多项式 & 生成函数从入门到出门

Wallbreaker5th

2021 年 3 月 24 日

约定

- 一个关于 x 的 n 次**多项式**可以写作 $F(x)=\sum_{i=0}^{n} f_i x^i$。多项式只有有限个系数非零。
- 如果有无限个系数非零，这是一个**形式幂级数**。OI 中一般在模 x^n 的精度下进行形式幂级数的运算，下文我们基本不区分多项式与形式幂级数。
- 其中 x 只是一个符号，我们一般不会关心幂级数代入具体的 x 后的结果，因此我们不用关心其是否收敛。

约定

- $[x^i]F(x) = f_i$ 为 $F(x)$ 的 i 次项系数。
- 后文我们一般用大写字母来代表一个多项式，用对应的小写字母代表其系数。
- $\deg(F(x)) = \max\{n \mid f_n \neq 0\}$ 为多项式的度数。
- $\operatorname{ord}(F(x)) = \min\{n \mid f_n \neq 0\}$ 为多项式的阶数。

代数意义

对于一个数列 $a = \langle a_0, a_1, a_2, \cdots \rangle$，其普通生成函数（OGF）为 $A(x) = \sum_i a_i x^i$。
可以看出，一个数列的普通生成函数包含了这个数列的全部信息。
例如 $b = \langle 1, 1, \cdots \rangle$，其普通生成函数为 $B(x) = \sum_{i=0}^{\infty} x^i$。由于我们不考虑其收敛性，可以得到 $B(x) = \dfrac{1}{1-x}$。

组合意义

我们经常会去研究一类组合对象，它们各自有各自的大小 a_i（和权值 b_i）。
它们可以组合起来，得到的对象的大小为二者之和，权值为二者之积。
于是我们可以把一个组合对象写成 $a_i \times x^{b_i}$ 的形式；在相乘的的时候，对象组合起来：权值相乘，大小相加。
一系列组合对象便可以写成一个多项式 $\sum a_i x^{b_i}$；在相乘的时候，结果便是二者各自取一个对象组合可以形成的所有新的组合对象。

代数意义

对于一个数列 $a = \langle a_0, a_1, a_2, \cdots \rangle$，其指数生成函数（EGF）为 $A(x) = \sum_i \frac{1}{i!} a_i x^i$。
可以看出，一个数列的指数生成函数包含了这个数列的全部信息。
例如 $b = \langle 1, 1, \cdots \rangle$，其生成函数为 $B(x) = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$。由于我们不考虑其收敛性，可以得到 $B(x) = e^x$。(这也是「指数生成函数」这个名称的来源)

组合意义

我们经常会去研究一类组合对象，它们各自由 a_i 个极小的单位组成，这 a_i 个极小的单位还各自有标号。
它们可以组合起来，得到的对象的大小为二者之和，权值为二者之积；同时我们还要为其分配标号，每个对象内部的标号的大小关系应当保持不变。
于是我们可以把一个组合对象写成 $\frac{1}{a_i!} \times x^{a_i}$ 的形式；在相乘的时候，得到 $\frac{1}{(a_i+a_j)!}\binom{a_i+a_j}{a_i}x^{a_i+a_j}$：对象组合起来，大小相加，分配编号的不同方式带来组合数。
一系列组合对象同样可以写成一个多项式；在相乘的时候，结果便是二者各自取一个对象组合可以形成的所有新的组合对象。

试看看

求下列数列的普通生成函数

- $f = \langle a^0, a^1, a^2, \cdots \rangle$
- g_i 代表「一个 $i+1$ 个点的无向图，$0 \leftrightarrow j(\forall 1 \leq j \leq n)$、$j \leftrightarrow j+1(\forall 1 \leq j < n)$ 之间有边」的生成树个数。
- $h_0 = h_1 = 1, h_i = h_{i-1} + 2h_{i-2} + (-1)^i$

试看看

Solution

- $F(x) = \dfrac{1}{1-ax}$
- $G(x) = \dfrac{x}{1-3x+x^2}$ $\quad (g_i = \sum g_{i-1} + \sum_{k<i} g_k + [i>0])$
- $H(x) = \dfrac{1+x+x^2}{(1-2x)(1+x)^2}$

乘法

多项式乘法与数列卷积相对应。减法卷积只需要翻转其中一个多项式的系数。

乘法

多项式乘法与数列卷积相对应。减法卷积只需要翻转其中一个多项式的系数。

乘 $\dfrac{1}{1-x}$ 相当于求前缀和；乘 $1-x$ 相当于差分。

乘法

多项式乘法与数列卷积相对应。减法卷积只需要翻转其中一个多项式的系数。

乘 $\frac{1}{1-x}$ 相当于求前缀和；乘 $1-x$ 相当于差分。

对于多项式 $F(x)$，将 $\langle i!f_i\rangle$ 与 $\frac{1}{i!}$ 减法卷积起来再各自乘 x^i 可以得到 $F(x+1)$ 的各项系数。

FFT

下文我们认为一个 n 次多项式一次 FFT 的用时为 $\mathsf{E}(n)$；两个 n 次多项式相乘的用时为 $\mathsf{M}(n) = 3\mathsf{E}(n)$。二者都是 $O(n\log n)$ 的。
下文我们记一个多项式 $F(x)$ 做长为 n 的 FFT 后的结果为 $\mathcal{F}_n(F)$。
通过长为 n FFT 做乘法，实质是求结果模 $x^n - 1$ 的结果，也就是循环卷积。

FFT

尽管 FFT（NTT）的算法过程十分板子（并且我们假设大家都背下来了），不同选手的写法实际上也有不同。
在预处理原根时，我们可以选择对于一个足够大的 N 预处理 ω_N^i。但是在 NTT 时，内存访问会不连续。另一种方法是对于 $n = 2^0 \sim 2^l$ 都预处理 ω_n^i，并顺序放入数组中。届时我们要访问原根即使用 `w[(i<<1)+k]`。
在做 IDFT 时，本应用共轭的单位根运算。但实际上可以直接用单位根，在完成后 `reverse` $[1, n)$，这样做刚好可以让各个位置 reverse 到对应的共轭根。

多项式乘法·三次变两次

通常做多项式乘法时，我们需要做两次 FFT 和一次 IFFT。而如果两个多项式都是实数系数的，我们可以优化到一次 FFT 和一次 IFFT。

原理很简单：若要计算 $A \cdot B$，可以计算 $(A+\mathrm{i}B)^2 = A^2 + B^2 + 2\mathrm{i}A \cdot B$。结果的虚部只包含 $2\mathrm{i}A \cdot B$，将其单独提出来，除以 $2\mathrm{i}$ 即可。

求逆

一般都是解方程冒出来的分数。似乎没啥特别的组合意义。

求逆·朴素求法

已知 F，求 $G = 1/F$。
显然 $GF - 1 = 0$，设 $G_0 \equiv G \mod x^n$，则：

$$
\begin{aligned}
G_0 F - 1 &\equiv 0 \pmod{x^n} \\
-(G_0 F - 1)^2 &\equiv 0 \pmod{x^{2n}} \\
F(2G_0 - G_0^2 F) - 1 &\equiv 0 \pmod{x^{2n}}
\end{aligned}
$$

一次迭代（$G \bmod x^n \to G \bmod x^{2n}$）需要 1 次长度为 $2n$ 的乘法、1 次长度为 $4n$ 的乘法，用时 $18\mathsf{E}(n)$。所以计算 $G \pmod{x^n}$ 所需总时间为 $18\mathsf{E}(n)$。

求逆·优化 1

考虑 $G \bmod x^{2n} = G_0 - (FG_0 - 1)G_0 \bmod x^{2n}$。显然 $\deg((F \bmod x^{2n})G_0 - 1) < 3n$，$\operatorname{ord}((F \bmod x^{2n})G_0 - 1) \geq n$。于是 $FG_0 - 1$ 有用的部分只有 $[n, 3n)$ 次项，我们可以做一个长为 $2n$ 的循环卷积（而非长为 $4n$ 的卷积）。

同理，计算 $(FG_0 - 1)G_0$ 时也只需要做长为 $2n$ 的循环卷积。

总共用时降到了 $12\mathrm{E}(n)$。由于计算时涉及两次关于 G_0 的循环卷积，我们记录下 $\mathcal{F}(G_0)$ 避免重复运算。总共用时降到了 $10\mathrm{E}(n)$。

求逆·优化 2

考虑 $G \bmod x^{2n} = G_0 - (G_0^2 F - G_0) \bmod x^{2n}$。如果我们能够做到长为 $3n$ 的循环卷积（得到 $T = G_0^2 F$ 有用的部分），我们可以做到 9E(n)。

实际上我们不需要支持长为 $3n$ 的 FFT。考虑将 $T = G_0^2 (F \bmod x^{2n})$ 表示为 $\boxtimes + Px^n + Qx^{2n} + Rx^{3n}$，其中 $\boxtimes, P, Q, R$ 都是低于 n 次的多项式（$\boxtimes$ 用不到）。

现在我们用循环卷积求出 $T \bmod (x^{2n} - 1)$ 和 $T \bmod (x^n - \mathrm{i}) = T(\omega_{4n} x) \bmod (x^n - 1)$。相当于我们求出了 $Q, P + R, \mathrm{i}P - Q - \mathrm{i}R$，还原出 P 即可。

牛顿迭代

后面我们会遇到许多已知函数 A，要解出 $A(F(x))=0$ 的情形。这时我们需要牛顿迭代。

设 $F_0 = F \bmod x^n$，则 $F \bmod x^{2n} = A_0 - \dfrac{A(F_0)}{A'(F_0)}$。通过在 F_0 处将 A 泰勒展开可以得到。

观察可以发现 $\operatorname{ord}(A(F)) \geq n$，也就是说 $A'(F)$ 的精度只需要达到 x^n。

$\exp$ & $\ln$

$$\exp(F(x)) = \sum_{i=0}^{\infty} \frac{F(x)^i}{i!}$$

EGF 的 $\exp$ 代表任意多个相应组合对象的组合。($F(x)^i$ 即为将 i 个对象组合；$\frac{1}{i!}$ 消除了各组之间的差异，避免重复计数)

$\exp$ & $\ln$

$$\exp(F(x)) = \sum_{i=0}^{\infty} \frac{F(x)^i}{i!}$$

EGF 的 $\exp$ 代表任意多个相应组合对象的组合。($F(x)^i$ 即为将 i 个对象组合；$\frac{1}{i!}$ 消除了各组之间的差异，避免重复计数)

$\ln$ 便是 $\exp$ 的逆运算。

先后使用 $\ln$ 与 $\exp$ 可以求出一个多项式的 n 次方。

求 ln

$(\ln F)' = F' \ln' F = F'/F$，因此问题转化为求两多项式的商。直接采用前文做法求逆元，再卷积一次能做到 $15\mathrm{E}(n)$。

求商·优化

已知 F, H，令 $G = 1/F, Q = HG = H/F$，求 Q。
令 $G_0 = g \bmod x^n$， $G_1 = (G \bmod x^{2n} - G_0)/x^n$；
$H_0 = H \bmod 2^n$， $H_1 = (H \bmod 2^{2n} - H_0)/x^n$。
如果我们需要求 G，考虑计算

$$\begin{aligned} Q \bmod x^{2n} =& (G \bmod x^{2n})(H \bmod x^{2n}) \bmod x^{2n} \\ =& H_0 G_0 + (G_0 H_1 + G_1 H_0) x^n \bmod x^{2n} \end{aligned}$$

求商·优化

- 计算 $\mathcal{F}_{2n}(G_0)$ 和 G_0, G_1 需要 $18\mathsf{E}(n)$（即长为 $2n$ 的求逆；前者在求逆过程中已经求出）；
- * 计算 $\mathcal{F}_{2n}(G_1), \mathcal{F}_{2n}(H_0), \mathcal{F}_{2n}(H_1)$ 需要 $6\mathsf{E}(n)$（三次 DFT）；
- * 计算 $H_0G_0, G_0H_1 + G_1H_0$ 需要 $4\mathsf{E}(n)$（两次 IDFT）

计算 $G \bmod x^{2n}$、$Q \bmod x^{2n}$ 总时间为 $28\mathsf{E}(n)$，因此计算 $G \bmod x^n$、$Q \bmod x^n$ 总时间为 $14\mathsf{E}(n)$。

求商·优化

上一页所用到的技巧（尤其是带 * 号的两点）可以描述为：对于 F, G，已知 $F \bmod x^n, G \bmod x^n, \mathcal{F}_n(F \bmod x^{n/2})$，我们可以用 $5\mathrm{E}(n)$ 的时间求得 $FG \bmod x^n$。后文会多次使用该技巧。

求商·优化

如果不需要求 G，我们可以做到更快：考虑

$$Q \bmod x^{2n} = Q_0 - (FQ_0 - H)G_0 \bmod x^{2n}$$

- 计算 G_0 需要 $9\mathrm{E}(n)$ 的时间（求逆）；
- 计算 $\mathcal{F}_{2n}(G_0), Q_0$ 需要 $6\mathrm{E}(n)$ 的时间（一次乘法）；
- 计算 FQ_0 需要 $6\mathrm{E}(n)$ 的时间（一次循环卷积）；
- 计算 $(FQ_0 - H)G_0$，已知 $\mathcal{F}_2n(G_0)$，需要 $4\mathrm{E}(n)$ 的时间。

计算 $Q \bmod x^{2n}$ 总时间为 $25\mathrm{E}(n)$，因此计算 $G \bmod x^n$、$Q \bmod x^n$ 总时间为 $12.5\mathrm{E}(n)$。

求商·优化

进一步，$(FQ_0-H)G_0 \bmod x^{2n}$ 可以视为 $((FQ_0-H)/x^n)G_0 \bmod x^n$。
我们注意到 $\mathcal{F}_n(G_0 \bmod x^{n/2})$ 在求 G_0 过程中已经求出。因此 $Q_0=G_0H_0 \bmod x^n, ((FQ_0-H)/x^n)G_0 \bmod x^n$ 二者都可以用上文所述技巧优化。
由于相同的 DFT 只用计算一次，这两个结果只需要 $9\mathsf{E}(n)$ 计算。总时间为 $12\mathsf{E}(n)$。

求 exp

已知常数项为 0 的 F，令 $G = \exp F, H = 1/G = 1/\exp F$，求 G。即 $\log G - F = 0$。令 $G_0 = G \bmod x^n$，那么有：

$$\begin{aligned} G \bmod x^{2n} =& G_0 - \frac{\ln G_0 - F}{1/G_0} \bmod x^{2n} \\ =& G_0 - (\ln G_0 - F) G_0 \bmod x^{2n} \\ =& G_0 - G_0(\textstyle\int (G_0'/G_0) - F) \bmod x^{2n} \end{aligned}$$

求 exp

从 $G \bmod x^n, H \bmod x^{2n}$ 计算 $G \bmod x^{2n}$ 需要计算：一次倒数、一次长度 $2n$ 的乘法、一次长度 $4n$ 的乘法；计算 $H \bmod x^{2n}$ 需要一次倒数。因此计算 $G \bmod x^n, H \bmod x^n$ 要用时 $54\mathsf{E}(n)$。
如果不需要计算 H，可以省略最后一次迭代的相关运算。用时 $45\mathsf{E}(n)$。
求倒数、求 $\ln$ 若都使用最快的一种，用时为 $28.5\mathsf{E}(n)$。

求 exp·优化 1

改写迭代式，令 $F_0 = F \bmod x^n$。注意到
$\mathrm{ord}((\int(G_0'/G_0) - F')G_0) \geq n$，即 $\mathrm{ord}((G_0'/G_0) - F') \geq n-1$；
于是有 $G_0H_0(G_0'/G_0 - F') \equiv G_0'/G_0 - F' \pmod{x^{2n-1}}$

$$
\begin{aligned}
G \bmod x^{2n} &= G_0 - G_0(\int(G_0'/G_0) - F) \bmod x^{2n} \\
&= G_0 - G_0 \int (G_0'/G_0 - F') \bmod x^{2n} \\
&= G_0 - G_0 \int (G_0H_0(G_0'/G_0 - F')) \bmod x^{2n} \\
&= G_0 - G_0 \int (G_0'H_0 - F' - (G_0H_0 - 1)F_0') \bmod x^{2n}
\end{aligned}
$$

求 exp·优化 1

再注意到
$\mathrm{ord}(G_0{}'H_0 - F' - (G_0H_0 - 1)F_0{}') = \mathrm{ord}(G_0'/G_0 - F') \geq n-1$,
左式中 $\mathrm{ord}(G_0H_0 - 1) \geq n$，所以 $\mathrm{ord}(G_0'H_0 - F') \geq n-1$

- 计算 G_0H_0 和 $G_0'H_0 - F'$ 需要 2 次长为 n 的循环卷积；
- 计算 $(G_0H_0 - 1)F_0'$ 和 $G_0 \int(G_0'H_0 - F' - (G_0'H_0 - 1)F_0')$ 需要两次长为 $2n$ 的循环卷积；
- 计算 $H \bmod x^2n$ 需要一次计算倒数的迭代。

计算 $G \bmod x^n, H \bmod x^n$ 共需 $30\mathrm{E}(n)$；最后一次迭代可能可以省略计算 H，共需 $24\mathrm{E}(n)$。

求 exp·优化 2

tl;dr
有兴趣的读者可以参见
`https://negiizhao.blog.uoj.ac/blog/4671`。最快可以做到 $15.5\mathrm{E}(n)$

试看看

题目

用生成函数来证明：一个无向完全图的生成树个数是 n^{n-2}.

试看看

题目

用生成函数来证明：一个无向完全图的生成树个数是 n^{n-2}.

Solution

考虑一个点，将其删去后整棵树会分裂出多个。设答案的 EGF 为 F，于是：

$$F(x) = x\,\mathrm{e}^{F(x)}$$

试看看

Solution

现在我们搬出《具体数学》，其中有定义 **广义指数级数** $\mathcal{E}_t(x)$：

$$\mathcal{E}_t(x) = \sum_{k\geq 0}(tk+1)^{k-1}\frac{x^k}{k!}$$

其中 $\mathcal{E}(x) = \mathcal{E}_1(x)$ 满足 $\mathcal{E}(x) = e^{x\mathcal{E}(x)}$。
我们发现 $F(x) = x\mathcal{E}(x)$。于是答案为 $(n-1)![x^{n-1}]\mathcal{E}(x) = n^{n-2}$。

开根

一般都是解方程冒出来的根号。似乎没啥特别的组合意义。

求平方根

一个多项式显然能解出两个根，对应常数项的两个根。
给定 F，令 $G=F^{1/2}, H=1/G=G^{-1/2}$，求 G。
相当于 $G^2-F=0$，令 $G_0=G \bmod x^n, H_0=H \bmod x^n$，有

$$
\begin{aligned}
G \bmod x^{2n} &= G_0-\frac{G_0^2-F}{2G_0} \bmod x^{2n}\\
&= G_0-\frac{(G_0^2-F)H_0}{2} \bmod x^{2n}
\end{aligned}
$$

求平方根

从 $G \bmod x^n, H \bmod x^n$ 计算 $G \bmod x^{2n}$ 需要一次长为 $2n$ 的乘法、一次长为 $4n$ 的乘法；计算 $H \bmod x^{2n}$ 需要一次计算倒数的迭代；用时 $36\mathsf{E}(n)$。如果不需要计算 H，总时间为 $27\mathsf{E}(n)$。倒数迭代若使用最快的一种，总时间为 $18\mathsf{E}(n)$。

求平方根·优化 1

$$G \bmod x^{2n} = G_0 - \frac{(G_0^2 - F)H_0}{2} \bmod x^{2n}$$

计算 G_0^2 需要一次长为 n 的循环卷积，计算 $(G_0^2 - F)H_0$ 需要一次长为 $2n$ 的循环卷积，计算 $H \bmod x^{2n}$ 需要一次计算倒数的迭代；一次迭代用时 $21\mathsf{E}(n)$。如果不需要计算 H，总用时为 $15\mathsf{E}(n)$。
倒数迭代若使用最快的一种，总时间为 $13.5\mathsf{E}(n)$。

求平方根·优化 2

$$G \bmod x^{2n} = G_0 - \frac{(G_0^2 - F)H_0}{2} \bmod x^{2n}$$

- 保留前一轮迭代计算倒数使用的 $\mathcal{F}_n(G_0)$;
- 计算 G_0^2 需要 $\mathsf{E}(n)$;
- 计算 $\mathcal{F}_{2n}(H_0), (G_0^2 - F)H_0$ 需要 $6\mathsf{E}(n)$;
- 计算 $\mathcal{F}_{2n}(G \bmod x^{2n}), H \bmod x^{2n}$ 需要一次计算倒数的迭代;已知 $\mathcal{F}_{2n}(H_0)$,需要 $7\mathsf{E}(n)$。

求平方根·优化 2

总时间 $14\mathrm{E}(n)$；若不计算 H，可以省略最后一次迭代相关运算，时间 $10.5\mathrm{E}(n)$。
再进一步，最后一轮不需要计算倒数，于是无需保留 $\mathcal{F}_{2n}(H_0)$，$G_0^2 - F)H_0$ 适用于上文所述技巧，需要 $5\mathrm{E}(n)$。总时间 $10\mathrm{E}(n)$。

欧拉变换

对于多项式 $F(x)$，我们定义其欧拉变换后结果为：

$$\text{Euler}(F(x)) = \prod_i \frac{1}{(1-x^i)^{f_i}}$$

其组合意义类似于 $\exp$，只是把有标号（EGF）变为无标号（OGF）。
感性理解一下就是大小为 i 的每个对象都可以取任意多个。

欧拉变换

$$
\begin{aligned}
\mathrm{Euler}(F(x)) &= \exp\left(\sum_i \ln \frac{1}{(1-x^i)^{f_i}}\right) \\
&= \exp\left(\sum_i -f_i \ln(1-x^i)\right) = \exp\left(\sum_i -f_i \sum_j \frac{x^{ij}}{j}\right) \\
&= \exp\left(\sum_j \frac{1}{j} \sum_i f_i x^{ij}\right) \quad = \exp\left(\sum_j \frac{f(x^j)}{j}\right)
\end{aligned}
$$

欧拉变换

上面那个式子是欧拉变换的另一个定义式。可以发现这个定义式可以在相对合理的时间复杂度内计算。

三角函数

$$\sin(F(x)) = \sum_{i=0}^{\infty} \frac{(-1)^i}{(2i+1)!} x^{2i+1}$$

$$\cos(F(x)) = \sum_{i=0}^{\infty} \frac{(-1)^i}{(2i)!} x^{2i}$$

可以发现这玩意挺像 exp，且只有奇/偶数位置有值。~~也不知道有什么用。~~实际上用欧拉公式来看就是：

$$\sin(F(x)) = (\exp(\mathrm{i}x) - \exp(-\mathrm{i}x)) / (2\mathrm{i})$$

$$\cos(F(x)) = (\exp(\mathrm{i}x) + \exp(-\mathrm{i}x)) / 2$$

带余除法

可能应用于多点求值/常系数齐次线性递推。
给定 $A(x), B(x)(\deg(A) = n, \deg(B) = m)$，求 $C(x), R(x)$ 满足 $A(x) = B(x)C(x) + R(x)$ 且 $\deg(R) < \deg(B)$。
定义 A^{R} 为 A 系数翻转后的结果。显然 $A^{\mathrm{R}}(x) = x^n A(1/x)$。因此

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

在模 x^{n-m+1} 意义下通过求逆解出 C^{R}，此时 R^{R} 恰好被忽略。

试看看

题目

无标号有根树计数。

试看看

题目

无标号有根树计数。

Solution

设 $F(x)$ 为其生成函数，容易得出：

$$F(x) = x \cdot \mathrm{Euler}(F(x))$$

接着两边求导。

试看看 Solution

$$
\begin{aligned}
F'(x) &= \exp\left(\sum_{i=1}^{\infty}\frac{F(x^i)}{i}\right) + x\exp\left(\sum_{i=1}^{\infty}\frac{F(x^i)}{i}\right)\sum_{i=1}^{\infty}\frac{ix^{i-1}\cdot F'(x^i)}{i} \\
&= \exp\left(\sum_{i=1}^{\infty}\frac{F(x^i)}{i}\right) + F(x)\sum_{i=1}^{\infty}x^{i-1}F'(x^i)
\end{aligned}
$$

$$
xF'(x) = F(x) + F(x)\sum_{i=1}^{\infty}x^i\cdot F'(x^i)
$$

试看看 Solution

设 $G = \sum_{i=1}^{\infty} x^{i-1} F'(x^i)$，可得：

$$f_i = \frac{1}{i-1} \sum_k f_k g_{i-k}, f_1 = 1$$

$$g_n = \sum_{d|n} d \cdot f_d$$

于是可以分治 FFT 做：分治过程中，我们把左半边 f 与开头一段 g 的卷积、左半边 g 与开头一段 f 的卷积贡献到右侧；在分治边界处乘上 $\frac{1}{i-1}$，并将 f_i 贡献到对应的 g 去。复杂度 $O(n \log^2 n)$

试看看

题目

无标号无根树计数。

试看看

题目

无标号无根树计数。

Solution

完成上一个问题后，这个问题就简单许多。设上个问题的答案为 f_i。
我们尝试强制规定重心为根，用所有答案减去根不是重心的答案。

试看看

Solution

当 $n = 2m + 1$ 是奇数时，一个点不是重心意味着它有一个子树大小大于 m，答案为：$f_n - \sum_{k=m+1}^{n-1} f_k f_{n-k}$。
当 $n = 2m$ 是偶数时，重心可能不止一个点；当两个子树大小都是 m 且它们形态不同时我们会重复计数。因此答案还要再减 $\binom{f_m}{2}$。

拉格朗日反演

若 $F(G(x)) = x$，我们称 F 和 G 互为复合逆。
F, G 都没有常数项，且它们的一次项互为逆元。
已知 G，我们可以用拉格朗日反演求出 F 的一项：

$$[x^n]F(x) = \frac{1}{n}[x^{-1}](\frac{1}{G(x)})^n = \frac{1}{n}[x^{n-1}](\frac{x}{G(x)})^n$$

扩展拉格朗日反演

若 $F(G(x))=x$，有：

$$[x^n]H(F(x))=\frac{1}{n}[x^{-1}]H'(x)(\frac{1}{G(x)})^n=\frac{1}{n}[x^{n-1}]H'(x)(\frac{x}{G(x)})^n$$

试看看

BZOJ3684 大朋友和多叉树

求有 n 个叶子，且非根节点的子节点数目在集合 S 内的多叉树个数。

试看看

BZOJ3684 大朋友和多叉树

求有 n 个叶子，且非根节点的子节点数目在集合 S 内的多叉树个数。

Solution

设答案生成函数为 $F(x)$，有：

$$F(x) = x + \sum_{i \in S} F(x)^i$$

记 $G(x) = x - \sum_{i \in S} x^i$，则 $G(F(x)) = x$，拉格朗日反演即可。

试看看

题目

无向带标号边双连通分量计数。

试看看

题目

无向带标号边双连通分量计数。

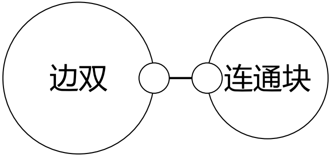
Solution

前置知识：无向带标号连通图计数。
设无向图的 EGF 为 $F(x)$，无向连通图的 EGF 为 $G(x)$，边双的 EGF 为 $H(x)$。
显然有 $f_i = 2^{i(i-1)/2}$，$F(x) = \exp(G(x))$，因此我们可以求出 $G(x)$。

试看看

Solution

现在设无向有根连通图的 EGF 为 $A(x)$，有根边双的 EGF 为 $B(x)$，显然有根无根只差了个 i 的系数。
对于一个有根连通图，根肯定恰好在一个（极大）边双中。剩下的部分都可以看作这个边双之外又挂了多个连通块。



试看看

Solution

如果根所在边双的大小为 k，那么任意多个连通块接在边双上面的方案数为 $\exp(kA(x))$。于是：

$$A(x) = \sum_k b_k \exp(kA(x)) \frac{x^k}{k!} = B(x \exp(A(x)))$$

令 $C(x) = x\exp(A(x))$，$C^{-1}(x)$ 为 $C(x)$ 的复合逆，则 $B(x) = A(C^{-1}(x))$。

试看看

Solution

由扩展拉格朗日反演得:

$$
\begin{aligned}
B(x) &= \frac{1}{n}[x^{n-1}]A'(x)\left(\frac{x}{C(x)}\right)^n \\
&= \frac{1}{n}[x^{n-1}]A'(x)\left(\frac{1}{\exp(A(x))}\right)^n \\
&= \frac{1}{n}[x^{n-1}]A'(x)\exp(-nA(x))
\end{aligned}
$$

常系数齐次线性递推

已知数列 g 满足 $g_i = \sum_{k=1}^{m} a_k g_{i-k}$，给定 g 的初值，求 g_n。以下做法是 trivial 的：

- 暴力递推，$O(nm)$；
- 矩阵快速幂，$O(m^3 \log n)$。

常系数齐次线性递推

我们构造多项式 $F(x) = x^m - \sum_{k=1}^{m} a_k x^{m-k}$。可以这样理解：一个 x^m 模 $F(x)$ 后会给 $x^{m-1} \cdots x^0$ 的系数贡献，正如 a_i 会给 $a_{i+1} \cdots a_{i_m}$ 贡献。

因此我们只需要求出 $G(x)x^{n-m} \bmod F(x)$ 即可。$x^{n-m} \bmod F(x)$ 可以倍增 + 多项式取模即可，复杂度 $O(m \log m \log n)$。

多项式多点求值

给定多项式 $F(x)$，求其在 $x_1, x_2, \cdots x_n$ 的点值。

多项式多点求值

给定多项式 $F(x)$，求其在 $x_1, x_2, \cdots x_n$ 的点值。
广为人知的做法是多项式取模：
将多项式 $F(x)$ 模上 $x - x_0$ 后会得到 $F(x_0)$。于是我们希望求出 $F(x) \bmod (x - x_i)$ 的所有结果。我们分治完成这个过程：求出 $F(x) \bmod \prod_{i=l}^{mid}(x - x_i)$ 的结果，右侧同理，并递归下去做。

转置原理

利用转置原理，我们可以减小多点求值的常数。
一个线性算法可以看作一个矩阵 A，其输入是一个向量，输出其左乘 A 矩阵的结果。我们假设这个矩阵是方阵，否则适当补 0。
包括多项式乘法（一个多项式作为矩阵，另一个作为向量）、多点求值（各个点值写入矩阵，多项式作为向量）都可以看作线性算法。
一个线性算法 $a = Ab$ 的转置为 $a = A^{\mathsf{T}}b$。
当我们发现一个线性算法本身较难解决，我们可以选择优化其转置，再把这算法转置回来。

转置原理

一个线性算法的过程可以看作只包含以下三种基本操作：

- 交换 v_i, v_j；
- $v_i \leftarrow v_i \times a$；
- $v_i \leftarrow v_i + v_j \times a$。

转置之后，可以看作将算法倒序执行，并将基本操作换为：

- 交换 v_i, v_j；
- $v_i \leftarrow v_i \times a$；
- $v_j \leftarrow v_j + v_i \times a$。

一个对 $x_0 \sim x_{n-1}$ 多点求值的过程可以看作这样一个线性变换：

$$\mathbf{V}(x_0, x_1, \cdots, x_{n-1}) = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{bmatrix}$$

[illegible]

它的转置如下

$$\mathbf{V}(x_0, x_1, \cdots, x_{n-1})^{\mathsf{T}} \times \mathbf{v} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_0 & x_1 & x_2 & \cdots & x_{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_0^{n-1} & x_1^{n-1} & x_2^{n-1} & \cdots & x_{n-1}^{n-1} \end{bmatrix} \times \mathbf{v}$$

转置原理

我们发现，这玩意实际上求的是：

$$[x^k]\sum_{i=0}^{n-1}\frac{v_i}{1-x_ix}$$

我们一般的求法是分治时维护分子和分母。
要把这个算法转置回去，最关键的是转置乘法操作。

我们冷静分析一下（具体过程可以参考 2020 年陈宇的集训队论文），发现乘法操作的转置就是减法卷积，即 v_j 乘上 a_i 贡献给 $j-i$ 位置。
于是最终算法流程可以描述如下：

- 分治求出每个结点对应的 $(1-x_ix)$ 之积；
- 对于要求值的多项式（写成 $\mathbf{v}$），求出 $\mathbf{MUL}(P^{-1})^{\mathsf{T}}\mathbf{v}$，保留前 n 位；
- 自顶向下递归，令下传的两部分向量为 $\mathbf{l}=\mathbf{MUL}(R)^{\mathsf{T}}\mathbf{v},\mathbf{r}=\mathbf{MUL}(L)^{\mathsf{T}}\mathbf{v}$；
- 最后叶节点向量长度均为 1，对应该处的点值。

多点插值

根据拉格朗日插值：

$$F(x) = \sum_{i=1}^{n} y_i \prod_{i \neq j} \frac{x - x_j}{x_i - x_j}$$

$$= \sum_{i=1}^{n} \frac{y_i}{\prod_{i \neq j} x_i - x_j} \prod_{i \neq j} (x - x_j)$$

多点插值

设 $G(x)=\prod_i(x-x_i)$。于是红色部分为 $\left.\dfrac{G(x)}{x-x_i}\right|_{i=x_i}$，发现分子分母都是 0，于是洛一洛这玩意就是 $G'(x_i)$，我们分治算出 $G(x)$ 再多点求值求出 $G'(x_i)$。
这样的话，有颜色的部分就求出来了，剩下的部分大力分治即可。

上升幂多项式/下降幂多项式

我们定义 $x^{\underline{m}} = \prod_{i=0}^{m-1}(x-i)$ 为 x 的 m 阶下降幂，
$x^{\overline{m}} = \prod_{i=0}^{m-1}(x+i)$ 为 x 的 m 阶上升幂。
上升幂和下降幂有许多优美的性质，例如：

$$\Delta x^{\underline{m}} = m x^{\underline{m-1}} \qquad \sum_{x=0}^{n-1} x^{\underline{m}} = \frac{1}{m+1} n^{\underline{m+1}}$$

$$(x+y)^{\underline{n}} = \sum_{i=0}^{n} \binom{n}{i} x^{\underline{i}} y^{\underline{n-i}} \qquad (x+y)^{\overline{n}} = \sum_{i=0}^{n} \binom{n}{i} x^{\overline{i}} y^{\overline{n-i}}$$

上升幂多项式/下降幂多项式

一个多项式也能被写作

$$\sum_i a_i x^{\underline{i}}$$

或

$$\sum_i a_i x^{\overline{i}}$$

的形式。在这种形式下，一些运算（如对 x 求和）可以更高效地完成。

上升幂多项式/下降幂多项式

借助斯特林数，不同多项式之间可以这样转化：

$$x^n = \sum_k \left\{ n \atop k \right\} x^{\underline{k}} = \sum_k \left\{ n \atop k \right\} (-1)^{n-k} x^{\overline{k}}$$

$$x^{\underline{n}} = \sum_k \left[n \atop k \right] (-1)^{n-k} x^k$$

$$x^{\overline{n}} = \sum_k \left[n \atop k \right] x^k$$

我们期待 asd_a 为我们带来关于斯特林数的**精彩、详尽**的讲解。

牛顿级数

一个多项式同样可以写成

$$\sum_k c_k \binom{x}{k}$$

的形式。

并且我们有结论 $c_i = \Delta^i f(0) = \sum_k \binom{i}{k} (-1)^{i-k} f(k)$。

有时题目会以点值的形式给出一个多项式，且以这个多项式作为贡献。我们可以先求多项式的 n 阶差分，将多项式转化为牛顿级数，于是这当中出现的组合数无论是代数推导还是组合意义都会很优美。

牛顿级数

进一步地，借助牛顿级数，我们可以做到已知 n 次多项式在 0 到 n 处的值，$O(n)$ 求多项式的一个点值：

$$
\begin{aligned}
f(x) &= \sum_{i=0}^{m} \binom{x}{i} \Delta^i f(0) \\
&= \sum_{i=0}^{n} \binom{x}{i} \sum_{j=0}^{i} f(j) \binom{i}{j} (-1)^{i-j} \\
&= \sum_{j=0}^{n} f(j) \sum_{i=j}^{n} \binom{x}{i} \binom{i}{j} (-1)^{i-j}
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{j=0}^{n} f(j) \sum_{i=j}^{n} \binom{x}{j} \binom{x-j}{i-j} (-1)^{i-j} \\
&= \sum_{j=0}^{n} f(j) \binom{x}{j} \sum_{i=0}^{n-j} \binom{x-j}{i} (-1)^{i} \\
&= \sum_{j=0}^{n} f(j) \binom{x}{j} \binom{x-j-1}{n-j} (-1)^{n-j}
\end{aligned}
$$

Warning

警告

下面你可能会遇到：

EI 题 Hard version solution by **Elegia**

极其丑陋的方程和它的解

$$\begin{cases} F_0(x) = G_0(x) + xG_0(x)F_0(x) + x^3G_1(x)F_i(x) \\ F_1(x) = G_1(x) + xG_1(x)F_0(x) + x^3G_2(x)F_1(x) \\ F_2(x) = G_2(x) + xG_1(x)F_1(x) + x^3G_2(x)F_2(x) \end{cases}$$

解得

$$\begin{cases} F_0(x) = \dfrac{x^3G_1(x) - G_0(x)(x^3G_2(x)-1)}{(xG_0(x)-1)(x^3G_2(x)-1) - z^4G_1(x)^2} \\ F_1(x) = \dfrac{G_1(x)}{(xG_0(x)-1)(x^3G_2(x)-1) - z^4G_1(x)^2} \\ F_2(x) = \dfrac{G_2(z) + xG_1(x)F_1(x)}{1 - x^3G_2(x)} \end{cases}$$

显然 因此，不难得到其反演：

Bernoulli 数的使用是众所周知的

CF848E Days of Floral Colours

CF848E Days of Floral Colours

$2n$ 朵花放在圆上，将其染色，每种颜色恰有两朵花，且满足：

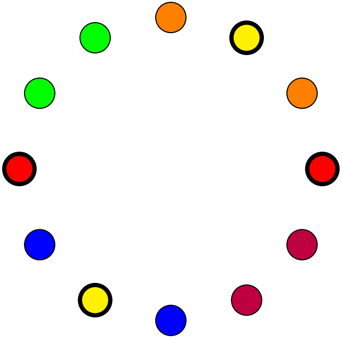
- 若两朵花颜色相同，在它们对面的一对花颜色也应当相同；
- 若两朵花颜色相同，它们必须互在对面或相隔至多一朵花。

若两朵相对的花颜色相同，将这两朵花标记起来。定义一种染色方案的分数为标记了的花分隔出的段的长度（可能为 0）之积；若没有标记了的花，分数为 0。求所有染色方案的分数之和。

$\cancel{n \le 50000}$ $n \le 5 \times 10^5$，7s。

两种方案不同当且仅当存在两朵花在一种方案中颜色相同，而另一种方案中不同。

CF848E Days of Floral Colours

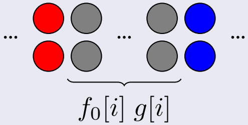


权值为 $3 \times 1 \times 3 \times 1 = 9$。

CF848E Days of Floral Colours

Solution

把环断开为两个半圆。我们首先计算 $f_0[i]$ 表示仅考虑长为 i 的一段，其染色的分数和：



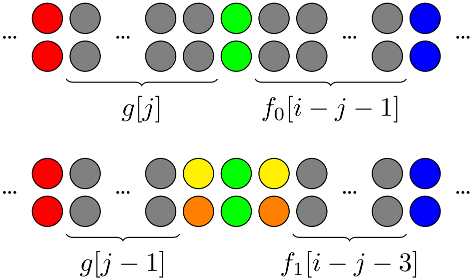
定义 $g[i]$ 为只靠“相隔至多一朵花”的方式对这一段染色的方案数，可得 $g[0]=1, g[1]=0, g[2]=g[i-2]+g[i-4]$。

CF848E Days of Floral Colours

Solution

求 f_0 分类讨论:

- 这里面没有相对的同色花，答案为 $g[i]i^2$；
- 这里面有相对的同色花，第一个在位置 j:
  - 没有一对有同色花跨过 j: $g[j]j^2 \times f_0[i-j-1]$；
  - 有一对有同色花跨过 j: 我们将右边的子问题（长为 i 的一段，在其左侧还有一对相对的异色花）定为 f_1，$g[j-1]j^2 \times f_1[i-j-3]$；


$g[j]$
$f_0[i-j-1]$
$g[j-1]$
$f_1[i-j-3]$

CF848E Days of Floral Colours

Solution

大力讨论一下，可以得到：

$$
\begin{aligned}
f_0[i] =& g[i]i^2 \\
&+ \sum_{j=0}^{i-1} g[j]j^2 \times f_0[i-j-1] \\
&+ \sum_{j=0}^{i-3} g[j](j+1)^2 \times f_i[i-j-3]
\end{aligned}
$$

CF848E Days of Floral Colours

Solution

$$
\begin{aligned}
f_1[i] =& g[i](i+1)^2 \\
&+ \sum_{j=0}^{i-1} g[j](j+1)^2 \times f_0[i-j-1] \\
&+ \sum_{j=0}^{i-3} g[j](j+2)^2 \times f_1[i-j-3]
\end{aligned}
$$

CF848E Days of Floral Colours

Solution

类似 f_1，我们再定义 f_2 表示长为 i 的一段，在其左右侧还有各有一对相对的异色花，染色方案的总分数。

$$
\begin{aligned}
f_2[i] = & g[i](i+2)^2 \qquad + \sum_{j=0}^{i-1} g[j](j+1)^2 \times f_1[i-j-1] \\
& + \sum_{j=0}^{i-3} g[j](j+2)^2 \times f_2[i-j-3]
\end{aligned}
$$

CF848E Days of Floral Colours

Solution

下面我们讨论环上的情况。我们固定一对相对的同色花为 1 与 $n+1$，并依靠将整个图形旋转得到其它解。为了不重复计数，我们还要枚举顺时针最近的下对相对的同色花是多少。

$$(g_{n-3}+g_{n-1})(n-1)^2+\sum_{i=2}^{n-2} i(i-1)^2(g[i-1]f_0[n-i-1]$$
$$+2g[i-2]f_1[n-i-2]+g[i-3]f_2[n-i-3])$$

求出 g 很容易，于是只要求出 f_0, f_1, f_2 我们就胜利了。

CF848E Days of Floral Colours

Solution

为了方便我们设 $G_i(x)=\sum_k (k+i)^2 g_k x^k$。$g$ 是已知的。
于是可以列出生成函数之间的方程（见下页）。

$$\begin{cases} F_0(x) = G_0(x) + xG_0(x)F_0(x) + x^3G_1(x)F_1(x) \\ F_1(x) = G_1(x) + xG_1(x)F_0(x) + x^3G_2(x)F_1(x) \\ F_2(x) = G_2(x) + xG_1(x)F_1(x) + x^3G_2(x)F_2(x) \end{cases}$$

解得

$$\begin{cases} F_0(x) = \dfrac{x^3G_1(x) - G_0(x)(x^3G_2(x) - 1)}{(xG_0(x) - 1)(x^3G_2(x) - 1) - x^4G_1(x)^2} \\ F_1(x) = \dfrac{G_1(x)}{(xG_0(x) - 1)(x^3G_2(x) - 1) - x^4G_1(x)^2} \\ F_2(x) = \dfrac{G_2(z) + xG_1(x)F_1(x)}{1 - x^3G_2(x)} \end{cases}$$

Solution

上面的东西可以 $O(n \log n)$ 求。
当然如果解出 $G(x)$、通过求导得到 $G_0(x), G_1(x), G_2(x)$，可以发现 F_0, F_1, F_2 都是两个低次多项式之比，于是也可以 $O(kn)$ 或者各种奇怪复杂度线性递推（最终答案是一个 16 项递推式）。

CF1349F1/F2 Slime and Sequences

题目

定义一个长为 n 的序列 p 是好的，当且仅当：对于任意序列中的数 $k>1$，都存在 $i,j(1\le i<j\le n)$ 满足 $p_i=k-1,p_j=k$。对于每个 $1\le k\le n$，统计长为 n 的所有好的序列中，k 一共出现了多少次。答案模 998244353。

$n\le 10^5$，3s。

CF1349F1/F2 Slime and Sequences Solution

Solution

首先做转化：长为 n 的好的序列可以与长为 n 的排列一一对应。

- 排列 $\rightarrow$ 序列：设排列为 a，则 p_{a_i} 为 $a[1..i]$ 中的上升个数 $+1$；证明它是好的：考虑 $a_i < a_{i+1}$，则 $p_{a_i} = k, p_{a_{i+1}} = k+1$，且满足 $a_i < a_{i+1}$；
- 序列 $\rightarrow$ 排列：找到序列中的所有 1，它们一定占据了排列最靠前的部分，且是各个下标降序放置。

“上升”指 $a_i < a_{i+1}$。

CF1349F1/F2 Slime and Sequences Solution

Solution

现在假设长为 $i+1$、已钦定 j 个上升的排列有 $d_{i,j}$ 个。我们将每个上升的左右两个数绑在一起，于是 $d_{i,j}$ 相当于把 $i+1$ 个不同的球放到 $i-j+1$ 个不同的盒子中，故

$$d_{i,j} = (i+1)![z^{i+1}](\exp(z) - 1)^{i-j+1}$$

注意这里我们 ~~为了留出足够多字母~~ ~~为了与官方题解一致~~ 按照数学习惯，将自变量写作 z。

现在我们考虑每个位置的贡献（式子见下页）：

$$
\begin{aligned}
ans_{i+1} &= \sum_{x=0}^{n-1} \binom{n}{x+1} (n-x-1)! \sum_{y=1}^{x} (-1)^{y-i} \binom{y}{i} d_{x,y} \\
&= \frac{n!}{i!} \sum_{x=0}^{n-1} \sum_{y=i}^{x} (-1)^{y-i} \frac{y!\, d_{x,y}}{(y-i)!(x+1)!} \\
&= \frac{n!}{i!} \sum_{y=i}^{n-1} \frac{(-1)^{y-i} y!}{(y-i)!} \sum_{x=y}^{n-1} \frac{d_{x,y}}{(x+1)!} \\
&= \frac{n!}{i!} \sum_{y=i}^{n-1} \frac{(-1)^{y-i} y!}{(y-i)!} \sum_{x=y}^{n-1} [z^{x+1}](\exp(z)-1)^{x-y+1}
\end{aligned}
$$

CF1349F1/F2 Slime and Sequences Solution

Solution

如果我们能够对每个 y 都求出 $\sum_{x=y}^{n-1}[z^{x+1}](\exp(z)-1)^{x-y+1}$，便可以一次卷积得到答案。

$$\sum_{x=y}^{n-1}[z^{x+1}](\exp(z)-1)^{x-y+1}=\sum_{x=y+1}^{n}[z^{x}](\exp(z)-1)^{x-y}$$

$$=[z^y]\sum_{x=y+1}^{n}\left(\frac{\exp z-1}{z}\right)^{x-y}=[z^y]\sum_{x=1}^{n-y}\left(\frac{e^z-1}{z}\right)^{x}$$

CF1349F1/F2 Slime and Sequences Solution

Solution

令 $F = \dfrac{\exp(z) - 1}{z}$，我们现在希望对每个 $0 \le i \le n-1$ 计算 $[z^i]\sum_{k=1}^{n-i} F(z)^i$。

$$[z^i]\sum_{k=1}^{n-i} F(z)^k = [z^i]\frac{1}{1-F(z)} - [z^i]\frac{F(z)^{n-i+1}}{1-F(z)}$$

求前半部分是 trivial 的。于是考虑怎么整后半部分。(可以化作 $[z^{n+1}]\frac{(zF)^{n-i+1}}{1-F}$)

CF1349F1/F2 Slime and Sequences Solution

构造 $W(z)=zF(z)$，$\Phi(z)$ 满足 $\dfrac{W(z)}{\Phi(W(z))}=z$。于是 $\dfrac{zF(z)}{\Phi(W(z))}=z$，$F(z)=\Phi(W(z))$。

$$[z^{n+1}]\frac{(zF(z))^{n-i+1}}{1-F(z)}=[z^{n+1}u^{n-i+1}]\sum_{k=0}^{\infty}\frac{(uzF)^k}{1-F}$$

$$=[z^{n+1}u^{n-i+1}]\frac{1}{1-\Phi(W(z))}\frac{1}{1-uW(z)}$$

接着开始（以 z 为主元）拉格朗日反演。

$$
\begin{aligned}
=&[u^{n-i+1}]\frac{1}{n+1}[z^n]((\frac{1}{1-\Phi(W(z))}\frac{1}{1-uW(z)})'\cdot\Phi(z)^{n+1})\\
=&\frac{1}{n+1}[z^nu^{n-i+1}](\Phi(z)^{n+1}\frac{u+\Phi'(z)-u\Phi(z)-uz\Phi'(z)}{(1-\Phi(z))^2(1-uz)^2})\\
=&\frac{1}{n+1}[z^nu^{n-i+1}](\Phi(z)^{n+1}(\frac{u}{(1-\Phi(z))(1-uz)^2}\\
&+\frac{\Phi'(z)}{(1-\Phi(z))^2(1-uz)}))
\end{aligned}
$$

$$
\begin{aligned}
=&\frac{1}{n+1}[z^n u^{n-i+1}](\Phi(z)^{n+1}(\frac{1}{1-\Phi(z)}\sum_{k=0}^{\infty}(k+1)z^k u^{k+1}\\
&+\frac{\Phi'(z)}{(1-\Phi(z))^2}\sum_{k=0}^{\infty}z^k u^k))\\
=&\frac{1}{n+1}[z^n](\Phi(z)^{n+1}(\frac{(n-i+1)z^{n-i}}{1-\Phi(z)}+\frac{\Phi'(z)z^{n-i+1}}{(1-\Phi(z))^2}))\\
=&\frac{1}{n+1}([z^i](\Phi(z)^{n+1}\frac{n-i+1}{1-\Phi(z)})+[z^{i-1}](\Phi(z)^{n+1}\frac{\Phi'(z)}{(1-\Phi(z))^2}))
\end{aligned}
$$

CF1349F1/F2 Slime and Sequences Solution

Solution

$\Phi(z)^{n+1}\dfrac{n-i+1}{1-\Phi(z)}$ 和 $\Phi(z)^{n+1}\dfrac{\Phi'(z)}{(1-\Phi(z))^2}$ 都可以计算。

至此我们完成了一道 EI 题。

UOJ #593. 新年的军队

题目

对于每个 $l(1 \le l \le n)$，统计所有【恰有 m 个位置满足 $p_i > p_{i+1}$ 的 $1 \sim n$ 的排列 p】中 $p_k = l$ 的方案数。$n \le 5 \times 10^5$，4s。

~~注意到字变小了吗?~~
后文有的部分是直接复制自 EI 的题解，并以这样的格式加入一定注解。

UOJ #593. 新年的军队. 问题转化

为了方便刻画排列的性质，我们将问题进行转化：

转化

对于全体满足 $\alpha_i > \alpha_{i+1}$ 的位置恰有 m 个的**实数数列** $\alpha_1, \cdots, \alpha_n$，计算 α_k 的**分布**。其中每个 α_i 在 $[0,1]$ 上均匀分布。

由对称性可知，按照 α 的排名分配的排列，每个排列都是等概率出现的。于是我们在考察其中一个特定实数的分布时，可以用类似于**概率密度函数**的工具来刻画。

概率密度函数

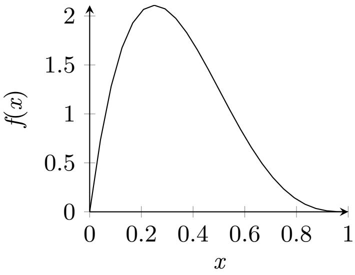
我们用一个函数 $f(x)$ 来刻画连续型随机变量 x 落在一个取值点附近的概率大小。随机变量的取值落在某个区域之内的概率则为概率密度函数在这个区域上的积分。

如果一个 n 阶排列中 α_k 的排名为 j，其对概率密度的贡献为：

$$\frac{x^{j-1}(1-x)^{n-j}}{(j-1)!(n-j)!}$$

这是因为当 $\alpha_k = x$ 时，有 $j-1$ 个数小于它，概率各是 x，而它们有 $(j-1)!$ 种排列方式；大于它的数同理。
之所以说是“贡献”是因为一共要 $n!$ 个这种东西加起来才是真正的概率密度函数。

下图以 5 阶排列中 α_k 排名为 2 为例，绘制了 α_k 的概率密度函数；可以发现，α_k 更有可能是比较小的数，这符合我们的直观感受。



转化为概率密度函数的好处在于，我们可以轻松刻画问题中出现的约束关系。假如原先 α_k 的概率密度函数为 $f(x)$，那么有：

- 添加实数 α_{n+1} 要求其 $> \alpha_k$ 那么新的关于 α_{n+1} 的概率密度为 $\int_0^x f(t)\mathrm{d}t$;
- 添加实数 α_{n+1} 要求其 $< \alpha_k$ 那么新的关于 α_{n+1} 的概率密度为 $\int_x^1 f(t)\mathrm{d}t$;
- 另有实数 $\beta_1, \cdots, \beta_m$ 关于 β_j 的概率密度为 $g(x)$，增添要求 $\alpha_k = \beta_j$，此时 α_k 的概率密度函数为 $f(x)g(x)$。

概率密度函数也是一个多项式。我们设排列中特定位置为 k 的方案数为 a_k，该位置的概率分布函数为 $f(x)$，可以给出转化关系：

$$f(x)=\sum_{x=0}^{n-1} f_k x^k=\sum_{k=0}^{n-1} a_k \frac{x^k(1-x)^{n-1-k}}{k!(n-k-1)!}$$

因此，「不难得到」其反演：

$$\sum_{k=0}^{n-1} \frac{a_k}{k!(n-1-k)!} x^k=\sum_{k=0}^{n-1} f_k x^k(1+x)^{n-1-k}$$

接下来，我们可以开始讨论怎么解决这个问题了。

概率密度函数也是一个多项式。我们设排列中特定位置为 k 的方案数为 a_k，该位置的概率分布函数为 $f(x)$，可以给出转化关系：

$$f(x)=\sum_{x=0}^{n-1} f_k x^k=\sum_{k=0}^{n-1} a_k \frac{x^k(1-x)^{n-1-k}}{k!(n-k-1)!}$$

因此，「不难得到」其反演：

$$\sum_{k=0}^{n-1} \frac{a_k}{k!(n-1-k)!} x^k=\sum_{k=0}^{n-1} f_k x^k(1+x)^{n-1-k}$$

接下来，我们可以开始讨论怎么解决这个问题了。
你觉得这个很难得到？先想几分钟，下一页给出答案。

上面那个式子大力转化一下：

$$
\begin{aligned}
\sum_{x=0}^{n-1} f_k x^k &= \sum_{k=0}^{n-1} a_k \frac{x^k(1-x)^{n-1-k}}{k!(n-k-1)!} \\
&= \sum_{k=0}^{n-1} \sum_{i=0}^{n-k-1} x^{i+k} \binom{n-k-1}{i} (-1)^i \frac{a_k}{k!(n-k-1)!} \\
&= \sum_{j=0}^{n-1} x^j \sum_{k=0}^{j} (-1)^{j-k} \binom{n-k-1}{j-k} \frac{a_k}{k!(n-k-1)!}
\end{aligned}
$$

也就是说：

$$
f_j = \sum_{k=0}^{j} (-1)^{j-k} \binom{n-k-1}{j-k} \frac{a_k}{k!(n-k-1)!}
$$

下面那个式子大力转化一下（中间有跳步）：

$$\begin{aligned}\sum_{k=0}^{n-1}\frac{a_k}{k!(n-1-k)!}x^k &= \sum_{k=0}^{n-1} f_k x^k (1+x)^{n-1-k} \\ &= \sum_{j=0}^{n-1} x^j \sum_{k=0}^{j} \binom{n-k-1}{j-k} f_k\end{aligned}$$

也就是说：

$$\frac{a_j}{j!(n-1-j)!} = \sum_{k=0}^{j} \binom{n-k-1}{j-k} f_k$$

我们对比一下两个式子：

$$f_j = \sum_{k=0}^{j} (-1)^{j-k} \binom{n-k-1}{j-k} \frac{a_k}{k!(n-k-1)!}$$

$$\frac{a_j}{j!(n-1-j)!} = \sum_{k=0}^{j} \binom{n-k-1}{j-k} f_k$$

记 $p_j = (-1)^j f_j$, $q_j = \dfrac{a_j}{j!(n-1-j)!}$，两式形如

$$p_j = \sum_{k=0} (-1)^k \binom{n-1-k}{j-k} q_k \Leftrightarrow q_j = \sum_{k=0} (-1)^k \binom{n-1-k}{j-k} p_k$$

这是一个二项式反演状物，大家敬请期待 TQX 讲容斥时证明这个式子。

UOJ #593. 新年的军队. Part 1

设 $F(u,v,t)$ 代表一个排列；用 u 的次数计量 < 的数量，用 v 的次数计量 > 的数量，用 t 计量概率密度，此时排列末尾的分布。可以列出方程：

$$F(t)=1+u\int_0^t F(\tau)\mathrm{d}\tau+v\int_t^1 F(\tau)\mathrm{d}\tau$$

进行微分，可得

$$F'=(u-v)F$$

「众所周知」，它的解为

$$F(t)=C\exp((u-v)t)$$

但 C 是包含 u,v 项的，我们需要定出其形式。将原式带入 $t=0,1$ 的情况，设 $I=\int_0^1 F(t)\mathrm{d}t$，

$$\begin{cases}1+vI=C\\1+uI=C\,\mathrm{e}^{(u-v)t}\end{cases}$$

解得：

$$\begin{cases} C &= \dfrac{u-v}{u-vl\,\mathrm{e}^{u-v}} \\ F &= \dfrac{(u-v)\mathrm{e}^{(u-v)t}}{u-v\,\mathrm{e}^{u-v}} \end{cases}$$

下一步我们要在一个位置左右两边各自粘上一个排列，在左边粘和在右边粘显然是相反的。

我们接下来让 x 的次数计量边数（$n-1$），y 计量 > 的数量。对于左侧，我们需要令 $u=x, v=xy$；对于右侧，我们需要令 $u=xy, v=x$。则有

$$L = \frac{(1-y)\,\mathrm{e}^{x(1-y)t}}{(1-y\,\mathrm{e}^{x(1-y)})}$$

$$R = \frac{(y-1)\,\mathrm{e}^{x(y-1)t}}{(y-\ \mathrm{e}^{x(y-1)})}$$

$$= \frac{(1-y)\,\mathrm{e}^{x(y-1)(t-1)}}{(1-y\,\mathrm{e}^{x(1-y)})}$$

相信大家还记得题面中希望求的位置是 k。设 $n_1 = k-1, n_2 = n-k$，我们需要提取 $[x^{n_1}]L$ 和 $[x^{n_2}]R$，「也就有」

$$\begin{aligned}[x^n]L &= (1-y)^{n+1}[x^n]\frac{\mathrm{e}^{xt}}{1-y\,\mathrm{e}^x} \\ &= (1-y)^{n+1}\sum_{j\geq 0} y^j [x^n]\mathrm{e}^{x(t+j)} \\ &= (1-y)^{n+1}\sum_{j\geq 0} y^j \frac{(t+j)^n}{n!}\end{aligned}$$

$$\begin{aligned}[x^n]R &= (1-y)^{n+1}[x^n]\frac{\mathrm{e}^{x(1-t)}}{1-y\,\mathrm{e}^x} \\ &= (1-y)^{n+1}\sum_{j\geq 0} y^j [x^n]\mathrm{e}^{x(j+1-t)} \\ &= (1-y)^{n+1}\sum_{j\geq 0} y^j \frac{(j+1-t)^n}{n!}\end{aligned}$$

~~大家可以思考一下这个“也就有”是怎么来的，我好再拖一会时。~~

这个「也就有」的正确性可以这样考虑:

$$
\begin{aligned}
&(1-y)[x^n]\frac{\mathrm{e}^{x(1-y)t}}{(1-y\,\mathrm{e}^{x(1-y)})}\\
=&(1-y)^{n+1}[((1-y)x)^n]\frac{\mathrm{e}^{x(1-y)t}}{(1-y\,\mathrm{e}^{x(1-y)})}\\
=&(1-y)^{n+1}[x^n]\frac{\mathrm{e}^{xt}}{1-y\,\mathrm{e}^{x}}
\end{aligned}
$$

就是说，每次 x 出现都一定跟 $(1-y)$ 一起，我们可以直接把与 x 一同出现的 $(1-y)$ 提取出来。

计算乘积：

$$
\begin{aligned}
&([x^{n_1}]L)\cdot([x^{n_2}]R)\\
=&\frac{1}{n_1!n_2!}(1-y)^{n_1+n_2+2}\left(\sum_{j\geq 0}y^j(t+j)^{n_1}\right)\left(\sum_{j\geq 0}y^j((j+1)-t)^{n_2}\right)
\end{aligned}
$$

我们要求有 m 个 >，因此我们只取这个乘积的 $[y^m]$ 项。我们考虑求出这个多项式 $f(t)$ 在 $0,\cdots,n$ 的点值，然后多点插值求出 f_i，再用上文的反演来求得答案。

我们发现对于 $\epsilon^{n_1}\cdot\eta^{n_2}$，它会以 $[y^m](1-y)^{n+1}y^{\epsilon+\eta-1}$ 的贡献给满足 $t\in[1-\eta,\epsilon]$ 的所有 $f(t)$。我们不妨做一个差分，就将所有 $\epsilon^{n_1}\cdot\eta^{n_2}$ 以 $[y^m](1-y)^{n+1}y^{\epsilon+\eta-1}$ 的 -1 贡献给 $t=\epsilon+1,+1$ 贡献给 $t=1-\eta$。这两个分别都是卷积。因此通过两次卷积就可以算出 $f(t)$ 的连续点值了。

复杂度 $O(n\log^2 n)$，期望得分 $70\sim 100$。

UOJ #593. 新年的军队. Part 2

插出点值这个过程其实并不是那么必要，我们考虑怎么直接计算系数。首先观察我们所求的和式

$$
\begin{aligned}
&[y^m]\frac{1}{n_1!n_2!}(1-y)^{n_1+n_2+2}\left(\sum_{j\geq 0} y^j(t+j)^{n_1}\right)\left(\sum_{j\geq 0} y^j((j+1)-t)^{n_2}\right)\\
=\ &\frac{(-1)^{n_2}}{n_1!n_2!}[y^m](1-y)^{n+1}\sum_{i\geq 0}\sum_{j\geq 0} y^{i+j}(t+i)^{n_1}(t-(j+1))^{n_2}\\
=\ &\frac{(-1)^{n_2}}{n_1!n_2!}\sum_{s=0}^{m}\left([y^m](1-y)^{n+1}y^s\right)\sum_{i=0}^{s}(t+i)^{n_1}(t+i-(s+1))^{n_2}
\end{aligned}
$$

里层的和是一个区间求和的形式 就是说 i 从 0 到 s 求和 (这和插值的前面步骤是一致的)，我们考虑拆成正方向的无穷和 $\sum_{i+}$ 指 $\sum_{i\geq 0}$ 以及一个差分 $\Delta_{s+1}f(t) = f(t) - f(t+s+1)$，设 $f_s = [y^m](1-y)^{n+1}y^s$ 没错这里 EI 又双叒叕用重复的符号了 就有：

$$
\begin{aligned}
&= \frac{(-1)^{n_2}}{n_1!n_2!}\sum_{s=0}^{m} f_s\Delta_{s+1}\left(\sum_{i+}(t+i)^{n_1}(t+i-(s+1))^{n_2}\right) \\
&= \frac{(-1)^{n_2}}{n_1!n_2!}\sum_{s=0}^{m} f_s\left(\sum_{i+}\Delta_{s+1}(t+i)^{n_1}(t+i-(s+1))^{n_2}\right) + C
\end{aligned}
$$

我们这里将 Δ 换到里面之后，无穷和的常数项其实就没法确定了。 差分算子放到里面的话会把多项式的常数项给差分掉；如果没有 $+C$ 的话求出来常数项会是 0。 我们发现这个变换后常数项对答案的影响恰好是给所有数加上一个量。我们知道答案之和为欧拉数 $\left\langle {n \atop m} \right\rangle$，于是只需要求出答案后整体偏移一定值即可。我们先不用管这个 C。

$$=\frac{(-1)^{n_2}}{n_1!n_2!}\sum_{i+}\left(\sum_{s=0}^{m} f_s\Delta_{s+1}(t+i)^{n_1}(t+i-(s+1))^{n_2}\right)+C$$

$$=\frac{(-1)^{n_2}}{n_1!n_2!}\sum_{i+}\left(\sum_{s=0}^{m} f_s\left[(t+i)^{n_1}(t+i-(s+1))^{n_2}-(t+i+(s+1))^{n_1}(t+i)^{n_2}\right]\right)$$
$$+C$$

根据 Taylor 公式，我们知道 $f(t+c)=\mathrm{e}^{c\mathrm{D}}f(t)$，其中 D 是对 t 的求导算子……

$$\begin{aligned} f(t+c) =& \mathrm{e}^{c\mathrm{D}}f(t) \\ =& \sum_i \frac{c^i}{i!}\mathrm{D}^i f(t) \\ =& \sum_i \frac{c^i}{i!} f^{(i)}(t) \end{aligned}$$

因而不妨设 $F(x)=\sum_s f_s x^{s+1}$，就有

$$
\begin{aligned}
&= \frac{(-1)^{n_2}}{n_1!n_2!} \sum_{i+} \left((t+i)^{n_1} F(\mathrm{e}^{-\mathrm{D}})(t+i)^{n_2} - (t+i)^{n_2} F(\mathrm{e}^{\mathrm{D}})(t+i)^{n_1} \right) + C \\
&= \frac{(-1)^{n_2}}{n_1!n_2!} \Sigma \left[\left(t^{n_1} F(\mathrm{e}^{-\mathrm{D}}) t^{n_2} - t^{n_2} F(\mathrm{e}^{\mathrm{D}}) t^{n_1} \right) \right] + C \\
&= \frac{(-1)^{n_2}}{n_1!n_2!} \int B(\mathrm{D}) \left[t^{n_1} F(\mathrm{e}^{-\mathrm{D}}) t^{n_2} - t^{n_2} F(\mathrm{e}^{\mathrm{D}}) t^{n_1} \right] + C, \quad B(t) = \frac{t}{1-\mathrm{e}^t}
\end{aligned}
$$

第一行，我们把内层求和号拆成两部分，把与 s 无关的部分提出来，有关的部分泰勒展开。
第二行，我们将其改写为有限微积分的形式。或者说，我们使用与 Δ 算子（差分算子）相对的 Σ 算子（逆差分算子/求和算子）。或者可以把它改写一下：

$$
\frac{(-1)^{n_2}}{n_1!n_2!} \sum_{t}^{\infty} \left[\left(\gamma^{n_1} F(\mathrm{e}^{-\mathrm{D}}) \gamma^{n_2} - \gamma^{n_2} F(\mathrm{e}^{\mathrm{D}}) \gamma^{n_1} \right) \right] \delta\gamma + C
$$

第三行是「众所周知」的。（见下一页）

(本页所用的 n, m, k 与原题无关。B_k 代表伯努利数，B 代表伯努利数的生成函数。)
第三行把求和转化成了积分，其中 B 是伯努利数的 EGF。伯努利数有性质：

$$\sum_{i=0}^{n-1} i^m = \frac{1}{m+1} \sum_{k=0}^{m} \binom{m+1}{k} B_k n^{m+1-k}$$

我们稍微推导一下相对简单的**自然数幂和**是如何通过伯努利数转化为积分的，原式可以类似得出。此处 D 作为对 n 求导的算子。

UOJ #593. 新年的军队. Part 2

$$
\begin{aligned}
&\sum_{i=0}^{n-1} i^m \\
=&\frac{1}{m+1}\sum_{k=0}^{m}\binom{m+1}{k}B_k n^{m+1-k} \\
=&\frac{1}{m+1}\sum_{k=0}^{m}\frac{B_k}{k!}\cdot(m+1)^{\underline{k}} n^{m+1-k} \\
=&\frac{1}{m+1}\sum_{k=0}^{m}\frac{B_k}{k!}\cdot \mathrm{D}^k n^{m+1} \\
=&\frac{1}{m+1}\sum_{k=0}^{m}B(\mathrm{D})n^{m+1} \\
=&\int B(\mathrm{D})n^m \mathrm{d}n
\end{aligned}
$$

UPD：讲课人忽然找到了一些关于伯努利数的内容。下面我们进行一些理性分析：
首先考虑我们有 $\Delta, \Sigma, \mathrm{D}, \int$ 这四个算子，且 $\Delta \cdot \Sigma = I, \mathrm{D} \cdot \int = 1$。
根据泰勒展开，我们有：

$$F(x+a) = F(x) + \frac{aF'(x)}{1!} + \frac{a^2 F''(x)}{2!} + \cdots$$

令 $a = 1$，有：

$$\begin{aligned}\Delta F(x) &= F(x+1) - F(x) \\ &= \frac{F'(x)}{1!} + \frac{F''(x)}{2!} + \cdots \quad = (\frac{\mathrm{D}}{1!} + \frac{\mathrm{D}^2}{2!} + \cdots)F(x) \\ &= (e^{\mathrm{D}} - 1)F(x)\end{aligned}$$

即 $\Delta = \mathrm{e}^{\mathrm{D}} - 1$，$\Sigma = 1/\Delta = 1/(\mathrm{e}^{\mathrm{D}} - 1) = B(\mathrm{D})/\mathrm{D} = \int B(\mathrm{D})$。

现在我们考虑如何快速计算 $F(\mathrm{e}^x)$ 希望大家还记得 $F(x) = \sum_s f_s x^{s+1}$，$f_s = [y^m](1-y)^{n+1}y^s$ ，这需要考虑 $F(x)$ 满足的微分方程。

$$
\begin{aligned}
F(x) &= \sum_{i=0}^{m} \binom{n+1}{m-i} (-1)^{m-i} x^{i+1} \\
&= xf(x) \\
(n-m+1+i)f_i &= -(m-i+1)f_{i-1} + [i=0]C \\
(n-m+1)f(x) + xf'(x) &= -mxf(x) + x^2 f'(x) + C \\
(n-m+1+mx)f(x) + (x-x^2)f'(x) &= C
\end{aligned}
$$

这里的 $f(x)$ 是以 f_s 为系数的多项式。

令 $G(x)=F(\mathrm{e}^x)=\mathrm{e}^x g(x)$，可得：

$$
\begin{aligned}
(n+1-m+m\mathrm{e}^x)g(x)+(1-\mathrm{e}^x)g'(x)&=C\\
((n-m)\mathrm{e}^{-x}+m+1)G+(\mathrm{e}^{-x}-1)G'&=C
\end{aligned}
$$

这个东西可以半在线卷积计算（相当于分治 FFT；在分治的每一层我们不只分两叉，而是分 $\log n$ 叉；复杂度为 $T(n)=bT(\frac{n}{b})+O(nb)+O(n\log\frac{n}{b})$，其中 b 为一层的分叉数)。复杂度 $O(\dfrac{n\log^2 n}{\log\log n})$。

算出 G 后，我们代回

$$
\frac{(-1)^{n_2}}{n_1!n_2!}\int B(\mathrm{D})\left[t^{n_1}F(\mathrm{e}^{-\mathrm{D}})t^{n_2}-t^{n_2}F(\mathrm{e}^{\mathrm{D}})t^{n_1}\right]+C
$$

最后我们只需要实现形如 $\Xi(\mathrm{D})\Psi(x)$ 的卷积，也就是求 $\sum_i \xi_i\Psi^{(i)}(x)$ 即可，这是一个卷积的形式。我们对着这个式子得到概率密度函数，再做一遍开头提到的反演 **即 可**。

完结撒花