arXiv:2302.03929v2 [math.CO] 1 Jun 2023

枚举网格带符号排列类的一种算法

Sa´ul A. Blanco*和 Daniel E. Skora†

计算机科学系 印第安纳大学布卢明顿分校，印第安纳州布卢明顿，邮编47408

2023年5月31日

摘要

本文提出一种算法，用于枚举一类特定的带符号排列，称为网格带符号排列类。对于排列而言，相应的网格类之所以受到关注，是因为它们等价于可用多项式枚举的排列类。此外，我们将研究结果应用于基因组重排，并证明具有固定前缀反转距离和反转距离的带符号排列数量可由多项式给出，而这些多项式可以通过我们的算法计算。

关键词： 排列类、前缀反转、多项式枚举

1 引言

长度为 n 的排列 σ 包含长度为 k 的排列 π，记作 $k \leq n$，当且仅当 σ 有一个子串 s，其中 s 的字符与作为字符串的 π 的字符具有相同的相对顺序。例如，$\sigma = 6127354$ 包含 4132，因为 σ 的子串 6254 中的字符与 4132 具有相同的相对顺序。如果 σ 包含 π，我们记作 $\sigma \geq \pi$。排列的下集是所有被包含于 σ. 的排列组成的集合。排列类是一个在包含关系下封闭的集合。换言之，如果对于每个 $\sigma \in \mathcal{C}$, $\sigma \geq \pi$ 都意味着 $\pi \in \mathcal{C}$. ，那么 $\mathcal{C}$ 就是一个排列类。用 S_n 表示所有长度为 n 的排列组成的集合，并令 $\mathcal{C}$ 为一个排列类，那么研究 $|\mathcal{C} \cap S_n|$，例如给出一个上界或简洁的表达式，是一个著名的组合问题（例如参见 [10, 11, 12, 15] 及其中的参考文献）。最著名的成果之一是 Markus 和 Tardos 于 [13] 证明了 Stanley–Wilf 猜想，该猜想指出，除非 $\mathcal{C}$ 是所有排列组成的集合，否则 $|\mathcal{C} \cap S_n|$ 至多呈指数增长。一个备受关注的问题是，哪些排列类具有多项式增长 [1]，也就是说，哪些排列类可以用多项式枚举 [12, 11, 10]。在回答这一问题时，该领域的另一个著名成果是所谓的斐波那契二分定理，该定理由 Kaiser 和 Klazer 首先证明 [12],，指出对于每个排列类 $\mathcal{C}$，要么 $|\mathcal{C} \cap S_n| \geq F_n$ 对所有 n 都成立，其中 F_n 表示第 n 个斐波那契数；要么当 n 充分大时，$|\mathcal{C} \cap S_n|$ 由一个多项式给出。此外，结合 [1] 和 [11], 的结果还可知，由多项式枚举的排列类恰好就是所谓的网格排列类。事实上，以下定理成立。

定理1.1。 *[10,* 定理*1.3]*设 $\mathcal{C}$ 是一个排列类， S_n 是由所有长度为 $n \geq 1$ 的排列构成的对称群。那么，以下陈述等价。

(a) 对所有充分大的 n， $|\mathcal{C} \cap S_n|$ 由一个多项式给出。

*电子邮件：sblancor@indiana.edu

†电子邮件：danskora@iu.edu

(b) 对某个 n， $|\mathcal{C}\cap S_n| < F_n$，其中 F_n 是第 n 个斐波那契数（如果 $n>2$，则 $F_1=1$、$F_2=2$ 和 $F_n=F_{n-1}+F_{n-2}$。）

(c) $\mathcal{C}$ 是有限个钉排列构成的网格类（这些定义见 *[10]*）。

Homberger–Vatter 算法 [10]，（以下简称 HVA）提供了一种高效方法，用于生成计算这些网格排列类基数的多项式。

与排列类类似，也可以定义带符号排列类（正式定义见第 2 节），并探讨其中哪些类可由多项式计数。在此，我们关注带符号排列，它们可以写成由集合 $\{1,2,\ldots,n\}$ 中 n 个互不相同字符组成的字符串，其中每个字符都带有正号或负号。与 S_n类似，全体带符号排列构成的集合记为 B_n，在复合运算下具有群结构。在这种情况下， B_n称为超八面体群。事实上， B_n 是超立方体或交叉多胞体的对称群。

1.1 我们的贡献

本文的主要贡献包括：(i) 将网格类 的概念扩展到带符号排列。我们 (ii) 证明，若 $\mathcal{C}$ 是这些网格带符号排列类之一，则对于所有 $n>1$， $|\mathcal{C}\cap B_n|$ 都由一个多项式给出。此外，我们 (iii) 提出并实现了算法1，该算法可计算这个多项式。该算法的一个有趣应用是枚举与基因组重排和焦煎饼问题相关的排列类。具体而言，我们 (iv) 证明，具有固定前缀反转距离和反转距离的带符号排列类可由多项式枚举，并计算了其中一些多项式。

本文其余部分安排如下。第2节介绍必要的背景知识和定义。第3节介绍用于枚举带符号排列网格类的算法，以及证明算法正确性和运行时间所需的辅助结果。最后，第4节介绍该算法在若干相关情形中的应用，这些情形与焦煎饼问题和基因组重排有关。事实上，在生物信息学中，前缀反转距离和反转距离具有生物学意义，可用于衡量某些物种在进化层面上彼此“相距”多远。我们证明，前缀反转距离和反转距离均不超过 k 的所有带符号排列的数量由多项式给出。

2 背景与符号

我们用 $\mathbb{Z}^n_{\geq 0}$（其中 $n\geq 1$）表示分量为非负整数的 n 维向量集合。类似地，我们用 $\mathbb{Z}^n_{>0}$ 表示分量为正整数的 n 维向量集合。我们用粗体表示向量，并约定用 $\mathbf{1}$ 表示每个分量均为 1 的向量（维数可从上下文中确定）。向量 $\mathbf{v}$ 的第 i 个分量记为 $\mathbf{v}(i)$。此外， S_n表示 $[n]:=\{1,\ldots,n\}$ 的所有排列构成的群，即阶为 $n!$ 的对称群。

带符号排列。给定一个整数 $n\geq 1$，定义集合 $[\pm n]:=\{\pm 1,\pm 2,\ldots,\pm n\}$。所有带符号排列 构成的集合，是满足以下性质的所有双射 $\pi:[\pm n]\to[\pm n]$ 的集合：对于所有 $1\leq i\leq n$，$\pi(-i)=-\pi(i)$。通常将集合 $[\pm n]$ 上所有带符号排列构成的集合记为 B_n，并将 $\sigma\in B_n$ 写成字符串 $\pi(1)\pi(2)\cdots\pi(n)$。我们用下划线表示负项; 例如， $14532\in B_5$。我们将带符号排列 $\pi=\pi(1)\cdots\pi(n)$ 的长度 定义为 π 中字符（正或负）的个数，并记作 $\mathsf{len}(\pi)=n$，表示 π 的长度为 n，换言之， $\pi\in B_n$

区间、递增区间、递减区间和单调区间。设 $\pi=\pi(1)\cdots\pi(n)$ 是一个带符号排列。我们称连续元素的子串 $\pi(i)\pi(i+1)\cdots\pi(m)$（其中 $i\leq m$）为一个区间，如果忽略符号，集合 $\{\pi(i)\ \pi(i+1)\ldots,\pi(m)\}$ 等于一个由 $m-i+1$ 连续整数组成的集合。如果 $\pi(i)\pi(i+1)\cdots\pi(m)$ 是一个区间，我们称 $\pi(i)\pi(i+1)\cdots\pi(m)$ 是递增的，如果

$$|\pi(i)|<|\pi(i+1)|<\cdots<|\pi(m)|$$

和递减当

$$|\pi(i)| > |\pi(i+1)| > \cdots > |\pi(m)|.$$

如果 $\pi(i)\pi(i+1)\cdots\pi(m)$ 是递增或递减的，我们称 $\pi(i)\pi(i+1)\cdots\pi(m)$ 为一个单调区间，或简称为单调。

序同构与标准化。给定带符号排列 $\sigma \in B_n$ 和 $\pi \in B_k$，其中有 k、$n \in \mathbb{Z}$ 和 $1 \le k \le n$，若 $\pi = \pi(1)\pi(2)\cdots\pi(k)$ 的字符与 $\sigma(i_1)\sigma(i_2)\cdots\sigma(i_k)$ 的字符具有相同的相对顺序，并且对于 $1 \le j \le k$，$\pi(j)$ $\sigma(i_j)$ 具有相同的符号，则称 $\pi = \pi(1)\pi(2)\cdots\pi(k)$ 与 $\sigma(i_1)\sigma(i_2)\cdots\sigma(i_k)$ 序同构。例如，在 12453 中，132 与 243 序同构。此外，给定由 m 个互不相同的整数构成的字符串 s ,¯每个整数都带有正号或负号,¯ s 的标准化是唯一一个与 s 序同构、长度为 m 的带符号排列 σ。例如，若 $s = 97435$，其标准化为 54213。

现在我们将 在带符号排列上定义一个偏序，使 B_n 具有偏序集结构 re.

定义 2.1（包含与带符号排列类）。若存在指标 $1 \le i_1 < \cdots < i_k \le n$，使得 $\sigma(i_1)\cdots\sigma(i_k)$ 与 $\pi = \pi(1)\cdots\pi(k)$ 序同构，则称长度为n的带符号排列 σ 包含长度为 $k \le n$的带符号排列 π。特别地，对于所有 $1 \le j \le k$，$\sigma(i_j)$ 与 $\pi(j)$具有相同的符号。

如果 σ 包含 π，则记作 $\pi \le \sigma$。直观地说，$\pi \le \sigma$ 如果可以从 σ 中删除若干项（必要时进行标准化），从而得到 π。例如，取 $\sigma = 41532$，注意到 σ 包含 3142; 它可以通过删除 σ中的“3”得到。

带符号排列类 是一个对包含封闭的带符号排列集合。也就是说，若 $\mathcal{C}$ 是带符号排列类，且 $\sigma \in \mathcal{C}$ 与 $\sigma \ge \pi$，则 $\pi \in \mathcal{C}$。

定义 2.2（下集）。给定一个带符号排列 $\sigma \in B_n$，我们称 σ 的下集为集合

$$\{\pi \in B_n \mid \pi \le \sigma\}.$$

An另一个对我们的算法行为至关重要的思想是膨胀的概念 n.

定义 2.3（膨胀）。给定一个带符号排列 $\pi \in B_n$ 和$\mathbf{v} \in \mathbb{Z}^n_{\ge 0}$，将 π 膨胀 by $\boldsymbol{v}$ 的过程，是将每个字符 $\pi(i)$ 替换为长度为$\mathbf{v}(i)$ 的单调区间。如果 $\pi(i)$ 为正，则该区间必须递增；如果 $\pi(i)$ 为负，则必须递减。此外，区间中的每个元素都必须与 $\pi(i)$ 具有相同的符号。最后，这些区间的顺序必须与π中各项的顺序一致。

将 π 膨胀$\mathbf{v}$ 后，我们记作 $\pi[\mathbf{v}]$。例如，$12[\langle \underline{3}, 4\rangle] = 321\underline{4567}$。我们指出，通过 0 进行膨胀用于移除字符；例如，注意到 $213[\langle 2, 3, 0\rangle] = 45123$，并且不存在与 3 对应的区间。此外，可以看出，当且仅当 $\sigma[\mathbf{v}] = \pi$ 时 $\pi \le \sigma$，其中$\mathbf{v}$ 是一个每个分量均为 0 或 1 的向量。

定义 2.4（填充）。如果 $\sigma = \pi[\mathbf{v}]$ 且 $\mathbf{v} \in \mathbb{Z}^m_{>0}$，我们称 σ 填充 π。本质上，填充带符号排列 π 的过程，与用一个所有条目均为正的向量 $\mathbf{v}$ 对其进行膨胀的过程相同。在此语境下，我们也称$\mathbf{v}$ 填充 π。

这只是稍 微严格一些的约束，比膨胀更严格，但它能保证任何由此得到的带符号排列 通过填充 π得到 都与 π 具有相同的整体“形状”。

定义 2.5（网格类）。带符号排列 π 的网格类 记作 Grid(π)，是集合 $\{\pi[\mathbf{v}]:\mathbf{v} \in \mathbb{Z}^n_{\ge 0}\}$。此外，带符号排列集合 Π 的网格类记作 Grid(Π)，是集合 $\{\pi[\mathbf{v}] : \pi \in \Pi, \mathbf{v} \in \mathbb{Z}^n_{\ge 0}\}$。

换句话说，Grid(Π) 是通过膨胀每个$\pi \in \Pi$而得到的所有带符号排列的集合。网格类的重要性在于它们与带符号排列类之间的关系。事实上，我们有如下观察。

命题 2.1 . 设 Π 是一个带符号排列集合。那么，*Grid*(Π) 是一个带符号排列类。

证明。令 $\pi[\mathbf{v}] \in$ 属于网格 Grid(Π)。若 $\tau \leq \pi[\mathbf{v}],$ ，则 τ 具有形式 $\pi[\mathbf{v}']$，其中 $\mathbf{v}'$ 是一个各分量均不大于 $\mathbf{v}$ 的向量。事实上，我们可以从 $\mathbf{v}$ 的每个分量中减去相应单调区间中被移除的字符数，从而得到 $\mathbf{v}'$ 。例如，若 $\pi = 123$且 $\mathbf{v} = \langle 3, 3, 3\rangle$，则 $\pi[\mathbf{v}] = 123654789$。如果从 $\pi[\mathbf{v}]$ 中移除前两个字符和最后四个字符，并进行标准化，我们就得到 $\tau = \underline{1}3\underline{2}$。实际上，我们从 $\pi[\mathbf{v}]$ 的第一个单调区间中移除了两个元素，从第二个区间中移除了一个元素，从第三个区间中移除了三个元素。事实上，我们注意到 $\tau = \pi[\langle 1, 2, 0\rangle]$。因此，根据 Grid(Π) 的定义， $\tau = \pi[\mathbf{v}']$ 也属于 Grid(Π)，从而 Grid(Π) 是一个符号排列类。

□

3 算法

算法以有限个带符号排列 Π 作为输入。它输出一个多项式，用于枚举网格 Grid(Π) 中的元素。本节将介绍我们的算法，解释其中的每个步骤，讨论其运行时间，并通过一个示例演示这一过程。

完备化步骤

定义 3.1 （带符号排列完备集）。 如果网格 Grid(Π) 中的每个元素都可以通过填充某个 $\pi \in \Pi$ 得到，我们就称带符号排列的集合 Π 是完备的。换言之，如果对于每个$\sigma \in$ Grid(Π)，都存在一个长度为 m 的带符号排列 $\pi \in \Pi$ 和一个向量 $\mathbf{v} \in \mathbb{Z}^m_{>0}$，使得 $\sigma = \pi[\mathbf{v}]$，那么 Π 就是完备的。

我们想强调，应当填充带符号排列，而不是简单地将其膨胀，因为如前所述，用含有零分量的向量膨胀带符号排列会使其丢失部分形状。注意，任何带符号排列用零向量膨胀后都会得到空排列。我们希望避免对此类排列重复计数。

我们首先尝试用一个完备集合 Π' 替换原集合 Π，使得 Grid(Π) = Grid(Π′)。然后，计算 Grid(Π) 中元素的数量，就等价于计算填充 Π' 中某个元素的带符号排列的数量。以下定理说明了如何得到这个集合 Π'。

引理 3.1. 任何带符号排列的下集都是完备的。

证明。 设 Π 是带符号排列的一个下集，且 $\sigma \in$Grid(Π)。 根据网格类的定义，存在一个长度为 m 的 $\pi \in \Pi$ 以及一个向量 $\mathbf{v} \in \mathbb{Z}^m_{\geq 0}$，使得 $\sigma = \pi[\mathbf{v}]$。如果 $\pi = \pi(1) \cdots \pi(m)$，我们可以从$\pi(1) \cdots \pi(m)$中移除每个与$\mathbf{v}$的某个分量相对应、且满足$\mathbf{v}(i) = 0$的 $\pi(i)$。移除这些项并进行标准化后，得到一个排列 $\pi' \leq \pi$；类似地，移除$\mathbf{v}$的所有零项后，可以得到一个向量 $\mathbf{v}'$ 。由于 Π 是一个下集，因此 $\pi' \in \Pi$。此外， $\sigma = \pi'[\mathbf{v}']$，所以$\sigma$ 填充 π'。 □

因此，给定输入 Π，算法的完备化步骤会将 Π 替换为 Π 中各元素的下集之并集。

压缩步骤

我们提到，完备化步骤的动机是避免对通过一个含有一个或多个零项的向量进行膨胀而得到的排列重复计数。但仍有一些排列会被重复计数。注意， $12[\langle 1, 3\rangle]$、 $12[\langle 2, 2\rangle]$和 $12[\langle 3, 1\rangle]$ 都会生成 1234。

定义3.2 （紧致带符号排列）。 我们称 $\pi \in \Pi$ 是紧致的，如果 $\pi' < \pi$ 意味着 Grid(π') ⊂Grid(π)。我们用“<”和“⊂”分别表示严格不等式和真子集。

例如，12 不是紧致的，因为 $1 < 12$，然而 Grid(1) = Grid(12)。定义告诉我们，可以放心地从集合 S 中移除这类非紧致带符号排列，而不会对 Grid(S) 产生任何影响。下面的命题和引理刻画了这些紧致排列，并说明紧致性对重复计数有何影响。

命题 3.2。 设 π 是一个带符号排列。则以下各项等价 $t.$

（a） π 是紧致的，

（b） π 不包含与 12 或 21序同构的区间。

（c）如果 $\pi[\boldsymbol{v}_1] = \pi[\boldsymbol{v}_2]$，其中$\boldsymbol{v}_1$ 具有正分量，而$\boldsymbol{v}_2$ 具有非负分量，那么$\boldsymbol{v}_1 = \boldsymbol{v}_2$。

证明。（a）蕴含（b）是因为 12 和 21 显然都不是紧致的。假设$\pi(i)\pi(i+1)$与 12 或 21 序同构。那么任意 $\pi[\mathbf{v}]$ 都可以写成 $\pi'[\mathbf{v}']$，其中 π' 是移除$\pi(i+1)$ 后的结果，而 $\mathbf{v}'$ 则是将 $\mathbf{v}(i)$ 和 $\mathbf{v}(i+1)$ 合并为一个单项并对它们求和后的结果。由于 $\pi' < \pi$, π 不是紧致的。

此外，假设（c）成立，并考虑带符号排列 $\sigma = \pi[\mathbf{1}]$，那么不存在带符号排列 $\sigma' < \sigma$ 使得 $\sigma \in$Grid(σ')，因此 π 是紧致的。所以，（c）蕴含（a）。

接下来只需证明 (b) 推出 (c)，对此我们通过逆否命题来证明。假设存在一个带符号排列 π，满足 $\pi[\mathrm{v}_1] = \pi[\mathrm{v}_2]$，其中 $\mathrm{v}_1 \in \mathbb{Z}^m_{>0}$ 且 $\mathrm{v}_2 \in \mathbb{Z}^m_{\geq 0}$，$\mathrm{v}_1 \neq \mathrm{v}_2$,，并且 m 是 $\pi.$ 的长度。注意，v_1 和 v_2 至少有两个分量不同，否则 $\pi[\mathrm{v}_1]$ 和 $\pi[\mathrm{v}_2]$ 的长度就不会相等。令 i 为满足 $\mathrm{v}_1(i) \neq \mathrm{v}_2(i)$ 的最小指标，令 j 为大于 i 且满足 $\mathrm{v}_1(j) \neq \mathrm{v}_2(j)$ 的最小指标。由于 $\pi[\mathrm{v}_1] = \pi[\mathrm{v}_2]$, $\mathrm{v}_1(i) \neq \mathrm{v}_2(i)$，且 $\mathrm{v}_1(j) \neq \mathrm{v}_2(j)$，因此 $\pi(i)$ 和 $\pi(j)$ 属于 π 中的同一个单调区间。又由于该区间的长度至少为二，我们知道 π 包含一个与 12 或 21 序同构的区间。因此 (b) 推出 (c)，证明至此完成。 □

命题 3.2 的一个实际意义是，条件 (b) 可以很容易地实现，以检查一个排列是否紧致。我们只需检查 $\pi(i+1) - \pi(i) = 1$ for $1 \leq i \leq \mathrm{len}(\pi) - 1$。我们在算法1的压缩步骤中明确说明了这一点

引理 3.3. 任意带符号排列恰好填充一个紧致带符号排列。也就是说，若 $\sigma \in B_n$，则存在唯一的紧致 $\pi \in B_m$，满足 $m \leq n$，且 $\boldsymbol{v} \in \mathbb{Z}^m_{>0}$ 满足 $\sigma = \pi[\boldsymbol{v}]$。

证明。我们将用反证法。假设 π 和 π' 是两个不同的紧致排列，且 v、$\mathbf{v}'$ 是两个分量为正的向量，满足 $\pi[\mathbf{v}] = \sigma = \pi'[\mathbf{v}']$。由于 $\pi \neq \pi'$，存在满足条件的最小指标 i：$\mathbf{v}(i)\mathbf{v}'(i) > 1$ （这是简略写法，意为 $\mathbf{v}(i) > 1$ 或 $\mathbf{v}'(i) > 1$），且 $\mathbf{v}(i) \neq \mathbf{v}'(i)$。由于 $\pi[\mathbf{v}] = \pi'[\mathbf{v}']$，因此 π 或 π' 包含一个区间，$\pi(i)\pi(i+1)$或 $\pi'(i)\pi'(i+1)$，它与 12 或 21 序同构；这与 π 和 π' 是紧致排列的假设矛盾。 □

考虑任意一个带符号排列 π。引理 3.3 告诉我们，π 恰好填充一个紧致的带符号排列；命题 3.2 (c) 告诉我们，π 以唯一的方式填充这个紧致的带符号排列。因此，经过算法的完备化和紧化步骤后，我们得到一个集合 S，使得 Grid(S) 中的元素与数对 (π，$\mathbf{v}$) 之间存在双射，其中 $\pi \in S$ 和 $\mathbf{v}$ 填充 π。换言之，我们证明了以下定理。

定理 3.4。 设 Π 是一个带符号排列的集合。那么

$$Grid(\Pi) = \bigsqcup_{\pi \in S} Grid(\{\pi[\boldsymbol{v}] : \boldsymbol{v} \in \mathbb{Z}_{>0}^{len(\pi)}\}),$$

其中 S 是应用算法的完备化和紧化步骤所得的结果，而“⊔”表示不交并。

枚举步骤

现在，令 S 为执行完备化和紧化步骤后得到的集合。为了枚举 Grid(S)，只需枚举填满每个 $\pi \in S$ 的向量集，并将它们全部相加。对于每个 $\pi \in S$,

我们希望求出填满 π的向量的生成函数。填满长度为 m 的带符号排列的向量的生成函数，即所有向量 $\mathbf{v} \in \mathbf{Z}_{>0}^m$，为

$$\left(\frac{x}{1-x}\right)^m.$$

因此，我们可以将 |网格(S)| 的生成函数写成如下形式：

$$\sum_{\pi \in S} \left(\frac{x}{1-x}\right)^{\mathrm{len}(\pi)}. \tag{3.1}$$

让我们回顾一下，生成函数x^m $(1-x)^m$ 有一个众所周知的显式公式，即 ：

$$\frac{x^m}{(1-x)^m} = \sum_{i \geq 0} \binom{i+m-1}{m-1} x^{i+m}.$$

因此，我们可以轻松地提取表达式 (3.1) 中 x^n 的系数，从而得到 |网格(Π) $\cap B_n$|的计数多项式。

我们在算法1中给出了算法的伪代码。我们还在 [14]中实现了该算法。

算法1： 枚举网格(Π)

输入： 带符号排列集合 Π
输出： 多项式 $P(n)$，枚举 |网格(Π) $\cap B_n$|
/* 完备化步骤: 按包含序加入所有排列 $\leq \pi \in S$。 */
集合 $S = \emptyset$;
对于 $\pi \in \Pi$，**执行**
  将所有排列 $\pi' \leq \pi$ 添加到 S
结束循环
/* 紧化步骤: 移除所有满足 12，$21 \leq \pi$ 的排列 $\pi \in S$。 这相当于
  检查是否存在某个索引 i，使得 $\pi(i+1) - \pi(i)$ 等于 1 */
对 $\pi \in S$ **执行**
  对 i 从 1 到 $len(\pi)$**执行**
    如果 $\pi(i+1) - \pi(i) = 1$ **则**
      从 S中移除 π
  结束循环
结束循环
/* 枚举步骤: 求得计数多项式 */
$gen_fcn = 0$;
对 $\pi \in S$ **执行**
  $gen_fcn \leftarrow gen_fcn + \left(\frac{x}{1-x}\right)^{\mathrm{len}(\pi)}$
结束循环
/* poly 返回在生成函数中给出 x^n 系数的多项式 */
$P(n) \leftarrow poly(gen_fcn)$;
返回 $P(n)$

例 3.3. 我们用输入 $\Pi = \{213\}$ 来演示算法1的运行过程。

完备化 第一步是将所有带符号排列 $\pi \leq 213$ 添加到 S 中。完备化步骤结束时，$S = \{\varepsilon, 1, 1, 12, 12, 21, 213\}$，其中我们使用 ε 表示空排列，空排列包含于任意排列中。

紧化 现在，我们移除 S 中不紧致的带符号排列。根据命题 3.2，我们需要移除那些包含与 12 或 21 序同构的区间的排列。在此情况下，我们移除 12，得到 $S = \{\varepsilon, 1, \underline{1}, 12, 21, \underline{\underline{213}}\}$。

枚举 生成函数如下

- 1对于空排列 ε
- $\frac{x}{1-x} = \sum_{n\geq 0} x^{n+1}$ 对于 $\underline{1}$ 和 $\underline{\underline{1}}$，
- $\frac{x^2}{(1-x)^2} = \sum_{n\geq 0}(n+1)x^{n+2}$ 对于 $\underline{\underline{12}}$ 和 $\underline{\underline{\underline{21}}}$，以及
- $\frac{x^3}{(1-x)^3} = \sum_{n\geq 0}\binom{n+2}{2}x^{n+3}$ 对于 $\underline{\underline{\underline{213}}}$。

因此，在总和展开式中提取 x^n 中 $n \geq 1$ 的系数

$$1 + \frac{2x}{1-x} + \frac{2x^2}{(x-1)^2} + \frac{x^3}{(1-x)^3},$$

我们得到该多项式

$$P(n) = 2 + 2(n-1) + \binom{n-1}{2} = \frac{n^2}{2} + \frac{n}{2} + 1.$$

3.1 算法1的运行时间

算法1的运行时间取决于带符号排列的输入集合 Π 。我们将对 Π 的描述简化为有序对 $(n,\ l)$，其中 n 是 Π 中排列的数量，l 是 Π 中最长排列的长度。

完备化和紧化步骤将添加包含在 Π 某个元素中的所有紧排列。每个 $\pi \in \Pi$ 至多包含 2^l 个子排列。因此，最多执行 $n \times 2^l$ 次操作，每次操作包括删除排列中的一个元素、检查所得排列是否为紧致排列，并在是时将其添加到集合中。删除元素和检查紧致性都需要 $O(n)$ 次操作，而添加到集合中的时间为常数。因此，该步骤的时间复杂度为 $O(n^2 2^l)$。

枚举步骤将为每个 $\pi \in \Pi$ 计算一个多项式，并计算总和。最多计算 $n \times 2^l$ 个多项式，每个多项式的计算时间为 $O(n)$。因此，该步骤的时间复杂度也为 $O(n^2 2^l)$。由此，我们得出算法1的时间复杂度为 $O(n^2 2^l)$。

3.2 与 HVA 的差异

本节比较用于统计排列网格类中元素的 HVA 与用于统计带符号排列网格类中元素的算法1。事实证明，对于带符号排列，整个过程得到了简化。

我们的设定与之相关的首要区别是，我们不需要钉排列。在 [10] 中，钉排列被定义为由不同字符组成的字符串，这些字符取自 1 到 n，且每个字符都带有 $\bullet$，$+$，$-$ 标记。在 [10], 中，如果某个字符带有 $+$ 标记，就用递增区间膨胀这些钉排列并填充它们；如果带有 $-$ 标记，则用递减区间；如果带有 $\bullet$ 标记，则用包含 1 个或 0 个元素的区间。在我们的设定中，不需要 $\bullet$ 标记，因此我们使用带符号排列，而非钉排列。

HVA 可概括为以下五个步骤。其输入是一个由 P 个钉排列组成的集合。

完备化 在这一步中，作者将 S中的钉排列所包含的所有钉排列添加到 S 中。我们的完备化步骤与此类似，但使用带符号排列而非钉排列。

紧化 在紧化步骤中，HVA 会从 S中移除非紧致的钉排列。我们的紧化步骤与此类似，但在算法1中，判断要移除哪些排列的条件更简单。

清理HVA 的枚举过程中存在一些冗余，紧化步骤并未将其移除。作者需要从 S 中移除并非清理过的钉排列（见 [10, 第 2.3] 节）。由于引理 3.3，算法1 不需要这一步。

合并在合并步骤中，HVA 会对清理过程中生成的某些凸集取并集。算法1 不使用 HVA 中采用的凸集，因此对于带符号排列，可以安全地跳过这一步。

枚举HVA 和算法1 的这一步相同。

此外，值得指出的是，HVA 适用于“ n 的值足够大”的情况。HVA 要求 n 满足这一条件，是因为若钉排列 $\tilde{\rho}$ 的最小填充向量$\mathbf{v}$ 要求$\mathbf{v}(i) = 2$ ，则需满足以下条件： $\tilde{\rho}(i)$ 被一个 $\bullet$ 修饰。相比之下，我们的最小填充向量始终为$\mathbf{1}$ ，因此我们的多项式适用于 n 的所有取值。

4 应用

4.1 煎饼问题

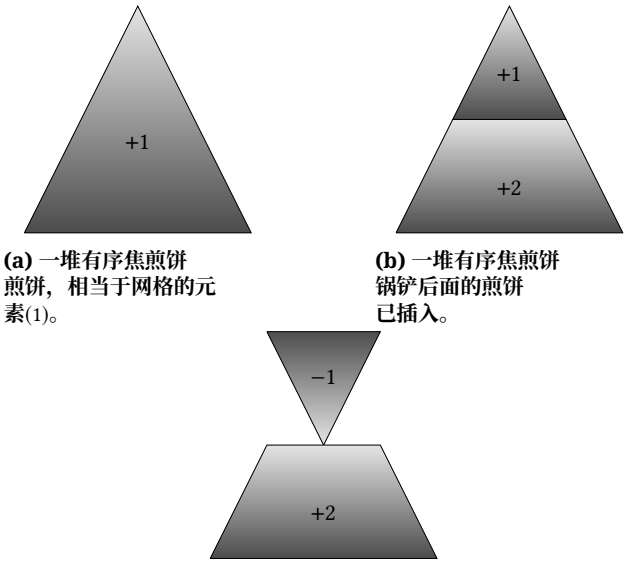
给定一个用单行记法表示的排列 $\pi \in S_n$ ，前缀反转 f_i 是一个作用于排列的函数，其中 $2 \leq i \leq n$ ，它会“反转” π的前 i 个字符。更准确地说，若 $\pi = \pi(1)\pi(2)\cdots\pi(n)$，则$f_i(\pi) = \pi(i)\pi(i-1)\cdots\pi(1)\pi(i+1)\cdots\pi(n)$，其中 $2 \leq i \leq n$。在其最初的表述中， [6], 的煎饼问题要求求出对给定排列进行排序所需的最少前缀反转次数（即前缀反转距离）。在微软比尔·盖茨唯一撰写过的学术论文中，盖茨和 Papadimitriou [7] 给出了前缀反转距离的首个非平凡界。此界后来在 [4] 中得到改进。此外，煎饼问题后来在 [7]中扩展到了带符号排列的情形，此时前缀反转还会将字符的符号从“+”切换为“–”，反之亦然。对于 B_n，有 n 种前缀反转 f_i^B, $1 \leq i \leq n$，它们会反转单行记法表示的排列的前 i 个字符，并切换这些字符的符号。例如， $f_1^B(132) = 132$ 和$f_2^B(132) = 312$。求出对带符号排列进行排序所需的最少带符号前缀反转次数，这个问题称为焦煎饼问题。

值得一提的是，是什么让我们关注这个项目。在 [3], 采用容斥原理以及对煎饼图和焦煎饼图中的小圈进行分类等纯计数方法后， [2, 3], 作者给出了一个多项式，用于计算恰好需要翻转四次才能整理好的 n煎饼堆的数量，以及需要翻转四次才能整理好的烧焦煎饼堆的数量。此外，作者还猜想了其他多项式，据此可以计算在 k 较小的情况下，恰好需要翻转 k 次才能整理好的煎饼堆或烧焦煎饼堆的数量（见 [3, 第 6] 节）。后来，V. Vatter 在 [3]向作者介绍了他们的 HVA [10] 。随即，两个问题便显而易见了。

Q1：能否修改 HVA，使其适用于带符号排列？以及

Q2：由于 HVA 的特性是在 n足够大的情况下才起作用，因此 HVA 漏掉了 [3] 中得到的一些数值。为什么 HVA 只在 n 足够大的情况下起作用，而不是对 n的所有取值都适用?

Q1 的答案是“是”，算法1 是针对枚举带符号排列类而定制的 HVA 修改版本。此外，关于 Q2，与 HVA 不同，算法1 适用于 n 的所有取值，而不仅仅适用于“足够大的 n”，这是因为在带符号排列的情况下，最小填充向量始终为 $\mathbf{1}$。在本小节接下来的内容中，我们应用算法1 来获得最多经过 k 次翻转即可排序的烧焦煎饼堆数量。特别地，我们证明了 [3, 猜想 6.2，6.3]。



(a) 一堆有序焦煎饼煎饼，相当于网格的元素(1)。

(b) 一堆有序焦煎饼锅铲后面的煎饼已插入。

(c) 翻转锅铲上方的所有部分后得到的煎饼堆，相当于网格(12)**中的元素。**

图1

4.1.1 从烧焦煎饼堆到网格类

我们知道，可以将一堆烧焦煎饼表示为一个带符号排列，因此我们的下一个目标是将能够在 $\leq k$ 次翻转内排序的烧焦煎饼堆表示为一个网格类。很容易看出，Grid(1) $= \{\varepsilon, 1, 12, 123, ...\}$ 表示能够在 ≤ 0 次翻转内排序的煎饼堆，但对于 k 的其他取值又该如何呢？在本节中，我们将说明如何选择 Π_k ，使得 Grid(Π_k) 成为需要 $\leq k$ 次前缀反转才能排序的带符号排列类。

首先，我们不妨花点时间直观地想一想网格类代表什么。在这个算法中，尤其是针对煎饼问题，我们可以根据上下文，用两种不同的方式来表示带符号排列。假设给定一个网格类 Grid(Π)。那么 Π和 Grid(Π) 都是带符号排列的集合，但我们可以用不同的方式来理解其中的元素。如果 $\pi \in$Grid(Π)，我们会将 π 的每个条目视为一个烧焦煎饼。此外，如果 $\pi \in \Pi$，我们也可以将 π 的每个条目视为由无限多个连续煎饼构成的块，且块内已经排好序。与单个烧焦煎饼类似，每个块都有方向和相对顺序，可以视为一个单一实体。从概念上讲，当我们膨胀 Π 中的一个元素时，只是在为每个块指定一个非负整数大小。

以 Grid(12) 为例。我们将 12 理解为一个倒置的煎饼块，置于一个更大的正置煎饼块之上，而 Grid(12) 中的每个带符号排列都符合这种描述。例如，$12[(3,3)] = 321456$，我们可以用类似于对 12 排序的方式对 321456 排序。

在本节余下的讨论中，我们将带符号排列可视化为块的堆叠，而非煎饼堆。这种直觉将帮助我们选择 Π_k。

要得到任何只需再翻转一次就能排好序的煎饼堆，我们先从一个有序堆叠开始，在堆叠中间某处插入锅铲，然后翻转。换句话说，我们从图1a中的带符号排列 1 开始。接着插入锅铲，将块分成两部分，如图1b所示。最后进行翻转，得到图1c中的 12。因此，直观来说，我们预计 Grid(12) 表示只需 ≤ 1 次翻转即可排好序的煎饼堆。

我们可以递归地继续这一过程。要得到任何距离恒等排列只差两次翻转的煎饼堆，我们先从距离恒等排列只差一次翻转的堆叠开始，在任意位置插入锅铲，然后翻转。但距离恒等排列只差一次翻转的堆叠由 12 表示，它由两个块组成。我们可以在任一块的中间翻转，这意味着会得到两个不同的带符号排列。如果在 1 块内翻转，会得到 213；如果在 2 块内翻转，也会得到 213。这些堆叠分别如图 2a 和图 2b 所示。因此，我们预计 Grid($\{213, 213\}$) 表示只需 ≤ 2 次翻转即可排好序的煎饼堆。

这个过程很简单，我们可以继续进行此递归，为任意 k生成 Π_k 。简单的归纳论证表明， Π 将包含 $k!$ 个元素，每个元素的长度为 $k+1$。

我们描述的递归过程旨在反映以 k 次翻转重新排列一个有序煎饼堆的所有可能方式。然而，我们的过程要求恰好进行 k 次翻转，而且每次翻转都必须发生在一个块的中间，将其一分为二。因此，有些煎饼堆显然不属于 Grid(Π_k)：任何仅通过 $< k$ 次翻转生成的堆，或通过至少一次未将煎饼“分开”的翻转生成的堆。

虽然我们的过程不允许在块的边缘处翻转，但允许在无限接近块边缘的位置翻转。换句话说，如果我们在一个块的中间翻转，然后将由此产生的两个块之一膨胀零倍，就模拟了在原块边缘处翻转。我们可以用这个技巧在块与块之间翻转，这等同于一次不会将煎饼分开的翻转。我们也可以用这个技巧在堆叠的最顶端或最底端翻转；在最顶端翻转等同于完全不翻转。以上分析表明，Grid(Π_k) 必须包含所有前缀反转距离为$\leq k$的煎饼堆，我们将在一个命题中对此进行形式化表述。

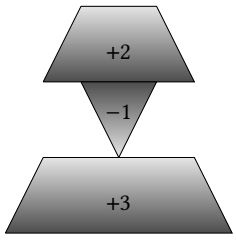
命题4.1。 对于每个 $k \geq 0$，所有前缀反转距离为 $\leq k$ 的烧焦煎饼堆所构成的集合等价于一个符号网格类。

证明。具体而言，该网格类是 Grid(Π_k)。我们从 $\Pi_0 - \{1\}$ 开始递归生成它。对于每个 $\pi \in \Pi_k$ 以及每个 $1 \leq i \leq k+1$，我们首先用向量 $e_i + \mathbf{1}$ 对 π 进行膨胀，然后对所得结果应用前缀反转 f_i，从而生成新的排列 π_i。这里，e_i 表示标准基向量，其中第 i 个分量为 1，其余分量均为 0。定义 Π

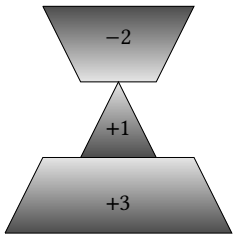
首先，我们证明 Grid(Π_k) 包含所有前缀反转距离为 $\leq k$ 的带符号排列。如果π 的前缀反转距离为 $\leq k$，那么它是某个紧排列的膨胀，该紧排列的前缀反转距离为 k（但不一定唯一）。根据上述讨论，这个紧排列包含于 Π_k中，因此 π 属于 Grid(Π_k)。

在证明逆命题之前，我们将前缀反转的概念扩展到向量。对$\mathbf{v}$ 应用 f_i ，会反转$\mathbf{v}$的前 i 个分量。

现在，我们证明 Grid(Π_k) 中的每个带符号排列的前缀反转距离为 $\leq k$。任意排列 $\sigma \in$Grid(Π_k) 都可以写成 $\pi[\mathbf{v}]$ ，其中 $\pi \in \Pi_k$ 且 $\mathbf{v} \in \mathbb{Z}^n_{\geq 0}$。根据我们的构造方式



(a) 由以下操作得到的煎饼堆在……内部进行翻转 12 的顶部块。



(b) 由以下操作得到的煎饼堆从内部翻转得到的结果底部的块12。

图 2

Π_k，显然，π 是对长度为 $k+1$ 的恒等排列执行一系列 $\leq k$ 次前缀反转 s 后的结果。因此，我们可以对长度为 $\mathsf{len}(\sigma)$ 的恒等排列执行类似的一系列前缀反转 s' 来生成 σ，由此 σ 的前缀反转距离为 $\leq k$。这一序列 s' 可以明确描述如下。我们遍历 s，对于每个前缀反转 f_i，s' 中对应的前缀反转为 f_x，其中 x 是 $\mathbf{v}$ 中前 i 个分量之和。然后，在应用 f_x 之后，我们对 $\mathbf{v}$ 应用 f_i，以整理每个膨胀区间的大小，并对 s 中的下一个前缀反转重复这一过程。例如，考虑 $\pi = 213$ 和 $\mathbf{v} = \langle 1, 2, 3\rangle$。注意，$f_1 f_2(\pi) = 123$，而将 $\pi[\mathbf{v}] = 312456$ 排序所对应的翻转序列以 $f_{1+2} = f_3$ 开始。接着，我们对 $\mathbf{v}$ 应用 f_2，得到 $\langle 2, 1, 3\rangle$；由于 2 是 $\langle 2, 1, 3\rangle$ 的第一个分量，因此对 f_1 执行对应的前缀反转是 f_2。于是，对 π 应用 f_2，然后应用 f_1，对应于对 $\pi[\mathbf{v}]$ 应用 f_3，然后应用 f_2。事实上，$f_2 f_3(312456) = 123456$。 □

既然我们已经证明，具有固定前缀反转距离的排列集合是一个网格类，就可以在下一节中应用算法1，计算某些 n 取值下对这些排列进行计数的多项式。我们指出，这也得到了 [3, 定理4.3]；此前该结果是利用基本计数技巧而非符号计算得到的。

4.1.2 结果

我们使用该算法得到计数多项式 $R^B_{\leq k}(n)$，该多项式给出需要 k 次或更少翻转才能排好序的不同煎饼堆叠的数量。我们计算了 k 从1到10时的多项式。结果如下。为便于展示，我们将多项式表示为系数数组，其中索引为 i、满足 $1 \leq i \leq k+1$的项，表示 $R^B_{\leq k}$中 n^{i-1} 的系数。例如，$R^B_{\leq 3}(n) = 1 + n - n^2 + n^3$，我们将其表示为 $[1, 1, -1, 1]$。

$$
\begin{aligned}
R^B_{\leq 1}(n) &= [1, 1]\,,\\
R^B_{\leq 2}(n) &= [1, 0, 1]\,,\\
R^B_{\leq 3}(n) &= [1, 1, -1, 1]\,,\\
R^B_{\leq 4}(n) &= \left[1, -\frac{1}{2}, 3, -\frac{5}{2}, 1\right],\\
R^B_{\leq 5}(n) &= \left[1, \frac{1}{2}, -\frac{25}{6}, \frac{17}{2}, -\frac{29}{6}, 1\right],\\
R^B_{\leq 6}(n) &= \left[1, \frac{299}{30}, -5, -\frac{73}{4}, 21, -\frac{463}{60}, 1\right],\\
R^B_{\leq 7}(n) &= \left[1, -\frac{3529}{30}, \frac{24697}{120}, -\frac{3167}{48}, -\frac{889}{16}, \frac{3569}{80}, -\frac{2699}{240}, 1\right],\\
R^B_{\leq 8}(n) &= \left[1, \frac{92843}{84}, -\frac{48217}{20}, \frac{1230329}{720}, -\frac{7787}{24}, -\frac{2659}{18}, \frac{10117}{120}, -\frac{77323}{5040}, 1\right],\\
R^B_{\leq 9}(n) &= \left[1, -\frac{1713461}{168}, \frac{28102741}{1120}, -\frac{3620111}{160}, \frac{52327853}{5760}, -\frac{13571}{12}, -\frac{997679}{2880}, \frac{163277}{1120}, -\frac{806941}{40320}, 1\right], \text{ and}\\
R^B_{\leq 10}(n) &= \left[1, \frac{29555642}{315}, -\frac{1264975307}{5040}, \frac{11803588051}{45360}, -\frac{77767535}{576}, \frac{307180691}{8640}, -\frac{4420823}{1440}, -\frac{22399579}{30240},\right.\\
&\qquad \left.\frac{948575}{4032}, -\frac{4576633}{181440}, 1\right].
\end{aligned}
$$

若要考虑前缀反转距离恰好为 k的烧焦煎饼堆，则计数多项式为 $R^B_{\leq k}(n) - R^B_{\leq k-1}(n)$。例如，

$$R^B_{\leq 4}(n) - R^B_{\leq 3}(n) = \frac{1}{2}(n-1)^2(2n-3),$$

这就是 [3, 定理 4.3]。事实上，我们能够确认 [3, 猜想 6.1]中给出的多项式都是正确的。为完整起见，我们在此列出该猜想，此处将其表述为一个命题。我们用R^B_k表示计数多项式，它统计前缀反转距离恰好为 k的带符号排列数量。

命题 4.2。若 $n \geq 1$，则

(i) $R^B_5(n) = \frac{1}{6}n(n-1)(n-2)(6n^2-17n+3)$，　*(ii)* $R^B_6(n) = \frac{1}{60}n(n-1)(n-2)($
$60n^3-343n^2+401n+284)$，　*(iii)* $R^B_7(n) = \frac{1}{240}n(n-1)(n-2)(n-3)(240n^3-1499n^2+925n+5104)$，
(iv) $R^B_8(n) = \frac{1}{5040}n(n-1)(n-2)(n-3)(5040n^4-52123n^3+113415n^2+314716n-1027242)$，以及
(v) $R^B_9(n) = \frac{1}{40320}(n-1)(n-2)(n-3)(n-4)(40320n^5-444061n^4+644746n^3+6638777n^2-18991470n)$。

此外，[3, 猜想 6.3]现在可通过使用整数值多项式的格雷戈里-牛顿插值公式 推导为一个推论。

推论 4.3。若 k，$n \geq 1$则

$$R^B_k(n) = \sum_{j=1}^{k}\left(\sum_{i=0}^{k-j}(-1)^i\binom{i+j-1}{i}\binom{n}{i+j}\right)R^B_k(j),\ \textit{for } k \geq 1.$$

4.2 基因组重排

基因组是细胞中全部 DNA 指令的集合。基因组包含生物体的遗传信息；例如，在人类中，基因组由 23 对染色体组成。DNA 被划分为称为“基因”的区段，其中每个基因都是生物体某一特定特征的遗传配方。假设每个基因都是由核苷酸组成的不对称序列，我们可以将基因组看作基因序列，其中每个基因都有方向。

当突变改变作用于基因序列的生物体基因组时，进化就会发生。其中最常见的突变是区块反转，或简称反转，即将一整个基因区段反向排列。更正式地说，如果 $\pi = \pi(1)\pi(2)\cdots\pi(n) \in B_n$，则区块反转 $b_{i,\ j}$ 通过 $1 \leq i \leq j \leq n$ 变换π 为排列

$$\pi(1)\cdots\pi(i-1)\pi'(j)\pi'(j-1)\cdots\pi'(i+1)\pi'(i)\pi(j+1)\cdots\pi(n),$$

其中 π'_k 表示 $-\pi_k$ 对于 $i \leq k \leq j$。

如果我们想比较某个生物体与其祖先，可以用带符号排列表示它们的基因组。后代的 DNA 可以写成恒等排列，基于这种基因标记方式，我们也可以用类似的方式写出祖先的 DNA。生物学家很关心如何确定将给定的带符号排列变为恒等排列所需的最少反转次数，因为这可能代表了进化过程中发生的突变次数。在这一背景下，通过反转对排列排序具有生物学意义，相关研究见 [5, 8, 9]。

我们关注这样一类带符号排列：对其进行 $\leq k$ 次区块反转即可排好序（即，反转距离 为 k）。这些类也等价于网格类，我们通过采用与上一节处理煎饼问题时类似的方法证明了这一点。事实上，在第 4.1.1 节中，我们通过模拟对 Π_{k-1} 中每个带符号排列的每个元素进行一次前缀反转，递归地得到了集合 Π_k。对于这个问题，我们通过选择两个块（可以是同一个块）作为分割点，并反转中间部分，来模拟块反转。

命题 4.4。对于每个 $k \geq 0$，所有反转距离为 $\leq k$ 的带符号排列所组成的集合都等价于一个网格类。

证明。 网格类是 Grid(Π_k)，我们从 $\Pi_0 = \{1\}$ 开始递归生成它。对于每个 $\pi \in \Pi_k$，以及每个 $1 \leq i \leq j \leq k$，我们先用向量 $e_i + e_j + \mathbf{1}$ 膨胀 π，再对结果应用区块反转$b_{i+1,\ j+1}$，从而生成一个新的排列 π_{ij}。定义 Π_{k+1} 为所有这些 π_{ij} 构成的集合。

我们首先证明，Grid(Π_k) 包含所有反转距离为 $\leq k$ 的带符号排列。如果 π 的反转距离为 $\leq k$，那么它是某个反转距离为 k 的紧排列的膨胀。这个紧排列包含于 Π_k 中，因此 π 属于 Grid(Π_k)。

我们将区块反转 的概念扩展到向量。因此，对$\mathbf{v}$ 应用 $b_{i,\ j}$，会将$\mathbf{v}$中第 i 个至第j个分量（含两端）反转。

现在我们证明，Grid(Π_k) 中的每个带符号排列的反转距离都是 $\leq k$。任意排列$\sigma \in$ Grid(Π_k) 都可以写成 $\pi[\mathbf{v}]$，其中 $\pi \in \Pi_k$ 和 $\mathbf{v} \in \mathbb{Z}^n_{\geq 0}$。我们通过对恒等排列依次执行 $\leq k$ 次块反转，生成了 π。因此，我们可以对长度为 len(σ) 的恒等排列执行相应的 $\leq k$ 次块反转序列 s' 来生成 σ，由此可知 σ 的反转距离为 $\leq k$。序列 s' 可以明确描述如下：我们遍历 s，对于每个块反转 $b_{i,\ j}$，在 s' 中对应的反转为 $b_{x,\ y}$；其中 x 是$\mathbf{v}$ 中前 $i-1$ 个分量之和加 1，而 y 是$\mathbf{v}$中前 j 个分量之和。接着，我们对$\mathbf{v}$ 应用 $b_{i,\ j}$，然后继续处理 s 中的下一个块反转。完成这一过程后，剩下的就是唯一确定的序列 s'，它可以对σ进行排序。

□

既然我们已经证明，具有固定反转距离的排列集合是一个网格类，那么在接下来的章节中，我们就可以应用算法1，计算某些 n取值下对这些排列进行计数的多项式。

4.2.1 结果

我们使用该算法求得计数多项式 $P^B_{\leq k}(n)$，它给出距离恒等排列 $\leq k$ 块反转的 B_n 的元素个数。我们求得了 k 从 1 到 5 时对应的多项式。结果如下。再次说明，我们用系数数组表示这些多项式，其中索引为 i 的项（$1 \leq i \leq 2k+1$）表示 $P^B_{\leq k}$ 中 n^{i-1} 的系数。

$$
\begin{aligned}
P^B_{\leq 1}(n) &= \left[1, \frac{1}{2}, \frac{1}{2}\right], \\
P^B_{\leq 2}(n) &= \left[1, \frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6}\right], \\
P^B_{\leq 3}(n) &= \left[1, \frac{1}{3}, \frac{35}{72}, \frac{7}{48}, -\frac{5}{144}, \frac{1}{48}, \frac{7}{144}\right], \\
P^B_{\leq 4}(n) &= \left[1, \frac{131}{420}, \frac{617}{1260}, -\frac{1}{120}, \frac{67}{1440}, \frac{53}{240}, -\frac{17}{360}, -\frac{41}{1680}, \frac{37}{3360}\right], \text{ and} \\
P^B_{\leq 5}(n) &= \left[1, \frac{331}{2520}, \frac{24727}{50400}, \frac{4703}{22680}, \frac{16945}{72576}, \frac{931}{17280}, -\frac{20059}{86400}, \frac{7267}{60480}, \frac{145}{24192}, -\frac{925}{72576}, \frac{3767}{1814400}\right].
\end{aligned}
$$

5 结论

我们的主要贡献是算法1，它枚举网格带符号排列类，我们在 [14]中实现了该算法。排列类大小的研究备受关注，其中包括证明 Stanley–Wilf 猜想和斐波那契二分定理等深刻成果。此外，Homberger 和 Vatter 提出了一种算法，可为所有满足$|\mathcal{C} \cap S_n|$ 在 n足够大时为多项式的排列类 $\mathcal{C}$ 输出一个次数为 n 的多项式。我们将网格排列类的概念扩展到

带符号排列，并证明如果 $\mathcal{C}$ 是网格带符号排列，那么 $|\mathcal{C} \cap B_n|$ 是由算法1计算得到的多项式。在这种情况下，无需 n 足够大。我们的网格带符号排列类涵盖了生物信息学中颇受关注的类别，尤其是与基因组重排相关的类别。事实上，在自然界中，物种进化的一种方式是对其基因组序列进行块反转。在这一生物信息学背景下，我们得出了彼此“距离” k 以内的潜在物种数量的界。作为未来的工作，我们希望获得类似于定理1.1、适用于带符号排列的结果。

致谢

我们感谢匿名审稿人提出的宝贵意见，这些意见提升了论文的质量。我们也感谢 V. Vatter，他向第一作者指出了 [10] 。

参考文献

[1] Michael Albert、M. Atkinson 和 Robert Brignall。多项式增长的排列类。组合年刊，11，2006年4月。[2] Sa´ul A. Blanco、Charles Buehrle 和 Akshay Patidar。焦煎饼图中的圈。离散应用数学，271：1–14，2019。[3] Sa´ul A. Blanco、Charles Buehrle 和 Akshay Patidar。关于需要四次翻转才能完成排序的煎饼堆数量。离散数学与理论计算机科学，第21卷第2期，《排列模式》2018，2019年11月。[4] B. Chitturi、W. Fahle、Z. Meng、L. Morales、C. O. Shields、I. H. Sudborough 和 W. Voit。通过前缀反转排序的一个 $(18/11)n$上界。理论计算机科学，410(36)：3372–3390，2009。[5] D. S. Cohen 和 M. Blum。关于烧焦煎饼排序问题。离散应用数学，61(2)：105–120，1995。[6] Harry Dweighter。问题与解答：初等问题：E2569。美国数学月刊，82(10)：1010，1975。[7] William H. Gates 和 Christos H. Papadimitriou。通过前缀反转排序的界。离散数学，27(1)：47–57，1979。[8] Sridhar Hannenhalli 和 Pavel A. Pevzner。将卷心菜变成芜菁：通过反转对带符号排列进行排序的多项式算法。*ACM* 学报，46(1)：1–27，1999。[9] Karmella Haynes、Marian Broderick、Adam Brown、Trevor Butner、James Dickson、W Harden、Lane Heard、Eric Jessen、Kelly Malloy、Brad Ogden、Sabriya Rosemond、Samantha Simpson、Erin Zwack、A. Campbell、Todd Eckdahl、Laurie Heyer 和 Jeff Poet。工程化细菌以解决焦煎饼问题。生物工程学报，2：8，2008年2月。[10] Cheyne Homberger 和 Vincent Vatter。关于多项式排列类的有效自动枚举。符号计算学报，76：84–96，2016。[11] Sophie Huczynska 和 Vincent Vatter。受限排列的网格类与斐波那契二分定理。组合学电子期刊，13，2006年3月。[12] Tom´a Kaiser 和 Martin Klazar。关于闭排列类的增长率。组合学电子期刊，9(2)：R10，2002–3。《排列模式》（奥塔哥，2003）。

[13] Adam Marcus 和 G´abor Tardos。禁排列矩阵与 Stanley–Wilf 猜想。组合理论杂志 *A*辑，107(1)：153–160，2004。[14]Daniel Skora。https://github.com/skora7/SignedPermutationClasses，2022。[最后修改时间：11月 2022]。[15] Vincent Vatter。排列类。载于 Mikl´os B´ona（编），枚举组合学手册，《离散数学及其应用》，第12章，第753–834页。Chapman and Hall/CRC，2015。