Sorting permutations with pile shuffle on queue-like and stack-like piles

Kyle B. Treleaven
ktreleav@alum.mit.edu

March 14, 2025

Abstract

Inspired by a common technique for shuffling a deck of cards on a table without riffling, we formalize the pile shuffle and investigate its capabilities as a sorting device. Our study is novel in that we consider pile shuffle in three variations: (1) using queue-like piles, (2) using stack-like piles, and (3) using a heterogeneous mixture of those two pile types, either given or decided during the shuffle by the dealer herself. We first characterize the sortable permutations of one or more sequential rounds of pile shuffle, under constraints on the number and types of piles to be used, and then derive formulas to obtain a sorting shuffle of a given permutation on the minimum number of piles, efficiently, whenever feasible. In the process, we present an interesting mathematical interpretation of any multi-round shuffle on fixed-type piles, as a single-round shuffle on a generally larger number of fixed-type "virtual piles"; we confirm that repetition augments the power of pile shuffle exponentially, in that m piles over T rounds enjoys the capacity of m^T piles in a single round. Finally, we motivate a forthcoming companion paper in which we prove that dealer choice—where the dealer is allowed to choose the types of the piles during the shuffle—makes deciding feasibility of sort NP-Hard for some variants of multi-round pile shuffle; proof is by a novel reduction from Boolean satisfiability (SAT).

Keywords. Pile shuffle, Sortable permutations, Computational complexity.

1 Introduction

Pile shuffle is a common method for shuffling a deck of cards on a table without riffling, and is more or less a non-mechanical version of the *shelf shuffle* [1] used in shuffling machines in casinos: A deck of cards is dealt into piles on an empty card table, and the resulting piles are stacked up in some order to form the new deck. The process may be repeated until the deck is "shuffled enough".

While the majority of the literature on card shuffling techniques is dedicated to randomization, in this paper we investigate the capabilities of pile shuffle as a sorting device. The study pertains to randomization, however, by recognizing that good mixing might be achieved most directly by combining a sorting device with a randomized labeler. We consider pile shuffle in three variations: (1) using queue-like piles, as when cards are flipped over as they are dealt, (2) using stack-like piles, as when they are not flipped over during the deal, and (3) using a heterogeneous mixture of those pile types, preferably decided during the deal by the dealer herself. As far as the author is aware, ours is the first study to consider arbitrary mixtures of the two pile types.

Pile shuffle is limited as a sorting device in that not every deck of cards can be sorted in a single "round" of shuffle on a bounded number of piles. As a trivial example, the only deck that can be sorted using a single queue is the already-sorted deck. Motivated by this observation, our focus is on (1) characterizing the sortable permutations of one or more sequential rounds of pile shuffle, under constraints on the number of rounds, and the number and types of piles to be used, and (2) obtaining efficient algorithms to compute a sorting shuffle of a given permutation whenever feasible.

Contributions: The contributions of this paper are as follows:

We formulate a mathematical model of pile shuffle, and present necessary and sufficient conditions for a single shuffle to sort an input permutation. Our formulation is more or less the same thing as P-partitions [1], but stated in a way most conducive to our analysis. From these conditions we obtain formulas for (1) the minimum number of piles required for sortation, and (2) transcription of a sorting shuffle on the minimum number of piles (a *minimal* sort). Our formulas are efficient in that they can be computed in a single scan of the input permutation, in time that is therefore linear in its length. In the homogeneous all-queues or all-stacks cases, the number of piles required is in terms of well-studied ascent and descent permutation statistics [2, 3] with deep connections to combinatorics. This allows us to reproduce basic results about the probability that a random permutation is sortable under given constraints, through a connection between permutation statistics and Eulerian numbers [4].

In the multi-round setting, we present a family of bijective embeddings mapping any sequence of shuffles on fixed pile types to an equivalent single-round shuffle on fixed-type "virtual piles". This mathematical reduction lifts nearly all of our results about single-round shuffle to the multi-round setting. We confirm that repetition augments the power of pile shuffle exponentially, in that m piles over T rounds has the capacity of m^T piles in a single round.

Finally, we expose non-trivial complexity in deciding sort feasibility, in the multi-round case, if the dealer is allowed to choose the types of the piles arbitrarily during the shuffle. (The single-round case remains tractable.) We present two such sort feasibility problems of general interest and motivate a forthcoming companion paper in which we demonstrate—by a novel reduction from Boolean satisfiability (SAT)—that at least one of those variants is NP-Hard.

1.1 Literature Review

Sorting is one of the fundamental problems in computer science [5], with deep connections to combinatorics and computational complexity theory. It has attracted significant research since the beginning of computing, no doubt because the theoretical properties of sorting algorithms have direct and significant impact on the performance of large-scale computer systems. The most popular branches of the literature tend to focus on the typical random-access model of computer memory, where data at a particular memory address may be mutated at unit cost. In such setting the most common objectives are to obtain space- and time- efficient sort algorithms, optimizing resource consumption.

Sorting problems based on the physical constraints of storage present other unique challenges, and are also of both practical and theoretical significance. In physically-motivated settings like pile shuffle, even asking whether a collection of items can be sorted at all (feasibility) can become an interesting question.

1.1.1 Patience sort

The present study is arguably closest to that of Patience sort [6, 7], in that it is similarly motivated by sorting in piles, rather than randomizing. Patience sort represents a greedy strategy for the so-called Floyd's Game, and happens to be an efficient means to compute longest increasing subsequences. Both Patience sort and sort by pile shuffle are so-called *distribution sorts* [5, p. 168], in that the sort occurs in a distribution phase, followed by a collection phase. However, Patience sort is a kind of *merge sort*—the piles are merged together into the output one card at a time—whereas pile shuffle sort is a so-called *bucket sort*—the piles are concatenated, each as a whole, in a chosen order.

While bucket sorts classically employ some kind of sub-sort within each bucket (sometimes recursively) that is not always physically practical. Instead, pile shuffle sort is a non- sub-sorted, a.k.a. *pure*, bucket sort. In lieu of sub-sorting, we rely on multiple rounds of shuffle to augment the power of a bounded number of piles, at the expense of more time to execute the shuffle. In the motivating case of sort with a physical facility (e.g., a table of a specific size) that is often a necessary trade-off.

1.1.2 Sorting on networks of stacks and queues

There is a significant body of literature about the sortable permutations of so-called sorting *networks* [2, 8, 9]. While the study was founded originally on networks of queues and stacks, a wide variety of more complex shuffle devices—including ones inspired by card shuffling—have been studied [10, 11]. Although bucket sorts like pile shuffle might also be cast in the network framework, the fit is not entirely natural, and the author is not aware of previous such treatments.

1.1.3 Pancake sort

Finally, the so-called "Pancake" sort [12] is another physically inspired problem, of sorting a disordered stack of pancakes by repeatedly inserting a spatula at some point in the stack and flipping all pancakes above it. Pancake sort appears in applications in parallel processor networks, and can provide an effective routing algorithm between processors [13, 14]. Pancake sort has also been called an "educational device", and it was shown to be NP-Hard in [15] by a reduction from 3SAT. In the forthcoming companion paper in the series, we prove NP-Hardness of multi-round heterogeneous variants of pile shuffle sort by reduction from SAT.

1.2 Organization

The rest of the paper is organized as follows. We define notation and summarize necessary background for the paper in Section 2. We formulate the pile shuffle mathematically and state the objectives of the paper in Section 3. We analyze our model and present results pertaining to a single round of pile shuffle—we cover shuffle on queues, on stacks, and on a mixture of pile types—in Sections 4, 5, and 7, respectively. (We briefly discuss sorting random permutations in Section 6.) In Section 8, we extend our single-round results to the case of multiple sequential rounds of pile shuffle. Then in Section 9, we expose the additional complexity of allowing the dealer to choose the types of piles in the multi-round setting, motivating a forthcoming paper on the topic. Finally, we summarize our results, and offer conclusions, in Section 10.

2 Background

In this section we introduce notation used throughout the paper, as well as necessary mathematical background about ordered sets and permutations.

2.1 Notation

We use the following notation throughout the paper.

Ranges. We will denote by $\mathbb{N}$ the natural numbers, and by $[n]$ the natural range $\{1, 2, \ldots, n\}$. We will denote by $[n] \pm k$ the shifted range $\{i \pm k : i \in [n]\}$; in particular, $[n] - 1 = \{0, 1, \ldots, n - 1\}$ is often useful.

Sequences. We will denote the set of all sequences of length n on a set S by $S^{[n]}$, or more lazily by S^n. In some instances we will denote sequences in string form: We may write $f = f_1 f_2 f_3$ to specify a sequence of three elements with $f(1) = f_1$, $f(2) = f_1$, $f(3) = f_3$. We use superscripting in string form to denote repetition. For example, if ABC is the partition of a sequence into three segments, then $AB^3C = ABBBC$.

Function composition. We use the notation $h = f(g)$ for the composition of functions $f : X \to Y$ and $g : Y \to Z$, i.e., $h(x) = f(g(x))$ for all $x \in X$.

Indicator expressions. In some equations we use a so-called indicator notation, where $[P]$ denotes the indicator function associated with a proposition P. That is, $[P] = 1$ if the proposition is true, otherwise $[P] = 0$. For example $f(x) = [x \geq 3]$ means that $f(x) = 1$ over $x \geq 3$ and 0 elsewhere.

Modular arithmetic. We use 2-modulo arithmetic sparingly, with notation $a \oplus b \doteq (a + b) \mod 2$.

2.2 Ordered sets

An *ordered set* is a pair $(X, \preceq)$ of a set X and a binary relation (the order) $\preceq$ that satisfies for all $a, b, c \in X$: $a \preceq a$ (reflexivity), $a \preceq b$ and $b \preceq a \implies a = b$ (antisymmetry), and $a \preceq b$ and $b \preceq c \implies a \preceq c$ (transitivity). If every two points are comparable, then $\preceq$ is a *total order*. Any order $\preceq$ can be equivalently expressed in terms of a strict order $\prec$ defined by $a \prec b \iff a \preceq b$ and $a \neq b$. We read $a \prec b$ as "a precedes b".

2.3 Permutations

A permutation of a finite set X is a bijective mapping $\sigma : X \to X$. It is well known that there are $n!$ (factorial) permutations of a set of size n. We will denote by $X!$ the set of all such permutations. The identity permutation is the special permutation σ_I where $\sigma_I(x) = x$ for all $x \in X$. A composition of two permutations is also a permutation. The inverse of a permutation σ, denoted σ^{-1}, is the permutation with the property $\sigma^{-1}(\sigma) = \sigma(\sigma^{-1}) = \sigma_I$.

3 Problem Statement

In this section we formalize the pile shuffle with a mathematical framework used throughout our study, and we state the objectives of the paper.

Pile shuffle begins with a deck of cards and an empty card table, and has two phases—distribution (or "the deal"), and collection: During the deal, until the deck is empty, we place the next card from the top of the deck onto the table, either directly on the table, creating a new *pile*, or on top of another pile previously created. During the collection phase, once the deck has been fully dealt out, we pick up each pile as a whole, one at a time in some order, and add it to the bottom of the new deck.

Deck representation: We assume a fixed assignment of labels $[n]$ to n distinct elements of a deck, so that any deck ordering can be represented by a permutation $\sigma \in [n]!$. In particular, we represent the order of a given deck by the permutation with the property that label s is in the $\sigma(s)$-th position for every $s \in [n]$. We call this the *embedding* convention, which is a departure from a perhaps more typical—and inverse—*sequence* convention, where $\sigma(k)$ would be the label in the k-th position. We use embedding notation throughout the study because of its property that s precedes t in the deck if and only if $\sigma(s) < \sigma(t)$. A deck is *sorted* if it is represented by the identity permutation σ_I.

Pile types: We consider piles of two types in this paper: queues and stacks. A queue maintains the order of cards placed on it. For example, if we deal the sequence 1234 into a queue, and pick it back up, we obtain 1234 again. On a card table, flipping the cards over during the deal creates this kind of behavior. If label s precedes t in the input deck, and they are placed into the same queue together, then s precedes t in the new deck also. In contrast, a stack reverses the order of placement: If s precedes t in the deck and they are placed into the same stack, then t precedes s in the new deck; if we deal 1234 into a stack, we obtain 4321 back. Dealing cards into a pile *without* flipping them over creates stack-like behavior.

Pile assignment: The outcome of a pile shuffle depends on (1) the input permutation, given, (2) the pile types used, (3) the assignment of cards onto piles during the deal, and (4) the order in which the piles are retrieved during collection. If we number the piles in the order that they are picked up, then the assignment of labels to piles can be described by a function $y : [n] \to \mathbb{N}$ of *pile assignments*, assigning each label $s \in [n]$ to the $y(s)$-th pile collected. Pile shuffle ensures that the new deck ς obeys $y(s) < y(t) \implies \varsigma(s) < \varsigma(t)$, for every pair of labels s and t; this is true regardless of the pile type(s) that are used.

Example: Suppose we shuffle a deck with label sequence $\sigma^{-1} = 456123$ using pile assignments $y = 421242$ on stacks. We can imagine dealing from a face-up deck of cards, and using y to determine which pile to place each card into. Note that y is a function of a card's label only, and ignores its position in the deck. First we place item $\sigma^{-1}(1) = 4$ into pile $y(4) = 2$, then item $\sigma^{-1}(2) = 5$ into pile $y(5) = 4$, and so on. After the deal we will see piles as below.

$$\begin{array}{cccc}
 & 2 & & \\
 & 6 & & 1 \\
3 & 4 & & 5 \\
\hline
P1 & P2 & P3 & P4
\end{array}$$

Note the number of non-empty piles used during shuffle is equal to the number of *distinct* pile assignments, in this case three. Collecting the piles in increasing order (left-to-right) we obtain the new deck ς; in this case $\varsigma^{-1} = 326415$.

Objectives: The present paper has two objectives. First, we seek to characterize the sortable permutations of one or more sequential rounds of pile shuffle, under constraints on the number of rounds and the number and types of piles to be used. A permutation is sortable if there exists a permissible pile shuffle—i.e., the pile assignments in one or more sequential rounds—that causes the deck to become sorted. Second, we seek efficient algorithms to compute a sort of an input permutation when feasible, or else identify that it is not sortable.

4 Sorting with a pile shuffle on queues

We first study pile shuffle on queues, or *queue shuffle*, which lays the foundation for the other cases. While some of the results of these early sections may be known generally, the techniques developed here are needed for later sections. We begin by analyzing our pile shuffle model to derive conditions for a single shuffle on queues to sort an input permutation. From those conditions we derive a simple lower bound on the number of queues required to sort a given permutation, in terms of well-known permutation statistics. Finally, we present a formula which constructs a minimal sort (i.e., a sort on the minimum number of piles) in time that is linear in the length of the permutation.

4.1 Analysis

Suppose that $\varsigma \in [n]!$ is the embedding representation of a deck resulting by shuffling a deck $\sigma \in [n]!$ on queues using pile assignments y. We will denote the relation

$$\textbf{q}\text{-shuffle}\, y\, \sigma = \varsigma.$$

We recall that for distinct labels $s, t \in [n]$, s precedes t in the output, i.e., $\varsigma(s) < \varsigma(t)$, either if its pile assignment is lower, $y(s) < y(t)$, or within the same pile if s precedes t in the original deck, $\sigma(s) < \sigma(t)$. We can express this mathematically by

$$\textbf{q}\text{-shuffle}\, y\, \sigma = \varsigma \quad \iff \quad \varsigma(s) < \varsigma(t) \iff \Big(y(s),\, \sigma(s)\Big) < \Big(y(t),\, \sigma(t)\Big). \tag{4.1}$$

(The final order is the lexicographical order.)

Relations like (4.1) would become increasingly tedious to carry around in the sequel, so we will introduce a more concise notation based on relations between *induced orders*:

If $f : X \to Y$ is an injective function to an ordered set $(Y, \prec)$, then we denote by $\prec_f$ the order induced on X by

$$x \prec_f x' \iff f(x) \prec f(x'). \tag{4.2}$$

When f is defined by an expression $f(x)$ we may write $\prec_f$ as $\operatorname{ord}_x f(x)$. Then if $(Y_1, \prec_1)$ and $(Y_2, \prec_2)$ are two ordered sets, and we have injections $f_1 : X \to Y_1$ and $f_2 : X \to Y_2$, then

$$\operatorname{ord}_x f_1(x) = \operatorname{ord}_x f_2(x) \quad \iff \quad f_1(x) \prec_1 f_1(x') \iff f_2(x) \prec_2 f_2(x'). \tag{4.3}$$

We read the the left-hand side as: f_1 and f_2 induce the same order on X.

We use this condition to express (4.1) as

$$\textbf{q}\text{-shuffle}\, y\, \sigma = \varsigma \quad \iff \quad \operatorname{ord}_s \varsigma(s) \;=\; \operatorname{ord}_s \Big(y(s),\, \sigma(s)\Big). \tag{4.4}$$

In words, a queue shuffle using pile assignments y transforms σ into ς if and only if ς induces the same order on $[n]$ as (y, σ) does lexicographically.

4.2 Sorting shuffles and feasibility

A pile shuffle is a sort if it produces as output the identity permutation, $\varsigma = \sigma_I$. When sort occurs, $\sigma_I(s) = s$ everywhere simplifies (4.4) to

$$\textbf{q}\text{-shuffle}\, y\, \sigma = \sigma_I \quad \iff \quad \operatorname{ord}_s s \;=\; \operatorname{ord}_s \Big(y(s), \sigma(s)\Big) \tag{4.5}$$

$$\iff \quad s < t \iff \Big(y(s), \sigma(s)\Big) < \Big(y(t), \sigma(t)\Big). \tag{4.6}$$

Our first result expresses this sort condition in an efficient form.

Lemma 4.1 (Sort with queues)**.** *A pile assignment function y is a sort of permutation $\sigma \in [n]!$ on queues ($\mathbf{q}$-shuffle $y\,\sigma = \sigma_I$) if and only if*

$$y(s+1) \geq y(s) + [\sigma(s+1) < \sigma(s)] \qquad \forall s \in [n-1]. \tag{4.7}$$

Note that, unlike (4.6), the condition (4.7) can be checked with a linear scan. An intuition behind the condition is as follows: A sort may never assign element $(s+1)$ to a lower-valued pile than s, i.e., one this is collected earlier in the collection phase. However, as long as s precedes $(s+1)$ during the deal, then $(s+1)$ may be placed in any pile $y(s+1) \geq y(s)$. Otherwise—mathematically, whenever $\sigma(s+1) < \sigma(s)$—$(s+1)$ must be placed into a *strictly* higher-valued pile, collected later, i.e., $y(s+1) \geq y(s) + 1$. For example, when dealing the sequence 1423, 2 could be dealt into the same queue as 1, since it is dealt after 1, but 4 must be dealt into a higher-valued queue than 3, since it precedes 3.

Proof. Due to transitivity, the right-hand side of (4.6) is true if and only if

$$\Big(y(s), \sigma(s)\Big) < \Big(y(s+1), \sigma(s+1)\Big) \qquad \forall s \in [n-1]. \tag{4.8}$$

Therefore, we may prove the lemma by equating (4.7) with (4.8). This can be done by cases: Wherever $\sigma(s) < \sigma(s+1)$, (4.8) is satisfied if and only if $y(s+1) \geq y(s)$; wherever $\sigma(s+1) < \sigma(s)$, if and only if $y(s+1)$ is *strictly* greater, i.e., $y(s+1) \geq y(s) + 1$. (4.7) is precisely the combined expression of these two cases using indicator notation. □

We note that (4.7) is equivalently stated:

1. y is monotonically non-decreasing; and
2. it is strictly increasing at every descent of σ.

A descent of a sequence x is a position t where $x(t+1) < x(t)$. The number of descents of a permutation,

$$\operatorname{desc}(\sigma) = \sum_{s=1}^{n-1} \left[\sigma(s+1) < \sigma(s)\right], \tag{4.9}$$

is a well-studied permutation statistic [2, p. 4]. It is one less than the number of ascending runs, $\operatorname{ascrun}(\sigma)$, where the ascending runs of a sequence are the contiguous increasing subsequences that cannot be extended on either end. The number of ascending runs of σ also corresponds to the number of "readings" $\operatorname{read}(\sigma^{-1})$ of the deck sequence σ^{-1}, which is the number of times one must scan through σ^{-1} to find the numbers $1, 2, \ldots, n$ in order without ever backtracking. For example the sequence 364152 has three readings: 12, 345, and 6.

Lemma 4.2. *If y is the pile assignments of a queue shuffle sort of permutation $\sigma \in [n]!$, where, without loss of generality, $y(1) = 1$, then the number of piles $y(n)$ satisfies*

$$y(n) \geq 1 + \operatorname{desc}(\sigma) = \operatorname{ascrun}(\sigma) = \operatorname{read}(\sigma^{-1}). \tag{4.10}$$

Proof. We obtain (4.10) by writing $y(n)$ as a telescoping sum starting from $y(1) = 1$, bounding the sum by rearranging (4.7), and then substituting (4.9) in the bound:

$$\begin{aligned} y(n) &= y(1) + \sum_{s=1}^{n-1} y(s+1) - y(s) \\ &\geq 1 + \sum_{s=1}^{n-1} [\sigma(s+1) < \sigma(s)] = 1 + \operatorname{desc}(\sigma)\,. \end{aligned}$$

□

Finally we obtain a minimal sort in the sense of fewest piles used.

Lemma 4.3. *Let y^* be defined by*

$$y^*(s) = 1 + \sum_{s'=1}^{s-1} [\sigma(s'+1) < \sigma(s')] \qquad \forall s \in [n]. \tag{4.11}$$

We call it the cumulative ascending runs function of the permutation σ. y^* is a minimal assignment function for sorting σ on queues.

Proof. It is easy to verify that the definition (4.11) of y^* satisfies (4.7) and that $y^*(n) = 1 + \operatorname{desc}(\sigma)$ by construction, matching the lower bound (4.10). □

Corollary 4.3.1. *A permutation σ can be sorted with m queues if and only if* $\operatorname{ascrun}(\sigma) \leq m$.

Proof. The corollary is an immediate result of the minimal construction (4.11) achieving the lower bound (4.10) in all instances. □

4.3 Demonstration

Lemma 4.1 provides a necessary and sufficient condition for a pile shuffle on queues to sort an input permutation. The condition is efficiently checkable in a linear scan of the input permutation. We have also derived the formula (4.10) for the minimum number of queues needed to sort an input permutation, in terms of the well-known permutation statistic of ascending runs. A companion formula (4.11) produces a sort if feasible on the minimum possible number of queues, in linear time.

We end this section with a demonstration of pile shuffle sort guided by Lemma 4.3. We start by writing an example permutation in the so-called two-line notation, a two-row matrix where the permutation pre-image is enumerated across the top row, and the matching image is written underneath it:

$$\left\{ \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 7 & 2 & 4 & 6 & 8 & 1 & 5 \end{array} \right\} = \left\{ \begin{array}{ccc} \dots & s & \dots \\ \dots & \sigma(s) & \dots \end{array} \right\}.$$

We use the set-notation brackets rather than the typical parentheses to emphasize that the column order does not matter. However, with the first row in the normal ascending order, the action of (4.11) is easily shown by adding y^* as a third row to the matrix

$$\left\{\begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 7 & 2 & 4 & 6 & 8 & 1 & 5 \\ 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 \end{matrix}\right\} = \left\{\begin{matrix} \dots & s & \dots \\ \dots & \sigma(s) & \dots \\ \dots & y^*(s) & \dots \end{matrix}\right\}.$$

It is easy to check that y^* increments at each of the descents of σ.

If we rearrange the columns so that the second row appears in ascending order, then the sequence representation of the deck appears in the top row,

$$\left\{\begin{matrix} 7 & 3 & 1 & 4 & 8 & 5 & 2 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 2 & 1 & 2 & 3 & 2 & 1 & 2 \end{matrix}\right\} = \left\{\begin{matrix} \dots & s & \dots \\ \dots & \sigma(s) & \dots \\ \dots & y^*(s) & \dots \end{matrix}\right\}.$$

(Note that the corresponding inverse permutation σ^{-1} can be obtained in this notation by exchanging the first two rows.) This column order supports the creation of a *shuffle tableau*: In the tableau below, each label s in a row $y^*(s)$ and column $\sigma(s)$ indicates that element s is placed into pile $y^*(s)$ as the $\sigma(s)$-th placement of the deal.

$y^*(s)$ \ $\sigma(s)$	1	2	3	4	5	6	7	8
1			1				2	
2		3		4		5		6
3	7				8			

The piles are collected in row order, top-to-bottom. Queue-like piles are collected left-to-right, whereas stack-like piles would be collected right-to-left. Collecting the contents of the tableau in this way demonstrates that y^* indeed sorts the input on three queues.

5 Sorting with a pile shuffle on stacks

Next we consider sorting with stacks, which requires only a minor modification of our analysis of shuffle on queues. We present companion results for pile shuffle on stacks to those just derived for queues.

Suppose that $\varsigma \in [n]!$ represents the result of shuffling a deck $\sigma \in [n]!$ on *stacks* using pile assignments y. We will denote the relation

$$\mathbf{s}\text{-shuffle}\, y\, \sigma = \varsigma.$$

A shuffle on stacks maintains the same order of the piles as a shuffle on queues, but within each pile labels are in the reverse order as if queues were used. Therefore, the version of (4.4) which holds for shuffling on stacks is

$$\mathbf{s}\text{-shuffle}\, y\, \sigma = \varsigma \qquad \Longleftrightarrow \qquad \operatorname{ord}_s \varsigma(s) = \operatorname{ord}_s \Big(y(s),\ -\sigma(s)\Big). \tag{5.1}$$

Somewhat intuitively then, our results when sorting with stacks are mirror images of those just developed for sorting with queues. For example, now each *ascent* of the input permutation requires a new stack for sort to occur. The proofs of the next results follow the same logic (mirrored) as their counterparts for queues, and we omit them as exercise(s) for the reader.

Lemma 5.1 (Sort with stacks)**.** *A pile assignment function y is a sort of permutation $\sigma \in [n]!$ on stacks ($\mathbf{s}\text{-shuffle}\, y\, \sigma = \sigma_I$) if and only if*

$$y(s+1) \geq y(s) + [\sigma(s+1) > \sigma(s)] \qquad \forall s \in [n-1]. \tag{5.2}$$

Lemma 5.2. *If y is the pile assignments of a stack shuffle sort of permutation $\sigma \in [n]!$, where $y(1) = 1$, then the number of piles $y(n)$ satisfies*

$$y(n) \geq 1 + \operatorname{ascs}(\sigma) = \operatorname{descrun}(\sigma)\,; \tag{5.3}$$

ascs *and* descrun *are the ascents and descending runs functions respectively; analogous to descents and ascending runs previously discussed.*

Lemma 5.3. *Let y^* be defined by*

$$y^*(s) = 1 + \sum_{s'=1}^{s-1} [\sigma(s'+1) > \sigma(s')] \qquad \forall s \in [n]. \tag{5.4}$$

We call it the cumulative descending runs function of the permutation σ. y^* is a minimal sort of σ on stacks.

Corollary 5.3.1. *A permutation σ can be sorted with m stack-like piles if and only if* $\operatorname{descrun}(\sigma) \leq m$.

6 Sorting random permutations in piles

In this brief section we consider the feasibility of sorting random permutations with pile shuffle. In particular, we examine what is the probability that m piles can sort a permutation σ_n sampled uniformly from $[n]!$.

According to Corollary 4.3.1 of the prequel, m queues can sort a permutation σ if and only if $m \geq 1 + \operatorname{desc}(\sigma)$; by Corollary 5.3.1, m stacks can sort it if $m \geq 1 + \operatorname{ascs}(\sigma)$. The number of permutations of length n with $k \geq 0$ ascents (descents) is known as the Eulerian number $\left\langle {n \atop k} \right\rangle$ [2, p. 4]. Eulerian numbers are a cornerstone of a deep and fascinating study of permutation statistics [4, 3, 16]. Therefore, with σ_n uniformly distributed over $[n]!$,

$$\mathbb{P}\left[\operatorname{ascs}(\sigma_n) = k\right] = \mathbb{P}\left[\operatorname{desc}(\sigma_n) = k\right] = \frac{1}{n!}\left\langle {n \atop k} \right\rangle.$$

It follows then that

$$\begin{aligned}\mathbb{P}\left[m \text{ queues (stacks) can sort } \sigma_n\right] &= \mathbb{P}\left[\operatorname{desc}(\sigma_n) \leq m-1\right] \\ &= \frac{1}{n!}\sum_{k=0}^{m-1} \left\langle {n \atop k} \right\rangle.\end{aligned}$$

For large n this probability can be well approximated, because it is known that Eulerian statistics follow a central limit theorem [17]. For example, the statistic

$$\frac{\operatorname{desc}(\sigma_n) - n/2}{\sqrt{n/12}}$$

converges in distribution to the Normal distribution, in the sense that

$$\sup_{x \in \mathbb{R}} \left| \mathbb{P}\left(\frac{\operatorname{desc}(\sigma_n) - n/2}{\sqrt{n/12}} \leq x \right) - \Phi(x) \right| \to 0;$$

here Φ denotes the standard Normal distribution function. That means both $\operatorname{desc}(\sigma_n)$ and $\operatorname{ascs}(\sigma_n)$ concentrate around $n/2$, which unfortunately means that for any given number m of piles, the probability of sorting σ_n vanishes as the permutation length n grows.

7 Sorting with a pile shuffle on queues and stacks

In the prequel, all piles within a given shuffle had the same type, either all queues or all stacks. Next we consider heterogeneous shuffle, where the piles may be of different types within a single deal. Once again we provide: (1) a mathematical shuffle relation, now on a heterogeneous mixture of queues and stacks, (2) necessary and sufficient conditions for heterogeneous shuffle to sort an input permutation, (3) a lower bound on the number of piles required for sort, and (4) the construction of a minimal sort whenever feasible in linear time.

7.1 Shuffle input-output relation

In the heterogeneous case, a shuffle is defined not only by the assignment y of labels into piles (as before), but additionally by an assignment χ of types to those piles. Using an alphabet $\mathcal{P} = \{(\mathbf{q})\text{ueue}, (\mathbf{s})\text{tack}\}$, if χ denotes the assignment of types to piles then

$$\chi(y) = \begin{cases} \mathbf{q}, & \text{pile } y \text{ is a queue} \\ \mathbf{s}, & \text{pile } y \text{ is a stack.} \end{cases}$$

Suppose that $\varsigma \in [n]!$ represents the result of shuffling a deck $\sigma \in [n]!$ using pile assignments y and type assignments χ. We denote the relation

$$\operatorname{shuffle} \chi\, y\, \sigma = \varsigma.$$

It is worth mentioning that the three modes of *shelf shuffle* studied in [1] map readily into this framework: One can verify that the *strict* mode of shelf shuffle on m shelves is the queues-only case $\chi = \mathbf{q}^m$, *standard* mode is the alternating types case $\chi = (\mathbf{sq})^m$, and *lazy* mode is the case $\chi = \mathbf{q}(\mathbf{sq})^m$.

Once again the piles are put into natural order, however within each pile order is determined by the pile type. We let κ denote the indicator function for the stack pile type, i.e.,

$$\kappa(y) = [\chi(y) = \mathbf{s}] = \begin{cases} 0 & \chi(y) = \mathbf{q} \\ 1 & \chi(y) = \mathbf{s}. \end{cases} \tag{7.1}$$

Then combining the models for each pile type—(4.4) and (5.1)—we obtain

$$\text{shuffle}\, \chi\, y\, \sigma = \varsigma \quad \Longleftrightarrow \quad \text{ord}_s\, \varsigma(s) \;=\; \text{ord}_s \Big(y(s),\, (-1)^{\kappa(y(s))} \sigma(s) \Big). \tag{7.2}$$

7.2 Sorting shuffles and feasibility

The condition for a heterogeneous shuffle to sort its input permutation follows the familiar pattern of Lemma 4.1:

Lemma 7.1 (Heterogeneous sort)**.** *Let $(\prec_{\mathbf{q}}, \prec_{\mathbf{s}}) = (<, >)$, i.e., $i \prec_{\mathbf{q}} j \iff i < j$, and $i \prec_{\mathbf{s}} j \iff i > j$. A heterogeneous shuffle (χ, y) sorts permutation $\sigma \in [n]!$ $(\text{shuffle}\, \chi\, y\, \sigma = \sigma_I)$ if and only if*

$$y(s+1) \geq y(s) + \big[\sigma(s+1) \prec_{\chi(y(s))} \sigma(s)\big] \qquad \forall s \in [n-1]. \tag{7.3}$$

Proof Sketch. The result is still proven by the same strategy as that of Lemma 4.1, but starting from (7.2) rather than (4.5). Each of the previous cases must now be considered separately when $\chi(y(s)) = \mathbf{q}$ and when $\chi(y(s)) = \mathbf{s}$. However, the additional complexity is minor, and we omit a full proof. □

Note (7.3) generalizes the forms of (4.7) and (5.2). Therefore, using essentially the same technique as before, we squeeze the inequality recurrence (7.3) to obtain bounds.

Lemma 7.2. *Given a pile shuffle (χ, y), starting w.l.g. from $y(1) = 1$, let y^* be defined by*

$$\begin{cases} y^*(1) = 1 \\ y^*(s+1) = y^*(s) + \big[\sigma(s+1) \prec_{\chi(y^*(s))} \sigma(s)\big] \quad s \in [n-1]. \end{cases} \tag{7.4}$$

If (χ, y) sorts the permutation $\sigma \in [n]!$, then $y(s) \geq y^*(s)$ for all $s \in [n]$.

Proof. The proof is by induction. The base case $y(1) \geq y^*(1) = 1$ is given. We may assume as the inductive hypothesis that for a given $s \in [n-1]$, $y(s) \geq y^*(s)$. If $y(s) = y^*(s)$, then we obtain $y(s+1) \geq y^*(s+1)$ immediately, since the right-hand side of (7.3) becomes $y^*(s+1)$ by definition. Otherwise, $y(s) \geq y^*(s)+1$, which we bookend with two uniform bounds, $y(s+1) \geq y(s)$ and $y^*(s) + 1 \geq y^*(s+1)$, to obtain the result. □

Corollary 7.2.1. *y^* is a minimal sort with respect to χ: For any sort (χ, y) of a permutation σ, the number of piles is $y(n) \geq y^*(n)$.*

Proof. y^* is a sort because (7.4) satisfies (7.3) everywhere, and the number $y^*(n)$ of piles used is the minimum by Lemma 7.2. $\square$

Lemma 7.2 and its corollary are fundamental results of this paper, and will be relied upon throughout the investigations of the sequel. They reveal the pivotal role that the type assignments χ plays—it fully specifies a minimal shuffle y^*—in deciding whether sort on a sequence of piles is feasible. If (7.4) has no solution—i.e., if χ is a finite sequence and too short—then there can be no solution to (7.3) either. Conversely, if a solution exists, then (7.4) obtains one efficiently in a linear scan.

Definition 7.1. *We say a type schedule χ (an assignment of pile types) sorts a permutation σ if there exists y such that (χ, y) sorts σ.*

7.3 Dealer's choice pile shuffle sort

If the dealer is free to choose the type of each of m piles arbitrarily during the deal, then a given deck may be sorted if and only if there is a type assignment χ among the 2^m elements of $\mathcal{P}^m$ which sorts it. In principle one could decide feasibility by checking each potential assignment until they find one for which (7.4) has a solution. However, while each check can be done in linear time, the total number of them is exponential in the number of piles. That could be problematic in practice since, e.g., sorting a random permutation of length n typically requires a non-trivial fraction of n piles.

Fortunately, a minimal sort can be obtained, if one exists, in time that is linear in the permutation length, by combining (7.4) with a greedy solution for choosing pile types χ^*: If label s begins a new pile, then we should choose the new pile's type so the next item (s+1) may be placed there also. Mathematically,

$$s = 1 \text{ or } y^*(s) > y^*(s-1) \implies \chi^*(y^*(s)) = \begin{cases} \mathbf{q} & \sigma(s+1) > \sigma(s) \\ \mathbf{s} & \sigma(s+1) < \sigma(s). \end{cases} \tag{7.5}$$

Doing otherwise cannot be advantageous, as we demonstrate next.

Lemma 7.3. *The pile shuffle (χ^*, y^*) defined by (7.4) and (7.5) is a minimal sort of input permutation σ: For any sort (χ, y) with $y(1) = 1$, $y(n) \geq y^*(n)$.*

Proof. Notably, the recurrence equations can be applied in the forward direction to refine any initial sort, i.e., solution (χ, y) to (7.3). For each position $s \in [n]$ in order, we do the following: First set $y(s)$ to $y^*(s)$, potentially reducing it—$y(s) \geq y^*(s)$ by Lemma 7.2. Next, modify χ at $y^*(s)$ if needed so that (7.5) holds there. Neither operation can cause (7.3) to become violated, so the pair (χ, y) remains a valid sort after each iteration of this process. Meanwhile, the number of piles $y(n)$ can never increase.

Since the pair of equations fully specifies (χ^*, y^*), then by the end of the procedure we obtain that solution regardless of starting sort. In particular, since we obtain the result starting from any minimal sort—and without introducing additional piles—then (χ^*, y^*) is itself minimal. $\square$

Other formulations of the Dealer's choice pile shuffle are potentially of interest. In particular, it could be natural to incorporate separate capacities, i.e. inequality constraints, on the numbers of queues, stacks, and total piles available for shuffle. Other formulations do not yield as readily to the greedy strategy for arbitrary choice, and this paper leaves such investigations for future work.

7.4 Reducing arbitrary shuffle to sort

Our discussion so far has pertained exclusively to the use of pile shuffle to sort a deck. However, any desired shuffle can be framed as a sort, through a transformation that is the subject of this brief section.

Inspired by the form of (4.2), if an order $\prec'$ on X is induced by

$$\sigma(x) \prec' \sigma(x') \iff f(x) \prec f(x'),$$

for some permutation $\sigma \in X!$, then let us express $\prec' = \operatorname{ord}_{\sigma(x)} f(x)$. (Speaking loosely, f induces the order on $\sigma(X)$ rather than on X directly.)

Lemma 7.4 (Domain invariance)**.** *For all $\sigma, \sigma' \in [n]!$,*

$$\operatorname{ord}_{\sigma(x)} f_1(x) = \operatorname{ord}_{\sigma(x)} f_2(x) \qquad \iff \qquad \operatorname{ord}_{\sigma'(x)} f_1(x) = \operatorname{ord}_{\sigma'(x)} f_2(x). \quad (7.6)$$

This technical lemma is simply to demonstrate that permuting the underlying domain of an induced-order equality does not actually change it.

Proof. (4.3) generalizes easily to

$$\operatorname{ord}_{\sigma(x)} f_1(x) = \operatorname{ord}_{\sigma(x)} f_2(x) \qquad \iff \qquad f_1(x) \prec_1 f_1(x') \iff f_2(x) \prec_2 f_2(x'), \quad (7.7)$$

for all $\sigma \in [n]!$, which obtains the lemma by the transitive property, as the right-hand side does not include σ at all. $\square$

Lemma 7.5. *If* $\operatorname{shuffle} \chi\, y\, \sigma = \varsigma$*, then* $\operatorname{shuffle} \chi\ (y(\sigma'))\ (\sigma(\sigma')) = \varsigma(\sigma')$ *for any third permutation $\sigma' \in [n]!$.*

Proof. The lemma is obtained by a change of variables $s = \sigma'(s')$ in the right-hand side of (7.2), while (7.6) allows us to freeze the domain subscripts:

$$\begin{aligned}
\operatorname{shuffle} \chi\, y\, \sigma = \varsigma \iff \quad & \operatorname{ord}_s \varsigma(s) \;=\; \operatorname{ord}_s \Big(y\big(s\big), (-1)^{\kappa(y(s))}\sigma(s)\Big) \\
\iff \quad & \operatorname{ord}_s \varsigma(\sigma'(s')) \;=\; \operatorname{ord}_s \Big(y\left(\sigma'(s')\right), (-1)^{\kappa\left(y\left(\sigma'(s')\right)\right)}\sigma\left(\sigma'(s')\right)\Big) \\
\iff \quad & \operatorname{shuffle} \chi\, \big(y(\sigma')\big)\, \big(\sigma(\sigma')\big) = \varsigma(\sigma').
\end{aligned}$$

$\square$

Corollary 7.5.1. $\operatorname{shuffle} \chi\, y\, \sigma = \varsigma$ *if and only if* $\big(\chi, y(\varsigma^{-1})\big)$ *is a pile shuffle sort of* $\sigma(\varsigma^{-1})$.

Proof. This is the special case of Lemma 7.5, with $\sigma' = \varsigma^{-1}$. $\square$

8 Sorting in multiple rounds of pile shuffle

In Section 6 we discovered that to sort a random permutation of length n, the number of piles required concentrates around $n/2$ in the all-queues or all-stacks case. (In the dealer choice case it seems to concentrate to a smaller but still constant fraction of n.) Therefore, for any given capacity m of piles, our chance of sorting a permutation in a single shuffle vanishes as the length n grows.

In the sequel we consider the problem of sorting in multiple rounds of shuffle, which augments the power of a bounded number of piles at the expense of more time to execute the sort. In the motivating case of sort with a physical facility (e.g., a table of a specific size) that is often a necessary trade-off.

8.1 Modeling

A multi-round pile shuffle is simply a sequence of basic shuffles where the output of one round becomes the input of the next one. A shuffle in T rounds can be modeled by a pair $(X, \mathbf{y})$ of a sequence of pile type assignments $X = (\chi_1, \ldots, \chi_T)$ and a sequence of pile assignments $\mathbf{y} = (y_1, \ldots, y_T)$; these induce a sequence of permutations (deck states) according to the recurrence

$$\begin{cases} \sigma_0 = \sigma, \\ \sigma_t = \operatorname{shuffle} \chi_t \, y_t \, \sigma_{t-1} \quad \text{for } 1 \le t \le T, \\ \varsigma = \sigma_T; \end{cases}$$

$\varsigma \in [n]!$ is the final deck order resulting by shuffling a deck starting in order $\sigma \in [n]!$ in this way. We denote the relation

$$\operatorname{shuffle} X \, \mathbf{y} \, \sigma = \varsigma.$$

Definition 8.1. *We say a multi-round assignment X of pile types sorts a permutation σ if there exists $\mathbf{y}$ such that $(X, \mathbf{y})$ sorts σ.*

8.2 Sorting with multiple pile shuffles on queues

We start by analyzing multi-round shuffle with only queues, i.e.,

$$X = (\mathbf{q}^{m_1}, \ldots, \mathbf{q}^{m_T}),$$

where m_t denotes the number of queues available in each round $t \in [T]$. We do not assume the number of queues is the same in each round.

We observe that if labels i and j are placed in the last round T so that $y_T(i) < y_T(j)$, then $\varsigma(i) < \varsigma(j)$. If they are placed in the same final pile $y_T(i) = y_T(j)$, but in the previous round they are placed in different piles so that $y_{T-1}(i) < y_{T-1}(j)$, then $\varsigma(i) < \varsigma(j)$. Continuing in this fashion it is easy to reason that

$$\operatorname{ord}_s \varsigma(s) = \operatorname{ord}_s \Big(y_T(s), y_{T-1}(s), \ldots, y_1(s), \, \sigma(s) \Big). \tag{8.1}$$

The final order component σ captures the fact that if two elements s and t are placed in all the same piles, they retain their original order after shuffle.

The sequence $(y_T, \dots, y_1)$ can be viewed as a mixed-radix digital representation of a number $\hat{y}_1$ given by a bijective embedding with recurrence

$$\begin{cases} \hat{y}_T = y_T, \\ \hat{y}_t = y_t + m_t \hat{y}_{t+1} \quad 1 \le t \le T-1. \end{cases} \tag{8.2}$$

This numerical embedding has the property that

$$\operatorname{ord}_s \Big(y_T(s), y_{T-1}(s), \dots, y_1(s),\, \sigma(s) \Big) = \operatorname{ord}_s \Big(\hat{y}_1(s),\, \sigma(s) \Big). \tag{8.3}$$

The right-hand side is recognizably the order expression (4.4) of a single queue shuffle with $\hat{y}_1$ as pile assignments. In this way the embedding $\hat{y}_1$ can be thought of as assigning the cards of the deck among a set of "virtual piles" in a corresponding single-round queue shuffle scenario. Letting

$$m_X := \prod_{t=1}^{T} m_t,$$

(8.2) defines $\hat{y}_1$ as into a co-domain $[m_X] - 1$ of m_X virtual piles. (On a strictly technical note, we enumerate multi-round and virtual pile assignments from $y(1) = 0$ instead of $y(1) = 1$, to simplify digital representation.)

Lemma 8.1. *A multi-round shuffle on queues with pile assignments* $\mathbf{y} = (y_t)_{t=1}^T$ *sorts permutation* σ *if and only if the corresponding single-round virtual pile assignments* $\hat{y}_1$ (8.2) *sorts* σ *on queues.*

Proof. Combining (8.1) with (8.3) obtains $\operatorname{ord}_s \varsigma(s) = \operatorname{ord}_s \Big(\hat{y}_1(s),\, \sigma(s) \Big)$. Subsequently, (4.4) demonstrates the equivalence to single-round shuffle using $\hat{y}_1$. □

Lemma 8.2. *A multi-round queue shuffle* $X = (\mathbf{q}^{m_1}, \dots, \mathbf{q}^{m_T})$ *can sort a permutation* σ *(in the sense of Definition 8.1) if and only if* $\operatorname{ascrun}(\sigma) \le \prod_t m_t$.

Proof. By Lemma 8.1, every shuffle $\mathbf{y}$ on X corresponds to a shuffle $\hat{y}_1$ (8.2) on $\le m_X$ queues; and vice versa. The result then follows from Corollary 4.3.1. □

Corollary 8.2.1. *A permutation* σ *can be sorted by* m *queues in* $T \ge 0$ *rounds if and only if* $\operatorname{ascrun}(\sigma) \le m^T$.

Proof. The result is a special case of Lemma 8.2. □

Note that, because (8.2) is reversible, any sorting schedule $\hat{y}_1$ on virtual queues—e.g., one obtained via (4.11)—can be transformed readily into a corresponding multi-round sort $\mathbf{y}$ on X.

8.3 Sorting with multiple pile shuffles on queues and stacks

When we sort with a mixture of queues and stacks the order logic is more convoluted. Now if i and j are in the same piles in the last k rounds, the final order of i and j depends not only on σ_{T-k}, but also on the types of those subsequent pile assignments: As we have observed in the prequel, each round in which two labels are assigned to the same stack reverses their order, whereas assignment to the same queue preserves their order. Fortunately these order dynamics remain compatible with a virtual piles interpretation albeit via a somewhat more convoluted embedding,

$$\operatorname{ord}_s \varsigma(s) = \operatorname{ord}_s \Big(\tilde{y}_T(s), \ldots, \tilde{y}_1(s),\, \tilde{y}_0(s)\Big) \tag{8.4}$$

$$= \operatorname{ord}_s \Big(\hat{y}_1(s),\, (-1)^{\hat{\kappa}_1(\hat{y}_1(s))}\, \sigma(s)\Big)\,; \tag{8.5}$$

with order components $(\tilde{y}_t)_{t=0}^T$ to be derived presently, along with $\hat{\kappa}_1$ and a generalized formulation of $\hat{y}_1$.

Letting κ_t denote the stack indicator function (7.1) for the pile types χ_t in each round $t \in [T]$, then the number of stacks that a label s is assigned to in rounds t through T is given by $\sum_{t \le t' \le T} \kappa_{t'}(y_{t'}(s))$. We can denote the parity of that count (zero or one) by

$$\tilde{\kappa}_t(s) \doteq \left[\sum_{t \le t' \le T} \kappa_{t'}(y_{t'}(s))\right] \mod 2. \tag{8.6}$$

Then based on our discussion above, the order component $\tilde{y}_t$ is governed by both y_t and $\tilde{\kappa}_{t+1}$: Suppose $y_t(i) < y_t(j)$ in round $t \le T$, and i and j are co-assigned in all future rounds. (If they are in the same piles in all the rounds, we can interpret the input permutation σ as pile assignments y_0 in an imaginary $t = 0$ round.) If i and j face an even number of stack assignments in the suffix ($\tilde{\kappa}_{t+1}(i) = \tilde{\kappa}_{t+1}(j) = 0$) then $\varsigma(i) < \varsigma(j)$; otherwise, they face an odd number of stack assignments ($\tilde{\kappa}_{t+1}(i) = \tilde{\kappa}_{t+1}(j) = 1$) and $\varsigma(i) > \varsigma(j)$. (Note that for the $t = T$ case, $\tilde{\kappa}_{T+1} = 0$.) This behavior is properly captured provided

$$\operatorname{ord}_s \tilde{y}_t(s) = \operatorname{ord}_s (-1)^{\tilde{\kappa}_{t+1}(s)}\, y_t(s) \qquad 0 \le t \le T, \tag{8.7}$$

(8.7) ensures, in the previous scenario, that i and j are indistinguishable according to the order prefix $(\tilde{y}_T, \ldots, \tilde{y}_{t+1})$, and that $\varsigma(i) < \varsigma(j) \iff \tilde{y}_t(i) < \tilde{y}_t(j)$. Something to bear in mind is that two labels with the same order component in a given round may be located in two different *actual* piles of potentially differing type, i.e. $\tilde{y}_t(i) = \tilde{y}_t(j) \not\Rightarrow y_t(i) = y_t(j)$.

Any choice of $(\tilde{y}_t)_{t=0}^T$ satisfying (8.7) characterizes sequential shuffle through (8.4). However, to obtain a virtual piles embedding of those order components as in the queues-only case, i.e., $\hat{y}_1$ satisfying (8.5) and given by

$$\begin{cases} \hat{y}_T = \tilde{y}_T, \\ \hat{y}_t = \tilde{y}_t + m_t \hat{y}_{t+1}, \quad 1 \le t \le T-1, \end{cases} \tag{8.8}$$

each $\tilde{y}_t$ must take value on the range $[m_t]-1$. To accomplish that, we introduce the interval-reversing function $\mathrm{rev}_m(y) = -y+m-1$; it maps the interval $[m]-1$ to itself in reverse order. Letting rev_m^n denote the composition of rev_m n times, it is easy to check rev_m^{2k} is identity for all $k \geq 0$, and $\mathrm{rev}_m^{2k+1} = \mathrm{rev}_m$; and therefore that for any y and $n \geq 0$,

$$\mathrm{ord}_s \, \mathrm{rev}_m^n(y(s)) = \mathrm{ord}_s \, (-1)^n \, y(s).$$

Therefore, defining

$$\tilde{y}_t(s) = \mathrm{rev}_{m_t}^{\tilde{\kappa}_{t+1}(s)}(y_t(s)) \tag{8.9}$$

satisfies our order requirements (8.7). The first order component is simply the last round pile assignment, $\tilde{y}_T = y_T$, since there are no future rounds to cause reversals; again, $\tilde{\kappa}_{T+1} = 0$. Now (8.8) is bijective, and defines $\hat{y}_t$ for each $t \in [T]$ as into a co-domain $[\hat{m}_t] - 1$, where

$$\hat{m}_t \doteq \prod_{t'=t}^{T} m_{t'}.$$

(Note, $\hat{m}_1 = m_X$.)

For the final component of our virtual piles interpretation of mixed multi-round shuffle, we introduce the new set of indicator functions $\{\hat{\kappa}_t\}$, defined by

$$\hat{\kappa}_t(\hat{y}_t) = \tilde{\kappa}_t \qquad 1 \leq t \leq T. \tag{8.10}$$

We are motivated to do so by the observation—comparing (8.5) to (7.2)—that $\hat{\kappa}_1$ can be interpreted as the indicator function (again in the sense of (7.1)) associated with some assignment $\hat{\chi}_1$ of types to the virtual piles; then:

Proposition 8.3. *A heterogeneous multi-round shuffle $(X, \mathbf{y})$ sorts permutation σ if and only if the corresponding single-round heterogeneous virtual shuffle $(\hat{\chi}_1, \hat{y}_1)$ sorts σ.*

Remarkably, we can fully precompute $\hat{\kappa}_1$—and therefore virtual types $\hat{\chi}_1$—given only the sequence X of type assignments. (We derive the equations at the end of this section.) Therefore, Prop. 8.3 brings to bear the full power of Lemma 7.2, generalizing its role in deciding sort feasibility to the multi-round case through our virtual piles interpretation. Moreover, a minimal sort on X can be computed again in linear time if feasible: First, we produce $\hat{\kappa}_1$ (up to the length n of the permutation) and obtain $\hat{y}_1^*$ if feasible using the single-round solution (7.4). We can then transform $\hat{y}_1^*$ into a multi-round assignment $\mathbf{y}^*$ by reversing (8.8) followed by (8.9). The whole process can be done in $O(n)$ time.

Precomputing $\hat{\kappa}_1$: To end the section we derive the equations to compute $\hat{\kappa}_1$ (and thereby $\hat{\chi}_1$) given the sequence of pile type assignments X:

To begin, we can write (8.6) as a recurrence

$$\begin{cases} \tilde{\kappa}_T = \kappa_T(y_T) \\ \tilde{\kappa}_t = \kappa_t(y_t) \oplus \tilde{\kappa}_{t+1}, \quad 1 \leq t \leq T-1. \end{cases} \tag{8.11}$$

Then, by substituting (8.10) into (8.11)—and given $\hat{y}_T = \tilde{y}_T = y_T$—we obtain

$$\begin{cases} \hat{\kappa}_T = \kappa_T \\ \hat{\kappa}_t(\hat{y}_t) = \kappa_t(y_t) \oplus \hat{\kappa}_{t+1}(\hat{y}_{t+1}), \quad 1 \le t \le T-1. \end{cases} \tag{8.12}$$

Finally we substitute expressions for $\hat{y}_t$ and y_t in (8.12): Namely, (8.8) provides $\hat{y}_t = \tilde{y}_t + m_t\hat{y}_{t+1}$, and (8.9) can be reversed to obtain $y_t = \mathrm{rev}_{m_t}^{\hat{\kappa}_{t+1}(\hat{y}_{t+1})}(\tilde{y}_t)$, for each of $1 \le t \le T-1$. Substituting these we obtain

$$\hat{\kappa}_t(\tilde{y}_t + m_t\hat{y}_{t+1}) = \kappa_t\left(\mathrm{rev}_{m_t}^{\hat{\kappa}_{t+1}(\hat{y}_{t+1})}(\tilde{y}_t)\right) \oplus \hat{\kappa}_{t+1}(\hat{y}_{t+1}), \qquad 1 \le t \le T-1. \tag{8.13}$$

Given that $\hat{y}_{t+1}$ and $\tilde{y}_t$ vary over $[\hat{m}_{t+1}] - 1$ and $[m_t] - 1$, respectively, (8.13) indeed defines $\hat{\kappa}_t$ for each $t \in [T-1]$—purely in terms of κ_t and $\hat{\kappa}_{t+1}$—over the whole co-domain $[\hat{m}_t] - 1$ of $\hat{y}_t$.

8.4 Sorting with multiple pile shuffles on stacks

In this brief section we analyze multi-round shuffle with only stacks, i.e.,

$$X = (\mathbf{s}^{m_1}, \dots, \mathbf{s}^{m_T}),$$

where m_t denotes the number of stacks available in each round $t \in [T]$.

In this special case of the general sort, for each $t \in [T]$, $\kappa_t = 1$ uniformly over all of $[m_t] - 1$. This reduces the recurrence of (8.12) considerably to $\hat{\kappa}_t(\hat{y}_t) = 1 \oplus \hat{\kappa}_{t+1}(\hat{y}_{t+1})$, from which we obtain $\hat{\kappa}_t = (1 + T - t) \mod 2$ uniformly over $[\hat{m}_t] - 1$; in particular, $\hat{\kappa}_1 = T \mod 2$. Therefore, if the number of rounds T is even, then X is equivalent to non-sequential pile shuffle on m_X virtual queues ($\mathbf{q}^{m_X}$); conversely, if the number of rounds T is odd, then it is equivalent to non-sequential pile shuffle on as many virtual stacks ($\mathbf{s}^{m_X}$).

Lemma 8.4. *A multi-round stack shuffle $X = (\mathbf{s}^{m_1}, \dots, \mathbf{s}^{m_T})$ can sort a permutation σ (in the sense of Definition 8.1) if and only if either T is even and* $\mathrm{ascrun}(\sigma) \le \prod_t m_t$ *or T is odd and* $\mathrm{descrun}(\sigma) \le \prod_t m_t$.

Proof. When the number of rounds is even multi-round stack sort feasibility is governed by the condition of Corollary 4.3.1 on the number of the permutation's ascending runs. When the number of rounds is odd, feasibility is governed by Corollary 5.3.1 instead, and the number of descending runs. □

9 Sorting in multiple rounds of Dealer's choice pile shuffle

We finalize the present study of multi-round pile shuffle by re-introducing dealer choice in the selection of pile types. In Section 7.3 we showed that sort feasibility remains tractable with dealer choice in the single-round case: A minimizing assignment of pile types χ^* can be obtained greedily, i.e. by (7.5), from a set whose size is exponential in the number of piles. Meanwhile feasibility remains tractable in the multi-round setting on *fixed* pile types, namely by the reduction to single-round shuffle of Prop. 8.3.

In this section—and in the forthcoming companion paper—we are motivated by two problems of general interest:

Problem 9.1 (Repeated-round Dealer's choice pile shuffle sort)**.** *Decide whether a permutation σ can be sorted in T rounds of Dealer's choice shuffle on m piles; the permutation σ and parameters T and m are the instance data.*

Problem 9.1 captures the motivating scenario of sort in bounded time, in physical piles, on a surface whose shape does not typically change from round to round.

Problem 9.2 (Variable-round Dealer's choice pile shuffle sort)**.** *Given per-round pile capacities $(m_1, \ldots, m_T)$, decide whether a permutation σ can be sorted in T rounds of Dealer's choice shuffle, where for each $t \in [T]$, m_t is the number of piles in round t; i.e., whether there exists a schedule $(X, \mathbf{y})$ sorting σ, such that $X = (\chi_1, \ldots, \chi_T)$, with $\chi_t \in \mathcal{P}^{m_t}$ for each $t \in [T]$.*

Problem 9.2 generalizes Problem 9.1—the former contains all the instances of the latter—in that each round may have different pile capacity.

Unfortunately, dealer choice brings non-trivial complexity to the multi-round setting. For example, in the case of Problem 9.2, while it is tempting to try and obtain a solution from a minimal sort on $\prod_t m_t$ virtual piles of dealer-chosen types—by invoking Prop. 8.3—that approach is not sound. That is because in general not every one of the virtual pile types may be chosen independently, since there are only a generally smaller number $\sum_t m_t$ of *actual* piles.

In fact, in the aforementioned companion paper we will demonstrate by reductions from the Boolean satisfiability problem (SAT) that multi-round Dealer's choice pile shuffle sort (feasibility) can be NP-Hard in general. In particular, the main result of the forthcoming paper is the following proposition:

Proposition 9.1. *The variable-round Dealer's choice pile shuffle sort problem is NP-Hard.*

Unfortunately our study leaves as an open question whether repeated-round sort feasibility is NP-Hard. However, the author's conjecture, which we discuss in the second paper, is that it is.

Conjecture 9.2. *The repeated-round Dealer's choice pile shuffle sort problem is NP-Hard.*

Sort feasibility is generally no harder than NP since the assignment X of all (real) pile types is a checkable certificate of feasibility.

9.1 Multi-round mixed sort with a fixed ***number*** of piles

Although we are conjecturing that the repeated-round Prob. 9.2 is NP-Hard, if we restrict it to an a priori fixed number m of piles, then a brute force search over all possible assignments decides feasibility in (technically) polynomial time: In that case, if there are greater than $T' = \lceil \log_m(n) \rceil$ rounds, then a permutation of length n is trivially sortable with $m^{T'} \geq n$ virtual piles. Otherwise, we need check only as many as $(2^m)^{T'} \in O\left(n^{(m/\log_2 m)}\right)$ certificates, each in $O(n)$ time. Note, while this approach is technically polynomial-time in the permutation length, it suffers exponential growth in the number of piles m. This is reminiscent of the way SAT is technically polynomial time on any fixed set of variables, even though it is NP-Hard in general.

10 Conclusion

In this paper we have formalized the pile shuffle, and analyzed its capabilities as a sorting device. We have characterized the sortable permutations of one or more sequential rounds of pile shuffle in three variations: (1) on queues, (2) on stacks, and (3) on a heterogeneous mixture of both piles types. As far as the author is aware, ours is the first formal study to consider arbitrary mixtures of the two pile types.

We formulated the pile shuffle as a parameterized relation between two permutations. The formulation characterizes pile shuffle as a set of functions which, when paired with permutations representing the starting states of a deck of cards, inject those cards into an ordered set reflecting the same label order as the deck state after shuffle. Our characterization is more or less the same thing as P-partitions [1], but stated in a way most conducive to our analysis.

The first part of the paper focused on the single-round case. Analyzing our model, we developed necessary and sufficient conditions (7.3) for a given shuffle to sort an input permutation. Notably, these conditions can be computed efficiently in a single scan of the permutation. From these we derived formulas, in each case, for the minimum number of piles required for sort, and to transcribe a sort on the minimum number of piles. In the homogeneous all-queues or all-stacks cases, the bounds are in terms of well-studied ascent and descent permutation statistics with deep connections to combinatorics. Finally, in the case that the dealer is allowed to choose the types of the piles arbitrarily during the shuffle, we augmented the scan for a minimal sort on fixed pile types (7.4) with a greedy method for choosing those types (7.5), and demonstrated that it obtains a minimal sort in the presence of dealer choice.

The second key contribution of the paper is a mathematical reduction of multi-round shuffle on fixed pile types, into single-round shuffle on fixed-type "virtual piles". The reduction lifts all of our results about fixed-type single-round pile shuffle to the multi-round setting. It has the fundamental property that the number of piles available for the virtual shuffle is equal to the product of the number of piles available in each round of the actual shuffle. Therefore, we confirm that repetition augments the power of pile shuffle exponentially, in that m piles over T rounds has the capacity of m^T piles in a single round. Feasibility of multi-round shuffle sort on fixed pile types remains decidable in linear time, because the formula of our reduction (8.13) can be truncated to the number of piles used, which should not be more than the size of the permutation.

In a sudden halt to the momentum of our study, we discovered that dealer choice brings non-trivial complexity to sort feasibility in the multi-round setting. We introduced two sort feasibility problems of multi-round Dealer's choice pile shuffle, and motivated a forthcoming companion paper in which we demonstrate, by a novel reduction from Boolean satisfiability (SAT), that at least one of those variants is NP-Hard.

11 References

[1] Jason Fulman and T. Kyle Petersen. Card shuffling and p-partitions. *Discrete Mathematics*, 344(8):112448, 2021.

[2] Miklos Bona. *Combinatorics of Permutations.* Chapman and Hall/CRC, Boca Raton, Fla., 1st edition edition, June 2004.

[3] Frederick Butler. Stirling-euler-mahonian triples of permutation statistics. *arXiv e-prints*, pages arXiv–2305, 2023.

[4] T Kyle Petersen. Eulerian numbers. *Birkhäuser Advanced Texts Basler Lehrbücher*, 2015.

[5] Donald E. Knuth. *The art of computer programming, volume 3: (2nd ed.) sorting and searching.* Addison Wesley Longman Publishing Co., Inc., USA, 1998.

[6] Badrish Chandramouli and Jonathan Goldstein. Patience is a virtue: Revisiting merge and sort on modern processors. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 731–742, 2014.

[7] Alexander Burstein and Isaiah Lankham. Combinatorics of patience sorting piles. *Séminaire Lotharingien de Combinatoire*, 54:B54Ab, 2006.

[8] Robert Tarjan. Sorting using networks of queues and stacks. *Journal of the ACM (JACM)*, 19(2):341–346, April 1972.

[9] Stefan Felsner and Martin Pergel. The complexity of sorting with networks of stacks and queues. In *Algorithms-ESA 2008: 16th Annual European Symposium, Karlsruhe, Germany, September 15-17, 2008. Proceedings 16*, pages 417–429. Springer, 2008.

[10] Lara Pudwell and Rebecca Smith. Sorting Permutations Via Shuffles. *SSRN Electronic Journal*, 2023.

[11] Stoyan Dimitrov. Sorting by shuffling methods and a queue, July 2022. arXiv:2103.04332 [math].

[12] William H. Gates and Christos H. Papadimitriou. Bounds for sorting by prefix reversal. *Discrete Mathematics*, 27(1):47–57, 1979.

[13] L. Gargano, U. Vaccaro, and A. Vozella. Fault tolerant routing in the star and pancake interconnection networks. *Information Processing Letters*, 45(6):315–320, 1993.

[14] Keiichi Kaneko and Shietung Peng. Disjoint paths routing in pancake graphs. In *2006 Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06)*, pages 254–259, 2006.

[15] Laurent Bulteau, Guillaume Fertin, and Irena Rusu. Pancake flipping is hard. *Journal of Computer and System Sciences*, 81(8):1556–1574, 2015.

[16] Dominique Foata and Doron Zeilberger. Denert's Permutation Statistic Is Indeed Euler-Mahonian. *Studies in Applied Mathematics*, 83(1):31–59, July 1990.

[17] Hsien-Kuei Hwang, Hua-Huai Chern, and Guan-Huei Duh. An asymptotic distribution theory for Eulerian recurrences with applications, November 2019. arXiv:1807.01412 [math].