，用 $g^{\frac{p-1}{2^t}}$ 代替 ω_{2^t} 做 DFT 的过程称为快速数论变换 (NTT)。

$998244353 = 2^{23} \times 119 + 1$ 是一个常见的 NTT 模数，它的原根是 3。

Bluestein's Algorithm

用于解决任意长度的 DFT。考虑 DFT 的本质

$$b_i = \sum_{j=0}^{n-1} a_j \omega_n^{ij}$$

利用

$$ij = \binom{i+j}{2} - \binom{i}{2} - \binom{j}{2}$$

于是有

$$b_i \omega_n^{\binom{i}{2}} = \sum_{j=0}^{n-1} a_j \frac{\omega_n^{\binom{i+j}{2}}}{\omega_n^{\binom{j}{2}}}$$

从而用卷积实现了 DFT。可以用来出毒瘤题。

形式幂级数

简单地来说，形式幂级数就是无穷项的多项式。

形式幂级数中， x 仅仅是一个符号，而不用代入具体数值运算，因此不需要考虑幂级数的敛散性。

普通生成函数

数列 $\{a_0, a_1, \dots\}$ 的普通生成函数 (OGF) 定义为形式幂级数

$$A(x) = \sum_{i \geq 0} a_i x^i$$




例如, 数列 $\{1, 1, \dots\}$ 的普通生成函数为 $\sum_{i \geq 0} x^i = \frac{1}{1-x}$ 。

普通生成函数的乘法 $C(x) = A(x)B(x)$ 的意义是

$$c_i = \sum_{j+k=i} a_j b_k$$

一个使用普通生成函数的例子

description




从前有一个 ，他非常地爱滚，并且掌握了很多种不同的滚法。已知  有 a_i 种滚法可以在 1 秒内滚 i 米，求  总共滚了 n 米的方案数模 998244353。

constriction

$$1 \leq n \leq 5 \times 10^5.$$

一个使用普通生成函数的例子

description

从前有一个，他非常地爱滚，并且掌握了很多种不同的滚法。已知有 a_i 种滚法可以在 1 秒内滚 i 米，求总共滚了 n 米的方案数模 998244353。

constriction

$$1 \leq n \leq 5 \times 10^5.$$

solution

设 $\{a_i\}$ 的普通生成函数为 $A(x)$ ，答案的普通生成函数为 $F(x)$ ，那么有 $F(x) = F(x)A(x) + 1$ （或者你也可以直接 $F(x) = \sum_{i \geq 0} A^i(x)$ ），即 $F(x) = \frac{1}{1-A(x)}$ ，使用多项式求逆可以在 $O(n \log n)$ 的时间复杂度内解决。

指数生成函数

数列 a_0, a_1, \dots 的指数生成函数 (EGF) 定义为形式幂级数

$$A(x) = \sum_{i \geq 0} \frac{a_i}{i!} x^i$$

例如, 数列 $\{1, 1, \dots\}$ 的指数生成函数为 $\sum_{i \geq 0} \frac{x^i}{i!} = e^x$ 。

指数生成函数的乘法 $C(x) = A(x)B(x)$ 的意义是

$$\frac{c_i}{i!} = \sum_{j+k=i} \frac{a_j b_k}{j! k!}, \quad c_i = \sum_{j+k=i} \binom{i}{j} a_j b_k$$

一个使用指数生成函数的例子

description

求 n 个点有标号无向连通图个数模 998244353。

constriction

$1 \leq n \leq 10^5$.

一个使用指数生成函数的例子

description

求 n 个点有标号无向连通图个数模 998244353。

constriction

$$1 \leq n \leq 10^5.$$

solution

记 $F(x)$ 表示答案的指数生成函数, $G(x)$ 表示有标号无向图数量的指数生成函数, 显然 $G(x) = \sum_{i \geq 0} 2^{\binom{i}{2}} \frac{x^i}{i!}$ 。考虑一张有标号无向图是由若干有标号无向连通图组成的, 枚举连通块个数, 可以得到

$$G(x) = \sum_{i \geq 0} \frac{(F(x) - 1)^i}{i!} = e^{F(x)-1}, F(x) = \ln G(x) + 1$$

使用多项式求对数函数可以在 $O(n \log n)$ 的时间复杂度内解决。

完全背包计数

description

大小为 i 的物品有 a_i 种，每种物品有无限个，求装满大小为 n 的背包的方案数模 998244353。

constriction

$$1 \leq n \leq 10^5.$$

完全背包计数

description

大小为 i 的物品有 a_i 种，每种物品有无限个，求装满大小为 n 的背包的方案数模 998244353。

constriction

$$1 \leq n \leq 10^5.$$

solution

设 $\{a_i\}$ 的普通生成函数为 $A(x)$ ，易得答案的生成函数为

$$\prod_{i=1}^n \left(\frac{1}{1-x^i} \right)^{a_i} = \exp\left(-\sum_{i=1}^n a_i \ln(1-x^i)\right) = \exp\left(-\sum_{i=1}^n a_i \sum_{j \geq 1} \frac{x^{ij}}{j}\right) = \exp\left(\sum_{j \geq 1} \frac{1}{j} A(x^j)\right)$$

$A(x^j)$ 中只有 n/j 项可用，因此可以 $O(n \ln n)$ 地求出 $\sum_{j \geq 1} \frac{1}{j} A(x^j)$ 后再 \exp 。

一个计数技巧

$$n^k = \sum_{0 \leq a_i \leq k, \sum_{i=1}^n a_i = k} \frac{k!}{\prod_{i=1}^n a_i!}$$

即， k 个球染 n 种颜色的方案数，等于枚举每种颜色的球有多少个后可重排列的方案数。

$$n^k = k! [x^k] e^{nx}$$

是其生成函数表示。好像是句废话这实现了 n 从底数到指数的转化。

loj6343 Sally Face 与地牢

description

一张 n 点 m 条边的无向图， q 次询问所有从 x 走到 y 长度 $\leq k$ 的路径长度的 r 次方和。

constriction

$1 \leq n \leq 5, 0 \leq r \leq 1000, 1 \leq k \leq 10^9$.

loj6343 Sally Face 与地牢

description

一张 n 点 m 条边的无向图， q 次询问所有从 x 走到 y 长度 $\leq k$ 的路径长度的 r 次方和。

constriction

$1 \leq n \leq 5, 0 \leq r \leq 1000, 1 \leq k \leq 10^9$.

solution

定义一个以多项式为元素的矩阵 A ，图中每存在一条边 (x, y) 就令 $A_{x,y} \leftarrow A_{x,y} + e^x$ ，答案就是 $r! [x^r] \sum_{i=0}^k A_{x,y}^i$ ，倍增求即可，时间复杂度 $O((n^2 r \log r + n^3 r) \log k)$ 。

概率生成函数

对于数列 $\{a_0, a_1, \dots\}$, 如果存在某个随机变量 X 满足 $\Pr(X = i) = a_i$, 那么 $\{a_0, a_1, \dots\}$ 的普通生成函数 $A(x) = \sum_{i \geq 0} a_i x^i$ 被称为 X 的概率生成函数。

显然任意的概率生成函数 $A(x)$ 均满足 $A(1) = \sum_{i \geq 0} \Pr(X = i) = 1$ 。

对 $A(x)$ 求导, 得到 $A'(x) = \sum_{i \geq 1} i \Pr(X = i) x^{i-1}$, 有 $A'(1) = \sum_{i \geq 1} i \Pr(X = i) = E(X)$ 。

进一步推导可以得到 $A^{(k)}(1) = \sum_{i \geq k} i^k \Pr(X = i) = E(X^k)$ 。

ctsc2006 歌唱王国

给出一个长为 n 的字符串 S ，字符集大小为 σ 。有一个字符序列，不停在尾部加入随机字符，直到 S 在序列中出现为止。求停止时序列的期望长度。

ctsc2006 歌唱王国

给出一个长为 n 的字符串 S ，字符集大小为 σ 。有一个字符序列，不停在尾部加入随机字符，直到 S 在序列中出现为止。求停止时序列的期望长度。

设 f_i 表示长度为 i 时恰好停止的概率， g_i 表示长度为 i 时尚未停止的概率， $F(x)$, $G(x)$ 分别为二者的生成函数。我们可以得到如下两个式子：

$$1 + G(x)x = F(x) + G(x)$$

$$G(x)\left(\frac{x}{\sigma}\right)^n = \sum_{i=1}^n a_i \left(\frac{x}{\sigma}\right)^{n-i} F(x)$$

第一个式子表示在未结束时加入一个随机字符，可能结束也可能没结束。第二个式子表示在未结束时加入串 S ，那么一定结束，只不过可能只加了 S 的某个前缀就已经结束了。这里的 a_i 表示的是串 S 是否存在长度为 i 的 border。

ctsc2006 歌唱王国

要求的是 $F'(1)$ 。

$$\begin{aligned}1 + G(x)x &= F(x) + G(x) \\ G'(x)x + G(x) &= F'(x) + G'(x) \\ F'(1) &= G(1)\end{aligned}$$

将 $x = 1$ 代入第二个式子可得

$$F'(1) = G(1) = \sum_{i=1}^n a_i \sigma^i$$

线性递推

给出递推式的前 m 项以及递推式 $f_i = \sum_{j=1}^m c_j f_{i-j}$, 求 f_n 。

朴素做法是设一个行向量 $[f_1 \ f_2 \ \cdots \ f_m]$ 以及一个转移矩阵

$$A = \begin{bmatrix} 0 & 0 & 0 & \cdots & c_m \\ 1 & 0 & 0 & \cdots & c_{m-1} \\ 0 & 1 & 0 & \cdots & c_{m-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & c_1 \end{bmatrix}$$

满足 $[f_1 \ f_2 \ \cdots \ f_m] A^n = [f_{n+1} \ f_{n+2} \ \cdots \ f_{n+m}]$, 矩阵乘法即可。

进一步的, 考虑求出 A 的特征多项式 $p(\lambda) = \lambda^m - \sum_{i=1}^m c_i \lambda^{m-i}$, 根据 Cayley-Hamilton 定理, 有 $p(A) = 0$, 也即

$$A^m = \sum_{i=1}^m c_i A^{m-i}$$

线性递推

$$A^m = \sum_{i=1}^m c_i A^{m-i}$$

这说明了任意 A^n 均可以用 $A^0, A^1, A^2, \dots, A^{m-1}$ 线性表示, 不妨记 $A^n = \sum_{i=0}^{m-1} b_{n,i} A^i$, 左右同左乘行向量 $[f_1 \ f_2 \ \cdots \ f_m]$ 后得

$$[f_{n+1} \ f_{n+2} \ \cdots \ f_{n+m}] = \sum_{i=0}^{m-1} b_{n,i} [f_{i+1} \ f_{i+2} \ \cdots \ f_{i+m}]$$

可以 $O(m)$ 求出其中一项或 $O(m \log m)$ 求出每一项。

记 $B_n(x) = \sum_{i=0}^{m-1} b_{n,i} x^i$, 那么 $B_{n+m}(x) = B_n(x) B_m(x) \bmod (x^m - \sum_{i=1}^m c_i x^{m-i})$, 倍增 + 多项式取模即可, 可以实现 $O(m^2 \log n)$ 或 $O(m \log m \log n)$ 的复杂度。

小技巧

$$\left(\sum_{i=0}^n a_i x^i\right)^p = \sum_{i=0}^n a_i x^{ip} \pmod{p}$$

其中 p 是质数。

证明的话，考虑有 p 个多项式 $\sum_{i=0}^n a_i x^i$ ，需要从每个多项式中选一项乘起来，设 $a_i x^i$ 项被选了 b_i 次 ($\sum_{i=0}^n b_i = p$)，那么这种组合的系数是

$$\frac{p!}{\prod_{i=0}^n b_i!} \pmod{p}$$

由于 p 是质数，因此当且仅当存在某个 $b_i = p$ 时，系数 $= 1$ ，否则系数 $= 0$ 。

第一类斯特林数

定义第一类斯特林数 $\left[\begin{smallmatrix} n \\ m \end{smallmatrix} \right]$ 表示 n 个不同元素构成 m 个非空圆排列的方案数。

经典递推: $\left[\begin{smallmatrix} n \\ m \end{smallmatrix} \right] = \left[\begin{smallmatrix} n-1 \\ m-1 \end{smallmatrix} \right] + (n-1) \left[\begin{smallmatrix} n-1 \\ m \end{smallmatrix} \right]$

求一行: $\left[\begin{smallmatrix} n \\ m \end{smallmatrix} \right] = [x^m] \prod_{i=0}^{n-1} (x+i)$

求一列: $\left[\begin{smallmatrix} n \\ m \end{smallmatrix} \right] = \frac{n!}{m!} [x^n] (\sum_{i \geq 1} (i-1)! \frac{x^i}{i!})^m = \frac{n!}{m!} [x^n] (\sum_{i \geq 1} \frac{x^i}{i})^m = \frac{n!}{m!} [x^n] \ln^m(1-x)$

第二类斯特林数

定义第二类斯特林数 $\left\{ \begin{smallmatrix} n \\ m \end{smallmatrix} \right\}$ 表示 n 个不同元素构成 m 个非空集合的方案数。

经典递推: $\left\{ \begin{smallmatrix} n \\ m \end{smallmatrix} \right\} = \left\{ \begin{smallmatrix} n-1 \\ m-1 \end{smallmatrix} \right\} + m \left\{ \begin{smallmatrix} n-1 \\ m \end{smallmatrix} \right\}$

求一行: $\left\{ \begin{smallmatrix} n \\ m \end{smallmatrix} \right\} = \frac{1}{m!} \sum_{i=0}^m (-1)^k \binom{m}{k} (m-k)^n$, 本质是在容斥“非空”这个条件。

求一列: $\left\{ \begin{smallmatrix} n \\ m \end{smallmatrix} \right\} = \frac{n!}{m!} [x^n] (e^x - 1)^m$

斯特林反演

考虑两类斯特林数的一些性质，我们有

$$x^n = \sum_m (-1)^{n-m} \begin{bmatrix} n \\ m \end{bmatrix} x^m, x^n = \sum_m \left\{ \begin{matrix} n \\ m \end{matrix} \right\} x^m$$

把两个式子互相带入可得

$$\sum_k (-1)^{n-m} \begin{bmatrix} n \\ m \end{bmatrix} \left\{ \begin{matrix} m \\ k \end{matrix} \right\} = \sum_k (-1)^{m-k} \left\{ \begin{matrix} n \\ m \end{matrix} \right\} \begin{bmatrix} m \\ k \end{bmatrix} = 0$$

第二类斯特林数求一列的另一种方法

之前用的是指数生成函数。考虑用普通生成函数，设 $F_m(x)$ 表示第二类斯特林数第 m 列的普通生成函数

$$\left\{ \begin{matrix} n \\ m \end{matrix} \right\} = \left\{ \begin{matrix} n-1 \\ m-1 \end{matrix} \right\} + m \left\{ \begin{matrix} n-1 \\ m \end{matrix} \right\} \Leftrightarrow F_m(x) = xF_{m-1}(x) + mx F_m(x) \Leftrightarrow F_m = \frac{x}{1-mx} F_{m-1}(x)$$

显然 $F_0(x) = 1$ ，所以就有

$$F_m(x) = \frac{x^m}{\prod_{i=1}^m (1 - ix)}$$

分母的部分很像第一类斯特林数的行生成函数，从中可以窥探到斯特林反演的端倪。

斯特林数的一些应用

除了定义之外，斯特林数还有一些这样的用法：

$$n^k = \sum_{i=0}^k \left\{ \begin{matrix} k \\ i \end{matrix} \right\} i! \binom{n}{i}$$

可以把求 n^k 转化为求 $\binom{n}{0}, \binom{n}{1}, \dots, \binom{n}{k}$ 。众所周知，组合数的转移是 $O(1)$ 而非 $O(k)$ 的。斯特林数还可以用来求自然数幂和。

$$\begin{aligned} \sum_{i=0}^n i^k &= \sum_{i=0}^n \sum_{j=0}^k \left\{ \begin{matrix} k \\ j \end{matrix} \right\} j! \binom{i}{j} = \sum_{j=0}^k \left\{ \begin{matrix} k \\ j \end{matrix} \right\} j! \sum_{i=0}^n \binom{i}{j} \\ &= \sum_{j=0}^k \left\{ \begin{matrix} k \\ j \end{matrix} \right\} j! \binom{n+1}{j+1} = \sum_{j=0}^k \left\{ \begin{matrix} k \\ j \end{matrix} \right\} \frac{(n+1)^{\underline{j+1}}}{j} \end{aligned}$$

伯努利数

定义伯努利数的指数生成函数为

$$B(x) = \sum_{i \geq 0} b_i \frac{x^i}{i!} = \frac{x}{e^x - 1}$$

一个性质是 $\sum_{i=0}^n b_i \binom{n+1}{i} = [n=0]$, 可以直接拿生成函数的定义来证。

伯努利数求自然数幂和

设自然数幂和的指数生成函数 $A_n(x) = \sum_{i \geq 0} (\sum_{j=0}^n j^i) \frac{x^i}{i!}$, 则

$$\begin{aligned} A_n(x) &= \sum_{j=0}^n \sum_{i \geq 0} \frac{(jx)^i}{i!} = \sum_{j=0}^n e^{jx} \\ &= \frac{1 - e^{(n+1)x}}{1 - e^x} = B(x) \frac{e^{(n+1)x} - 1}{x} \\ &= B(x) \left(\sum_{i \geq 0} \frac{(n+1)^{i+1}}{(i+1)!} x^i \right) \end{aligned}$$

于是有

$$\sum_{i=0}^n i^k = k! A_n(x) [x^k] = \frac{1}{k+1} \sum_{i=0}^k \binom{k+1}{i} b_i (n+1)^{k+1-i}$$

杜教筛

假设要求前缀和的积性函数为 $f(n)$ ，记之前缀和为 $S(n) = \sum_{i=1}^n f(i)$ 。
构造 Dirichlet 卷积 $h = f * g$ 。

$$\begin{aligned} h(i) &= \sum_{d|i} f\left(\frac{i}{d}\right) g(d) \\ \sum_{i=1}^n h(i) &= \sum_{i=1}^n \sum_{d|i} f\left(\frac{i}{d}\right) g(d) \\ &= \sum_{d=1}^n g(d) \sum_{d|i} f\left(\frac{i}{d}\right) \\ &= \sum_{d=1}^n g(d) S\left(\left\lfloor \frac{n}{d} \right\rfloor\right) \end{aligned}$$

于是有 $g(1)S(n) = \sum_{i=1}^n h(i) - \sum_{d=2}^n g(d)S\left(\left\lfloor \frac{n}{d} \right\rfloor\right)$ 。

杜教筛的时间复杂度证明

可以发现在求 $S(n)$ 的过程中，只会递归调用到 $S(\lfloor \frac{n}{d} \rfloor)$ ，同时还需要 $O(\sqrt{n})$ 的复杂度进行累加（这里认为 g 和 h 均可以在 $O(1)$ 的时间内求任意前缀和），因此若实现记忆化，时间复杂度大约是

$$\sum_{x=1}^{\sqrt{n}} O(\sqrt{x}) + O(\sqrt{\frac{n}{x}})$$

积分可知时间复杂度为 $O(n^{\frac{3}{4}})$ 。

考虑加以预处理，假设预处理出前 k 项的前缀和，则时间复杂度为

$$O(k) + \sum_{x=1}^{\frac{n}{k}} O(\sqrt{\frac{n}{x}})$$

积分可知时间复杂度为 $O(k + \frac{n}{\sqrt{k}})$ ，当 k 取 $O(n^{\frac{2}{3}})$ 时取到最优时间复杂度 $O(n^{\frac{2}{3}})$ 。

Min_25 筛

筛法的过程大致分为两步：先对所有 $x = \lfloor \frac{n}{d} \rfloor$ 筛出不超过 x 的所有质数的 f 值之和，再求出剩下的合数的 f 值之和。定义 $g(i)$ 为所有数按照 $f(i)$ 在 i 为质数时的式子算出来的结果，并记 pri_j 表示第 j 小的质数。

设 $G(i, j)$ 表示 i 以内所有质数以及最小质因子 $> pri_j$ 的合数的 g 之和。从小到大枚举质因子 p ，把最小质因子恰为 pri_j 的合数去掉，这要求能在已知 $\sum g(i)$ 的前提下快速计算 $\sum g(i \times pri_j)$ 。枚举 $\geq pri_j^2$ 的所有 i ， $G(i, j)$ 需要去掉的部分是 $G(\lfloor \frac{i}{pri_j} \rfloor, j-1) - sum_{j-1}$ 并把每个数从 $g(i)$ 变成 $g(i \times pri_j)$ ，其中 sum_j 表示前 j 个质数的 f 值之和。

接下来就用所有质数的 f 来求出剩下的合数的 f 。设 $F(i, j)$ 表示 i 以内所有质数以及最小质因子 $> pri_j$ 的合数的 f 之和。从大到小枚举质因子 pri_j ，枚举 $\geq pri_j^2$ 的所有 i ，再枚举一个 $pri_j^{k+1} \leq i$ 的 k ，把 $F(\lfloor \frac{i}{pri_j^k} \rfloor, j) \times f(pri_j^k) + f(pri_j^{k+1})$ 累加进 $F(i, j-1)$ 。

最终可以实现对所有 $x = \lfloor \frac{n}{d} \rfloor$ 筛出不超过 x 的 f 值之和。

Min_25 筛的时间复杂度证明

对于一个数 x , 满足 $p^2 \leq x$ 的质数 p 的个数是 $O(\frac{\sqrt{x}}{\log \sqrt{x}})$ 级别的。

只考虑 Min_25 筛中前一部分的时间复杂度。类似杜教筛, 由于只需要求所有 $\lfloor \frac{n}{d} \rfloor$ 处的前缀和, 总时间复杂度是

$$\sum_{x=1}^{\sqrt{n}} O\left(\frac{\sqrt{x}}{\log \sqrt{x}}\right) + O\left(\frac{\sqrt{\frac{n}{x}}}{\log \sqrt{\frac{n}{x}}}\right)$$

所有的 \log 都是同级的, 提出后复杂度就不加预处理的杜教筛完全一样, 因此时间复杂度为 $O(\frac{n^{\frac{3}{4}}}{\log n})$ 。

后一部分需要额外枚举质数 p 的出现次数 k , 复杂度 (大概/也许/可能) 不会差太多。

行列式

$$\det(A) = \begin{vmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{vmatrix} = \sum_p (-1)^{\tau(p)} \prod_{i=1}^n a_{i,p_i}$$

其中 $\tau(p)$ 表示排列 p 的逆序对数。

矩阵树定理

定义图 G 的基尔霍夫矩阵为其度数矩阵减去邻接矩阵。

该矩阵任意一个主子式（去掉第 i 行第 i 列后剩下的 $n - 1$ 阶行列式的值）即为图 G 的生成树个数。

实际上求出来的是所有生成树的边权乘积之和。

如果要求的是有根树，生成树个数为去掉根所在的那一行一列后的主子式。

矩阵元素不一定要求是一个数，还可以是多项式，只要保证有逆就行了。

证明咕咕咕了。

线性基

基本上能够理解线性相关的概念就可以了。

讲一下带删除线性基的实现。对线性基中的每个向量维护它是由原本的那些向量组成的，删除向量 x 时先找到包含 x 的一个零向量，将其异或到其他所有包含 x 的向量上（不改变数值，只改变了向量组成）再直接删除即可。若不存在包含 x 的零向量，则删除 x 势必会导致秩减小，找到包含 x 的最低位向量，同样异或到其他所有包含 x 的向量上即可。模板题可以去做【集训队互测 2015】最大异或和。

集合幂级数

类比于生成函数，集合幂级数是解决一些集合计数问题的常用手段与利器。
比较学术严谨的定义这里不会讲，可以去参考吕凯风的 2015 年候选队论文《集合幂级数的性质与应用及其快速算法》。
主要需要掌握的算法有集合与/或卷积、集合异或卷积、子集卷积。

集合与/或卷积

$$c_i = \sum_{j \wedge k = i} a_j b_k$$

$$c_i = \sum_{j \vee k = i} a_j b_k$$

以上两者为集合与/或卷积。

以集合或卷积为例，定义 $a'_i = \sum_{j \subseteq i} a_j$ ，类似定义 b'_i, c'_i ，可以发现 c'_i 数组就是 a'_i 与 b'_i 的点积。这些做法有时候会被称为快速莫比乌斯变换 (FMT) 或是快速子集变换 (FST)。

时间复杂度 $O(n2^n)$ 。

本质是高维前/后缀和。

集合异或卷积

$$c_i = \sum_{j \oplus k = i} a_j b_k$$

可以把集合异或卷积视作每一维长度为 2 的高维循环卷积。因此构造快速沃尔什变换 (FWT) 时, 只需要对每一维做一个长度为 2 的 DFT 就行了。时间复杂度 $O(n2^n)$ 。这种理解方式可以很容易扩展到 k 进制 FWT。

子集卷积

$$c_i = \sum_{j \vee k = i, j \wedge k = \emptyset} a_j b_k$$

相当于是在集合或卷积的基础上加上了交集为空的限制，这个限制实际上可以视作 $|j| + |k| = |i|$ 。把每个 a_i 乘上一个 $x^{|i|}$ ，即每个元素是一个多项式，这样套用集合或卷积的做法，加以 $O(n^2)$ 的多项式乘法，即可在 $O(n^2 2^n)$ 的复杂度内解决子集卷积。这里的多项式实际上起到了占位的效果，因而有时这个多项式会被称为占位多项式。

谢谢大家！

感谢👤、👤、👤以及👤的友情出镜。