Sorting via shuffles with a cut after the longest increasing prefix

Lara Pudwell[a], Rebecca Smith[b]

[a]*Department of Mathematics and Statistics, Valparaiso University, Valparaiso, IN, USA*
[b]*Department of Mathematics, SUNY Brockport, Brockport, NY, USA*

Abstract

We define four new shuffling algorithms on permutations. For each algorithm, we characterize and enumerate the sets of permutations that are sorted after k iterations of the algorithm and we determine properties of the corresponding generating functions. For three of the algorithms, the sets of sortable permutations can be seen to be permutation classes for any nonnegative integer k.

Keywords: permutation, shuffle, sorting, algorithm, stacks, queues, permutation pattern, insertion encoding
2000 MSC: 05A05, 68P10, 68R01, 68R05

1. Introduction

Algorithms inspired by card shuffling have long been studied by researchers including an early combinatorial approach by Atkinson [4] that considered among other things, permutations obtainable by applying a riffle shuffle to the identity permutation. Applications of this riffle shuffle on permutations were investigated by others in a variety of contexts, including a study of supervised adaptive learning models by Žliobaitė [21] and a study of dominance drawings by Bannister, Devanney, and Eppstein [5]. In a more recent paper, Dimitrov [10] examined shuffle queues with certain restrictions and their use to sort permutations. While our shuffle algorithms can also be described in terms of queues and stacks, our approach to the use of these devices leads to different results than in previous works.

The individual steps of a shuffle algorithm can also be seen to have applications to biology. For example, all shuffles include cuts and some also include reversals of part of the permutation. These two actions on permutations each represent a genome rearrangement model studied by several researchers including Cerbai and Ferrari [8].

In this paper, we introduce four different shuffle sorting algorithms which can also be described in terms of systems of queues and stacks. All consider the longest increasing prefix as a natural starting point in the sorting process as a shuffle will cut a permutation into a prefix and suffix. Two of the algorithms retain the previous entries in the increasing prefix from one stage to the next. The other two prioritize moving smaller entries to the left of the permutation, but potentially break the monotonicity property held by the original increasing prefix. We note that other sorting algorithms such as stack sort also retain the longest increasing prefix as an increasing subsequence in the output. Some of our enumerative results include connections to the Eulerian numbers and Stirling numbers of the second kind. We first give necessary definitions and notation as well as a brief survey of related results. In Section 2, we consider two algorithms that consist of a well-defined cut and a riffle shuffle, and in Section 3 we explore the effects of adding an intermediate reversal step.

1.1. Permutations

Let $\mathcal{S}_n$ be the set of permutations of $[n] = \{1, 2, \ldots, n\}$. Given $\pi \in \mathcal{S}_n$ and $\rho \in \mathcal{S}_k$, we say that π *contains* ρ as a pattern if there exist $1 \leq i_1 < \cdots < i_k \leq n$ such that $\pi_{i_a} < \pi_{i_b}$ if and only if $\rho_a < \rho_b$; in this case we say that $\pi_{i_1}\pi_{i_2}\cdots\pi_{i_k}$ is *order-isomorphic* to ρ. Otherwise, π *avoids* ρ. Alternatively, let the *reduction* of the word w, denoted $\text{red}(w)$, be the word formed by replacing the ith smallest letter(s) of w with i. Then π contains ρ if there is a subsequence of π whose reduction is ρ.

Example 1.1. *The permutation* $\pi = 35841726$ *contains the permutation* $\rho = 3241$ *since the reduction of the subsequence* 5472 *is* $\text{red}(5472) = 3241$.

A *permutation class* C is defined to be a set of permutations that is "downward closed", that is, if $\pi \in C$ and π contains σ, then $\sigma \in C$. Specifically, the permutation class $\text{Av}(\rho)$ is the set of all permutations that avoid ρ. The growth rate of any permutation class C, is defined as:

$$\text{gr}(C) = \lim_{n\to\infty} \sqrt[n]{|C_n|} \qquad \text{where } C_n = \{\tau \in C \mid \text{length of } \tau \text{ is } n\}.$$

Note that while it is conjectured that such limits exist and are finite for classes avoiding any nonempty basis set of permutations, it is only known that $\limsup_{n\to\infty} \sqrt[n]{|Av(B)|}$ exists and is finite for nonempty B by Marcus and Tardos [14]. We discuss the growth rates of sortable permutation classes in Section 4.

While patterns will show up in part of our work, a pair of more critical definitions is that of ascents and descents. An *ascent* of permutation π is an index i where $\pi_i < \pi_{i+1}$, while a *descent* is an index i where $\pi_i > \pi_{i+1}$. We denote the number of ascents of π by $\operatorname{asc}(\pi)$ and the number of descents by $\operatorname{des}(\pi)$.

1.2. Sorting Functions and Sorting Networks

A *sorting function* is a function $f : \mathcal{S}_n \to \mathcal{S}_n$ such that for all $\pi \in \mathcal{S}_n$, there exists a non-negative $i \in \mathbb{Z}$ such that $f^i(\pi) = 123\cdots n$. There are many well-studied sorting functions including bubble sort, insertion sort, and selection sort. In this paper, we develop four sorting algorithms that correspond to sorting functions motivated by shuffling cards. In particular, a common way to shuffle is to cut a deck into two non-empty parts and then to *riffle* the two parts together, so that each part remains in order, but the two parts are interleaved. In practice, a deck can be cut anywhere, and a riffle may interleave the two parts of the deck in many different ways. However, we choose conventions that make these operations well-defined.

In particular, in Section 2 we consider two algorithms that consist of a cut after the longest increasing prefix of the permutation followed by a riffle shuffle with specific rules on which part contributes the next card to output. In Section 3, we consider two algorithms that again begin with a cut after the longest increasing prefix, but then reverse the second part of the permutation prior to using a riffle shuffle, again with specific rules on which part contributes the next card to output. The cut (and reverse) portions of all four algorithms can be modeled by using the standard data structures of stacks and queues.

A *stack* is a last-in first-out data structure with push and pop operations. Knuth [13] studied permutations that are sortable by one pass through a stack; in other words, there is a sequence of push and pop operations to transform the permutation $\pi \in \mathcal{S}_n$ into the increasing permutation $1\cdots n$ as output. Knuth showed that a permutation is sortable by one pass through a stack if and only if π avoids the pattern 231. There are C_n such permutations of length n, where $C_n = \dfrac{\binom{2n}{n}}{n+1}$ is the nth Catalan number. Other researchers have studied networks with multiple stacks in series or in parallel, including Even and Itai [11], Pratt [16], and Tarjan [19]. For more information on stack sorting, see the survey by Bóna [7].

On the other hand, a *queue* is a first-in first-out data structure. It is quick to see that using a single queue, the only permutation that can be sorted is the increasing permutation $12\cdots n$. However, networks of multiple queues or queues and stacks allow for more possibilities.

When we cut a deck and riffle it together, we may view this as a system of two queues in parallel. After the cut, the elements of one part go into one queue and the elements of the other part go into the other queue. The riffle determines the sequence in which the elements of the two queues are interleaved in output.

When we cut a deck, then reverse the second half before riffling, we may view this is a system of a queue and a stack in parallel. The elements of the first part are placed into a queue, while the elements of the second part are pushed one at a time into a stack, which reverses them. Again, the riffle determines the sequence in which the elements of the queue and the stack are interleaved in output.

2. Simple shuffle algorithms

In this section, we consider two algorithms based on a cut-and-riffle shuffle algorithm. In both cases, the permutation is cut after the longest increasing prefix. However, we use two different conventions governing the riffle that interleaves the two parts. The Prefix-preserving Shuffle (PRE) prioritizes keeping all of the original prefix as part of the maximum increasing prefix of the newly shuffled permutation. The Minimum Shuffle (MIN) instead prioritizes shuffling so that smaller entries appear before larger entries whenever possible.

Algorithm 2.1. *Prefix-preserving Shuffle: PRE*

Given a permutation π with first descent at $\pi_{i-1} > \pi_i$, let $\pi' = \pi_1 \cdots \pi_{i-1}$ and $\pi'' = \pi_i \cdots \pi_n$. Then interleave π' with π'' as follows:

1. *If the next available entry b of π'' is smaller than the next available entry a of π' but larger than the current last entry of output (or $b < a$ and output is currently empty), then pop b to the end of output.*
2. *Else, if π' and π'' both still have entries, so $a < b$ or b is smaller than the current last entry of the output, pop a to the end of output.*

3. *Once either π' or π'' has been pushed entirely to the output, push the remaining entries of the other sequence to the output.*

We consider the action of algorithm PRE in more detail following an example of the algorithm at work.

Example 2.2. *Let $\pi = 261543$. The cut after the longest increasing prefix gives us $\pi' = 26$ and $\pi'' = 1543$. We illustrate the shuffle process taken by PRE in Figure 1 that gives an output of PRE$(261543) = 125643$.*

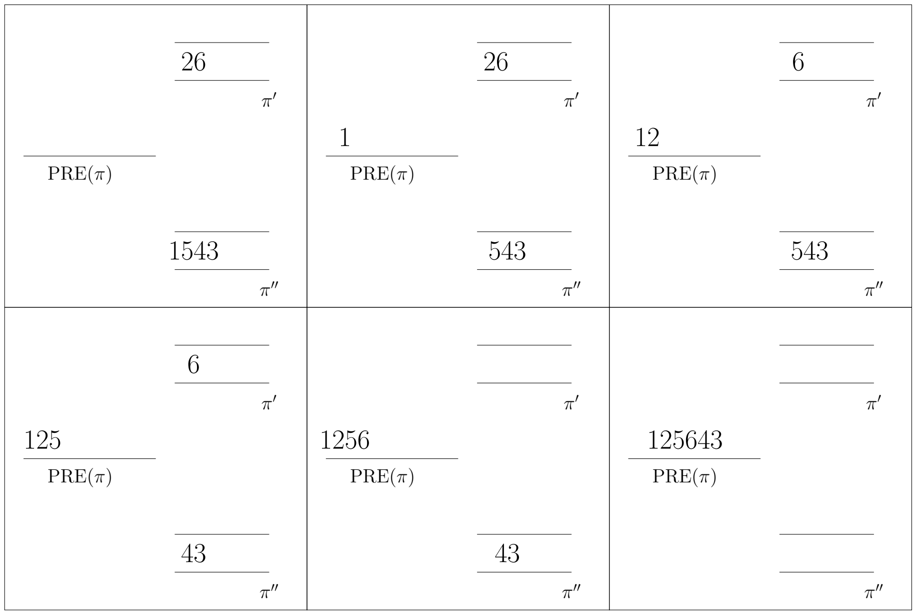

Figure 1: One iteration of the PRE algorithm applied to $\pi = 261543$.

Proposition 2.3. *Algorithm PRE is a sorting algorithm.*

Proof. During algorithm PRE, π' is an increasing prefix. As long as π'' is nonempty, at least the first entry of π'' is moved to its appropriate position within among the entries of π' to form a new increasing prefix in PRE(π). As such, the output of algorithm PRE on a non-identity permutation π has a longer increasing prefix than the input. Since the increasing prefix grows with each iteration until the resulting permutation is the identity, the permutation will be sorted once the increasing prefix grows to be the entire permutation. □

In fact, we can characterize the permutations that require exactly k iterations of algorithm PRE to be sorted.

Proposition 2.4. *Any permutation π is sorted after exactly* $\operatorname{des}(\pi)$ *iterations of algorithm PRE.*

Proof. We know that $\pi' = \pi_1 \cdots \pi_{i-1}$ is increasing and that $\pi_{i-1} > \pi_i$. Further, we may locate the first descent of π'' (if it exists) and decompose it into $\pi''^{\star} = \pi_i \cdots \pi_{j-1}$ and $\pi''' = \pi_j \cdots \pi_n$ where $\pi''^{\star}$ is increasing and $\pi_{j-1} > \pi_j$, otherwise $\pi''^{*} = \pi''$ if π'' has no descents.

Notice that steps 1 and 2 interleave π' and $\pi''^{\star}$ in increasing order. However, since $\pi_j < \pi_{j-1}$, based on step 2, all elements of π''' are at the end of the permutation. In other words, the descent between π_{i-1} and π_i is removed. If π'' had a descent to begin with, there is still a descent ending in π_j and all descents within π''' are still descents in the output.

Since the number of descents of π is reduced by exactly 1 after each iteration of algorithm PRE, $\operatorname{des}(\pi)$ is the number of iterations required to sort π. □

Given that the number of permutations of length n with exactly k descents is the Eulerian number $A(n, k)$, the enumeration of the sortable permutations is known.

Corollary 2.5. *For all $k \geq 0$, the number of permutations in $\mathcal{S}_n$ that are sortable after exactly k passes of algorithm PRE is given by the Eulerian number $A(n, k)$ (OEIS A008292).*

Eulerian numbers have been connected to other shuffle-based sorting algorithms. For example, Bayer and Diaconis [6] consider sorting via a cut and riffle shuffle where the cut is arbitrary and divides the deck into a packets of cards and the riffle is determined probabilistically by the relative sizes of the parts of the deck. They determine the probability that an a-packet shuffle results in a particular permutation π based on the number of descents of π. In a similar vein, they refer to rising sequences as the “basic invariant of riffle shuffling”, where, naturally the Eulerian numbers count permutations with a prescribed number of rising sequences.

Next, we introduce MIN, an alternative algorithm to PRE, where we prioritize moving smaller entries to the left over maximizing the length the increasing prefix.

Algorithm 2.6. *Minimum-first Shuffle: MIN*

Given a permutation π with first descent at $\pi_{i-1} > \pi_i$, let $\pi' = \pi_1 \cdots \pi_{i-1}$ and $\pi'' = \pi_i \cdots \pi_n$. Then interleave π' with π'' as follows:

1. *If the next available entry a of π' is smaller than the next available entry b of π'', then pop a to the end of output.*
2. *Else, if π' and π'' both still have entries, so $b < a$, pop b to the end of output.*
3. *Once one sequence has been pushed entirely to the output, push the remaining entries of the other sequence to the output.*

Notice that algorithm MIN is distinct from algorithm PRE. For example, if $\pi = 261543$, both algorithms cut π so that $\pi' = 26$ and $\pi'' = 1543$. While the output of algorithm PRE is 125643 (shown in Example 2.2), the output of algorithm MIN is 125436 as shown in Example 2.7. However, the two algorithms share some interesting features which are noted following Example 2.7.

Example 2.7. *Let $\pi = 261543$. The cut after the longest increasing prefix gives us $\pi' = 26$ and $\pi'' = 1543$. We illustrate the shuffle process taken by MIN in Figure 2.*

Proposition 2.8. *Algorithm MIN is a sorting algorithm.*

Proof. By definition, π' is an increasing subsequence of π. The MIN algorithm shuffles the initial prefix with remainder of the permutation retaining the order of the two subsequences and prioritizing outputting the minimum entry of each subsequence. Hence all of the entries of the initial increasing prefix π' will remain in the same relative order after every iteration of the MIN algorithm.

Notice any entries that become part of the increasing prefix of the permutation output by the MIN algorithm are now correctly placed relative to entries of the original π' that remain part of the increasing prefix. Further, these new members of the increasing prefix must be part of an increasing subsequence with all of the entries of the entire original π'.

Specifically, at least the first entry of π'' is moved to its appropriate position within the increasing prefix of the output and thus the increasing subsequence. As such, this increasing subsequence of current and former members of increasing prefixes of π and the output permutations after applying the MIN algorithm grows in length after each iteration of the MIN algorithm until there are no new entries available to be inserted into this increasing subsequence. Thus the permutation will be sorted by MIN once this increasing subsequence grows to be the entire permutation. □

Like the PRE algorithm, we can characterize the permutations that require exactly k iterations of algorithm MIN to be sorted by again considering the descents of a permutation and the image of the permutation under MIN. In fact, the permutations sortable by k iterations of MIN are exactly the same as those permutations sortable by k iterations of PRE despite the fact that the intermediate steps may result in different permutation images.

Proposition 2.9. *Any permutation π is sorted after exactly* $\text{des}(\pi)$ *iterations of algorithm MIN.*

Proof. We know that $\pi' = \pi_1 \cdots \pi_{i-1}$ is increasing and that $\pi_{i-1} > \pi_i$.

Notice that steps 1 and 2 interleave π' and π'' so that π_i is no longer the second entry in a descent in output. All other descents must have both elements in π''. However, if $\pi_j > \pi_{j+1}$ for two consecutive digits in π'' and π_j

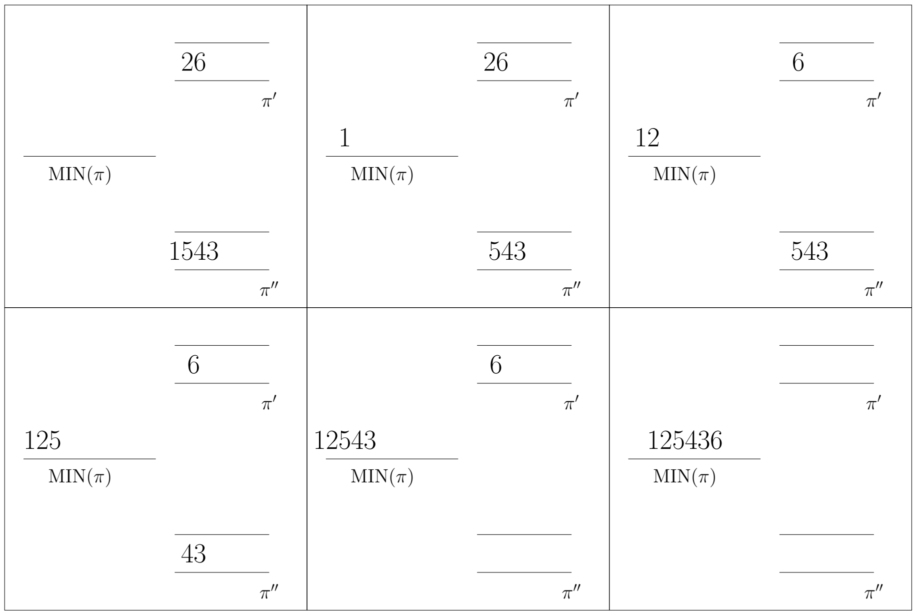

Figure 2: One iteration of the MIN algorithm applied to $\pi = 261543$.

is less than the current available element of π', then so is π_{j+1}, so they will remain a descent in output.

Since the number of descents of π is reduced by exactly 1 after each iteration of algorithm MIN until the identity permutation is achieved, $\mathrm{des}(\pi)$ is the number of iterations required to sort π. $\square$

Corollary 2.10. *For all $k \geq 0$, the number of permutations in $\mathcal{S}_n$ that are sortable after exactly k passes of algorithm MIN is given by the Eulerian number $A(n, k)$ (OEIS A008292).*

3. Reverse shuffle algorithms

While the two algorithms of Section 2 were relatively straightforward to analyze, in this section, we consider two new algorithms (PRE-REV) and (MIN-REV) which act as algorithms PRE and MIN respectively, but where the second part of the original permutation is reversed before being interleaved with the longest increasing prefix.

Algorithm 3.1. *Prefix-preserving Reverse Shuffle: PRE-REV*

Given a permutation π with first descent at $\pi_{i-1} > \pi_i$, let $\pi' = \pi_1 \cdots \pi_{i-1}$ and $(\pi'')^{rev} = \pi_n \cdots \pi_i$ (i.e. the reversal of $\pi_i \cdots \pi_n$). Then interleave π' with $(\pi'')^{rev}$ as follows:

1. *If the next available entry b of $(\pi'')^{rev}$ is smaller than the next available entry a of π' but larger than the current last entry of output (or $b < a$ and output is currently empty), then pop b to the end of output.*
2. *Else, if π' and $(\pi'')^{rev}$ both still have entries, so $a < b$ or b is smaller than the current last entry out output, pop a to the end of output.*
3. *Once one sequence has been pushed entirely to the output, push the remaining entries of the other sequence to the output.*

Example 3.2. *Let $\pi = 261543$. As stated before in the earlier algorithm examples, the cut after the longest increasing prefix gives us $\pi' = 26$ and $\pi'' = 1543$, making $(\pi'')^{rev} = 3451$. We illustrate the shuffle process taken by PRE-REV in Figure 3.*

Algorithm 3.3. *Minimum-first Reverse Shuffle: MIN-REV*

Given a permutation π with first descent at $\pi_{i-1} > \pi_i$, let $\pi' = \pi_1 \cdots \pi_{i-1}$ and $(\pi'')^{rev} = \pi_n \cdots \pi_i$ (i.e. the reversal of $\pi_i \cdots \pi_n$). Then interleave π' with $(\pi'')^{rev}$ as follows:

1. *If the next available entry a of π' is smaller than the next available entry b of $(\pi'')^{rev}$, then pop a to the end of output.*
2. *Else, if π' and $(\pi'')^{rev}$ both still have entries, so $b < a$, pop b to the end of output.*
3. *Once one sequence has been pushed entirely to the output, push the remaining entries of the other sequence to the output.*

Example 3.4. *Let $\pi = 261543$. Once again we note the cut after the longest increasing prefix gives us $\pi' = 26$ and $\pi'' = 1543$, making $(\pi'')^{rev} = 3451$. We illustrate the shuffle process taken by MIN-REV in Figure 4.*

As in Section 2, the PRE-REV algorithm can be seen to sort every permutation after a sufficient number of iterations by showing the length of the increasing prefix increases with each implementation of the algorithm.

Proposition 3.5. *Algorithm PRE-REV is a sorting algorithm.*

Proof. After each iteration of algorithm PRE-REV, all entries of the input's longest increasing prefix remain in the prefix of the output, and the entry π_n of input is also put in its appropriate position in the longest increasing prefix

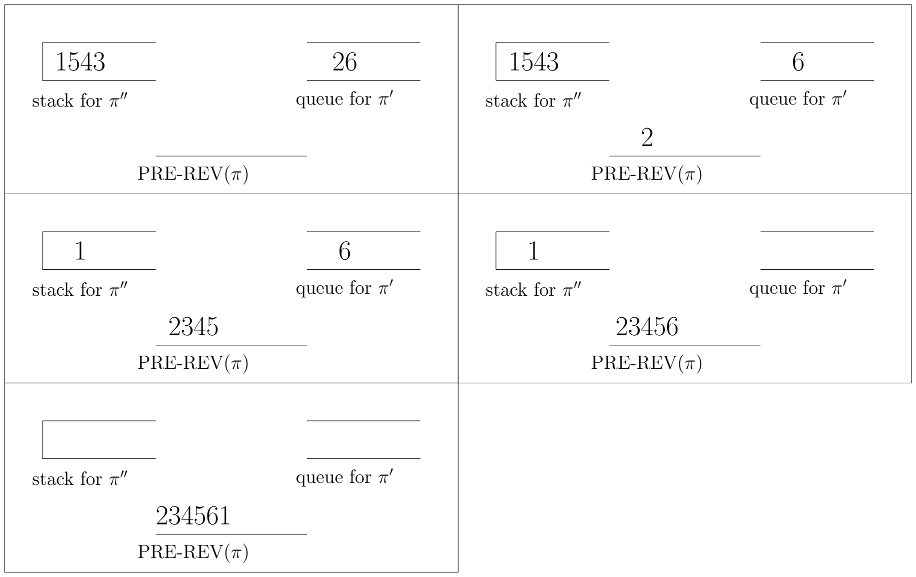

Figure 3: One iteration of the PRE-REV algorithm applied to $\pi = 261543$.

of the output. The longest increasing prefix grows with each iteration of the algorithm until the entire output is the identity permutation. □

Notice that the difference between the output of PRE-REV and that of MIN-REV when applied to $\pi = 261543$ as shown in Figure 3 and Figure 4 respectively occurs when the application of the PRE-REV algorithm pops the 6 as the penultimate entry of the output preserving the initial increasing prefix, but the application of the MIN-REV algorithm pops the 1 in that same position, prioritizing the smaller entry.

It is more difficult to see that algorithm MIN-REV is a sorting algorithm. However, in the next subsection we give a characterization of the permutations sortable after k iterations of the PRE-REV algorithm and a characterization of the permutations sortable after k iterations of the MIN-REV algorithm. In the latter case, this characterization proves MIN-REV is also a sorting algorithm.

3.1. Characterization

In order to characterize permutations sortable after k iterations of algorithm PRE-REV, we introduce the *prefix-suffix decomposition* of π as follows: Let $\pi^{(1)} = \pi' = \pi_1 \cdots \pi_{i-1}$ be the longest increasing prefix of π and let

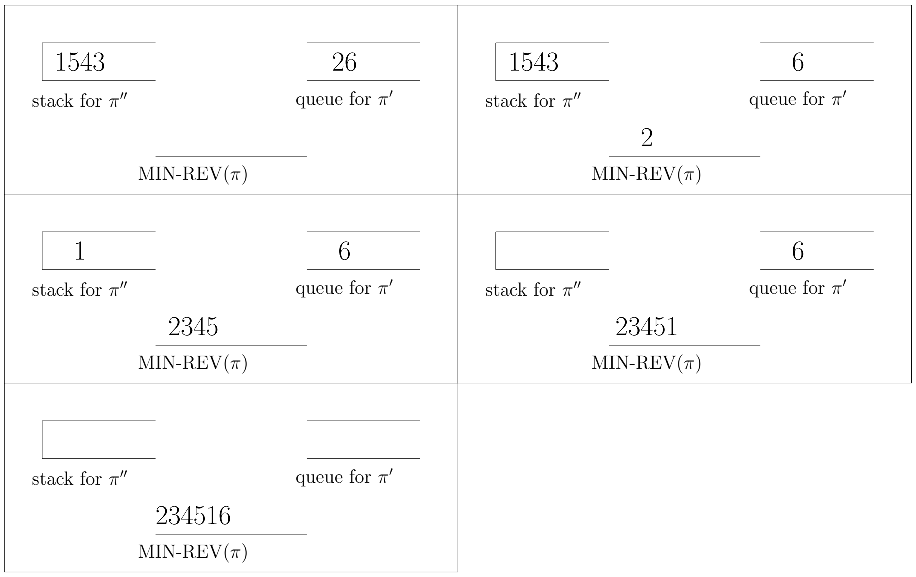

Figure 4: One iteration of the MIN-REV algorithm applied to $\pi = 261543$.

$\pi^{rev(1)} = (\pi'')^{rev}$ be the reversal of $\pi_i \cdots \pi_n$. If $\pi^{rev(1)}$ is empty, then we are done. Otherwise, given $\pi^{(1)}, \ldots, \pi^{(\ell)}$, set $\pi^{(\ell+1)}$ to be the longest increasing prefix of $\pi^{rev(\ell)}$ and recursively define $\pi^{rev(\ell+1)}$ to be the reversal of the remaining digits. Then if j is the smallest integer such that $\pi^{rev(j)}$ is empty, the prefix-suffix decomposition of pi is $\pi^{(1)}\pi^{(2)} \cdots \pi^{(j)}$.

For example, the permutation $\pi = 562793841$ has $\pi^{(1)} = 56$, $\pi^{rev(1)} = 1483972$, $\pi^{(2)} = 148$, $\pi^{rev(2)} = 2793$, $\pi^{(3)} = 279$, $\pi^{rev(3)} = 3$, and $\pi^{(4)} = 3$.

Theorem 3.6. *Consider $\pi \in \mathcal{S}_n$. If there are $k+1$ parts in the prefix-suffix decomposition of π, then algorithm PRE-REV requires exactly k iterations to sort π.*

Proof. As with algorithm PRE, algorithm PRE-REV interleaves two monotone sequences within π and then places the rest of the permutation at the end. In this case, the two interleaved monotone sequences on the first pass of the algorithm are $\pi^{(1)} = \pi'$ and $\pi^{(2)}$, which is the longest increasing prefix of $(\pi'')^{rev}$. After reversal of $\pi'' = \pi_i \cdots \pi_n$, the other elements outside of $\pi^{(1)}$ and $\pi^{(2)}$ are at the end of the permutation, in the reverse of their original order. More generally, after the ith iteration, all elements of monotone subsequences $\pi^{(1)}, \ldots, \pi^{(i+1)}$ will be interleaved in increasing order as the current longest increasing prefix, while the remaining elements are at the end of the permutation. All parts of a $(k+1)$-part prefix-suffix decomposition will be interleaved in increasing order after k iterations of the algorithm. $\square$

While the prefix-suffix decomposition of a permutation completely determines the sortability by PRE-REV, it is not relevant to sorting by MIN-REV because the two algorithms have different priorities in how they interleave subsequences. However, we can also completely characterize the set of permutations sortable after exactly k iterations of the MIN-REV algorithm in terms of the quantities of ascents and descents relative to largest entry of the permutation.

Theorem 3.7. *Consider a non-identity permutation $\pi \in \mathcal{S}_n$. Suppose there are d descents before n and a ascents after n. Then π requires exactly $\max(2d, 2a+1)$ applications of MIN-REV to be sorted.*

Proof. First, consider the case where $d = 0$ and $a = 0$. This implies that π is unimodal with the longest increasing prefix ending with n. After cutting the permutation and reversing the digits after n, we riffle together two increasing subsequences, and so $\max(0,1) = 1$ iterations of the algorithm are required, as expected.

Now, suppose at least one of d and a is positive. Let D be the set of smaller digits involved in a descent before n and let A be the set of smaller digits involved in an ascent after n in π. Also, let p be the entry of π immediately after the longest increasing prefix of π. Now, let $D^\star$ and $A^\star$ be the analogous sets for $\pi^\star$ where $\pi^\star$ is obtained by applying one iteration of MIN-REV to π. We claim that $D^\star = A$ and $A^\star = D \setminus \{p\}$.

First, we show that $D^\star = A$. Consider an arbitrary entry $\pi_j \in A$. Since π_j appears after n in π, π_j is not part of the longest increasing prefix. Thus π_j is in π''. Recall π'' is the part of the permutation that gets reversed during the algorithm. Specifically, π_{j+1} appears to the left of π_j in $(\pi'')^{rev}$, therefore π_{j+1} gets interleaved with π' before π_j. We can see π_{j+1} also will be shuffled in to the left of n in this algorithm. This is because n is either part of the increasing prefix π' and every element of $(\pi'')^{rev}$ will be shuffled in before n or n appears to the left of π_{j+1} in π'' and so n appears to the right of π_{j+1} in $(\pi'')^{rev}$.

Once π_{j+1} is placed into its position in $\pi^\star$, it must be the case π_j is smaller than the next available digit of π' (if any exist). Hence $\pi_{j+1}\pi_j$ becomes a descent in $\pi^\star$. Descents formed this way, in fact, are exactly the descents of $\pi^\star$ before n since there are no descents in the longest increasing prefix, and the algorithm uses minimums to decide which elements of the prefix and the non-prefix portion of π to output first. Thus exactly the elements π_j described here will make up the set $D^\star$.

Next, we claim that $A^\star = D \setminus \{p\}$. Suppose that $\pi_j \in D \setminus \{p\}$. This means that π_j is not part of the longest increasing prefix of π and so $\pi_j \in \pi''$. Since $\pi_j > \pi_{j+1}$, and because π_j appears before n in π, we will read these entries after n is placed into the output $\pi^\star$. Thus the prefix of π will be completely output by the time these entries are read and they will be output as the ascent $\pi_{j+1}\pi_j$ in $\pi^\star$. Thus $\pi_j \in A^\star$. Conversely, any entries of $A^\star$ appear after n in $\pi^\star$ and so must have been before n but after the prefix in π, so except for p these entries of $D \setminus \{p\}$ are the only possible entries of $A^\star$.

Next, we claim that $A^\star = D \setminus \{p\}$. Suppose that $\pi_j \in D \setminus \{p\}$. This means that π_j is not part of the longest increasing prefix of π and so $\pi_j \in \pi''$. Also, because $\pi_j \neq p$, we have $\pi_{j-1} \in \pi''$. Since $\pi_{j-1} > \pi_j$, and because π_j appears before n in π, we will read these entries after n is placed into the output $\pi^\star$. Thus the prefix of π will be completely output by the time these entries are read and they will be output as the ascent $\pi_j\pi_{j-1}$ in $\pi^\star$. Thus

$\pi_j \in A^\star$. Conversely, any entries of $A^\star$ appear after n in $\pi^\star$ and so must have been before n but after the prefix in π, so except for p these entries of $D \setminus \{p\}$ are the only possible entries of $A^\star$.

Now, we show $p \notin A^\star$. If p appears after n, then n is part of the increasing prefix, so $D = \emptyset$ and thus $D \setminus \{p\} = \emptyset$. However, if p appears before n in π, we have $p \in D$, but also p will be the first entry of π''. Thus p will be the last entry of $\pi^\star$ and so cannot be the first entry of an ascent in $\pi^\star$. Hence $p \notin A^\star$

Now, we see that if D contains d elements, then it will take $2d$ iterations to address all of d elements from D. Specifically, in an odd iteration, one element, p, is removed from D and the rest are sent to $A^\star$. In even iterations, those elements are sent back to set D, so that after $2d$ iterations is the first time $D = \emptyset$. Similarly, if A contains a elements, it will take $2a + 1$ iterations to address all of them. On the first iteration, the elements of A are all sent to $D^\star$ and then it takes $2a$ further iterations for all of them to be addressed. □

We are now ready to prove the following:

Corollary 3.8. *Algorithm MIN-REV is a sorting algorithm.*

Proof. If the input permutation π is the identity, then it requires 0 iterations of any algorithm to be sorted. Similarly, if π is unimodal (increasing before n and decreasing after n), then it requires 1 iteration of MIN-REV to be sorted. Otherwise, we know that if the input permutation π has d descents before n and a ascents after n, then the output after one iteration has a descents before n and $\max(d-1, 0)$ ascents after n. Since the numbers in the ordered pair (d, a) decrease with successive iterations, eventually, we have the first instance where $d = 0$ and $a = 0$ which implies we have a unimodal permutation that can be sorted after one more iteration of the algorithm. □

While these two characterizations seem quite different, the simplest case of permutations sortable after one iteration of either PRE-REV or MIN-REV is the same.

Proposition 3.9. *Both the PRE-REV-sortable permutations and the MIN-REV-sortable permutations are precisely the unimodal permutations, i.e. the* $\{213, 312\}$*-avoiding permutations.*

Proof. Permutations avoiding 213 and 312 are precisely the unimodal permutations. Consider a unimodal permutation π where $\pi_j = n$. The PRE-REV algorithm will prioritize not creating a descent before π_j in the shuffle, but otherwise interleave entries from the prefix and the reverse of the suffix in increasing order. The MIN-REV algorithm will prioritize smaller entries before larger entries. These priorities align when shuffling two increasing sequences, so in this instance, these algorithms act identically. Namely, both algorithms will interleave the two increasing sequences $\pi_1 \cdots \pi_j$ and $\pi_n \cdots \pi_{j+1}$ to create one increasing sequence which is simply the identity permutation.

Now, instead suppose that a single application of either reverse shuffle algorithm can be used to sort π. Then, since the increasing prefix is shuffled with the reverse of the remainder of the permutation, it must be the case that the entries after the maximum increasing prefix are in descending order. Hence any permutation sortable by PRE-REV or MIN-REV in a single iteration must be unimodal. □

However, as we consider permutations that require $k > 1$ iterations of the algorithms, the behaviors of PRE-REV and MIN-REV diverge. Consider, for example, what Theorem 3.6 and Theorem 3.7 tell us about the permutations sortable after two iterations.

By Theorem 3.6, the permutations sortable after exactly two iterations of Algorithm PRE-REV are precisely those that have three parts in their prefix-suffix decomposition. In other words, these permutations consist of an increasing prefix, another increasing subsequence, and then a decreasing sequence. In particular, we have $\pi = \pi^{(1)}\pi^{(3)}\pi^{(2*)}$ where $\pi^{(1)} = \pi_1 \cdots \pi_{i-1}$ is increasing, $\pi^{(3)} = \pi_i \cdots \pi_{j-1}$ is increasing and $\pi^{(2*)} = \pi_j \cdots \pi_n$ is the reversal of $\pi^{(2)}$ in the original prefix-suffix decomposition, and therefore is decreasing. We also necessarily have that $\pi_{i-1} > \pi_i$ and $\pi_{j-1} < \pi_j$.

On the other hand, by Theorem 3.7, permutations in $\mathcal{S}_n$ are sortable after exactly two iterations of Algorithm MIN-REV are those where there is exactly one descent before n and no ascents after n. These permutations also have three parts in their prefix-suffix decomposition, but there is a more specific condition on the placement of n, so permutations sortable after two passes of algorithm MIN-REV are a subset of permutations sortable after two passes of algorithm PRE-REV. This will be examined further in the enumeration section with the above result explicitly stated in Proposition 3.18 and extended in Theorem 3.21.

Example 3.10. *The permutation* $\pi = 24816753$ *requires five iterations of* *the MIN-REV algorithm to be sorted:*

$$\begin{aligned}\textit{MIN-REV}(\pi) &= 23457618\\ \textit{MIN-REV}^2(\pi) &= 23457816\\ \textit{MIN-REV}^3(\pi) &= 23456178\\ \textit{MIN-REV}^4(\pi) &= 23456871\\ \textit{MIN-REV}^5(\pi) &= 12345678\end{aligned}$$

However, sorting π only requires two applications of the PRE-REV algorithm.

$$\begin{aligned}\textit{PRE-REV}(\pi) &= 23457861\\ \textit{PRE-REV}^2(\pi) &= 12345678\end{aligned}$$

These algorithms also differ in their worst case scenarios. The largest number of parts in a prefix-suffix decomposition of $\pi \in \mathcal{S}_n$ is n parts, which requires $n-1$ iterations. This occurs with the permutations consisting of a decreasing prefix of $\lfloor \frac{n+1}{2} \rfloor$ elements ending in 1, followed by an increasing suffix of the remaining elements. This analysis is recorded in Corollary 3.14. On the other hand, for $\pi \in \mathcal{S}_n$ the number of ascents after n is maximized for the permutation $\pi = n12\cdots(n-1)$, which has $n-2$ ascents, and therefore requires $2(n-2)+1 = 2n-3$ iterations of algorithm MIN-REV to be sorted.

3.2. Enumeration

In Proposition 3.9, we showed that permutations sortable after one iteration of either of these cut, reverse, then riffle algorithms are exactly the $\{213, 312\}$-avoiding permutations, which gives the following corollary.

Corollary 3.11. *The number of permutations of length n sortable by at most one iteration of PRE-REV or MIN-REV is 2^{n-1}.*

However, it is also possible to develop a general formula for permutations sortable after exactly k passes through either of our algorithms. To that end, let $A(n,k)$ be the number of permutations of length n with exactly k descents; i.e., $A(n,k)$ is the triangle of Eulerian numbers (OEIS A008292). Reversing permutations provides a bijection between those with k ascents and those with k descents, so $A(n,k)$ is also the number of permutation of length n with exactly k ascents.

We introduce a refinement of the Eulerian numbers where $B(n,k,i)$ is the number of permutations of length n with exactly k ascents that begin with the digit i, or equivalently, $B(n,k,i)$ is the number of permutations of length n with exactly k descents that end with the digit i. While limited numerical evidence seems to suggest a relationship between $B(n,k,i)$ sequences and inflated s-Eulerian Polynomials (OEIS A333270) studied by Pensyl and Savage in 2013 [15], we could not find an indication that this refinement of the Eulerian numbers has been studied before. From the definitions for $A(n,k)$ and $B(n,k,i)$, we have:

$$A(n,k) = \sum_{i=1}^{n} B(n,k,i).$$

We can also give a recurrence for $B(n,k,i)$.

Proposition 3.12. *If $B(n,k,i)$ is the number of permutations of length n with exactly k ascents that begin with i, then for $1 \leq i \leq n$ and $0 \leq k \leq n-1$*

$$B(n,k,i) = \begin{cases} 1 & n = 1 \\ \displaystyle\sum_{j=1}^{i-1} B(n-1,k,j) + \sum_{j=i}^{n-1} B(n-1,k-1,j) & \textit{otherwise.} \end{cases}$$

Proof. When $n = 1$, the bounds on i and k imply that $i = 1$ and $k = 0$. And the one permutation of length $n = 1$ is $\pi = 1$ which begins with 1 and has 0 ascents.

For the second case, we consider the relationship between the first digit i and the second digit j of such a permutation. If $1 \leq j \leq i-1$, then the first two entries in the permutation form a descent, so the rest may be recursively filled in with a permutation of length $n-1$ with k ascents and first value j on the remaining entries of $[n] = \{1,2,3,\ldots,n\}$. When that permutation of length $n-1$ is reduced to a permutation on $[n-1]$, the value of j remains the same since $j < i$ meaning these permutations are counted by $\displaystyle\sum_{j=1}^{i-1} B(n-1,k,j)$.

If instead, $i+1 \leq j \leq n$, then the first two entries in the permutation form an ascent, so the rest may be recursively filled in with a permutation of length $n-1$ with $k-1$ ascents on the remaining entries of $[n]$. Now, reducing the recursively constructed permutation of length $n-1$ to a permutation on $[n-1]$ will decrease the value of j by 1 in the recursion since $j > i$. That is, we sum over $i \leq j \leq n-1$ and so these permutations are counted by $\displaystyle\sum_{j=i}^{n-1} B(n-1, k-1, j)$. □

We now have the necessary tools to count permutations sortable after k iterations of the PRE-REV and MIN-REV algorithms.

Theorem 3.13. *Let* $\mathrm{pr}(n,k)$ *be the number permutations of size* n *sortable after exactly* k *iterations of the PRE-REV algorithm. First we have,*

$$\mathrm{pr}(n,0) = 1 \text{ for } n \geq 1.$$

Then if $k > 0$ *is even, then*

$$\mathrm{pr}(n,k) = \sum_{m=0}^{n} \sum_{i=1}^{n-m} \sum_{j=1}^{m} \sum_{\ell=0}^{m-j} B^*_{even}(n,k,m,i,j,\ell)$$

where $B^*_{even}(n,k,m,i,j,\ell)$ *is equal to*

$$B\left(n-m, \frac{k}{2}-1, i\right) B\left(m, \frac{k}{2}, j\right) \binom{\ell+n-m-i}{\ell} \binom{m-\ell+i-1}{i-1}.$$

Finally, if k *is odd, then*

$$\mathrm{pr}(n,k) = \sum_{m=0}^{n} \sum_{i=1}^{n-m} \sum_{j=1}^{m} \sum_{\ell=0}^{n-m-i} B^*_{odd}(n,k,m,i,j,\ell)$$

where $B^*_{even}(n,k,m,i,j,\ell)$ *is equal to*

$$B\left(n-m, \frac{k-1}{2}, i\right) B\left(m, \frac{k-1}{2}, j\right) \binom{\ell+m-j}{\ell} \binom{n-m-\ell+j-1}{j-1}.$$

Proof. The only permutation that requires 0 iterations of any algorithm to be sorted is the increasing permutation. There is one such permutation of each length.

For the even case, since any permutation counted requires exactly k iterations of the PRE-REV algorithm to be sorted, the permutation's prefix-suffix decomposition has $k+1$ parts, so the number of parts is odd. This means there are $\frac{k}{2}$ decreasing suffixes (with $\frac{k}{2}-1$ ascents among them) and $\frac{k}{2}+1$ increasing prefixes (with $\frac{k}{2}$ descents among them). Furthermore, the first digit of the innermost suffix must be larger than the last digit of the innermost prefix. If there are m digits involved in the increasing prefixes, then there are $n-m$ digits involved in the decreasing suffixes. Altogether, if the last prefix ends with digit j (when restricting to only the digits involved in the prefixes), then there are $B\left(m, \frac{k}{2}, j\right)$ ways to choose (the reverse of) the permutation made of the prefix digits and if the first suffix begins with digit i (when restricting to only the digits involved in the suffixes), then there are $B\left(n-m, \frac{k}{2}-1, i\right)$ ways to choose the permutation made of the suffix digits.

It remains to choose the actual values that are involved in the prefix versus the suffix. Suppose there are ℓ digits in the prefix that are larger than i. Clearly $0 \leq \ell$, but we can also see $\ell \leq m-j$. This is because the last of the prefix portion of the permutation is the jth largest entry of those first m entries, but is also smaller than the first entry of the suffix portion of the permutation which implies that at least j of the m prefix digits are smaller than the ith largest entry of the suffix portion of the permutation. There are exactly $n-m-i$ digits larger than the first entry of the suffix portion of the permutation within the suffixes, so there are $\binom{\ell+n-m-i}{\ell}$ ways to choose which of the largest $\ell+n-m-i$ are part of the prefixes and which are part of the suffixes. Similarly, there are $m-\ell$ digits less than the first entry of the suffix portion of the permutation in the prefixes and $i-1$ digits less than that first entry of the suffix portion of the permutation in the suffixes, so there are $\binom{m-\ell+i-1}{i-1}$ ways to choose which of the smallest $m-\ell+i-1$ digits are part of the prefixes and which are part of the suffixes.

For the odd case, we have a similar argument, however now there are $\frac{k+1}{2}$ prefixes and $\frac{k+1}{2}$ suffixes. Also, the last digit of the innermost prefix must be larger than the first digit of the innermost suffix, so we use ℓ to count digits larger than than j in the suffix portion of the permutation. $\square$

Theorem 3.13 gives a direct way to count permutations requiring exactly k iterations of algorithm PRE-REV in terms of refined Eulerian numbers. The difference in the odd vs. even cases will make finding a simpler form challenging. However, this recurrence is sufficiently efficient to provide enumerative data. Table 1 shows the exact values of $\mathrm{pr}(n,k)$ for small values of n and k. Of note – while we explicitly addressed the $k=0$ and $k=1$ columns already, another nice pattern exists in the $n=k+1$ diagonal. Permutations requiring a maximum number of sorts are exactly the $\binom{n-1}{\lfloor \frac{n-1}{2} \rfloor}$ permutations initially described in conclusion of Subsection 3.1.

$n\backslash k$	0	1	2	3	4	5	6	7
1	1							
2	1	1						
3	1	3	2					
4	1	7	13	3				
5	1	15	58	40	6			
6	1	31	221	325	132	10		
7	1	63	774	2086	1711	385	20	
8	1	127	2577	11655	16841	7931	1153	35

Table 1: Values of $\mathrm{pr}(n,k)$ for small n and k

Corollary 3.14. *There are $\binom{n-1}{\lfloor\frac{n-1}{2}\rfloor}$ permutations of length n that require the maximum number of iterations, namely $k = n-1$, to be sorted by the PRE-REV algorithm. The permutations are precisely the permutations that are formed by the concatenation of a decreasing sequence of length $\lceil\frac{n-1}{2}\rceil$, the entry 1 and then followed by an increasing subsequence of length $\lfloor\frac{n-1}{2}\rfloor$.*

Proof. This proof again utilizes the prefix-suffix structure developed for Theorem 3.6 and the corresponding result, namely that a permutation having $k+1$ parts in the prefix-suffix decomposition requires exactly k iterations of the PRE-REV algorithm to be sorted. As a permutation of length n has a maximum of n parts, $k = n-1$ is indeed the maximum number of iterations required to sort any permutation in S_n.

Further, these parts would be $\lceil\frac{n}{2}\rceil$ increasing sequences followed by $\lfloor\frac{n}{2}\rfloor$ decreasing sequences. For a permutation to have $n-1$ parts, if n is odd, the 1 must be the last increasing "sequence" and if n is even, the 1 must be the last (leftmost) decreasing "sequence". Such a permutation can thus be described as beginning with $\lceil\frac{n-1}{2}\rceil$ descents followed by $\lfloor\frac{n-1}{2}\rfloor$ ascents. These permutations are enumerated by $\binom{n-1}{\lfloor\frac{n-1}{2}\rfloor}$ because they are found by choosing the $\lfloor\frac{n-1}{2}\rfloor$ entries that make up the increasing sequence that follows the 1 and placing all other entries in decreasing order before the 1. $\square$

Example 3.15. *The permutation $\pi = 53124$ requires exactly 4 iterations of the PRE-REV algorithm to be sorted. Similarly, the permutation $\tau = 652134$ requires exactly 5 iterations of the PRE-REV algorithm to be sorted.*

Next, we give a formula for the number of permutations of length n sorted by exactly k iterations of the MIN-REV algorithm. This formula is also dependent on the parity of k, but the refinement of the Eulerian numbers is not needed here and each expression requires only two summations.

Theorem 3.16. *Let* $\mathrm{mr}(n,k)$ *be the number of permutations of size* n *sortable after exactly* k *iterations of the MIN-REV algorithm. We have*

$$\begin{aligned}&\mathrm{mr}(n,0)=1 \textit{ for } n\geq 1 \textit{ and}\\&\mathrm{mr}(n,1)=2^{n-1}-1 \textit{ for } n\geq 1.\end{aligned}$$

Further, if $k>1$ *is even, then*

$$\mathrm{mr}(n,k)=\sum_{i=0}^{n-1}\sum_{a=0}^{\frac{k}{2}-1}A\left(i,\frac{k}{2}\right)A(n-i-1,a)\binom{n-1}{i}.$$

Finally, if $k>1$ *is odd, then*

$$\mathrm{mr}(n,k)=\sum_{i=0}^{n-1}\sum_{d=0}^{\frac{k-1}{2}}A\left(i,d\right)A\left(n-i-1,\frac{k-1}{2}\right)\binom{n-1}{i}.$$

Proof. The $k=0$ case addresses the already-sorted increasing permutation. The $k=1$ case was addressed in Corollary 3.11.

By Theorem 3.7, we know the number of iterations required to sort permutation $\pi\in\mathcal{S}_n$ is given by $\min(2d,2a+1)$ where d is the number of descents before n in π and a is the number of ascents after n.

If $k=\max(2d,2a+1)$ is even, then $d=\frac{k}{2}$ and $2a+1\leq 2d$. Combining these observations yields that $a\leq\frac{k-1}{2}$. If there are i digits before the n, we may choose the relative order of the digits of the permutation before n with $d=\frac{k}{2}$ descents in $A\left(i,\frac{k}{2}\right)$ ways. We may choose the relative order of the digits of permutation after n, which has $n-i-1$ digits an a ascents in $A(n-i-1,a)$ ways. Then, we have $\binom{n-1}{i}$ ways to choose the actual values of the digits that go before n versus after n. Summing over all possible values of a and i yields the result.

If $k=\max(2d,2a+1)$ is odd, then $a=\frac{k-1}{2}$ and $2d\leq 2a+1$. Combining these observations yields that $d\leq\frac{k}{2}$, and since k is odd and d is an integer, we can more precisely write this bound as $d\leq\frac{k-1}{2}$. A similar argument to the even case yields the result. □

This result is simpler than the formula for $\mathrm{pr}(n,k)$ and it is even more efficient at quickly providing exact values of $\mathrm{mr}(n,k)$ for small values of n and k. A table of initial values is given in Table 2.

$n\backslash k$	0	1	2	3	4	5	6	7	8	9	10	11
1	1											
2	1	1										
3	1	3	1	1								
4	1	7	7	7	1	1						
5	1	15	33	39	15	15	1	1				
6	1	31	131	211	141	141	31	31	1	1		
7	1	63	473	1123	1128	1148	488	488	63	63	1	1

Table 2: Values of $\mathrm{mr}(n,k)$ for small n and k

It follows from Theorem 3.16 and known values of $A(n,k)$, that

$$\mathrm{mr}(n,2) = 3^{n-1} - (n+1)2^{n-2},$$

$$\mathrm{mr}(n,3) = 4^{n-1} - (2n+1)3^{n-2} + \frac{(n^2-n)2^n}{8},$$

and

$$\mathrm{mr}(n,4) = 5^{n-1} - (3n+1)4^{n-2} + \frac{(5n^2-3n-2)3^n}{54} - \frac{(n^3-2n^2-n+2)2^n}{32}.$$

It can be shown that the leading term of $\mathrm{mr}(n,k)$ is $(k+1)^{n-1}$ for arbitrary k. Via Comtet [9] it is known that

$$A(n,k) = \sum_{j=0}^{k+1}(-1)^j\binom{n+1}{j}(k-j+1)^n,$$

which has leading term $(k+1)^n$. Focusing on leading terms, the even case of our expression for $\mathrm{mr}(n,k)$ has leading term

$$\sum_{i=0}^{n-1}\left(\frac{k}{2}+1\right)^i\left(\frac{k}{2}\right)^{n-i-1}\binom{n-1}{i} = (k+1)^{n-1}.$$

Similarly, the odd case of our expression for $\mathrm{mr}(n,k)$ has leading term

$$\sum_{i=0}^{n-1}\left(\frac{k-1}{2}+1\right)^i\left(\frac{k-1}{2}+1\right)^{n-i-1}\binom{n-1}{i} = (k+1)^{n-1}.$$

In both cases, the simplification of the summation is an application of the binomial theorem. More patterns or perhaps even a more direct path to these closed formulas may be possible with further analysis in this direction. For example, in these smaller cases, there appears to be a pattern in $\mathrm{mr}(n, k)$ where the second term is $-((k-1)n+1)k^{n-2}$.

The proof of the formula for the case when $k \leq 2$ using inclusion-exclusion is given in Proposition 3.17 below.

Proposition 3.17. *The number of permutations of length $n \geq 1$, that require $k \leq 2$ iterations of algorithm MIN-REV shuffles to be sorted is $3^{n-1} - (n-1)2^{n-2}$.*

Proof. Such permutations of length n can have at most one descent before the n and no ascents after the n and so can consist of at most two monotone increasing subsequences followed by at most one decreasing subsequence. There are 3^{n-1} ways to place the entries $1, 2, \ldots, n-1$ into one of the three monotone sequences. However, this enumeration counts each unimodal permutation beginning with exactly j entries before the n of the permutation $j+1$ times for the $j+1$ ways to break up a single increasing sequence into two parts (where one part is possibly empty). Thus, we subtract out the overcount of $\sum_{j=0}^{n-1} \binom{n-1}{j} j = (n-1)2^{n-2}$ to get the aforementioned enumeration. That is,

$$\mathrm{mr}(n, 2) + \mathrm{mr}(n, 1) + \mathrm{mr}(n, 0) = 3^{n-1} - (n-1)2^{n-2}.$$

□

The case when $k = 2$ is where the sorting power of the PRE-REV algorithm and MIN-REV algorithm diverge. Further, the earlier structures of sortable permutations under both algorithms can be used to explicitly describe and count the permutations that are sorted by two iterations of PRE-REV, but not by two iterations of MIN-REV.

Proposition 3.18. *Every permutation sortable after two iterations of algorithm MIN-REV is also sortable after two iterations of algorithm PRE-REV.*

Proof. The permutations sortable only after exactly two iterations of algorithm MIN-REV have one descent before n and no ascents after n. That is, they consist of two increasing sequences followed by a decreasing sequence. The prefix-suffix decomposition of such permutations have exactly these three parts. □

Theorem 3.19. *The permutations of length n that are sortable by after two iterations of algorithm PRE-REV, but not two iterations of MIN-REV are counted by $S(n, 3)$ where $S(n, k)$ denotes the Stirling numbers of the second kind.*

Proof. To prove this theorem, we must show these permutations sortable by PRE-REV, but not MIN-REV are in bijection with the number of ways to place the entries of $[n]$ into three nonempty sets. From Proposition 3.17, we know that the permutations sortable by at most $k = 2$ iterations of the MIN-REV algorithm are either unimodal or have $k = 3$ parts in the prefix-suffix decomposition and have n being the end of the second increasing subsequence in the prefix-suffix decomposition.

Thus permutations we are counting are those of the form
$\pi = \pi_1 \cdots \pi_{i-1}\pi_i \cdots \pi_{j-1}\pi_j \cdots \pi_n$ where

1. $\pi_1 \cdots \pi_{i-1}$ is the maximum length increasing prefix and $\pi_{i-1} = n$,
2. $\pi_i \cdots \pi_{j-1}$ is a nonempty increasing subsequence,
3. $\pi_j \cdots \pi_n$ is a nonempty decreasing subsequence where π_j is a peak of π,
4. and each of these sequences is nonempty.

Notice these three sequences of entries are in natural bijection with unlabeled sets containing the corresponding entries because n is always in the first sequence and of the other two sequences, $\{\pi_j, \ldots, \pi_n\}$ contains the largest remaining entry. □

Given that $S(n, 3) = \dfrac{3^{n-1} + 1}{2} - 2^{n-1}$, we have the following result.

Corollary 3.20. *The number of permutations sortable after two iterations of algorithm PRE-REV of length $n \geq 1$ is $\dfrac{3^n + 1}{2} - (n + 1)2^{n-2}$.*

Proof. Combining the results of Proposition 3.17, Proposition 3.18, and Theorem 3.19, one can see the number of permutations of length n sortable after two iterations of algorithm PRE-REV is

$$3^{n-1} - (n-1)2^{n-2} + \frac{3^{n-1}+1}{2} - 2^{n-1} = \frac{3^n+1}{2} - (n+1)2^{n-2}.$$

□

The numerical evidence given Tables 1, 2 shows for at least small values of n and k, PRE-REV sorts at least as many permutations as MIN-REV does after k iterations. In fact, we can also extend Proposition 3.18 to show the application of PRE-REV k times sorts every permutation sorted by MIN-REV applied k times. That is, the PRE-REV algorithm is at least as efficient as the MIN-REV algorithm for sorting any permutation.

Theorem 3.21. *Any permutation sorted by k iterations of the MIN-REV algorithm can be sorted by at most k iterations of the PRE-REV algorithm.*

Proof. Suppose a permutation π requires exactly k iterations of MIN-REV to be sorted. The prefix-suffix structure of the permutation can be partially determined as described in Theorem 3.7 and then further investigated depending on whether k is even or odd.

Let d be the number of descents in π occurring before n appears and a be the number of ascents in π occurring after n. Regardless of the parity of k, we know $\pi = \rho_1\rho_2\cdots\rho_{d+1}\rho_{d+2}\cdots\rho_{d+a+2}$ where $\rho_1, \rho_2, \ldots, \rho_{d+1}$ are increasing subsequences and $\rho_{d+2}, \rho_{d+3}, \ldots, \rho_{d+a+2}$ are decreasing subsequences. The position of the n means that we can say either ρ_{d+2} begins with n or ρ_{d+1} ends with n.

In the case when k is even, $k = 2d$ and $a < d$. The first $a+1$ increasing subsequences $\rho_1, \rho_2, \ldots, \rho_{a+1}$ and the $a+1$ decreasing subsequences $\rho_{d+2}, \rho_{d+3}, \ldots, \rho_{d+a+2}$ are part of the prefix-suffix decomposition of π utilized in Theorem 3.6. Note that ρ_{a+2} is also preserved in the prefix-suffix decomposition of π, but will be handled with the remaining parts.

The remaining part of the prefix-suffix decomposition of π comes from balancing the monotone subsequences of π that are not already accounted for, namely $\rho_{a+2}\rho_{a+3}\cdots\rho_{d+1}$. However, because this consists of precisely $d-a$ increasing subsequences, the prefix-suffix decomposition from this portion can have a maximum of $d-a$ increasing subsequences and $d-a-1$ decreasing subsequences.

Thus π has at most $2(a+1) + (d-a) + (d-a-1) = 2d+1$ parts in its prefix-suffix decomposition and can be sorted in at most $2d = k$ iterations of the PRE-REV algorithm.

The case when k is odd is similar in a symmetric sense. Now $k = 2a + 1$ and $d \leq a$. The $d + 1$ increasing subsequences $\rho_1, \rho_2, \ldots, \rho_{d+1}$ and the last $d + 1$ decreasing subsequences $\rho_{a+2}, \rho_{a+3}, \ldots, \rho_{d+a+2}$ are part of the prefix-suffix decomposition of π utilized in Theorem 3.6.

The rest of the prefix-suffix decomposition of π again involves balancing what remains of π in $\rho_{d+2}\rho_{d+3} \cdots \rho_{a+1}$. However, because this portion of π consists of $a-d$ decreasing subsequences, the prefix-suffix decomposition here can have a maximum of $a - d$ increasing subsequences and $a - d$ decreasing subsequences.

Thus once again, π has at most $2(d+1)+(a-d)+(a-d) = 2a+2$ parts in its prefix-suffix decomposition and can be sorted in at most $2a + 1 = k$ iterations of the PRE-REV algorithm. $\square$

At the end of Section 3, we showed that the worst case scenario for MIN-REV is the input $\pi = n12 \cdots (n - 1)$, which requires $2n - 3$ iterations of the algorithm, as it is the only permutation to have $n - 2$ ascents after n. However, there is also the numerical evidence showing there is also exactly one second-worst permutation of length $n \geq 2$ requiring $2n - 4$ iterations of the MIN-REV algorithm that can be explained by the simple reversal bijection which when applied to π gives $\pi^{\text{rev}} = (n-1) \cdots 21n$, a permutation with $n - 2$ descents before n which gives us the following result.

Proposition 3.22. *For $n \geq 2$, there is a single permutation of length n requiring the maximum $2n-3$ iterations of the MIN-REV algorithm. Further, there is exactly one permutation of length n requiring $2n-4$ iterations of the MIN-REV algorithm.*

Using the same technique as was used in the more general Theorem 3.16, we can enumerate the permutations of length $n \geq 3$ requiring $2n-5$ iterations of MIN-REV to be sorted and apply the reverse bijection to get the same enumeration for permutations requiring $2n - 6$ iterations of MIN-REV.

Proposition 3.23. *For $n \geq 3$, there are $2^{n-1} - 1$ permutations of length n requiring $2n-5$ iterations of the MIN-REV algorithm. Further, there are also $2^{n-1}-1$ permutations of length n requiring $2n-6$ iterations of the MIN-REV algorithm if $n \geq 4$.*

Proof. When $n = 3$, the $2^{3-1} - 1 = 3$ permutations requiring $2 \cdot 3 - 5 = 1$ iteration of the MIN-REV algorithm have already been determined to be the non-identity unimodal permutations 132, 231, and 321.

When $n \geq 4$, the permutations requiring $2n-5$ iterations of MIN-REV are exactly the permutations with $n-3$ ascents after n. These consist of two types of permutations. One type is made up of the $n-1$ permutations that begin with any entry other than n, have n in the second position, and then have the remaining entries in ascending order. The other type is made up of the $2^{n-1}-n$ permutations beginning with n followed by a permutation on $[n-1]$ having exactly one descent. This gives us the desired enumeration of $2^{n-1}-1$ such permutations.

Then, if $n \geq 4$, the reversal of all of the permutations described above not only guarantee a permutation with $n-3$ descents before the n, but also, with at most one entry after the n after the reversal is complete, it is impossible that the permutation could have any ascents after the n. Thus these permutations all require $2n-6$ iterations of the MIN-REV algorithm. □

The symmetry in the previous propositions continues to hold for smaller values of k provided that we can guarantee there are not as many ascents after the n than than descents before the n after reversing the permutations whose sortability was determined by the ascents after n.

4. Permutation Classes

We now consider the possible description of permutations sortable by at most k shuffles in terms of pattern classes.

For both MIN and REV, recall the patterns sortable by at most k shuffles are exactly those permutations that have at most k descents. Because the insertion of new elements into a permutation can never decrease the number of descents in a permutation, for each value k the permutations sortable by k iterations of PRE or MIN can also be described using classical pattern avoidance. For example, the permutations that have at most one descent and are thus sortable by (at most) one iteration of either the PRE or MIN algorithm are the permutations that avoid $321, 2143,$ and 3142.

As mentioned in the justification for Corollary 3.11, the permutations sortable by at most one iteration of either PRE-REV or MIN-REV are the $\{213, 312\}$-avoiding permutations. For larger values of k, recall the permutations sortable by k iterations of the PRE-REV algorithm have structure defined by their prefix-suffix decomposition. Once again, there is no way to "repair" an unsortable permutation by the insertion of more elements. That is, the insertion of more elements into a permutation will not reduce the number of components in its prefix-suffix decomposition. Thus, for each value of k, the permutations sortable by k iterations of PRE-REV form a permutation class as they are characterized by pattern avoidance. We give the characterization for the case of $k = 2$ next.

Proposition 4.1. *The permutations sortable by at most $k = 2$ iterations of PRE-REV avoid the permutations in the basis*

$$B = \{3214, 4213, 4312, 21435, 21534, 31425, 31524, 41523\}.$$

Proof. One can verify all of the permutations in B have four components in their prefix-suffix decomposition. That is, these permutations begin with two contiguous increasing subsequences and end with two contiguous decreasing subsequences with no way to reduce to beginning with two contiguous increasing subsequences followed by only one contiguous decreasing subsequence.

Conversely, consider a smallest permutation σ ordered by pattern containment that has at least, and thus exactly, four components in its prefix-suffix decomposition. That is, a permutation that begins with two contiguous increasing subsequences and end with two contiguous decreasing subsequences with no way to reduce to beginning with two contiguous increasing subsequences followed by only one contiguous decreasing subsequence. As such, the permutation σ contains at least two descents followed by at least one ascent.

First, any permutation that begins with an increasing sequence of length greater than one can be reduced to a smaller permutation with the same number of components in its prefix-suffix decomposition by removing all but the last entry in this increasing sequence. Thus (σ_1, σ_2) forms a descent and so σ_2 would be the first entry of our second contiguous increasing subsequence in the prefix-suffix decomposition.

In the case that $\sigma_2 \neq 1$, we claim (σ_2, σ_3) forms a second descent. To see this, suppose that instead there is a longer than length one contiguous increasing subsequence starting at σ_2. At some point, there must be a descent and later an ascent after which σ will be complete. Thus the 1 of σ must appear as part of the first contiguous descending subsequence and thus the other entries of both the second contiguous increasing sequence and the first contiguous decreasing subsequence besides σ_2 and 1 can be removed leaving a smaller forbidden permutation $\sigma_1\sigma 21\sigma_m$. The only permutations of this form are $3214, 4213, 4312$.

Otherwise, if $\sigma_2 = 1$, then $(1, \sigma_3)$ must form an ascent. Then, as before, (σ_3, σ_4) a descent as otherwise, the entries after 1 and before the first entry of the second descent are again redundant. Similarly, (σ_4, σ_5) form an ascent completing σ. Permutations of this form do not contain $3214, 4213, 4312$ exactly when $\sigma_1 < \sigma_3$. Thus the only remaining elements of the basis are $21435, 21534, 31425, 31524, 41523$. $\square$

We can say more about the generating functions $P_k(x) = \sum_{n\geq 0} pr(n,k)x^n$ using the notion of insertion encoding introduced by Albert, Linton, and Ruškuc [2], utilized in works including that of Vatter and the second author [18], and further studied by Vatter [20].

Definition 4.2. *A permutation can be defined in terms of a word describing the* insertion encoding*. This is done by considering how a permutation would be constructed inserting one entry at a time in increasing order of the values of the entries into* slots *which will be spaces that will be filled by at least one entry. The possible letters of the word are:*

- l_i*: Indicates the insertion of* π_j *into the leftmost position of slot* i*. Hence* $\diamond \rightarrow \pi_j\diamond$*.*
- m_i*: Indicates the insertion of* π_j *into a middle position of slot* i*. Hence* $\diamond \rightarrow \diamond\pi_j\diamond$*.*
- r_i*: Indicates the insertion of* π_j *into the rightmost position of slot* i*. Hence* $\diamond \rightarrow \diamond\pi_j$*.*
- f_i*: Indicates the insertion of* π_j *into the last available entry of slot* i*. Hence* $\diamond \rightarrow \pi_j$*.*

Note that the label on a slot is relative to the other slots at that moment in the algorithm. The resulting word will give the insertion encoding of the permutation.

Example 4.3. *The permutation* $\pi = 6725143$ *has insertion encoding* $m_1m_1r_3f_3f_2l_1f_1$

where the procedure is:

$\diamond_1$	
$\diamond_1 1 \diamond_2$	m_1
$\diamond_1 2 \diamond_2 1 \diamond_3$	$m_1 m_1$
$\diamond_1 2 \diamond_2 1 \diamond_3 3$	$m_1 m_1 r_3$
$\diamond_1 2 \diamond_2 143$	$m_1 m_1 r_3 f_3$
$\diamond_1 25143$	$m_1 m_1 r_3 f_3 f_2$
$6 \diamond_1 25143$	$m_1 m_1 r_3 f_3 f_2 l_1$
6725143	$m_1 m_1 r_3 f_3 f_2 l_1 f_1$

Using insertion encoding as well as the structure of each permutation class of permutations sortable by at most k iterations of PRE-REV, we will show each class forms a regular language by invoking Corollary 10 from the paper "The insertion encoding of permutations" by Albert, Linton, and Ruškuc [2]. This corollary tells us that any permutation class that can be constructed by an insertion encoding where the number of slots is bounded is a regular language. Since regular languages have rational generating functions, this completes our proof.

Theorem 4.4. *For any nonnegative integer k, the generating function*

$$P_k(x) = \sum_{n \geq 0} pr(n,k) x^n$$

is rational.

Proof. Consider a nonnegative integer k. The permutations sortable by k iterations of PRE-REV have at most $k+1$ components in their prefix-suffix notation. That is, these permutations begin with at most $\lceil \frac{k+1}{2} \rceil$ increasing sequences followed by $\lfloor \frac{k+1}{2} \rfloor$ decreasing sequences.

To see that the class of permutations sortable by k iterations of PRE-REV is regular, we will show that this class can be constructed by an insertion coding that never has more than k open slots. First, consider the number of open slots appearing starting at the left side of encoding of a permutation. Each open slot that appears before an already inserted entry and will be filled by a larger entry, and thus will result in a descent.

There are at most $\lceil \frac{k+1}{2} \rceil$ increasing sequences at the start of this permutation before at most $\lfloor \frac{k+1}{2} \rfloor$ decreasing sequences at the end of the permutations, so we can have up to $\lceil \frac{k+1}{2} \rceil$ open slots for the descents that will eventually indicate the end of these increasing sequences. Then, by a symmetric argument, there can be at most $\lfloor \frac{k+1}{2} \rfloor - 1$ open slots to the right of entries forming the up to $\lfloor \frac{k+1}{2} \rfloor$ decreasing sequences because these slots would be ascents indicating the end of a decreasing sequence (and the start of a new sequence).

Hence the class of permutations sortable by k iterations of PRE-REV has an insertion encoding that only has k open slots at any stage and is thus regular. Since this permutation class is regular, it has a rational generating function. $\square$

We remark that Theorem 4.4 can also be seen as a special case of Theorem 8.1 of "Geometric grid classes of permutations" by Albert, Atkinson, Bouvel, Ruškuc, and Vatter [1]. Further, we can obtain the growth rate of this class of sortable permutations.

Theorem 4.5. *Let PR_k be the class of permutations sorted after k iterations of the PRE-REV algorithm. Then $gr(PR_k) = k + 1$ for any nonnegative integer k.*

Proof. The permutations of PR_k are exactly the juxtapositions of $\lceil \frac{k+1}{2} \rceil$ Av(21) permutations followed by $\lfloor \frac{k+1}{2} \rfloor$ Av(12) permutations. Thus, as noted in work by Albert, Pantone, and Vatter [3], the growth rate of the entire class is the sum of the $k + 1$ individual growth rates of the monotone classes. $\square$

There is no classical pattern class for MIN-REV for $k = 2$ which can be determined by the enumeration shown in Table 2 that indicates we have 5 sortable permutations of length three, but 15 of length 4. This is because every principle class of permutations avoiding one permutation of length 3 is enumerated by the Catalan numbers as shown by Knuth [12] and so only has 14 permutations of length 4. Looking closer at the sortable permutations, we can specifically see that 312 is the only permutation of length three that is not sortable by two iterations of MIN-REV. However, 3142, which contains 312 is sortable by two iterations in MIN-REV since $(\text{MIN-REV})^2(3142) = \text{MIN-REV}(2341) = 1234$.

It is expected inserting a larger element in the permutation will continue to change the process substantially enough so that MIN-REV will not yield pattern classes for any $k > 2$, but we have not further studied what the characterizations might be in terms of mesh patterns, what the generating functions would look like, or what the asymptotic behavior for the class might look like.

5. Conclusion

It is our hope that this analysis of deterministic shuffle sorting increases interest in these specific algorithms as well as in the broader areas of sorting and shuffling algorithms. In addition to the enumerative results found in this paper, numerical evidence indicates many additional directions to explore and we note a few options here.

The formulas given for $pr(n, k)$ in Theorem 3.13 and $mr(n, k)$ Theorem 3.16 completely determine the enumeration of sortable permutations for both PRE-REV and MIN-REV, but it would be nice to have simpler descriptions if possible.

Open Question 5.1. *Is there a simpler formula for either $pr(n, k)$ or $mr(n, k)$ for general k or specific values of k beyond what was found here?*

In particular, values $\mathrm{mr}(n, k)$ are directly related to the Eulerian numbers and we used the Eulerian numbers to determine the leading term of $\mathrm{mr}(n, k)$ for any fixed k and arbitrary n. What more can be said about subsequent terms? We conjecture the following pattern, but it remains open to prove it.

Open Question 5.2. *Is it true that the second term of* $\mathrm{mr}(n, k)$ *is* $-((k-1)n+1)k^{n-2}$*?*

Proposition 3.9 shows the number of permutations sortable after one iteration of either PRE-REV or MIN-REV is 2^{n-1}, i.e., the number of unimodal permutations. Theorem 3.21, gives the enumeration of the permutations sortable by two iterations of PRE-REV, but not two iterations of MIN-REV as the Stirling number of the second kind $S(n, 3)$. Further, in Theorem 3.21, we showed that all permutations sorted by k iterations of MIN-REV are also sorted by k iterations of PRE-REV. Finding $b(n) = pr(n, k) - mr(n, k)$, i.e., the generalized numerical difference in the number of permutations sorted by k iterations of each algorithm, could be useful in determining simpler formulas for $pr(n, k)$ and $mr(n, k)$.

Open Question 5.3. *What can be determined about $b(n, k)$ for $k > 2$?*

Even determining $b(n,3)$ could lead to nice ways to describe $pr(n,3)$ and $mr(n,3)$. There is limited numerical evidence that suggests $b(n,3)$ could be related to a known sequence in OEIS. Specifically, that evidence suggests $b(n,3)$ could be twice an offset of sequence $a(n)$ defined in A102296 from the OEIS [17].

Open Question 5.4. *Does $b(n,3) = 2a(2(n-5)) = (1/3)((2(n-5)+1)(10[2(n-5)]^2 + 17[2(n-5)] + 12)$? If so, why? Either way, is there a nice combinatorial way to describe the difference $b(n,3) = pr(n,3) - mr(n,3)$?*

We have only taken a limited look at the relationship between our algorithms and pattern classes, so there is much more to investigate here, particularly in regards to the permutation classes formed by the sortable permutations by k iterations of PRE-REV.

Open Question 5.5. *What are the explicit permutation classes that form the sortable permutations by $k > 2$ iterations of PRE-REV? In fact, any further characterization or simplification the enumeration of the permutations sortable by k iterations of PRE-REV would be interesting.*

We have determined some specifics about the PRE-REV algorithm that establish certain characteristics of the bivariate generating function

$$P(x,y) = \sum_{n\geq 0} pr(n,k)x^n y^k$$

including the main diagonal where $n = k+1$, from Corollary 3.14, However, we have not explored the general properties of $P(x,y)$.

Open Question 5.6. *What else can be said about $P(x,y)$? Specifically, is $P(x,y)$ a rational generating function and what is the asymptotic behavior of $P(x,y)$?*

Looking the permutations sortable by $k > 1$ iterations of MIN-REV is also a potentially interesting question.

Open Question 5.7. *Can the permutations sortable by $k > 1$ iterations of MIN-REV be described nicely in terms of mesh patterns?*

6. Acknowledgements

The authors wish to thank the referees for helpful feedback that improved the presentation of this paper. They also thank Jay Pantone for helpful input that informed the lines of exploration taken in this project.

References

[1] M. H. Albert, M. D. Atkinson, M. Bouvel, N. Ruškuc, and V. Vatter, Geometric grid classes of permutations, *Transactions of the American Mathematical Society* **365.11** (2013) 5859–5881.

[2] M. H. Albert, S. Linton, and N. Ruškuc, The Insertion Encoding of Permutations, *Electron. J. Comb.* **12** (2005) #R47.

[3] M. H. Albert, J. Pantone, and V. Vatter, On the growth of merges and staircases of permutation classes, *The Rocky Mountain Journal of Mathematics* **49.2** (2019).

[4] M. D. Atkinson, Restricted permutations, *Discrete Math.* **195**, 1-3 (1999), 27–38.

[5] M. J. Bannister, W. E. Devanny, and D. Eppstein, Small superpatterns for dominance drawing, In 2014 Proceedings of the Eleventh Workshop on Analytic Algorithmics and Combinatorics (ANALCO), Society for Industrial and Applied Mathematics, 92–103.

[6] D. Bayer and P. Diaconis, Trailing the Dovetail Shuffle to its Lair, *Ann. Appl. Probab.* **2** (1992) 294–313.

[7] M. Bóna, A survey of stack-sorting disciplines, *Electron. J. Comb.* **9.2** (2002-3) #A1.

[8] G. Cerbai and L. Ferrari, Permutation patterns in genome rearrangement problems: the reversal model, *Discret. Appl. Math.* **279** (2020) 34–48.

[9] L. Comtet, Permutations by Number of Rises; Eulerian Numbers. Section 6.5 in *Advanced Combinatorics: The Art of Finite and Infinite Expansions*, rev. enl. ed. Dordrecht, Netherlands: Reidel, p. 243, 1974.

[10] S. Dimitrov, Sorting by shuffling methods and a queue, *Electron. J. Comb.* **29.3** (2022) #P3.23.

[11] S. Even and A. Itai, Queues, stacks, and graphs, In Theory of machines and computations (Proc. Internat. Sympos., Technion, Haifa, 1971). Academic Press, New York, 1971, 71–86.

[12] D. E. Knuth, The art of computer programming. Volume 1, Addison-Wesley Publishing Co., Reading, Mass., 1969. Fundamental algorithms, Addison-Wesley Series in Computer Science and Information Processing.

[13] D. E. Knuth, The art of computer programming. Volume 3, Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont., 1973. Sorting and searching, Addison-Wesley Series in Computer Science and Information Processing.

[14] A. Marcus and G. Tardos, Excluded permutation matrices and the Stanley–Wilf conjecture, *J. Combin. Theory Ser. A* **107.1** (2004) 153–160.

[15] T. W. Pensyl and C. D. Savage. Rational lecture hall polytopes and inflated Eulerian polynomials, *The Ramanujan Journal* **31.1-2** (2013), 97–114.

[16] V. R. Pratt, Computing permutations with double-ended queues, parallel stacks and parallel queues, In STOC-73: Proceedings of the fifth annual ACM symposium on Theory of computing (New York, NY, USA, 1973), ACM Press, 268–277.

[17] N. J. A. Sloane, editor, *The On-Line Encyclopedia of Integer Sequences*, published electronically at `https://oeis.org`, 2022.

[18] R. Smith and V. Vatter, The enumeration of permutations sortable by pop stacks in parallel, *Inform. Process. Lett.* **109** (2009), 626–629.

[19] R. Tarjan, Sorting using networks of queues and stacks, *J. Assoc. Comput. Mach.* **19** (1972), 341–346.

[20] V. Vatter, Finding regular insertion encodings for permutation classes, *Journal of Symbolic Computation* **47.3** (2012), 259–265.

[21] I. Žliobaitė, Controlled permutations for testing adaptive classifiers, In International Conference on Discovery Science (Berlin, Heidelberg 2011), Springer, 365–379.