Algorithmica manuscript No.
(will be inserted by the editor)

Hamiltonicity of k-Sided Pancake Networks with Fixed-Spin: Efficient Generation, Ranking, and Optimality

Ben Cameron · Joe Sawada · Wei Therese · Aaron Williams

Received: October 18, 2021/ Accepted: date

Abstract We present a Hamilton cycle in the k-sided pancake network and four combinatorial algorithms to traverse the cycle. The network's vertices are coloured permutations $\pi = p_1p_2 \cdots p_n$, where each p_i has an associated colour in $\{0, 1, \ldots, k-1\}$. There is a directed edge (π_1, π_2) if π_2 can be obtained from π_1 by a "flip" of length ℓ, which reverses the first ℓ elements and increments their colour modulo k. Our particular cycle is created using a greedy min-flip strategy, and the average flip length of the edges we use is bounded by a constant. By reinterpreting the order recursively, we can generate successive coloured permutations in $O(1)$-amortized time, or each successive flip by a loop-free algorithm. We also show how to compute the successor of any coloured permutation in $O(n)$ time. Our greedy min-flip construction generalizes known Hamilton cycles for the pancake network (where $k = 1$) and the burnt pancake network (where $k = 2$). Interestingly, a greedy max-flip strategy works on the pancake and burnt pancake networks, but it does not work on the k-sided network when $k > 2$. In addition to our generation results, we provide ranking and unranking algorithms for our Hamiltion cycle that run in $O(n^2)$ time, and show that the cycle is globally optimal in terms of minimizing the total number of pancakes that are flipped. Finally, we characterize the Hamiltonicity of k-sided pancake networks with any fixed "spin" s.

B. Cameron
School of Computer Science, University of Guelph, Ontario, CANADA
E-mail: ben.cameron@uoguelph.ca

J. Sawada
School of Computer Science, University of Guelph, CANADA
E-mail: jsawada@uoguelph.ca

W. Therese
Dept. of Computer Science, Memorial University of Newfoundland, CANADA
E-mail: weitherese1@gmail.com

A. Williams*
Dept. of Computer Science, Williams College, Massachusetts, USA
E-mail: aaron.williams@williams.edu

1 Introduction

Many readers will be familiar with the story of Harry Dweighter, the harried waiter who sorts stacks of pancakes for his customers. He does this by repeatedly grabbing some number of pancakes from the top of the stack and flipping them over. For example, if the chef in the kitchen creates the stack , then Harry can sort it by flipping over all four pancakes , and then the top two .

This story came from the imagination of Jacob E. Goodman [14], who was inspired by sorting folded towels [38]. His original interest was an upper bound on the number of flips required to sort a stack of n pancakes. Despite its whimsical origins, the problem attracted interest from many mathematicians and computer scientists, including a young Bill Gates [18]. Eventually, it also found serious applications, including genomics [17].

A variation of the original story involves burnt pancakes. In this case, each pancake has two distinct sides: burnt and unburnt. When Harry flips the pancakes, the pancakes involved in the flip also turn over, and Harry wants to sort the pancakes so that the unburnt sides are facing up. For example, Harry could sort the stack by flipping all four , then the top two , and the top one . Similar lines of research developed around this problem (e.g. [8], [17]). The physical model breaks down beyond two sides, however, many of the same applications do generalize to "k-sided pancakes".

1.1 Pancake Networks

Interconnection networks connect single processors, or groups of processors, together. In this context, the underlying graph is known as the network, and classic graph measurements (e.g. diameter, girth, connectivity) translate to different performance metrics. Three networks related to pancake flipping are provided in Figure 1.

The pancake network $\mathbb{G}(n)$ is an undirected graph that was introduced in the 1980s [1] and various measurements were established (e.g. [21]). Its vertex set is the set of permutations of $\{1, 2, \ldots, n\}$ in one-line notation, which is denoted $\mathbb{P}(n)$. For example, $\mathbb{P}(2) = \{12, 21\}$. There is an edge between permutations that differ by a *prefix-reversal of length* ℓ, which reverses the first ℓ symbols. For example, $(3421, 4321)$ is the $\ell = 2$ edge between and . Goodman's original problem is finding the maximum shortest path length to the identity permutation. Since $\mathbb{G}(n)$ is vertex-transitive, this value is simply its diameter.

The burnt pancake network $\overline{\mathbb{G}}(n)$ is an undirected graph that was introduced in the 1990s [8]. Its vertex set is the set of signed permutations of $\{1, 2, \ldots, n\}$, which is denoted $\overline{\mathbb{P}}(n)$. For example, $\overline{\mathbb{P}}(2) = \{12, 1\overline{2}, \overline{1}2, \overline{1}\overline{2}, 21, 2\overline{1}, \overline{2}1, \overline{2}\overline{1}\}$ where overlines denote negative symbols. There is an edge between signed permutations that differ by a *sign-complementing prefix-reversal of length* ℓ, which reverses the order and sign of the first ℓ symbols. For example, $(\overline{2}\overline{1}34, 1234)$ is the $\ell = 2$ edge between and .

The *k-sided pancake network* $\mathbb{G}_k(n)$ is a directed graph that was first studied in the 2000s [26]. Its vertex set is the set of k-coloured permutations of $\{1, 2, \ldots, n\}$ in one-line notation, which is denoted $\mathbb{P}_k(n)$. For example, $\mathbb{P}_3(2)$ is illustrated below, where colours the 0, 1, 2 are denoted using superscripts, or in **black**, **red**, **blue**.

$$\mathbb{P}_3(2) = \{\mathbf{12}, \mathbf{1}\textcolor{red}{\mathbf{2}}, \mathbf{1}\textcolor{blue}{\mathbf{2}}, \textcolor{red}{\mathbf{1}}\mathbf{2}, \textcolor{red}{\mathbf{12}}, \textcolor{red}{\mathbf{1}}\textcolor{blue}{\mathbf{2}}, \textcolor{blue}{\mathbf{1}}\mathbf{2}, \textcolor{blue}{\mathbf{1}}\textcolor{red}{\mathbf{2}}, \textcolor{blue}{\mathbf{12}}, \mathbf{21}, \mathbf{2}\textcolor{red}{\mathbf{1}}, \mathbf{2}\textcolor{blue}{\mathbf{1}}, \textcolor{red}{\mathbf{2}}\mathbf{1}, \textcolor{red}{\mathbf{21}}, \textcolor{red}{\mathbf{2}}\textcolor{blue}{\mathbf{1}}, \textcolor{blue}{\mathbf{2}}\mathbf{1}, \textcolor{blue}{\mathbf{2}}\textcolor{red}{\mathbf{1}}, \textcolor{blue}{\mathbf{21}}\} \tag{1}$$

$$= \{1^0 2^0, 1^0 2^1, 1^0 2^2, 1^1 2^0, 1^1 2^1, 1^1 2^2, 1^2 2^0, 1^2 2^1, 1^2 2^2, 2^0 1^0, \ldots, 2^2 1^1, 2^2 1^2\}. \tag{2}$$

There is a directed edge from $\pi_1 \in \mathbb{P}_k(n)$ to $\pi_2 \in \mathbb{P}_k(n)$ if π_1 can be transformed into π_2 by a *colour-incrementing prefix-reversal of length* ℓ, which reverses the order and increments the colour modulo k of the first ℓ symbols. For example, $(\textcolor{red}{\mathbf{2}}\textcolor{blue}{\mathbf{1}}\mathbf{34}, \mathbf{1}\textcolor{blue}{\mathbf{2}}\mathbf{34}) = (2^1 1^2 3^0 4^0, 1^0 2^2 3^0 4^0)$ is a directed $\ell = 2$ edge. Since the vertices of the k-sided pancake network $\mathbb{G}_k(n)$ are k-coloured permutations, we will also refer to $\mathbb{G}_k(n)$ as the *k-coloured pancake network* and its vertices as *k-coloured pancakes*.

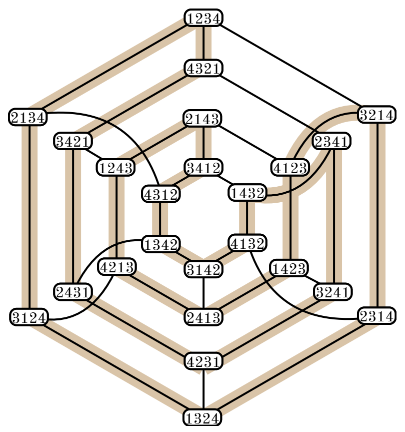
Notice that $\mathbb{G}(n)$ and $\mathbb{G}_1(n)$ are isomorphic, while $\overline{\mathbb{G}}(n)$ and $\mathbb{G}_2(n)$ are isomorphic, so long as we view each undirected edge as two opposing directed edges. It also bears mentioning that $\mathbb{G}_k(n)$ is a strongly connected directed Cayley graph, and its underlying group is the wreath product of the cyclic group of order k and the symmetric group of order n.

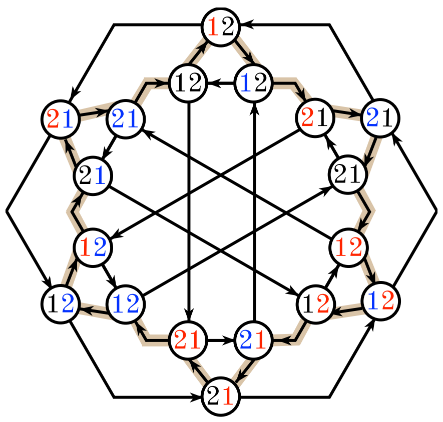
When the context is clear, or the distinction is not necessary, we use the term *flip* for prefix-reversal (when $k = 1$), sign-complementing prefix-reversal (when $k = 2$), and colour-incrementing prefix-reversal (when $k > 2$).

1.2 (Greedy) Hamilton Cycles

In this paper, we are not interested in shortest paths in pancake networks, but rather Hamilton cycles. There are myriad ways that researchers attempt to build Hamilton cycles in highly-symmetric graphs, and the greedy approach is perhaps the simplest (see Williams [46]). This approach initializes a path at a specific vertex, then repeatedly extends the path by a single edge. More specifically, it uses the highest priority edge (according to some criteria) that leads to a vertex that is not on the path. The path stops growing when the current vertex is only adjacent to vertices on the path. A Hamilton cycle has been found if every vertex is on the path, and there is an edge from the final vertex to the first vertex. Despite its simplicity, the approach is known to work on many well-known graphs [46].

We show that the greedy approach generates a Hamilton cycle in the coloured pancake network $\mathbb{G}_k(n)$ when we prioritize the edges by shortest flip length. More specifically, we start a path at $1^0 2^0 \cdots n^0 \in \mathbb{P}_k(n)$, then repeatedly extend it to a new vertex along the edge that corresponds to the shortest colour-incrementing prefix-reversal. We refer to this as the *greedy min-flip construction*, denoted $\mathsf{GreedyMin}_k(n)$, and it is illustrated in Figure 1. When $k = 1$, the cycle that we create is identical to the one given by Zaks [48] (also see Section 1.4), and when $k = 2$, our cycle in the burnt pancake network was previously produced by Suzuki, N. Sawada, and Kaneko [27]; however, both of these papers describe their cycles recursively. The greedy construction of the cycles in the pancake and burnt pancake networks was


(a) The pancake network $\mathbb{G}(4)$.


(c) The 3-sided pancake network $\mathbb{G}_3(2)$.

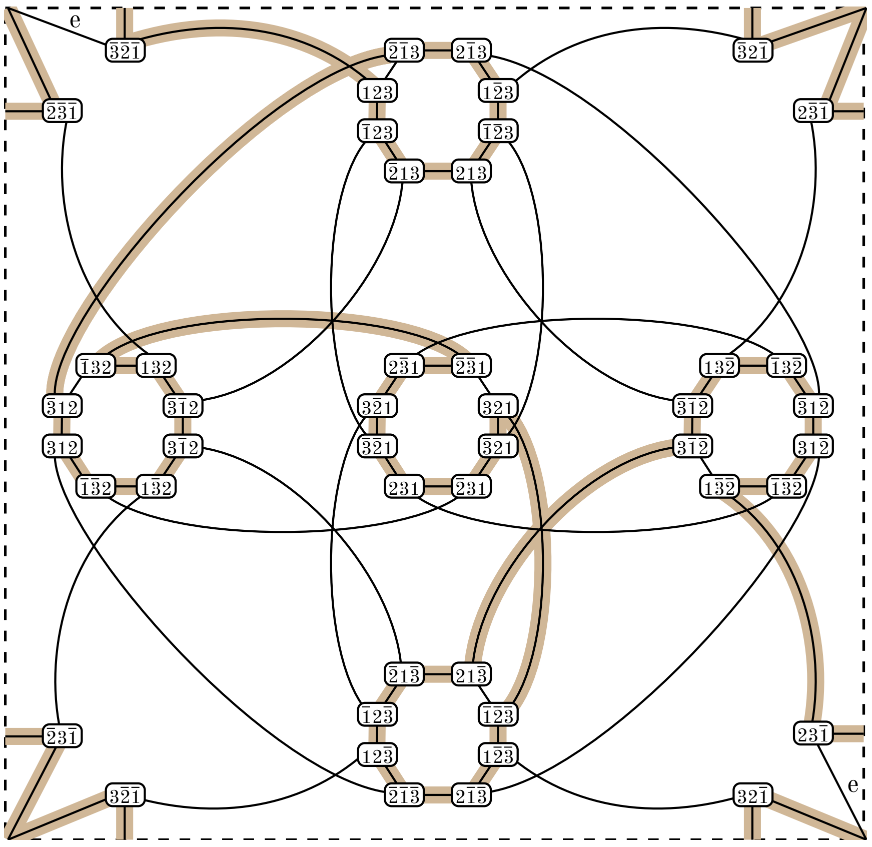

(b) The burnt pancake network $\overline{\mathbb{G}}(3)$. Straight edges are flips of length 1 or 2, and they wrap-around through the exterior, and criss-cross at the corners. For example, the edge labeled e connects $\overline{3}\,\overline{2}\,\overline{1}$ (top-left) to $2\,3\,\overline{1}$ (bottom-right) and is not contained in the Hamilton cycle.

Fig. 1: Hamilton cycles in (a) the pancake network with $n = 4$ pancakes, (b) the burnt pancake network with $n = 3$ burnt pancakes, and (c) the 3-sided pancake network with $n = 2$ pancakes that have $k = 3$ sides (or colours). The highlighted cycles start at $12 \cdots n = 1^0 2^0 \cdots n^0$ and are constructed by the greedy min-flip strategy. The colours $0, 1, 2$ in (b) correspond to black, red, and blue.

previously given by J. Sawada and Williams [47,34]. Interestingly, the latter result also demonstrates that a greedy *max-flip* construction generates a Hamilton cycle in $\mathbb{G}(n)$ and $\overline{\mathbb{G}}(n)$; however, as we note later, this approach does not generalize to $\mathbb{G}_k(n)$ for $k > 2$.

1.3 Combinatorial Generation

Ostensibly, the primary contribution of this paper is the Hamiltonicity of k-sided pancake networks. However, the authors' primary motivation was not in finding a Hamilton cycle, but rather in investigating its contributions to combinatorial generation. *Combinatorial generation* is the research area devoted to the efficient and clever generation of combinatorial objects. By *efficient* we mean that successive objects can be generated in amortized $O(1)$ time or worst-case $O(1)$ time, regardless of their size. The former is known as *constant amortized time (CAT)*, while the latter is known as *loop-free*. By *clever* we mean that non-lexicographic orders are often desirable. When describing these alternate orders, the authors make liberal use of the term *Gray code* — in reference to the eponymous binary reflected Gray code patented by Frank Gray [19]) — and we refer to our Hamilton cycle as a *colour-incrementing prefix-reversal Gray code* for coloured permutations. Informally, it is a *flip Gray code*.

There are numerous algorithms to exhaustively list permutations; comprehensive discussions on this topic date back to Sedgewick's survey in 1977 [36], with more modern coverage in Volume 4 of Knuth's *The Art of Computer Programming* [28]. However, to our knowledge, there are no published Gray codes for coloured permutations. This is surprising as the combinatorial [4,7,11,29,30] and algebraic [2,3,37] properties of coloured permutations have been of considerable interest. We find our new Gray code of interest for two additional reasons.

1. Other greedy approaches for generating $\mathbb{P}(n)$ do not seem to generalize to $\mathbb{P}_k(n)$.
2. Flips are natural and efficient operations in certain contexts.

To expand on the first point, consider the Steinhaus-Johnson-Trotter (SJT) order of permutations, which was independently developed several times the 1960s [39] [25] [41]. In this order, successive permutations differ by an *adjacent-transition* (or *swap*) meaning that adjacent values in the permutations change place. In other words, the order for $\mathbb{P}(n)$ traces a Hamilton path in the permutohedron of order n. For example, SJT order for $n = 4$ appears below

$$\begin{gathered}12\underline{3\mathbf{4}}, 1\underline{2\mathbf{4}}3, \underline{1\mathbf{4}}23, 41\underline{2\mathbf{3}}, \underline{\mathbf{4}1}32, 1\underline{\mathbf{4}3}2, 13\underline{\mathbf{4}2}, \underline{1\mathbf{3}}24, 31\underline{2\mathbf{4}}, 3\underline{1\mathbf{4}}2, \underline{3\mathbf{4}}12, 43\underline{1\mathbf{2}}, \\ \underline{\mathbf{4}3}21, 3\underline{\mathbf{4}2}1, 32\underline{\mathbf{4}1}, \underline{\mathbf{3}2}14, 23\underline{1\mathbf{4}}, 2\underline{3\mathbf{4}}1, \underline{2\mathbf{4}}31, 42\underline{\mathbf{3}1}, \underline{\mathbf{4}2}13, 2\underline{\mathbf{4}1}3, 21\underline{\mathbf{4}3}, \underline{\mathbf{2}1}34.\end{gathered} \tag{3}$$

The symbols that are swapped to create the next permutation are underlined, and the larger value is in bold. The latter demarcation shows the order's underlying greedy priorities: Swap the largest value. For example, consider the fourth permutation in the list, 4123. The largest value 4 cannot be swapped to the left (since it is in the leftmost position) or the right (since 1423 is already in the order), so the next option is to consider 3, and it can only be swapped to the left, which gives the fifth permutation 4132. If this description is perhaps too brief, then we refer the reader to [46].

Now consider greedy generalizations of SJT to signed permutations. The most natural generalization would use *sign-complementing adjacent-transpositions*, which swap and complement the sign of two adjacent values. Unfortunately, any approach using these operations is doomed to fail. This is because the operation does not change the parity of positive and negative values. The authors experimented with other types of signed swaps — complementing the leftmost or rightmost value in the swap, or the larger or small value in the swap — without success.

At this point, it is important to note that one can generate coloured permutations by treating the colours and the permutations separately: For each permutation of $\mathbb{P}(n)$ generated by SJT, one could generate each of the k^n colour vectors to be overlaid onto the permutation, or vice versa. In fact, this approach arises quite naturally from a broad generalization of SJT known as *Algorithm J* [20] and its application to elimination trees [6], whenever k is a power of two.

Surprisingly, our greedy min-flip strategy works for coloured permutations, but the analogous max-flip strategy does not. For example, the max-flip strategy creates the following path in $\mathbb{G}_3(2)$ before getting stuck.

$$1^0 2^0,\ 2^1 1^1,\ 1^2 2^2,\ 2^0 1^0,\ 1^1 2^1,\ 2^2 1^2,\ 2^0 1^2,\ 1^0 2^1,\ 2^2 1^1,\ 1^2 2^0,\ 2^1 1^0,\ 1^1 2^2,\ \lightning$$

The issue is that all the neighbours of the last coloured permutation are already on the path. More specifically, a flip of length one transforms $1^1 2^2$ into $1^2 2^2$, and a flip of length two transforms $1^1 2^2$ into $2^0 1^2$, both of which appear earlier. The failure of the max-flip strategy on coloured permutations is surprising due to the fact that it works for both permutations and signed-permutations [34]. For $n, k \leq 5$ and $k \geq 2$, we observe that the max-flip greedy strategy produces a path of length $\frac{2}{k}(k^n n!)$.

To expand on our second point that flips are natural and efficient operations in certain contexts, note that the time required to flip a prefix is proportional to its length. In particular, if a permutation over $\{1, 2, \ldots, n\}$ is stored in an array or linked list of length n, then it takes $O(m)$ time to flip a prefix of length m[1]. Our min-flip strategy ensures that the shortest possible flips are used. In fact, the average flip length used in our Gray codes is bounded by $e = 2.71828\cdots$ when $k = 1$, and the average is even smaller for $k > 1$.

We also note that flips can be the most efficient operation in certain situations. For example, consider a brute force approach to the undirected travelling salesman problem, wherein every Hamilton path of the n cities is represented by a permutation in $\mathbb{P}(n)$. If we iterate over the permutations using a prefix-reversal Gray code, then successive Hamilton paths differ in a single edge. For example, the edges in 12345678 and 43215678 are identical, except for $(4, 5)$ being replaced by $(1, 5)$. Thus, the cost of each Hamilton cycle can be updated from permutation to permutation using one addition and subtraction. More generally, flip Gray codes are the most efficient choice when the cost (or value) of each permutation depends on its unordered pairs of adjacent symbols. Similarly, our generalization will be the most efficient choice when the cost (or value) of each coloured permutation depends on its unordered pairs of adjacent symbols *and* the minimum distance between their colours.

[1] Some unusual data structures can support flips of any lengths in constant time [45].

1.4 Historical Notes

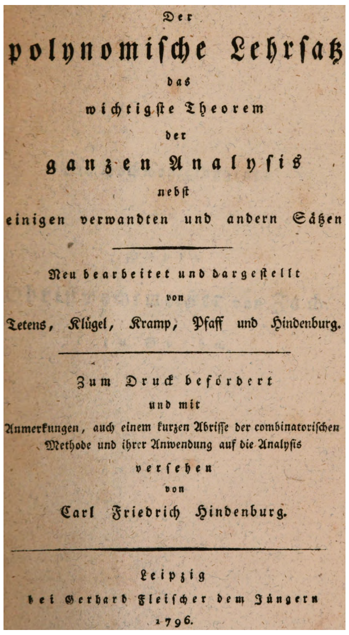
It is relatively well-known that the swap Gray code in (3) dates back to the bell-ringing community in the 17th century. In particular, "single changes" and "plain changes" were discussed in Duckworth's *Tintinnalogia* (1668) [12] and Stedman's *Campanalogia* (1677), and the contributions of these monographs are now being credited in a variety of academic papers (e.g., see White [44], Holroyd [23], Verhoeff [43], and Hunt [24]). Tracking all of the citations to these titles can be difficult, as they are in the public domain, and are reprinted with various attributions and dates (e.g., [13] and [10]). Earlier references to the pattern for $n = 4$ have also been uncovered, as noted by McGuire [31].

Plain change order is a classic example of *Stigler's law of eponymy* [40]: scientific discoveries are not named after thier original discoverers. Similarly, the min-flip order that the authors have referred to as *Zaks's order* in previous publications [5, 16, 34, 35, 47], also has a longer, but lesser-known, history.

In 1967, Ord-Smith referred to the min-flip order as *pseudo-lexicographic order* in Algorithm 308 [32]. He also provided a transposition-based implementation called ECONOPERM, and pointed out that the average number of transpositions between successive permutations tends to $1.175n!$. (Note that prefix-reversals of length two and three can be accomplished by a single transposition.) A slightly modified version was also provided in Ord-Smith's comparison of permutation generation algorithms in 1971 [33]. It's worth noting that both presentations do not explicitly mention prefix-reversals, nor do they contain formal proofs.

Zaks's 1984 paper — *A new algorithm for generation of permutations* [48] — is presented using reversals, although suffixes are modified rather than prefixes. Like Ord-Smith's papers, there is no mention of how the order can be constructed greedily, but it does include a proof of correctness, and a ranking algorithm (see Section 1.5).

Unbeknownst to Ord-Smith and Zaks, and the authors of this paper, the origin of the min-flip order reaches back to the 18th century. Carl Friedrich Hindenburg edited a book entitled *Sammlung Combinatorisch-Analytischer Abhandlungen* (*Collection of Combinatorial-Analytical Treatises*) in 1796 [22]. The second chapter focused on work by Georg Simon Klügel called *Klügels Bemerkungen* (*Klügels Remarks*). Although there are barriers to fully understanding older texts, the chapter includes clear outlines for the $\binom{7}{4} = 35$ combinations with $n = 7$ and $k = 4$ (page 54), and the $p(7) = 15$ integer partitions of $n = 7$ (page 59), both listed in lexicographic order. More strikingly, it outlines the $6! = 720$ permutations of $n = 6$ (page 53) in min-flip order. It's also worth noting that the writing appears to describe these orders in general (i.e., $n, k \geq 1$). Figure 2 shows scans of this public domain work from the Hathi Trust digital library [42]. For the broader context of [22] and the Combinatorial School in Germany, refer to *Hindenburg's Hype* in Section 7.2.1.7 of Knuth's *The Art of Computer Programming* (TAOCP) [28], as well as Eppstein's blog post on Hindenburg's student Heinrich August Rothe [15]. More broadly, Section 7.2.1.7 of TAOCP provides a wonderfully expansive historical survey of combinatorial generation. Knuth also shows that the min-flip order can be understood as a special case of Algorithm G, which is a general-purpose permutation generation algorithm, in Section 7.2.1.2 of TAOCP.



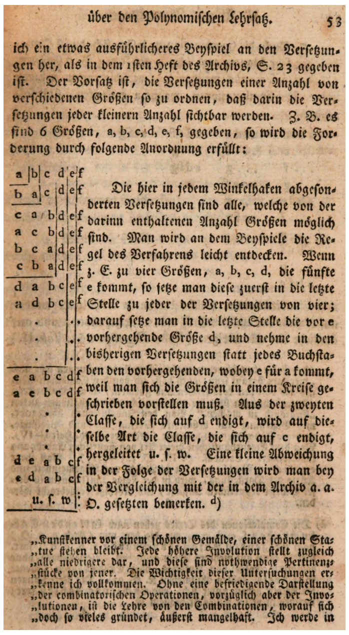
Der
polynomiſche Lehrſatz
das
wichtigſte Theorem
der
ganzen Analyſis
nebſt
einigen verwandten und andern Sätzen

Neu bearbeitet und dargeſtellt
von
Tetens, Klügel, Kramp, Pfaff und Hindenburg.

Zum Druck befördert
und mit
Anmerkungen, auch einem kurzen Abriſſe der combinatoriſchen Methode und ihrer Anwendung auf die Analyſis
verſehen
von
Carl Friedrich Hindenburg.

Leipzig
bei Gerhard Fleiſcher dem Jüngern
1796.

(a) Title page.



über den Polynomiſchen Lehrſatz. 53

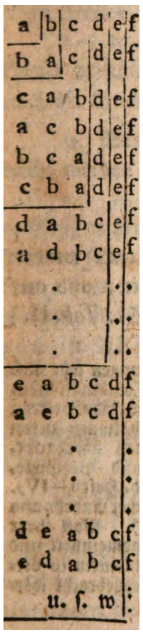
ich ein etwas ausführlicheres Beyſpiel an den Verſetzungen her, als in dem 1ſten Heft des Archivs, S. 23 gegeben iſt. Der Vorſatz iſt, die Verſetzungen einer Anzahl von verſchiedenen Größen ſo zu ordnen, daß darin die Verſetzungen jeder kleinern Anzahl ſichtbar werden. Z. B. es ſind 6 Größen, a, b, c, d, e, f, gegeben, ſo wird die Forderung durch folgende Anordnung erfüllt:

```
a b c d e f
b a c d e f
c a b d e f
a c b d e f
b c a d e f
c b a d e f
d a b c e f
a d b c e f
    .      .
    .      .
    .      .
e a b c d f
a e b c d f
    .      .
    .      .
    .      .
d e a b c f
e d a b c f
 u. ſ. w.  .
```

Die hier in jedem Winkelhaken abgeſonderten Verſetzungen ſind alle, welche von der darinn enthaltenen Anzahl Größen möglich ſind. Man wird an dem Beyſpiele die Regel des Verfahrens leicht entdecken. Wenn z. E. zu vier Größen, a, b, c, d, die fünfte e kommt, ſo ſetze man dieſe zuerſt in die letzte Stelle zu jeder der Verſetzungen von vier; darauf ſetze man in die letzte Stelle die vor e vorhergehende Größe d, und nehme in den bisherigen Verſetzungen ſtatt jedes Buchſtaben den vorhergehenden, wobey e für a kommt, weil man ſich die Größen in einem Kreiſe geſchrieben vorſtellen muß. Aus der zweyten Claſſe, die ſich auf d endigt, wird auf dieſelbe Art die Claſſe, die ſich auf c endigt, hergeleitet u. ſ. w. Eine kleine Abweichung in der Folge der Verſetzungen wird man bey der Vergleichung mit der in dem Archiv a. a. O. geſetzten bemerken. d)

„Kunſtkenner vor einem ſchönen Gemälde, einer ſchönen Sta-
„tue ſtehen bleibt. Jede höhere Involution ſtellt zugleich
„alle niedrigere dar, und dieſe ſind nothwendige Pertinenz-
„ſtücke von jener. Die Wichtigkeit dieſer Unterſuchungen er-
„kenne ich vollkommen. Ohne eine befriedigende Darſtellung
„der combinatoriſchen Operationen, vorzüglich aber der Invo-
„lutionen, iſt die Lehre von den Combinationen, worauf ſich
„doch ſo vieles gründet, äußerſt mangelhaft. Ich werde in

(b) Page 53.



```
a b c d e f
b a c d e f
c a b d e f
a c b d e f
b c a d e f
c b a d e f
d a b c e f
a d b c e f
    .      .
    .      .
    .      .
e a b c d f
a e b c d f
    .      .
    .      .
    .      .
d e a b c f
e d a b c f
 u. ſ. w.  .
```

(c) Min-flip order.

Fig. 2: Hindenburg's *Sammlung Combinatorisch-Analytischer Abhandlungen* (1796) includes Klügel's presentation of the min-flip order of permutations.

1.5 New Results

We present a flip Gray code for $\mathbb{P}_k(n)$ that corresponds to a Hamilton cycle in the k-sided pancake network $\mathbb{G}_k(n)$. This is accompanied by four combinatorial algorithms for traversing the Hamilton cycle, each having unique and interesting properties:

1. A greedy algorithm that is easy to describe, but requires an exponential amount of memory.
2. A recursive algorithm, that reveals the structure of the listing and can be implemented in $O(1)$-amortized time.
3. A simple successor rule approach that allows the cycle to start from any vertex (coloured permutation) and takes on average $O(1)$ time when amortized over the entire listing.
4. A loop-free algorithm to generate the flip-sequence iteratively.

In addition to these generation algorithms, we provide the following results.

– *Ranking and unranking*: $O(n^2)$-time algorithms for determining the position of each coloured permutation in the order, and which coloured permutation appears in each position in the order.
– *Optimality*: Our Hamilton paths and cycles minimize the total number of pancakes flipped.

– *Generalized Hamiltonicity*: We introduce k-sided pancake networks with different 'spin' values (i.e. the amount in which each flipped element's colour is changed) and characterize when these graphs have Hamilton cycles.

1.6 Outline

Before we present our algorithms in Section 3, we first present some notation in Section 2. Our ranking and unranking results are given in Section 4, and our optimality results appear in Section 5. The Hamiltonicity of the generalized k-sided s-spin pancake networks is established in Section 6. We conclude with final remarks and open problems in Section 7.

A preliminary version of this paper was presented at IWOCA 2021 [5]; this extended version adds Sections 4–6 and open problems in Section 7. Our greedy algorithm generalizes the min-flip strategy for pancakes and burnt pancakes in [47, 34] (with non-greedy descriptions from [48] and [27]), while the successor rule and optimality result generalize those for pancakes and burnt pancakes in [16,35].

2 Notation

Let $\pi = p_1p_2\cdots p_n$ be a coloured permutation where each $p_i = v_i^{c_i}$ has value $v_i \in \{1,2,\ldots,n\}$ and colour $c_i \in \{0,1,\ldots,k-1\}$. Recall that $\mathbb{P}_k(n)$ denotes the set of k-coloured permutations of $\{1,2,\ldots,n\}$. Observe that $\mathbb{P}_1(n)$ corresponds to regular permutations and $\mathbb{P}_2(n)$ corresponds to signed permutations. For the remainder of this paper, it is assumed that all permutations are coloured.

As mentioned earlier, a flip of a permutation π, denoted $\mathsf{flip}_i(\pi)$, applies a prefix-reversal of length i on π that also increments the colour of the flipped elements by 1 (modulo k). As an example for $k = 3$:

$$\mathsf{flip}_4(7^0 1^2 6^1 5^0 3^1 4^1 2^1) = \underline{5^1 6^2 1^0 7^1} 3^1 4^1 2^1.$$

A *pre-perm* is any prefix of a permutation in $\mathbb{P}_k(n)$, i.e., $\mathbf{p} = p_1p_2\cdots p_j$ is a pre-perm if there exist $p_{j+1},\ldots p_n$ such that $p_1p_2\cdots p_n$ is a permutation. Note that if $j = n$, then the pre-perm is a permutation. Let $\mathbf{p} = p_1p_2\cdots p_j$ be an arbitrary pre-perm for given a k. For a given element $p_i = v_i^{c_i}$, let $p_i^{+s} = v_i^{(c_i+s) \pmod k}$. For $0 \le i < k$, let $\mathbf{p^{+i}}$ denote $p_1^{+i}p_2^{+i}\cdots p_j^{+i}$, i.e., $\mathbf{p}$ with the colour of each element incremented by i modulo k. Note, $\mathbf{p^{+0}} = \mathbf{p}$. Furthermore, let

$$\rho(\mathbf{p}) = \mathbf{p^{+(k-1)}} \cdot \mathbf{p^{+(k-2)}} \cdots \mathbf{p^{+0}} = r_1r_2\cdots r_m$$

be a circular string of length $m = kj$ where $\cdot$ denotes concatenation. Let $\rho(\mathbf{p})_\mathbf{i}$ denote the length $j-1$ subword ending with r_{i-1}.

Example 1 Consider a pre-perm $\mathbf{p} = 1^0 2^0 3^2$ where $j = 3$ and $k = 4$. Then

$$\rho(\mathbf{p}) = 1^3 2^3 3^1 \cdot 1^2 2^2 3^0 \cdot 1^1 2^1 3^3 \cdot 1^0 2^0 3^2 \quad \text{where} \quad \rho(\mathbf{p})_\mathbf{5} = 3^1 1^2 \quad \text{and} \quad \rho(\mathbf{p})_\mathbf{2} = 3^2 1^3.$$

For any pre-perm $\mathbf{p} = p_1 p_2 \cdots p_j$, let $\overleftarrow{\mathbf{p}}$ denote the reverse of $\mathbf{p}$, i.e., $\overleftarrow{\mathbf{p}} = p_j p_{j-1} \cdots p_2 p_1$. Note that $\overleftarrow{\mathbf{p}}$ is not equivalent to applying a flip of length j to $\mathbf{p}$ when $k > 1$ as the colours of each symbol do not change in $\overleftarrow{\mathbf{p}}$. For the remainder of this paper we will use $\mathbf{p}$ to denote a pre-perm, and when it is clear we will use π to denote a permutation.

3 Constructions of a Cyclic Flip Gray code for $\mathbb{P}_k(n)$

In this section we present four different combinatorial algorithms for generating the same cyclic flip Gray code for $\mathbb{P}_k(n)$. We begin by studying the listing of permutations generated by a greedy min-flip algorithm. We define the *flip-sequence* of a listing of permutations as the sequence of the flip lengths used to generate the listing beginning with the first permutation. By studying the underlying recursive structure of the greedy listing, we provide a recursive description and its corresponding flip-sequence and prove it is equivalent to the flip-sequence generated by the greedy algorithm. This proves that the greedy algorithm generates all permutations in $\mathbb{P}_k(n)$. We then present a successor-rule that determines the successor of a given permutation in the greedy min-flip listing in expected $O(1)$ time. We conclude by showing how the flip-sequence can be generated via a loop-free algorithm.

3.1 Greedy Algorithm

Recall that $\mathsf{GreedyMin}_k(n)$ denotes the greedy algorithm on $\mathbb{P}_k(n)$ that starts at permutation $1^0 2^0 \cdots n^0$ and prioritizes the neighbours of each permutation in the k-sided pancake network by increasing flip length.

Example 2 The following listing (left of the vertical bar) denotes the output of $\mathsf{GreedyMin}_3(3)$ (read top to bottom, then left to right), where black, red and blue correspond to the colours 0,1 and 2 respectively. This listing is exhaustive and cyclic; the last permutation differs from the first permutation by a flip of length $n = 3$. To the right of the vertical line is the flip length required to get from the permutation in that position to its successor.

123	312	231	123	312	231	123	312	231	1 1 1 1 1 1 1 1 1
123	312	231	123	312	231	123	312	231	1 1 1 1 1 1 1 1 1
123	312	231	123	312	231	123	312	231	2 2 2 2 2 2 2 2 2
213	132	321	213	132	321	213	132	321	1 1 1 1 1 1 1 1 1
213	132	321	213	132	321	213	132	321	1 1 1 1 1 1 1 1 1
213	132	321	213	132	321	213	132	321	2 2 2 2 2 2 2 2 2
123	312	231	123	312	231	123	312	231	1 1 1 1 1 1 1 1 1
123	312	231	123	312	231	123	312	231	1 1 1 1 1 1 1 1 1
123	312	231	123	312	231	123	312	231	2 2 2 2 2 2 2 2 2
213	132	321	213	132	321	213	132	321	1 1 1 1 1 1 1 1 1
213	132	321	213	132	321	213	132	321	1 1 1 1 1 1 1 1 1
213	132	321	213	132	321	213	132	321	2 2 2 2 2 2 2 2 2
123	312	231	123	312	231	123	312	231	1 1 1 1 1 1 1 1 1
123	312	231	123	312	231	123	312	231	1 1 1 1 1 1 1 1 1
123	312	231	123	312	231	123	312	231	2 2 2 2 2 2 2 2 2
213	132	321	213	132	321	213	132	321	1 1 1 1 1 1 1 1 1
213	132	321	213	132	321	213	132	321	1 1 1 1 1 1 1 1 1
213	132	321	213	132	321	213	132	321	3 3 3 3 3 3 3 3 *3*

Observe that each column of permutations ends with the same element. Furthermore, the last permutation in each column is a subword of the cyclic word **321321321**.

Unlike the max-flip approach, we will prove that $\mathsf{GreedyMin}_k(n)$ exhaustively generates all permutations in $\mathbb{P}_k(n)$ for all $n, k \geq 1$. We also show that the last permutation in the listing differs by a flip of length n from the first permutation, so the listing is a cyclic flip Gray code. To prove this result, we study the underlying recursive structure of the resulting listings and examine the flip-sequences.

3.2 Recursive Construction

By applying the two observations made following the listing of $\mathsf{GreedyMin}_3(3)$ in Example 2, we arrive at the following recursive definition for a listing of pre-perms, given a pre-perm $\mathbf{p}$ of a permutation in $\mathbb{P}_k(n)$:

$$\mathbf{Rec}_k(\mathbf{p}) = \mathbf{Rec}_k(\rho(\mathbf{p})_\mathbf{m}) \cdot r_m,\ \mathbf{Rec}_k(\rho(\mathbf{p})_\mathbf{m-1}) \cdot r_{m-1}, \ldots,\ \mathbf{Rec}_k(\rho(\mathbf{p})_\mathbf{1}) \cdot r_1, \tag{4}$$

where $\mathbf{Rec}_k(p_x) = p_x^{+0}, p_x^{+1}, p_x^{+2}, \ldots, p_x^{+(k-1)}$ and $\rho(\mathbf{p}) = r_1 \cdots r_m$. Here, the operation $\mathcal{L} \cdot r$ denotes the listing $\mathcal{L}$ with r appended to every element in the listing. We prove that $\mathbf{Rec}_k(1^0 2^0 \cdots n^0)$ generates the same (exhaustive) listing of permutations as $\mathsf{GreedyMin}_k(n)$.

Lemma 1 *Let $\mathbf{p} = p_1 p_2 p_3 \cdots p_j$ be a pre-perm of a permutation in $\mathbb{P}_k(n)$ for some $j \leq n$. Then the first and last pre-perms in $\mathbf{Rec}_k(\mathbf{p})$ are $\mathbf{p}$ and $\overleftarrow{\mathbf{p^{+(k-1)}}}$, respectively.*

Proof The proof proceeds by induction on j. When $j = 1$, we have $\mathbf{p} = \overleftarrow{\mathbf{p}} = p_1$, so $\mathbf{Rec}_k(\mathbf{p}) = \mathbf{p}, \mathbf{p^{+1}}, \mathbf{p^{+2}}, \ldots, \mathbf{p^{+(k-1)}}$. Since $\mathbf{p^{+(k-1)}} = \overleftarrow{\mathbf{p^{+(k-1)}}}$ the claim holds. Now for $1 \leq j < n$ and any pre-perm $\mathbf{p} = p_1 p_2 \cdots p_j$ of a permutation in $\mathbb{P}_k(n)$, suppose that the first and last pre-perms in $\mathbf{Rec}_k(\mathbf{p})$ are $\mathbf{p}$ and $\overleftarrow{\mathbf{p^{+(k-1)}}}$ respectively. Let $\mathbf{p} = p_1 p_2 \cdots p_j p_{j+1}$ be a pre-perm of a permutation in $\mathbb{P}_k(n)$. By definition, the first pre-perm of $\mathbf{Rec}_k(\mathbf{p})$ is the first pre-perm of $\mathbf{Rec}_k(\rho(\mathbf{p})_\mathbf{m}) \cdot r_m$ where $m = (j+1)k$. By definition of $\rho(\mathbf{p})$ and $\rho(\mathbf{p})_\mathbf{m}$, it is clear that $r_m = p_{j+1}$ and $\rho(\mathbf{p})_\mathbf{m} = p_1 p_2 \cdots p_{j-1} p_j$. Applying the inductive hypothesis, the first pre-perm of $\mathbf{Rec}_k(p_1 p_2 \cdots p_{j-1} p_j)$ is $p_1 p_2 \cdots p_{j-1} p_j$. Therefore, the first pre-perm of $\mathbf{Rec}_k(\mathbf{p})$ is $p_1 p_2 \cdots p_{j-1} p_j \cdot p_{j+1} = \mathbf{p}$. Similarly, the last pre-perm of $\mathbf{Rec}_k(\mathbf{p})$ is the last pre-perm of $\mathbf{Rec}_k(\rho(\mathbf{p})_\mathbf{1}) \cdot r_1$. Now, $r_1 = p_1^{+(k-1)}$ and $\rho(\mathbf{p})_\mathbf{1} = p_2 p_3 \cdots p_j p_{j+1}$ and, by the inductive hypothesis, the last pre-perm in $\mathbf{Rec}_k(\rho(\mathbf{p})_\mathbf{1})$ is $p_{j+1}^{+(k-1)} p_j^{+(k-1)} \cdots p_2^{+(k-1)}$. Therefore, the last pre-perm of $\mathbf{Rec}_k(\mathbf{p})$ is $\overleftarrow{\mathbf{p^{+(k-1)}}}$. □

Let τ^j denote j copies of a sequence τ concatenated together. Define the sequence $\sigma_{k,n}$ recursively as

$$\sigma_{k,n} = \begin{cases} 1^{k-1} & \text{if } n = 1 \\ (\sigma_{k,n-1}, n)^{kn-1}, \sigma_{k,n-1} & \text{if } n > 1. \end{cases} \tag{5}$$

We will show that $\sigma_{k,n}$ is the flip-sequence for both $\mathbf{Rec}_k(\mathbf{p})$ and $\mathsf{GreedyMin}_k(n)$. This flip-sequence is a straightforward generalization of the recurrences for non-coloured permutations [48] and signed permutations [34]. Note that $\sigma_{3,3}$ is shown to the right of the vertical bar in Example 2.

Lemma 2 *Let $n \geq 1$, $k \geq 1$, $1 \leq j \leq n$, and $\mathbf{p} = p_1p_2p_3 \cdots p_j$ be a pre-perm of a permutation in $\mathbb{P}_k(n)$. Then the flip-sequence for $\mathbf{Rec}_k(\mathbf{p})$ is $\sigma_{k,j}$.*

Proof If $j = 1$, then $\mathbf{Rec}_k(p_1) = p_1, p_1^{+1}, p_1^{+2}, \ldots, p_1^{+(k-1)}$ and the flip-sequence is $\sigma_{k,1} = 1^{k-1}$. Otherwise assume $n, j \geq 2$ and proceed by induction on j. For all ℓ such that $1 \leq \ell \leq j < n$, suppose that the sequence of flips used to generate $\mathbf{Rec}_k(\mathbf{p})$ is given by $\sigma_{k,\ell}$ for every pre-perm $\mathbf{p}$ of length ℓ. Let $\mathbf{p}' = p_1p_2 \cdots p_{j+1}$ be a pre-perm of a permutation in $\mathbb{P}_k(n)$ and let $\rho(\mathbf{p}') = r_1 \cdots r_m$ where $m = k(j+1)$. Consider $\mathbf{Rec}_k(\mathbf{p}')$. Applying the inductive hypothesis and the definition of $\sigma_{k,j+1}$, it suffices to show that the last pre-perm of $\mathbf{Rec}_k(\rho(\mathbf{p}')_i) \cdot r_i$ and the first pre-perm of $\mathbf{Rec}_k(\rho(\mathbf{p}')_{i-1}) \cdot r_{i-1}$ differ by a flip of length $j+1$ for $i = 2, 3, \ldots, m$. By definition, $\rho(\mathbf{p}')_i = r_{i-j}r_{i-(j-1)} \cdots r_{i-2}r_{i-1}$ where the indices are taken modulo m. Therefore, by Lemma 1, the last pre-perm in $\mathbf{Rec}_k(\rho(\mathbf{p}')_i)$ is $(r_{i-1}r_{i-2} \cdots r_{i-(j-1)}r_{i-j})^{+(\mathbf{k-1})}$. Thus, applying a flip of length $j+1$ to the last pre-prem of $\mathbf{Rec}_k(\rho(\mathbf{p}')_i) \cdot r_i$ yields

$$r_i^{+1}r_{i-j}r_{i-(j-1)} \cdots r_{i-2}r_{i-1}. \tag{6}$$

By Lemma 1, the first pre-perm of $\mathbf{Rec}_k(\rho(\mathbf{p}')_{i-1})$ is $r_{i-(j+1)}r_{i-j} \cdots r_{i-3}r_{i-2}$. By the definition of $\rho(\mathbf{p}')$, it follows that $r_{i-(j+1)} = r_i^{+1}$. Thus, from (6), it follows that the last pre-perm of $\mathbf{Rec}_k(\rho(\mathbf{p}')_i) \cdot r_i$ and the first pre-perm of $\mathbf{Rec}_k(\rho(\mathbf{p}')_{i-1}) \cdot r_{i-1}$ differ by a flip of length $j+1$. By applying the inductive hypothesis, the flip-sequence for $\mathbf{Rec}_k(\mathbf{p}')$ is $(\sigma_{k,j+1}, j+1)^{k(j+1)-1}, \sigma_{k,j}$ which is exactly $\sigma_{k,j+1}$. □

Theorem 1 *For $n \geq 1$, $k \geq 1$, and $\pi \in \mathbb{P}_k(n)$, $\mathbf{Rec}_k(\pi)$ is a cyclic flip Gray code for $\mathbb{P}_k(n)$, where the first and last permutations differ by a flip of length n.*

Proof From Lemma 2, the flip-sequence for $\mathbf{Rec}_k(\pi)$ is given by $\sigma_{k,n}$. Inductively, it is easy to see that the length of the flip-sequence $\sigma_{k,n}$ is $k^n n! - 1$ and that each permutation of $\mathbf{Rec}_k(\pi)$ is unique. Thus, each of the $k^n n!$ permutations is listed exactly once and, from Lemma 1, the first and last permutations of the listing differ by a flip of length n, making $\mathbf{Rec}_k(\pi)$ a cyclic flip Gray code for permutations. □

Lemma 3 *For $n \geq 1$ and $k \geq 1$, the flip-sequence for* $\mathsf{GreedyMin}_k(n)$ *is $\sigma_{k,n}$.*

Proof By contradiction. Suppose the sequence of flips used by $\mathsf{GreedyMin}_k(n)$ differs from $\sigma_{k,n}$ and let j be the smallest value such that the j-th flip used to create $\mathsf{GreedyMin}_k(n)$ differs from the j-th value of $\sigma_{k,n}$. Let these flip lengths be s and t, respectively. Since $\mathsf{GreedyMin}_k(n)$ follows a greedy minimum-flip strategy and because $\sigma_{k,n}$ produces a valid flip Gray code for permutations by Theorem 1 where no permutation is repeated, it must be that $s < t$. Let $\pi = p_1p_2p_3 \cdots p_n$ denote the j-th permutation in the listing $\mathsf{GreedyMin}_k(n)$, i.e., the permutation immediately prior to the j-th flip. Since $\sigma_{k,n}$ is the flip-sequence for $\mathbf{Rec}_k(1^02^0 \cdots n^0)$ by Lemma 2, from the recursive definition it follows inductively that all other permutations with suffix $p_t p_{t+1} \cdots p_n$ appear before π in $\mathbf{Rec}_k(1^0 2^0 \cdots n^0)$, since no permutations are repeated by Theorem 1. Since $\sigma_{k,n}$ and the sequence of flips used by $\mathsf{GreedyMin}_k(n)$ agree until the j-th value, all other permutations with suffix $p_t p_{t+1} \cdots p_n$ appear before π in $\mathsf{GreedyMin}_k(n)$. Therefore, flipping π by a flip of length $s < t$ results in a permutation already visited in $\mathsf{GreedyMin}_k(n)$ before index j contradicting the fact that $\mathsf{GreedyMin}_k(n)$ produces a list of permutations without repetition. □

By definition, $\mathsf{GreedyMin}_k(n)$ starts with the permutation $1^0 2^0 \cdots n^0$ and by Lemma 1, $\mathbf{Rec}_k(1^0 2^0 \cdots n^0)$ also starts with $1^0 2^0 \cdots n^0$. Since they are each created by the same flip-sequence by Lemma 2 and Lemma 3, we get the following corollary.

Corollary 1 *For $n \geq 1$ and $k \geq 1$, the listings $\mathsf{GreedyMin}_k(n)$ and $\mathbf{Rec}_k(1^0 2^0 \cdots n^0)$ are equivalent.*

3.3 Successor Rule

In this section, we will generalize the successor rules found for non-coloured permutations and signed permutations in [35] for $\mathsf{GreedyMin}_k(n)$ for $k > 2$. We say a permutation in $\mathbb{P}_k(n)$ is *increasing* if it corresponds to a length n subword of the circular string $\rho(1^0 2^0 \cdots n^0)$.

Example 3 Recall the definition of ρ in Section 2 and consider $n = 6$, $k = 4$. Since

$$\rho(1^0 2^0 \cdots n^0) = 1^3 2^3 3^3 4^3 5^3 6^3 \cdot 1^2 2^2 3^2 4^2 5^2 6^2 \cdot 1^1 2^1 3^1 4^1 5^1 6^1 \cdot 1^0 2^0 3^0 4^0 5^0 6^0,$$

the following permutations are all increasing:

$$2^3 3^3 4^3 5^3 6^3 1^2, \quad 5^1 6^1 1^0 2^0 3^0 4^0, \quad 1^0 2^0 3^0 4^0 5^0 6^0, \quad 5^0 6^0 1^3 2^3 3^3 4^3.$$

A permutation is *decreasing* if it is a reversal of an increasing permutation. A pre-perm is *increasing* (*decreasing*) if it corresponds to a subsequence of an increasing (decreasing) permutation (when the permutation is thought of as a sequence). For example, $5^1 6^1 2^0 4^0$ is an increasing pre-perm, but $5^1 2^0 4^0 6^0$ and $1^2 2^2 3^1 4^0$ are not. Given a permutation π_2, let $\mathsf{succ}(\pi_2)$ denote the successor of π_2 in $\mathbf{Rec}_k(\pi)$ when the listing is considered to be cyclic.

Lemma 4 *Let $\pi_2 = q_1 q_2 \cdots q_n$ be a permutation in the (cyclic) listing $\mathbf{Rec}_k(\pi)$, where $\pi = p_1 p_2 \cdots p_n$ is increasing. Let $q_1 q_2 \cdots q_j$ be the longest prefix of π_2 that is decreasing. Then $\mathsf{succ}(\pi_2) = \mathsf{flip}_j(\pi_2)$.*

Proof By induction on n. When $n = 1$, the result follows trivially as only flips of length 1 can be applied. Now, for $n > 1$, we focus on the permutations whose successor is the result of a flip of length n and the result will follow inductively by the recursive definition of $\mathbf{Rec}_k(\pi)$. By Lemma 2, the successor of π_2 will be $\mathsf{flip}_n(\pi_2)$ if and only if it is the last permutation in one of the recursive listings of the form $\mathbf{Rec}_k(\rho(\pi)_\mathbf{i}) \cdot r_i$. Recall that r_i is the i-th element in $\rho(\pi)$ when indexed from $r_1 = p_1^{+(k-1)}$ to $r_m = p_n$. As it is clear that at most one permutation is decreasing in each recursive sublist, it suffices to show that the last permutation in each sublist is decreasing to prove the successor rule holds for flips of length n. By Lemma 1, the last permutation in $\mathbf{Rec}_k(\rho(\pi)_\mathbf{i}) \cdot r_i$ is $\overleftarrow{\mathbf{s}} \cdot r_i$ where $\mathbf{s} = \rho(\pi)_\mathbf{i}^{+(\mathbf{k-1})}$. Since π is increasing, it is clear that $\rho(\pi)_\mathbf{i}$ is increasing and therefore that $\mathbf{s}$ is increasing. Hence, $\overleftarrow{\mathbf{s}}$ is decreasing by definition. Furthermore, by the definition of the circular word $\rho(\pi)$, the element immediately before $r_{i-1-(n-1)}^{+(k-1)}$ in $\rho(\pi)$ is r_i (note the subscript $i-1-(n-1)$ is considered modulo nk here). Therefore, $\overleftarrow{\mathbf{s}} \cdot r_i$ is decreasing. Therefore, the successor rule holds for flips of length n and thus for flips of all lengths by induction. □

Example 4 With respect to the listing $\mathbf{Rec}_{10}(1^0 2^0 3^0 4^0 5^0 6^0)$,

$$\mathsf{succ}(3^8 2^8 5^9 4^9 1^7 6^3) = \mathsf{flip}_4(3^8 2^8 5^9 4^9 1^7 6^3) = 4^0 5^0 2^9 3^9 1^7 6^3$$

and

$$\mathsf{succ}(1^8 3^7 2^6 5^5 4^3 6^2) = \mathsf{flip}_1(1^8 3^7 2^6 5^5 4^3 6^2) = 1^9 3^7 2^6 5^5 4^3 6^2.$$

By applying the previous lemma, computing $\mathsf{succ}(\pi_2)$ for a permutation in the listing $\mathbf{Rec}_k(\pi)$ can easily be done in $O(n)$ time as described in the pseudocode given in Algorithm 1.

Algorithm 1 Computing the successor of π in the listing $\mathbf{Rec}_k(1^0 2^0 \cdots n^0)$

```
1: function SUCCESSOR($\pi$)
2:     $incr \leftarrow 0$
3:     for $j \leftarrow 1$ to $n-1$ do
4:         if $v_j < v_{j+1}$ then $incr \leftarrow incr + 1$
5:         if $incr = 2$ or ($incr = 1$ and $v_{j+1} < v_1$) then return flip$_j(\pi)$
6:         if $k > 1$ and $v_j < v_{j+1}$ and ( $(c_{j+1} - c_j + k) \bmod k \neq 1$) then return flip$_j(\pi)$
7:         if $k > 1$ and $v_j > v_{j+1}$ and ( $c_j \neq c_{j+1}$) then return flip$_j(\pi)$
8:     return flip$_n(\pi)$
```

Theorem 2 SUCCESSOR(π) *returns the length of the flip required to obtain the successor of π in the listing $\mathbf{Rec}_k(1^0 2^0 \cdots n^0)$ in $O(n)$ time.*

Though the worst case performance of SUCCESSOR(π) is $O(n)$ time, on average it is much better. Let $\overline{\sigma}_{k,n}$ denote $(\sigma_{k,n}, n)$, i.e., the sequence of flips used to generate the listing $\mathbf{Rec}_k(\pi)$ with an extra flip of length n at the end to return to the starting permutation. Our goal is to determine the average flip length of $\overline{\sigma}_{k,n}$, denoted AVG(k,n). Our analysis generalizes the results for AVG$(1,n)$ [48] and AVG$(2,n)$ [34].

Lemma 5 *For $n \geq 1$ and $k \geq 1$,*

$$\text{AVG}(k,n) = \sum_{j=0}^{n-1} \frac{1}{k^j j!}.$$

Moreover, $\text{AVG}(k,n) < \sqrt[k]{e}$.

Proof By definition of $\sigma_{k,n}$, it is not difficult to see that $\overline{\sigma}_{k,n+1}$ is equivalent to the concatenation of $(n+1)k$ copies of $\overline{\sigma}_{k,n}$ with the last element in every copy of $\overline{\sigma}_{k,n}$ incremented by 1. Therefore, we have

$$\begin{aligned}\text{AVG}(k,n+1) &= \frac{\left(1 + \sum\limits_{f \in \overline{\sigma}_{k,n}} f\right)(n+1)k}{(n+1)!k^{n+1}} \\ &= \frac{\sum\limits_{f \in \overline{\sigma}_{k,n} f}}{n!k^n} + \frac{1}{n!k^n} \\ &= \text{AVG}(k,n) + \frac{1}{n!k^n}.\end{aligned}$$

Hence, with the trivial base case that $\text{AVG}(k,1) = 1$, we have

$$\text{AVG}(k,n) = \sum_{j=0}^{n-1} \frac{1}{k^j j!}.$$

Therefore,

$$\text{AVG}(k,n) < \sum_{j=0}^{\infty} \frac{1}{k^j j!} = \sqrt[k]{e}$$

by applying the well-known Maclaurin series expansion for e^x. □

Observe that the SUCCESSOR function runs in expected $O(1)$ time when the permutation is passed by reference because the average flip length is bounded above by the constant $\sqrt[k]{e}$. Thus, by repeatedly applying the successor rule, we obtain a CAT algorithm for generating $\mathbf{Rec}_k(1^0 2^0 \cdots n^0)$.

3.4 Loop-free Generation of the Flip-Sequence $\sigma_{k,n}$

Based on the recursive definition of the flip-sequence $\sigma_{k,n}$ given in (5), Algorithm 2 will generate $\sigma_{k,n}$ in a loop-free manner. The algorithm generalizes a similar algorithm presented by Zaks for non-coloured permutations [48]. The next flip length x is computed using an array of counters $c_1, c_2, \ldots, c_{n+1}$ initialized to 0, and an array of flip lengths $f_1, f_2, \ldots, f_{n+1}$ with each f_i initialized to i. For a formal proof of correctness, we invite the readers to see the simple inductive proof for the non-coloured case in [48], and note the primary changes required to generalize to coloured permutations are in handling of the minimum allowable flip lengths (when $k = 1$, the smallest allowable flip length is 2) corresponding to lines 5-6 and adding a factor of k to line 8.

Algorithm 2 Loop-free generation of the flip-sequence $\sigma_{k,n}$

```
1: procedure FLIPSEQ
2:     $c_1, c_2, \ldots, c_{n+1} \leftarrow 0, 0, \ldots, 0$
3:     $f_1, f_2, \ldots, f_{n+1} \leftarrow 1, 2, \ldots, n+1$
4:     repeat
5:         if $k = 1$ then $x \leftarrow f_2$;  $f_2 \leftarrow 2$
6:         else  $x \leftarrow f_1$;  $f_1 \leftarrow 1$
7:         $c_x \leftarrow c_x + 1$
8:         if  $c_x = kx-1$  then
9:             $c_x \leftarrow 0$
10:            $f_x \leftarrow f_{x+1}$
11:            $f_{x+1} \leftarrow x + 1$
12:        OUTPUT($x$)
13:    until $x > n$
```

An example illustrating the changing values of the auxiliary values is given in Appendix A.

Theorem 3 *The algorithm* FLIPSEQ *is a loop-free algorithm to generate the flip-sequence $\sigma_{k,n}$ one element at a time.*

Since the average flip length in $\sigma_{k,n}$ is bounded by a constant, as determined in the previous subsection, Algorithm 2 can be modified to generate $\mathbf{Rec}_k(\pi)$ by passing the initial permutation π as a parameter, outputting π at the start of the **repeat** loop, and updating $\pi \leftarrow \mathsf{flip}_x(\pi)$ at the end of the loop instead of outputting the flip length.

Corollary 2 *The algorithm* FLIPSEQ *can be modified to generate successive permutations in the listing $\mathbf{Rec}_k(\pi)$ in $O(1)$-amortized time.*

4 Efficient Ranking and Unranking

In this section, we provide efficient $O(n^2)$-time algorithms for ranking and unranking the listing $\mathbf{Rec}_k(1^0 2^0 \cdots n^0)$.

4.1 Ranking

Let $\mathsf{Rank}(\pi)$ denote the rank of permutation $\pi = p_1 p_2 \cdots p_n = v_1^{c_1} v_2^{c_2} \cdots v_n^{c_n}$ in the listing $\mathbf{Rec}_k(1^0 2^0 \cdots n^0)$. The rank of π can be computed by considering the recursive structure of this listing given in (4), as made precise in the following lemma.

Lemma 6 *For all $n, k \geq 1$,*

$$\mathsf{Rank}(\pi) = \begin{cases} c_i + 1 & \textit{if } n = 1; \\ (n(c_n{+}1){-}v_n) \cdot k^{n-1}(n{-}1)! \ + \ \mathsf{Rank}(q_1 q_2 \cdots q_{n-1}) & \textit{otherwise}, \end{cases}$$

where $q_i = u_i^{d_i}$ (u_i with colour d_i) and $u_i = (v_i{-}v_n) \bmod n$ and $d_i = (c_i{-}c_n) \bmod k$ if $v_i < v_n$ and $d_i = (c_i{-}c_n{-}1) \bmod k$ otherwise.

Proof If $n = 1$ the result clearly follows, so assume $n > 1$. Let $\mathbf{p}' = \mathbf{ab}$ where $\mathbf{a} = (v_{n+1}^{c_n} v_{n+2}^{c_n} \cdots n^{c_n})^{+1}$ and $\mathbf{b} = 1^{c_n} 2^{c_n} \cdots v_{n-1}^{c_n}$. By Lemma 1 and the definitions of $\mathbf{Rec}_k(1^0 2^0 \cdots n^0)$ and $\rho(1^0 2^0 \cdots n^0)$, the first permutation in $\mathbf{Rec}_k(1^0 2^0 \cdots n^0)$ that ends with p_n is $\mathbf{p}' p_n$. From the definition of $\mathbf{Rec}_k(1^0 2^0 \cdots n^0)$, each of the nk recursive sub-lists contain $k^{n-1}(n{-}1)!$ permutations and there are $n(c_n{+}1){-}p_n$ sub-lists that appear before the sub-list containing all permutations ending with p_n. Therefore, $\mathsf{Rank}(\pi)$ is equal to $(n(c_n{+}1){-}v_n) \cdot k^{n-1}(n{-}1)!$ plus the rank of $p_1 p_2 \cdots p_{n-1}$ in $\mathbf{Rec}_k(\mathbf{p}')$. Subtracting v_n modulo n from each element in $\mathbf{p}'$, $c_n{+}1$ modulo k from the colour of every element in $\mathbf{p}' \cap \mathbf{a}$ (i.e., for elements larger than v_n), and c_n modulo k from the colour of every element in $\mathbf{p}' \cap \mathbf{b}$ (i.e., for elements smaller than v_n) yields $1^0 2^0 \cdots (n{-}1)^0$. By Lemma 2, $\mathbf{Rec}_k(1^0 2^0 \cdots (n{-}1)^0)$ and $\mathbf{Rec}_k(\mathbf{p}')$ are generated by the same flip-sequence, so the rank of $q_1 q_2 \cdots q_{n-1}$ in $\mathbf{Rec}_k(1^0 2^0 \cdots (n{-}1)^0)$ is equal to the rank of $p_1 p_2 \cdots p_{n-1}$ in $\mathbf{Rec}_k(\mathbf{p}')$. This completes the proof. □

Example 5 Consider the listing $\mathbf{Rec}_3(\mathbf{123})$ where $n = k = 3$ and the permutation $\pi = \mathbf{132}$. The recursive decomposition for computing $\mathsf{Rank}(\mathbf{132})$ in the ordering $\mathbf{Rec}_3(\mathbf{123})$ is as follows:

$$\begin{aligned} \mathsf{Rank}(\mathbf{132}) &= (3 \cdot 3 - 2) \cdot 3^2 \cdot 2! + \mathsf{Rank}(\mathbf{21}) \\ &= 126 + (2 \cdot 2 - 1) \cdot 3^1 \cdot 1! + \mathsf{Rank}(\mathbf{1}) \\ &= 126 + 9 + 3 = 138 \end{aligned}$$

The computation of $\mathsf{Rank}(p_1 p_2 \cdots p_n)$ requires n recursive calls, where $O(n)$ work is required at each call.

Theorem 4 *The algorithm $\mathsf{Rank}(\pi)$ returns the rank of the permutation π in the listing $\mathbf{Rec}_k(1^0 2^0 \cdots n^0)$ in $O(n^2)$ time.*

4.2 Unranking

To find the permutation $\pi = p_1 p_2 \cdots p_n \in \mathbb{P}_k(n)$ at position $rank$ in the listing $\mathbf{Rec}_k(1^0 2^0 \cdots n^0)$, we essentially inverse the operations applied in the ranking procedure: We recursively determine the last symbol given the current rank, then appropriately relabel a permutation found (recursively) at a specific rank in $\mathbf{Rec}_k(1^0 2^0 \cdots (n{-}1)^0)$. The justification is very similar to that for ranking, so we omit the details. Pseudocode for such an unranking algorithm is as follows.

```
1: function UnRank($rank, t$)
2:     if $t = 1$ then return $1^{rank-1}$
3:     $x \leftarrow \lfloor \frac{rank-1}{k^{t-1}(t-1)!} \rfloor$
4:     $p_t \leftarrow (t - (x \bmod t))^{\lfloor x/t \rfloor}$
5:     $q_1 q_2 \cdots q_{t-1} \leftarrow$ UnRank($rank - xk^{t-1}(t-1)!,\ t-1$)
6:     for $j \leftarrow 1$ to $t-1$ do
7:         $p_j \leftarrow 1 + ((v_j + v_t - 1) \bmod t)$
8:         if $v_j < v_t$ then $c_j \leftarrow (c_j + c_t) \bmod k$
9:         else $c_j \leftarrow (c_j + c_t + 1) \bmod k$
10:    return $p_1 p_2 \cdots p_t$
```

Example 6 Consider the permutation $\pi = p_1p_2p_3$ at $rank = 138$ in the listing $\mathbf{Rec}_3(\mathbf{123})$. Stepping through the function UnRank(138,3), the variable x is assigned $\lfloor 137/(9 \cdot 2!) \rfloor = 7$. Thus p_3 will be the last element in the 8th sublist from the recursive description in (4) which is given by $v_3 = 3 - (7 \bmod 3) = 2$ and $c_3 = \lfloor 7/3 \rfloor = 2$. Thus $p_3 = \mathbf{2}$. Subtracting the $7 \cdot (9 \cdot 2!) = 126$ permutations found in the first 7 sub-lists from the $rank = 138$, we are interested in the 12th permutation in the 8th sub-list ending with $\mathbf{2}$. This permutation is computed by recursively determining the 12th permutation in $\mathbf{Rec}_3(12)$, which is $\mathbf{21}$ and applying an appropriate relabelling to obtain $p_1p_2 = \mathbf{13}$. Thus $\pi = \mathbf{132}$.

Theorem 5 *The algorithm* UnRank$(rank, n)$ *returns the permutation at position $rank$ in the listing $\mathbf{Rec}_k(1^0 2^0 \cdots n^0)$ in $O(n^2)$ time.*

5 Optimality: Minimum Cardinality

In this section, we prove that our Hamilton paths and cycles in $\mathbb{G}_k(n)$ are optimal in the following sense: They flip the fewest total number of pancakes possible. In other words, the local greedy min-flip strategy creates orders that minimize a natural global quantity. Moreover, we will provide the exact total number of flipped pancakes for our Hamiltion paths and cycles. These results generalize previous results for pancakes and burnt pancakes in [35].

Let the *cardinality* of a path or cycle in the k-sided pancake network $\mathbb{G}_k(n)$ be the total number of pancakes that are flipped along it. In other words, the cardinality is the sum of the flip lengths used in the path or cycle. For example, the greedy-min Hamilton cycle in $\mathbb{G}_3(2)$ has cardinality $(1+1+2) \cdot 6 = 24$ (see Figure 1). We start by proving that our Hamilton cycle has minimum possible cardinality.

Theorem 6 *The Hamilton cycle given by* $\mathsf{GreedyMin}_k(n)$ *has cardinality*

$$k^n n! \cdot \sum_{j=0}^{n-1} \frac{1}{k^j j!}, \tag{7}$$

and this is the minimum total number of flips of any Hamilton cycle in the k-sided pancake network $\mathbb{G}_k(n)$.

Proof To prove that (7) is correct, recall that the number of vertices in the k-sided pancake network $\mathbb{G}_k(n)$ is the number of permutations: $|\mathbb{P}_k(n)| = k^n n!$. Therefore, (7) is simply the number of vertices multiplied by the average flip length per edge in the Hamilton cycle (see Lemma 5).

Now we prove that (7) is the minimum possible number of total flipped pancakes. For the sake of contradiction, suppose that another Hamilton cycle of $\mathbb{G}_k(n)$ has a smaller cardinality, and that its flip sequence is $\overline{\sigma}'$. Since $\overline{\sigma}_{k,n}$ and $\overline{\sigma}'$ have the same length, and $\overline{\sigma}'$ has a smaller sum, it must be that $\overline{\sigma}'$ uses more "smaller" flips. More formally, there exists a value m in the range $1 \leq m < n$, such that $\overline{\sigma}'$ contains more entries of value at most m than $\overline{\sigma}_{k,n}$ does. To see why this is not possible, note that $\overline{\sigma}_{k,n}$ consists of $t = \frac{k^{n-m} n!}{m!}$ copies of $\sigma_{k,m}$ that are separated by individual values that are larger than m. That is, (5) implies that

$$\overline{\sigma}_{k,n} = \sigma_{k,m}, v_1, \sigma_{k,m}, v_2, \ldots, \sigma_{k,m}, v_t \tag{8}$$

where each v_i is a single value satisfying $v_i > m$. Also note that every value in $\sigma_{k,m}$ is at most m, and the length of each $\sigma_{k,m}$ is $|\sigma_{k,m}| = k^m m! - 1$.

Now consider the structure of $\overline{\sigma}_{k,n}$ in (8). Notice that it is maximal in the following sense: It contains the maximum number of values that are at most m without including any subsequence[2] of length greater than $|\sigma_{k,m}|$. In other words, it packs in as many values that are at most m as possible without exceeding the run-length of each $\sigma_{k,m}$. Since σ' has more entries of value at most m than $\overline{\sigma}_{k,n}$ does, it must be that $\overline{\sigma}'$ contains a longer such subsequence. That is, $\overline{\sigma}'$ has a subsequence of $|\sigma_{k,m}| + 1 = k^m m!$ flips of length at most m. But such a subsequence must create a repeated permutation in $\mathbb{P}_k(n)$. This is because the flips only change the first m symbols of each permutation, and a sequence of $k^m m!$ flips will create $k^m m! + 1$ permutations, which exceeds $|\mathbb{P}_k(m)| = k^m m!$. This contradicts that $\overline{\sigma}'$ is a flip sequence for a Hamilton cycle of $\mathbb{G}_k(n)$, since a strict subsequence of it must revisit a vertex. □

We complete this section by noting that our Hamilton path also has minimum cardinality.

Corollary 3 *The Hamilton path given by* $\mathsf{GreedyMin}_k(n)$ *has cardinality*

$$\left(k^n n! \cdot \sum_{j=0}^{n-1} \frac{1}{k^j j!} \right) - n, \tag{9}$$

and this is the minimum total number of flips of any Hamilton path in the k-sided pancake network $\mathbb{G}_k(n)$.

Proof This follow from Theorem 6 and the fact that the last entry in $\overline{\sigma}_{k,n}$ is n. □

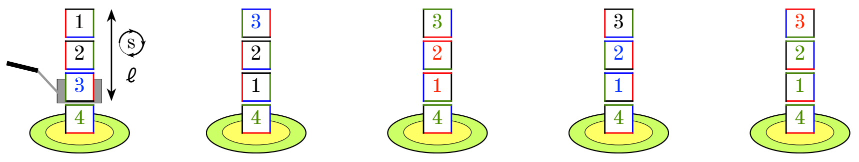
[2] Here a *subsequence* refers to consecutive (i.e., neighbouring) values in the sequence.

6 Hamiltonicity of k-Sided s-Spin Pancake Networks

In this section, we consider generalizations of k-sided pancake networks with a fixed "spin" value, and then characterize when they are Hamiltonian. As we will see, the key to the characterization is the Hamiltonicity of regular k-sided pancake networks.

6.1 s-Spin Prefix-Reversals

So far, this paper has focused on flips that are colour-incrementing prefix-reversals. In other words, the colour (or side) of every flipped element is increased by one modulo k. More generally, one could consider flips that increase the colour of every flipped element by some fixed value s modulo k. To better fit Goodman's initial formulation [14], we refer to the fixed increment as the *spin* of the flip, and we define the operation as an *s-spin prefix-reversal*. To avoid needlessly overexuberant waiters, we limit the spin to the range $0 \leq i < k$. For example, Fig. 3 illustrates four different types of flips on a stack of square[3] pancakes.



(a) Initial stack $1^0 2^0 3^2 4^3 = 1234$. (b) Spin $s = 0$: $3^2 2^0 1^0 4^3 = 3214$. (c) Spin $s = 1$: $3^3 2^1 1^1 4^3 = 3214$. (d) Spin $s = 2$: $3^0 2^2 1^2 4^3 = 3214$. (e) Spin $s = 3$: $3^1 2^3 1^3 4^3 = 3214$.

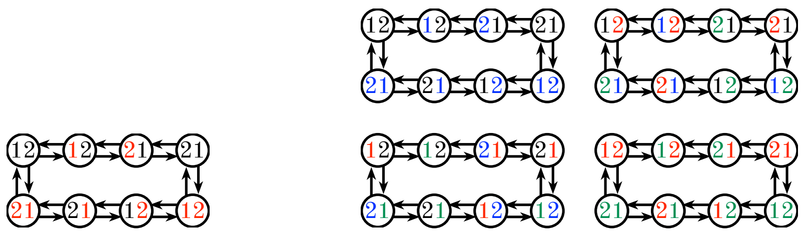
Fig. 3: (a) A stack of four 4-sided pancakes ("toutons") on a plate, and (b)–(e) the result of a flip of length $\ell = 3$ and each spin value $s = 0, 1, 2, 3$. The top of each pancake is its colour, with 0, 1, 2, 3 for **black**, **red**, **blue**, **green**.

6.2 k-Sided s-Spin Pancake Networks

The *k-sided s-spin pancake network* $\mathbb{G}_{k,s}(n)$ is a directed graph with vertex set $\mathbb{P}_k(n)$, and a directed edge from $\pi_1 \in \mathbb{P}_k(n)$ to $\pi_2 \in \mathbb{P}_k(n)$ if π_1 can be transformed into π_2 by an s-spin prefix-reversal. The original k-sided pancake networks are obtained with spin $s = 1$. That is, $\mathbb{G}_k(n) = \mathbb{G}_{k,1}(n)$. Figs. 4b and 5a illustrate networks with spin $s = 2$.

Observe that $\mathbb{G}_{4,2}(2)$ in Fig. 4b is disconnected, while $\mathbb{G}_{3,2}(2)$ in Fig. 5a is reminiscent of $\mathbb{G}_3(2)$ from Fig. 1c. Theorem 7 will explain these observations more generally. Before stating the theorem, recall that the *period* of $x \in \mathbb{Z}_k$ (i.e. an element

[3] A *touton* (or *tiffin*) is a piece of fried or baked bread dough that is a traditional dish from Newfoundland and Labrador. It resembles a square pancake, and is often served with butter, jam, molasses, or maple syrup. These serve as close approximations of real-world square pancakes.


(a) In $\mathbb{G}_2(2)$ each edge increments the colour of the flipped elements modulo 2.

(b) In $\mathbb{G}_{4,2}(2)$ each edge adds 2 to the colour of the flipped elements modulo 4.

Fig. 4: (a) The 2-sided pancake network $\mathbb{G}_2(2)$, and (b) the 4-sided 2-spin pancake network $\mathbb{G}_{4,2}(2)$. The latter consists of $c = \left(\frac{k}{t}\right)^n = \left(\frac{4}{2}\right)^2 = 4$ components that are each isomorphic to the former. The colours are 0, 1, 2, 3 for **black**, **red**, **blue**, **green**.

in the cyclic group of order k) is the smallest integer i such that $i \cdot x \equiv 0 \bmod k$. In other words, it is the number of times x must be added to itself before the sum is equal to $0 \bmod k$.

Theorem 7 *The k-sided s-spin pancake network $\mathbb{G}_{k,s}(n)$ consists of c connected components, each of which is isomorphic to the t-sided pancake network $\mathbb{G}_t(n)$, where t is the period of $s \in \mathbb{Z}_k$, and $c = \left(\frac{k}{t}\right)^n$.*

Before proving the theorem, it is helpful to consider the special case where $\mathbb{G}_{k,s}(n)$ is connected. The following corollary provides this simplification of Theorem 7, and its result is illustrated by Fig. 5a.

Corollary 4 *The k-sided s-spin pancake network $\mathbb{G}_{k,s}(n)$ is isomorphic to the k-sided pancake network $\mathbb{G}_k(n)$ whenever s and k are coprime.*

Proof When k and s are coprime, the period of $s \in \mathbb{Z}_k$ is $t = k$. Therefore, by Theorem 7, the number of components in $\mathbb{G}_{k,s}(n)$ is $c = \left(\frac{k}{t}^n\right) = 1^n = 1$. Furthermore, this component is isomorphic to $\mathbb{G}_t(n) = \mathbb{G}_k(n)$ as claimed. $\square$

Now we prove Theorem 7.

Proof (Theorem 7) Consider the vertex $\pi_1 = 1^0 2^0 \cdots n^0 \in \mathbb{P}_k(n)$ in $\mathbb{G}_{k,s}(n)$. Each edge of $\mathbb{G}_{k,s}(n)$ either increases the colour of an element by s, or does not change the colour of the element. Therefore, if there is a directed path from π_1 to $\pi_2 \in \mathbb{P}_k(n)$, then the colour of an element in π_2 must satisfy $i \cdot s \bmod k$ for some i. The number of possible element colours of the form $i \cdot s \bmod k$ is simply the period of $s \in \mathbb{Z}_k$, which we denote by t. Therefore, an upperbound on the size of the strongly connected component containing π_1 is $n! \cdot t^n$. Note that this quantity is precisely the order of $\mathbb{G}_t(n)$. Furthermore, it is easy to see that this (strongly) connected component is isomorphic to $\mathbb{G}_t(n)$. To formally establish the isomorphism, let $R \subseteq \mathbb{P}_k(n)$ be the set of vertices that are reachable from π_1. A suitable mapping from the vertices in $\mathbb{G}_{k,t}(n)$ is $f : \mathbb{P}_t(n) \to R$ is

$$f(p_1^{c_1} p_2^{c_2} \cdots p_n^{c_n}) = p_1^{s \cdot c_1} p_2^{s \cdot c_2} \cdots p_n^{s \cdot c_n} \tag{10}$$

where the colours are taken modulo k.

To complete the proof, note that $\mathbb{G}_{k,s}(n)$ is vertex-transitive, so every connected component must be isomorphic to $\mathbb{G}_t(n)$. By considering the relative orders of these graphs, the number of components in $\mathbb{G}_{t,s}(n)$ is

$$\frac{n! \cdot k^n}{n! \cdot t^n} = \left(\frac{k}{t}\right)^n = c$$

as claimed. □

As an example of Corollay 4, $k = 3$ and $s = 2$ are co-prime, so the 3-sided 2-spin pancake network $\mathbb{G}_{3,2}(2)$ is isomorphic to the 3-sided pancake network $\mathbb{G}_3(2)$. The graph $\mathbb{G}_3(2)$ previously appeared in Fig. 1c, and we provide two embeddings of $\mathbb{G}_{3,2}(2)$ in Fig. 5. The first is obtained from Fig. 1c by applying the mapping given by (10) to each vertex in-place. For example, the top vertex is $f(1^1 2^0) = 1^{2\cdot 1} 2^{2\cdot 0} = 1^2 2^0$, or $f(12) = 12$. The second is obtained from Fig. 1c by reversing every edge. The second approach is valid in this case due to the fact that 1-spin and 2-spin edges are inverses of each other when the pancakes have $k = 3$ sides.

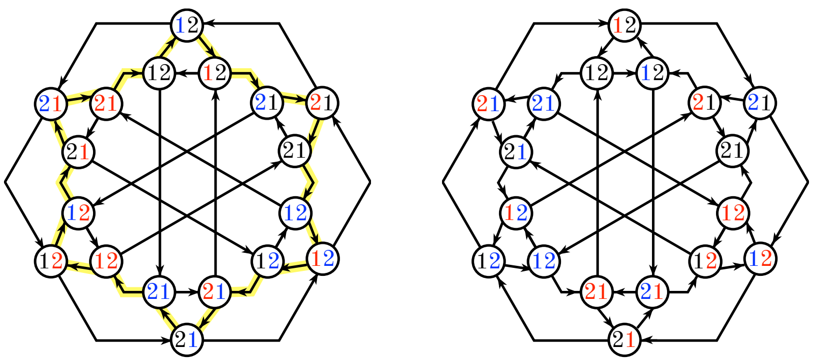

(a) One embedding of $\mathbb{G}_{3,2}(2)$.   (b) A second embedding of $\mathbb{G}_{3,2}(2)$.

Fig. 5: Two embeddings of $\mathbb{G}_{3,2}(2)$: the k-sided s-spin pancake network on n pancakes with $k = 3$, $s = 2$, and $n = 2$. The network is isomorphic to $\mathbb{G}_3(2)$ from Fig. 1c by Corollary 4 (since $k = 3$ and $i = 2$ are co-prime). The embeddings are obtained from Fig. 1c by (a) the mapping provided in the proof of Theorem 7, and (b) reversing the edges. The greedy min-flip strategy $\mathsf{GreedyMin}_{3,2}(2)$ creates a Hamilton cycle as seen in (a).

6.3 Generalized Hamiltonicity Result

Theorem 7 and Corollary 4 lead to natural generalizations of our results. In particular, the *greedy min-flip construction*, denoted $\mathsf{GreedyMin}_{k,i}(n)$, starts a path at vertex

$1^0 2^0 \cdots n^0 \in \mathbb{P}_k(n)$ in $\mathbb{G}_{k,s}(n)$, and then repeatedly extends the path to a new vertex along the edge that corresponds to the shortest s-spin prefix-reversal.

Corollary 5 *The greedy min-flip strategy* $\mathsf{GreedyMin}_k(n)$ *creates a Hamilton cycle in the k-sided s-spin pancake network* $\mathbb{G}_{k,s}(n)$ *when k and s are coprime.*

Corollary 5 is illustrated in Fig. 5a.

7 Final Remarks

We presented four different combinatorial algorithms for traversing a specific Hamilton cycle in the k-sided pancake network:

- a min-flip greedy algorithm which requires exponential space to store the network,
- a recursive construction that traverses the network in $O(1)$-amortized time using $O(n)$-space,
- a successor-rule that allows the cycle to be traversed starting from any initial permutation (which the previous approaches do not allow for) in $O(1)$-amortized time per permutation, and
- a loop-free algorithm is given for the associated flip-sequence.

The Hamilton cycle corresponds to a flip Gray code listing of coloured permutations. Based on the recursive description, simple $O(n^2)$-time ranking and unranking algorithms have also been presented for a corresponding listing of coloured permutations. A complete C implementation of our algorithms is available on The Combinatorial Object Server [9] at `http://combos.org/cperm`.

We also proved that our Hamilton paths and cycles are optimal in the sense that they flip the fewest total number of (coloured) pancakes possible. In the process, we obtain an exact number for the total number of pancakes flipped by traversing these paths/cycles, generalizing known results for pancakes and burnt pancakes. Finally, we generalized the notion of a flip by considering a "spin value" and characterize when the corresponding pancake networks are Hamiltonian.

An interesting line of future research is to consider prefix-reversals on combinations and more generally, the permutations of a multiset.

Open Problem #1: Find a simple and efficient algorithm to list combinations where successive elements differ by a prefix-reversal.

Open Problem #2: Find a simple and efficient algorithm to list the permutations of a multiset where successive elements differ by a prefix reversal.

The first problem essentially considers iterating through all stacks of n pancakes where each pancake is one of two sizes, say 0 or 1. The second problem generalizes the first problem by allowing more than two sizes of pancakes, or an alphabet or arbitrary size.

Acknowledgements. We'd like to thank Torsten Mütze for helpful discussions on the history of the min-flip algorithm for permutations, and its connection to Algorithm J.

References

1. S. Akers and B. Krishnamurthy. A group-theoretic model for symmetric interconnection networks. *Computers, IEEE Transactions on*, 38(4):555–566, 1989.
2. C. A. Athanasiadis. Binomial Eulerian polynomials for colored permutations. *J. Combin. Theory Ser. A*, 173:105214, 2020.
3. E. Bagno, D. Garber, and T. Mansour. On the group of alternating colored permutations. *Electron. J. Combin.*, 21(2):Paper 2.29, 2014.
4. A. Borodin. Longest increasing subsequences of random colored permutations. *Electron. J. Combin.*, 6(13):12pp, 1999.
5. B. Cameron, J. Sawada, and A. Williams. A Hamilton cycle in the k-sided pancake network. In P. Flocchini and L. Moura, editors, *IWOCA '21: 32nd International Workshop on Combinatorial Algorithms*, volume 12757 of *Lecture Notes in Computer Science*, pages 137—-151, Ottawa, ON, Canada, 2021. Springer.
6. J. Cardinal, A. Merino, and T. Mütze. Combinatorial generation via permutation languages. IV. Elimination trees, 2021.
7. W. Y. C. Chen, H. Y. Gao, and J. He. Labeled partitions with colored permutations. *Discrete Math.*, 309(21):6235–6244, 2009.
8. D. S. Cohen and M. Blum. On the problem of sorting burnt pancakes. *Discrete Appl. Math.*, 61(2):105–120, July 1995.
9. COS++. The Combinatorial Object Server. `http://combos.org/cperm`.
10. J. Doleman et al. *Campanalogia Improved: Or, the Art of Ringing Made Easy,...* C. Hitch and L. Hawes; and J. Hodges, 1733.
11. A. Duane and J. Remmel. Minimal overlapping patterns in colored permutations. *Electron. J. Combin.*, 18(2):Paper 25, 38, 2011.
12. R. Duckworth and F. Stedman. *Tintinnalogia: Or, The Art of Ringing*. London, 1668.
13. R. Duckworth and F. Stedman. *Tintinnalogia: Or, The Art of Ringing*. Kingsmead Reprints, 1970.
14. H. Dweighter. Problem E2569. *American Mathematical Monthly*, 82:1010, 1975.
15. D. Eppstein. An almost-forgotten combinatorist: Heinrich August Rothe. `https://11011110.github.io/blog/2012/03/27/almost-forgotten-combinatorist-heinrich.html`, 2012.
16. H. Essed and W. Therese. The harassed waitress problem. In A. Ferro, F. Luccio, and P. Widmayer, editors, *Fun with Algorithms*, volume 8496 of *Lecture Notes in Computer Science*, pages 325–339. Springer International Publishing, 2014.
17. G. Fertin, A. Labarre, I. Rusu, E. Tannier, and S. Vialette. *Combinatorics of Genome Rearrangements*. MIT Press, August 2009.
18. W. H. Gates and C. H. Papadimitriou. Bounds for sorting by prefix reversal. *Discrete Math.*, 27(1):47–57, 1979.
19. F. Gray. Pulse code communication. *U.S. Patent 2,632,058*, 1947.
20. E. Hartung, H. P. Hoang, T. Mütze, and A. Williams. Combinatorial generation via permutation languages. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1214–1225. SIAM, 2020.
21. M. H. Heydari and I. H. Sudborough. On the diameter of the pancake network. *J. Algorithms*, 25(1):67–94, 1997.
22. C. F. Hindenburg. *Sammlung combinatorisch-analytischer Abhandlungen*, volume 1. ben Gerhard Fleischer dem Jungern, 1796.
23. A. E. Holroyd. Perfect snake-in-the-box codes for rank modulation. *IEEE Transactions on Information Theory*, 63(1):104–110, 2016.
24. K. Hunt. The art of changes: bell-ringing, anagrams, and the culture of combination in seventeenth-century england. *Journal of Medieval and Early Modern Studies*, 48(2):387–412, 2018.
25. S. M. Johnson. Generation of permutations by adjacent transposition. *Mathematics of computation*, 17(83):282–285, 1963.
26. M. P. Justan, F. P. Muga, and I. H. Sudborough. On the generalization of the pancake network. In *Proceedings International Symposium on Parallel Architectures, Algorithms and Networks. I-SPAN'02*, pages 173–178, 2002.
27. K. Kaneko. Hamiltonian cycles and Hamiltonian paths in faulty burnt pancake graphs. *IEICE - Trans. Inf. Syst.*, E90-D(4):716–721, Mar. 2007.

28. D. E. Knuth. *The Art of Computer Programming*, volume 4: Combinatorial Algorithms, Part 1. Addison-Wesley, 2010.
29. T. Mansour. Pattern avoidance in coloured permutations. *Sém. Lothar. Combin.*, 46:Art. B46g, 12, 2001/02.
30. T. Mansour. Coloured permutations containing and avoiding certain patterns. *Ann. Comb.*, 7(3):349–355, 2003.
31. G. McGuire. Bells, motels and permutation groups. *arXiv preprint arXiv:1203.1835*, 2012.
32. R. Ord-Smith. Generation of permutations in pseudolexicographic order. *Communications of the ACM*, 10(7):452, 1967.
33. R. Ord-Smith. Generation of permutation sequences: Part 2. *The Computer Journal*, 14(2):136–139, 1971.
34. J. Sawada and A. Williams. Greedy flipping of pancakes and burnt pancakes. *Discrete Appl. Math.*, 210:61–74, 2016.
35. J. Sawada and A. Williams. Successor rules for flipping pancakes and burnt pancakes. *Theoret. Comput. Sci.*, 609(part 1):60–75, 2016.
36. R. Sedgewick. Permutations generation methods. *ACM Comput. Surv.*, 9(2):137–164, 1977.
37. H. Shin and J. Zeng. Symmetric unimodal expansions of excedances in colored permutations. *European J. Combin.*, 52(part A):174–196, 2016.
38. S. Singh. Flipping pancakes with mathematics. *The Guardian*, 2013.
39. H. Steinhaus. *One hundred problems in elementary mathematics*. Basic Books, 1964.
40. S. M. Stigler. Stigler's law of eponymy. *Transactions of the New York academy of sciences*, 39(1 Series II):147–157, 1980.
41. H. F. Trotter. Algorithm 115: perm. *Communications of the ACM*, 5(8):434–435, 1962.
42. H. Trust. Record 000208301. `https://catalog.hathitrust.org/Record/000208301`.
43. T. Verhoeff. The spurs of dh lehmer. *Designs, Codes and Cryptography*, 84(1):295–310, 2017.
44. A. T. White. Fabian stedman: The first group theorist? *The American mathematical monthly*, 103(9):771–778, 1996.
45. A. Williams. O(1)-time unsorting by prefix-reversals in a boustrophedon linked list. In *5th International Conference on FUN with Algorithms*, volume 6099 of *Lecture Notes in Computer Science*, pages 368–379. Springer, 2010.
46. A. Williams. The greedy Gray code algorithm. In *Algorithms and Data Structures Symposium, WADS 2013*, volume LNCS 8037, pages 525–536, 2013.
47. A. Williams and J. Sawada. Greedy pancake flipping. In *The VII Latin-American Algorithms, Graphs and Optimization Symposium (LAGOS 2013)*, volume 45, pages 357–362, 2013.
48. S. Zaks. A new algorithm for generation of permutations. *BIT*, 24(2):196–204, 1984.

A An example illustrating the auxiliary variable for the loop-free generation

The following array (read top to bottom, then left to right) shows how the c_i's and f_i's evolve for each flip generated by FLIPSEQ for $\sigma_{3,3}$. Each entry corresponds to the flip in the same position in the array to the right of the vertical bar in Example 2. Each entry is a 3×3 matrix of the form $\begin{pmatrix} c_1 & c_2 & c_3 \\ f_1 & f_2 & f_3 \end{pmatrix}$. Note that $c_4 = 0$ except for in the last entry where $c_4 = 1$ and f_4 is always equal to 4, so we chose to omit these.

$$\begin{pmatrix}1&0&0\\1&2&3\end{pmatrix} \begin{pmatrix}1&0&1\\1&2&3\end{pmatrix} \begin{pmatrix}1&0&2\\1&2&3\end{pmatrix} \begin{pmatrix}1&0&3\\1&2&3\end{pmatrix} \begin{pmatrix}1&0&4\\1&2&3\end{pmatrix} \begin{pmatrix}1&0&5\\1&2&3\end{pmatrix} \begin{pmatrix}1&0&6\\1&2&3\end{pmatrix} \begin{pmatrix}1&0&7\\1&2&3\end{pmatrix} \begin{pmatrix}1&0&0\\1&2&4\end{pmatrix}$$

$$\begin{pmatrix}0&0&0\\2&2&3\end{pmatrix} \begin{pmatrix}0&0&1\\2&2&3\end{pmatrix} \begin{pmatrix}0&0&2\\2&2&3\end{pmatrix} \begin{pmatrix}0&0&3\\2&2&3\end{pmatrix} \begin{pmatrix}0&0&4\\2&2&3\end{pmatrix} \begin{pmatrix}0&0&5\\2&2&3\end{pmatrix} \begin{pmatrix}0&0&6\\2&2&3\end{pmatrix} \begin{pmatrix}0&0&7\\2&2&3\end{pmatrix} \begin{pmatrix}0&0&0\\2&2&4\end{pmatrix}$$

$$\begin{pmatrix}0&1&0\\1&2&3\end{pmatrix} \begin{pmatrix}0&1&1\\1&2&3\end{pmatrix} \begin{pmatrix}0&1&2\\1&2&3\end{pmatrix} \begin{pmatrix}0&1&3\\1&2&3\end{pmatrix} \begin{pmatrix}0&1&4\\1&2&3\end{pmatrix} \begin{pmatrix}0&1&5\\1&2&3\end{pmatrix} \begin{pmatrix}0&1&6\\1&2&3\end{pmatrix} \begin{pmatrix}0&1&7\\1&2&3\end{pmatrix} \begin{pmatrix}0&1&0\\1&2&4\end{pmatrix}$$

$$\begin{pmatrix}1&1&0\\1&2&3\end{pmatrix} \begin{pmatrix}1&1&1\\1&2&3\end{pmatrix} \begin{pmatrix}1&1&2\\1&2&3\end{pmatrix} \begin{pmatrix}1&1&3\\1&2&3\end{pmatrix} \begin{pmatrix}1&1&4\\1&2&3\end{pmatrix} \begin{pmatrix}1&1&5\\1&2&3\end{pmatrix} \begin{pmatrix}1&1&6\\1&2&3\end{pmatrix} \begin{pmatrix}1&1&7\\1&2&3\end{pmatrix} \begin{pmatrix}1&1&0\\1&2&4\end{pmatrix}$$

$$\begin{pmatrix}0&1&0\\2&2&3\end{pmatrix} \begin{pmatrix}0&1&1\\2&2&3\end{pmatrix} \begin{pmatrix}0&1&2\\2&2&3\end{pmatrix} \begin{pmatrix}0&1&3\\2&2&3\end{pmatrix} \begin{pmatrix}0&1&4\\2&2&3\end{pmatrix} \begin{pmatrix}0&1&5\\2&2&3\end{pmatrix} \begin{pmatrix}0&1&6\\2&2&3\end{pmatrix} \begin{pmatrix}0&1&7\\2&2&3\end{pmatrix} \begin{pmatrix}0&1&0\\2&2&4\end{pmatrix}$$

$$\begin{pmatrix}0&2&0\\1&2&3\end{pmatrix} \begin{pmatrix}0&2&1\\1&2&3\end{pmatrix} \begin{pmatrix}0&2&2\\1&2&3\end{pmatrix} \begin{pmatrix}0&2&3\\1&2&3\end{pmatrix} \begin{pmatrix}0&2&4\\1&2&3\end{pmatrix} \begin{pmatrix}0&2&5\\1&2&3\end{pmatrix} \begin{pmatrix}0&2&6\\1&2&3\end{pmatrix} \begin{pmatrix}0&2&7\\1&2&3\end{pmatrix} \begin{pmatrix}0&2&0\\1&2&4\end{pmatrix}$$

$$\begin{pmatrix}1&2&0\\1&2&3\end{pmatrix} \begin{pmatrix}1&2&1\\1&2&3\end{pmatrix} \begin{pmatrix}1&2&2\\1&2&3\end{pmatrix} \begin{pmatrix}1&2&3\\1&2&3\end{pmatrix} \begin{pmatrix}1&2&4\\1&2&3\end{pmatrix} \begin{pmatrix}1&2&5\\1&2&3\end{pmatrix} \begin{pmatrix}1&2&6\\1&2&3\end{pmatrix} \begin{pmatrix}1&2&7\\1&2&3\end{pmatrix} \begin{pmatrix}1&2&0\\1&2&4\end{pmatrix}$$

$$\begin{pmatrix}0&2&0\\2&2&3\end{pmatrix} \begin{pmatrix}0&2&1\\2&2&3\end{pmatrix} \begin{pmatrix}0&2&2\\2&2&3\end{pmatrix} \begin{pmatrix}0&2&3\\2&2&3\end{pmatrix} \begin{pmatrix}0&2&4\\2&2&3\end{pmatrix} \begin{pmatrix}0&2&5\\2&2&3\end{pmatrix} \begin{pmatrix}0&2&6\\2&2&3\end{pmatrix} \begin{pmatrix}0&2&7\\2&2&3\end{pmatrix} \begin{pmatrix}0&2&0\\2&2&4\end{pmatrix}$$

$$\begin{pmatrix}0&3&0\\1&2&3\end{pmatrix} \begin{pmatrix}0&3&1\\1&2&3\end{pmatrix} \begin{pmatrix}0&3&2\\1&2&3\end{pmatrix} \begin{pmatrix}0&3&3\\1&2&3\end{pmatrix} \begin{pmatrix}0&3&4\\1&2&3\end{pmatrix} \begin{pmatrix}0&3&5\\1&2&3\end{pmatrix} \begin{pmatrix}0&3&6\\1&2&3\end{pmatrix} \begin{pmatrix}0&3&7\\1&2&3\end{pmatrix} \begin{pmatrix}0&3&0\\1&2&4\end{pmatrix}$$

$$\begin{pmatrix}1&3&0\\1&2&3\end{pmatrix} \begin{pmatrix}1&3&1\\1&2&3\end{pmatrix} \begin{pmatrix}1&3&2\\1&2&3\end{pmatrix} \begin{pmatrix}1&3&3\\1&2&3\end{pmatrix} \begin{pmatrix}1&3&4\\1&2&3\end{pmatrix} \begin{pmatrix}1&3&5\\1&2&3\end{pmatrix} \begin{pmatrix}1&3&6\\1&2&3\end{pmatrix} \begin{pmatrix}1&3&7\\1&2&3\end{pmatrix} \begin{pmatrix}1&3&0\\1&2&4\end{pmatrix}$$

$$\begin{pmatrix}0&3&0\\2&2&3\end{pmatrix} \begin{pmatrix}0&3&1\\2&2&3\end{pmatrix} \begin{pmatrix}0&3&2\\2&2&3\end{pmatrix} \begin{pmatrix}0&3&3\\2&2&3\end{pmatrix} \begin{pmatrix}0&3&4\\2&2&3\end{pmatrix} \begin{pmatrix}0&3&5\\2&2&3\end{pmatrix} \begin{pmatrix}0&3&6\\2&2&3\end{pmatrix} \begin{pmatrix}0&3&7\\2&2&3\end{pmatrix} \begin{pmatrix}0&3&0\\2&2&4\end{pmatrix}$$

$$\begin{pmatrix}0&4&0\\1&2&3\end{pmatrix} \begin{pmatrix}0&4&1\\1&2&3\end{pmatrix} \begin{pmatrix}0&4&2\\1&2&3\end{pmatrix} \begin{pmatrix}0&4&3\\1&2&3\end{pmatrix} \begin{pmatrix}0&4&4\\1&2&3\end{pmatrix} \begin{pmatrix}0&4&5\\1&2&3\end{pmatrix} \begin{pmatrix}0&4&6\\1&2&3\end{pmatrix} \begin{pmatrix}0&4&7\\1&2&3\end{pmatrix} \begin{pmatrix}0&4&0\\1&2&4\end{pmatrix}$$

$$\begin{pmatrix}1&4&0\\1&2&3\end{pmatrix} \begin{pmatrix}1&4&1\\1&2&3\end{pmatrix} \begin{pmatrix}1&4&2\\1&2&3\end{pmatrix} \begin{pmatrix}1&4&3\\1&2&3\end{pmatrix} \begin{pmatrix}1&4&4\\1&2&3\end{pmatrix} \begin{pmatrix}1&4&5\\1&2&3\end{pmatrix} \begin{pmatrix}1&4&6\\1&2&3\end{pmatrix} \begin{pmatrix}1&4&7\\1&2&3\end{pmatrix} \begin{pmatrix}1&4&0\\1&2&4\end{pmatrix}$$

$$\begin{pmatrix}0&4&0\\2&2&3\end{pmatrix} \begin{pmatrix}0&4&1\\2&2&3\end{pmatrix} \begin{pmatrix}0&4&2\\2&2&3\end{pmatrix} \begin{pmatrix}0&4&3\\2&2&3\end{pmatrix} \begin{pmatrix}0&4&4\\2&2&3\end{pmatrix} \begin{pmatrix}0&4&5\\2&2&3\end{pmatrix} \begin{pmatrix}0&4&6\\2&2&3\end{pmatrix} \begin{pmatrix}0&4&7\\2&2&3\end{pmatrix} \begin{pmatrix}0&4&0\\2&2&4\end{pmatrix}$$

$$\begin{pmatrix}0&0&0\\1&3&3\end{pmatrix} \begin{pmatrix}0&0&1\\1&3&3\end{pmatrix} \begin{pmatrix}0&0&2\\1&3&3\end{pmatrix} \begin{pmatrix}0&0&3\\1&3&3\end{pmatrix} \begin{pmatrix}0&0&4\\1&3&3\end{pmatrix} \begin{pmatrix}0&0&5\\1&3&3\end{pmatrix} \begin{pmatrix}0&0&6\\1&3&3\end{pmatrix} \begin{pmatrix}0&0&7\\1&3&3\end{pmatrix} \begin{pmatrix}0&0&0\\1&4&3\end{pmatrix}$$

$$\begin{pmatrix}1&0&0\\1&3&3\end{pmatrix} \begin{pmatrix}1&0&1\\1&3&3\end{pmatrix} \begin{pmatrix}1&0&2\\1&3&3\end{pmatrix} \begin{pmatrix}1&0&3\\1&3&3\end{pmatrix} \begin{pmatrix}1&0&4\\1&3&3\end{pmatrix} \begin{pmatrix}1&0&5\\1&3&3\end{pmatrix} \begin{pmatrix}1&0&6\\1&3&3\end{pmatrix} \begin{pmatrix}1&0&7\\1&3&3\end{pmatrix} \begin{pmatrix}1&0&0\\1&4&3\end{pmatrix}$$

$$\begin{pmatrix}0&0&0\\3&2&3\end{pmatrix} \begin{pmatrix}0&0&1\\3&2&3\end{pmatrix} \begin{pmatrix}0&0&2\\3&2&3\end{pmatrix} \begin{pmatrix}0&0&3\\3&2&3\end{pmatrix} \begin{pmatrix}0&0&4\\3&2&3\end{pmatrix} \begin{pmatrix}0&0&5\\3&2&3\end{pmatrix} \begin{pmatrix}0&0&6\\3&2&3\end{pmatrix} \begin{pmatrix}0&0&7\\3&2&3\end{pmatrix} \begin{pmatrix}0&0&0\\4&2&3\end{pmatrix}$$

$$\begin{pmatrix}0&0&1\\1&2&3\end{pmatrix} \begin{pmatrix}0&0&2\\1&2&3\end{pmatrix} \begin{pmatrix}0&0&3\\1&2&3\end{pmatrix} \begin{pmatrix}0&0&4\\1&2&3\end{pmatrix} \begin{pmatrix}0&0&5\\1&2&3\end{pmatrix} \begin{pmatrix}0&0&6\\1&2&3\end{pmatrix} \begin{pmatrix}0&0&7\\1&2&3\end{pmatrix} \begin{pmatrix}0&0&0\\1&2&4\end{pmatrix} \begin{pmatrix}0&0&0\\1&2&3\end{pmatrix}$$