VARIATIONS OF STACK SORTING AND PATTERN AVOIDANCE

by

Yoong Kuan Goh (Andrew)

University of Technology Sydney, Australia

Thesis
Doctor of Philosophy Degree (Mathematics)

School of Mathematical and Physical Sciences
Faculty of Science
University of Technology Sydney
February 2020

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Yoong Kuan Goh (Andrew) declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy Degree (Mathematics), in the School of Mathematical and Physical Science (Faculty of Science) at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise reference or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Production Note:
Signature: Signature removed prior to publication.

Date: 10 February 2020

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to Professor Murray Elder for his invaluable assistance and insight leading to the writing of this dissertation. Moreover, I would like to thank everyone especially Alex Bishop and Michael Coons for their advice and feedback. Lastly, special thanks to my family and friends for their support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

AN ABSTRACT OF THE DISSERTATION OF

Title: Variations of stack sorting and pattern avoidance

Supervisor: Professor Murray Elder

Sorting is a process of arranging certain objects into an ordered sequence. Real world problems such as sorting using switchyard networks, genome arrangement, and delivery of network data packets can be realised as sorting problems. The study of these problems can be translated into the study of sorting permutations using a system of data structures which can store and output data. For instance, the sorting problem in certain switchyard networks can be formulated as the problem of sorting permutations using *stacks*. The results of this thesis address the following research questions related to sorting permutations.

Open research questions

1. In a sorting process with a finite stack followed by an infinite stack in series, what is the depth of the finite stack at which the basis becomes infinite?
2. Is there a pattern avoidance characterisation for k-pass pop stack sortable permutations?

Apart from answering these questions, we develop a new notion of barred pattern avoidance[1] to accommodate some of the limitations in the existing barred pattern avoidance definition. With the new notion of barred pattern avoidance, a proof can be established to answer question 2 above.

The organisation of this thesis is as follows. Chapter 1 gives a detailed introduction to the research in permutation patterns and stack sorting. It includes some history and

[1]A barred pattern is a type of non-classical pattern that describes a set of permutations. More details will be explained in next chapter.

some major research outcomes. Moreover, several variations of sorting machines are described. Finally, a few types of non-classical patterns are explained. Chapter 2 answers the first question above. Based on some experimental data, a conjecture was made that the basis changes from finite to infinite when the depth of the finite stack is 3. We found an infinite antichain in the form of extendable sequence of numbers to prove the conjecture. Chapter 3 answers the second question and introduces a new notion of barred pattern avoidance to characterise permutations sortable by a k-pass pop-stack. Then, we finish the chapter by proving the number of forbidden patterns that the permutations sortable by k-pass pop-stack must avoid is finite. The set of forbidden patterns can be algorithmically constructed. Chapter 4 concludes the thesis with some directions for future research.

CHAPTER 1
INTRODUCTION

1.1 PERMUTATION PATTERNS

The study of permutation patterns is a significant research area in combinatorics which primarily involves identification and counting of patterns in permutations. The study began in 1968 when well-known computer scientist and mathematician Donald Knuth investigated the patterns in permutations sortable by a single stack. In his book titled *The Art of Computer Programming* [36] he showed that the permutations sortable by a stack are exactly those that avoid the pattern 231. In other words, the permutations must not contain numbers with relative ordering *middle, BIG, small*. For example, a permutation 1342 cannot be sorted by a stack because it contains the numbers 3,4 and 2 with the relative ordering *middle (3), BIG (4), small* (2) as shown in Figure 1.1.

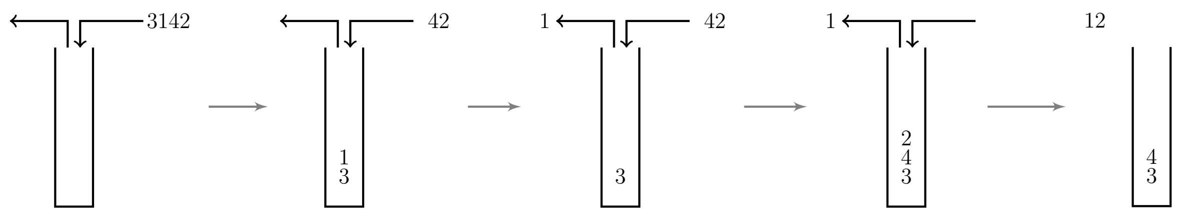


Figure 1.1: Attempting to sort the permutation 1342 with a stack

The problem of sorting permutation with stacks will be further discussed in Section 1.2

Definition 1. A permutation σ is a bijection of an ordered set to itself. In this research, the ordered set refer to a set of distinct integers. For example, we may take $\{1, 2, 3, \cdots, n\}$ with the relation $<$, and write a permutation $\left(\begin{smallmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{smallmatrix}\right)$ as 312.

Definition 2. A *subpermutation* of a permutation $\sigma = a_1 \dots a_n$ is a word $p = a_{i_1} \dots a_{i_m}$

with $1 \leqslant i_1 < \cdots < i_m \leqslant n$. For example, 1, 3, 4, 2, 13, 14, 12, 34, 32, 42, 134, 132, 142, 342 and 1342 are all the subpermutations of the permutation 1342.

Definition 3. A subpermutation p can be written in reduced form as $\mathrm{red}(p)$ by replacing its ith smallest token with i. For example, $\mathrm{red}(342) = 231$.

Definition 4. Two subpermutations with the same relative ordering are said to be *order-isomorphic*. We write $\alpha \sim \beta$ if α, β are order-isomorphic. For instance, 342 is order-isomorphic to 231, written $342 \sim 231$. Alternatively, $\alpha \sim \beta$ if and only if $\mathrm{red}(\alpha) = \mathrm{red}(\beta)$.

Definition 5. A permutation σ is said to *contain* a pattern p if it has a subpermutation that is order isomorphic to p. Otherwise, we say σ *avoids* p.

Before the study of permutation patterns was initiated in 1968, some results related to this field had already existed. The Erdős-Szekeres theorem [28] can be interpreted in the language of permutation patterns to show that for any positive integers a and b, every permutation of length equal or greater than $(a-1)(b-1)+1$ must contain at least one subpermutation $\gamma \sim 123\ldots a$ or $\gamma \sim b\ldots 321$. Therefore, the number of permutations that avoid a set of patterns $\{123\ldots a, b\ldots 321\}$ is finite.

Another result due to MacMahon [39] shows that the number of permutations (for each length) that avoid the pattern 123 can be counted by the well-known Catalan numbers, named after Eugene Charles Catalan. The first few Catalan numbers for $n = 1, 2, 3, \ldots$ are

$$1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, \ldots$$

which appearsas sequence A000108 in the Online Encyclopedia of Integer Sequences (OEIS) [30].

Research problems in permutation patterns can be categorised into characterisation, enumeration and decision problems.

1.1.1 Characterisation problems

Characterisation refers to the problem of finding (forbidden) patterns to describe a set of permutations. It is considered the origin of problems in permutation patterns since the problem that initiated the study of permutation patterns was none other than finding patterns to characterise the set of permutations sortable by a stack. Most of the problems in this category can be formulated as follows: *"Given a set of permutations S, what is the minimal set of patterns that every permutation in S must avoid?"*. In the case of characterisation of permutations sortable by a stack, *the set of permutations S* refers to all permutations sortable by a stack and the *minimal set of patterns* is $\{231\}$ which every permutation in S must avoid.

Definition 6. A set of patterns is an *antichain* if no pattern is contained in another pattern in the set. In other words, patterns in the set are pairwise avoiding.

An antichain can be used as a tool to prove minimality of a set of patterns. In general it is defined as a subset of a partially ordered set such that no two distinct elements are comparable. Notice that a set of permutations with the containment relation is a partially ordered set up to order isomorphism as in Lemma 1.1.1.

Lemma 1.1.1. *A set of permutations with the containment relation is a partial ordered set up to order isomorphism.*

Proof. Let α, β and δ be arbitrary permutations in a set of permutations S. Denote $R(\alpha, \beta)$ as "α contains β" and $|\alpha|$ to be length of α or number of tokens in α. We will prove that the set S is a partial order set by showing the containment relation over the set S is reflexive, antisymmetric and transitive.

Reflexive

If $\alpha \sim \alpha$, for all permutations by definition. So, $R(\alpha, \alpha)$.

Antisymmetric

If $R(\alpha, \beta)$ and $R(\beta, \alpha)$, then

1. α has a subpermutation $\gamma \sim \beta$, and $|\alpha| \geqslant |\beta|$

2. β has a subpermutation $\kappa \sim \alpha$, and $|\beta| \geqslant |\alpha|$

These imply that $|\alpha| = |\beta|$ and $|\gamma| = |\beta| = |\alpha| = |\kappa|$. Since γ is contained in α and κ is contained in β, $\beta \sim \gamma = \alpha \sim \kappa = \beta$. Thus, we have $\alpha \sim \beta$.

Transitive

If $R(\alpha, \beta)$ and $R(\beta, \delta)$, then we have the following

1. for $R(\alpha, \beta)$, $|\alpha| \geqslant |\beta|$ and there exist $\gamma \sim \beta$ in α

2. for $R(\beta, \delta)$, $|\beta| \geqslant |\delta|$ and there exist $\kappa \sim \delta$ in β

These imply that $|\alpha| \geqslant |\delta|$ and α contains $\gamma \sim \beta$ and β contains $\kappa \sim \delta$. Thus κ is order-isomorphic to a subpermutation in γ. So, α containing γ also contains $\kappa \sim \delta$. Thus $R(\alpha, \delta)$. □

In many cases of the characterisation problem, the size of antichains are finite [48, 23, 7]. However, an antichain can also be infinite [24].

Definition 7. A set of permutations S is a *pattern avoidance class* if there exists a set P of patterns such that $\sigma \in S$ if and only if it avoids all patterns $p \in P$. We will denote this as $S = Av(P)$. Moreover, if P is a singleton set $\{p\}$, then we will often write $Av(P)$=$Av(p)$.

We call P a *basis* for Av(P) if P is an antichain.

1.1.2 Enumeration problems

Enumeration is the problem of counting all permutations that avoid a given set of patterns.

A generating function is a useful tool to enumerate permutations in a pattern avoidance set. If a_n is the number of permutations of length n that avoid some set P, then

$$f(x) = \sum_{n=0}^{\infty} a_n x^n$$

is the generating function for $\mathrm{Av}(P)$.

For example, the generating function of $Av(213)$ is

$$\sum_{n=0}^{\infty} C_n x^n \text{ such that } C_n = \frac{1}{n+1}\binom{2n}{n}$$

[37] and the coefficient C_n of x^n is n-th Catalan number in Section 1.1. The generating functions for other prominent number sequences such as *Bell numbers* [31] and *Schröder numbers* [32] have been found to be the generating functions of some pattern avoidance sets [44, 56].

1.1.3 Decision problems

Given a permutation σ, one may ask if there is there an algorithm to decide whether σ avoids a pattern p, or set of permutations P. If so, then how efficient is that algorithm? The problems in this area mainly focus on the computational complexity of such an algorithm. Most of the solutions can be categorised into complexity classes such as polynomial time (P), nondeterministic polynomial time (NP) and polynomial space (PSPACE).

Research in this area is less developed and thus only a few notable results are known. For example, Pierrot and Rossin proved that there is a polynomial algorithm to decide whether a permutation is sortable by two stacks in series [42].

1.2 STACK SORTING

Stack sorting is one of the major branches in the study of permutation patterns. In fact, the earliest findings in the study of permutation patterns fall into this area. The study of stack sorting is about sorting permutations into ordered permutations by using some stacks[1] as a *sorting machine*. Sorting with stacks can be used to model real life problems. One example is the ordering problem in switchyard networks [37, 51]. As mentioned in Section 1.1, Knuth characterised and enumerated the set of permutations sortable by a single stack [37]. Since then, other variations of sorting machines have been considered in the literature. Such variations are detailed as follows.

1.2.1 Variations of sorting machines

In the study of stack sorting, a sorting machine refers to a system of objects that can be used to sort permutations. For example, the objects can be structures such as stacks, queues, finite token passing networks, pop-stacks and deques. Most of the objects have two main operations known as *push* and *pop*. The push operation inserts/inputs elements and the pop operation removes/outputs elements from the object. Different objects may have different ways to push and pop elements.

Stacks

A stack follows the *last in, first out* (LIFO) principle. That is, the pop operation will remove the element which was most recently pushed into the stack and still lies in the stack. For instance, if a sequence of n elements, $a_1a_2a_3\dots a_n$ are pushed into a stack in the order $a_1a_2a_3\dots a_n$, then the order of the elements leaving the stack will be $a_na_{n-1}a_{n-2}\dots a_1$. Note that the push and pop operations of a stack refer to insertion and removal of a single element one at a time. In general, sorting permutations using a stack is a nondetermistic process. During the sorting process, push operation can

[1]In computer science, "stack" is an abstract data type that is used to store a collection of elements.

happen at any stage as long as there is an element to be inserted into the stack. A pop operation can also happen at any stage of the sorting process as long as there is an element to be removed from the stack. Figure 1.2 shows how the permutation 213 can be sorted by a stack with push and pop operations.

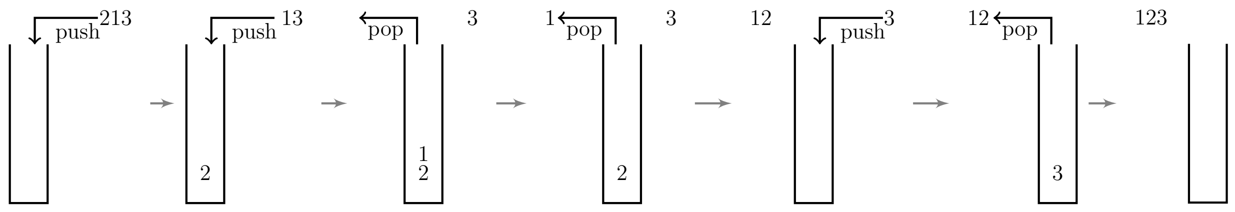


Figure 1.2: A simple representation of push and pop operations of a stack

In a stack sorting machine, stacks can be arranged in parallel or series. Different arrangements of stacks may produce different sets of sortable permutations.

Stacks in parallel: Two stacks L and R are in parallel if an element from the input can either be pushed directly into L or R. During a pop operation, an element cannot move from one stack to another except to the output. Diagram 1.3 shows the arrangement of two stacks in parallel. Note that with this configuration, a permutation can only be sorted when the elements in the stacks remain in ascending order when read from top (opening) to bottom throughout the sorting process.

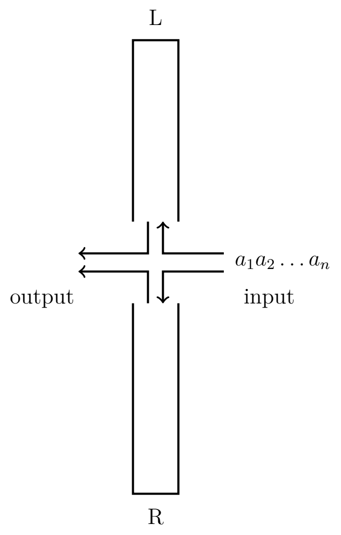


Figure 1.3: Two stacks in parallel

In 1970s, Even, Itai and Pratt considered the problem of sorting with stacks in parallel. Even and Itai solved the decision problem for such a machine by proving that there is an algorithm to decide whether a permutation is sortable by two stacks in parallel that runs in polynomial time [29]. Later in 1984, Rosenstiehl and Tarjan presented a faster algorithm to decide whether a permutation is sortable by two stacks in parallel that runs in linear time [46]. Meanwhile, Pratt showed that the permutations sortable by two stacks in parallel must avoid a minimal infinite set of patterns [43]. Furthermore, Pratt extended his result for $k \geqslant 2$ stacks in parallel and found that the bases are still infinite. The enumeration problem was left open until Albert and Bousquet–Mélou showed that the generating function for permutations sortable by two stacks in parallel is the solution of a system of functional equations [1].

Stacks in series: Two stacks L and R are in series if both are adjacent to each other as shown in the Figure 1.4. Thus, an input has to pass through both R and L stacks in a consecutive order before it can be output. Moreover, we can only pop an element from a stack if it can be immediately pushed into the following stack.

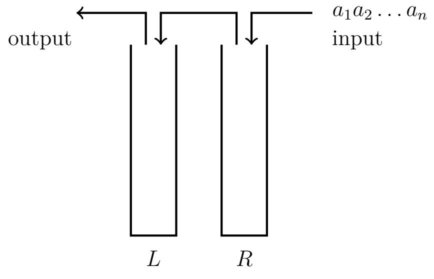


Figure 1.4: Two stacks in series

Sorting permutations with stacks in series is more complicated than stacks in parallel. This is because there are more choices for where to move an inserted element during the sorting process. Therefore, the problems pertaining to stacks in series appear harder to solve compared to stacks in parallel. The basis for two stacks in series was found to be infinite by Murphy, but the enumeration problem remains open [41]. As mentioned above, the decision problem was also recently shown to be in polynomial time [42]. For $k > 2$ stacks in series, there is no known result.

In 2006, Elder introduced a new variation of two stacks in series. Instead of having both stacks to be infinite as in other research, Elder restricted one of the stacks to be finite. Interestingly, Elder proved that the set of permutations *generated* [2] by a stack of depth 2 and an infinite stack in series has a basis consisting of the following 20 permutations [23].

$$\left\{ \begin{array}{lllll} 51234, & 51243, & 51423, & 52134, & 52143 \\ 52413, & 645123 & 416235, & 416253, & 645213 \\ 426135, & 426153, & 4175623, & 4137256 & 4137265 \\ 4275613, & 4237156, & 4237165, & 41386725, & 42386715 \end{array} \right\}$$

Equivalently, the basis for permutations *sorted* by a stack of depth 2 and an infinite

[2]all possible permutations that are formed by passing the sequence $1, 2, \ldots, n$ through a stack

stack in series is

$$\left\{\begin{array}{lllll} 23451 & 23541 & 24531 & 32451 & 32541 \\ 42531 & 245163 & 246153 & 425163 & 426153 \\ 456231 & 546231 & 2531674 & 2531764 & 2671453 \\ 5231674 & 5231764 & 6271453 & 27318564 & 72318564 \end{array}\right\}$$

Moreover, Elder, Lee and Rechnitzer proved that the generating function for this variation of two stacks in series is an algebraic function [25].

Elder also proved that if the basis for a depth k stack followed by an infinite depth stack is infinite, then so is the basis for a depth $k+1$ stack followed by and infinite depth stack [23]. Recall that the basis for two infinite stacks in series is infinite. *Therefore, there must be some value of k at which the basis changes from finite to infinite.* Chapter 2 of this thesis will address this boundary.

In 1993, West considered sorting permutations by passing the permutations through a single stack with a greedy sorting process such that an element is input into the stack if it is smaller than the top element of the stack; otherwise an element is output from the stack. Instead of performing one sorting process through a stack, he performed the sorting process twice through the same stack.

In the literature, this is known as West 2-stack sorting. That is, he investigated the permutations that are sortable by passing twice through a stack using the deterministic greedy process above. Note that this sorting process forces elements in the stack to remain ordered when read from top to bottom. West showed that the permutations sorted by this process must avoid the minimal set of two patterns $\{2341, 3\bar{5}241\}$ [55]. Note that the pattern $3\bar{5}241$ with a barred element is not the usual classical pattern but it is a type of non-classical pattern. Section 1.3 will introduce some of the common types of non-classical patterns that appear in the literatures.

In 2002, Atkinson, Murphy and Ruškuc investigated the problems of two infinite stacks in series by putting a restriction on both stacks. The restriction is that the elements in both stacks must always appear in ascending order when read from top to bottom. This system configuration similar to West system configuration but the algorithm on how to sort a permutation is different in both configurations. West's algorithm depends on a greedy algorithm that favours operations "to the right" of the system while their system favour "left" greedy algorithm. For details, refers to [5]. They proved that the pattern avoidance set has an infinite basis and the generating function is

$$\sum_{n=0}^{\infty} z_n x^n = \frac{32x}{-8x^2 + 20x + 1 - (1 - 8x)^{3/2}}$$

where z_n is the number of permutations of length n [5]. Surprisingly, this generating function is also the generating function for $Av(1342)$ [12]. It can be seen that $Av(1342)$ and $Av(\text{infinite basis})$ are having the same generating function as above. Therefore, pattern avoidance sets with the same generating function can have completely different minimal sets of patterns that describe the avoidance sets and vice versa. This variation of two stacks in series is almost similar to two stacks in parallel as the elements of both stacks in parallel also have to be in ascending order when read from top to bottom. Smith consider a similar variation of stacks where the elements in one of the two stacks must always appear in descending order when read from top to bottom. She found that the basis is a finite set of two patterns $\{3142, 3241\}$ [48] and Kremer has shown that the pattern avoidance set of these two patterns can be enumerated by the generating function of the Schröder number [38]

Pop-stacks

A pop-stack is similar to a stack except that its pop operation removes all elements from the stack at once. There are two types of pop-stacks; *deterministic* and *nondeterministic*. In a deterministic pop-stack, the pop operation is only triggered when the next element to be pushed has a value bigger than the element on top of the pop-stack or when there is no element to be pushed from the input. Figure 1.5 shows a permutation 4213 passing through a deterministic pop-stack.

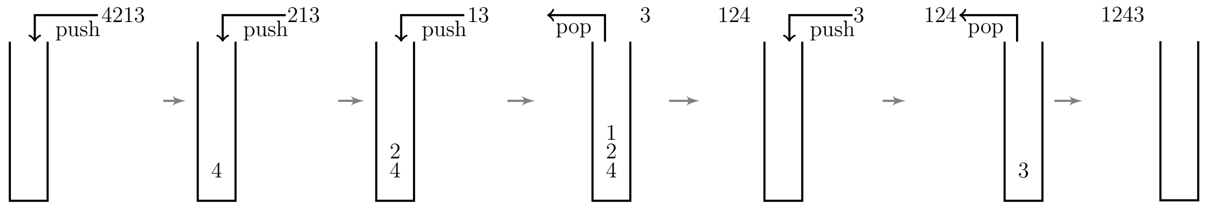


Figure 1.5: Passing 4213 through a deterministic pop-stack

Meanwhile, a nondeterministic pop-stack does not compare the value of the elements to determine the pop or push operations. The pop and push operations are nondeterministic like the operations of a stack. Due to the nondeterministic nature of this pop operation, a permutation such as 231 may pass through the pop-stack in several different ways, potentially producing different outputs as in Figure 1.6.

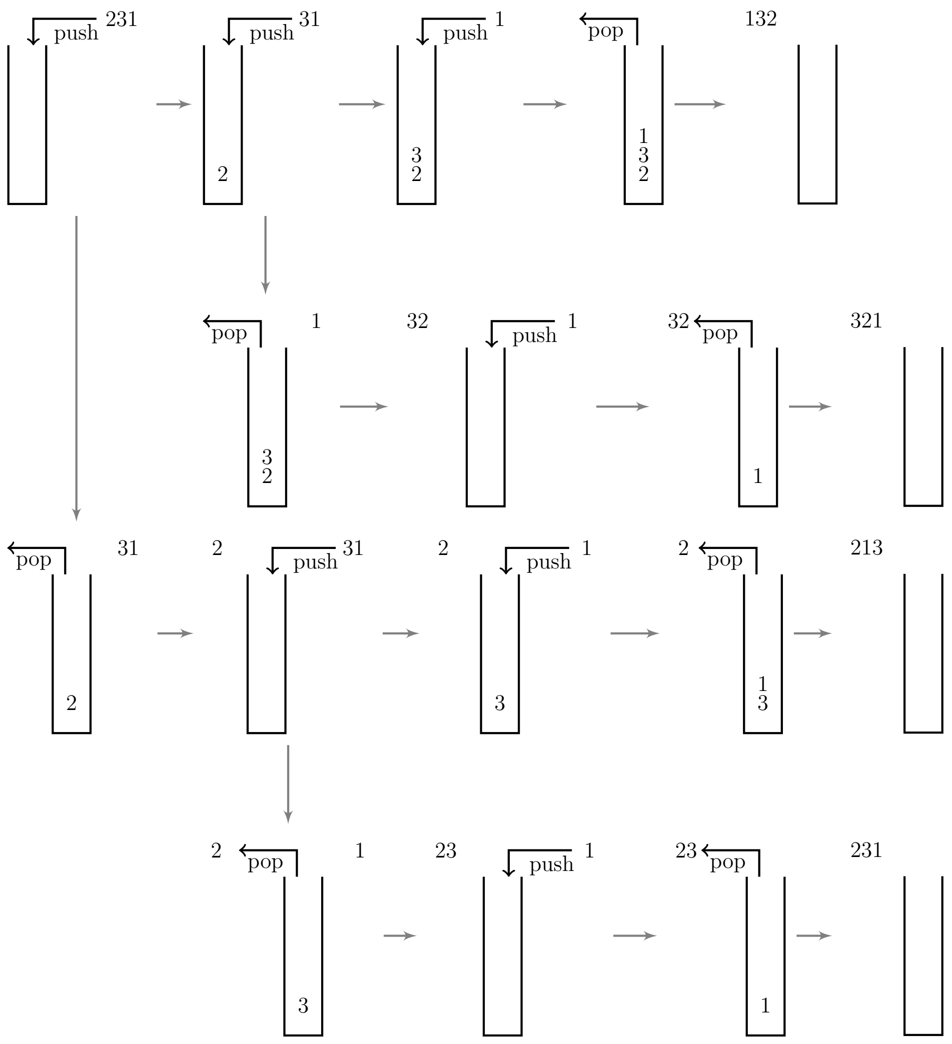


Figure 1.6: Passing 231 through a nondeterministic pop-stack

Sorting with a pop-stack has less power than a normal stack because each pop operation forces every element in the pop-stack to be removed. That is, fewer permutations can be sorted by a pop-stack than a normal stack. Explicitly, Avis and Newborn [7] proved a permutation can be sortable by a pop-stack if and only if it avoids the set of patterns $\{231, 312\}$, and the number permutations of length n that can be sorted is 2^{n-1}, while as we know from Knuth permutations sortable by an ordinary stack only need to avoid the set $\{231\}$ and the number of permutations of length n that can be sorted is C_n which is asymptotically 4^n.

The study of sorting permutations with pop-stacks began in 1980s when Avis and Newborn investigated the characterisation and enumeration problems of sorting permutations by using pop-stacks in series. Pop-stacks in series operate as follows. Tokens can be passed from the output of one pop-stack to the input of the next stack at any time, as long as the entire contents are popped. By contrast, a k-pass pop-stack requires that i-th pop-stack must complete its output before the $(i+1)$-st pop-stack can be used. They proved that the set of permutations sortable by k pop-stacks in series is characterised by a minimal set of finite patterns [7] and they also provided the enumerations of these permutations for every length k.

Later, Atkinson and Sack considered pop-stacks in parallel. They proved that for two pop-stacks in parallel, the characterisation is a minimal set of 7 patterns, $\{3214, 2143, 24135, 41352, 14352, 13542, 13524\}$. They were also able to characterise the set of permutations sortable by k pop-stacks in parallel with a minimal finite set of patterns [6]. Then, Smith and Vatter proved the conjecture of Atkinson and Sack that the generating function for k pop-stacks in parallel is a rational function [49].

Recently, Pudwell and Smith considered permutations that are sortable after passing twice through a deterministic pop-stack. Similar to West's result with a 2-pass stack [55], they proved that these permutations must avoid not only some classical patterns but also non-classical *barred patterns*. In particular, the patterns are $2341, 3412, 3421, 4123, 4231, 4312, 41\bar{3}52$ and $413\bar{5}2$ [45]. They also proved that the generating function is in bijection with a special family of polynominoes. After that, Claesson and Guðmundsson extended their enumeration result by proving that the generating function for permutations that are sortable after k passes through a pop-stack is a rational function [18]. Following this result, the characterisation problem for k-pass deterministic pop-stacks remained open. In Chapter 3, we provide a solution to this problem.

Deques

A double-ended-queue or deque is similar to a stack but with two openings where the push and pop operations can occur at either openings. Therefore, a deque with push and pop operations can be visualised as in Figure 1.7.

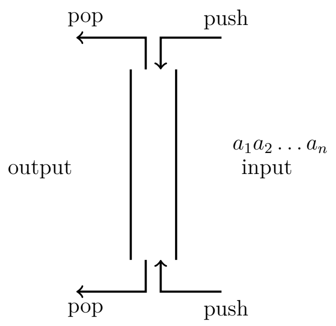


Figure 1.7: A deque

There is less literature available here, compared to stacks and pop-stacks. Some notable results in sorting permutations with a deque are that Pratt showed that the basis for sorting permutations with a deque is infinite [43] extending the research of Knuth who had shown that set of bases for an *input-restricted* deque is a set of 2 patterns $\{4231, 3241\}$ [36]. An input-restricted deque is a deque where push operations can only happen at one end. The enumeration problem for deque sortable permutations is open.

1.2.2 Problems and motivations

Stack sorting has been an active area of research and many results have been proven since 1968. Numerous interesting variation of sorting machines have been introduced and studied by many well-known researchers. Table 1.1 provides a summary of results that have been obtained with different variations of sorting machines. The motivation of this thesis is based on previous research and open questions (Open) in stack sorting.

Variation of sorting machine	Characterisation	Enumeration	Decision
single stack	finite [36]	Catalan [36]	linear†
2 stacks in parallel	infinite [43]	[1]	linear [46]
$k > 2$ stacks in parallel	infinite [43]	Open	Open
2 stacks in series	infinite [41]	Open	polynomial [42]
$k > 2$ stacks in series	Open	Open	Open
West-2-stack	finite [55]	Algebraic [58]	linear†
West-3-stack	finite [54]	[19]	linear†
West-k-stack	Open	Open	linear†
2 stacks in series (ascending)	infinite [5]	[5]	Open
2 stacks in series (descending)	finite [48]	[48]	polynomial* [48]
(k, ∞) stacks in series, $k = 2$	finite [23]	Algebraic [25]	polynomial* [23]
(k, ∞) stacks in series, $k > 2$	Open	Open	Open
single pop-stack	finite [7]	[7]	linear†
$k \geqslant 2$ pop-stacks in series	finite [7]	[7]	polynomial* [7]
$k \geqslant 2$ pop-stacks in parallel	finite [6]	Rational [49]	linear [6]
2-pass deterministic pop-stack	finite [45]	Rational [45]	linear†
$k > 2$ pass deterministic pop-stack	Open	Rational [18]	linear†
single deque	infinite [43]	Open	[46][20]
single input-restricted deque	finite [36]	[36]	polynomial [36]

Table 1.1: Summary of past results in stack sorting

$\dagger$ Since the process is deterministic, it can be done in linear time.

* Decision results were not stated explicitly by the authors of the papers but we interpreted that the algorithm to decide a permutation is sortable based on a finite number of patterns to avoid can run in polynomial time.

Our first motivation is to extend Elder's result in sorting permutations with a stack of depth k and an infinite stack in series, written as (k, ∞) stacks in series in the table. Elder proved that if $k = 2$, then the number of basis elements is finite. However, Murphy proved that if k is infinite, then the number of basis elements is infinite. Thus, there is a question whether there is a minimum finite k so that the set of basis elements can become infinite. In Chapter 2, we proved that the number of basis elements changes from finite to infinite when $k = 3$. Then we extend this result by showing that for any $k > 2$, the basis is infinite.

Our second motivation is to solve the characterisation problem for k-pass pop-stack answering a question of Claesson and Guðmundsson. [18]. Note that, solving characterisation problems may also solve the decision problem. This is because whether a permutation is sortable can be decided in polynomial time if it only has to avoid finitely many patterns. So, if the number of forbidden patterns for permutations sortable by k-pass pop-stack is finite, then the algorithm to decide whether the permutation is sortable by k-pass pop-stack runs in polynomial time. In Chapter 3, we prove that the number of basis elements for k-pass pop-stack is finite. Therefore, the algorithm is a polynomial algorithm. In next section, a few significant types of non-classical pattern will be explored especially the barred pattern before we give our next problem and motivation in non-classical pattern context.

1.3 NON-CLASSICAL PATTERNS

The study of pattern avoidance in permutation patterns becomes very challenging when it involves more restrictions to avoid a single or multiple patterns. For instance, one can ask what is the pattern avoidance set for a pattern 4213 such that the permutations that contain it must have the subpermutation $a_ia_ja_ka_l \sim 4213$ with the *restriction* a_i *and* a_k *must be adjacent* in the permutations. The permutation 5**4213** contains it but not **42**5**13** because the tokens 2 and 1 are not adjacent. In this case, we cannot use classical pattern to represent the pattern directly. Therefore, several types of non classical pattern such as *vincular patterns*, *bivincular pattern* and *barred patterns* were introduced.

1.3.1 Vincular patterns

Vincular patterns were introduced by Babson and Steingrímsson [8] in 2000 to put extra restrictions on a classical pattern $p = p_1p_2p_3 \dots p_k$ in a permutation $\sigma = a_1a_2a_3 \dots a_n$. These restrictions can be divided into three types such as follows: For each subpermutation $a_{i_1}a_{i_2}a_{i_3} \dots a_{i_k}$ order isomorphic to p=$p_1p_2p_3 \dots p_k$ in σ,

1. (Type 1) a_{i_j} and $a_{i_{j+1}}$ corresponding to p_m, p_{m+1} must be adjacent in σ that is $i_{j+1} = i_j + 1$
2. (Type 2) a_{i_1} corresponding to p_1 must be a_1
3. (Type 3) a_{i_k} corresponding to p_k must be a_n

In order to reflect these restrictions on a classical pattern, some decorations are added to the pattern.

Definition 8. A vincular pattern, p is a classical pattern where some of its consecutive tokens are underlined as follows:

1. If p contains $\underline{p_m, p_{m+1}, p_{m+2}, \dots, p_{m+l}}$, then each occurrence of p in σ must be order isomorphic to p and contain a subpermutation, $a_{i_1}a_{i_2}a_{i_3} \dots a_{i_l}$ order isomorphic to $p_m, p_{m+1}, p_{m+2}, \dots, p_{m+l}$ such that i_{j+1}=$i_j + 1$ for each $m \leqslant j < m + l$.
2. If p begins with $\lceil p_1 \dots$, then the occurrence of p in σ must start with the first token a_1 in σ.
3. If p ends with $\dots p_{|p|}\rceil$, then the occurrence of p in σ must end with the last token a_n in σ.

Note: A vincular pattern p without any decoration on its tokens is also a classical pattern.

Example 1.3.1. Let p= $\underline{54}3\underline{21}$ and σ=971865342. In σ, there are only four occurrences of p which are the subpermutations 97653, 97642, 97542 and 86542. Each of these occurrences has its first two tokens in descending order ($\sim$ 54) and adjacent to each other in σ. Similarly, its last two tokens correspond to $\underline{21}$. Meanwhile, the subpermutations 98653, 98642 and 98542 are not occurrences of p, since the first two tokens 9 and 8 are not adjacent in σ. Table 1.2 shows a few more examples of vincular patterns in a permutation.

Pattern	Occurrences in 365142
132	$365, 364, 354, 142$
$\underline{13}2$	$365, 364, 142$
$1\underline{32}$	$365, 142$
$\underline{132}$	$365, 142$
$\lceil 132$	$365, 364, 354$
$132\rceil$	142

Table 1.2: Examples of vincular patterns in a permutation

A permutation σ is said to contain a vincular pattern p if there exists at least one occurrence of p in σ. Otherwise σ avoids p. Even though a vincular pattern has more restrictions than a classical pattern, there are some cases when it characterises the same set of permutations as a classical pattern. For example, $Av(2\underline{13})$=$Av(213)$ [15].

Vincular patterns can be used to describe *Baxter permutations* which were introduced by Glen Baxter in 1964. A permutation σ is a Baxter permutation if it does not have any subpermutation $a_ia_ja_ka_l$ order isomorphic to 2413 or 3142 and k=$j+1$. So, $Av(\{2\underline{41}3, 3\underline{14}2\})$ represents the set of all Baxter permutations. Since the introduction of vincular patterns, much research involving these patterns has been published [2, 3, 10, 11, 15, 16, 17, 26, 27, 34, 50, 52].

1.3.2 Bivincular patterns

A bivincular pattern is an extension of vincular pattern. It can have all the restrictions of a vincular pattern as well as additional restriction to control values in a pattern.

Definition 9. A bivincular pattern p is a written in a two-row notation such that the bottom row is a vincular pattern and the top row is a increasing permutation $123\dots|p|$ with some tokens overlined.

1. If the bottom row is a vincular pattern $\underline{p_m, p_{m+1}, p_{m+2}, \dots, p_{m+l}}$, then each occurrence of p in σ must be a subpermutation, $a_{i_j}a_{i_{j+1}}a_{i_{j+2}}\dots a_{i_{j+l}}$ order isomorphic to $p_m, p_{m+1}, p_{m+2}, \dots, p_{m+l}$ and $i_{j+1} = i_j + 1$.

2. If the top row contains $\overline{m(m+1)(m+2)\dots(m+l)}$, then any occurrence of p in σ must contain a subpermutation $a_{i_j}a_{i_{j+1}}a_{i_{j+2}}\dots a_{i_{j+l}}$ order isomorphic to $p_m p_{m+1} p_{m+2}\dots p_{m+l}$ with values $a_{i_{j+x}} = a_{i_{j+x+1}} - 1$.

3. If the top row begins with $\lceil 1\dots$, then any occurrence of p in σ must start with the smallest token in σ

4. If the top row ends with $\dots|p|\rceil$, then any occurrence of p in σ must end with the largest token in σ

Example 1.3.2. Let $\sigma = 241635$. The subpermutation 235 is an occurrence of the bivincular pattern $\begin{smallmatrix}\overline{1\,2}\,3\\1\,2\,3\end{smallmatrix}$ because 235 is order isomorphic to the bottom notation 123 and the tokens 2 and 3 are adjacent such that $2 = 3-1$ which satisfies the above notation $\overline{12}$. Meanwhile, $\begin{smallmatrix}1\,2\,\overline{3}\rceil\\\lfloor\underline{1\,2}\,3\end{smallmatrix}$ is the pattern of the subpermutation 246. This is because $246 \sim 123$ has its first two tokens adjacent in σ and the token 2 is the first token in σ (bottom row restriction - vincular pattern). Moreover, the token 6 is the largest token in σ (top row restriction). On the other hand, σ does not have an occurrence of the pattern $\genfrac{}{}{0pt}{}{\overline{1\,2\,3}}{1\,2\,3}$ because the only possible occurrence for this bivincular pattern is $a_{i_j}a_{i_{j+1}}a_{i_{j+2}}$ such that $a_{i_j}{=}a_{i_{j+1}}-1$ and $a_{i_{j+1}}{=}a_{i_{j+2}}-1$.

In term of pattern avoidance, σ contains a bivincular pattern p if there exist at least one occurrence of p in σ. Otherwise, σ avoids p.

1.3.3 Barrred patterns

A barred pattern is a classical pattern where some of its tokens are barred. Barred patterns were introduced by West to characterise the permutations that are sortable after passing twice through a stack with certain restrictions [55]. The set of all barred patterns of length 2 is

$$B_2 = \{12, 21, \bar{1}2, 1\bar{2}, \bar{2}1, 2\bar{1}, \bar{1}\bar{2}, \bar{2}\bar{1}\}$$

Given a barred pattern β, we denote by unbar(β) the permutation obtained after removing the bar symbol from any token in β. For example, if $\beta = 3\bar{2}41$, then unbar(β) $= 3241$. Meanwhile removebar(β) is the permutation obtained by deleting tokens with bars and then reducing. So, removebar(β) $= 231 \sim 341$. Note that a barred pattern which satisfies $\beta =$ unbar(β) can be considered as a barred pattern or a classical permutation pattern.

In terms of pattern avoidance, a permutation σ is said to avoid the barred pattern β if *each* occurrence of removebar(β) in σ (if any) is a part of an occurrence of unbar(β) in σ ([35, Definition 1.2.3]). A similar definition is mentioned in [21] and [55]. For example, $\sigma = 13524$ is said to avoid a barred pattern $\beta = 2\bar{4}13$ because the only subpermutation in σ that is order-isomorphic to removebar(β) $= 213$ is 324 which is also a part of the subpermutation, $3524 \sim$ unbar(β). Meanwhile, 53124 does not avoid but *contains* β

because it has subpermutation, 324 that is order-isomorphic to removebar(β) = 213 but is not part of any subpermutation that is order-isomorphic to unbar(β).

Jean-Luc Baril [9] categorised this definition of barred pattern avoidance as *weak avoidance.* In [9] and [44], barred pattern avoidance is defined with additional restrictions. We define the bar positions in a barred pattern β_n of length n as $\beta_{pos} = t_1t_2\ldots t_n \in \{0,1\}^n$ such that 0 means no bar and 1 means there is a bar. For example, the bar position of $\beta = 2\bar{4}13$ is $\beta_{pos} = 0100$. A permutation σ is said to *avoid* (not *weakly avoid*) a barred pattern β if *each* occurrence of removebar(β) in σ (if any) can be extended into an occurrence of the pattern β in σ according to β_{pos}. Note that this avoidance is more restrictive than the weak avoidance. For instance, given a barred pattern $\beta = 1\bar{2}$ with $\beta_{pos} = 01$, a permutation 12 does not avoid β since the subpermutation $2 \sim$ removebar(β) = 1 cannot be extended to 12 according to β_{pos}. Another example is σ=123. In this case, the subpermutation $2 \sim$ removebar(β) = 1 is able to extend based on β_{pos} by being part of the subpermutations 23 in σ. However, the subpermutation $3 \sim$ removebar(β) = 1 cannot be extended to be part of a subpermutation order-isomorphic to 12 according to β_{pos} unless there is another token $a_m > 3$ and appear on the right of 3 in σ. Thus it can be seen that $Av(1\bar{2})$ is an empty set based on this strict avoidance. However, the permutation 12 avoids $\beta = 1\bar{2}$ in the context of weak avoidance.

Based on this strict barred pattern avoidance, Pudwell gave a collection of enumeration results for permutations that avoid barred patterns of length $\leqslant 4$ and showed that there is no single permutation avoiding a barred pattern with only 1 unbar token [44]. Note that this strict avoidance is known as *strong avoidance.*

Both barred pattern avoidance definitions disagree with the usual classical pattern avoidance when β has no bar tokens. For example, $\sigma = 21$ does not avoid 21 in the usual sense of classical pattern avoidance, but if 21 is considered as a barred pattern then σ avoids 21. *This is one of the reasons why we developed a better notion of barred* *pattern avoidance in Chapter 3.*

Barred patterns can be useful in characterising some permutations characterised by other patterns. For instance, $Av(21\bar{3}54) = Av(2\underline{14}3)$ [53]. Moreover, $Av(\{25\bar{3}14, 41\bar{3}52\})$ represents the set of Baxter permutations [22, 52]. Note that in our previous example, $Av(\{2\underline{41}3, 3\underline{14}2\})$ also represents the set of Baxter permutations. Thus, we have $Av(\{25\bar{3}14, 41\bar{3}52\})$=$Av(\{2\underline{41}3, 3\underline{14}2\})$. Other earlier literatures that used using barred patterns include [13, 57]

1.3.4 Problems and motivations

Throughout the study of non-classical pattern in permutation patterns, the existence of barred pattern provides an interesting area of study. One of the reasons is because of its presence in the study of stack sorting. Moreover, its pattern avoidance definition has some issues in characterising permutations sortable by k-pass pop stack when $k > 2$. We will discuss the issues in Chapter 3 before we define a new barred pattern avoidance definition (which we call *PB-avoidance* to distinguish it) that fills the gaps of existing definition.

CHAPTER 2

PERMUTATIONS SORTED BY A FINITE AND AN INFINITE STACKS IN SERIES

2.1 AIM

In this chapter, we want to prove that the set of permutations sorted by a stack of depth $t \geqslant 3$ and an infinite stack in series has infinite basis, by constructing an infinite antichain of unsortable permutations. This answers the open question of identifying the point at which, in a sorting process with two stacks in series, the basis changes from finite to infinite.

2.2 INTRODUCTION

Sorting mechanisms are natural sources of pattern avoidance classes, since (in general) if a permutation cannot be sorted then neither can any permutation containing it. Recall from Chapter 1 that the set of permutations sortable by a stack of depth 2 and an infinite stack in series has a basis of 20 permutations [23], while for two infinite stacks in series there is no finite basis [41]. For systems of a finite stack of depth 3 or more and infinite stack in series, it was not known whether the basis was finite or infinite.

The outcome from this chapter shows that for depth 3 or more the basis is infinite. An infinite antichain belonging to the basis of the set of permutations sortable by a stack of depth 3 or more and an infinite stack in series is explicitly constructed. A simple lemma then implies the result for depth 4 or more. For completeness we also give an explicit construction of an infinite antichain belonging to the basis for (k, ∞) when $k \geqslant 4$. The main result in this chapter has been published as a paper [24].

2.3 EXPERIMENTAL DATA

A computer program implemented in Python and C++ programming languages was used[1] to find the number of potential basis permutations of lengths up to 13. The computer search yielded 8194 basis permutations of lengths up to 13 (see Table 2.1; basis permutations are listed at `https://github.com/gohyoongkuan/stackSorting-3`). The antichain used to prove our theorem was found by examining this data and looking for patterns that could be arbitrarily extended.

Table 2.1: Number of basis elements for $S(3, \infty)$ of length up to 13

Permutation length	Sortable permutations (count)	Basis elements (count)	
5	120	0	
6	711	9	+9
7	4700	83	+74
8	33039	169	+86
9	239800	345	+176
10	1769019	638	+293
11	13160748	1069	+431
12	98371244	1980	+911
13	737463276	3901	+1921

The size of the basis keeps increasing, unlike the situation for $k = 2$. With this numerical evidence, a conjecture was made at this stage that the size of the basis might be infinite. Thus, an alternative approach to prove the conjecture was to find an infinite antichain by examining this data and looking for patterns that could be arbitrarily extended.

2.4 PRELIMINARIES

The notation $\mathbb{N}$ denotes the non-negative integers $\{0, 1, 2, \dots\}$ and $\mathbb{N}_+$ the positive integers $\{1, 2, \dots\}$. Let S_n denote the set of permutations of $\{1, \dots, n\}$ and let $S^\infty = \bigcup_{n \in \mathbb{N}_+} S_n$. A permutation $\sigma \in S_n$ can be represented as a plot which is the set of points

[1] enumerating sorting codewords of length $3n$ as described in Chapter 4, rather than enumerating the $n!$ permutations.

$\{(i, \sigma_i) | 1 \leqslant i \leqslant n\}$ [45]. For instance, Figure 2.1 shows the plot for the permutation 243651.

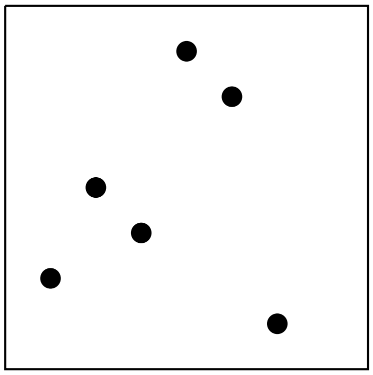


Figure 2.1: Permutation plot of 243651

As before, the set of all permutations in S^∞ which avoid every permutation in $\mathscr{B} \subseteq S^\infty$ is denoted $Av(\mathscr{B})$. A set of permutations is a *pattern avoidance class* if it equals $Av(\mathscr{B})$ for some $\mathscr{B} \subseteq S^\infty$. A set $\mathscr{B} = \{q_1, q_2, \ldots\} \subseteq S^\infty$ is an *antichain* if no q_i contains q_j for any $i \neq j$. An antichain $\mathscr{B}$ is a *basis* for a pattern avoidance class $\mathscr{C}$ if $\mathscr{C} = Av(\mathscr{B})$.

Let M_t denote the machine consisting of a stack, R, of depth $t \in \mathbb{N}_+$ and infinite stack, L, in series as in Fig. 2.2. A *sorting process* is the process of moving entries of a permutation from right to left from the input to stack R, then to stack L, then to the output, in some order. Each item must pass through both stacks, and at all times stack R may contain no more than t items (so if at some point stack R holds t items, the next input item cannot enter until an item is moved from R to L).

A permutation $\alpha = a_1 a_2 \ldots a_n$ is in $S(t, \infty)$ if it can be sorted to $123 \ldots n$ using M_t. For example, $243651 \in S(t, \infty)$ for $t \geqslant 3$ since it can be sorted using the following process: place $2, 4$ into stack R, move $4, 3, 2$ across to stack L, place $6, 5, 1$ into stack R, then output $1, 2, 3, 4, 5, 6$. Note $243651 \notin S(2, \infty)$ by [23].

The following lemmas will be used to prove our main result.

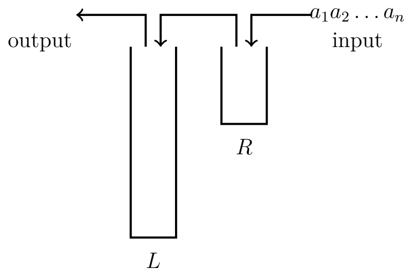


Figure 2.2: A stack R of depth t and an infinite stack L in series

Lemma 2.4.1. *Let $\alpha = a_1a_2\dots a_n \in S(t,\infty)$ for $t \in \mathbb{N}_+$. If $i < j$ and $a_i < a_j$ then in any sorting process that sorts α, if both a_i and a_j appear together in stack L then a_i must be above a_j.*

Proof. If a_j is above a_i in stack L then the permutation will fail to be sorted. □

Lemma 2.4.2. *Let $\alpha = a_1a_2\dots a_n \in S(t,\infty)$ for $t \geqslant 3$ and suppose $1 \leqslant i < j < k \leqslant n$ with $a_ia_ja_k$ order-isomorphic to 132. Then in any sorting process that sorts α, a_i, a_j, a_k do not appear together in stack R.*

Proof. If a_i, a_j, a_k appear together in stack R, we must move a_k then a_j onto stack L before we can move a_i, but this means a_j, a_k violate Lemma 2.4.1. Note that this is followed from the Knuth's 231 result as shown in Figure 1.1 □

Lemma 2.4.3. *Let $\alpha = a_1a_2\dots a_n \in S(t,\infty)$ for $t \geqslant 3$ and $1 \leqslant i_1 < i_2 < \dots < i_6 \leqslant n$ with $a_{i_1}a_{i_2}\dots a_{i_6}$ order isomorphic to 243651. Then in any sorting process that sorts α, at some step of the process a_{i_4} and a_{i_5} appear together in stack R.*

Proof. For simplicity let us write $a_{i_1} = 2, a_{i_2} = 4, a_{i_3} = 3, a_{i_4} = 6, a_{i_5} = 5, a_{i_6} = 1$. Before 6 is input, $2, 3, 4$ are in the two stacks in one of the following configurations:

1. $2, 4, 3$ are all in stack R. In this case we violate Lemma 2.4.2.

2. two items are in stack R and one is in stack L. In this case by Lemma 2.4.1 we cannot move 6 to stack L, so 6 must placed and kept in stack R. If $t = 3$ stack

R is now full, so 5 cannot move into the system, and if $t \geqslant 4$, when 5 is input we violate Lemma 2.4.2.

3. one item, say a, is in stack R and two items are in stack L. In this case we cannot move $6, 5$ into stack L by Lemma 2.4.1 so they remain in stack R on top of a, violating Lemma 2.4.2.

4. stack R is empty. In this case, $2, 3, 4$ must be placed in stack L in order, else we violate Lemma 2.4.1. We cannot place $6, 5$ into stack L until it is empty, so they must both stay in stack R until 4 is output.

In particular, the last case is the only possibility and in this case a_{i_4}, a_{i_5} appear in stack R together. $\square$

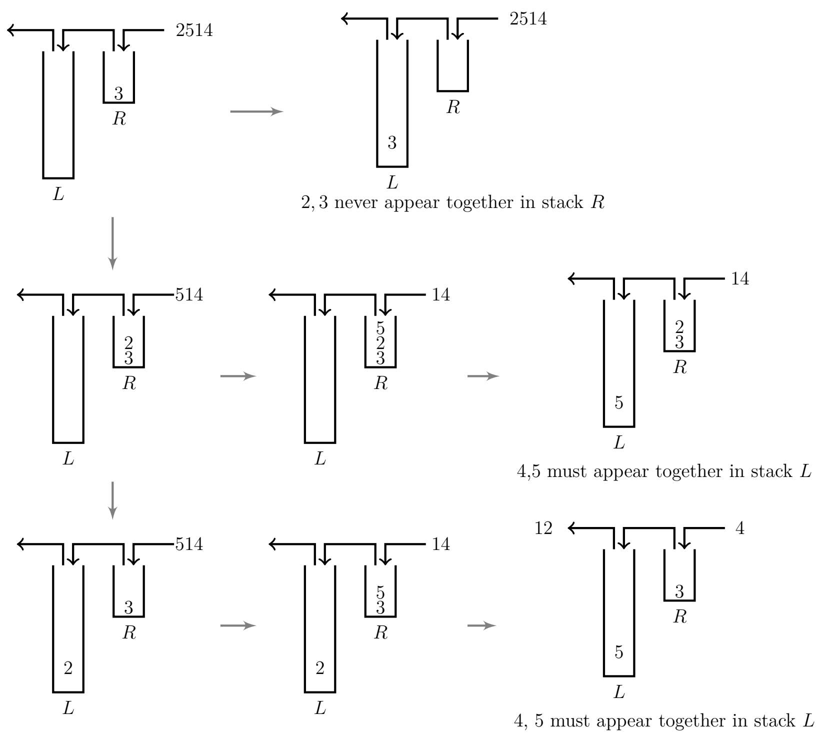
Lemma 2.4.4. *Let* $\alpha = a_1a_2 \dots a_n \in S(t, \infty)$ *for* $t \geqslant 3$ *and suppose* $1 \leqslant i_1 < i_2 < \cdots < i_5 \leqslant n$ *with* $a_{i_1}a_{i_2} \dots a_{i_5}$ *order-isomorphic to* 32514*. Then, in any sorting process that sorts* α*, if* a_{i_1}, a_{i_2} *appear together in stack* R*, then at some step in the process* a_{i_3}, a_{i_5} *appear together in stack* L*.*

Proof. For simplicity let us write $a_{i_1} = 3$, $a_{i_2} = 2$, $a_{i_3} = 5$, $a_{i_4} = 1$, $a_{i_5} = 4$. Figure 2.3 indicates the possible ways to sort these entries, and in the case that $2, 3$ appear together in stack R we see that $4, 5$ must appear in stack L together at some later point. $\square$

Lemma 2.4.5. *Let* $\alpha = a_1a_2 \dots a_n \in S(t, \infty)$ *for* $t \geqslant 3$ *and suppose* $1 \leqslant i_1 < i_2 < \cdots < i_5 \leqslant n$ *with* $a_{i_1}a_{i_2} \dots a_{i_5}$ *order-isomorphic to* 32541*. Then, in any sorting process that sorts* α*, if* a_{i_1}, a_{i_2} *appear together in stack* L*, then at the step that* a_{i_1} *is output,*

1. a_{i_3}, a_{i_4} *are both in stack* R*, and*

2. *if* a_k *is in stack* L *then* $k < i_2$*.*



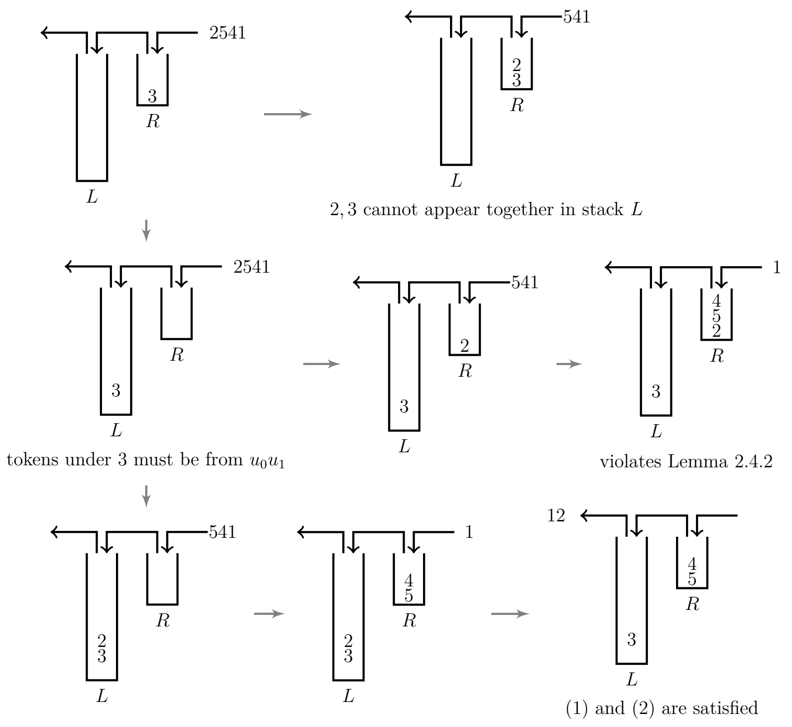
2514
3
R
L
2514
R
3
L
2, 3 never appear together in stack R
514
2
3
R
L
14
5
2
3
R
L
14
2
3
R
5
L
4,5 must appear together in stack L
514
3
R
2
L
14
5
3
R
2
L
12
4
3
R
5
L
4, 5 must appear together in stack L

Figure 2.3: Sorting 32514

Proof. For simplicity let us write $a_{i_1} = 3$, $a_{i_2} = 2$, $a_{i_3} = 5$, $a_{i_4} = 4$, $a_{i_5} = 1$, and $\alpha = u_0 3 u_1 2 u_2 5 u_3 4 u_4 1 u_5$. Figure 2.4 indicates the possible ways to sort these entries. In the case that $2, 3$ appear in stack R together, Lemma 2.4.1 ensures $2, 3$ do not appear together in stack L. In the other case, before 3 is moved into stack L, any tokens in stack L come from $u_0 u_1$. Thus when 3 is output the only tokens in stack L will be a_k with $k < i_2$. Lemma 2.4.1 ensures that $4, 5$ are not placed on top of 3 in stack L, so that the step that 3 is output they sit together in stack R. Lemma 2.4.2 also ensures that $4, 5$ are not placed on top of 2 in stack R as shown in Figure 2.4. $\square$

2.5 AN INFINITE ANTICHAIN

We use the following notation. If $\alpha = a_1 \dots a_n$ is a permutation of $12 \dots n$ and $m \in \mathbb{Z}$ then let α_m be the subpermutation obtained by adding m to each entry of α.



2541
3
R
L
541
2
3
R
L
2, 3 cannot appear together in stack L
2541
R
3
L
tokens under 3 must be from u_0u_1
541
2
R
3
L
1
4
5
2
R
3
L
violates Lemma 2.4.2
541
R
2
3
L
1
4
5
R
2
3
L
12
4
5
R
3
L
(1) and (2) are satisfied

Figure 2.4: Sorting 32541

For example $(1\ 2\ 3)_4 = 5\ 6\ 7$ and $13_6 = 19$.

We construct a family of permutations $\mathscr{G} = \{G_i \mid i \in \mathbb{N}\}$ as follows. Define

$$\begin{aligned} P &= 2\ 4\ 3\ 7\ 6\ 1 \\ x_j &= (10\ 5\ 9)_{6j} \\ y_j &= (13\ 12\ 8)_{6j} \\ S_i &= (14\ 15\ 11)_{6i} \\ G_i &= P\ x_0\ y_0\ x_1\ y_1\ \ldots\ x_i\ y_i\ S_i \end{aligned}$$

The first three terms are

$G_0 = 2\ 4\ 3\ 7\ 6\ 1\ (10\ 5\ 9)\ (13\ 12\ 8)\ 14\ 15\ 11,$

$G_1 = 2\ 4\ 3\ 7\ 6\ 1\ (10\ 5\ 9)\ (13\ 12\ 8)\ (16\ 11\ 15)\ (19\ 18\ 14)\ 20\ 21\ 17,$

$G_2 = P\ (10\ 5\ 9)\ (13\ 12\ 8)\ (16\ 11\ 15)\ (19\ 18\ 14)\ (22\ 17\ 21)(25\ 24\ 20)\ 26\ 27\ 23.$

A diagram [2] of the permutation G_2 is shown in Figure 2.5 which illustrates the general pattern.

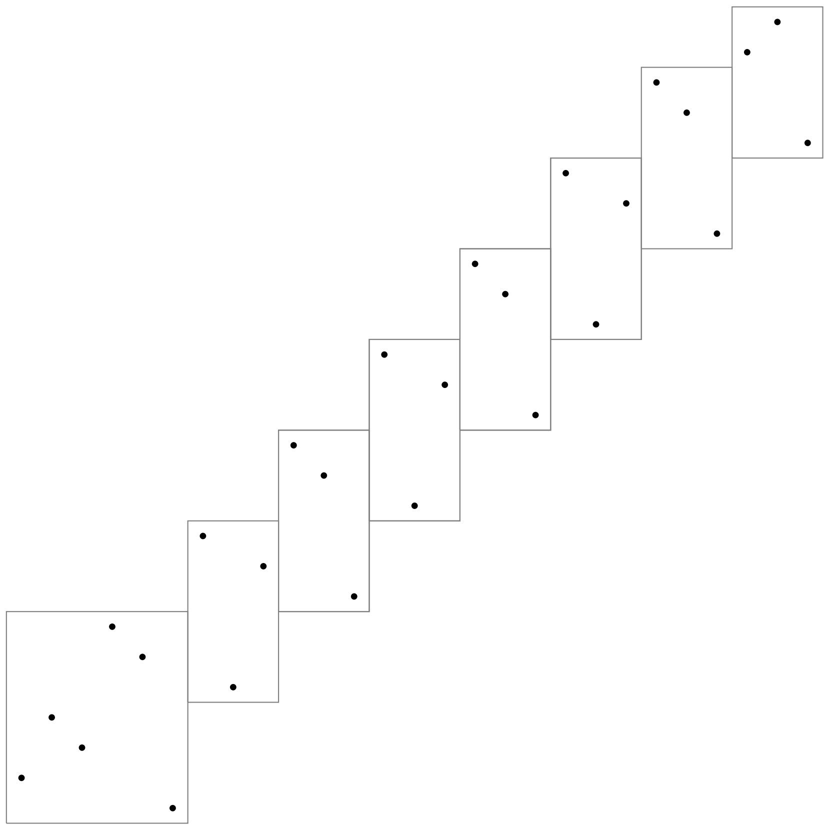


Figure 2.5: Permutation plot of $G_2 = 2\ 4\ 3\ 7\ 6\ 1\ x_0\ y_0\ x_1\ y_1\ x_2\ y_2\ 26\ 27\ 23$

We will prove that each G_i is an element of the basis of $S(3, \infty)$ for all $i \in \mathbb{N}$. Note that if we define x_{-1}, y_{-1} to be empty, $G_{-1} = 243761895$ is also an element of the basis. We noticed this and G_0 had a particular pattern which we could extend using $x_j y_j$. However, we exclude G_{-1} from our antichain to make the proofs simpler.

Proposition 2.5.1. *The permutation $G_i \notin S(3, \infty)$ for all $i \in \mathbb{N}$.*

Proof. Suppose for contradiction that G_i can be sorted by some sorting process. Since P is order isomorphic to 243651, by Lemma 2.4.3 in any sorting process $7, 6$ appear together in stack R. Next, 7 6 10 5 9 is order isomorphic to 32514 so by Lemma 2.4.4

[2]a map of the bijection G_2 from $[1, 27]$ to $[1, 27]$

since $7, 6$ appear together in stack R we must have that $10, 9$ appear together in stack L at some point in the process.

Now consider $x_jy_j = (10\ 5\ 9\ 13\ 12\ 8)_{6j}$, and assume that $10_{6j}, 9_{6j}$ both appear in stack L together. Since $(10\ 9\ 13\ 12\ 8)_{6j}$ is order isomorphic to 32541 by Lemma 2.4.5 $13_{6j}, 12_{6j}$ must be placed together in stack R and stay there until 10_{6j} is output.

Next consider $y_jx_{j+1} = (13\ 12\ 8\ 16\ 11\ 15)_{6j}$, and assume that $13_{6j}, 12_{6j}$ both appear in stack R together. Then since $(13\ 12\ 16\ 11\ 15)_{6j}$ is order isomorphic to 32514 by Lemma 2.4.4 we have that $16_{6j}, 15_{6j}$ appear together in stack L. Note that $16_{6j}, 15_{6j} = 10_{6(j+1)}, 9_{6(j+1)}$, so putting the above observations together we see that for all $0 \leqslant j \leqslant i$ we have that $10_{6j}, 9_{6j}$ both appear in stack L together and $13_{6j}, 12_{6j}$ appear together in stack R and stay there until 10_{6j} is output.

Now we consider the suffix

$$x_iy_iS_i = (10\ 5\ 9\ 13\ 12\ 8\ 14\ 15\ 11)_{6i}$$

where $10_{6i}, 9_{6i}$ are together in stack L. Lemma 2.4.5 tells us not only that $13_{6i}, 12_{6i}$ appear together in stack R and stay there until 10_{6i} is output, but that anything sitting underneath 10_{6i} in stack L comes *before* 9_{6i} in G_i, so in particular $14_{6i}, 15_{6i}$ are not underneath 10_{6i}. All possible processes to sort x_iy_iS are shown in Fig. 2.6. All possible sorting moves fail, which means G_i cannot be sorted. □

The idea of the preceding proof can be summarised informally as follows. The prefix P forces $7, 6$ to be together in stack R, then Lemmas 2.4.4 and 2.4.5 alternately imply that the $10_{6j}, 9_{6j}$ terms of x_j must be in stack L and the $13_{6j}, 12_{6j}$ terms of y_j must be in stack R. When we reach the suffix S_i the fact that certain entries are forced to be in a particular stack means we are unable to sort the final terms. We now show that if a single entry is removed from G_i, we can choose to place the $10_{6j}, 9_{6j}$ terms in stack R and $13_{6j}, 12_{6j}$ terms in stack L, which allows the suffix to be sorted.

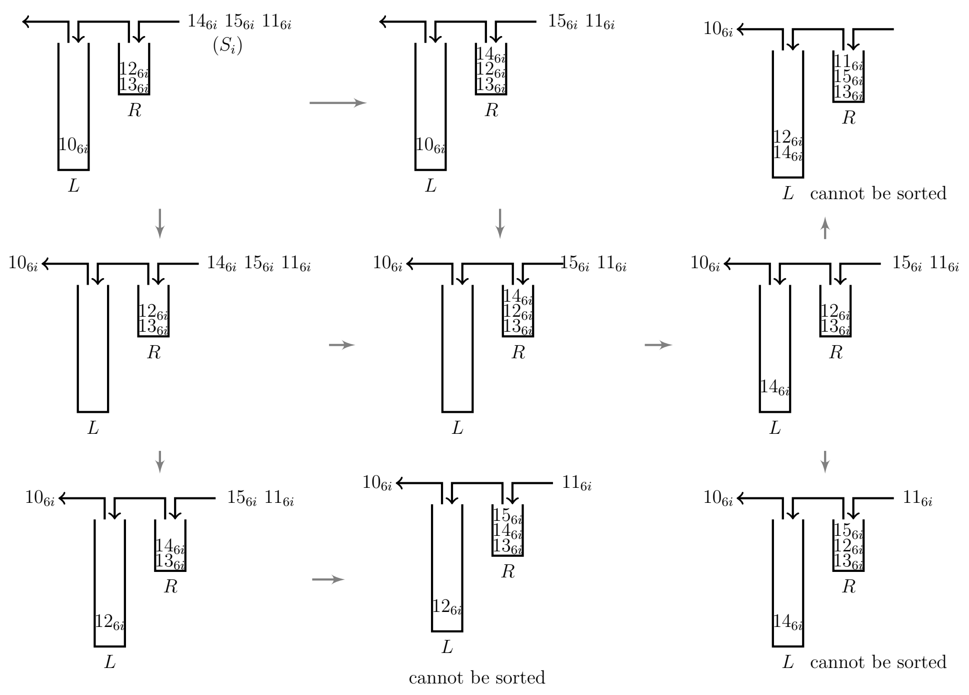


Figure 2.6: All possible ways to sort $x_i y_i S$

Lemma 2.5.2. *Let* $0 \leqslant j \leqslant i$. *If stack* R *contains one or both of* $10_{6j}, 9_{6j}$ *in ascending order, and* $y_j \dots y_i S_i$ *is to be input as in Fig. 2.7, then there is a sorting procedure to output all remaining entries in order.*

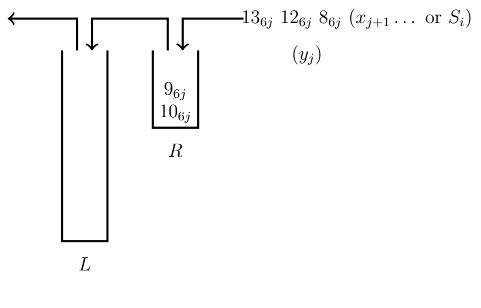


Figure 2.7: A sortable configuration

Proof. For $j < i$ move $13_{6j}, 12_{6j}$ into stack L, output $8_{6j}, 9_{6j}, 10_{6j}$, move $16_{6j} = 10_{6(j+1)}$ into stack R, output $11_{6j} = 5_{6(j+1)}$, output $12_{6j}, 13_{6j}$ from stack L and input $15_{6j} = 9_{6(j+1)}$ so that the configuration has the same form as Fig. 2.7 with j incremented by 1.

For $j = i$ the remaining input is $(13\ 12\ 8\ 14\ 15\ 11)_{6j}$. Put $13_{6i}, 12_{6i}$ in stack L in order, output $8_{6i}, 9_{6i}, 10_{6i}$, put $14_{6i}, 15_{6i}$ in stack R and output 11_{6i}, $12_{6i}, 13_{6i}$, move 15_{6i} into stack L and output 14_{6i} then 15_{6i}.

If one of $9_{6j}, 10_{6j}$ is missing, use the same procedure ignoring the missing entry. □

Lemma 2.5.3. *Let $0 \leqslant j \leqslant i$. If stack L contains one or both of $12_{6j}, 13_{6j}$ in ascending order, and $x_{j+1} \dots S_i$ (or just S_i if $j = i$) is to be input as in Fig. 2.8, then there is a sorting procedure to output all remaining entries in order.*

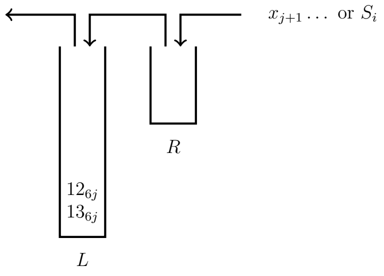


Figure 2.8: Another sortable configuration

Proof. If $j < i$ move $10_{6(j+1)}$ into stack R, output $5_{6(j+1)}, 12_{6j}, 13_{6j}$, move $9_{6(j+1)}$ to stack R to reach the configuration in Fig. 2.7, which we can sort by Lemma 2.5.2. If $j = i$ then the remaining input is just $S_i = (14\ 15\ 11)_{6i}$: move $14_{6i}, 15_{6i}$ to stack R, then output all entries.

If one of $12_{6j}, 13_{6j}$ is missing, use the same procedure ignoring the missing entry.

□

Proposition 2.5.4. *Let G_i' be a permutation obtained by removing a single entry from G_i. Then $G_i' \in S(3, \infty)$.*

Proof. We give a deterministic procedure to sort G_i'. There are three cases depending on from where the entry is removed.

Term removed from P. Let P' be the factor P with one entry removed. We claim that there is a sorting sequence for $P'x_0$ which outputs the smallest six items in order and leaves $10, 9$ in stack R. To show this we simply consider all cases.

1. If 1 is removed, $2, 4, 3$ can be output in order. After that, 7 is moved to stack R and then stack L. The entry 6 follows the same processes so that it stays on top of entry 7 in stack L. 10 in stack R, then $5, 6, 7$ output, and 9 placed on top of 10 in stack R.

2. If $2, 3$, or 4 are removed, write $P' = ab761$ with $a, b \in \{2, 3, 4\}$. Place a, b in stack R, move $7, 6$ into stack L, output 1, then output a, b in the correct order, then move 10 into stack R, output $5, 6, 7$ and move 9 into stack R.

3. If 6 or 7 is removed, write $P' = 243a1$ with $a \in \{7, 6\}$. Place $4, 3, 2$ in stack L in order, move a into stack R, output 1 then $2, 3, 4$, then move a into stack L, move 10 into stack R, output $5, a$ and move 9 into stack R.

Thus after inputting $P'x_0$ we have the configuration shown in Fig. 2.7 with $j = 0$, which we can sort by Lemma 2.5.2.

Term removed from $x_s, 0 \leqslant s \leqslant i$.

Input P leaving $6, 7$ in stack R, which brings us to the configuration in Fig. 2.9 with $j = 0$. Now assume we have input $P \ldots x_{j-1}y_{j-1}$ with $j \leqslant s$ (note the convention that x_{-1}, y_{-1} are empty) and the configuration is as in Fig. 2.9.

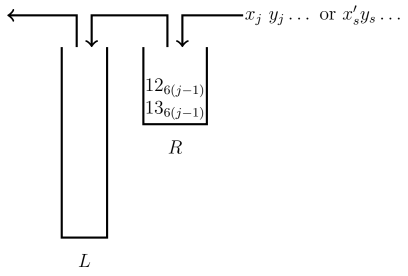


Figure 2.9: Configuration after $P \ldots x_{j-1}y_{j-1}$ is input

If $j < s$ we can input $x_j y_j$ into the stacks to arrive at the same configuration with j incremented by 1, as follows: move 10_{6j} to stack L, output $5_{6j}, 6_{6j} = 12_{6(j-1)}, 7_{6j} = 13_{6(j-1)}$, move 9_{6j} to stack L, move $13_{6j}, 12_{6j}$ to stack R, output $8_{6j}, 9_{6j}, 10_{6j}$.

If $j = s$, we proceed as follows:

1. If 5_{6s} removed, output $6_{6s} = 12_{6(s-1)}, 7_{6s} = 13_{6(s-1)}$, move $9_{6s}, 10_{6s}$ to stack R, to reach the configuration in Fig. 2.7 with $j = s$. From here the remaining entries can be sorted by Lemma 2.5.2.

2. If 10_{6s} is removed, output $5_{6s}, 6_{6s}, 7_{6s}$ and place 9_{6s} in stack R, to reach the configuration in Fig. 2.7 with $j = s$ and 10_{6s} missing. From here the remaining entries can be sorted Lemma 2.5.2.

3. If 9_{6s} is removed, move 6_{6s} to stack L, move 10_{6s} on top of 7_{6s} in stack R, output $5_{6s}, 6_{6s}$, move $13_{6s}, 12_{6s}$ into L, then output $8_{6s}, 10_{6s}$. This gives the configuration in Fig. 2.8 with $j = s$. From here the remaining entries can be sorted by Lemma 2.5.3.

Term removed from $y_s, 0 \leqslant s \leqslant i$ or S_i. Input Px_0 to reach the configuration in Fig. 2.10 with $j = 0$: move $2, 3, 4$ into stack L, $7, 6$ to R, output $1, 2, 3, 4$, move 10 into L, output $5, 6, 7$ then move 9 into L.

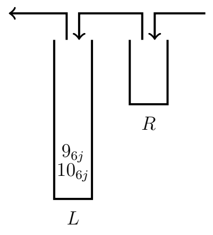


Figure 2.10: Configuration after $Px_0y_0 \ \ldots \ x_j$ is input

Now suppose we have input $Px_0y_0 \ \ldots \ x_j$ to reach the configuration in Fig. 2.10. If no entry is removed from y_j and $j < i$ then we can input y_jx_{j+1} to return to the configuration in Fig. 2.10 with j incremented by 1 as follows: move $13_{6j}, 12_{6j}$ to stack

R, output $8_{6j}, 9_{6j}, 10_{6j}$, move $10_{6(j+1)}$ to L, output $5_{6(j+1)} = 11_{6j}, 12_{6j}, 13_{6j}$, then move $9_{6(j+1)}$ to stack L.

If $j = s$ (y_s is removed):

1. If 8_{6s} is removed, output $9_{6s}, 10_{6s}$, move $13_{6s}, 12_{6s}$ to stack L to reach the configuration in Fig. 2.8, from which the remaining entries can be sorted by Lemma 2.5.3.

2. If $b \in \{13_{6s}, 12_{6s}\}$ is removed, place b in stack R, output $8_{6s}, 9_{6s}, 10_{6s}$, move b to stack L to reach the configuration in Fig. 2.8 with one of $12_{6s}, 13_{6s}$ removed, from which the remaining entries can be sorted a by Lemma 2.5.3.

If $j = i$ and the entry is removed from S_i, sort the remaining entries as follows:

1. If 11_{6i} is removed, place $13_{6i}, 12_{6i}$ into stack R, output $8_{6i}, 9_{6i}, 10_{6i}$, then $12_{6i}, 13_{6i}, 14_{6i}, 15_{6i}$.

2. If $b \in \{14_{6i}, 15_{6i}\}$ is removed, place $13_{6i}, 12_{6i}$ into stack R, output $8_{6i}, 9_{6i}, 10_{6i}$, move 12_{6i} into stack L, place b on top of 13_{6i} in stack R, output 11_{6i} then 12_{6i}, move b into stack L, output 13_{6i} then b.

□

Theorem 2.5.5. *The set of permutations that can be sorted by a stack of depth 3 and an infinite stack in series has an infinite basis.*

Proof. Proposition 2.5.1 shows that each G_i cannot be sorted, and Proposition 2.5.4 shows that no G_i can contain G_j for $j \neq i$ as a subpermutation since any subpermutation of G_i can be sorted. Thus $\mathscr{G} = \{G_i \mid i \in \mathbb{N}\}$ is an infinite antichain in the basis for $S(3, \infty)$. □

2.6 FROM FINITE TO INFINITELY BASED

Let $\mathscr{B}_t$ be the basis for $S(t,\infty)$ for $t \in \mathbb{N}_+$. Lemma 1 in [23] is modified by taking the consideration that permutations passing through the stacks must be sorted. Then, we have the following:

Lemma 2.6.1. *If $\sigma \in \mathscr{B}_t$ has length n then either σ or $(213)_n\sigma$ belongs to $\mathscr{B}_{t+1}$.*

Proof. If $\sigma \notin S(t+1,\infty)$ then since $\sigma \in \mathscr{B}_t$, deleting any entry gives a permutation in $S(t,\infty) \subseteq S(t+1,\infty)$, so $\sigma \in \mathscr{B}_{t+1}$. Else $\sigma \in S(t+1,\infty)$. In any sorting process for $(213)_n\sigma$ the entries $1_n, 2_n, 3_n$ cannot appear together in stack L, so at least one entry must remain in stack R which means we must sort σ with stack R of depth at most t, which is not possible, so $(213)_n\sigma$ cannot be sorted. If we remove an entry of the prefix then the two entries $a, b \in \{1_n, 2_n, 3_n\}$ can be placed in stack L in order, leaving stack R depth $t+1$ so the permutation can be sorted, and if an entry is removed from σ then since $\sigma \in \mathscr{B}_t$ it can be sorted with R having one space occupied. $\square$

Theorem 2.6.2. *The set of permutations that can be sorted using a stack of depth $t \in \mathbb{N}_+$ and an infinite stack in series is finitely based if and only if $t \in \{1, 2\}$.*

Proof. We have $|\mathscr{B}_1| = 1$ and $|\mathscr{B}_2| = 20$ [37, 23]. Theorem 2.5.5 shows that $\mathscr{B}_3$ is infinite. Lemma 2.6.1 implies if $\mathscr{B}_t$ is infinite then so is $\mathscr{B}_{t+1}$. $\square$

2.6.1 An explicit antichain for $S(t,\infty)$

The antichain $\mathscr{G} = \{G_i \mid i \in \mathbb{N}\}$ can be extended to become an infinite antichain in the basis of $S(t,\infty)$ for $t \geqslant 3$.

Lemma 2.6.3. *The explicit antichain of $S(t,\infty)$ for $t \geqslant 3$ is the set $\mathscr{G}_t = \{G_{i,t}\}$, where $G_{i,t} = P(x_0y_0)\dots(x_iy_i)(14\ 15\ 16\ \dots\ 12_t\ 11)_{6i}$*

Proof. Recall that the Lemmas 2.4.1, 2.4.2, 2.4.3, 2.4.4 and 2.4.5 that were used to prove Proposition 2.5.1 as well as Proposition 2.5.4 are not specifically written for $S(3,\infty)$

but for every $t \geqslant 3$ in $S(t,\infty)$. Moreover, note that $G_{i,t}$ is similar to the G_i of $S(3,\infty)$ except the length (number of entries) of the suffix in $G_{i,t}$ is dependent on the depth of stack R. So, the similar arguments to Propositions 2.5.1 and 2.5.4 are valid for $S(t,\infty)$ before any entry from the suffix $(14\ 15\ 16\ \ldots\ 12_t\ 11)_{6i}$ is input.

We start by proving $G_{i,t}$ is not sortable by $S(t,\infty)$. Based on the same argument in Proposition 2.5.1, when the entries from $P(x_0y_0)\ldots(x_iy_i)$ are output before any entry from the suffix $(14\ 15\ 16\ \ldots\ 12_t\ 11)_{6i}$ is input, $S(t,\infty)$ will have the entries 12_{6j} and 13_{6j} appear together in stack R such as in Figure 2.11.

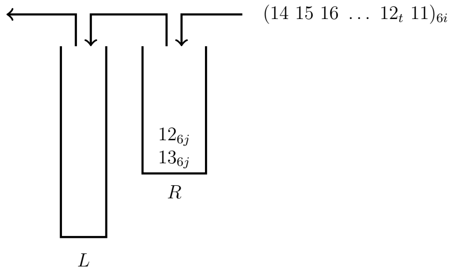


Figure 2.11: Configuration before $(14\ 15\ 16\ \ldots\ 12_t\ 11)_{6i}$ is input

At this configuration, there are t-1 entries before (to the left of) the next smallest entry 11_{6i} in the suffix $(14\ 15\ 16\ \ldots\ 12_t\ 11)_{6i}$ and only t-2 spaces left in stack R.

(Case 1) Suppose both the entries 12_{6j} and 13_{6j} are kept to stay in stack R until the entry 11_{6i} is input. Then at least one entry from the suffix need to move to stack L else there will be not space for the entry 11_{6i} to move into stack R. Besides that, some entries from the suffix need to move to stack R with the largest entry $(12_t)_{6i}$ stay on top of the stack or else the entries in stack L are not in ascending order when read from top to bottom and the sorting process will fail as in Lemma 2.4.1.

After all the entries before 11_{6i} are input, the entry 11_{6i} can move to stack R and then stack L before getting output. Once 11_{6i} is output, the next smallest element to be output is the entry 12_{6j}. Since the largest entry $(12_t)_{6i}$ is above 12_{6j}, then it has to move into stack L to allow 12_{6j} to be output. Since entries in stack L are smaller than

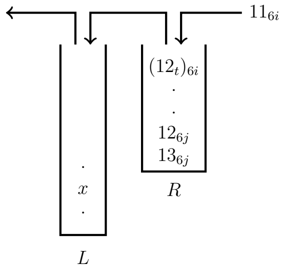


Figure 2.12: Configuration before 11_{6i} is input such that x is an entry from suffix

$(12_t)_{6i}$, the stack L is not in ascending order. Based on Lemma 2.4.1, the permutation $G_{i,t}$ is not sortable.

(Case 2) Suppose the entry 12_{6j} stays in stack L and 13_{6j} stays in stack R until the entry 11_{6i} is input. Note that both entries cannot appear together in stack L or else 13_{6j} will be above 12_{6j} causing stack L not in ascending order when read from top to bottom. Then, all the entries from the suffix before 11_{6i} is input have to stay in stack R since each entry is bigger than 12_{6j}. At this stage of configuration, the number of entries in stack R is t since there are t-1 entries stay above 13_{6j}. Since stack R has only depth t, then there is no space for the entry 11_{6i} to be input. So, $G_{i,t}$ is not sortable.

Let $G'_{i,t}$ be a permutation obtained by removing a single entry from $G_{i,t}$. We will prove that $G'_{i,t}$ is sortable by $S(t, \infty)$. Similar to the argument in Proposition 2.5.4, removing any entry from P, x_s or y_s such that $0 \leqslant s \leqslant i$ can end up to the configuration as in Figure 2.13 before any entry from suffix is input.

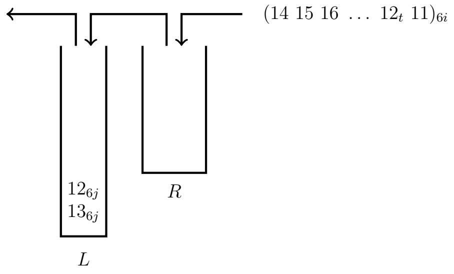


Figure 2.13: Configuration after an entry is deleted from P, x_s or y_s

At this stage of configuration, there are t empty spaces in stack R and there are only t entries in suffix. So, every entry from suffix can move to stack R with the next smallest entry 11_{6i} stays on top of the stack. Then output the entry 11_{6i}. At this stage of configuration, entries in stack L are in ascending order while entries in stack R are in descending order when read from top to bottom as in Figure 2.14. So, $G'_{i,t}$ can be sorted by output the entry 12_{6j} and 13_{6j} first. Then, move each entry in stack R into stack L and the order is increasing. After that, output each entry one by one from stack R until $G'_{i,t}$ is sorted.

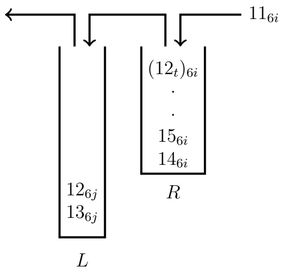


Figure 2.14: Configuration before 11_{6i} is input from suffix

Then $G_{i,t}$ cannot be sorted by $S(t,\infty)$ and $G_{i,t}$ cannot contain $G_{j,t}$ for $j \neq i$ as a subpermutation since any subpermutation of $G_{i,t}$ can be sorted. So, $\mathscr{G} = \{G_{i,t} \mid i \in \mathbb{N}\}$ is the antichain for $S(t,\infty)$ such that $t \geqslant 3$. $\square$

2.7 CONCLUSION

The main result from this chapter answers the characterisation problem for sorting with a finite and an infinite stack in series. The result has been published in *Lecture Notes in Computer Science Vol. 10792* [24]. The open problem that is yet to be solved is the enumeration problem which we had attempted to solve but with no breakthrough. By using language-theoretic approach as in [25], we tried to describe the permutations in $S(3,\infty)$ as some formal language belonging to a class such as context-free, ET0L, EDT0L or indexed, which could help us determine the generating function of the set.

Unfortunately, we did not get very far with this technique because we were not able to find a bijection between $S(3,\infty)$ and a formal language class. However, we did observe that the set of *encoded permutations* in the avoidance set is most likely not a context free language. More details are provided in Chapter 4.

CHAPTER 3
ON PERMUTATIONS SORTED BY k PASSES THROUGH A DETERMINISTIC POP-STACK

3.1 AIM

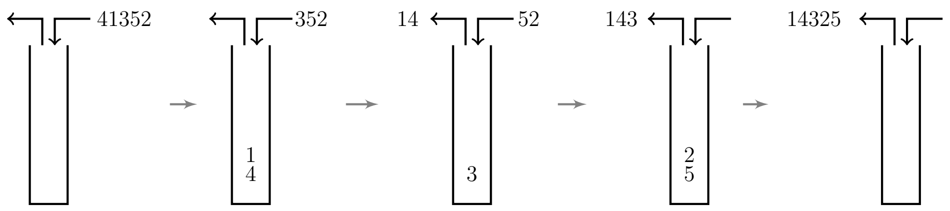
In this chapter we consider permutations sortable by k passes through a deterministic pop-stack. We show that for any $k \in \mathbb{N}$ the set is characterised by finitely many forbidden patterns, answering a question of Claesson and Guðmundsson.

Our characterisation demands a more precise definition than in previous literature of what it means for a permutation to avoid a set of barred and unbarred patterns. We propose a new notion called *PB-containment* and prove some useful results about this notion.

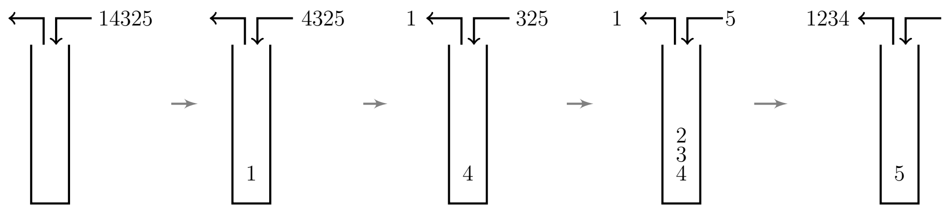
3.2 INTRODUCTION

Recall that a *pop-stack* is a variation of sorting machine which operates as follows: at each step it can either push one token from the input stream onto the top of the stack, or else pop the *entire stack contents* to the output stream. Here, we consider the tokens to be distinct real numbers with the usual ordering and the pop-stack is a deterministic pop-stack. A *deterministic* pop-stack always performs the push move unless the token on the top of the stack is smaller in value that the token to be pushed from the input, or if there is no further input. Observe that by definition the stack remains ordered from smallest on top to largest on the bottom during the operation of a deterministic pop-stack. For example, 41352 can be sorted by two passes (Figure 3.1).

Let $p_1(\alpha)$ denote the sequence obtained by passing a sequence α through a deterministic pop-stack once, and define $p_k(\alpha) = p_{k-1}(p_1(\alpha))$. For example $p_1(41352) = 14325$ and $p_2(41352) = 12345$. We say α is *k-pass deterministic pop-stack sortable* if $p_k(\alpha)$ is an increasing sequence.



(a) First pass.



(b) Second pass.

Figure 3.1: Sorting 41352 with a 2-pass pop-stack

In this chapter, we show that k-pass deterministic pop-stack sortable permutations are characterised by a finite list of forbidden (usual and barred) patterns for all $k \in \mathbb{N}$. In order to prove this, we realised the current notions of what it means to avoid a set of barred and unbarred patterns would not suffice. Because of this we have introduced a new notion called *PB-containment* which we define in Section 3.3. We show that this is exactly the right notion for k-pass pop-stack sorting especially when $k > 3$

The proof of our characterisation is in the form of a constructive algorithm, which we can be implemented to obtain a finite list of patterns [1] for each $k > 1$. For $k = 2$ our list includes the patterns obtained by Pudwell and Smith [45], but also contains many additional (redundant) patterns. We give several lemmas which show how to remove some redundant patterns in general but further lemmas are needed to obtain a "minimal" list.

We make the following observations. First, our present result is in contrast to the

[1]Note that the list is a huge list with many redundant patterns that can be removed from the set without affecting the avoidance set. Thus, it is not a minimal list because there might be another smaller list that characterised the same pattern avoidance set. More details will explained in later section in this chapter.

usual (nondeterministic) stacks-in-series model where in many cases no finite pattern-avoidance characterisation is possible due to the existence of infinite antichains , as we saw in the previous chapter. Second, the operation of a pop-stack is related to a classical sorting: "bubble-sort" is exactly sorting by arbitrarily many passes through a pop-stack of depth 2. Third, pop-stacks are a natural model for genome rearrangement [45].

3.2.1 B-sequences

Definition 10 (B-sequence)**.** Let X^* denote the set of all finite sequences of letters from some alphabet X, and $|u|$ the length of a sequence $u \in X^*$. Consider $\mathbb{N} = \{1, 2, 3, \dots\}$ as an infinite alphabet of letters and define a disjoint copy $\overline{\mathbb{N}} = \{\bar{i} \mid i \in \mathbb{N}\}$ of *barred* letters. Define a map unbar: $\mathbb{N} \cup \overline{\mathbb{N}} \to \mathbb{N}$ by $\text{unbar}(\bar{i}) = i = \text{unbar}(i)$, which extends to $(\mathbb{N} \cup \overline{\mathbb{N}})^*$ entry-by-entry.

A finite sequence $\beta \in (\mathbb{N} \cup \overline{\mathbb{N}})^*$ is called a *B-sequence* if $\text{unbar}(\beta)$ is a permutation. For example $4\bar{1}352$ is a B-sequence, since $\text{unbar}(4\bar{1}352) = 41352$, whereas $31\bar{1}2$ is not a B-sequence. [2]

Define a second map $\nu\colon \mathbb{N} \cup \overline{\mathbb{N}} \to \mathbb{N}$ by $\nu(\bar{i}) = \varepsilon$ (the empty letter) and $\nu(i) = i$, which also extends to sequences entry-by-entry. Then define removebar: $(\mathbb{N} \cup \overline{\mathbb{N}})^* \to \mathbb{N}^*$ by $\text{removebar}(\alpha) = \text{red}(\nu(\alpha))$. For example $\text{removebar}(4\bar{1}352) = 3241$.

Informally we will say that $\text{unbar}(\beta)$ is the permutation obtained by erasing the bar above any barred token in β, and $\text{removebar}(\beta)$ a permutation obtained by deleting barred tokens then reducing. Note that a B-sequence which satisfies $\beta = \text{unbar}(\beta)$ can either be considered as a B-sequence or a permutation, and depending on the context this will be an important distinction. If F is any set of B-sequences, let $F_p = \{\beta \in F \mid \beta = \text{unbar}(\beta)\}$ and $F_b = F \setminus F_p$ be the sets of *unbarred* and *barred* elements of F

[2] In [35] and elsewhere the term *barred pattern* is used, although the string $1\bar{4}3\bar{4}2$ is also referred to as a barred pattern [35, p. 310]. So to avoid confusion we use the new term "B-sequence".

respectively.

Let $\mathscr{B} \subseteq (N \cup \overline{N})^*$ denote the set of all B-sequences. A *subsequence* of a B-sequence $\beta = b_1 \dots b_r$, $b_i \in \mathbb{N} \cup \overline{\mathbb{N}}$ is a sequence $b_{i_1} \dots b_{i_s}$ where $1 \leqslant i_1 < \dots < i_s \leqslant r$.

3.3 NEW NOTION OF BARRED PATTERN AVOIDANCE

Definition 11 (PB-containment)**.** Let σ be a permutation and $F \subseteq \mathscr{B}$ a set of B-sequences. We say that σ *PB-contains* F if there exists $\beta \in F$ and a subpermutation γ of σ such that

– $\gamma \sim \text{removebar}(\beta)$ and

– for all $\alpha \in F_b$, if $\gamma \sim \text{removebar}(\alpha)$ then γ is not subpermutation of $\delta \sim \text{unbar}(\alpha)$ of σ.

Informally, the second condition says that the forbidden subpermation $\gamma \sim \text{unbar}(\beta)$ of σ can potentially be "saved" by some $\alpha \in F_b$ (α possibly different to β) if γ is a subpermutation of some $\delta \sim \text{unbar}(\alpha)$ where δ is itself a subpermutation of σ. The requirement that $\gamma \sim \text{removebar}(\alpha)$ is somehow optional, but we found it convenient to include. In this way, PB-containment differs considerably from existing notions of containing barred patterns.

We admit that "PB-contains" sounds technical, but we persist with it because the nature of avoiding a set of barred patterns turns out to be very subtle, as we describe in Section 3.4. The present definition captures precisely the properties we need to characterise k-pass pop-stack sortable permutations. P, B stands for "permutation" versus "set of B-sequences".

Example 3.3.1. Let $F = \{3241, 4\bar{1}352\}$ and consider the permutations $\sigma_1 = 143562$ and $\sigma_2 = 152463$. Then

– σ_1 has the subpermutation $4352 \sim 3241$ which is not part of a longer subpermutation of σ_1 order-isomorphic to $\text{unbar}(4\bar{1}352) = 41352$, so σ_1 PB-contains

F.

– σ_2 has subpermutation $5463 \sim 3241$, however 5463 is a subpermutation of $52463 \sim \text{unbar}(4\bar{1}352) = 41352$ (so σ_2 (or strictly speaking its subpermutation 5462) is "saved" by the existence of this other pattern). Since there are no other subpermutations of σ_2 that are order-isomorphic to either 3241 or $\text{removebar}(4\bar{1}352) = 3241$, then σ_2 does not PB-contain F.

Further examples showing the subtlety of PB-containment are given below as Examples 3.3.2–3.6.4.

We remark that 32451 is not 3-pass pop-stack sortable, however both 4631572 and 4731562 are, so Example 3.3.3 is relevant when we try to characterise 3-pass pop-stack sortable permutations. See Section 3.4.

We say σ *PB-avoids* F if it does not PB-contain F. Using propositional logic, we can express this as follows:

Definition 12 (PB-avoidance)**.** Let σ be a permutation and $F \subseteq \mathscr{B}$ a set of B-sequences. We say that σ *PB-avoids* F if for every $\beta \in F$ and every subpermutation γ of σ either

– $\gamma \not\sim \text{removebar}(\beta)$, or

– there exists $\alpha \in F_b$ such that $\gamma \sim \text{removebar}(\alpha)$ and γ is a subpermutation of $\delta \sim \text{unbar}(\alpha)$ of σ.

For any set of B-sequences F, let $\text{Av}_B(F)$ denote the set of permutations that PB-avoid F. We call $\text{Av}_B(F)$ the *PB-avoidance set* of F. If F has no barred elements (so $F = F_p$) then the second condition in Definitions 11–12 is vacuous and $\text{Av}_B(F) = \text{Av}(F)$.

Example 3.3.2. Let $F = \{1\bar{2}, 2\bar{1}\}$. Suppose σ PB-contains F. If σ contains some $\gamma \sim \text{removebar}(1\bar{2}) = 1$ (so γ can be any entry of σ), then the second condition in

Definition 11 say we must check every $\alpha \in F_b$ to see that γ is not contained in a subpermutation $\delta \sim 12$ or $\delta \sim 21$ of σ. This means σ cannot have more than one token. The same argument also holds for σ containing some $\gamma \sim \text{removebar}(2\bar{1}) = 1$. Thus, σ of length at least two cannot PB-contains F. So, $\text{Av}_B(F) = S^\infty \setminus \{1\}$.

Example 3.3.3. [3] Let $\beta_1 = 4\bar{6}3\bar{1}572, \beta_2 = 4\bar{7}3\bar{1}562$, $F = \{\beta_1, \beta_2\}$ and consider the permutation $\sigma = 4731562$.

- σ has subpermutation $43562 \sim \text{removebar}(\beta_1)$, however there exists $\alpha = \beta_2 \in F_b$ such that $43562 \sim \text{removebar}(\beta_2)$ is a subpermutation of $4731562 = \text{unbar}(\beta_2)$. Since there are no other subpermutations of σ that are order-isomorphic to $\text{removebar}(\beta_1) = \text{removebar}(\beta_2)$, then σ does not PB-contain F. Note that it is β_2 (and not β_1) that "saves" σ because $43562 \sim \text{removebar}(\beta_1)$ is not a subpermutation of 4631572=unbar(β_1) in σ. In short, σ PB-contains $\{\beta_1\}$ but does not PB-contain F because of the existence of β_2 in F.

3.4 ISSUES IN EXISTING NOTION OF BARRED PATTERN AVOIDANCE

In [35, Definition 1.2.3] a permutation σ is said to avoid a barred pattern β if each occurrence of removebar(β) in σ (if any) is a part of an occurrence of unbar(β) in σ. There are two issues with this definition. Firstly, as written, this does not agree with the usual pattern avoidance when β has no bar tokens. For example, $\sigma = 21$ obviously does not avoid $\beta = 21$ in the usual sense of pattern avoidance, but if β is considered as a barred pattern then σ avoids β since there exist a subsequence, $\kappa = 21 = \text{removebar}(\beta)$ in σ and κ is part of the occurrence of $\text{unbar}(\beta) = 21$ in σ.

Secondly and more seriously, in applications such as [45, 55] some set of permutations S is characterised by being those permutations avoiding some list of barred and

[3] $4\bar{6}3\bar{1}572$ and $4\bar{7}3\bar{1}562$ are taken from the set of forbidden patterns for permutations sortable by 3-pass pop-stack which we obtained by a computer program. See the full list in Section **??**

unbarred patterns, where, as we understand it, this means that each permutation in S must avoid every pattern individually. Explicitly, Tenner [52] defines that if P is a collection of pattern, then $Av(P)$ means is the set of permutations simultaneously avoiding all patterns in P:

$$Av(P) = \bigcap_{p \in P} Av(p)$$

If this interpretation were used for Example 3.3.3, we would say that 4731562 does not avoid any list containing $4\bar{6}3\bar{1}572, 4\bar{7}3\bar{1}562$ since it fails to avoid the barred pattern $4\bar{6}3\bar{1}572$. Our definition of PB-avoids says that even though some permutation may not avoid some pattern in a list of barred and unbarred patterns, it might be *saved* by another barred pattern in the list. This interpretation is what is needed to characterise 3-pass pop-stack sortable permutations, since both $4631572, 4731562$ are 3-pass pop-stack sortable and 32451 is not. For the applications in [45, 55] either interpretation is correct since the sets of barred and unbarred patterns to be avoided have no "overlap": in the case of [45], the two barred patterns have removebar equal to 2341 and 4312 which are both different to the unbarred patterns in their list; and in the case of [55] there is only one barred pattern whose removebar is different to the unbarred pattern.

3.5 SETS OF B-SEQUENCES

We make the following definition to compare different sets of B-sequences, in analogy with usual pattern avoidance.

Definition 13 (B-isomorphic)**.** Let $F, G \subseteq \mathscr{B}$. We say that F and G are *B-isomorphic*, written $F \sim_B G$, if $\mathrm{Av}_B(F) = \mathrm{Av}_B(G)$.

For example, $\{3\bar{5}412\} \not\sim_B \{354\bar{1}2\}$ (since for example $3412 \in \mathrm{Av}_B(354\bar{1}2) \setminus \mathrm{Av}_B(3\bar{5}412)$), but it can be shown (by Lemma 3.5.1 next) that $\{3\bar{5}412\} \sim_B \{35\bar{4}12\}$.

Our definition characterises "sameness" of sets of B-sequences according to the permutations which PB-avoid them. Following Example 3.3.2 we have $\{1\bar{2}, 2\bar{1}\} \sim_B$

$\{1\bar{2},\bar{2}1\} \sim_B \{\bar{1}2,\bar{2}1\} \sim_B \{\bar{1}2,2\bar{1}\}$ since in each case the PB-avoidance set is $S^\infty \setminus \{1\}$.

The following lemma gives a method to check B-isomorphism for singleton sets without having to construct entire PB-avoidance sets.

Lemma 3.5.1. *Let α, β be B-sequences. Then $\{\alpha\} \sim_B \{\beta\}$ if and only if* $\mathrm{unbar}(\alpha) \sim \mathrm{unbar}(\beta)$ *and* $\mathrm{removebar}(\alpha) = \mathrm{removebar}(\beta)$*.*

Proof. Suppose that $\mathrm{unbar}(\alpha) \sim \mathrm{unbar}(\beta)$ and $\mathrm{removebar}(\alpha) = \mathrm{removebar}(\beta)$. Let $\sigma \notin \mathrm{Av}_B(\beta)$, then σ contains a subpermutation $\gamma \sim \mathrm{removebar}(\beta)(= \mathrm{removebar}(\alpha))$ that is not part of a longer subpermutation $\delta \sim \mathrm{unbar}(\beta)(\sim \mathrm{unbar}(\alpha))$ in σ, which means $\sigma \notin \mathrm{Av}_B(\alpha)$. A similar argument gives $\sigma \notin \mathrm{Av}_B(\alpha)$ implies $\sigma \notin \mathrm{Av}_B(\beta)$, so $\alpha \sim_B \beta$.

For the other direction, if $\mathrm{removebar}(\alpha) \neq \mathrm{removebar}(\beta)$, then either one is shorter, or if they have the same length, they are not order-isomorphic. Without loss of generality say $\sigma = \mathrm{removebar}(\beta)$ is not longer than $\mathrm{removebar}(\alpha)$. Then $\sigma \notin \mathrm{Av}_B(\beta)$. However $\sigma \in \mathrm{Av}_B(\alpha)$ because no subpermutation γ of σ is order-isomorphic to $\mathrm{removebar}(\alpha)$, so $\{\alpha\} \not\sim_B \{\beta\}$.

Otherwise we have $\mathrm{removebar}(\alpha) = \mathrm{removebar}(\beta)$ and $\mathrm{unbar}(\alpha) \not\sim \mathrm{unbar}(\beta)$. Let us consider the cases separately.

1. If one has no barred tokens, say β, then α must contain barred tokens and be longer. We have $\sigma = \mathrm{unbar}(\alpha) \in \mathrm{Av}_B(\alpha)$, but σ contains $\gamma = \mathrm{removebar}(\beta)$ which is not saved by β, so $\sigma \notin \mathrm{Av}_B(\beta)$.

2. Else both have barred tokens.

    (a) If one is shorter than the other, say $\sigma = \mathrm{unbar}(\beta)$ is shorter than $\mathrm{unbar}(\alpha)$, then $\sigma \in \mathrm{Av}_B(\beta)$, but σ PB-contains $\{\alpha\}$ because it contains $\mathrm{removebar}(\alpha)$ which cannot be part of a subsequence $\delta \sim \mathrm{unbar}(\alpha)$ because δ would be longer than σ itself.

(b) Else $\mathrm{unbar}(\alpha), \mathrm{unbar}(\beta)$ have the same length, and $\mathrm{unbar}(\alpha) \not\sim \mathrm{unbar}(\beta)$. Let $\sigma = \mathrm{unbar}(\beta)$ so $\sigma \in \mathrm{Av}_B(\beta)$. Now σ contains $\mathrm{removebar}(\alpha)$ which is not part of a subsequence $\delta \sim \mathrm{unbar}(\alpha)$ because the only permutation having the same length as $\mathrm{unbar}(\alpha)$ is σ itself, and $\sigma = \mathrm{unbar}(\beta) \not\sim \mathrm{unbar}(\alpha)$. Thus $\sigma \notin \mathrm{Av}_B(\alpha)$.

Thus in all cases we have $\{\alpha\} \not\sim_B \{\beta\}$. □

It follows from the lemma that for example $\{3\bar{5}412\} \sim_B \{35\bar{4}12\}$.

3.6 REMOVING REDUNDANT B-SEQUENCES

In this section we give several lemmas which tell us when we may remove elements from a set F without changing its PB-avoidance set. We make no claim that the lemmas in this section are exhaustive, for example there are sets $F, G \subseteq \mathscr{B}$ for which none of the three lemmas applies but they have the same PB-avoidance sets for other reasons. See Example 3.6.4 below for an example of this. However, the rules we give in this section are useful in Theorem 3.8.1 to reduce the size of the finite sets we obtain there.

We start with a useful (reflexive and transitive) relation on $\mathscr{B}$.

Definition 14. Let $\kappa, \lambda \in \mathscr{B}$. We say $\kappa \preceq_B \lambda$ if

- $\mathrm{removebar}(\kappa) = \mathrm{removebar}(\lambda)$, and
- $\mathrm{unbar}(\kappa) \leqslant \mathrm{unbar}(\lambda)$ ($\mathrm{unbar}(\kappa)$ is order-isomorphic to a subpermutation of $\mathrm{unbar}(\lambda)$).

For example:

– $1\bar{2} \preceq_B 1\bar{3}\bar{2}$

– $12 \preceq_B 1\bar{3}2$

– $3\bar{5}412 \preceq_B 35\bar{4}12$

– $35\bar{4}12 \preceq_B 3\bar{5}412$

It follows from Lemma 3.5.1 that $\kappa \preceq_B \lambda$ and $\lambda \preceq_B \kappa$ implies $\{\kappa\} \sim_B \{\lambda\}$. For convenience in our proofs we formalise "because" and "saves". We will also denote γ is a subpermutation of σ by $\gamma <_{\text{subperm}} \sigma$.

Definition 15. Let $\beta \in F \subseteq \mathscr{B}$. A permutation σ PB-contains F *because of* $\beta \in F$ and $\gamma <_{\text{subperm}} \sigma$ if:

- $\gamma \sim \text{removebar}(\beta)$, and
- for all $\alpha \in F_b$, if $\gamma \sim \text{removebar}(\alpha)$ then γ is not subpermutation of $\delta \sim \text{unbar}(\alpha)$ of σ.

An element $\alpha \in F_b$ *saves* $\gamma <_{\text{subperm}} \sigma$ if $\gamma \sim \text{removebar}(\alpha)$ and γ is not subpermutation of any $\delta \sim \text{unbar}(\alpha)$ of σ.

Of course in general σ could PB-contain F because of several different β and γ. Here are the first three lemmas.

Lemma 3.6.1. *Let $F \subseteq \mathscr{B}$ with $\kappa, \lambda \in F_b$. If $\kappa \preceq_B \lambda$ then $F \sim_B F \setminus \{\lambda\}$.*

Lemma 3.6.2. *Let $F \subseteq \mathscr{B}$ with $\kappa \in F_p, \lambda \in F_b$. If $\kappa \preceq_B \lambda$ then $F \sim_B F \setminus \{\kappa\}$.*

Lemma 3.6.3. *Let $F \subseteq \mathscr{B}$ with $\kappa, \lambda \in F_p$. If $\kappa \leqslant \lambda$ and for all $\alpha \in F_b$, $\kappa \not\preceq_B$* removebar(α)*, then $F \sim_B F \setminus \{\lambda\}$.*

Proof of Lemma 3.6.1. First, suppose σ PB-contains F. Then this is either because of $\beta \in F \setminus \{\kappa, \lambda\}$, or κ, or λ.

– In the first case, there is some $\beta \in F \setminus \{\kappa, \lambda\}$ and $\gamma \sim \text{removebar}(\beta)$ a subpermutation of σ with γ not contained in any $\delta \sim \text{unbar}(\alpha)$ for all $\alpha \in F_b$ where δ is a subpermutation of σ. If so, then γ is not contained in any $\delta \sim \text{unbar}(\alpha)$ for any $\alpha \in (F_b \setminus \{\lambda\})$, so σ PB-contains $F \setminus \{\lambda\}$.

  – Else, σ contains a subpermutation $\gamma \sim \text{removebar}(\lambda) = \text{removebar}(\kappa)$, and γ is not a subpermutation of any subpermutation δ of σ with $\delta \sim \text{unbar}(\alpha)$ for any $\alpha \in F_b$, so in particular γ is not contained in any $\delta \sim \text{unbar}(\alpha)$ for any $\alpha \in (F_b \setminus \{\lambda\})$, so σ PB-contains $F \setminus \{\lambda\}$. Thus σ PB-contains $F \setminus \{\lambda\}$.

This shows $\text{Av}_B(F \setminus \{\lambda\}) \subseteq \text{Av}_B(F)$.

Now suppose σ PB-contains $F \setminus \{\lambda\}$ but does not PB-contain F. This means that there must be some subpermutation γ of σ with $\gamma \sim \text{removebar}(\beta)$ for some $\beta \in F \setminus \{\lambda\}$ which is not saved by any $\alpha \in (F \setminus \{\lambda\})_b$, but is saved by $\lambda \in F_b$. This means γ is a subpermutation of $\delta \sim \text{unbar}(\lambda)$ of σ. But this means γ is also a subpermutation of $\delta' \sim \text{unbar}(\kappa)$ since $\text{unbar}(\kappa) \leqslant \text{unbar}(\lambda)$, which contradicts that σ PB-contains $F \setminus \{\lambda\}$. This shows

$$\text{Av}_B(F) \cap \overline{\text{Av}_B(F \setminus \{\lambda\})}$$

is empty, hence the result. $\square$

Proof of Lemma 3.6.2. Suppose σ PB-contains F, then this is either because of $\beta \in F \setminus \{\kappa\}$, or because of κ. In the first case we have σ PB-contains $F \setminus \{\kappa\}$. In the second case σ contains a subpermutation $\gamma \sim \kappa = \text{removebar}(\lambda)$ and for all $\alpha \in F_b$, γ is not a subpermutation of a subpermutation $\delta \sim \text{unbar}(\alpha)$ of σ. This means σ PB-contains F because of λ as well, so σ PB-contains $F \setminus \{\kappa\}$. Thus by contrapositive we have shown $\text{Av}_B(F \setminus \{\kappa\}) \subseteq \text{Av}_B(F)$.

Now suppose σ PB-contains $F \setminus \{\kappa\}$ because of β. Since $\kappa \in F_p$ it can play no role in saving σ, this means σ PB-contains F also because of β, so $\text{Av}_B(F) \subseteq \text{Av}_B(F \setminus \{\kappa\})$. $\square$

Proof of Lemma 3.6.3. Suppose σ PB-contains F. Then this is either because of λ, or not. If it is because of $\beta \neq \lambda$ then since $\lambda \in F_p$ plays no role in saving β, σ also PB-contains $F \setminus \{\lambda\}$. If it is because of λ, then σ contains a subpermutation $\gamma \sim \lambda$.

Since $\kappa \leqslant \lambda$ this means σ has a subpermutation $\gamma' \sim \kappa$ and by hypothesis no $\alpha \in F_b$ can save σ since $\kappa \not\sim$ removebar(α) for any $\alpha \in F_b$. Thus σ PB-contains F because of κ, so σ PB-contains $F \setminus \{\lambda\}$, and $\mathrm{Av}_B(F \setminus \{\kappa\}) \subseteq \mathrm{Av}_B(F)$.

Now suppose σ PB-contains $F \setminus \{\lambda\}$ because of β. Since $\lambda \in F_p$ plays no role in saving σ, σ PB-contains F also because of β. Thus $\mathrm{Av}_B(F) \subseteq \mathrm{Av}_B(F \setminus \{\lambda\})$. □

Here are some examples which demonstrate the lemmas.

Example 3.6.1. Let $F = \{5\bar{1}\bar{2}43, 4\bar{1}32\}$. We claim that the first pattern is redundant. Suppose you want to decide whether σ PB-contains F because of $\beta = 5\bar{1}\bar{2}43$, then first find a subpermutation $\gamma \sim 543 \sim 321$, then check whether this is part of a longer subpermutation $\delta \sim 4132$ or $\delta \sim 51243$. But if it is part of 4132 then it is automatically also part of 51243, so the decision is already made and there is no need to check for 51243 as well. This example generalises to Lemma 3.6.1.

The next example shows the situation in Lemma 3.6.3.

Example 3.6.2. Let $F = \{54321, 4321, 321, 3\bar{4}21\}$. We claim the pattern 54321 is redundant. Suppose you want to decide whether σ PB-contains F because of $\beta = 54321$. If so σ contains the subpermutation $\gamma = 4321$ as well, and since there is only one $\alpha \in F_b$, with removebar$(\alpha) = 321$, and neither 54321 nor 4321 is order-isomorphic to 321, so the second condition of the Definition does not apply. Thus 4321 suffices to deal with it.

Note that for $F' = \{4321, 321, 3\bar{4}21\}$, Lemma 3.6.3 cannot be applied to remove 4321, and one can show that $F' \not\sim_B \{321, 3\bar{4}21\}$. However, Lemma 3.6.2 does apply, and shows $F' \sim_B \{4321, 3\bar{4}21\}$.

The next examples show situations where none of Lemmas 3.6.1–3.6.3 apply.

Example 3.6.3. Let $F = \{5\bar{1}\bar{2}463, 4\bar{1}32\}, G = \{4\bar{1}32\}$ and consider the permutation $\sigma = 41352$. We claim that σ PB-contains F but PB-avoids G.

To see the first claim, there exists $\beta = 5\bar{1}\bar{2}463 \in F$ and $\gamma = 3241$ a subpermutation of σ so that $\gamma \sim \text{removebar}(5\bar{1}\bar{2}463)$, and for each $\alpha \in F_b$:

– $\alpha = 5\bar{1}\bar{2}463$ satisfies $\gamma \sim \text{removebar}(5\bar{1}\bar{2}463)$ but is not a subpermutation of any $\delta \sim 512463$ of σ, so this α does not save it.

– $\alpha = 4\bar{1}32$ does not satisfy $\gamma \sim \text{removebar}(4\bar{1}32) = 432$ so this α does not save it either.

To see the second claim, the only possible β is $4\bar{1}32$, and the only subpermutation of σ that is $\sim \text{removebar}(\beta)$ is $\gamma = 432$, but γ is a subpermutation of $\delta = 4132$ of σ and $\delta \sim \text{unbar}(\beta)$. So by Definition 2 σ PB-avoids $\{4\bar{1}32\}$.

Example 3.6.4. $F = \{3\bar{1}24, 312\}$, $G = \{213, 312\}$ have same PB-avoidance sets, and again none of the three lemmas applies. We claim that both F and G are "minimal" in that there is no set $H \subseteq \mathscr{B}$ with $|H| = 1$ and $\text{Av}_B(H) = \text{Av}_B(F)$.

3.6.1 Further lemma

We have one further way to remove redundant B-sequences which is slightly more involved, and will be useful below.

Lemma 3.6.4. *Let $F \subseteq \mathscr{B}$ with $\kappa, \lambda \in F_b$,* $\text{removebar}(\kappa) \leqslant \text{removebar}(\lambda)$ *and* $\text{unbar}(\kappa) \sim \text{unbar}(\lambda)$. *If*

– *$\alpha \in F$ with* $\text{removebar}(\alpha) = \text{removebar}(\kappa)$ *implies $\alpha = \kappa$, and*

– *$\alpha \in F$ with* $\text{removebar}(\alpha) = \text{removebar}(\lambda)$ *implies $\alpha = \lambda$,*

then $F \sim_B F \setminus \{\lambda\}$.

Proof. If σ PB-contains F because of λ and γ, then by hypothesis there exists $\gamma' <_{\text{subperm}} \gamma$ with $\gamma' \sim \text{removebar}(\kappa)$. If there exists $\alpha \in F_b$ and δ with $\gamma' <_{\text{subperm}} \delta <_{\text{subperm}} \sigma$ and $\text{removebar}(\alpha) \sim \delta$ then by hypothesis the only possible α is κ, so

σ contains $\delta \sim \text{unbar}(\kappa) \sim \text{unbar}(\lambda)$ which is a contradiction that σ PB-contains F because of λ, so σ PB-contains F because of κ, hence σ PB-contains $F \setminus \{\lambda\}$, so $\text{Av}_B(F \setminus \{\lambda\}) \subseteq \text{Av}_B(F)$.

Conversely if σ PB-contains $F \setminus \{\lambda\}$, but PB-avoids F, then it must be that some $\gamma <_{\text{subperm}} \sigma$ is saved by λ. But this cannot be since there is no $\alpha \in (F \setminus \{\lambda\})_b$ with $\text{removebar}(\alpha) \sim \text{removebar}(\lambda)$. $\square$

Example 3.6.5. Let

$$F = \{613\bar{5}2\bar{7}\bar{8}4, 61352\bar{7}\bar{8}4, 613\bar{5}27\bar{8}4, 613\bar{5}2\bar{7}84, 613527\bar{8}4, 61352\bar{7}84\}.$$

The lemma shows $F \sim_B \{613\bar{5}2\bar{7}\bar{8}4\}$.

We know there are further lemmas that can be proved, with more complicated hypotheses, which can remove more redundant elements of a set F. The next example shows a situation where a lemma is needed.

Example 3.6.6. $F = \{4123, 4231, 43251, 4\bar{1}352\}$. We claim that the pattern 43251 is redundant. Suppose σ PB-contains F. Either this is because of some $\beta \in F \setminus \{43251\}$, else it is only because of $\kappa = 43251$. So, each of the subpermutation $\gamma \sim 4351 \sim 4251 \sim 3251 \sim \text{removebar}(4\bar{1}352) <_{\text{subperm}} \delta$ must be saved by $4\bar{1}352$ with $\gamma <_{\text{subperm}} \tau \sim \text{unbar}(4\bar{1}352)$. Thus, σ will contains subpermutation $4a3b251$ such that $4a351 \sim 4a251 \sim 3b251 \sim \text{unbar}(4\bar{1}352)$. So, $4a3b251$ is either order isomorphic to **61**5**2**47**3** or **6251**473. Both of these subpermutations contains removebar(4123) and removebar(4231) respectively. So, σ PB-contains F because of removebar(4123) or removebar(4231) contradicts that it is because of κ only. Thus, σ PB-contains F and also PB-contains $F \setminus \{43251\}$

However, we were unable to generalise this in time for the thesis deadline.

3.7 BLOCKS

Here we follow Pudwell and Smith [45]. Let σ be a permutation. Call a factor $B_i = a_{i,1}a_{i,2}\dots a_{i,n_i}$ of σ a *block* if $n_i > 0$ and $a_{i,j} > a_{i,j+1}$ for all $1 \leqslant j < n_i$. (Recall that factor means the entries are contiguous in σ.) A *(maximal) block decomposition* of σ is an expression of the form $\sigma = B_1B_2B_3\dots B_m$ where each B_i is a block and for any two adjacent blocks $B_i = a_{i,1}a_{i,2}\dots a_{i,n_i}$ and $B_{i+1} = a_{i+1,1}a_{i+1,2}\dots a_{i+1,n_{i+1}}$ we have $a_{i,n_i} < a_{i+1,1}$. For example $\sigma = 87634521$ has block decomposition $B_1 = 8763, B_2 = 4, B_3 = 521$. For convenience we indicate the block decomposition of σ by inserting $|$ symbols to separate blocks, so for our example we write $8763 \mid 4 \mid 521$.

If $B_i = a_{i,1}a_{i,2}\dots a_{i,n_i}$ is a block, let $\widetilde{B_i} = a_{i,n_i}\dots a_{i,2}a_{i,1}$. We have the following.

Lemma 3.7.1 ([45])**.** *If σ has block decomposition $B_1B_2B_3\dots B_m$ then*

$$p_1(\sigma) = \widetilde{B_1}\widetilde{B_2}\widetilde{B_3}\dots\widetilde{B_m}.$$

For example $\sigma = 987354621 = 9873 \mid 54 \mid 621$ so $p_1(\sigma) = 3789\,45\,126$. Note that $p_1(\sigma)$ essentially reverses blocks of $B_1B_2B_3\dots B_m$. The largest token in a block will move to the right across all the smaller elements in the block while the smallest element in a block will move to the left across the larger elements in the block. Thus, the elements in a block are sorted after 1 pass.

Lemma 3.7.2. *Let $\gamma = B_1B_2B_3\dots B_m$ be a factor of σ, $\alpha \in B_1$, $\beta \in B_m$ such that $\alpha > \beta$. After k-passes through a deterministic pop stack, γ is not sorted if $m > 4k$.*

Proof. Due to the deterministic sorting process, after 1 pass, the size of each block is at most 3 (having 3 elements) [18]. Note that if γ is sorted after $k-1$ passes, then the state where α and β in a block such that $\dots|\dots\alpha\dots\beta\dots|\dots$ must have happened so that after k passes, α will be on the right of β such as $\gamma = \dots\beta\dots\alpha\dots$. We will prove that this state never happen at the end of $k-1$ passes, thus γ is not sorted after k

passes.

Let assume each block in γ including B_1 and B_m has at most 1 element. Note that this assumption is the worse case scenario for the initial state of γ before 1 pass because after 1 pass, the number of blocks in $p_1(\gamma)$ will remain the same. So, there are total $4k$ elements between α and β. After 1 pass, there will still be $4k$ elements between α and β because initially there is no other element in B_1 and B_m for α to move to right and β to move to the left. In the best case, assume starting from $p_1(\gamma)$, each subsequence pass will move α to the right across two tokens and move β to the left across two tokens. Thus, the the number of tokens between α and β will be reduced by 4 tokens in each pass. Thus, after $(k-1)$ passes which means after $(k-2)$ passes from $p_1(\gamma)$, the number of tokens between α and β will be more than or equal to $4k - 4(k-2)$. So, it is obvious that the state $\ldots|\ldots\alpha\ldots\beta\ldots|\ldots$ has not yet happened and γ cannot be sorted at the end of k passes. In other cases, the number of tokens that move passed α or β might be less than 2 tokens as the size of blocks is at most 3. Therefore, at the end of $(k-1)$ passes, number of tokens separating α or β will still be more than 8. Thus, γ is still not able to be sorted at the end of k passes. □

3.8 GENERAL CHARACTERISATION OF k-PASS POP-STACK SORTABLE PERMUTATIONS

Theorem 3.8.1. *Let $k \in \mathbb{N}_+$. There exists a finite set $F_k \subseteq \mathscr{B}$ such that the set of all k-pass pop-stack sortable permutations is equal to $Av_B(F_k)$. Moreover, the set F_k can be algorithmically constructed.*

Proof. Let G_k denote the set of all k-pass pop-stack sortable permutations. We proceed by induction, with the base case $k = 1$ established by Avis and Newborn [7] (specifically, $F_1 = \{231, 312\}$). Assume F_{k-1} has been constructed, is finite, and $G_{k-1} = \mathrm{Av}_B(F_{k-1})$.

Let $r_{\max} = \max\{|\text{removebar}(\beta)| \mid \beta \in F_{k-1}\}$ and $C = (3+8k)3r_{\max}$. Then define

$$\Omega_k = \{\tau \in \mathscr{B}_p \mid |\tau| \leqslant 3r_{\max}, \tau \notin G_k\} \cup$$
$$\{\tau \in \mathscr{B}_b \mid |\tau| \leqslant C, |\text{removebar}(\tau)| \leqslant 3r_{\max}, \text{removebar}(\tau) \notin G_k, \text{unbar}(\tau) \in G_k\}.$$

Claim 1: Ω_k is finite: we have

$$\sum_{i=1}^{C} i!2^i$$

B-sequences ($i!$ permutations and 2^i ways to assign bars) to consider to add to Ω_k. This count includes those in $\mathscr{B}_p$ since $3r_{\max} \leqslant C$.

Claim 2: Ω_k is algorithmically constructible: since we only have finitely many τ of length at most C, for each τ we can check $\tau \notin G_k, \text{removebar}(\tau) \notin G_k$ and $\text{unbar}(\tau) \in G_k$ in linear time by passing them according to the deterministic procedure.

Claim 3: $\sigma \notin G_k$ if and only if σ PB-contains Ω_k.

Proof of Claim 3. Recall that $\sigma \notin G_k$ if and only if $p_1(\sigma) \notin G_{k-1}$ if and only if $p_1(\sigma)$ PB-contains F_{k-1}.

To prove the forward direction, suppose $p_1(\sigma)$ PB-contains F_{k-1}, and further assume this is because of some $\zeta <_{\text{subperm}} p_1(\sigma)$ and $\beta \in F_{k-1}$ with $\text{removebar}(\beta) \sim \zeta$, and there is no $\alpha \in (F_{k-1})_b$ and $\delta <_{\text{subperm}} \sigma$ with $\zeta <_{\text{subperm}} \delta$, $\zeta \sim \text{removebar}(\alpha)$ and $\delta \sim \text{unbar}(\alpha)$. Note, there may be many choices of β and ζ to take, but fix one choice.

1. Mark tokens corresponding to ζ in $p_1(\sigma)$ **bold**. Let $\zeta' <_{\text{subperm}} \sigma$ be such that after one pass, the tokens belonging to ζ' are the bold tokens corresponding to $\zeta <_{\text{subperm}} p_1(\sigma)$. Mark the ζ' tokens bold as well. Note that $|\zeta'| \leqslant r_{\max}$.

   For example, if $\sigma = 987354621$ then $p_1(\sigma) = 378945126$ which PB-contains $F_1 = \{231, 312\}$ because of (for instance) $\beta = 312$ and the subpermutation $\zeta = 946$ of

$p_1(\sigma)$. We write $p_1(\sigma)$ as $378\mathbf{94}512\mathbf{6}$, and thus σ as $\mathbf{9}87354\mathbf{6}21$.

2. Next, write σ in block decomposition $\sigma = B_1B_2B_3\dots B_m$. Say that B_i is **bold** if it contains at least one bold entry (from ζ'). We wish to delete non-bold entries of σ but we do not want to *merge* bold blocks, so we apply the following subroutine.

   ```
   – set $\kappa = \sigma$
   – while $a_{i,j}$ is a non-bold letter,
           if removing $a_{i,j}$ from $\kappa$ does not cause two or more bold blocks to merge,
           delete $a_{i,j}$ from $\kappa$.
   ```

   We claim that at the end of this process $|\kappa| \leqslant 3|\zeta'| \leqslant 3r_{\max}$. Let $a_{i,j} \in B_i$ with $1 < i < m$ be a non-bold token in κ. If at most one of B_{i-1}, B_i, B_{i+1} is bold, then removing $a_{i,j}$ cannot merge bold blocks. Else assume at least two of B_{i-1}, B_i, B_{i+1} are bold. If $a_{i,j}$ is not the first or last entry in B_i, it can be deleted without merging blocks. This leaves at most two unbold entries in each block. For B_1 (resp. B_m) we can delete all except the last (resp. first) entry without merging bold blocks. This leaves at most two unbold entries in each block. Then in the worst case each block contains just one bold entry, with an unbold entry on either side. For example, if we get to $\kappa = \cdots \mid 12, \mathbf{10}, 8 \mid 9\mathbf{7}5 \mid 6\mathbf{4}2 \mid \mathbf{3}1$ then we cannot delete $8, 9, 5, 6, 2, 3$ without merging blocks.

3. After this, we obtain a permutation $\kappa <_{\text{subperm}} \sigma$ such that the bold letters $\zeta' <_{\text{subperm}} \kappa$ and $|\kappa| \leqslant 3r_{\max}$.

We now claim that $p_1(\kappa)$ PB-contains F_{k-1} because of the same ζ and β as σ. Since bold blocks are preserved in κ, we know that $p_1(\kappa)$ will also contain ζ. Now suppose there is some $\alpha \in (F_{k-1})_b$ and $\delta <_{\text{subperm}} p_1(\kappa)\,(<_{\text{subperm}} p_1(\sigma))$ with $\zeta' <_{\text{subperm}} \delta$, $\text{removebar}(\alpha) \sim \zeta$ and $\delta \sim \text{unbar}(\alpha)$. This means that the same α saves σ, which contradictions our original assumption. Thus $p_1(\kappa)$ PB-contains F_{k-1} which implies $p_1(\kappa) \notin G_{k-1}$ which implies $\kappa \notin G_k$.

Thus since $|\kappa| \leqslant 3r_{\max}$ and $\kappa \notin G_k$, we have $\kappa \in (\Omega_k)_p$ by definition. To finish this direction, we will show that κ is not saved by any $\tau \in (\Omega_k)_b$.

Suppose (for contradiction) that $\kappa <_{\text{subperm}} \sigma$ is saved by some $\tau \in (\Omega_k)_b$. Thus we have removebar$(\tau) \sim \kappa$, and some subpermutation $\delta <_{\text{subperm}} \sigma$ with $\kappa <_{\text{subperm}} \delta$ and $\delta \sim$ unbar$(\tau) \in G_k$. This means $p_1(\delta)$ PB-avoids F_{k-1}.

Now $p_1(\delta)$ will contain ζ since blocks containing $\zeta' <_{\text{subperm}} \kappa <_{\text{subperm}} \delta$ will not merge after one pass. Since $p_1(\delta)$ PB-avoids F_{k-1} and contains ζ, there must be some $\alpha \in (F_{k-1})_b$ which saves $\zeta <_{\text{subperm}} p_1(\delta)$. This means there is some $\delta' <_{\text{subperm}} p_1(\delta)$ with $\zeta <_{\text{subperm}} \delta'$, $\zeta \sim$ removebar(α) and $\delta' \sim$ unbar(α).

We claim α saves $\zeta <_{\text{subperm}} p_1(\sigma)$, since there exists $\delta' <_{\text{subperm}} p_1(\delta) <_{\text{subperm}} p_1(\sigma)$ with $\zeta \sim$ removebar(α) and $\delta' \sim$ unbar(α). This contradicts that $p_1(\sigma)$ PB-contains F_{k-1} because of ζ and β. Thus we have shown $\sigma \notin G_k$ implies σ PB-contains Ω_k.

Now for the converse direction, suppose that σ PB-contains Ω_k, and so we can assume that this is because of $\gamma <_{\text{subperm}} \sigma$ and $\tau \in (\Omega_k)_p$ with $\gamma \sim \tau$ (which is not saved by any $\alpha \in (\Omega_k)_b$), and so by definition $\gamma \notin G_k$ so $p_1(\gamma)$ PB-contains F_{k-1}.

Assume (for contradiction) that $\sigma \in G_k$. We will show that this implies we can construct some $\kappa <_{\text{subperm}} \sigma$ such that $\kappa \in G_k$, $\gamma <_{\text{subperm}} \kappa$ and $|\kappa| \leqslant C = (3+8k)3r_{\max}$. If so, then we can construct $\alpha \in (\Omega_k)_b$ with unbar$(\alpha) = \kappa$ and removebar$(\alpha) = \gamma$, which means α saves $\gamma <_{\text{subperm}} \sigma$, and this gives a contradiction.

Here is how we construct κ. In σ, mark the tokens corresponding to γ **bold**.

Call a block of σ bold if it contains at least one bold token, and otherwise a block is called non-bold. Starting with $\kappa = \sigma$, we delete non-bold tokens using the following procedure, which is more careful than the similar subroutine used in the proof of the forward direction above. The goal is to delete non-bold tokens to obtain a permutation κ with subpermutation γ such that for *every* block B in κ there is a block B' in σ so that the tokens in B are tokens in B'. That is, we do not allow *any* blocks to merge, only to be deleted entirely.

```
– set $\kappa = \sigma$
– while $a_{i,j} \in B_i$ is a non-bold letter,
    - if removing $a_{i,j}$ from $\kappa$ does not cause two or more blocks *of any kind* (bold
      or non-bold) in $\kappa$ to merge, delete $a_{i,j}$ from $\kappa$,
    - if $B_i$ is non-bold and removing the entire block $B_i$ at once does not cause
      any of the remaining blocks to merge, then delete $B_i$.
```

We claim that at the end of this process each block contains at most two non-bold entries, which will be the first and last entries of the block. However, since we have not deleted non-bold blocks if their removal would cause other blocks to merge, we could have arbitrarily long factors of non-bold blocks, as in Figure 3.2.

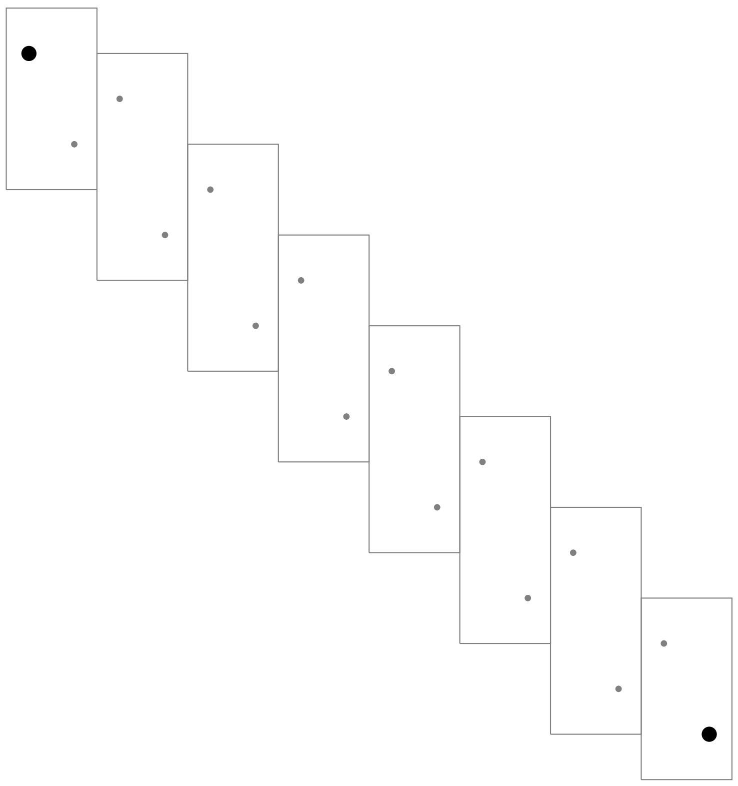


Figure 3.2: Worst case situation: $\kappa = \mathbf{16}, 14, 15, 12, 13, 10, 11, 89674523\mathbf{1}$

However, we claim that this is the only situation that can occur with arbitrarily long blocks of size 2, and in this case if the factor is longer than $4k$ blocks by Lemma 3.7.2, the factor cannot be sorted by k passes and so neither can σ, contradiction. Thus we have $|\kappa|$ is at most $3|\gamma|$ (the bold blocks with a non-bold entry first and last) plus $8k|\gamma|$ ($4k$ factors each containing 2 tokens, as in Figure 3.2, in the worst case occurring between every pair of bold tokens from γ). Thus

$$|k| \leqslant (3+8k)|\gamma| \leqslant (3+8k)3r_{\max} = C$$

If κ cannot be sorted, then $p_1(\kappa) \notin G_{k-1}$ because of some subpermutation $\tau \sim$ removebar(β) with $\beta \in F_{k-1}$, but since no block has merged in obtaining $p_1(\kappa)$, $p_1(\sigma)$ also contains τ which cannot be saved since blocks containing the tokens forming τ are fixed. Thus $p_1(\sigma) \notin G_{k-1}$, so $\sigma \notin G_k$, contradiction. □

Claim 3 implies that we could take $F_k = \Omega_k$ and the theorem is done. However, we can first apply Lemmas 3.6.1–3.6.4 to Ω_k to obtain a smaller set with the same PB-avoidance set, so we will call this F_k. As remarked in the previous section, the result of appying the lemmas is not guaranteed to give a set that is minimal or unique. Note that each of these lemmas needs to check a finite set so each is algorithmic. □

3.9 CONCLUSION

We have shown in Theorem 3.8.1 that there are only finitely forbidden patterns characterising the permutations sortable by k-pass pop-stack. Before the proof was constructed, a computer program was written based on some unproved assumptions to find the forbidden patterns for permutations sortable by 3-pass pop-stack. By using the computer program, we were able to find the same 8 forbidden patterns for 2-pass pop-stack as in [45], as well as obtaining a list of 49 patterns for 3-pass pop-stack. The list of 49 patterns found is shown in Figure 3.3.

23451,	54312,	54231,	54132,	54123,	34512,	34521
35412,	43251,	43512,	43521,	45123,	45132,	45213
45231,	45321,	51234,	53421,	53412,	53241,	51432
52413,	513462,	$52\bar{6}413$,	$51\bar{6}423$,	$513\bar{6}24$,	$5124\bar{6}\bar{7}3$,	$453\bar{1}62$
$4\bar{6}3\bar{1}572$,	$36\bar{1}452$,	$35\bar{1}462$,	$523\bar{6}14$,	$4\bar{7}3\bar{1}562$,	$5\bar{1}\bar{2}4673$,	$5\bar{1}\bar{2}6473$
$463\bar{1}52$,	$5\bar{1}\bar{2}7463$,	$512\bar{7}4\bar{6}3$,	$5\bar{1}4\bar{2}673$,	$5142\bar{6}\bar{7}3$,	$5\bar{2}4\bar{1}673$,	$5241\bar{6}\bar{7}3$
$613\bar{7}5\bar{2}4$,	$623\bar{7}5\bar{1}4$,	$5\bar{1}\bar{2}7\bar{4}683$,	$5\bar{1}\bar{2}8\bar{4}673$,	$613\bar{5}2\bar{7}\bar{8}4$,	$623\bar{5}1\bar{7}\bar{8}4$,	$512\bar{6}4\bar{7}3$

Figure 3.3: List of 49 patterns in Ω_3 found by computer search

Based on these patterns, and the various anomalies contained in them such as that shown in Example 3.3.3, we formulated our new definitions of B-sequence and PB-contains, and then proved that the set of forbidden patterns for permutations sortable by k-pass pop-stack is finitely based. The proof is in the form of constructive algorithm and it can be implemented as a computer program to find the Ω_k which contains the forbidden patterns. However, it is not feasible to generate the complete Ω_k list because of the high upper bound. Even though the complete Ω_k list could be found with the aid of a supercomputer, the list still contains a lot of redundant patterns. Lemmas 3.6.1, 3.6.2, 3.6.3 and 3.6.4 can only remove some of the redundant patterns. For future research, one can find more Lemmas to remove all redundant patterns.

Rather than using the correct bound to generate Ω_3, we implemented a computer program based on the proof but we use a lower value for the bound to generate the Ω_3 list. We found that the list contains all the 49 patterns found above. So, we believe that the smallest set of forbidden patterns that characterise the permutations sortable by 3-pass pop-stack could be exactly the list in Figure 3.3.

CHAPTER 4
FUTURE RESEARCH

4.1 STACKS IN SERIES

Let α, β and γ be the operations of passing an entry from input to the output through the stacks in $S(t,\infty), t \in \mathbb{N}$ as shown in Figure 4.1, where α is the operation of moving an entry from input into stack R, β is the operation of moving an entry from stack R into stack L, and the final output operation γ moves an entry from stack L to output.

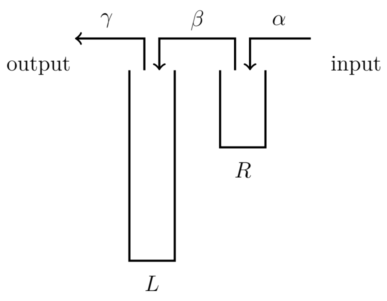


Figure 4.1: A stack R of depth t and an infinite stack L in series

Every entry i_j from a permutation $i_1 ... i_n \in S(t,\infty)$ with $j = 1, ..., n$ is moved exactly once by each of the three operations, as it moves from input to output. Therefore, all sortable permutations of length n in $S(t,\infty)$ can be encoded as words of length tn. We call these words *sorting codewords* and denote the set of all possible sorting codewords for $S(t,\infty)$ as $\mathbb{W}_{t,\infty}$.

Sorting codewords can potentially be used to describe a pattern avoidance set as a formal language class which in turn may provide a description for the generation function that enumerates the permutations in the avoidance set. For instance, any pattern avoidance set that can be described as a regular language has a rational generating function [47] while a pattern avoidance set that can be described by an unambiguous context-free language has an algebraic generating function [14]. Atkinson, Livesey and

Tulley proved the generating function for the set of permutations generated by passing an ordered sequence through a finite token-passing network is rational by showing the set is in bijection with a regular language [4]. Elder, Lee and Rechnitzer proved that the generating function of $S(2, \infty)$ is algebraic by proving the pattern avoidance set of $S(2, \infty)$ is in bijection with an unambiguous context-free language [25] of words in $\mathbb{W}_{2,\infty}$.

We attempted to find the generating function for $S(3, \infty)$ by following a similar strategy. Due to the nondeterministic nature of sorting with stacks, there can be more than one sorting codeword for each sortable permutation. Thus, we need to find a subset of $\mathbb{W}_{3,\infty}$ that is in bijection with $S(3, \infty)$. To do so, we tried selecting the codewords that output tokens as soon as possible, that is, has more γ letters closer to the front. We denote such a codeword as *greedy codeword.* For instance, if $l_1 = \alpha\beta\gamma\alpha\alpha\beta\gamma\beta\gamma$ and l_2=$\alpha\beta\alpha\gamma\alpha\beta\gamma\beta\gamma$ are the only codewords that sort the permutation 132, so we choose l_1 as the greedy codeword for permutation 132. Note that l_1 can be obtained by l_2 by replacing the factor $\alpha\gamma$ with $\gamma\alpha$, which has no effect on permutation being sorted.

So, we find some additional rules that can help us to remove all redundant codewords that sort the same permutation and from the list of rules obtained, we choose the one that can produce a greedy codeword. For instance, clearly that we can either replace $\gamma\alpha$ with $\alpha\gamma$ or $\alpha\gamma$ with $\gamma\alpha$ but we choose the latter case because it obviously can produce a greedy codeword. However, so far we were not able to find sufficient rules to remove all redundant codewords for each sortable permutations of length greater than 12 even though we already found hundreds of such rules. While doing so, we observed the following.

Example 4.1.1. The codeword

$$\alpha\beta\alpha\alpha(\alpha\beta)^i\alpha\beta\gamma(\beta\alpha)^{i+1}\alpha\beta\gamma(\gamma)^{2i+1}\gamma\beta\beta\gamma\gamma; \quad i > 1$$

sorts the permutation $P_i =$

$$(4{+}2i, 5{+}2i)(3{+}i)(4{+}(2i{-}1), \ldots, 4{+}(i{+}1), 4{+}i)1(2{+}i, \ldots, 2{+}2, 2{+}1)(6{+}2i)2 \in S(3, \infty).$$

For instance, P_1=67451382 can be sorted by following the sorting codeword $\alpha\beta\alpha\alpha(\alpha\beta)\alpha\beta\gamma(\beta\alpha)^2\alpha\beta\gamma(\gamma)^3\gamma\beta\beta\gamma\gamma$. Such words cannot be part of a context-free language by the pumping lemma for context-free languages. Therefore we suspect there is no subset of sorting codewords in bijection with $S(3, \infty)$ that is (unambiguous) context-free.

Finding generating functions for $S(k, \infty)$ would be interesting in its own right, but also be a step towards the open problem of enumeration for two infinite stacks in series.

4.2 k-PASS POP-STACKS

Recall that the algorithm in Section 3.8 will produce a large set of Ω_k which contains a lot of redundant patterns with respect to PB-containment. Applying the Lemmas 3.6.1−4 can only remove some of the redundant patterns. For future research, it is possible that more Lemmas can be constructed to remove all the redundant patterns, but it will take a lot of effort because a pattern in F is redundant when **it is not only contains another patterns in F but also saved by some patterns in F**. So, the approach to construct and prove further lemmas is to check every possible cases for each pattern in F whether it contains some patterns or saved by some patterns. As a warm up exercise, one can try to construct a lemma for the Example 3.6.6.

Another interesting question is to **answer the characterisation problem of nondeterministic k-pass pop-stack**. The sorting procedure in Chapter 3 is deterministic because a pop operation is performed when the next input element is bigger than the top element in the pop-stack. Meanwhile, a nondeterministic procedure has no such restriction and allow a pop operation to happen at anytime as long as there are some elements in the pop-stack. Due to the nondeterministic procedure, there is no bar pattern in the expected basis. So, there is a possibility that an infinite antichain might exist for k passes. To prove this, the similar technique from Chapter 2 might be used.

Our notion of PB-avoidance also opens up some interesting possibilities. Recall that by Kaiser-Klazar [33, Thm. 3.4] and Marcus-Tardos [40] the function counting the number of permutations of length n in any $\mathrm{Av}(F)$ for F non-empty is either polynomial or exponential. It is conceivable some sets F could have PB-avoidance set with growth function strictly between polynomial and exponential, or strictly between exponential and factorial. Example 3.3.2 shows a non-trivial PB-avoidance set with factorial growth.

Generating functions for PB-avoidance sets might also exhibit interesting behaviour. For the sets F_k in 3.8.1 we know by [18] the generating functions are rational for all k, but for general set F the set $\mathrm{Av}_B(F)$ could have interesting enumerations.

REFERENCES

[1] Michael Albert and Mireille Bousquet-Mélou. Permutations sortable by two stacks in parallel and quarter plane walks. *European J. Combin.*, 43:131–164, 2015.

[2] R. E. L. Aldred, M. D. Atkinson, and D. J. McCaughan. Avoiding consecutive patterns in permutations. *Adv. in Appl. Math.*, 45(3):449–461, 2010.

[3] José María Amigó, Sergi Elizalde, and Matthew B. Kennel. Forbidden patterns and shift systems. *J. Combin. Theory Ser. A*, 115(3):485–504, 2008.

[4] M. D. Atkinson, M. J. Livesey, and D. Tulley. Permutations generated by token passing in graphs. *Theoret. Comput. Sci.*, 178(1-2):103–118, 1997.

[5] M. D. Atkinson, M. M. Murphy, and N. Ruškuc. Sorting with two ordered stacks in series. *Theoret. Comput. Sci.*, 289(1):205–223, 2002.

[6] M. D. Atkinson and J.-R. Sack. Pop-stacks in parallel. *Inform. Process. Lett.*, 70(2):63–67, 1999.

[7] David Avis and Monroe Newborn. On pop-stacks in series. *Utilitas Math.*, 19:129–140, 1981.

[8] Eric Babson and Einar Steingrímsson. Generalized permutation patterns and a classification of the Mahonian statistics. *Sém. Lothar. Combin.*, 44:Art. B44b, 18, 2000.

[9] Jean-Luc Baril. Classical sequences revisited with permutations avoiding dotted pattern. *Electron. J. Combin.*, 18(1):Paper 178, 18, 2011.

[10] Andrew M. Baxter and Lara K. Pudwell. Enumeration schemes for vincular patterns. *Discrete Math.*, 312(10):1699–1712, 2012.

[11] Antonio Bernini, Luca Ferrari, and Renzo Pinzani. Enumeration of some classes of words avoiding two generalized patterns of length three. *J. Autom. Lang. Comb.*, 14(2):129–147, 2009.

[12] Miklós Bóna. Exact enumeration of 1342-avoiding permutations: a close link with

labeled trees and planar maps. *J. Combin. Theory Ser. A*, 80(2):257–272, 1997.

[13] Mireille Bousquet-Mélou and Steve Butler. Forest-like permutations. *Ann. Comb.*, 11(3-4):335–354, 2007.

[14] N. Chomsky and M. P. Schützenberger. The algebraic theory of context-free languages. In *Computer programming and formal systems*, pages 118–161. North-Holland, Amsterdam, 1963.

[15] Anders Claesson. Generalized pattern avoidance. *European J. Combin.*, 22(7):961–971, 2001.

[16] Anders Claesson and Toufik Mansour. Counting occurrences of a pattern of type $(1, 2)$ or $(2, 1)$ in permutations. *Adv. in Appl. Math.*, 29(2):293–310, 2002.

[17] Anders Claesson and Toufik Mansour. Enumerating permutations avoiding a pair of Babson-Steingrímsson patterns. *Ars Combin.*, 77:17–31, 2005.

[18] Anders Claesson and Bjarki Ágúst Guðmundsson. Enumerating permutations sortable by k passes through a pop-stack. *Sém. Lothar. Combin.*, 80B:Art. 43, 12, 2018.

[19] Colin Defant. Counting 3-Stack-Sortable Permutations. *arXiv e-prints*, arXiv:1903.09138, Mar 2019.

[20] Daniel Denton. Methods of computing deque sortable permutations given complete and incomplete information. *arXiv e-prints*, arXiv:1208.1532, Aug 2012.

[21] Phan Thuan Do, Dominique Rossin, and Thi Thu Huong Tran. Permutations weakly avoiding barred patterns and combinatorial bijections to generalized Dyck and Motzkin paths. *Discrete Math.*, 320:40–50, 2014.

[22] S. Dulucq and O. Guibert. Stack words, standard tableaux and Baxter permutations. In *Proceedings of the 6th Conference on Formal Power Series and Algebraic Combinatorics (New Brunswick, NJ, 1994)*, volume 157, pages 91–106, 1996.

[23] Murray Elder. Permutations generated by a stack of depth 2 and an infinite stack in series. *Electron. J. Combin.*, 13(1):Research Paper 68, 12, 2006.

[24] Murray Elder and Yoong Kuan Goh. Permutations sorted by a finite and an infinite stack in series. In *Language and automata theory and applications*, volume 10792 of *Lecture Notes in Comput. Sci.*, pages 220–231. Springer, Cham, 2018.

[25] Murray Elder, Geoffrey Lee, and Andrew Rechnitzer. Permutations generated by a depth 2 stack and an infinite stack in series are algebraic. *Electron. J. Combin.*, 22(2):Paper 2.16, 23, 2015.

[26] Sergi Elizalde. Asymptotic enumeration of permutations avoiding generalized patterns. *Adv. in Appl. Math.*, 36(2):138–155, 2006.

[27] Sergi Elizalde and Marc Noy. Consecutive patterns in permutations. volume 30, pages 110–125. 2003. Formal power series and algebraic combinatorics (Scottsdale, AZ, 2001).

[28] P. Erdős and G. Szekeres. A combinatorial problem in geometry. *Compositio Math.*, 2:463–470, 1935.

[29] S. Even and A. Itai. Queues, stacks, and graphs. In *Theory of machines and computations (Proc. Internat. Sympos., Technion, Haifa, 1971)*, pages 71–86, 1971.

[30] OEIS Foundation Inc. The on-line encyclopedia of integer sequences. `https://oeis.org/A000108`, 2020.

[31] OEIS Foundation Inc. The on-line encyclopedia of integer sequences. `https://oeis.org/A000110`, 2020.

[32] OEIS Foundation Inc. The on-line encyclopedia of integer sequences. `https://oeis.org/A006318`, 2020.

[33] Tomáš Kaiser and Martin Klazar. On growth rates of closed permutation classes. volume 9, Research paper 10, 20. 2002/03. Permutation patterns (Otago, 2003).

[34] Sergey Kitaev. Multi-avoidance of generalised patterns. *Discrete Math.*, 260(1-3):89–100, 2003.

[35] Sergey Kitaev. *Patterns in permutations and words*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, Heidelberg, 2011. With a foreword

by Jeffrey B. Remmel.

[36] Donald E. Knuth. *The art of computer programming. Vol. 1: Fundamental algorithms*. Second printing. Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont, 1969.

[37] Donald 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.

[38] Darla Kremer. Postscript: "Permutations with forbidden subsequences and a generalized Schröder number" [Discrete Math. **218** (2000), no. 1-3, 121–130; MR1754331 (2001a:05005)]. *Discrete Math.*, 270(1-3):333–334, 2003.

[39] Percy A. MacMahon. *Combinatory analysis. Vol. I, II (bound in one volume)*. Dover Phoenix Editions. Dover Publications, Inc., Mineola, NY, 2004. Reprint of ıt An introduction to combinatory analysis (1920) and ıt Combinatory analysis. Vol. I, II (1915, 1916).

[40] Adam Marcus and Gábor Tardos. Excluded permutation matrices and the Stanley-Wilf conjecture. *J. Combin. Theory Ser. A*, 107(1):153–160, 2004.

[41] Max Murphy. *Restricted permutations, antichains, atomic classes, stack sorting*. PhD thesis, University of St Andrew, 2002.

[42] Adeline Pierrot and Dominique Rossin. 2-stack sorting is polynomial. *Theory Comput. Syst.*, 60(3):552–579, 2017.

[43] Vaughan R. Pratt. Computing permutations with double-ended queues. Parallel stacks and parallel queues. In *Fifth Annual ACM Symposium on Theory of Computing (Austin, Tex., 1973)*, pages 268–277. 1973.

[44] Lara Pudwell. Enumeration schemes for permutations avoiding barred patterns. *Electron. J. Combin.*, 17(1):Research Paper 29, 27, 2010.

[45] Lara Pudwell and Rebecca Smith. Two-stack-sorting with pop stacks. *Australas. J. Combin.*, 74:179–195, 2019.

[46] Pierre Rosenstiehl and Robert E. Tarjan. Gauss codes, planar Hamiltonian graphs, and stack-sortable permutations. *J. Algorithms*, 5(3):375–390, 1984.

[47] M. P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4:245–270, 1961.

[48] Rebecca Smith. Two stacks in series: a decreasing stack followed by an increasing stack. *Ann. Comb.*, 18(2):359–363, 2014.

[49] Rebecca Smith and Vincent Vatter. The enumeration of permutations sortable by pop stacks in parallel. *Inform. Process. Lett.*, 109(12):626–629, 2009.

[50] Einar Steingrímsson. Generalized permutation patterns—a short survey. In *Permutation patterns*, volume 376 of *London Math. Soc. Lecture Note Ser.*, pages 137–152. Cambridge Univ. Press, Cambridge, 2010.

[51] Robert Tarjan. Sorting using networks of queues and stacks. *J. Assoc. Comput. Mach.*, 19:341–346, 1972.

[52] Bridget Eileen Tenner. Coincidental pattern avoidance. *J. Comb.*, 4(3):311–326, 2013.

[53] Henning Úlfarsson. A unification of permutation patterns related to Schubert varieties. *Pure Math. Appl. (PU.M.A.)*, 22(2):273–296, 2011.

[54] Henning Úlfarsson. Describing West-3-stack-sortable permutations with permutation patterns. *Sém. Lothar. Combin.*, 67:Art. B67d, 20, 2011/12.

[55] Julian West. Sorting twice through a stack. *Theoret. Comput. Sci.*, 117(1-2):303–313, 1993. Conference on Formal Power Series and Algebraic Combinatorics (Bordeaux, 1991).

[56] Julian West. Generating trees and the Catalan and Schröder numbers. *Discrete Math.*, 146(1-3):247–262, 1995.

[57] Alexander Woo and Alexander Yong. When is a Schubert variety Gorenstein? *Adv. Math.*, 207(1):205–220, 2006.

[58] Doron Zeilberger. A proof of Julian West's conjecture that the number of two-

stack-sortable permutations of length n is $2(3n)!/((n+1)!(2n+1)!)$. *Discrete Math.*, 102(1):85–93, 1992.