STUDY REPORT of ANALYTIC COMBINATORICS

因为这是一个 **学习笔记**，所以会尽可能简洁，只会简要记录一些关键概念和核心例子，并且会夹杂一些主观见解。

STUDY REPORT of ANALYTIC COMBINATORICS
- I. Ordinary Generating Functions (OGF)
  - I.1 Symbolic Enumeration Methods
  - I.2 Admissible Constructions and Specifications
  - I.3 Integer Compositions and Partitions
  - I.4 Words and regular languages
  - I.5 Tree Structures
- II. Exponential Generating Functions (EGF)
- III. Multivariate Generating Functions

I. Ordinary Generating Functions (OGF)

I.1 Symbolic Enumeration Methods

解析组合主要用来解决这一类计数问题。有一些重要概念如 `组合对象`、`组合类`、`生成函数`.

Definition 1.1 组合对象：

> 组合对象是满足3某一条件的、可数的、可以定义 `大小` 且 `大小` 为非负整数的事物。组合对象记作一个小写字母（如 a），组合对象的大小记作 $|a|$。

Definition 1.2 组合类：

> 组合类是由组合对象组成的集合，记作这种字体的大写字母（如 $\mathcal{A}$），$\mathcal{A}_n$ 表示组合类中 `大小` 为 n 的组合对象构成的集合，$\text{card}(\mathcal{A}_n)$ 表示这个集合的大小；一般来说 $\text{card}(\mathcal{A}) = \aleph_0$。

Definition 1.3 计数序列：

> 我们称序列 $A\{A_n = \text{card}(\mathcal{A}_n)\}$ 为计数序列，它一般来说就是我们要寻找的答案。对于有着两个有着完全相同的计数序列的组合类 $\mathcal{A}$ 和 $\mathcal{B}$，我们称 $\mathcal{A} \cong \mathcal{B}$。

Definition 1.4 组合类导出生成函数（OGF）：

> 定义一个组合类的生成函数为 $T(z)$，其中 $T(z)$ 满足：
>
> $$T(z) = \sum_{a \in A} z^{|a|} = \sum_{i=0}^{+\infty} A[i] \cdot z^i, A_n = [z^n]T(z) \tag{1}$$

注意，上面的这个 z 不是未知数，而是一个组合对象，因此作为一种表示单独”物品“的运算符，不能以变量与函数的角度去看待生成函数的收敛性等；当然生成函数可以看做是形式幂级数，有着与幂级数相似的运算。

事实上你会发现这个序列仅仅保留了 `大小` 这一个信息，并将 `大小` 保存在了指数的位置；倘若我们关心的不只有这一个信息，那么我们可以引入其他的变量，称为多元生成函数；但是这个 `大小` 必须满足可加性（ 原文中称作 `admissibility` ，也可以译作可接受性、可容），然后才能定义我们的乘法运算。

Definition 1.5 组合结构：

> Let Φ be an m-ary construction that associates to any collection of classes $B^{(1)},\ldots B^{(m)}$ a new class
>
> $$A=\Phi[B^{(1)},B^{(2)},\ldots,B^{(n)}] \tag{2}$$
>
> The construction Φ is admissible iff the counting sequence A_n of A only depends on the counting sequences of Bs. For such an admissible construction, there then exists a well-defined operator Ψ acting on the corresponding ordinary generating functions:
>
> $$A(z)=\Psi[B^{(1)}(z),B^{(2)}(z),\ldots,B^{(n)}(z)] \tag{3}$$
>
> and it is this basic fact about admissibility that will be used throughout the book.

其中一个比较常见的例子就是 `笛卡尔积`。

Definition 1.6 笛卡尔积：

> 笛卡尔积 $A=B\times C$ 表示由 B 和 C 中各拿出一个组合对象组成一个有序对，这些有序对组成了组合类 A。
>
> $$A=B\times C\quad iff\quad A=\{\alpha=(\beta,\gamma),\beta\in B,\gamma\in C\} \tag{4}$$
>
> 其中有序对的 `大小` 定义为：
>
> $$|\alpha|_A=|\beta|_B+|\gamma|_C \tag{5}$$
>
> 于是 A 的 OGF 实际上是 B 和 C 的 OGF 的加法卷积：
>
> $$A_n=\sum_{i=0}^{n}B_iC_{n-i},\quad A=B\times C \tag{6}$$

其实对于加法也有类似的定义：

$$A=B\cup C,\quad |\alpha|_A=|\beta|_B\ or\ |\gamma|_C,\quad A=B+C \tag{7}$$

实际上，连乘可以看做是若干段按顺序拼起来的感觉，这个长度就是那个可以累加的大小。注意在做 OGF 相乘的时候千万不要忽略了这个顺序，这是我自己曾经的大误区。

连续性，Lipschitz和Hölder条件： gugugu，qyjj 的博客中有很好的表述。

I.2 Admissible Constructions and Specifications

回忆上一节中 `组合结构` 这一概念（`Admissible constructions`），这一节主要讲解一些简单的无标号组合结构和它们的形式化描述（`specifications`）。

Neutral object：代表乘法单位元、空集。

$$\epsilon = \{z^0\}, \mathcal{A} \cong \mathcal{A} \times \epsilon \cong \epsilon \times \mathcal{A} \tag{8}$$

Combinatorial sum (disjoint union)：我们如此定义与 OGF 加法同构的组合类加法：

$$\mathcal{B} + \mathcal{C} = (\epsilon_1 \times \mathcal{B}) \cup (\epsilon_2 \times \mathcal{C}) \tag{9}$$

Cartesian product：请见上节。

Sequence construction：类似由字符集 $\mathcal{A}$ 组成的字符串的一种构造：这是一个有序的过程，并且 $|\alpha| = \sum |\beta|$.

$$\mathrm{SEQ}(\mathcal{A}) = 1 + \mathcal{A} + \mathcal{A} \times \mathcal{A} + \mathcal{A} \times \mathcal{A} \times \mathcal{A} + \mathcal{A} \times \mathcal{A} \times \mathcal{A} \times \mathcal{A} + \cdots \tag{10}$$

Cycle construction： 把 SEQ 旋转后相同的去掉的这样一个组合类（ S 所有环置换 ）。

$$\mathrm{CYC}(\mathcal{A}) = (\mathrm{SEQ}(\mathcal{A}) \setminus \epsilon)/S \tag{11}$$

Cycle K construction： 把 SEQ 旋转 K 位后相同的去掉的这样一个组合类。

Multiset construction：组成元素相同的集合算一个（也就是无限背包）（ R 是所有置换构成的集合 ）。

$$\mathrm{MSET}(\mathcal{A}) = \mathrm{MSET}(\mathcal{A})/R \tag{12}$$

Powerset construction：`01` 背包。

形式化地：我们可以得到上面这些构造的生成函数表达（原文中使用了 `translate` 这个非常形象的词）：

$$\mathcal{A} = \mathcal{B} + \mathcal{C} \quad \Rightarrow \quad A(z) = B(z) + C(z) \tag{13}$$

$$\mathcal{A} = \mathcal{B} \times \mathcal{C} \quad \Rightarrow \quad A(z) = B(z) \times C(z) \tag{14}$$

$$\mathcal{A} = \mathrm{SEQ}(\mathcal{B}) \quad \Rightarrow \quad A(z) = \frac{1}{1 - B(z)} \tag{15}$$

$$\mathcal{A} = \mathrm{PSET}(\mathcal{B}) \quad \Rightarrow \quad A(z) = \begin{cases} \prod_{n=1} (1+z^n)^{B_n} \\ \exp\left(\sum B_n \sum_{i=1} \frac{(-1)^{-i}}{i} z^{in}\right) \\ \exp\left(\sum_{k=1} \frac{(-1)^{k-1}}{i} B(z^k)\right) \end{cases}$$

$$\mathcal{A} = \mathrm{MSET}(\mathcal{B}) \quad \Rightarrow \quad A(z) = \begin{cases} \prod_{n=1} (1-z^n)^{-B_n} \\ \exp\left(\sum B_n \sum_{i=1} \frac{z^{in}}{i}\right) \\ \exp\left(\sum_{k=1} \frac{B(z^k)}{i}\right) \end{cases}$$

$$\mathcal{A} = \mathrm{CYC}(\mathcal{B}) \quad \Rightarrow \quad \sum_{k=1}^{+\infty} \frac{\varphi(k)}{k} \log \frac{1}{1-B(z^k)} \tag{16}$$

现在我们就可以非常形式化地表述问题了，比如二进制串：

$$\mathcal{W} = \mathrm{SEQ}(\mathcal{A}) \quad where \quad \mathcal{A} \cong \mathcal{Z} + \mathcal{Z} \tag{17}$$

从而快速地得到 $\mathcal{W}$ 的生成函数 $W(z) = \frac{1}{1-2z}$。

事实上这个语义是可以递归的，就像我们在解卡特兰数的时候一样：

$$\mathcal{T} = \{\epsilon\} + \mathcal{T} \times \Delta \times \mathcal{T} \tag{18}$$

又比如在求解有根树个数时的：

$$\mathcal{G} = \mathcal{Z} \times \mathrm{SEQ}(\mathcal{G}) \quad \Rightarrow \quad G(z) = \frac{1}{2}(1 - \sqrt{1-4z}) = \sum_{n=1} \frac{1}{n} \binom{2n-2}{n-1} z^n \tag{19}$$

Specification： 设 $\mathcal{A}^{[n]}$ 表示 $sze \le n$ 的对象组成的组合类，则对于组合类 $(\mathcal{A}^{[1]}, \mathcal{A}^{[2]}, \cdots, \mathcal{A}^{[r]})$ 来说，`Specification` 定义为这样 r 个等式：

$$\begin{cases} \mathcal{A}^{[1]} = \Phi_1(\mathcal{A}^{[1]}, \mathcal{A}^{[2]}, \cdots, \mathcal{A}^{[r]}) \\ \mathcal{A}^{[2]} = \Phi_2(\mathcal{A}^{[1]}, \mathcal{A}^{[2]}, \cdots, \mathcal{A}^{[r]}) \\ \cdots \\ \mathcal{A}^{[r]} = \Phi_r(\mathcal{A}^{[1]}, \mathcal{A}^{[2]}, \cdots, \mathcal{A}^{[r]}) \end{cases}$$

其中这些 Φ 表示一个由上述无标号的各种组合构造和 $\mathcal{Z}$ 和 $\mathcal{E}$ 组成的 `construction`。

特别地，当 Φ_i 与任意 $\mathcal{A}^{[k]}(k > i)$ 无关，则称这个规则是非递归的，也就是这些类间的依赖关系是无环的。

如果这个Φ 不可以用上述的构造给出，那么我们称这个组合类是 `unconstructible` 或 `unspecificaion` 的。

反之，对于一个 `constructible` 的 `class`，它的生成函数可以用以下运算（符）表示：

$$
1,\ z,\ +,\ \times,\ Q,\ \mathrm{Exp},\ \mathrm{Log},\ \overline{\mathrm{Exp}}
\begin{cases}
\mathrm{Q}[f]=\dfrac{1}{1-f}\\
\mathrm{Exp}[f]=\exp\left(\sum_{k=1}\dfrac{f(z^k)}{i}\right)\\
\mathrm{Log}[f]=\sum_{k=1}^{+\infty}\dfrac{\varphi(k)}{k}\log\dfrac{1}{1-f(z^k)}\\
\overline{\mathrm{Exp}}[f]=\exp\left(\sum_{k=1}\dfrac{(-1)^{k-1}}{i}f(z^k)\right)
\end{cases}
$$

`Polya operator:` 上面这些长得像指数对数的称作 `Polya` 运算符。

I.3 Integer Compositions and Partitions

`Compositions:`

> 把整数 n 划分成 $x_1+x_2+\cdots+x_m=n,\forall i\in[1,m],x_i>0$ 的方案数， 把有理数集 $\frac{z}{1-z}$ SEQ 一下得出方案数是 2^{n-1}。

`Partitions:`

> 把整数 n 划分成 $x_1+x_2+\cdots+x_m=n,\forall i\in[1,m],x_i>0,x_{i-1}\le x_i$ 的方案数。
>
> 这个可以通过 EXP 得出，前几项是 $1,1,2,3,5,7,11,15,22$，有近似公式：
>
> $$P_n\sim\frac{1}{4n\sqrt{3}}\cdot\exp\left(\pi\cdot\sqrt{\frac{2n}{3}}\right)\tag{20}$$
>
> 就可以用于很多看似暴力做法的复杂度分析。n 可能在 50 左右。

事实上 P_n 有更快速的求法：（书中的做法比较傻）

$$z\frac{P'(z)}{P(z)}=\sum_{n=1}^{+\infty}\frac{nz^n}{1-z^n}\quad\texttt{implying}\quad nP_n=\sum_{j=1}^{n}\sigma_1(j)\cdot P_{n-j}\tag{21}$$

我们作为 `OIer` 的正确解法应该是：

$$P=\mathrm{Exp}\left(\frac{z}{1-z}\right)=\exp\left(\sum_{i=1}^{+\infty}\frac{\sigma_1(i)}{i}z^i\right)\tag{22}$$

话归正传，常规来说用 $\mathcal{L}$ 表示自然数类，用 $\mathcal{C}$ 和 $\mathcal{P}$ 来表示这两种类。显然有 $\mathcal{L}=\mathrm{SEQ}(\mathcal{Z}),\mathcal{C}=\mathrm{SEQ}(\mathcal{L}),\mathcal{P}=\mathrm{MSET}(\mathcal{L})$。当然也可以不依赖于自然数集，设 $\mathcal{C}^{\mathcal{T}}$ 表示数集 $\mathcal{T}$ 下的整数划分（ $\mathcal{P}^{\mathcal{T}}$ 也有类似定义，这两个也有着类似的解法，会比原来更加实用一些，例如斐波那契数列的生成函数 $\mathcal{C}^{\{1,2\}}$。

固定 $\mathcal{T}$ 的整数拆分：

来个 `Example`，求 $1\sim r$ 组成数 n 的方案数，那么答案就是：

$$C_n^{\{1,2,3,\cdots,r\}} = [z^n]\sum_j \left(\frac{z(1-z^r)}{1-z}\right)^j = \sum_{j,k}(-1)^k \binom{j}{k}\binom{n-rk+1}{j-1} \tag{23}$$

事实上我们有这样一个近似公式：$C_n^{\{1,\cdots,r\}} \sim c_r\rho^{-n}$，其中 c 和 ρ 是关于 r 的常数，例如 $r=2$ 时 ρ 为黄金比。

同理，有：

$$P_n^{\{1,2,3,\cdots,r\}} \sim c_r n^{r-1} \quad \texttt{with} \quad c_r = \frac{1}{r!\cdot(r-1)!}. \tag{24}$$

更进一步，有：

$$P_n^{\mathcal{T}} \sim \frac{1}{\tau}\cdot\frac{n^{r-1}}{(r-1)!} \quad \texttt{with}\ \tau := \prod_{n\in\mathcal{T}} n,\ r := \mathrm{card}(\mathcal{T}). \tag{25}$$

特别地：

$$P_n^{\{1,2\}} = \mathrm{int}\left(\frac{2n+3}{4}+\frac{1}{2}\right), \quad P_n^{\{1,2,3\}} = \mathrm{int}\left(\frac{(n+3)^2}{12}+\frac{1}{2}\right) \tag{26}$$

固定元素个数的整数拆分：

即使是组合意义也可以很快得到下面这个式子：

$$\mathcal{C}^{(k)} = \mathrm{SEQ}(\mathcal{T}) = \mathcal{T}^k,\ C_n^{(k)} = [z^n]\frac{z^k}{(1-z)^k} = \binom{n-1}{k-1} \tag{27}$$

问题在于下面这个怎么办？

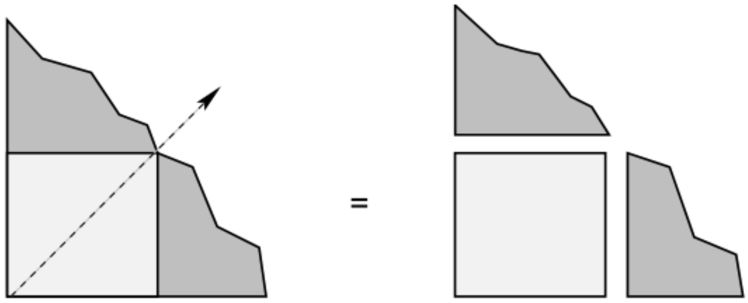
$$\mathcal{P}^{(\leq k)} = \mathrm{MSET}_{\leq k}(\mathcal{T}), \tag{28}$$

我们可以惊奇地发现：$\mathcal{P} \cong P_n^{\{1,2,3,\cdots,k\}}$，于是就愉快地得到：

$$P^{(\leq k)} = \sum_{m=1}^{k}\frac{1}{1-z^m} \tag{29}$$

这个同构就揭示了拆分数内部结构的巧妙，但是还有更妙的：

`The Durfee square`：一个拆分，我们把它画成折线图，显然一定会有一个正方形：

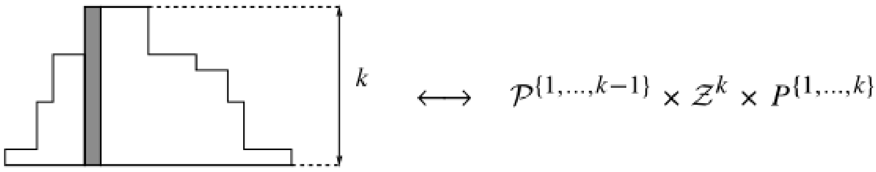


于是就可以看出下面这个等式：

$$\mathcal{P} = \sum_h \mathcal{Z}^h \times \mathcal{P}^{(\le h)} \times P^{\{1,2,3,\cdots,h\}} \tag{30}$$

`Stack polyominoes`：求 $\sum x_i = n$ 且 $x_1 \le x_2 \le \cdots \le x_k \ge x_{k+1} \ge \cdots \ge x_m$ 的方案数：

树形结合一下得到：



于是可以写出答案的生成函数：

$$S_k = \sum_{k\ge 1} \frac{z^k}{1-z^k} \cdot \left(P^{\{1,2,3,\cdots,h\}}\right)^2 \tag{31}$$

`Cyclic compositions`：在这时套用环构造会有奇效：

$$D(z) = \sum_{k=1}^{+\infty} \frac{\varphi(k)}{k} \log\left(1 - \frac{z^k}{1-z^k}\right) \tag{32}$$

$$D_n = -1 + \frac{1}{n} \sum_{k|n} \varphi(k) \cdot 2^{\frac{n}{k}} \sim \frac{2^n}{n} \tag{33}$$

`Euler’s pentagonal number theorem`：五边形数定理。

假设我们在求拆分数的时候不能取模怎么办呢？我们就不能求 log 了呀！考虑分析这个分母：

$$\prod (1 - z^k) \tag{34}$$

根据五边形数定理得：

$$\prod (1 - z^k) = \sum (-1)^k z^{k(3k+1)/2} \tag{35}$$

注意到在 $O(n)$ 以下只有 $O(\sqrt{n})$ 个地方有值，暴力求逆元就可以实现 $O(n\sqrt{n})$。

I.4 Words and regular languages

本章将会从 `Regular Specification` 和 `Finite Automaton` 两个角度来理解字符串计数问题。

S-Regular：能够被组合类的 `Specification` 的形式表示出来的就是字符串类，我们称它为 `S-Regular` 的。

事实上，这些 `S-Regular` 的字符串类都可以通过有理函数的形式表示出它的 `OGF`，而不需 $\ln$ 等。

例如，一个二进制串 $\mathcal{W}$，它的字符集是 $\mathcal{A}=\{a,b\}$，那么它的生成函数表示是：

$$\mathcal{W}=\operatorname{SEQ}(\mathcal{A}) \quad \Rightarrow \quad W=\frac{1}{1-a} \tag{36}$$

当然也可以这么得到如上的式子：

$$\mathcal{W}=\operatorname{SEQ}(a)\cdot\operatorname{SEQ}(b\cdot\operatorname{SEQ}(a)) \tag{37}$$

也可以限制某个字母的最长连续出现次数 `Longest runs`：

$$\mathcal{W}^{<k>}=a^{<k}\operatorname{SEQ}(b\cdot a^{<k}) \tag{38}$$

`Longest runs` 拓展到字符集大小为 m 的情况有：

$$W_n^{<k>}=[z^n]\frac{1-z^k}{1-mz+(m-1)z^{k+1}} \tag{39}$$

字符 a 恰好出现 k 次：

$$\mathcal{W}=\operatorname{SEQ}(b)\cdot(a\operatorname{SEQ}(b))^k \tag{40}$$

来个例题：求长度为 n 的 2 进制串中最长连续相同子串长度为 k 的概率，$n\le 1e5$。

定义 $\binom{n}{k}_{<d}$ 为 n 个字母，b 出现 k 次，b 出现间距小于等于 d 的二进制字符串类，则：

$$\mathcal{L}^{[d]}=\operatorname{SEQ}(a)\cdot(b\cdot\operatorname{SEQ}(a))^{k-1}\cdot b\cdot\operatorname{SEQ}(a) \tag{41}$$

如果两种字母都有最多连续出现次数的限制呢？

$$\mathcal{W}^{<p,q>}=\operatorname{SEQ}_{\le q}(b)\cdot\left(a\operatorname{SEQ}_{\le p-1}(a)\cdot b\operatorname{SEQ}_{q-1}(b)\right)\operatorname{SEQ}_{\le p}(a) \tag{42}$$

于是，最大连续出现次数为 k 的概率是：

$$\frac{1}{2^n}\cdot\left(W_n^{<k,k>}-W_n^{<k-1,k-1>}\right) \tag{43}$$

考虑怎么快速求 W，将 $W^{<k,k>}$ 大力展开得：

$$W_n^{<k,k>}=[z^n]\frac{1-z^{k+1}}{1-2z+z^{k+1}} \tag{44}$$

就可以 $O(n\log n)$ 解决问题了。

`Parterns` **问题**：给定一个模式串 p，问它作为子串 / 子序列在随机生成的长度为 n 的串的出现的概率 / 期望次数。

先考虑有作为子序列出现的概率，设 $\mathcal{L}$ 为满足条件的组合类，结合子序列自动机的思想得：

$$\mathcal{L}=\left(\prod_{i=1}^{|p|}\text{SEQ}(\mathcal{A}\setminus p_i)\cdot p_i\right)\cdot\text{SEQ}(\mathcal{A}) \tag{45}$$

那么出现概率为：

$$\frac{1}{m^n}[z^n]\frac{z^k}{(1-(m-1)z)^k}\cdot\frac{1}{1-mz} \tag{46}$$

那期望的出现次数的做法是：（这是一个比较巧妙的构造，很有推广价值）

$$\mathcal{L}=\left(\prod_{i=1}^{|p|}\text{SEQ}(\mathcal{A})\cdot p_i\right)\cdot\text{SEQ}(\mathcal{A}) \tag{47}$$

那么可以求出期望出现次数是：

$$\Omega_n=\frac{1}{m^n}[z^n]\frac{z^k}{(1-mz)^{k+1}}=m^{-k}\binom{n}{k} \tag{48}$$

和组合意义解法的答案相同。我们看它不出现的概率：

$$1-\frac{L_n}{m^n}=O\left(n^{k-1}\left(1-\frac{1}{m}\right)^n\right) \tag{49}$$

是以指数级与 n、k 相关的，所以会变得很小。

有限状态自动机：每条边标有字母的有向图，点成为状态，边成为转移。

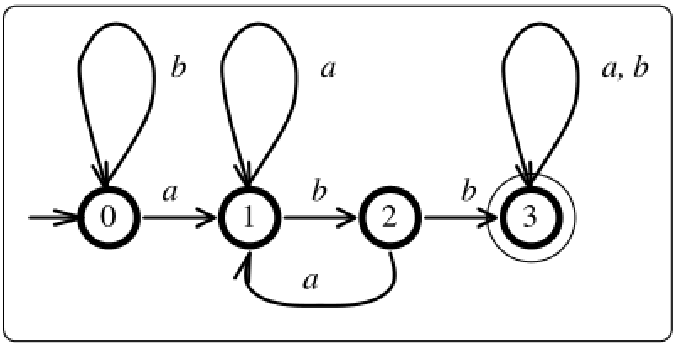
确定性有限状态自动机：确定性表示边 (u,c) 指向最多一个、确定的节点。

有限状态自动机是概率论中马尔科夫链的对应物，只不过加上了 `有限` 的要求。

接受：如果 `DFA` 存在一条从源点到汇点的路径使得这条路径和给定字符串相同，则称这个字符串可被接受。

用 $Q=\{q_0,q_1,\cdots,q_s\}$ 表示状态集合，$\bar{Q}$ 表示汇点集合。

例如：接受所有包含子序列 $\{a,b\}$ 的字符串的确定性有限状态自动机：(3 是汇点)



A-Regular：若一个字符串类存在一个恰好能接受它的 `DFA`，则称它为 `A-Regular`。

Equivalence theorem (Kleene - Rabin - Scott)：`A-Regular` 等价于 `S-Regular`。

这是一个非常震撼人心的定理，更震撼的是：

$$L(z)=\mathrm{u}(I-zT)^{-1}\,\mathrm{v} \tag{50}$$

这一步用逆矩阵的形式直接给出了它的 `OGF` 形式，而且出人意料地简洁有力，T（转移矩阵）中 $T_{i,,j}$ 表示点 i 到 j 的边的数量，$\mathrm{u}=(1,0,0,0,0,\cdots)$，$\mathrm{v}=(0\ or\ 1,\cdots)$ 分别表示每个节点是不是汇点。

证明：

> 对于每个节点，设以它开始的闭合子图的能接受的字符串类为 $\mathcal{L}_j$，则显然有以下 `Specification`：
>
> $$\mathcal{L}_j=\Delta_j+\sum_{\alpha\in\mathcal{A}}\{\alpha\}\mathcal{L}_{(q_j,\alpha)} \tag{51}$$
>
> 写成生成函数的形式：
>
> $$L_j(z)=[q_j\in\bar{Q}]+z\sum L_{(q_j,\alpha)}(z) \tag{52}$$
>
> 把所有的限制条件摆在一起形成一个线性系统 $\mathrm{L}(z)=(L_0(z),L_1(z),\cdots,L_s(z))$，则：
>
> $$\mathrm{L}(z)=\mathrm{v}+zT\,\mathrm{L}(z) \tag{53}$$
>
> 大力求逆元得：
>
> $$L(z)=\frac{\mathrm{v}}{1-zT} \tag{54}$$
>
> 那么答案就在第一项：$L_0=\mathrm{u}(I-zT)^{-1}\,\mathrm{v}$，证明完毕。

例如：上文中的子序列自动机，转换为形式化语言就是：

$$\begin{cases} L_0 = zL_1 + z_L 0 \\ L_1 = zL_1 + zL_2 \\ L_2 = zL_1 + zL_3 \\ L_3 = zL_3 + zL_3 + 1 \end{cases}$$

这时只要手动求出方程的解就可以了。

现在我们回归到字符串匹配问题，现在我们可以解决连续子串存在概率问题了：设 $\mathcal{T}$ 表示恰好以第 n 位出现了一次的字符串，$\mathcal{S}$ 表示前 n 个没有出现的字符串，p 是模式串，于是：

$$S + T = \{\epsilon\} + \mathcal{A} \times \mathcal{S} \tag{55}$$

现在只需要再添加一个方程就可以求出 S 和 T。考虑 `KMP` 自动机，设 c_i 表示最后 i 位是否是模式串的 `Border`，$c(z) = \sum c_i z^i$。考虑往 S 的后面添上 $|p| = m$ 个字母，那么有若干种情况——这恰好是第一次出现、可能这个字符串的一个 `Border` 可能与前面的 S 中的一段恰好能连成一段模式串，大概如下：

$$\begin{array}{llll} \mathfrak{p} \equiv & p_1 \cdots p_i & \boxed{p_{i+1} \cdots p_k} & \\ & & \boxed{p_1 \cdots p_{k-i}} & p_{k-i+1} \cdots p_k \quad \equiv \mathfrak{p}. \end{array}$$

翻译成形式化的语言就是：

$$S \times \{p\} = \mathcal{T} \times \sum_{c_i \neq 0} \mathrm{suf}_i \tag{56}$$

大意其实就是 $T_{n-i+1} \times z^{m-i}$，那么联立两个方程，得：

$$S + T = mzS, \quad s \cdot z^k = Tc(z) \tag{57}$$

解得：

$$S(z) = \frac{c(z)}{z^k + (1 - mz) \cdot c(z)} \tag{58}$$

这样就可以在 $n \log n$ 的时间内计算完成了；对于期望出现次数就比较简单：

$$\hat{O} = \mathrm{SEQ}(\mathcal{A}) \times p \times \mathrm{SEQ}(\mathcal{A}), \quad \hat{O} = \frac{z^k}{(1 - mz)^2}, \quad \Omega_n = m^{-k}(n - k + 1) \tag{59}$$

`Set partitions and Stirling partition numbers`：

考虑把一个字符串分进若干个集合的方案数，用类似子序列自动机的方法——枚举每个集合第一次出现的位置，得：

$$\mathcal{S}^{(r)} = b_1 \times \mathrm{SEQ}(b_1) \times b_2 \times \mathrm{SEQ}(b_1, b_2) \times \cdots \tag{60}$$

其中 $S^{(r)}$ 表示分成 r 段；写成生成函数为：

$$S^{(r)} = z^r \times \prod_{i=1}^{r} \frac{1}{1-iz} \tag{61}$$

使用 Ln 即可计算，如果手算出系数，为：

$$S^{(r)} = \frac{1}{r!} \sum_{j=0}^{r} \binom{r}{j} \frac{(-1)^{r-j}}{1-jz}, \quad S_n^{(r)} = \frac{1}{r!} \sum_{j=0}^{r} (-1)^{r-j} \binom{r}{j} \cdot j^n. \tag{62}$$

Circular words ：

考虑二进制串的情况，那么显然有：

$$\mathcal{N} = \mathrm{CYC}(\mathcal{A}), \quad N(z) = \sum_{k=0}^{+\infty} \frac{\varphi(k)}{k} \ln \frac{1}{1-2z^k} \tag{63}$$

很轻松可以写成通项形式：

$$N_n = \frac{1}{n} \sum_{k|n} \varphi(k) 2^{\frac{n}{k}} \tag{64}$$

就可以 $n \ln n$ 计算了。

Finite languages：$\mathcal{FL} = \mathrm{PSET}(\mathrm{SEQ}_{\geq 1}(\mathcal{A}))$。

I.5 Tree Structures

本章节会解决一类树相关计数问题。

拉格朗日反演：用于解包含多项式的方程，写作多项式复合逆的形式：

$$G(F(z)) = z \quad \Rightarrow \quad F_n = \frac{1}{n} [z^{-1}] \frac{1}{G^n(z)} \tag{65}$$

这样就能够在 $O(n \log n)$ 的时间内求出答案的某一项；如果这个解还需要进行其他运算才能得出解，则需要下面这个形式：

$$G(F(z)) = z \quad \Rightarrow \quad [z^n] H(F(z)) = \frac{1}{n} [z^{-1}] H'(z) \frac{1}{G^n(z)} \tag{66}$$

这是非常强大的武器，能够解决各种各样难以直接解开的方程。实际使用中请注意，在 $\pmod{z^n}$ 意义下，$[z^{-1}]$ 不等价于 z^{n-1}；需要转换成如下形式：

$$F_n = \frac{1}{n} [z^{n-1}] \left(\frac{x}{G(z)} \right)^n \tag{67}$$

II. Exponential Generating Functions (EGF)

III. Multivariate Generating Functions