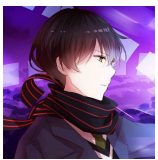
多项式

租酥雨

2021 年 7 月 10 日



outline

- 多项式
- 快速傅里叶变换
- 生成函数
- 集合幂级数

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

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 $F(x_i) = y_i$。

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

(组合数 $\binom{n}{m}$ 其实是关于 n 的一个 m 次多项式。)

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。

Lagrange 插值

给出 $n+1$ 个点 (x_i, y_i)，其中 x_i 互不相同，那么经过这 $n+1$ 个点的 n 次多项式是存在且唯一的：

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

(唯一性：一个存在至少 $n+1$ 个零点的不超过 n 次多项式一定是零多项式。)
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 $\omega_n^0, \omega_n^1, \omega_n^2, ...\omega_n^{n-1}$ 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 本质是循环卷积”一说。

多项式求导、求积分

给出 $A(x)$，求 $A'(x), \int A(x)\mathrm{d}x$。

令 $A(x)=\sum_{i=0}^{n} a_i x^i$，则

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

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

多项式牛顿迭代

给出 $G(x)$，求 $F(x)$ 满足 $G(F(x)) = 0 \mod x^n$。
考虑倍增，已知 $G(F_t(x)) = 0 \mod x^{2^t}$，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 $(F_{t+1}(x) - F_t(x))^i = 0 \mod x^{2^{t+1}}$，那么

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

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

多项式求逆

给出 $A(x)$，求 $B(x)$ 满足 $A(x)B(x) = 1 \mod x^n$。
套用牛顿迭代

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

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 $B^2(x) = A(x) \mod x^n$。
套用牛顿迭代

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

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 $\ln A(x) = B(x) \mod x^n$。

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

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 $e^{A(x)} = B(x) \mod x^n$ 即 $\ln B(x) = A(x) \mod x^n$。
套用牛顿迭代

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

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 $A^k(x) = B(x) \mod x^n$。

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

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 $\deg D(x) < \deg B(x)$。
设 $\deg A(x) = n, \deg B(x) = m$, 则 $\deg C(x) = n - m, \deg D(x) \le m - 1$。记
$A^R(x) = x^n A(\frac{1}{x}), B^R(x) = x^m B(\frac{1}{x}), C^R(x) = x^{n-m} C(\frac{1}{x}), D^R(x) = x^{m-1} D(\frac{1}{x})$，那么

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

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

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

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

求 $F(x)$ 在 x_i 处的点值即求 $F(x) \mod (x-x_i)$，考虑分治，维护 $L(x)=\prod_{i=l}^{mid}(x-x_i)$ 和 $R(x)=\prod_{i=mid+1}^{r}(x-x_i)$，求出 $F(x) \mod L(x)$ 和 $F(x) \mod R(x)$ 后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 $F(x)$ 满足 $F(x_i) = y_i$。

根据 Lagrange 插值公式，这个多项式是 $\sum_{i=0}^{n} y_i \prod_{j\neq i} \frac{x-x_j}{x_i-x_j}$.

先考虑对每个 i 求出 $v_i = y_i \prod_{j\neq i} \frac{1}{x_i-x_j}$，令 $P(x) = \prod_{j=0}^{n}(x-x_j)$，那么 $\prod_{j\neq i}(x_i-x_j)$ 就是求 $\frac{M(x)}{x-x_i}$ 在 $x=x_i$ 处的点值，根据 L'Hospital 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 $\sum_{i=0}^{n} v_i \prod_{j\neq i}(x-x_j)$。

考虑分治，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 $R(x)\left(\sum_{i=l}^{mid} v_i \prod_{j\in[l,mid],j\neq i}(x-x_j)\right) + L(x)\left(\sum_{i=mid+1}^{r} v_i \prod_{j\in[mid+1,r],j\neq i}(x-x_j)\right)$，两个括号里都是相同的形式，递归即可，时间复杂度 $T(n) = 2\,T(\frac{n}{2}) + O(n\log n) = O(n\log^2 n)$。

单位根

方程 $x^n = 1$ 在复数域 $\mathbb{C}$ 下的 n 个解 $\omega_n^0, \omega_n^1, \omega_n^2, ..., \omega_n^{n-1}$ 称为 n 次单位根。
其中 $\omega_n^k = \cos \frac{2k\pi}{n} + \sin \frac{2k\pi}{n} i$，$n$ 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：

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

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 $\omega_n^0, \omega_n^1, ..., \omega_n^{n-1}$ 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 $\omega_{n/2}^0, \omega_{n/2}^1, ..., \omega_{n/2}^{n/2-1}$ 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 $\{a_0, a_1, ..., a_{n-1}\}$，求出了数组 $\{b_0, b_1, ..., b_{n-1}\}$ 满足

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

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

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

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 的实质

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

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”的意思是说 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敛散性的问题。

普通生成函数

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

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

例如，数列 $\{1, 1, ...\}$ 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

从前有一只小猫，它非常地爱滚，并且掌握了很多种不同的滚法。已知小猫有 a_i 种滚法可以在 1 秒内滚 i 米，求它总共滚了 n 米的方案数模 998244353。

constraint

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

一个使用普通生成函数的例子

description

从前有一只小猫，它非常地爱滚，并且掌握了很多种不同的滚法。已知小猫有 a_i 种滚法可以在 1 秒内滚 i 米，求它总共滚了 n 米的方案数模 998244353。

constraint

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

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 $F(x) = F(x)A(x) + 1$（或者你也可以直接令 $F(x) = \sum_{i \geq 0} A^i(x)$），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 $a_0, a_1, ...$ 的指数生成函数 (exponential generating function, 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 $\{1, 1, ...\}$ 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

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

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。

constraint

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

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。

constraint

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

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 $G(x) = \sum_{i \ge 0} 2^{\binom{i}{2}} \frac{x^i}{i!}$。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

$$G(x) = \sum_{i \ge 0} \frac{F^i(x)}{i!} = \exp F(x), F(x) = \ln G(x)$$

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

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

solution

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

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

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

一个计数技巧

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

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做可重排列的方案数。

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

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。(每次询问的 k 都是相同的，只有 x, y, r 不同。)

constraint

$n = 5, 0 \leq r \leq 1000, 1 \leq k \leq 10^9, q \leq 25000.$

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。(每次询问的 k 都是相同的，只有 x, y, r 不同。)

constraint

$n = 5, 0 \leq r \leq 1000, 1 \leq k \leq 10^9, q \leq 25000.$

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 $O((n^2 r \log r + n^3 r) \log k + q)$。

概率生成函数

对于数列 $\{a_0, a_1, ...\}$，如果存在某个随机变量 X 满足 $\Pr(X = i) = a_i$，那么 $\{a_0, a_1, ...\}$ 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 $A^{(k)}(1) = \sum_{i\geq k} i^{\underline{k}} \Pr(X = i) = E(X^{\underline{k}})$。

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：

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

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

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

需要求的是 $F'(1)$。

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

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 $\begin{bmatrix} f_1 & f_2 & \cdots & f_m \end{bmatrix}$ 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 $\begin{bmatrix} f_1 & f_2 & \cdots & f_m \end{bmatrix} A^n = \begin{bmatrix} f_{n+1} & f_{n+2} & \cdots & f_{n+m} \end{bmatrix}$，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 $A^0, A^1, A^2, ..., A^{m-1}$ 线性表示，不妨记 $A^n = \sum_{i=0}^{m-1} b_{n,i} A^i$，左右两边左乘行向量 $\begin{bmatrix} f_1 & f_2 & \cdots & f_m \end{bmatrix}$，可以得到

$$\begin{bmatrix} f_{n+1} & f_{n+2} & \cdots & f_{n+m} \end{bmatrix} = \sum_{i=0}^{m-1} b_{n,i} \begin{bmatrix} f_{i+1} & f_{i+2} & \cdots & f_{i+m} \end{bmatrix}$$

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 $B_{n+m}(x) = B_n(x) B_m(x) \mod (x^m - \sum_{i=1}^{m} c_i x^{m-i})$，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。

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 $O(n2^n)$。
假设数组 $\{a_n\}$ 进行 FWT 后的结果是 $\{a'_n\}$，那么存在等式

$$a'_n = \sum_m (-1)^{popcount(n\&m)} a_m$$

$popcount(x)$ 是 x 作为二进制数在多少个为上为 1。这个结果其实从 DFT 的过程中得到。
这种基于 DFT 的理解方式可以很容易扩展到 k 进制 FWT 。

子集卷积

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

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。

JOI2018Final 毒蛇越狱

description

有 2^n 条蛇，每条蛇唯一对应一个长度为 n 的二进制编号和一个权值，有 q 次询问，每次询问会给出一个长度为 n 的包含 `0`，`1`，`*`(通配符) 的模式串，求所有编号能被这个模式串匹配的蛇的权值之和。

constraint

$n \leq 20, q \leq 10^6$.

JOI2018Final 毒蛇越狱

description

有 2^n 条蛇，每条蛇唯一对应一个长度为 n 的二进制编号和一个权值，有 q 次询问，每次询问会给出一个长度为 n 的包含 `0`，`1`，`*`(通配符) 的模式串，求所有编号能被这个模式串匹配的蛇的权值之和。

constraint

$n \leq 20, q \leq 10^6$.

solution

如果 `*` 的数量很少，就可以直接枚举每个 `*` 的取值。
如果 `0` 的数量很少（先考虑没有 `0`，发现相当于是一个高维后缀和），可以枚举 `0` 的位置并容斥成 `*` 和 `1`.
如果 `1` 的数量很少，和前一种情形类似也可以先做高维前缀和然后对 `1` 的位置容斥。
由于长度为 n 的串中包含的 `0`，`1`，`*` 数量的最小值不会超过 $\frac{n}{3}$，所以我们实际上得到了一个复杂度为 $O(q2^{\frac{n}{3}})$ 的算法，足以通过该题目。

uoj310 黎明前的巧克力

description

给出一个大小为 n 的集合 $\{a_1, a_2, \cdots, a_n\}$，求有多少对有序的不交子集满足两个集合元素的异或和相等。

constraint

$n, a_i \leq 10^6$.

uoj310 黎明前的巧克力

description

给出一个大小为 n 的集合 $\{a_1, a_2, \cdots, a_n\}$，求有多少对有序的不交子集满足两个集合元素的异或和相等。

constraint

$n, a_i \leq 10^6$.

solution

一个数如果被选入某个集合中，那么它具体在哪边是无所谓的，所以问题相当于一个数选它有两种方案，不选有一种方案，问选出一些数异或和为 0 的总方案数。

我们企图构造出 n 个“多项式”进行异或卷积，但限于复杂度只得作罢，但我们可以得到这样的观察结果：单个多项式进行 FWT 变换后，数组中只会包含 3 和 -1，而我们是可以知道这两种数字的具体分布位置的。

uoj310 黎明前的巧克力

solution con't

具体地，只要计算出最终的 FWT 异或卷积数组的每个位置 k 被乘上了 b_k 个 3 和 $n-b_k$ 个 -1，就可以通过 FWT 的一边逆运算得到答案。而 b_k 的计算方式是：

$$b_k = \sum_{i=1}^{n} [popcount(k \& a_i) \text{ is even}]$$

可以通过一次 FWT 来求出。

例题 集合划分计数

description

给出一个集合 $S = \{x_1, \cdots, x_n\}$ 和一个 S 上的集合族 $\mathcal{F} = \{S_0, \cdots, S_m\}$ 满足 $S_i \subseteq S, \forall 1 \leq i \leq m.$
一个划分 $\mathcal{P}$ 是 F 的一个子集，满足 $\mathcal{P}$ 中所有集合并集为 S，且两两交集为空。
求所有划分的数量。进一步地，求大小不超过 k 的划分的数量 (loj154)。

constraint

$|S| \leq 20.$

例题 集合划分计数

solution

把给出集合族的集合幂级数记为 f，划分的集合幂级数记为 g，那么我们形式化地有

$$g = \sum_{i=0}^{+\infty} \frac{f^i}{i!} = \exp f$$

对集合幂级数做 $\exp$ 相当于是在集合或卷积下对占位多项式做 $\exp$。可以朴素地实现多项式 $\exp$ 做到 $O(n^2 2^n)$ 的复杂度。
当限制划分大小不超过 k 时，上式的枚举上限从 $+\infty$ 变成了 k，过程其实也是类似的，具体可以参考这里的实现方法。

Q.& A.

大家可以自由提问。

谢谢大家！

祝大家学业有成！