大数数学简介

曹知秋

2025 年 6 月 8 日

本文是作者为《大数理论》撰写的导读材料，它利用较短的篇幅概要地阐述了《大数理论》的主线思想。文中凡以楷体排印的部分均为补充材料，它们涉及了一些更加深入和细致的内容。即使只读由宋体排印的正文部分，其内容也是自足的。

1 绪论

我们从一个简单的游戏开始：请在纸上写下一个尽可能大的自然数，越大越好。

初看起来，只要一直在纸上不停地写 9，那么这个数字要多大就可以有多大。这样做是没有意义的，为了解决这一问题，我们现在引入一些额外的规则。

(1) 我们必须预先给定一个所用字符数的上限，例如 1000 个字符或者 10000 个字符。这个字符数上限是多少都可以，但是一旦给定就不能再更改了，并且写下的这个数所用的字符数不能够超过这个上限。

(2) 不能写无穷大，而只能写有限的数字。

(3) 在写的过程中可以引入一些自定义的符号或者是函数，但是其定义必须是明确的。也就是说，写在纸上的表达式应当能够唯一地将这个自然数确定下来。

以上就是这个游戏的全部规则，现在让我们简单地尝试一下用 10 个字符所能够写出的大数。很容易就可以想到，可以写下一个由 10 个 9 构成的十进制数，即

$$9999999999. \tag{1}$$

但是这并不是能够写下的最大数字。如果进一步思考，那么我们会发现可以利用指数表示法写下下面这个数字

$$9^{999999999}. \tag{2}$$

当然，我们还不会止步于此，事实上可以进一步将这个数字的指数也进一步地表示为指数的形式，这样重复下去，我们将得到一个 10 层的指数塔

$$9^{9^{9^{9^{9^{9^{9^{9^{9^{9}}}}}}}}}}. \tag{3}$$

如果没有更好的数字表示方法，那么这就是我们所能够写出的最大数字了。需要注意的是，指数塔是从上到下计算的，即先算上层的指数塔，再算下层的指数塔，例如

$$9^{9^9} = 9^{(9^9)} = 9^{387420489}. \tag{4}$$

然而请注意，我们的规则之中允许我们引入自定义的符号或者是函数，只要它的规则是确定的，并且根据它可以唯一地确定下一个自然数即可。因此现在让我们尝试用一个变量来表示指数塔的层数，并将其表示为

$$\underbrace{9^{9^{\cdots^{9^9}}}}_{9^9\text{个}9}. \tag{5}$$

如果把大括号算作一个字符的话，那么这个表达式正好有 10 个字符，并且它唯一地确定下了一个大数。它是一个完全由 9 构成的指数塔，其层数为 $9^9 = 387420489$。这当然已经远远超过此前所能够写出的数字了。

上述的例子可以给我们一些启发。我们也许立刻会想到，可以在表达式中引入更复杂的结构，这样所能表示的数字就可以极大地增加。现在我们可以写下包含 1000 个字符、10000 个字符乃至更多个字符的表达式，但要注意，这里所能容许的总字符数不论多大，它仍然是有限的。因此我们仍然不能够采用“暴力”的方式去进行无限制的堆叠，而必须在有限的空间中写下更加精巧的结构。

大数数学 (Googology) 就是系统性地研究如何构造非常巨大（但是有限）的自然数的数学分支。通过对大数的研究，我们能够对自然数的结构具有更加深刻的认识。那么，现在让我们踏上通往大数世界的旅程吧。

2 超运算

下面我们只考虑以自然数作为变量的记号。一个自然的想法是，既然将加法重复多次可以得到乘法，将乘法重复多次可以得到乘方，那么将乘方重复多次也可以得到某些更加强大的东西。如果沿着这条路走下去，那么我们就得到了超运算。现在我们将加法 $a+b$ 称为第一级运算。将加法重复多次就得到了乘法 $a \times b$，我们称之为第二级运算

$$a \times b = \underbrace{a + a + \ldots + a}_{b\text{个}a}. \tag{6}$$

这里 a 代表参与加法运算的那个数，而 b 代表加法重复的次数。进一步，将乘法重复多次就得到了乘方 a^b。在大数数学中，我们引入一个新的符号，称为高德纳 (Knuth) 箭头，它定义为 $a \uparrow b = a^b$。采用这一记号的优点是它可以对称地表示运算中的两个数，因此我们可以将乘方表示为

$$a \uparrow b = \underbrace{a \times a \times \ldots \times a}_{b\text{个}a} = a^b. \tag{7}$$

这就是第三级运算，这里 a 代表参与乘法的那个数，而 b 代表乘法重复的次数，$\uparrow$ 代表第三级运算（乘方）的运算符。

那么，第三级运算如何推广为第四级运算呢？仔细观察前面的两个表达式，我们可以发现后一级运算都是把前一级运算重复多次得到的。因此，我们现在引入一个新的运算符 $\uparrow\uparrow$ 来表示第四级运算，它就是将第三级运算（乘方）重复多次得到的结果

$$a \uparrow\uparrow b = \underbrace{a \uparrow a \uparrow \ldots \uparrow a}_{b\text{个}a} = \underbrace{a^{a^{\cdots^{a}}}}_{b\text{个}a}. \tag{8}$$

这里 a 代表参与第三级运算（乘方）的那个数，而 b 代表第三级运算重复的次数。需要注意的是，我们约定超运算都是右结合的，即

$$a \uparrow b \uparrow c = a \uparrow (b \uparrow c), \tag{9}$$

而不是 $(a \uparrow b) \uparrow c$。

以此类推，第五级运算的运算符可以定义为三箭头，它是第四级运算（双箭头）的重复

$$a \uparrow\uparrow\uparrow b = \underbrace{a \uparrow\uparrow a \uparrow\uparrow \ldots \uparrow\uparrow a}_{b\text{个}a}. \tag{10}$$

第六级运算的运算符可以定义为四箭头，它是第五级运算（三箭头）的重复

$$a \uparrow\uparrow\uparrow\uparrow b = \underbrace{a \uparrow\uparrow\uparrow a \uparrow\uparrow\uparrow \ldots \uparrow\uparrow\uparrow a}_{b\text{个}a}. \tag{11}$$

现在我们用 $a \uparrow^n b$ 来代表 a 和 b 中间有 n 个箭头的运算，那么一般地，$(n+1)$ 箭头的运算就是多个 n 箭头的运算的重复

$$a \uparrow^{n+1} b = \underbrace{a \uparrow^n a \uparrow^n \ldots \uparrow^n a}_{b\text{个}a}. \tag{12}$$

这样我们就得到了超运算的定义。

递归定义 上述超运算可以进一步地用如下三个公式进行递归定义

$$a \uparrow^1 b = a^b, \tag{13}$$

$$a \uparrow^n 1 = a, \tag{14}$$

$$a \uparrow^{n+1} (b+1) = a \uparrow^n \left(a \uparrow^{n+1} b\right). \tag{15}$$

在上述递归定义中，我们给出了从较大的表达式变换为较小的表达式的方法。重复应用式 (15) 一共 b 次，直到其第二个变量变为 1，然后应用式 (14)，我们将得到式 (12)，这和我们此前给出的定义是等价的。

下面我们对超运算的强度进行一个简要的说明。考虑对两个 3 进行的超运算，我们有

$$3 \uparrow 3 = 3^3 = 27, \tag{16}$$

$$3 \uparrow\uparrow 3 = 3 \uparrow 3 \uparrow 3 = 3^{3^3} \approx 7.63 \times 10^{12}. \tag{17}$$

再增加一个箭头，我们将得到

$$3 \uparrow\uparrow\uparrow 3 = 3 \uparrow\uparrow 3 \uparrow\uparrow 3 = \underbrace{3^{3^{\cdots^3}}}_{7.63\times10^{12}\text{个}3}. \tag{18}$$

这就是说，它是一个具有 7.63×10^{12} 层的指数塔。如果将这个指数塔写下来的话，那么它可以直接从地球写到太阳。而如果我们考虑一个四箭头的大数 $3 \uparrow^4 3$，那么它将可以表示为一个巨大的指数塔，这指数塔的层数同样是一个指数塔，其层数又是一个指数塔，……，这个过程总共要重复 $3 \uparrow\uparrow\uparrow 3$ 次，最后才能够得到一个 3（这个重复的次数本身就是一个足以从地球写到太阳的超大指数塔了）。超运算记号的强大可见一斑。

3 增长率

一般的大数记号中通常含有多个变量，这是不便于研究的。为系统性地描述各个大数记号的强度，我们现在在大数记号之中只保留一个变量 n，其余的量设置为常数，这样我们将得到一个以正整数 n 为变量的函数（数列）。例如，含有 m 个箭头的超运算所对应的函数可以选取为

$$f(n) = n \uparrow^m n, \tag{19}$$

其中 m 是一个常数，n 是一个正整数。容易发现，一个记号所对应的函数增长得越快，它所对应的大数记号也就越强大。假如我们能够找到一个标尺来标记这个函数的强度，那么我们也就能够定量地去分析和认识一个大数记号了。

现在让我们把不同的超运算按照其增长快慢从小到大排成一列，然后从 0 开始对它们进行编号

$$\begin{array}{ccccccc} n+1 & n+n & n\uparrow^1 n & n\uparrow^2 n & n\uparrow^3 n & \ldots \\ 0 & 1 & 2 & 3 & 4 & \ldots \end{array} \tag{20}$$

这里的编号就称为函数的增长率。利用上述增长率的记号，我们就为函数的增长速度找到了一个标尺。每将前一个函数迭代 n 次，我们就得到了一个增长更快的函数，其增长率增加 1。值得说明的是，增长率是一个宽泛的概念。事实上每个相同的增长率都对应着一大批增长速度并不尽相同的函数，构造具有某一特定增长率的函数的方式也不是唯一的。

然而，上述讨论还远远没有穷尽所有可能的大数函数。现在让我们明显地构造一个增长速度超过所有有限箭头数的 $n\uparrow^m n$ 的函数。我们将所有 $n\uparrow^1 n, n\uparrow^2 n, n\uparrow^3 n$ 等数列中的所有元素排成一个表格，如下所示

$$\begin{array}{llll} 1\uparrow^1 1 & 2\uparrow^1 2 & 3\uparrow^1 3 & \dots \\ 1\uparrow^2 1 & 2\uparrow^2 2 & 3\uparrow^2 3 & \dots \\ 1\uparrow^3 1 & 2\uparrow^3 2 & 3\uparrow^3 3 & \dots \\ \dots & \dots & \dots & \dots \end{array} \tag{21}$$

然后我们选取这个表格中的对角元素构成一个新数列

$$\{1\uparrow^1 1,\ 2\uparrow^2 2,\ 3\uparrow^3 3,\ \dots\}, \tag{22}$$

它实际上就是

$$f(n) = n\uparrow^n n. \tag{23}$$

我们把上述 $n\uparrow^n n$ 构造的过程称为“对角化”。与此前的函数不同的是，它的箭头个数现在也变成了一个变量。将其与拥有固定箭头数的函数 $n\uparrow^m n$ 相比，我们发现不论这里的 m 有多大，在 n 足够大的情况下，$n\uparrow^n n$ 的增长速度都要超过 $n\uparrow^m n$。因此，如果将 $n\uparrow^n n$ 放在函数增长速度的标尺中，那么它将排在所有具有有限箭头数的 $n\uparrow^m n$ 之后，这种情况类似于指数函数的增长速度超越了一切多项式函数一样。看起来，上述函数标尺还要继续延伸下去。

既然这个标尺能够继续延伸下去，那么我们自然就要问，在这个标尺之上，函数 $n\uparrow^n n$ 的增长率应当为多少？我们仍仿照此前的做法，按照增长速度的快慢从小到大为这一系列数列进行标号。但是我们很快就发现了困难：我们在描述具有有限个箭头数的 $n\uparrow^m n$ 增长率的过程中已经耗尽了所有的自然数。既然如此，那么 $n\uparrow^n n$ 的增长率应当用什么来标记呢？

$$\begin{array}{cccccccc} n & n+n & n\uparrow^1 n & n\uparrow^2 n & \dots & n\uparrow^{99} n & \dots & n\uparrow^n n & \dots \\ 0 & 1 & 2 & 3 & \dots & 100 & \dots & ? & \dots \end{array} \tag{24}$$

为了解决这一问题，我们需要引入一个新的“数”，它能够排在所有的自然数之后，我们将用它来为 $n\uparrow^n n$ 进行标号。我们把这个“数”称为 ω，它就是 $n\uparrow^n n$ 的增长率

$$\begin{array}{cccccccc} n & n+n & n\uparrow^1 n & n\uparrow^2 n & \dots & n\uparrow^{99} n & \dots & n\uparrow^n n & \dots \\ 0 & 1 & 2 & 3 & \dots & 100 & \dots & \omega & \dots \end{array} \tag{25}$$

在数学上，我们称 ω 这样的数为“序数”。序数是自然数的扩展，直观地说它表明了某个东西在一个已经排好顺序的队列中所处的位置，即“第几个”。如果愿意的话，我们可以说 $n\uparrow^n n$ 在上述函数增长率的标尺之中排到了“第 ω 个”（尽管这一说法是不严格的）。

从 ω 出发，我们可以得到一系列增长速度更快的数列。例如，我们可以将超运算的箭头数量作为变量再次迭代 n 次。由于将函数迭代 n 次，其增长率将增加 1，因此这样得到的就是一个增长率为 $\omega+1$ 的函数。这里的 $\omega+1$ 也是一个序数，它就是排在 ω 之后的下一个序数。进一步将增长率为 $\omega+1$ 的函数迭代 n 次，我们将得到一个增长率为 $\omega+2$ 的函数，然后又有增长率为 $\omega+3, \omega+4, \dots$ 的函数，以此类推。

为了进一步超越所有增长率为 $\omega+m$ 的函数，我们可以将所有增长率为有限的 $\omega+m$ 的函数进行对角化。这样得到的函数的增长率需要一个新的序数来刻画，我们将其定义为 $\omega\cdot 2 = \omega+\omega$。当然，这一过程还可以无限进行下去，我们能够得到一系列增长率为 $\omega\cdot 3, \omega\cdot 4, \dots$ 的函数。将这一系列函数对角化可以得到一个增长速度更快的函数，其增长率定义为 $\omega^2 = \omega\cdot\omega$。进一步又有增长率为 $\omega^3, \omega^4, \dots$ 的函数，直到增长率为 ω^ω，乃至 $\omega^{\omega^\omega}, \omega^{\omega^{\omega^\omega}}$ 的函数。

对所有增长率为 ω 指数塔的函数进行对角化，我们将得到一个增长率更大的函数，我们将其所对应的增长率序数定义为 ε_0。以此类推，我们还可以继续进行下去。于是，现在大数记号的增长率标尺就不仅仅由自然数刻画，而是由一系列越来越大的序数来进行刻画

$$0,\ 1,\ 2,\ \dots,\ \omega,\ \omega+1,\ \dots,\ \omega\cdot 2,\ \dots,\ \omega^2,\ \dots,\ \omega^\omega\ \dots,\ \varepsilon_0,\ \dots \tag{26}$$

序数的结构是复杂的，但是总体上可以表现为两种规则：

(1) 对于任意一个序数 α，总存在一个序数 $\alpha' = \alpha + 1$ 紧随其后。

(2) 对于任意一系列序数 $\{\alpha_0, \alpha_1, \alpha_2, \ldots\}$,总存在一个最小的序数 $\alpha = \sup\{\alpha_0, \alpha_1, \alpha_2, \ldots\}$（这里 sup 代表上确界，即大于等于所有 α_n 的最小序数），它大于（或等于）上述所有的序数。

除了 0 之外，所有的序数都满足以上两种情况之一。我们称第一种序数为后继序数，第二种序数为极限序数。对于极限序数 $\alpha = \sup\{\alpha_0, \alpha_1, \alpha_2, \ldots\}$ 来说，我们称序数序列 $\{\alpha_0, \alpha_1, \alpha_2, \ldots\}$ 为其基本列，将基本列的第 n 项记为 $\alpha[n] = \alpha_n$，注意基本列中项的编号都是从 0 开始的。因此，通过取后继和取极限的方式，我们就可以在序数世界中不断地前进了。

而上述生成序数的两种方式，恰好对应于我们前文中介绍的两种构造大数函数的方式：

(1) 对函数迭代 n 次，可以得到一个比该函数增长速度更快的函数，其增长率 +1。

(2) 对一系列函数进行对角化，可以得到一个比这一系列函数增长速度都更快的函数，它的增长率是这一系列函数增长率的极限序数。

反复应用以上两种方式，我们就可以得到越来越强大的大数函数。

序数的集合论定义 不失一般性地，我们考虑一个集合，并在集合上建立起一个二元关系 $<$。我们称二元关系 $<$ 为一个良序，如果它满足如下四条性质：(1) 对任意 x，$x < x$ 均不成立；(2) 若 $x < y, y < z$，则 $x < z$；(3) 对任意 x, y，要么 $x < y$，要么 $y < x$，要么 $x = y$；(4) 对集合中的任意一个子集来说，都存在该序下的一个最小元素。直观地说，一个良序关系就是对集合中所有元素的排序 (比大小)，$x < y$ 即代表 x 排在 y 之前 (或 x 比 y 小)。同时我们还要求其任意一个子集中总要有一个最小元，即不能够出现无穷递降的情况。例如，自然数在通常的大小关系下就构成了一个良序，而整数在通常的大小关系下则不是一个良序。

序数是一种特殊的集合，它在 $\in$ 下良序，且其所有元素均为其子集。我们定义

$$0 = \varnothing, \quad \alpha + 1 = \alpha \cup \{\alpha\}, \tag{27}$$

对极限序数有

$$\alpha = \sup\{\alpha_n\} = \bigcup_n \alpha_n. \tag{28}$$

一个带有良序关系的集合称为良序集。可以证明，任何一个良序集都与一个序数同构，即可以建立起保持序关系不变的一一映射。因此，任意一个良序关系都可以用一个序数来刻画。我们所构造的一系列函数可以按其增长速度排成一个良序，因此它也可以用序数来刻画，我们就将其所对应的序数称为增长率。

4 快速增长层次

根据前文中的讨论我们看出，函数的增长率结构实际上与序数的结构是相对应的，这一点在快速增长层次之中得到了更为完整的体现。

快速增长层次 (Fast Growing Hierarchy, FGH) 是一族以序数 α 标记的函数 $f_\alpha(n)$，它为每个序数 α 都提供了一个快速增长的函数。首先，对于序数 0 我们定义

$$f_0(n) = n + 1, \tag{29}$$

这是 FGH 的起点。根据我们之前的讨论，每将函数迭代 n 次，其增长率增加 1，因此我们对后继序数 $\alpha + 1$ 定义

$$f_{\alpha+1}(n) = f_\alpha^n(n) = \underbrace{f_\alpha(f_\alpha(\ldots f_\alpha(n)\ldots))}_{n\text{层}}, \tag{30}$$

式中上标 n 代表将函数 f_α 复合 n 次。例如，从 0 开始将函数逐渐复合，我们可以得到

$$f_0(n) = n+1, \quad f_1(n) = 2n, \quad f_{m+1}(n) \approx 2 \uparrow^m n. \tag{31}$$

由于极限序数 ω 的基本列为 $\{0,1,2,\ldots\}$，因此为了得到 $f_\omega(n)$，我们需要对所有有限的 $f_m(n)$ 进行对角化。现在将这一系列的 $f_m(n)$ 排成一个表格，并取其对角元素

$$\begin{array}{cccc} f_0(0) & f_0(1) & f_0(2) & \ldots \\ f_1(0) & f_1(1) & f_1(2) & \ldots \\ f_2(0) & f_2(1) & f_2(2) & \ldots \\ \ldots & \ldots & \ldots & \ldots \end{array} \tag{32}$$

事实上这里所选取的一系列序列的下标恰好为 $\omega = \sup\{0,1,2,\ldots\}$ 的基本列

$$\begin{array}{cccc} f_{\omega[0]}(0) & f_{\omega[0]}(1) & f_{\omega[0]}(2) & \ldots \\ f_{\omega[1]}(0) & f_{\omega[1]}(1) & f_{\omega[1]}(2) & \ldots \\ f_{\omega[2]}(0) & f_{\omega[2]}(1) & f_{\omega[2]}(2) & \ldots \\ \ldots & \ldots & \ldots & \ldots \end{array} \tag{33}$$

因此我们得到

$$f_\omega(n) = f_{\omega[n]}(n) = f_n(n) \approx 2 \uparrow^{n-1} n. \tag{34}$$

这确实是一个增长率为 ω 的函数。

进一步将 f_ω 复合 n 次，我们将得到 $f_{\omega+1}(n) = f_\omega^n(n)$。这样不断重复下去，我们将得到一系列的序列 $f_{\omega+m}(n)$。要得到 $f_{\omega\cdot 2}$，我们需要将所有的 $f_{\omega+m}$ 对角化，而这一系列序列的下标恰好为 $\omega\cdot 2$ 的基本列

$$f_{\omega\cdot 2}(n) = f_{\omega\cdot 2[n]}(n) = f_{\omega+n}(n). \tag{35}$$

于是又有一系列的 $f_{\omega\cdot m}(n)$，直到将上述所有序列再次对角化得到

$$f_{\omega^2}(n) = f_{\omega^2[n]}(n) = f_{\omega\cdot n[n]}(n) = f_{\omega\cdot(n-1)+n}(n). \tag{36}$$

不断重复上述过程，最后可以得到 $f_{\varepsilon_0}(n)$。

一般地，对于极限序数 $\alpha = \sup\{\alpha[0],\ \alpha[1],\ \alpha[2],\ \ldots\}$ 来说，f_α 定义为所有 $f_{\alpha[n]}$ 序列的对角化

$$f_\alpha(n) = f_{\alpha[n]}(n). \tag{37}$$

需要注意的是，这里的每个 $\alpha[n]$ 也都是一个序数。式 (29),(30),(37) 共同给出了 FGH 的定义，反复应用上述三条递推公式，我们就能够得到序数 α 所对应的函数 $f_\alpha(n)$。

基本列 对于 ε_0 之前的序数来说，定义标准形式如下：0 是标准形式；如果 α,β 是标准形式，而且 $\alpha \geq \beta$，那么 $\alpha+\beta$ 是标准形式；如果 α 是标准形式，且 $\omega^\alpha > \alpha$，那么 ω^α 是标准形式。在化成标准形式后，对于极限序数 α 和任意序数 β 来说，定义标准基本列如下

$$(\beta+\alpha)[n] = \beta + \alpha[n], \quad \omega^{(\alpha+1)}[n] = \omega^\alpha \cdot n, \quad \omega^\alpha[n] = \omega^{\alpha[n]}. \tag{38}$$

这样我们就给出了 ε_0 以下序数的标准基本列。只有给定了基本列的选取方法后，FGH 才是完整的。

增长率的困难 从逻辑上说，我们是先有 FGH，然后再以 FGH 作为基准定义函数的增长率。不过在考虑函数的增长率时，我们实际上使用的是下面这两条性质：(1) 迭代 n 次增长率加 1；(2) 对角化增长率取极限。或者说，以上两条性质是对我们一个良好的增长率定义的期望。然而遗憾的是，我们目前还无法找到一个严格的增长率

定义，它可以满足上述两条性质。或许函数的增长率结构比我们朴素的想像要更加复杂。

除此之外，我们也缺乏分析函数增长率的手段。目前基本只能通过枚举记号中的大量节点来进行不完全的归纳，而这一做法是不可靠的，它在相当大的程度上依赖于分析者的主观判断。这一问题仍然需要通过进一步的研究来解决。

FGH 给出了一种能够将大序数映射为快速增长的函数的方式，它所考虑的序数 α 越大，那么相应的函数 $f_\alpha(n)$ 的增长速度就越快。ε_0 当然不是序数的终点，通过进一步地取后继和极限，我们可以构造出一系列更加庞大的序数。因此，如果我们能够写出更大的序数，那么它也可以帮助我们构建更加强大的大数记号。这就像是在大数的世界中前进，我们感到举步维艰；但是一旦我们能够进入序数世界，那么我们前进起来就要容易得多了。（例如，用序数写出 ε_0 是容易的，而用大数记号写出一个增长率为 ε_0 的函数则要困难得多。）而每在序数世界前进一步，当我们再返回大数世界时，我们就能够比之前走得更远。可以说，序数就像一架梯子一样，能够帮助我们爬到更高的地方。

直观地说，序数记号就是一套为序数起名字的系统。一个序数记号越强大，它在序数世界中能够前进的距离就越远。除非我们明确地给出了某个层次的序数记号，否则我们对这一层次的序数是不会有直观的认识的。因此，接下来我们就将目光转向对序数结构的进一步挖掘，它将带领我们走到更远的地方。

序数不动点 对于自然数来说，对其不断地进行一些增大其取值的变换 (例如将 n 变换为 2^n)，它最终会超过任意一个给定的自然数。但是序数则不一样，例如我们考虑变换 $\alpha \to \omega^\alpha$，我们会发现如果从 ε_0 以下的序数出发，得到的序数将永远不会超过 ε_0。特别地，对于 ε_0 本身有 $\varepsilon_0 = \omega^{\varepsilon_0}$，我们称 ε_0 为映射 $\alpha \to \omega^\alpha$ 的不动点，这预示着上述变换的极限。

一般地，将变换重复作用于一个序数之上会产生一系列序数，而这一系列序数的极限就是该变换的不动点。各种递归序数记号所引入的增长模式都会被自身的不动点所卡住，它为序数记号设置了一个难以逾越的上限。通过枚举序数不动点的方式，我们可以得到维布伦 (Veblen) 函数，这是一个广泛应用的序数记号。

5 序数折叠函数

为了得到更大的大数，我们引入了序数。那么为了得到更大的序数，我们是否可以设想引入某些更加强大的东西，然后再通过某种方式将这些更加强大的东西对应到更大的序数上呢？沿着这条道路继续前进下去，我们就得到了序数折叠函数 (Ordinal Collapsing Function, OCF)，它将为我们提供更加强大的序数记号。

我们在前面讨论的序数都是那种可以由确定的规则出发，自下而上地得到的序数，我们将这样的序数称为递归序数。然而，并非是所有的序数都是递归序数。现在让我们设想将从 0 出发得到的所有可能的递归序数从小到大排成一列，那么将存在着一个序数，它大于上述所有的递归序数。这个序数就是最小的非递归序数，我们将其记为 Ω。直观地说，它就是最小的、不能够以“自下而上”的方式得到的序数。

非递归序数的定义 要想严格地给出非递归序数的定义，我们需要引入集合论和数理逻辑中的概念。我们称一阶逻辑公式为那些由 $\wedge, \vee, \to, \leftrightarrow, \neg, \forall, \exists$ 以及一系列常元、变元、函数和谓词所组成的公式，其中的量词 $(\forall, \exists)$ 只能作用于变元（而非谓词）之上。对序数 α 定义哥德尔 (Gödel) 可构造层次如下

$$L_0 = \varnothing, \quad L_{\alpha+1} = \mathrm{Def}(L_\alpha), \quad L_\alpha = \bigcup_{\beta<\alpha} L_\beta, \tag{39}$$

其中第三式的 α 为极限序数，$\mathrm{Def}(A)$ 定义为集合 A 中所有能够用一阶逻辑公式定义的子集构成的集合。进一步递归地定义公式层次如下：不包含无界量词的一阶逻辑公式为 Σ_0 以及 Π_0 公式；如果 φ 为 Π_n 公式，则 $\exists x_1 \ldots \exists x_m \varphi$ 为 Σ_{n+1} 公式；如果 φ 为 Σ_n 公式，则 $\forall x_1 \ldots \forall x_m \varphi$ 为 Π_{n+1} 公式。

设 α 是非递归序数或 0，则我们定义一切从 L_α 出发可用 Σ_1 公式定义的序数的上确界 β 为 α 之后的下一个非递归序数。

序数的层次结构 如前所述，序数是一种在 $\in$ 下良序，且其所有元素均为其子集的集合。这一定义中包含了许多性质非常丰富的集合，而递归序数只是其中的一部分，除此之外还有一系列非递归序数。

如果两个集合中的元素可以建立起一一映射，则称其为等势的。直观地说，等势的概念是“元素个数相等”在无穷情况下的推广。通过在自然数集上构建各种良序，我们可以得到大量与自然数集等势的序数，我们称之为可数序数。可数序数中包含了全部从 0 出发能够得到的递归序数，以及大量的非递归序数。我们在前几节中提到的序数 (也就是那些能够用在 FGH 之中的序数) 都是可数的递归序数，而我们将用在序数折叠函数之中的序数都是可数的非递归序数。

而如果在那些比自然数集的势要高的集合 (例如实数集) 上构建良序，那么我们将得到一些更大的序数，我们称之为不可数序数。最小的非递归序数仍然是可数的，而最小的不可数序数则是非递归的。

下面我们我们引入一个最小的非递归序数 Ω 以及一个相应的序数折叠函数 ψ，这个序数折叠函数 ψ 可以接收一个非递归序数，输出一个递归序数。首先我们定义

$$\psi(0) = \varepsilon_0. \tag{40}$$

接下来，ψ 内层的序数每增加 1，就取原来指数塔的极限，即

$$\psi(\alpha+1) = \psi(\alpha)^{\cdots^{\psi(\alpha)}}. \tag{41}$$

但是需要说明的是，这里的序数 α 并不是任意的，而应当是可以利用 OCF 以及 Ω 进行“有限表示”的。例如从 $\psi(0)$ 开始，内层的序数可以不断增大，每次遇到极限序数都可以取其相应基本列的极限，直到 $\psi(\psi(0)) = \psi(\varepsilon_0), \psi(\psi(\psi(0)))$ 乃至任意层 ψ 嵌套的极限 $\psi(\psi(\ldots\psi(0)\ldots))$。但是我们此时已经不能再通过在最外层的 ψ 内部加 1 来得到一个更大的序数 $\psi(\psi(\ldots\psi(0)\ldots)+1)$ 了，因为 $\psi(\ldots\psi(0)\ldots)+1$ 已经不是可以通过 OCF 有限地表达出来的了。事实上，$\psi(\psi(\ldots\psi(0)\ldots)+1)$ 与 $\psi(\psi(\ldots\psi(0)\ldots))$ 是相等的，在 ψ 内层引入的任何更大的递归序数都将被卡在这里。

为了跳出这一瓶颈，我们现在在 OCF 的表达式之中引入非递归序数 Ω，并定义

$$\psi(\Omega) = \psi(\psi(\ldots\psi(0)\ldots)). \tag{42}$$

我们称 $\psi(\Omega)$ 折叠了任意层 $\psi(\ldots)$ 嵌套的极限。与此前的递归序数不同的是，这里的 Ω 是一个非递归序数，$\Omega+1$ 已经是可以有限表示的了，因此它可以跳出此前的瓶颈而得到

$$\psi(\Omega+1) = \psi(\Omega)^{\cdots^{\psi(\Omega)}}. \tag{43}$$

于是 ψ 内层的序数又可以逐渐增大，直到 $\psi(\Omega+\psi(\Omega)), \psi(\Omega+\psi(\Omega+\psi(\Omega)))$ 乃至任意层嵌套的极限 $\psi(\Omega+\ldots\psi(\Omega+\psi(\Omega))\ldots)$。到这里 OCF 再一次进入了瓶颈，直到

$$\psi(\Omega\cdot 2) = \psi(\Omega+\Omega) = \psi(\Omega+\ldots+\psi(\Omega+\psi(\Omega))\ldots), \tag{44}$$

我们称 $\psi(\Omega+\Omega)$ 折叠了任意层 $\psi(\Omega+\ldots)$ 嵌套的极限。于是 OCF 又可以不断增大，直到

$$\psi(\Omega\cdot 3) = \psi(\Omega\cdot 2+\Omega) = \psi(\Omega\cdot 2+\ldots+\psi(\Omega\cdot 2+\psi(\Omega\cdot 2))\ldots), \tag{45}$$

我们称 $\psi(\Omega{\cdot}2{+}\Omega)$ 折叠了任意层 $\psi(\Omega{\cdot}2{+}\ldots)$ 嵌套的极限。进一步有 $\psi(\Omega{\cdot}\psi(\Omega)),\psi(\Omega{\cdot}\psi(\Omega{\cdot}\psi(\Omega)))$，直到

$$\psi(\Omega^2)=\psi(\Omega\cdot\Omega)=\psi(\Omega\cdot\ldots\cdot\psi(\Omega\cdot\psi(\Omega))\ldots), \tag{46}$$

我们称 $\psi(\Omega\cdot\Omega)$ 折叠了任意层 $\psi(\Omega\cdot\ldots)$ 嵌套的极限。然后又有 $\psi(\Omega^{\psi(\Omega)}),\psi(\Omega^{\psi(\Omega^{\psi(\Omega)})})$，直到

$$\psi(\Omega^\Omega)=\psi(\Omega^{\psi(\Omega^{\cdots^{\psi(\Omega)}})}), \tag{47}$$

我们称 $\psi(\Omega^\Omega)$ 折叠了任意层 $\psi(\Omega^{\cdots})$ 嵌套的极限。以此类推。

一般地，OCF 之中的 Ω 满足如下的折叠规则

$$\psi(\#\sim\Omega)=\psi(\#\sim\ldots\sim\psi(\#\sim\psi(\#))\ldots), \tag{48}$$

这里 $\sim$ 是序数加法、乘法、乘方运算的一种，$\#$ 是任意序列，省略号代表任意有限层的极限。这就是说，在 ψ 之中引入一个 Ω，相当于取此前任意有限层 $\psi(\#\sim\ldots)$ 嵌套的极限，即 $\psi(\#\sim\Omega)$ 折叠了任意层 $\psi(\#\sim\ldots)$ 嵌套的运算。我们将满足上述规则的 OCF 称为马多尔 OCF(Madore's OCF, MOCF)，式 (40),(41),(48) 就是含有一个最小的非递归序数 Ω 的 MOCF 的规则。

BOCF OCF 有许多种不同的类型，下面我们将介绍另一种常见的 OCF，称为布赫霍兹 OCF(Buchholz's OCF, BOCF)。与 MOCF 相比，BOCF 的折叠规则只是将 (40) 改为 $\psi(0)=\omega$，将 (41) 改为 $\psi(\alpha+1)=\psi(\alpha)\cdot\omega$，折叠规则不变。

相比于 MOCF 来说，BOCF 似乎要更弱一些。不过我们有

$$\psi^{\mathrm{B}}(\Omega)=\psi^{\mathrm{M}}(0),\ \psi^{\mathrm{B}}(\Omega\cdot2)=\psi^{\mathrm{M}}(1),\ \psi^{\mathrm{B}}(\Omega^2)=\psi^{\mathrm{M}}(\Omega),\ \psi^{\mathrm{B}}(\Omega^\omega)=\psi^{\mathrm{M}}(\Omega^\omega), \tag{49}$$

式中上标 M,B 分别代表 MOCF 和 BOCF。我们发现，虽然两函数起初有一定差距，但是在 Ω^ω 处，二者的取值相等了，我们称这种现象为追平 (Catching) 现象。追平现象意味着，对序数记号进行的某些提升很有可能在开始时可以拉开差距，但是它们会在更加巨大的序数下被抹平。这一现象是由于序数结构的复杂性引起的。

当然，我们还可以在序数折叠函数之中引入更高层次的非递归序数。例如，我们可以在 MOCF 之中引入一个第二个非递归序数 Ω_2，它是无法从 Ω 出发自下而上地得到的最小的序数。然后，我们可以引入一个序数折叠函数 ψ_1，它可以接受一个 Ω_2 层次的非递归序数，输出一个 Ω 层次的非递归序数。因此，可以设想用 ψ_1 将 Ω_2 层次的序数折叠为更大的 Ω 层次的序数，然后再利用 ψ 将得到的 Ω 层次的序数折叠出一个更大的递归序数。ψ 内层的序数又可以不断增大，直到将 Ω_2 直接放在 ψ 之中。上述过程可以不断地进行下去，我们可以引入一系列越来越强的 Ω_3,Ω_4，直到 Ω_ω。将 Ω_ω 放入 ψ 之中得到的序数称为布赫霍兹序数 (Buchholz's Ordinal, BO)，这是一个非常重要的序数。

MOCF 的集合论定义 MOCF 的集合论定义如下

$$C_\nu^0(\alpha)=\{\xi\mid\xi<\Omega_\nu\}\cup\{\Omega_\mu\mid\mu<\omega\}, \tag{50}$$

$$C_\nu^{n+1}(\alpha)=\left\{\gamma+\delta,\gamma\cdot\delta,\gamma^\delta,\psi_\mu(\eta)\mid\gamma,\delta,\mu,\eta\in C_\nu^n(\alpha),\mu<\omega,\eta<\alpha\right\}, \tag{51}$$

$$C_\nu(\alpha)=\bigcup_{n<\omega}C_\nu^n(\alpha), \tag{52}$$

$$\psi_\nu(\alpha)=\min\{\beta<\Omega_{\nu+1}\mid\beta\notin C_\nu(\alpha)\}. \tag{53}$$

粗略地说，上述定义说的是，$\psi_\nu(\alpha)$ 就是通过 $\xi<\Omega_\nu$、Ω_μ $(\mu<\nu)$ 以及 $\psi_\nu(\eta)$ $(\eta<\alpha)$ 经过有限次序数加法、乘法、乘方所不能构造出来的最小序数。特别地，我们称上述定义中的 Ω_1 为 Ω，而 ψ_0 为 ψ。可以验证，上述定义与前述的 OCF 折叠规则是一致的。相应地，BOCF 的集合论规则在 $C_\nu^{n+1}(\alpha)$ 的定义中去掉了 $\gamma\cdot\delta,\gamma^\delta$，也就是说只允许用序数加法构建更大的集合。

值得说明的是，真正意义上的 OCF 一定是以集合论方式定义的。如果只用折叠规则定义 OCF，那么我们得到的实际上只是一个长得像 OCF 的递归序数记号而已。

初看起来，在相同的强度之下，序数记号比大数记号的构造要容易得多 (例如，序数 ε_0 的构造比增长率为 ε_0 的函数的构造要容易得多)。但是一旦序数增长到了 $\psi(\Omega_\omega)$，那么构造序数记号和大数记号的难度就是相同的了。直观地说，如果将 FGH 的从递归序数到自然数的映射也视为一次折叠的话，那么到 $\psi(\Omega_\omega)$ 为止就已经进行 ω 次折叠了，而此时最初的这一次折叠的效应已经被紧随其后的 ω 次折叠抹平了 (即 $1+\omega=\omega$)。在这个层次的序数之下，序数记号和大数记号将不再有明显的区别。

事实上，在大数数学中研究序数有更加深刻的内涵，这是因为大数函数的递归结构与序数的联系是十分紧密的。若将一个 BO 之上的大数记号的序数核心取出，则它也可以对应于一个同等强度的序数记号。而若我们能够得到一个 BO 之上的序数记号，那么将其稍作改动（例如套上 FGH 的壳子）就可以得到一个相同强度的大数记号。因此，对序数的研究实际上就是对递归结构的深入挖掘，它可以让我们对大数记号的强度以及结构有更深刻的认识。

缓慢增长层次 缓慢增长层次 (Slow-Growing Hierarchy, SGH) 定义如下

$$g_0(n)=0,\quad g_{\alpha+1}(n)=g_\alpha(n)+1,\quad g_\alpha(n)=g_{\alpha[n]}(n), \tag{54}$$

其中第三式的 α 为极限序数。形式上地，将 SGH 中的变量 n 替换为 ω，我们可以得到一个序数记号，这个序数记号与大数记号拥有相同的递归结构，而其强度由 SGH 序列的下标 α 标记。例如，我们有 $n\uparrow^n n\approx g_{\psi(\Omega^\omega)}(n)$，因此形式上地，我们可以认为 $\omega\uparrow^\omega\omega\sim\psi(\Omega^\omega)$。也就是说，利用序数记号构造出一个大小为 $\psi(\Omega^\omega)$ 的序数，就相当于利用大数记号构造出一个强度为 $n\uparrow^n n$ 的函数。

直观地说，如果 FGH 给出了一个大数记号的话，那么作为序数记号的 SGH 就给出了具有该大数记号递归结构的序数。虽然 SGH 的增长速度确实是非常缓慢的，但是在 BO 下，FGH 和 SGH 仍然可以发生追平

$$f_{\mathrm{BO}}(n-1)\le g_{\mathrm{BO}}(n)\le f_{\mathrm{BO}}(n+1). \tag{55}$$

这说明了在 BO 之下，大数记号和序数记号将不再有明显的差别。

大数相关问题 在数学之中存在着一些数值非常大的问题，这些问题中常常隐含着更深层次的序数结构。例如，tree 序列考虑了一个由树 (一种特殊的图) 构成的序列，序列中的第 k 棵树最多包含 $n+k$ 个节点，并且排在前面的树不能够“嵌入”到排在后面的树中。满足这样的条件的序列长度的最大值定义为 $\mathrm{tree}(n)$。要对 $\mathrm{tree}(n)$ 的取值进行估计，我们可以按照“嵌入”关系恰当地在所有树构成的全体上构建一个良序。结果表明，完成这样的编序所需要的序数最大为 $\psi(\Omega^{\Omega^\omega})$。因此我们首先得到，$\mathrm{tree}(n)$ 一定是有限的，因为良序关系下没有无穷降链。其次，$\mathrm{tree}(n)$ 序列是一个增长速度很快的函数，其增长率的上界为 $\psi(\Omega^{\Omega^\omega})$。

序数和大数记号的统一是整个大数数学领域中极为重要的发现，它引发了大数数学中迄今为止最深刻的变革。从此之后，大数数学的研究从早期以大数为中心的范式，彻底转向了以序数为中心的范式。

6 非递归分析

FGH 给出了从递归序数到快速增长的函数的映射。因此为了得到更大的自然数，我们转而去寻找更大的递归序数。而 OCF 进一步给出了从非递归序数到大递归序数的映射。因此要想得到更大的递归序数，我们需要进一步挖掘更大的非递归序数的结构。

由于非递归序数不能够像递归序数一样，以自下而上的方式得到，因此要想得到更大的非递归序数，我们需要“用性质当定义”，即声明某些具有恰当的非递归性质的序数。例如，我们可以说前文中定义的 Ω 是“最小的不能够从在它之下的序数递归地得到的序数”，这就是用“存在性”定义得到的序数，而非通过“构造性”定义得到的序数。其他的非递归序数也都是按照这种方法得到的。

反射序数 (Reflecting Ordinal) 是一类重要的非递归序数。我们可以定义一系列公式层次 Π_n，n 越大公式越复杂。如果一个序数足够大，以至于它能够对于所有的 Π_n 公式都能够将其“反射”到某些更小的序数之上，则称其为 Π_n 反射序数。n 越大，对序数的要求就越苛刻，它也就提供了更加强大的非递归性质。例如 Π_1 反射序数仅仅为极限序数，而 Π_2 反射序数就已经达到了非递归序数（或者更严格地说，应当是容许序数），以此类推。

> **反射序数的定义** 令所有集合构成的全体为 V。若可以构建一个从 V 到集合 M 的恰当的映射，使得公式 φ 在 V 中与在 M 中的性质是相同的，则称 M 是 φ 的一个模型，记为 $M \models \varphi$。若“L_α 是 φ 的模型”可推出“存在一个 $\beta < \alpha$，使得 L_β 是 φ 的模型”，则称 α 反射了公式 φ。若 α 反射了所有的 Π_n 公式，则称 α 是 Π_n 反射序数。

所有（可数）的 Π_n 反射序数构成一个集合，我们可以在这一集合上进行一些进一步的操作。例如，我们可以在 Π_n 反射上再作用一次 Π_m 反射，记为 $m-n$。我们也可以取 Π_m 反射和 Π_n 反射的交集，记为 $m\ n$。未做特殊说明时，上述两个运算都是从左向右进行的，而且它们的优先级相同。习惯上，上述反射表达式指的都是集合中的最小元素。更高的反射表达式具有更强的非递归性质，因此它可以在 OCF 之中折叠更低的非递归序数。

例如，Π_2 反射序数为非递归序数，因此我们有 $2 = \Omega$。在其上取 Π_1 反射即可以得到 Π_2 反射序数的极限点，我们将之记为 $1-2 = \Omega_\omega$。我们还可以不断地将 Π_1 反射作用于其上，得到任意深度的 $(1-)^\alpha\ 2$，但是这样取极限点的操作终究是有极限的。现在我们可以考虑一个更强的反射表达式，它是 2 与 $1-2$ 的交集，即 $2\ 1-2$。这个操作超越了取极限点所能够达到的极限，我们称其为递归不可达序数 I，它在 OCF 之中折叠了 $(1-)^\alpha\ 2$。进一步又可以在 Π_2 反射序数上不断地取不可达点，直到任意深度的 $(2\ 1-)^\alpha\ 2$，它又被一个更强的反射序数 $M = 2-2$ 折叠，我们称其为递归马洛 (Mahlo) 序数。上述过程可以层层叠叠地不断进行下去，我们有 $2\ 1-2-2,\ 2-2\ 1-2-2,\ 2-2-2,\ 3$，以此类推，直到 Π_ω，这是一个强大的非递归序数。将它们恰当地放进 OCF 之中，我们就可以得到更大的递归序数。

> **OCF 的困难** 反射序数的性质过于复杂，以至于很难用集合论的方式定义恰当的序数折叠函数。尽管在证明论中有一些这样的尝试，但是这些结果目前并不能被大数数学的研究者所理解。而即使仅仅使用折叠规则来定义反射 OCF，目前所采用的规则也是非常不完备的。折叠规则之中仍然存在着大量含糊不清的地方，并且每个人所使用的记号和折叠规则都不尽相同。至少在目前看来，将大非递归序数恰当地放进 OCF 之中仍然是一个十分困难的问题。

稳定序数 (Stable Ordinal) 刻画了比反射序数更强的非递归序数。我们可以定义一系列公式层次 Σ_n，n 越大公式越复杂。如果对于所有的 Σ_n 公式 φ 来说，序数 β 都可以将其所满足的某些关于 φ 的性质“稳定”到序数 α 之上，则称 α 关于 β 是 Σ_n 稳定的。习惯上我们考虑 β 是 α 的一个函数 $f(\alpha)$，即考虑“α 关于 $f(\alpha)$ 是 Σ_n 稳定的”。满足上述条件的 α 给出了一条非常强大的非递归性质，并且 n 越大或者是函数 $f(\alpha)$ 越大，满足这一稳定条件的序数 α 就具有越强的非递归性质（我们实际上是要找到一个足够大的 α，使得其关于 Σ_n 公式的性质在更大的 $f(\alpha)$ 之上仍然能够得到保留）。特别地，我们将“α 关于 $f(\alpha)$ 是 Σ_1 稳定的”记为 $\lambda\alpha.f(\alpha)$。事实上仅仅是 $\lambda\alpha.(\alpha+1)$ 稳定就等价于 Π_ω 反射序数了，稳定序数的强大由此可见一斑。

稳定序数的定义 *若存在一个恰当的从 M 到 N 的映射，使得对所有 Σ_n 公式 φ，都有“M 是 φ 的模型等价于 N 是 φ 的模型”，则称 M 是 N 的一个 Σ_n 初等子结构。若 $\alpha \leq \beta$，且 L_α 是 L_β 的一个 Σ_n 初等子结构，则称 α 关于 β 是 Σ_n 稳定的。*

与递归序数一样，在非递归世界中的前进也是没有尽头的。目前大数数学的研究者只对较小的一部分 Σ_1 稳定序数有了一定的了解，对于较大的 Σ_1 稳定序数了解不多，对于 Σ_2 乃至更高的稳定序数则知之甚少。在稳定序数之上还有更加强大的非递归序数，例如间隙序数 (Gap Ordinal) 等，这些序数的性质几乎仍然不为大数数学的研究者所知。

7 蠕虫型记号

除了借助于非递归序数之外，我们也可以用更加强大的递归方法来直接构造递归序数。历史上曾经涌现出一批强大的递归记号体系，例如数阵 (Array) 型记号、下降 (Dropping) 型记号以及塔拉诺夫斯基 (Taranovski) 记号等。然而随着时间的推移，这些记号体系已经慢慢地被人们淘汰了。目前最为强大，且被人们所广泛接受的递归记号体系就是蠕虫 (Worm) 型记号。

最简单的蠕虫型记号为初等序列 (Primitive Sequence System, PrSS)，它是由一系列自然数构成的序列，形如 $(0,1,2,1)$，其中我们要求其第一项为零，而后一项的取值至多比前一项大 1。PrSS 的规则可以简要叙述如下。如果序列中一个元素都没有，则其取值为 0

$$(\,) = 0. \tag{56}$$

在序列的最后一项上加上一个 0，代表序列的取值加一

$$(\#, 0) = (\#) + 1, \tag{57}$$

其中 $\#$ 为任意的序列。如果序列的最后一项 a_k 不是 0，那么可以从后往前逐项找到第一个小于最后一项的项 a_i，并将序列表示为 $(\#_1, a_i, \#_2, a_k)$。我们称 $\#_1$ 为好部，$a_i, \#_2$ 为坏部，a_i 为坏根。要展开上述序列，我们只需去掉序列的最后一项 a_k，然后保留好部 $\#_1$ 不动，而将坏部 $a_i, \#_2$ 不断地在序列后进行复制

$$(\#_1, a_i, \#_2, a_k) = (\#_1, a_i, \#_2, a_i, \#_2, a_i, \#_2, \ldots), \tag{58}$$

省略号代表有限次复制的极限。PrSS 的强度可以简单分析如下，我们有

$$(\,) = 0, \quad (0) = 1, \quad (0,0) = 2, \quad (0,1) = (0,0,0,\ldots) = \omega, \tag{59}$$

$$(0,1,0) = \omega + 1, \quad (0,1,0,1) = (0,1,0,0,\ldots) = \omega \cdot 2, \quad (0,1,1) = (0,1,0,1,\ldots) = \omega^2, \tag{60}$$

$$(0,1,2) = (0,1,1,\ldots) = \omega^\omega, \quad (0,1,2,3) = \omega^{\omega^\omega}, \quad (0,1,2,3,\ldots) = \varepsilon_0. \tag{61}$$

因此 PrSS 的极限就是 ε_0。

从 PrSS 之中，我们可以看到蠕虫型记号的核心特征。蠕虫型记号不是像其他较为初等的记号一样，展开过程是在某个元素之上进行迭代，迭代过程难以为继了就进位到下一个元素之上，而各个元素之间又以层层叠叠的分隔符区分开（这是数阵型记号的典型特征）。它的展开过程实际上是提取出内部的元素，然后按照某些特定的规则不断地复制其自身。事实上蠕虫型记号的展开过程的是一种“树状结构”，而并非普通的线性结构。

尽管 PrSS 的极限并不是很大，但是它蕴含着非常强大的递归核心。我们将看到，如果能够恰当地将 PrSS 推广到多行，那么我们将得到一个极为强大的递归序数记号，称为巴什库矩阵 (Bashicu Matrix System, BMS)。一个 BMS 是形如

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 2 \\ 0 & 1 & 2 & 1 & 2 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} \tag{62}$$

的矩阵。通常我们将其每一列取出，然后并排地将其记为一维形式，例如上述 BMS 可以记为 $(0,0,0)(1,1,1)(2,2,1)(3,1,0)(2,2,0)$。BMS 的展开规则为：如果最末一列全部为零，则其取值为去掉最末一列得到的 BMS 加一。如果最末一列不全为零，则去掉最末一列，保留好部（左边的黑色部分）不动，将坏部（红色部分）每一列加上阶差序列（本例中是 $(1,0,0)$）后不断复制在 BMS 的末尾，每复制一次就加上一次阶差序列。于是我们得到上述 BMS 的展开式为

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 5 & \dots \\ 0 & 1 & 2 & 1 & 1 & 2 & 1 & 1 & 2 & 1 & \dots \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & \dots \end{pmatrix}, \tag{63}$$

即

$$(0,0,0)(1,1,1)(2,2,1)(3,1,0)(2,1,1)(3,2,1)(4,1,0)(3,1,1)(4,2,1)(5,1,0)\dots \tag{64}$$

BMS 的规则 下面我们给出寻找 BMS 的坏部与阶差序列的具体方法。第一行元素的父项为与该元素处在同一行、在其左边且小于该元素的第一个项，而其祖先项为其父项以及父项的父项等一系列项构成的全体。在本例中，第一行最末元素 2 的祖先项就是第一行中的元素 1。如果所考虑的元素不处在第一行，那么我们还要加上一条规则：它正上方的项应当同时是该元素正上方项的祖先项。例如，第二行最后一个元素 2 的父项并不是在其左边的 1，因为这个 1 上方的元素并非是 2 上方元素的祖先项。它真正的父项应当是第二行第二个元素 1。

我们在最后一列中从下往上找到第一个不为零的元素 2，然后找到它的父项，即第二行第二个元素 1，这一项所在的列就是坏根。从坏根出发 (包含坏根) 到最后一列 (不包含最后一列) 之间的部分 (红色部分) 就是坏部，坏根之前 (不包含坏根) 的部分 (左边的黑色部分) 就是好部。阶差序列为最右边一列 (2,2,0) 减去坏根 (1,1,1) 的值，但是需要注意，阶差序列的最后一项为零，并且如果最后一列中某项为零，那么阶差序列中相应的上一项也为零。在本例中，阶差序列为 (1,0,0)。有一个例外的情况，即如果坏部中某一项的祖先项不包含坏根，那么它在后续的展开之中将保持不变，在本例中这种情况对最终的结果没有影响。

对非递归序数研究得越深入，就越能体会到 BMS 的强度。在 BMS 面前，其他强大的序数记号几乎寸步难行。单行 BMS 就是 PrSS，对于多行 BMS 我们有

$$(0,0)(1,1) = \psi(0), \quad (0,0)(1,1)(1,1) = \psi(1), \tag{65}$$

$$(0,0)(1,1)(2,1) = \psi(\Omega), \quad (0,0,0)(1,1,1) = \psi(\Omega_\omega), \tag{66}$$

$$(0,0,0)(1,1,1)(2,1,1)(3,1,0) = \psi(\Omega_\Omega), \tag{67}$$

$$(0,0,0)(1,1,1)(2,1,1)(3,1,0)(4,2,0) = \psi\left(\psi_{\Omega_{I+1}}(0)\right), \tag{68}$$

$$(0,0,0)(1,1,1)(2,1,1)(3,1,1)(3,1,0)(4,2,0) = \psi\left(\psi_{\Omega_{M+1}}(0)\right), \tag{69}$$

$$(0,0,0)(1,1,1)(2,1,1)(3,1,1)(4,1,0)(5,2,0) = \psi\left(\psi_{\Omega_{\Pi_3+1}}(0)\right), \tag{70}$$

$$(0,0,0)(1,1,1)(2,2,0) = \psi(\Pi_\omega) = \psi(\lambda\alpha.(\alpha+1) - \Pi_0), \tag{71}$$

$$(0,0,0)(1,1,1)(2,2,1) = \psi(\lambda\alpha.\Omega_{\alpha+2} - \Pi_1) \tag{72}$$

$$(0,0,0)(1,1,1)(2,2,2) = \psi(\omega - \pi - \Pi_0). \tag{73}$$

式中 $\omega - \pi - \Pi_0$ 是一条长度为 ω 的稳定链。事实上，三行 BMS 几乎已经穷尽了我们目前对于非递归序数的全部理解。对更高的 BMS 来说，由于缺乏记号与之对应，因此我们难以对其强度有直观的认识。就目前来看，对 BMS 及其他递归记号强度的分析，已经成为了大数数学领域中最重要的工作之一（尽管这仍然是以枚举的不完全归纳方法来完成的）。

大数记号 对 BMS 稍作改动，即可得到一个相应的大数记号，它与 BMS 有相同的递归结构。假设在展开的第 n 步中不复制无穷多次，而只复制有限的 $n+1$ 次，则我们可以得到一个有限的大数记号，其增长率等于作为序数记号的 BMS 的极限。特别地，如果我们对 PrSS 做类似的改动，那么我们将得到一个增长率为 ε_0 的大数记号，这是得到增长率为 ε_0 的函数最容易的方式之一。

Y 序列是一个比 BMS 更加强大的蠕虫型记号，其表达式形如 Y$(1,3,4,6,7,9)$。为展开 Y 序列，我们不是像 PrSS 一样直接复制元素，而是先在序列的各项之间恰当地作差，得到更进一步的阶差序列，并根据各元素之间差值的关系绘制山脉图。然后，我们在山脉图中选择恰当的结构进行不断地复制，然后将余下的部分按照山脉图的规则进行补全，最终就可以得到其展开式。由于山脉图中蕴含着非常复杂的结构，因此在 Y 序列短短的表达式中压缩了很多的信息。在展开的过程中，这些信息得到了充分的利用，因此 Y 序列可以具有很高的强度。BMS 的极限在 Y 序列中仅为 Y$(1,3)$，而即使是将 BMS 的行数推广到任意序数，其极限也只能达到 Y$(1,3,4,2,5,8,10)$，其强度由此可见一斑。Y 序列山脉图的做法以及详细的展开规则较为复杂，这里从略。

近年来，在 Y 序列之上，研究者们还进一步地发展了一系列记号，例如变异矩阵系统 (Mutant Matrix System, MMS)、山脉记号 (Mountain Notation, MN) 以及 X-Y 序列等。除此之外还有一些超越蠕虫型记号体系的尝试，如基本序数序列 (Fundamental Ordinal Sequence, FOS) 以及伪伪伪 z(Fake Fake Fake Zeta, FFFZ) 等。我们期望这些记号体系的强度能够超越 Y 系列记号的极限，然而我们目前对它们的了解尚不充分。我们既不知道这些记号的理想极限是多少，也不知道它们是否仍然存在着无法弥补的漏洞。

良定义 我们对一个记号的最基本要求，就是要求它能够唯一地确定下来一个大数 (或者序数)。如果一个记号满足了这条性质，则我们称其为良定义的 (Well-defined)。一个良定义的记号的展开过程应当总能够在有限步骤内结束，而不能出现无穷降链。这意味着如果写一段程序来描述这个记号的展开，那么它应当是可以停机的，而不能陷入死循环。而如果将大数记号的表达式按照大小排成一列，那么它应当构成一个良序 (这是因为良序集的每个子集都必有最小元)。

良定义是悬在所有记号头上的达摩克里斯之剑，越强的记号越不容易良定义。可以设想，为了提高记号的强度，我们需要尽可能地让展开过程变得复杂。而一旦展开过程过于复杂，那么它就很有可能无法停止，最终变为非良定义的记号。如果把 BMS 的例外规则“如果坏部中某一项的祖先项不包含坏根，那么它在后续的展开之中将保持不变”删去，则展开过程将出现无穷降链，BMS 将变为不良定义的。已经有无数的记号被发现存在无穷降链，倒在了良定义的路上，从而消失在了历史的尘埃之中。但是反过来，找不到无穷降链却并不能说明它是良定义的。目前我们严重缺乏证明记号良定义的方式，已经被证明良定义的最强记号就是 BMS。对于比 BMS 更强的记号，我们尚不知道它们是不是良定义的，它们随时都有可能被更深入的分析所排除。

证明论序数 在数学上，要想证明一个结论，我们必须选定一系列不言自明的命题，然后从它们出发，通过严密的逻辑进行证明。我们将这一系列作为前提的命题所构成的体系称为“公理体系”。有些公理体系是比较强大的，而另一些公理体系是更弱的，它们的强弱可以由证明论序数 (Proof Theory Ordinal, PTO) 来刻画。一个重要的公理体系为皮亚诺算术体系 (Peano Arithmetic, PA)，它以公理的形式声明了自然数及其加法和乘法。著名的哥德尔不完备定理指出，PA 是不完备的，即存在着一个命题，它既不能在 PA 中证明，也不能在 PA 中证伪。更具体地，我们若在 PA 之中施以超限归纳，则只有 ε_0 以下的超限归纳是可证的，我们称 ε_0 为 PA 的证明论序数，记为 $\mathrm{PTO(PA)}=\varepsilon_0$。

一个公理体系越强，其证明论序数就越大。若一个序数记号在某公理体系中可以证明良序，则其极限必须不能超过该公理体系的证明论序数。目前通用的集合论公理体系是策梅洛-弗兰克尔-选择公理体系 (Zermelo-Frankel-Choice, ZFC)，它几乎是除集合论自身之外的全部数学理论的基础，其证明论序数也是非常强大的。如果一个序数记号超过了 PTO(ZFC)，那么其良序性将不能够在 ZFC 之中得到证明，我们必须诉诸更强的公理体系。

8 非递归函数

前文中所讨论的各种快速增长的函数都是自下而上地通过递推公式得到的，我们称这样的函数为递归函数。递归函数是目前大数数学研究的核心，因为只有一个能够递归地得到的函数才是能够真正被我们所理解的。然而确实存在着一些增长率超越了所有递归函数的大数函数，这样的函数称为非递归函数。下面我们将对非递归函数进行一个简要的介绍。

要想得到一个增长率超过所有递归序数的函数，我们可以将这些递归函数以恰当的方式排成一列，然后对其进行对角化，这样我们就得到了一个超越了所有这些递归函数的更强的函数，这就是一个非递归函数。一个简单的做法是考虑一种足够强大的编程语言，它可以编写出所有的递归函数。然后，我们将用该语言写出的、包含 n 个字符的程序所能够输出的最大的数排成一个序列。由于任何递归函数都可以由该语言编写出，因此当 n 足够大时，该语言将总能够编写出超越这个递归函数的程序。

作为例子，利用 Javascript 语言编写的具有 20 个字符的程序就可以写出四级运算

```
for(a=i=9;i--;)a**=a
```

33 个字符的程序就可以写出超运算

```
n=y=>x=x--*y?n(y)*n(y-1):9;n(x=9)
```

68 个字符的程序就可以写出 PrSS

```
for(a=b='43210';a=a.slice(1)
.replace(a[0]-1,'$&$`\$\&'.repeat(++b));)b
```

114 字符的程序就可以写出 BMS

```
for(a=[[1,1,1,1,i=1],r=[]];
i>r?r=(c=a.shift(i=0)).pop():a=[Object.assign(a[r-1]
.map(e=>e>i?e+r:e),i++||c),...a];)r
```

由此可见，程序所能够刻画出的最大数值的增长速度是十分快速的，事实上它超过了所有的递归函数。

因此，“用不超过 n 个字符的某种编程语言编写的程序所输出的最大数”就是一个超越了所有递归函数的非递归函数。我们看到，这样一个函数将不能够以递归的方式得到，它并非是一个“构造性”的定义，而是一个“存在性”的定义。相应地，它的增长率也超越了所有的递归序数，达到了第一个非递归序数 Ω。习惯上我们将这个函数的增长率称为丘奇 (Church)-克林 (Kleene) 序数，记作 ω_1^{CK}。

递归函数 递归函数定义为能够从初始函数出发，经过合成模式、原始递归模式和正则极小化模式得到的函数。其中初始函数包含：(1) 零函数 $O(x)=0$；(2) 后继函数 $S(x)=x+1$；(3) 投影函数 $P_i^n(x_1,\cdots,x_n)=x_i$。合成模式为选取一个函数 $h(x)$，使得 $h(x)=f(g_1(x),\cdots,g_k(x))$，其中各 f,g_i 均为递归函数。原始递归模式为选取

一个函数 $h(x)$，使得 $h(x,0)=f(x), h(x,y+1)=g(x,y,h(x,y))$，其中 f,g 均为递归函数。正则极小化模式为选取一个函数 $g(x)$，其取值为使得“$f(x,y)=0$，且对于任意 $z<y$，$f(x,z)$ 都有定义”的 y 的最小值。

图灵机 图灵 (Turing) 机是对一般的计算过程的抽象。图灵机由一条无限长的纸带以及一个读头构成，纸带上有无限多个顺次排列的方格，每个方格都具有两种可能的颜色，而读头可能处于多种不同的状态。在起始时刻，图灵机的纸带上已经预先写好了每个格子的状态，然后将图灵机的读头放在某一个给定的格子上。然后，图灵机将按照自身编码的指令集（即图灵程序）进行运动。在每个时刻，图灵机将读取纸带上当前格子的颜色以及自身的状态，然后将这个格子的颜色改变为某一状态、将图灵机自身的状态改变为某一状态，并将读头移动到某个相邻的位置。如果图灵机运行到了停机状态，那么它将停止运动，图灵机的程序结束。

通过在图灵机上进行恰当的编码，我们可以利用它来计算函数。可以证明，一个函数是图灵可计算的，等价于它是递归函数。更进一步地，丘奇-图灵论题指出，图灵可计算型性等价于直观可计算性。如果这一论题成立的话，那么人类可计算的函数，就等价于图灵机可计算的函数，同时也就就等价于递归函数。

定义能够停机的 n 状态图灵机在停机前所能运行的最大步数为忙碌海狸函数 (Busy Beaver, BB)，记作 $\mathrm{BB}(n)$。$\mathrm{BB}(n)$ 是最著名的非递归函数，其前几个取值为

$$\mathrm{BB}(2)=6,\ \mathrm{BB}(3)=21,\ \mathrm{BB}(4)=107,\ \mathrm{BB}(5)=47176870,\ \mathrm{BB}(6)\geq 10\uparrow\uparrow 15. \quad (74)$$

$\mathrm{BB}(n)$ 超过了所有的递归函数，其增长率同样为 $\Omega=\omega_1^{\mathrm{CK}}$。

既然得到了增长率为 $\Omega=\omega_1^{\mathrm{CK}}$ 的函数，那么我们就可以在非递归的世界之中继续前进。我们看到，上述利用编程语言给出的非递归函数的表达能力受到了计算体系本身的限制。如果我们能够引入超越我们这个物理世界的更加强大的计算模型，那么我们将可以得到一系列更强的非递归序数。

更强大的非递归函数可以通过直接诉诸公理体系来进行构造。我们定义拉约 (Rayo) 函数 $\mathrm{Rayo}(n)$ 为不能够用包含字符数为 n 的一阶逻辑语句定义的最小的自然数 n。拉约函数的前几个取值为

$$\mathrm{Rayo}(0)=0,\quad \mathrm{Rayo}(10)=1,\quad \mathrm{Rayo}(30)\geq 2,\quad \mathrm{Rayo}(56)\geq 3. \quad (75)$$

虽然拉约函数在初期的增长速度是比较慢的，但是在 n 较大的情况下，其增长速度将变得非常快。拉约函数的强度由公理体系自身的表达能力所限制。可以设想，如果我们能够恰当地采用更强的公理体系，那么我们将可能得到更加强大的非递归函数。

非良定义 由于没有合适的基本列，因此我们尚无法为非递归的 $f_\Omega(n)$ 给出恰当的定义。即使可以通过枚举图灵机的方式给出 Ω 的基本列，它也不是能够递归地得到的。

为了利用公理体系去构建大数，我们必须预先指定公理体系所使用的模型，这一部分需要许多艰深的严格化工作。我们目前尚不清楚拉约函数是否是良定义的，除此之外在更加强大的公理体系中扩展拉约函数的尝试至今也没有取得公认的成功。

培里悖论 虽然我们很想定义“能够用 n 个字符定义的最大自然数”(这就是本文开篇的那个例子所要探求的答案)，然而遗憾的是，除非我们为自然语言设定明确的规则，否则它将不是良定义的。例如，我们可以考虑“不能用二十个字确定的最小自然数”，这句话确定了一个自然数。但是这句话本身又少于 20 个字，因此这个数也能够用小于 20 个字来确定。因此这句话产生了悖论，我们称之为培里 (Perry) 悖论，它是由于自然语言的模糊性导致的。

跨过了递归序数的限制之后，大数函数就可以在非递归的广阔世界之中继续前进了。然而我们目前对于非递归函数及其增长率的认识还十分有限，我们也不清楚如何能够有效地构造增长速度更快的非递归函数。

若按照增长速度对所有可能的大数函数（不论是递归的还是非递归的）进行排序，并利用序数来标记这些函数的增长率，那么在所有的这些序数之上还将存在一个更加强大的序数。这个序数是第一个不可数序数 ω_1，它就是整个大数数学的终点。

9 总结

在本文中，我们简要介绍了大数数学领域的基本问题、研究方法以及进展。为了写出更大的大数，我们需要发展更强大的大数表示法，这催生了对于快速增长的函数的研究。快速增长层次提供了从序数到快速增长的递归函数的变换，因此研究快速增长的递归函数的问题，就转化为了研究更大的递归序数的问题。为了得到更大的递归序数，一种做法是寻找更加强大的递归序数记号，以递归的方式在序数世界中前进；另一种做法是寻找更加强大的非递归序数，并将它们放进序数折叠函数之中从而得到更大的递归序数。所有递归函数的增长率均为递归序数，在这之上存在着更加强大的非递归函数，其增长率为非递归序数。在目前看来，对大数数学的研究，就是不断地深入挖掘序数结构的过程。

若要进一步了解大数数学的相关内容，读者可以参考其他更详细和深入的资料，例如[1-6]。

Bibliography

[1] 曹知秋. 大数理论[M/OL]. 2024. https://github.com/ZhiqiuCao/Googology.

[2] Googology wiki[EB/OL]. https://googology.fandom.com/wiki/Googology_Wiki.

[3] HYPCOS. 大数入门[M/OL]. 2014. https://www.docin.com/p-2128722751.html.

[4] CORE.EXE. 大数数学入门-重置版[EB/OL]. 2023. https://www.zhihu.com/column/c_1697290814588301312.

[5] 梅天狸. 从一写到无穷大[EB/OL]. 2023. https://www.zhihu.com/column/c_1579269652357877760.

[6] 一只绵羊 233. Googology：大数数学初步[EB/OL]. 2023. https://www.zhihu.com/column/c_1618698953565618176.