Freie Universität Berlin
Institut für Informatik

Masterarbeit

Complexity of Permutation Pattern Matching

Benjamin Aram Berendsohn

Betreuer: Prof. Dr. László Kozma
Zweitgutachter: Prof. Dr. Wolfgang Mulzer

29.08.2019

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, den

.....
Benjamin Aram Berendsohn

Abstract

Permutation pattern matching (PPM) is the problem of determining whether a permutation π is contained in another permutation τ, i.e. τ has a subsequence that is order-isomorphic to π. We study the complexity of PPM. In particular, we consider the variants of PPM where π itself does not contain a fixed permutation σ.

We first investigate the incidence graphs avoiding a fixed permutation σ, and show that the treewidth of these graphs can be almost linear for almost all σ. This indicates that the restricted variants may be as hard as unrestricted PPM. For the counting variant #PPM, we show that, indeed, almost all restricted variants require exponential time, unless the exponential time hypothesis fails. Finally, we consider two on-line versions of PPM where π is fixed, and give upper and lower bounds on their space requirements depending on π.

Contents

1 Introduction

A permutation τ *contains* a permutation π if τ has a subsequence that is order-isomorphic to π. For example, the length-5 permutation 12453 contains 132, as it has 253 as a subsequence. However, it does not contain (*avoids*) 321, as it does not have a descending subsequence of length 3. Permutation avoidance was first studied in 1968, when Knuth [21] observed that the permutations that can be sorted by a single stack are exactly those which avoid the permutation 231. While permutation avoidance continues to be used in the characterization of sorting in restricted models [28, 11], it has become an active research topic on its own. In particular, there are numerous papers on the subject of counting the length-n permutations avoiding a fixed permutation π (see, e.g., the survey of Vatter [30]). The proof of the *Stanley-Wilf-conjecture* by Marcus and Tardos in 2004 [26] may be considered the most prominent result in this area. The Stanley-Wilf-conjecture asserts that for each π there is a constant c such that the number of permutations of length n avoiding π is bounded by c^n.

In this thesis, we consider the *permutation pattern matching* (PPM) problem of deciding whether a permutation π is contained in a permutation τ. To our knowledge, this problem has first been studied by Bose et al. in 1993 [8], who showed that it is NP-complete if π and τ are arbitrary permutations and both are part of the input. They also gave a polynomial time algorithm if π is *separable*.

The general hardness result combined with the positive result for the restricted problem begs the question *which* restrictions can be made to make the problem polynomial-time solvable. A rather well-studied approach is to require π to avoid a fixed permutation σ. Jelínek and Kynčl [19] provided a complete characterization of these restrictions in terms of NP-completeness, and also proved that even requiring τ to avoid a fixed permutation σ does not help in almost all cases. Kozma [22] conjectured that all such restricted variants of PPM are solvable in subexponential time, and that pattern avoidance might be related to minor avoidance in the incidence graph. We will provide evidence against both conjectures.

Apart from the question of polynomial tractability, there has been some research to find upper bounds to the complexity of PPM of the form c^n for a constant c and of the form $n^{f(k)}$. The trivial $n^{k+o(k)}$ was first improved to $n^{2k/3+o(k)}$ by Albert, Aldred, Atkinson and Holton [2] and to $n^{0.47k+o(k)}$ by Ahal and Rabinovich [1]. Bruner and Lackner [10] improved the trivial $O(2^n)$ bound to $O(1.79^n)$. Recently, Berendsohn, Kozma and Marx [3] improved these upper bounds to $n^{k/4+o(k)}$ and $O(1.6181^n)$ and also gave a lower bound of $n^{\Omega(k/\log k)}$ for the counting variant of PPM (assuming that the *exponential time hypothesis* of Impagliazzo and Paturi [18] holds).

It should be noted that Guillemot and Marx [16] gave an FPT algorithm of PPM with parameter $|\pi|$ that has running time linear in n. For very small k, this beats the

algorithms mentioned above.

This thesis investigates the complexity of several variants of PPM. In chapter 2, we give formal definitions used in the rest of the thesis. Chapter 3 discusses the performance of *treewidth-based* algorithms on restricted variants of PPM. This is done by giving upper and lower bounds on the treewidth of the incidence graph of certain classes of permutations. Chapter 4 proves hardness of the counting variant of PPM even if π avoids a fixed permutation σ, for almost all σ. This extends the result in Berendsohn, Kozma and Marx cited above. Finally, in chapter 5 we consider the *space* requirement of on-line algorithms solving PPM with fixed π.

2 Preliminaries

Throughout the thesis, when referring to the image of some set S under some function f, we simply write $f(S)$. For $n \in \mathbb{N}_+$, we write $[n] := \{1, 2, \ldots, n\}$.

2.1 Permutations and point sets

An *n-permutation* is a bijective function $\pi\colon [n] \to [n]$. We also simply call π a *permutation* and write $|\pi| = n$. We can alternatively characterize π as the sequence $(\pi(1), \pi(2), \ldots, \pi(n))$. For brevity, we sometimes omit commas and parentheses when writing down π, e.g. 1243. The operation $\pi_1 \circ \pi_2$ always refers to the composition of the *functions* π_1, π_2, also called the *product* of the two permutations π_1 and π_2. For example, $1342 \circ 4312 = 2413$. To concatenate two sequences ω_1 and ω_2, we write $\omega_1\omega_2$. We call the permutation $\mathrm{Id}_n = (1, 2, \ldots, n)$ the *identity permutation*.

For an n-permutation π, we define the *point set* S_π of π as follows:

$$S_\pi = \{(i, \pi(i)) \mid i \in [n]\} \subseteq [n]^2.$$

Observe that S_π is in *general position* in the sense that there is no pair of distinct points in S_π that agree on one coordinate. For a clearer and more compact notation, we will refer to the two coordinates of a point $p \in \mathbb{N}^2$ as $p.x$ and $p.y$, i.e. $p = (p.x, p.y)$. We call the points $p, q \in S_\pi$ *horizontal neighbors* if $|p.x - q.x| = 1$ and call them *vertical neighbors* if $|p.y - q.y| = 1$.
Let $S \subseteq [n]^2$ be a point set in general position and let π be the unique permutation such that S_π and S are isomorphic (there is a bijection $\phi\colon S_\pi \to S$ such that for all $p, q \in S_\pi$ we have $p.x < q.x$ if and only if $\phi(p).x < \phi(q).x$, and the same when replacing x with y). Then we call π the *reduction* of S and write $\pi = \mathrm{red}(S)$. Two points in S are called *horizontal (vertical) S-neighbors* if the respective points in the point set of $\mathrm{red}(S)$ are horizontal (vertical) neighbors. If the set S is clear from context, we write just *neighbors*.

For a finite sequence of distinct integers ω, we similarly define $\mathrm{red}(\omega)$, the *reduction* of ω, to be the unique permutation that is order-isomorphic to ω.

2.2 Incidence graphs

The *incidence graph* of an n-permutation π is the graph $G_\pi = (V, E)$, where $V = [n]$ and $\{u, v\} \in E$ if and only if $|u - v| = 1$ or $|\pi(u) - \pi(v)| = 1$.

Observe that we can obtain a drawing of G_π by drawing the points set S_π in the plane, and adding two Hamiltonian paths, one connecting the points in horizontal order,

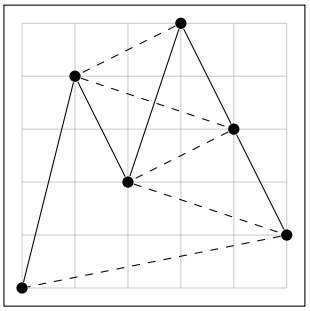

Figure 2.1: The point set of the permutation 153642 (with the point $(1, 1)$ in the bottom left corner) and its incidence graph. The horizontal path consists of the solid edges, the vertical graph of the dashed ones.

and one connecting the points in vertical order. The former *horizontal path* corresponds to the path $(1, 2, \ldots, n)$ in G_π. The latter *vertical path* corresponds to the path $(\pi^{-1}(1), \pi^{-1}(2), \ldots, \pi^{-1}(n))$ in G_π. We call the edges of the horizontal (vertical) path *horizontal* (*vertical*) edges. Observe that the edges correspond exactly to the pairs of horizontal respectively vertical neighbors in S. Figure 2.1 shows an example of a permutation point set and incidence graph. We further observe that permutation incidence graphs are exactly the graphs that arise from the union of two Hamiltonian paths.

For convenience, we extend the notion of incidence graphs to arbitrary sets S of points in general position by defining $G_S = (S, E)$, where $\{p, q\} \in E$ if and only if p and q are S-neighbors. Observe that G_S is isomorphic to $G_{\mathrm{red}(S)}$.

2.3 Operations on permutations

Let π be an n-permutation. We define the *reverse* π^{R}, the *complement* π^{C} and the *inverse* π^{-1} of π as follows. For $i \in [n]$, let

$$\begin{aligned}\pi^{\mathsf{R}}(i) &= \pi(n + 1 - i),\\ \pi^{\mathsf{C}}(i) &= n + 1 - \pi(i),\\ \pi^{-1}(i) &= j \text{ where } \pi(j) = i.\end{aligned}$$

It is easy to see that $\pi = (\pi^{\mathsf{R}})^{\mathsf{R}} = (\pi^{\mathsf{C}})^{\mathsf{C}} = (\pi^{-1})^{-1}$. We call the permutations that arise from a sequence of these operations *symmetries* of π.
The effects of taking reverse, complement and inverse on the point set S_π are intuitive: Taking reverse flips S_π on a vertical axis, taking complement flips S_π on a horizontal axis, and taking inverse flips S_π on a diagonal axis from the bottom left to the top right. Using these three operations, we can obtain all rotations of S_π and their orthogonal flips. Our observations in Section 2.2 imply that reversal, complement and inversion do not change the incidence graph (up to isomorphism).

Let π be an n-permutation and ρ be an m-permutation. Then the *direct sum* $\pi \oplus \rho$ and the *skew sum* $\pi \ominus \rho$ are the following $(m+n)$-permutations:

$$(\pi \oplus \rho)(i) = \begin{cases} \pi(i), & \text{if } i \leq n, \\ n + \rho(i-n), & \text{otherwise,} \end{cases}$$
$$(\pi \ominus \rho)(i) = \begin{cases} m + \pi(i), & \text{if } i \leq n, \\ \rho(i-n), & \text{otherwise.} \end{cases}$$

See figure 2.2 for examples on how the defined operations affect permutation point sets.

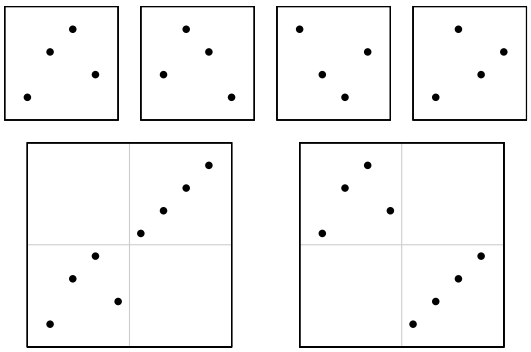

Figure 2.2: *(top row)* The point set of 1342, its reverse 2431, its complement 4213 and its inverse 1423. *(bottom row)* The direct sum $1342 \oplus 1234$ and the skew sum $1342 \ominus 1234$.

2.4 Subpermutations

A k-permutation π is a *subpermutation* of an n-permutation τ if there is an order-preserving index mapping ϕ from π to τ. More formally, π is a subpermutation of τ if there is a monotonically increasing injective function $\phi\colon [k] \to [n]$ such that for $i, j \in [k]$,

$$\pi(i) < \pi(j) \implies \tau(\phi(i)) < \tau(\phi(j)),$$

in other words, both ϕ and $\tau \circ \phi \circ (\pi^{-1})$ are strictly increasing. We also say τ *contains* π and write $\pi \preceq \tau$. If π is *not* a subpermutation of τ, we also say τ *avoids* π. We will implicitly use the following properties of $(\preceq)$ in the rest of the thesis.

Lemma 2.1. $(\preceq)$ *is a partial order and invariant with respect to reversal, complement, and inversion.*

Proof. Let π, ρ and τ be permutations and let $k = |\pi|$, $m = |\rho|$ and $n = |\tau|$. We have $\pi \preceq \pi$ via $\phi = \mathrm{Id}_n$, so $(\preceq)$ is reflexive. Let $\pi \preceq \tau$ via ϕ and $\tau \preceq \pi$. Then $k = n$, so ϕ is a strictly increasing function from $[n]$ to $[n]$, meaning $\phi = \mathrm{Id}_n$, implying $\pi = \tau$. Thus, $(\preceq)$

is antisymmetric. Finally, let $\pi \preceq \rho$ via ϕ and $\rho \preceq \tau$ via ψ. Then $\phi \circ \psi$ is monotonically increasing and

$$\pi(i) < \pi(j) \implies \rho(\psi(i)) < \rho(\psi(j)) \implies \tau(\phi(\psi(i))) < \tau(\phi(\psi(j)))$$

for $i, j \in [k]$. Thus, $\pi \preceq \tau$, implying that $(\preceq)$ is transitive. This concludes the proof that $(\preceq)$ is a partial order.

Now let $\pi \preceq \tau$ via ϕ. Then $\pi^{\mathsf{C}} \preceq \tau^{\mathsf{C}}$ via ϕ, moreover $\pi^{-1} \preceq \tau^{-1}$ via $\psi = \tau \circ \phi \circ (\pi^{-1})$ and $\pi^{\mathsf{R}} \preceq \tau^{\mathsf{R}}$ via ψ where $\psi(k-i) = n - \phi(i)$. □

We now define an analog of $(\preceq)$ for point sets. Let $P \subseteq [k]^2$ and $T \subseteq [n]^2$ be point sets in general position. We write $P \preceq T$ if there exists a mapping $\phi\colon P \to T$ that preserves relative position, i.e., for each $p, q \in P$,

$$p.x < q.x \implies \phi(p).x < \phi(q).x$$
$$p.y < q.y \implies \phi(p).y < \phi(q).y.$$

Alternatively, we can obtain a point set isomorphic to P by taking T and deleting some points if and only if $P \preceq T$.

Lemma 2.2 (Kozma [22, Lemma 5])**.** *Let π and τ be permutations. Then $\pi \preceq \tau$ if and only if $S_\pi \preceq S_\tau$.*

If $S_\pi \preceq T$ for some permutation π, we also write $\pi \preceq T$. Finally, if ω and μ are two finite sequences of distinct integers (each integer may occur in both ω and μ), we also define $\omega \preceq \mu$ as $\mathrm{red}(\omega) \preceq \mathrm{red}(\mu)$.

We remark that while the above shows a simple adaption of the notion or subpermutations to the realm of point sets, there is no intuitive way of doing the same for incidence graphs. Rather, when we delete an element from π to obtain a subpermutation, we have to delete the corresponding vertex v in G_π and then *add* an edge between the two horizontal neighbors of v and an edge between the two vertical neighbors of v. This operation is somewhat similar to contraction, but the distinction between horizontal and vertical edges seems to prevent using known results on contraction and minors.

2.5 Permutation classes

A *permutation class* is a set of permutations closed under taking subpermutations. For some set Π of permutations, define $\mathrm{Av}(\Pi)$ as the set of permutations that avoid all permutations from Π. Observe that for each permutation class $\mathcal{C}$, there is a (not necessarily finite) set Π such that $\mathcal{C} = \mathrm{Av}(\Pi)$. If Π is given explicitly, we will usually omit braces, and e.g. write $\mathrm{Av}(2413, 3142)$. If $\mathcal{C} = \mathrm{Av}(\pi)$ for a single permutation π, we call $\mathcal{C}$ a *principal permutation class*.

For $k \geq 2$, let the *k-increasing permutations* Inc_k (*k-decreasing permutations* Dec_k, *k-monotone permutations* Mon_k) be the class of permutations that can be partitioned into k increasing (decreasing, monotone) subsequences. It is easy to see that $\mathrm{Inc}_k = \mathrm{Av}(\mathrm{Id}_{k+1})$ and $\mathrm{Dec}_k = \mathrm{Av}(\mathrm{Id}_{k+1}^{\mathsf{R}})$.

We now state a version of Lemma 2.1 for permutation classes:

Corollary 2.3.

(i) $\mathrm{Av}(\sigma^{\mathsf{R}}) = \{\pi^{\mathsf{R}} \mid \pi \in \mathrm{Av}(\sigma)\}$

(ii) $\mathrm{Av}(\sigma^{\mathsf{C}}) = \{\pi^{\mathsf{C}} \mid \pi \in \mathrm{Av}(\sigma)\}$

(iii) $\mathrm{Av}(\sigma^{-1}) = \{\pi^{-1} \mid \pi \in \mathrm{Av}(\sigma)\}$

(iv) $\sigma_1 \preceq \sigma_2 \implies \mathrm{Av}(\sigma_1) \subseteq \mathrm{Av}(\sigma_2)$

Corollary 2.3 (iv) has an interesting consequence regarding the classes of k-increasing respectively k-decreasing permutations. Let π be a permutation of length at least k^2+1. Then, by the Erdős-Szekeres theorem [14] we know that π contains an increasing or decreasing subsequence of length k, thus $\mathrm{Id}_k \preceq \pi$ or $\mathrm{Id}_k^{\mathsf{R}} \preceq \pi$. Now corollary 2.3 (iv) implies the following:

Corollary 2.4. *Let σ be a permutation of length at least k^2+1. Then* $\mathrm{Av}(\mathrm{Inc}_k) \subseteq \mathrm{Av}(\sigma)$ *or* $\mathrm{Av}(\mathrm{Dec}_k) \subseteq \mathrm{Av}(\sigma)$.

We now define some operations on permutation classes. If $\mathcal{C}_1$ and $\mathcal{C}_2$ are permutation classes, we define $\mathcal{C}_1 \circ \mathcal{C}_2 = \{\pi_1 \circ \pi_2 \mid \pi_1 \in \mathcal{C}_1, \pi_2 \in \mathcal{C}_2, |\pi_1| = |\pi_2|\}$. Moreover, we define $(\mathcal{C}_1|\mathcal{C}_2)$ to be the set containing each permutation π for which there exist two sequences ω_1 and ω_2 such that $\pi = \omega_1\omega_2$ and $\mathrm{red}(\omega_i) \in \mathcal{C}_i$ for $i \in \{1, 2\}$. For a permutation class $\mathcal{C}$ define $\mathcal{C}^0 = \{\mathrm{Id}_n \mid n \in \mathbb{N}^+\}$ and $\mathcal{C}^{i+1} = \mathcal{C}^i \circ \mathcal{C}$. We remark that the results of all those operations are again permutation classes.

2.6 Computational problems

Finally, we define computational problems regarding subpatterns, which is the main focus of the thesis.

Given a k-permutation π and an n-permutation τ, the problem Permutation Pattern Matching (PPM for short) consists in determining whether $\pi \preceq \tau$. For a permutation class $\mathcal{C}$, the problem $\mathcal{C}$-Pattern PPM is PPM with the additional restriction that the input must satisfy $\pi \in \mathcal{C}$. We will further explore the counting variants #PPM and $\mathcal{C}$-Pattern #PPM where we ask for the *number of occurrences* of π in τ.

3 Treewidth of permutation classes

In this chapter, we investigate the performance of algorithms that use the *treewidth* of the incidence graph of the pattern π. The treewidth of a graph is intended as a measure of how "tree-like" G is. The main motivation to investigate the treewidth in the context of algorithms and complexity is that problems that are easy on trees are often also easy on graphs with low treewidth (see, e.g., Bodlaender [6]). It turns out that the same is true for permutation pattern matching. We do not define treewidth here, as we will not use the formal definition. Instead, we will use well-known results, for example relating the treewidth to minors. A weaker version of the results presented here was already published in Berendsohn et al. [3].

Ahal and Rabinovich [1] gave an algorithm for PPM (which can be seen to apply to #PPM as well) with running time $n^{2\mathrm{tw}(G_\pi)}$. This was recently improved to $n^{\mathrm{tw}(G_\pi)+1}$ in Berendsohn et al. [3]. Using a previously known result about 4-regular graphs and an additional technique due to Cygan et al. [13], this yields the $n^{k/4+o(k)}$-time algorithm mentioned in the introduction. However, Ahal and Rabinovich also observed that random permutation incidence graphs are expanders, so there are permutation incidence graphs with linear treewidth. This means that in the general case, the treewidth-based algorithms cannot (asymptotically) beat the trivial $n^{\mathcal{O}(k)}$-time algorithm.

Note that the algorithms mentioned require an appropriate tree decomposition of the incidence graph as part of the input. However, it is possible to compute a minimal tree decomposition in $n^{\mathcal{O}(\mathrm{tw}(G))}$ time (see, e.g., Berndt [4]), which means the treewidth-based algorithms still have running time $n^{\mathcal{O}(\mathrm{tw}(G_\pi))}$ without having to show how to find the tree decomposition.

In this chapter, we will explore how the treewidth-based algorithms perform when we restrict the pattern π to a principal permutation class. To be able to state our results more briefly, we will call the treewidth of the incidence graph of a permutation π just the *treewidth of* π. Given a permutation class $\mathcal{C}$, we define $\mathrm{tw}_{\mathcal{C}}(n)$, the *maximal treewidth of* $\mathcal{C}$, to be the maximal treewidth of an n-permutation $\pi \in \mathcal{C}$. Our work contributes towards a complete characterization of the maximal treewidth of each principal class, leaving open essentially only seven cases.

There are two crucial observations that reduce the infinite number of principal classes down to a manageable number of cases. First, two symmetric permutations have isomorphic incidence graphs (see section 2.3), so we only need to consider one permutation out of each symmetry group. Second, by corollary 2.3 (iv), if we can prove a lower bound for $\mathrm{tw}_{\mathrm{Av}(\sigma)}$, it also applies to $\mathrm{tw}_{\mathrm{Av}(\sigma')}$ for all σ' with $\sigma \preceq \sigma'$. In particular, lower bounds for $\mathrm{tw}_{\mathrm{Inc}_k}$ also apply to $\mathrm{tw}_{\mathrm{Dec}_k}$ by symmetry and thus apply to all $\mathrm{tw}_{\mathrm{Av}(\sigma)}$ with $|\sigma| > k^2$ (see section 2.5).

In this chapter, we prove upper and lower bounds on the maximal treewidth of specific

principal permutation classes. Section 3.1 presents the already solved case of *separable* permutations. Section 3.2 gives upper bounds by showing that incidence graphs of 2-monotone permutations are planar or almost-planar. Note that $\mathrm{Av}(321) = \mathrm{Inc}_2 \subseteq \mathrm{Mon}_2$.

Theorem 3.1. *The maximal treewidth of* Mon_2 *is* $\mathcal{O}(\sqrt{n})$.

In section 3.3, we give lower bounds of $\Omega(\sqrt{n})$ on the maximal treewidth of Av(321), Av(3412) and Av(3142) by embedding *grid graphs* as minors (see, e.g., Kozma [22]). It can be verified that *every* permutation of size at least 4 contains one of the three permutations 321, 3412 and 3142 or their symmetries. Thus, we obtain

Theorem 3.2. *The maximal treewidth of* $\mathrm{Av}(\sigma)$ *is* $\Omega(\sqrt{n})$ *if* $\sigma \in \{123, 321\}$ *or* $|\sigma| \geq 4$.

We remark that this result strengthens the dichotomy of Jelínek and Kynčl [19], who showed that $\mathrm{Av}(\sigma)$-Pattern PPM is NP-complete in the cases mentioned in theorem 3.2, and otherwise polynomial-time solvable.

Finally, in section 3.4 we present a technique to embed *arbitrary* permutation incidence graphs as minors into other permutation incidence graphs of specific permutation classes. This implies the following bounds.

Theorem 3.3. *The maximal treewidth of* $\mathrm{Av}(\sigma)$ *is* $\Omega(n/\log n)$ *if* $|\sigma| \geq 4$ *and* σ *is not in the same symmetry group as one of* 3412, 3142, 4213, 4123, 42153, 41352 *and* 42513.

Figure 3.1 shows a summary of the bounds currently known.

σ	$\mathrm{tw}_{\mathrm{Av}(\sigma)}$	Comments
21, 312	$\Theta(1)$	Separable (section 3.1), due to Ahal and Rabinovich [1].
321	$\Theta(\sqrt{n})$	Planar incidence graph (theorem 3.1) and contains grid graph (lemma 3.4).
3412, 3142	$\Omega(\sqrt{n})$	Contains grid graph (lemma 3.5).
4213, 4123, 42153, 41352, 42513	$\Omega(\sqrt{n})$	Contains 123 or 321.
4321, 4312, 4231, 45123, 32541, 426153	$\Omega(n/\log n)$	Embeds arbitrary permutation incidence graphs. (section 3.4.3)
All other symmetry groups	$\Omega(n/\log n)$	Contains one of 4321, 4312, 4231, 45123, 32541, 426153 or their symmetries.

Figure 3.1: Bounds on the maximal treewidth of principal permutation classes. Only one permutation out of each symmetry group is listed. The cases where no tight bounds (up to a logarithmic factor) are known are marked in bold.

3.1 Separable permutations

Separable permutations are the permutations that avoid both 3142 and 2413. Alternatively, they can be characterized as the smallest nonempty permutation class which is closed under direct sum and skew sum [30]. Av(3142, 2413)-Pattern PPM is an extensively studied special case of PPM [1, 2]. Bose et al. [8] gave an $\mathcal{O}(kn^6)$-time algorithm, which was shortly afterwards improved to $\mathcal{O}(kn^4)$ by Ibarra [17]. As remarked by Ahal and Rabinovich [1, p. 643], the treewidth of separable permutations is at most 7.

Let $\sigma \notin \{123, 321\}$ be a permutation of length at most 3. Observe that $\sigma \preceq 3142$ and $\sigma \preceq 2413$. Thus $\mathrm{Av}(\sigma) \subseteq \mathrm{Av}(3142, 1324)$, and therefore $\mathrm{tw}_{\mathrm{Av}(\sigma)} \leq 7$.

3.2 Planar and almost-planar incidence graphs

In this section, we provide $\mathcal{O}(\sqrt{n})$ upper bounds on the treewidth of 2-monotone permutations. The results of this section and the following section have been published in Berendsohn et al. [3] in a more compact form. The crucial observation is that planar graphs of order n have treewidth $\mathcal{O}(\sqrt{n})$ by the planar separator theorem [7, 24].

2-increasing permutations. Let $\pi \in \mathrm{Inc}_k$. Recall that G_π consists of the horizontal and the vertical Hamiltonian path. Fix a direction for each path. The partitioning of π into k increasing subsequences yields a partition of S_π into k sequences $T_1, T_2, \ldots, T_k$ such that for each $i \in [k]$, both Hamiltonian paths are monotone on T_i. We call $T_1, T_2, \ldots, T_k$ the *tracks* of S_π respectively π.

We now prove that each 2-increasing permutation has a planar incidence graph. This implies the same for 2-decreasing permutations. Let $\pi \in \mathrm{Inc}_2$. To see that G_π is planar, we arrange the two tracks on a line l in the plane, the first track in reverse, followed by the second track. Given a Hamiltonian path respecting the order of the two tracks, we can draw it on one side of l without crossing. Drawing each of the two Hamiltonian paths of G_π on one side of l yields a planar drawing. See fig. 3.2 (left) for an example.

We remark that this proof additionally shows that G_π has *book thickness* at most 2. Indeed, this is true for every planar incidence graph, as permutation incidence graphs are Hamiltonian and the book thickness of Hamiltonian planar graphs is at most 2. [5]

2-monotone permutations. We now make a slight detour from principal permutation classes, and consider the 2-monotone permutations Mon_2. While the incidence graphs of 2-monotone permutations are not necessarily planar, their incidence graphs can be drawn with only one crossing. We have already shown that permutations consisting of two increasing or two decreasing subsequences are planar. Consider a permutation π that can be partitioned into one increasing and one decreasing subsequence. We show that the straight-line drawing of G_π that draws vertices according to S_π has at most one crossing. For an example, see fig. 3.2 (right). The drawing has only one crossing, but contracting all edges not involving the 4 "central" points yields K_5, thus the graph is not planar by Wagner's theorem.

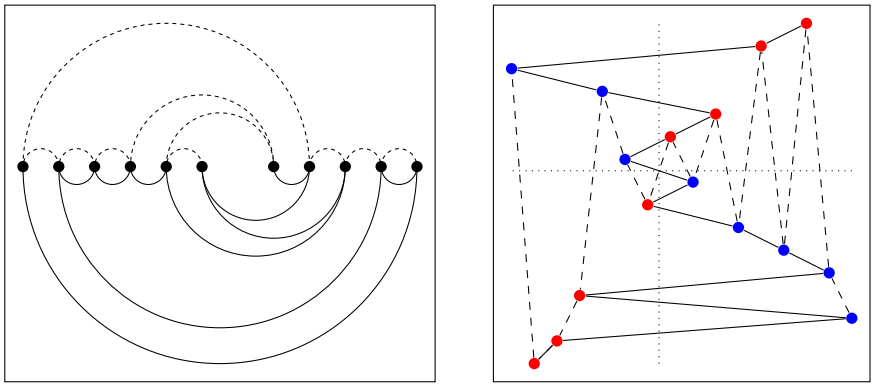

Figure 3.2: (*left*) A planar drawing of a 2-track graph, with one of the two Hamiltonian paths drawn with dashed arcs. Note that edges contained in both Hamiltonian paths are drawn twice for clarity. (*right*) A drawing of the incidence graph of a 2-monotone permutation. Red and blue dots indicate an increasing (resp. decreasing) subsequence.

Divide S_π by one horizontal and one vertical line, such that each of the resulting four sectors contains a monotone sequence. More precisely, the top left and bottom right sectors contain decreasing subsequences, and the other two sectors contain increasing subsequences. Let $e = \{u, v\}$ be an edge of the horizontal Hamiltonian path, i.e. $|u.x - v.x| = 1$. Let $u.x < v.x$ and without loss of generality, assume $u.y < v.y$. Suppose there is some edge $f = \{s, t\}$ that intersects e. The edge f must come from the vertical Hamiltonian path. Let $s.x < t.x$. This implies $s.y > t.y$, as otherwise π contains the pattern $(2, 1, 4, 3)$ and cannot consist of an increasing and a decreasing subsequence. Now (s, u, v, t) form the pattern $(3, 1, 4, 2)$, thus s and t belong to the decreasing and u and v to the increasing subsequence.

It is now easy to see that s, u, v, t must be in pairwise distinct sectors, and u (s, t, v) is the unique rightmost (bottommost, topmost, leftmost) point of the bottom left (top left, bottom right, top right) sector. We have shown:

Theorem 3.1. *The maximal treewidth of* Mon_2 *is* $\mathcal{O}(\sqrt{n})$.

3.3 Embedding Grid Graphs

Let $m, n \in \mathbb{N}$. The $m \times n$ *grid graph* is the graph $G_{m,n} = (V, E)$ where $V = [m] \times [n]$ and $E = \{\{u, v\} \mid u, v \in V, |u.x - v.x| + |u.y - v.y| = 1\}$.

The treewidth of an $m \times n$ grid graph is known to be exactly $\min\{m, n\}$ [7, Corollary 89]. Moreover, if G is a minor of H, then $\mathrm{tw}(G) \leq \mathrm{tw}(H)$ [7, Lemma 16]. In this section, we embed grid graphs of size roughly $\sqrt{n} \times \sqrt{n}$ as minors into the incidence graphs of certain n-permutations, thus giving a lower bound of $\mathcal{O}(\sqrt{n})$ on the treewidth of respective classes.

Lemma 3.4. *The maximal treewidth of* $\mathrm{Inc}_2 = \mathrm{Av}(321)$ *is* $\Omega(\sqrt{n})$.

Proof. Let m be even. We explicitly construct a set $S \subseteq \mathbb{N}^2$ with $2m^2$ points in general position such that $\pi = \mathrm{red}(S) \in \mathrm{Inc}_2$ and $G_\pi = G_S$ contains a $m \times m$ grid graph as a *subgraph*, and thus minor. For $i \in [m^2]$, let

$$a_i = \begin{cases} (2i-1, 2i-1), & \text{if } i \text{ is odd,} \\ (2i, 2i-2), & \text{if } i \text{ is even,} \end{cases}$$
$$b_i = \begin{cases} (2i, 2m+2i-2), & \text{if } i \text{ is odd,} \\ (2i-1, 2m+2i-1), & \text{if } i \text{ is even.} \end{cases}.$$

Let $A = \{a_1, a_2, \ldots, a_m\}$, $B = \{b_1, b_2, \ldots, b_m\}$ and let $S = A \cup B$. Observe that $\mathrm{red}(A)$ and $\mathrm{red}(B)$ each are 1-increasing and S is in general position, thus $\mathrm{red}(S) \in \mathrm{Inc}_2$. We now relabel the vertices to show the contained grid. For $j \in [m/2]$ and $k \in [m]$, set

$$g_{2j-1,k} = a_{(j-1)m+k}$$
$$g_{2j,k} = b_{(j-1)m+k}$$

For $i \in [m^2]$, the points (i) a_i and a_{i+1} are neighbors (if $i < m^2$); (ii) a_i and b_i are neighbors; (iii) a_{m+i} and b_i are neighbors (if $i \leq m^2 - m$). This implies, for $j \in [m/2]$ and $K \in [m]$, that (i) $g_{j,k}$, $g_{j,k+1}$ are neighbors (ii) $g_{2j-1,k}$, $g_{2j,k}$ are neighbors and (iii) $g_{2j+1,k}$, $g_{2j,k}$ are neighbors(as m is even). This concludes the proof that G_S has a grid of size $m \times m$ as a subgraph. □

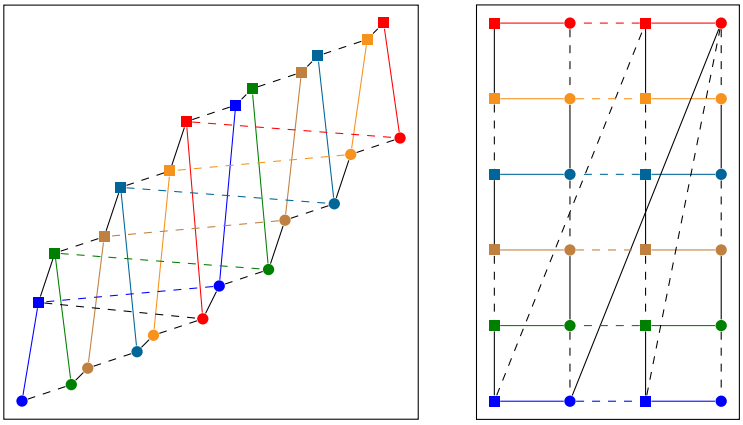

Figure 3.3: A 2-increasing permutation of length 32 whose incidence graph contains a 6×4 grid. Vertex shape indicates the track, colors are for emphasis of the grid structure.

We now embed a grid graph into an incidence graph of a permutation that avoids 3412, 2143, and 3142, thereby showing the following:

Lemma 3.5. *The maximal treewidth of* $\mathrm{Av}(3412, 2143, 3142)$ *is* $\Omega(\sqrt{n})$.

Proof. We construct a set $S \subseteq [4m^2] \times [4m^2]$ of $4m^2 - m$ points in general position such that S avoids 3412, 2143 and 3142 and contains a $m \times m$ grid graph as a minor. The construction is shown in fig. 3.3, and is inspired by the *spiral decomposition* of Jelínek and Kynčl [19].

Let

$$\begin{aligned} a_i &= (4m^2 - 2(m+i) + 1, 2i), && \text{for } i \in [m^2 - m], \\ b_i &= (2i - 1, 2i - 1), && \text{for } i \in [m^2], \\ c_i &= (2i, 4m^2 - 2i + 1), && \text{for } i \in [m^2], \\ d_i &= (4m^2 - 2i + 2, 4m^2 - 2i + 2), && \text{for } i \in [m^2]. \end{aligned}$$

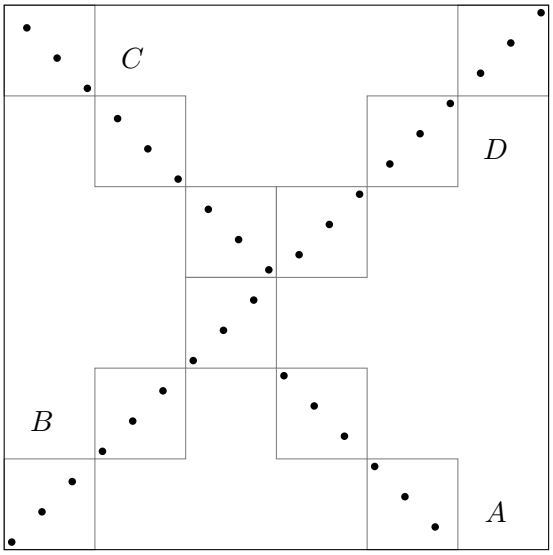

Figure 3.4: The point set S with $m = 3$.

Let $A = \{a_i \mid i \in [m^2 - m]$ and B, C, D be defined analogously. Let $S = A \cup B \cup C \cup D$ (see fig. 3.4 for an example). It can be easily checked that S is in general position. We now argue that the three permutations 3412, 2143, and 3142 are not contained in S. First observe that $B \cup D$ is increasing and $C \cup A$ is decreasing. The two permutations 3412 and 2143 cannot be partitioned into an increasing and a decreasing subsequence, so S avoids them. Now assume (c, b, d, a) is an occurrence of 3142 in S. As the naming suggests, we know $c \in C$, $b \in B$, $d \in D$ and $a \in A$ (check the only possible partitioning of 3142 into an increasing and a decreasing subsequence). We therefore can write $c_i = c$,

$b_j = b$, $d_k = d$ and $a_\ell = a$. Now

$$\begin{aligned}
c_i.x < b_j.x &\implies 2i < 2j - 1 \implies i \leq j,\\
b_j.y < a_\ell.y &\implies 2j - 1 < 2l \implies j \leq \ell,\\
c_i.y < d_k.y &\implies 4m^2 - 2i + 1 < 4m^2 - 2k + 2 \implies i \geq k,\\
d_k.x < a_\ell.x &\implies 4m^2 - 2k + 2 < 4m^2 - 2(m + \ell) + 1 \implies k > m + \ell.
\end{aligned}$$

This implies $i + m \leq j + m \leq \ell + m < k \leq i$, a contradiction.
It remains to show that G_S has a $m \times m$ grid minor. Let $g_i = \{a_i, b_i, c_i, d_i\}$ for $i \in [m^2 - m]$ and let $g_i = \{b_i, c_i, d_i\}$ for $m^2 - m < i \leq [m^2]$. We claim that we can contract each set g_i into a single vertex, and doing so creates a graph G' that is a grid with some additional edges that we ignore. Let g'_i be the vertex of G' that corresponds to the set g_i.

Observe that for each i where the respective points are defined, the three pairs (a_i, b_i), (b_i, c_i) and (c_i, d_i) are neighbors. This means we can indeed contract the set g_i into one vertex. Moreover, for each $i \in [m - 1]$, the points c_i and b_{i+1} are neighbors, and for $i \in [m^2 - m]$, the points a_i and d_{m+i} are neighbors. This means there is an edge from g_i to g_{i+1} respectively to g_{m+i}, so g'_i is adjacent to g'_{i+1} respectively to g'_{m+i} in G. Now G contains a grid graph where the j-th row consists of the vertices $g'_{m(j-1)+1}, g'_{m(j-1)+2}, \ldots, g'_{mj}$. □

Recall the main theorem of this section:

Theorem 3.2. *The maximal treewidth of* $\mathrm{Av}(\sigma)$ *is* $\Omega(\sqrt{n})$ *if* $\sigma \in \{123, 321\}$ *or* $|\sigma| \geq 4$.

To see that lemmas 3.4 and 3.5 imply theorem 3.2, note that by the Erdős-Szekeres theorem, each permutation of length at least 5 contains 123 or 321. It can be easily verified manually or computationally that 3412, 3142 and their reverses are the only 4-permutations which do not contain 123 or 321. The lower bound we proved for Av(321), Av(3412) and Av(3142) thus also applies to $\mathrm{Av}(\sigma)$ for an arbitrary σ of length at least 4.

3.4 Embedding arbitrary permutation graphs

In this section, we give almost-linear lower bounds for the maximal treewidth of almost all principal permutation classes. We say an n-permutation π can be *embedded* into a permutation ρ if G_π is a minor of G_ρ. For a constant $c \in \mathbb{R}$, we say π can be *c-embedded* into a permutation class $\mathcal{C}$ if there is a permutation ρ of length at most cn such that π can be embedded into ρ. We present a technique to embed an *arbitrary* m-permutation π into an $\mathcal{O}(m \log m)$-permutation that avoids a fixed pattern σ, meaning π can be $\mathcal{O}(\log m)$-embedded into $\mathrm{Av}(\sigma)$. As the treewidth of an m-permutation can be $\Theta(m)$, we know that $\mathrm{Av}(\sigma)$ contains $\mathcal{O}(m \log m)$-permutations with treewidth $\Omega(m)$, which yields an $\Omega(n/\log n)$ lower bound for the treewidth of $\mathrm{Av}(\sigma)$.[1]

The technique can be summarized as follows: we first show that every n-permutation can be written as the product of $\mathcal{O}(\log n)$ so-called *split permutations*. Then we show

[1] Set $m = n/\log n$ and note that then $m \log m \in \Theta(n)$.

that there are a number of *staircase classes* in which we can embed a product of split permutations. Finally, we show that for almost all permutations σ, the class $\mathrm{Av}(\sigma)$ contains one of those staircase classes.

3.4.1 Splitting permutations

Given an n-permutation π and an index set $I \subseteq [n]$, we write $\mathrm{split}(\pi, I)$ for the permutation that arises from π when taking all values with index in I and moving them to the front of the permutation [1]. Observe that $\mathrm{split}(\pi, I) = \pi \circ \mathrm{split}(\mathrm{Id}_n, I)$. We call $\mathrm{split}(\mathrm{Id}_n, I)$ a *split permutation*.

Lemma 3.6. *Each permutation of length at most $2^n - 1$ can be written as the product of n split permutations.*

Proof. Let π be an m-permutation with $m \leq 2^n - 1$. Observe that each value in π can be written as an n-bit string. We index the bits of such a value from least significant to most significant, i.e. the first bit is the least significant.

The idea is to sort the inverse permutation π^{-1} by radix sort [12, section 8.3]. Set $\rho_0 = \pi^{-1}$. In the i-th of n steps, we examine the i-th bit of each value of ρ_{i-1}. Then we move all values where the i-th bit is 0 to the front, say those with indices $I_i \subseteq [m]$, and let ρ_i be the result. In the end, $\rho_n = \mathrm{Id}_n$. We have $\rho_i = \mathrm{split}(\rho_{i-1}, I_i) = \rho_{i-1} \circ \mathrm{split}(\mathrm{Id}_n, I_i)$, and thus

$$\begin{aligned} \mathrm{Id}_n = \rho_n &= \pi^{-1} \circ \mathrm{split}(\mathrm{Id}_n, I_1) \circ \mathrm{split}(\mathrm{Id}_n, I_2) \circ \cdots \circ \mathrm{split}(\mathrm{Id}_n, I_n) \\ \implies \pi &= \mathrm{split}(\mathrm{Id}_n, I_1) \circ \mathrm{split}(\mathrm{Id}_n, I_2) \circ \cdots \circ \mathrm{split}(\mathrm{Id}_n, I_n). \end{aligned}$$

□

We now prove a technical lemma that we use afterwards to make another observation about split permutations. First, we need to extend notation regarding sequences. Let ω be a sequence of length n. Then we interpret ω as a *function* analogously to permutations, i.e. ω maps $i \in [n]$ to the i-th value of ω.

Lemma 3.7. *Let ω_1 be a sequence of length m and let ω_2 be a sequence of length n such that $\omega_1\omega_2$ is an $(m+n)$-permutation. Let π_1 be an m-permutation and let π_2 be an n-permutation. Then*

$$(\omega_1\omega_2) \circ (\pi_1 \oplus \pi_2) = (\omega_1 \circ \pi_1)(\omega_2 \circ \pi_2).$$

Proof. Let $i \in [m+n]$. If $i \leq m$, then $(\omega_1\omega_2) \circ (\pi_1 \oplus \pi_2)$ maps i to $\omega_1(\pi_1(i))$. If $i > m$, then $(\omega_1\omega_2) \circ (\pi_1 \oplus \pi_2)$ maps i to $\omega_2(\pi_2(i-m))$. This implies the claim. □

Recall that $\pi \in (\mathcal{C}_1|\mathcal{C}_2)$ if and only if $\pi = \omega_1\omega_2$ for some ω_1, ω_2 with $\mathrm{red}(\omega_i) \in \mathcal{C}_i$ for $i \in \{1, 2\}$, and that $\mathcal{C}^2 = \mathcal{C} \circ \mathcal{C}$. We now prove an observation about split permutations that will be used later.

Lemma 3.8. *Let σ be a split permutation. Then $\sigma \in \mathrm{Inc}_2 \cap (\mathrm{Inc}_1|\mathrm{Dec}_1)^2 \cap (\mathrm{Dec}_1|\mathrm{Inc}_1)^2$.*

Proof. Let $I \subseteq [n]$ such that $\sigma = \text{split}(\text{Id}_n, I)$. We can write $\sigma = \omega_1\omega_2$ where ω_1 and ω_2 are increasing sequences and $k = |\omega_1| = |I|$. This already makes it clear that $\sigma \in \text{Inc}_2$. Now let $\rho_1 = \omega_1^{\mathsf{R}}\omega_2$ and let $\rho_2 = \text{Id}_k^{\mathsf{R}} \oplus \text{Id}_{n-k}$. We have $\rho_1, \rho_2 \in (\text{Inc}_1|\text{Dec}_1)$ and, by lemma 3.7,

$$\rho_1 \circ \rho_2 = (\omega_1^{\mathsf{R}}\omega_2) \circ (\text{Id}_k^{\mathsf{R}} \oplus \text{Id}_{n-k}) = (\omega_1^{\mathsf{R}} \circ \text{Id}_k^{\mathsf{R}})(\omega_2 \circ \text{Id}_{n-k}) = \omega_1\omega_2 = \sigma.$$

Analogously, if we let $\rho_1 = \omega_1\omega_2^{\mathsf{R}}$ and $\rho_2 = \text{Id}_k \oplus \text{Id}_{n-k}^{\mathsf{R}}$, then $\rho_1, \rho_2 \in (\text{Dec}_1|\text{Inc}_1)$ and $\rho_1 \circ \rho_2 = \sigma$. □

3.4.2 Staircase classes

We will define staircase classes using the notion of *generalized grid classes* due to Vatter [29]. Let $M = (\mathcal{P}_{i,j})_{i,j\in[n]}$ be a $n \times n$ matrix of permutation classes. To follow the conventions we use for point sets, when referring to an entry in M, the first coordinate indexes the columns from left to right and the second coordinate indexes the rows from bottom to top. Figure 3.5 shows a matrix where each entry is labeled in our notation.

$$\begin{pmatrix} (1,2) & (2,2) & (3,2) \\ (1,1) & (2,1) & (3,1) \end{pmatrix}$$

Figure 3.5: A 3×2 matrix where each entry is labeled with its indices.

The *generalized grid class* $\text{Grid}(M)$ is the class of permutations that contains exactly those permutations π for which the following holds: S_π can be partitioned by a $n \times n$ grid such that the cell (i, j) contains a point set isomorphic to the point set of a permutation in $\mathcal{P}_{i,j}$. More formally, there are integers $1 = c_1 \leq c_2 \leq \cdots \leq c_{n+1} = n + 1$ and $1 = r_1 \leq r_2 \leq \cdots \leq r_{n+1} = n + 1$ such that for each i, j, there is a (possibly empty) permutation $\rho \in \mathcal{P}_{i,j}$ such that the set $S_\pi \cap ([c_i, c_{i+1} - 1] \times [r_i, r_{i+1} - 1])$ is isomorphic to S_ρ. We call these integers a *gridding* of π respectively S_π and define the *cell* with coordinates (i, j) as $C_{i,j} = [c_i, c_{i+1} - 1] \times [r_i, r_{i+1} - 1]$.

Staircase classes are the generalized grid classes of where all diagonal entries are Inc_1 and that otherwise only have nonempty entries next to the diagonal. Let $\mathfrak{T} = \mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$ and $\mathfrak{R} = \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n$ be sequences of permutation classes. Let the *staircase class* $\text{Stair}(\mathfrak{T}, \mathfrak{R}) = \text{Grid}(M)$ where $M = (\mathcal{P}_{i,j})_{i,j\in[n+1]}$ and for all $i, j \in [n+1]$,

$$\begin{aligned}
&\mathcal{P}_{i,i} = \text{Inc}_1 && \text{for each } i \in [n], \\
&\mathcal{P}_{i+1,i} = \mathcal{R}_i && \text{for each } i \in [n-1], \\
&\mathcal{P}_{i,i+1} = \mathcal{T}_i && \text{for each } i \in [n-1], \text{ and} \\
&\mathcal{P}_{i,j} = \emptyset && \text{for each } i, j \in [n] \text{ with } |i - j| \geq 2.
\end{aligned}$$

We call n the *height* of $\text{Stair}(\mathfrak{T}, \mathfrak{R})$. If $\pi \in \text{Stair}(\mathfrak{T}, \mathfrak{R})$ and $C_{i,j}$ is the cell with coordinates (i, j) in the corresponding gridding, we say $C_{i+1,i}$ is a *right-lane* cell and $C_{i,i+1}$ is a *top-lane* cell for $i \in [n]$. We say $C_{i,i}$ is a *middle-lane* cell for $i \in [n + 1]$. Figure 3.6 shows the matrix M of a staircase class with height 3.

$$\begin{pmatrix} \emptyset & \emptyset & \mathcal{T}_3 & \mathrm{Inc}_1 \\ \emptyset & \mathcal{T}_2 & \mathrm{Inc}_1 & \mathcal{R}_3 \\ \mathcal{T}_1 & \mathrm{Inc}_1 & \mathcal{R}_2 & \emptyset \\ \mathrm{Inc}_1 & \mathcal{R}_1 & \emptyset & \emptyset \end{pmatrix}$$

Figure 3.6: The 4×4 matrix of a staircase class of height 3.

We now show that (most) staircase classes of height n can n-embed the product of n permutations which obey certain restrictions (depending on the staircase class). For a staircase class $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$ and an $i \in [n]$, we define the *i-th embedding class* $\mathcal{C}_i$ to be the class of permutations π such that S_π can be partitioned in two sets S_i^{T} and S_i^{R} such that $\mathrm{red}(S_i^{\mathrm{T}})^{-1} \in \mathcal{T}_i$ and $\mathrm{red}(S_i^{\mathrm{R}}) \in \mathcal{R}_i$.

Lemma 3.9. *Let $\mathfrak{T} = \mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$ and $\mathfrak{R} = \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n$ be sequences of permutation classes, and let $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n$ be the embedding classes of* $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$.

Let $\sigma_1, \sigma_2, \ldots, \sigma_n$ be m-permutations such that $\sigma_i \in \mathcal{C}_i$ for each $i \in [n]$, and let the m-permutation $\pi = \sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_n$. Then π can be n-embedded into $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$.

Proof. Let $\pi_i = \sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_i$ for each $0 \le i \le n$, so $\pi_0 = \mathrm{Id}_m$ and $\pi_n = \pi$. Instead of a permutation in $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$, we will construct a point set S in general position such that $\mathrm{red}(S) \in \mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$. To ease understanding, we give S as a set of *colored points* from $[3m(n+1)] \times [3m(n+1)] \times [m]$, where the third component indicates the color of the point. We say that the graph G_S is colored accordingly.

We want to construct S with three properties:

(i) The subgraph of G_S induced by a single color shall be connected;

(ii) For each $i \in [m-1]$, there shall be an edge between a vertex of color i and a vertex of color $(i+1)$;

(iii) For each $i \in [m-1]$, there shall be an edge between a vertex of color $\pi(i)$ and a vertex of color $\pi(i+1)$.

These properties imply that if we take G_S and contract all edges between two vertices of the same color, we obtain a supergraph of G_π, which shows G_π is a minor of G_S.

Let $C_{i,j} = [3m(i-1)+1, 3mi] \times [3m(j-1)+1, 3mj]$. The cells $(C_{i,j})_{i,j \in [n+1]}$ induce a gridding of S which will show that $\mathrm{red}(S) \in \mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$.

First, each middle-lane cell will contain a point set isomorphic to S_{Id_m} where each of the m colors occurs exactly once, and the order of colors corresponds to π_{i-1}. Let us start by defining these points. For $i \in [n+1]$, let $f_i = 3m(i-1)$ and

$$M_i = \{(f_i + 3j - 1, f_i + 3j - 1, \pi_{i-1}(j)) \mid j \in [m]\} \subseteq C_{i,i}.$$

Observe that $\mathrm{red}(M_i) \in \mathrm{Inc}_1$. Note that we only used coordinates equivalent to -1 mod 3; this is to make room for points in the top-lane and right-lane cells.

We now define the remaining points. For $i \in [n]$, let

$$T_i = \{(f_i + 3p.y, f_{i+1} + 3p.x - 2, \pi_i(p.x)) \mid p \in S_i^{\mathrm{T}}\} \subseteq C_{i,i+1}, \text{ and}$$
$$R_i = \{(f_{i+1} + 3p.x - 2, f_i + 3p.y, \pi_i(p.x)) \mid p \in S_i^{\mathrm{R}}\} \subseteq C_{i+1,i}.$$

T_i is isomorphic to the inverse of S_i^{T} (note the interchangement of x and y), thus $\mathrm{red}(T_i) \in \mathcal{T}_i$. The set R_i is isomorphic to S_i^{R}, thus $\mathrm{red}(R_i) \in \mathcal{R}_i$. Finally, let

$$S = \bigcup_{i=1}^{n} M_i \cup T_i \cup R_i.$$

By what we observed while constructing S, we have $\mathrm{red}(S) \in \mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$.
We now prove (i). Let $c \in [m]$ be a color. We show that the vertices of color c induce a connected subgraph of G_S. First observe that for each $i \in [n]$, the two sets M_i and $T_i \cup R_i$ each contain exactly one point of color c. Let $i \in [n]$ and let $j \in [m]$ such that $c = \pi_{i-1}(j)$. Let $j' = \sigma_i^{-1}(j)$, so $c = \pi_i(\sigma_i^{-1}(j)) = \pi_i(j')$. Let $p \in M_i$, $q \in T_i \cup R_i$ and $p' \in M_{i+1}$ be the uniquely determined points of color c. We show that p is connected to q and q is connected to p' in G_S. We have

$$p = (f_i + 3j - 1, f_i + 3j - 1, c), \text{ and}$$
$$p' = (f_{i+1} + 3j' - 1, f_{i+1} + 3j' - 1, c).$$

We further know $(j', j) = (\sigma_i^{-1}(j), j) \in S_{\sigma_i} = S_i^{\mathrm{T}} \cup S_i^{\mathrm{R}}$. We first suppose $(j', j) \in S_i^{\mathrm{T}}$. This implies

$$q = (f_i + 3j, f_{i+1} + 3j' - 2, \pi_i(j')) \in T_i.$$

Observe that p, q are horizontal neighbors and q, p' are vertical neighbors.
Now suppose that $(j', j) \in S_i^{\mathrm{R}}$. This means

$$q = (f_{i+1} + 3j' - 2, f_i + 3j, \pi(j')) \in R_i.$$

In this case, p, q are vertical and q, p' are horizontal neighbors.
It remains to show (ii) and (iii). We claim that for each $i \in [n+1]$ and $j \in [m-1]$, the up to three points with abscissa in $[f_i + 3j - 2, f_i + 3j]$ have color $\pi_{i-1}(j)$. This even implies a stronger statement, namely that there is a vertical edge between vertices of the two colors $\pi_i(j)$ and $\pi_i(j+1)$.

For points on the middle or right lane, our claim follows directly from the definition of M_i and R_i. Let q be some point on the top lane, say with abscissa $f_i + 3p.y$ and color $\pi_i(p.x)$ for some point $p \in S_i^{\mathrm{R}}$. We know $p \in S_{\sigma_i}$, so $p = (\sigma_i^{-1}(j), j)$ for some $j \in [m]$. Thus, the color of q is $\pi_i(\sigma_i^{-1}(j)) = \pi_{i-1}(j)$, and the abscissa of q is $f_i + 3j$. This concludes the proof of our claim. □

We now want to bring lemmas 3.6 and 3.9 together to see which staircase classes can embed *all* permutations of appropriate size. First, this is true if all embedding classes contain the set of split permutations. However, we will later define staircase classes with

some more restricted embedding classes, so we need a more general lemma. The idea is that instead of requiring that each embedding class contains all split permutations, we require that the sequence of embedding classes can be partitioned into contiguous blocks, and that the composition of the classes in each block contains all split permutations.

Lemma 3.10. *Let $k \in \mathbb{N}_+$ and let* $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$ *be a staircase class with embedding classes $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{kn}$. If for every $i \in [n]$, the product $\mathcal{C}_{(i-1)n+1} \circ \mathcal{C}_{(i-1)n+2} \circ \cdots \circ \mathcal{C}_{im}$ contains all split permutations, then every permutation of length at most $2^n - 1$ can be n-embedded into* $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$.

Proof. Let π be an arbitrary permutation of length at most $2^n - 1$. Then, by lemma 3.8, π can be written as the product $\sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_n$ of n split permutations. By assumption, we can write σ_i as a product $\mu_1 \circ \mu_2 \circ \cdots \circ \mu_j$, where $\mu_j \in \mathcal{C}_j$ and $(i-1)n + 1 \leq j \leq im$. Thus, $\pi = \mu_1 \circ \mu_2 \circ \cdots \circ \mu_{km}$ and by lemma 3.9, we can embed π into $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$. □

3.4.3 Special staircase classes

We now proceed with the construction of staircase classes that avoid a fixed permutation σ and can n-embed arbitrary permutations of size at most $2^n - 1$, which implies $\mathrm{tw}_{\mathrm{Av}(\sigma)} \in \Omega(n/\log n)$. In all cases, the definition of $\mathfrak{T}$ and $\mathfrak{R}$ follows a certain pattern that repeats every one, two or three cells. We experimentally tested a number of such patterns and found the pattern that yields 426153-avoiding staircase classes this way.

Lemma 3.11. *Let $\sigma \in \{4321, 4231, 4312, 45123, 32541, 426153\}$. Then, for each $n \in \mathbb{N}$ with $6|n$, we can n-embed arbitrary permutations of length at most $2^n - 1$ into* $\mathrm{Av}(\sigma)$.

Proof. We present five construction schemes for staircase classes, one for each of the first four permutations σ mentioned in the lemma and one for the last two. Let $n \in \mathbb{N}$ be divisible by six, $\mathfrak{T} = \mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$ and $\mathfrak{R} = \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n$, where $\mathcal{T}_i$ and $\mathcal{R}_i$ are to be defined later in several different ways, depending on the permutation σ we want to avoid. In each case, we first show how $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$ can n-embed arbitrary permutations of length at most $2^n - 1$ by applying lemma 3.10, and then show that all permutations in $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$ avoid a certain σ.

Let $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n$ be the embedding classes of $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$. Given some $\pi \in \mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$, let $C_{i,j}$ for $i, j \in [n+1]$ be the cells of S_π induced by $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$. We write $T_i = C_{i,i+1}$, $R_i = C_{i+1,i}$ for $I \in [n]$. Note that $\mathrm{red}(M_i) \in \mathrm{Inc}_1$, $\mathrm{red}(T_i) \in \mathcal{T}_i$ and $\mathrm{red}(R_i) \in \mathcal{R}_i$.[2]

(i) Let $\mathcal{T}_i = \mathcal{R}_i = \mathrm{Inc}_1$ for $i \in [n]$. We have $\mathcal{C}_i = \mathrm{Inc}_2$ for each $i \in [n]$, and Inc_2 contains all split permutations by lemma 3.8. This means we can apply lemma 3.10 with $k = 1$, thus $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$ can embed arbitrary permutations of length at most $2^n - 1$.

We show $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R}) \subseteq \mathrm{Inc}_3 = \mathrm{Av}(4321)$. Let $\pi \in \mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$. Then we can partition S_π into the points belonging to the cells in the top lane, middle lane and bottom lane. Each of these three sets is increasing.

[2] We will never consider more than a single such π at the time, so the definitions do not need to be explicitly associated with π.

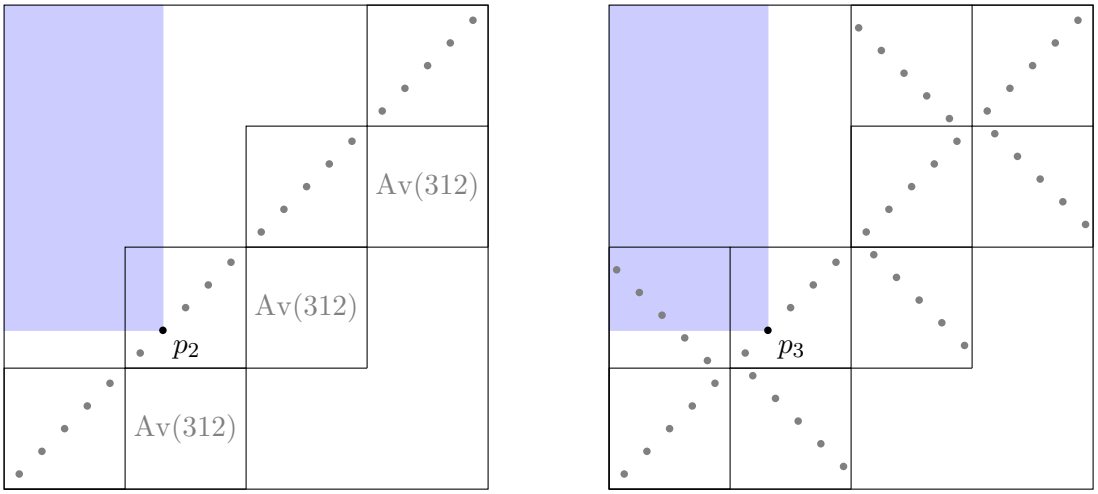

Figure 3.7: *(left)* An example of case (ii) where p_2 is on the middle lane. p_1 must be in the shaded area. *(right)* An example of case (iv) where p_3 is on the middle lane. p_1 and p_2 must be in the shaded area.

(ii) Let $\mathcal{T}_i = \emptyset$ and $\mathcal{R}_i = \mathrm{Av}(231)$. Each embedding class of $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$ is $\mathrm{Av}(231) \supseteq (\mathrm{Inc}_1|\mathrm{Dec}_1)$, thus the product of two embedding classes contains all split permutations by lemma 3.8. We can therefore apply lemma 3.10 with $k = 2$.

We show $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R}) \subseteq \mathrm{Av}(4231)$. Suppose $\pi \in \mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$ and (p_1, p_2, p_3, p_4) form an occurrence of 4231 in S_π. The top lane of S_π is empty. First, p_2 and p_3 cannot be in the middle lane, as this would mean there are no points in S_π that are above and left to p_2 (p_3) to accommodate p_1. As such, p_2 and p_3 must be in the right lane (see fig. 3.7, left). Let $p_2 \in R_i$. As p_4 is below and right to p_2, it must be in a cell that is not above and not left to R_i. There are no such cells except R_i itself. As $p_2.x < p_3.x < p_4.x$, the only right-lane cell that can contain p_3 is again R_i. But now p_2, p_3, p_4 are an occurrence of 213 in $R_i \cap S_\pi$, a contradiction.

(iii) Let $\mathcal{T}_i = \emptyset$ and $\mathcal{R}_i = \mathrm{Av}(312)$. Each embedding class is $\mathrm{Av}(312) \supseteq (\mathrm{Dec}_1|\mathrm{Inc}_1)$. As the product of two embedding classes contains all split permutations by lemma 3.8, we can apply lemma 3.10 with $k = 2$ as in (ii).

We show $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R}) \subseteq \mathrm{Av}(4312)$. Suppose $\pi \in \mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$ and (p_1, p_2, p_3, p_4) form an occurrence of 4312 in S_π. The top lane is again empty and by a similar argument as in (ii), the point p_2 cannot be in the middle lane. Let R_i be the (right-lane) cell of p_2. The points p_3 and p_4 both have to be below and right of p_2, so $p_3, p_4 \in R_i$. Now p_2, p_3, p_4 are an occurrence of 312 in $R_i \cap S_\pi$, a contradiction.

(iv) Let $\mathcal{T}_i = \emptyset$ for even $i \in [n]$, $\mathcal{T}_i = \mathrm{Dec}_1$ for odd $i \in [n]$ and $\mathcal{R}_i = \mathrm{Dec}_1$ for $i \in [n]$. Then, for an odd $i \in [n]$, we have $\mathcal{C}_i = \mathrm{Dec}_2$ and $\mathcal{C}_{i+1} = \mathrm{Dec}_1$, and thus $\mathcal{C}_i \circ \mathcal{C}_{i+1} = \mathrm{Inc}_2$, as the 2-increasing permutations are the reverses of the 2-decreasing permutations. By lemma 3.8, the product of two embedding classes contains all split permutations, so we apply lemma 3.10 with $k = 2$.

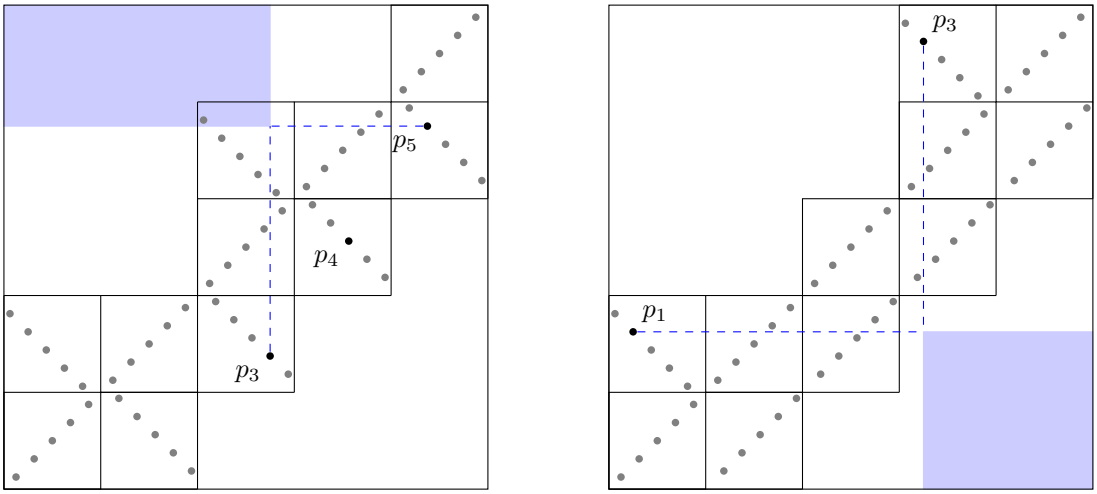

Figure 3.8: *(left)* An example of case (iv) where p_3, p_4, p_5 are in distinct right-lane cells. p_1 and p_2 must be in the shaded area. *(right)* An example of (v) where p_1 and p_3 are in distinct top-lane cells. p_5 must be in the shaded area.

We show $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R}) \subseteq \mathrm{Av}(45123)$. Suppose $\pi \in \mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$ and $(p_1, p_2, p_3, p_4, p_5)$ form an occurrence of 45123 in S_π. If one of p_3, p_4, p_5 is in the top or middle lane, then the ascending pair p_1, p_2 has to be in the same top-lane cell, a contradiction (see fig. 3.7, right). Thus, p_3, p_4, p_5 are on the right lane. As $R_i = \mathrm{Dec}_1$, they must be in pairwise distinct cells. Let $p_4 \in R_i$. Then p_1 and p_2 have to be above and left to the *entire cell* R_i (as $p_3, p_5 \notin R_i$), which means both have to be in the opposing top-lane cell T_i, again a contradiction (see fig. 3.8, left).

(v) For $i \in [n]$, let $\mathcal{T}_i = \mathrm{Dec}_1$ if $3|i$, and $\mathcal{T}_i = \emptyset$ otherwise, and let $\mathcal{R}_i = \mathrm{Inc}_1$. If $i \in \{0, 1, \ldots, n-3\}$ is divisible by three, then $\mathcal{C}_{i+1} = \mathcal{C}_{i+2} = \mathrm{Inc}_1$ and $\mathcal{C}_{i+3}$ is the class of permutations that can be partitioned into an increasing and a decreasing sequence. Now if $j \in \{0, 1, \ldots, n-6\}$ is divisible by six, we have $\mathcal{C}_{j+1} \circ \mathcal{C}_{j+2} \circ \mathcal{C}_{j+3} \circ \mathcal{C}_{j+4} \circ \mathcal{C}_{j+5} \circ \mathcal{C}_{j+6} \supseteq (\mathrm{Inc}_1|\mathrm{Dec}_1) \circ (\mathrm{Inc}_1|\mathrm{Dec}_1)$, which contains every split permutation by lemma 3.8, so we can apply lemma 3.10 with $k = 6$.

We now first show $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R}) \subseteq \mathrm{Av}(32541)$. Suppose $\pi \in \mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$ and the points $(p_1, p_2, p_3, p_4, p_5)$ form an occurrence of 32541 in S_π. The point p_2 cannot be on the right lane, as p_5 is below and right to p_2 and the whole right lane is increasing. As the middle lane is increasing, too, this implies that p_1 (which is above and left to p_2) must be on the top lane. By the same argument, p_4 is not on the right lane and p_3 is on the top lane. p_1 and p_3 must be in distinct top-lane cells, as no top-lane cells may contain an increasing pair. This means $p_1 \in T_i$ and $p_3 \in T_{i+3}$, which implies p_5 is below R_{i+1} and right to R_{i+2}, a contradiction (see fig. 3.8, right).

Second, we show $\mathrm{Stair}(\mathfrak{T}, \mathfrak{R}) \subseteq \mathrm{Av}(426153)$. Suppose $\pi \in \mathrm{Stair}(\mathfrak{T}, \mathfrak{R})$ and the points $(p_1, p_2, p_3, p_4, p_5, p_6)$ form an occurrence of 426153. By the same argument

as above, p_1 respectively p_3 must be above and left to the decreasing pair (p_2, p_4) respectively (p_5, p_6). Thus, p_1 and p_3 must be in different top-lane cells, which again means we cannot accommodate p_6, which would have to be below and right to both p_1 and p_3.

□

Corollary 3.12. *If* $\sigma \in \{4321, 4231, 4312, 45123, 32541, 426153\}$*, then the maximal treewidth of* $\mathrm{Av}(\sigma)$ *is* $\Omega(n/\log n)$*.*

Again due to the Erdős-Szekeres theorem, our lower bound for Av(4321) also applies to all classes $\mathrm{Av}(\sigma)$ where $|\sigma| \geq 10$. We computationally verified that, additionally, all permutations of size at least 6 as well as all permutations of size 4 and 5 *except* 3412, 3142, 4213, 4123, 42153, 41352 and 42513 contain one of the permutations covered above. This concludes the proof of

Theorem 3.3. *The maximal treewidth of* $\mathrm{Av}(\sigma)$ *is* $\Omega(n/\log n)$ *if* $|\sigma| \geq 4$ *and* σ *is not in the same symmetry group as one of* 3412*,* 3142*,* 4213*,* 4123*,* 42153*,* 41352 *and* 42513*.*

4 Hardness of #PPM

In the previous chapter, we were able to prove lower bounds for the treewidth of certain permutation classes. However, this does not immediately imply that the corresponding problems $\mathcal{C}$-Pattern PPM or their counting variants are hard. In this chapter, we prove this for almost all principal classes in the *counting* case, i.e. for $\mathcal{C}$-Pattern #PPM (assuming ETH). Again, it suffices to give lower bounds for a single class Inc_k by the Erdős-Szekeres theorem.

Theorem 4.1. Inc_5-Pattern #PPM *cannot be solved in* $f(k)n^{o(k/\log^4 k)}$ *time for any function* f*, unless ETH fails.*

Observe that due to the existence of the treewidth-based algorithms, this implies that the treewidth of 5-increasing n-permutations cannot be $o(n/\log^4 n)$, thus implying a weaker version of the results in Section 3.4 (weaker, as it has additional logarithmic factors and applies to fewer permutation classes).

We prove Theorem 4.1 by extending the result of Berendsohn, Kozma and Marx [3] that there is no algorithm for general #PPM that is substantially faster than the trivial algorithm (unless ETH fails). We restate and adapt parts of the original proof (which is in turn based on the work of Guillemot and Marx [16]).

We will make use of the known hard problem *partitioned subgraph isomorphism* (PSI). PSI is the well-known *subgraph isomorphism*, just that we are given a coloring χ of the large graph H, and each vertex of the small graph G must be mapped to a vertex of H that has a prescribed color. We formally define PSI later.

Lemma 4.2 (Marx [27], Bringman et al. [9])**.** *Unless ETH fails, PSI cannot be solved in* $f(k)n^{o(k/\log k)}$ *for any function* f*, where* n *is the number of vertices of* H *and* k *is the number of* edges *of* G*. This is true even we require* G *to have exactly as many vertices as edges.*

We further use the problem SCPPM (*surjective colored* permutation pattern matching) along with its counting version #SCPPM and their restricted versions $\mathcal{C}$-SCPPM and $\mathcal{C}$-#SCPPM. Roughly, SCPPM is equal to PPM with the exception that we additionally receive a coloring χ of the text τ, and have to hit every color when mapping the pattern π to τ. In each of the following sections, we prove one of the following three reductions.

$$\text{PSI} \leq \text{SCPPM} \leq \mathrm{Inc}_5\text{-Pattern SCPPM} \leq \mathrm{Inc}_5\text{-Pattern \#PPM}.$$

Each reduction corresponds to one of the lemmas below. First, we reduce from PSI to SCPPM.

Lemma 4.3 (Berendsohn et al. [3])**.** *An instance (G, H, χ) of PSI can be reduced to an instance (π, τ, χ') of SCPPM where $|\pi| \in \mathcal{O}(|E(G)|)$ and $|\tau| \in \mathcal{O}(|E(H)|)$ in polynomial time.*

Together with lemma 4.2, this implies an $f(k)n^{\Omega(k/\log k)}$ lower bound for SCPMM, where $k = |\pi|$ and $n = |\tau|$. Second, we reduce from SCPPM to the 5-increasing case.

Lemma 4.4. *An instance (π, τ, χ) of SCPPM can be reduced to an instance (π', τ', χ') where π' is 5-increasing, $|\pi'| \in \mathcal{O}(k \log^3 k)$ and $|\tau'| \in \mathcal{O}(n\sqrt{k} \log^4 k)$ where $n = |\tau|$ and $k = |\pi|$.*

This increases the size of the pattern by $\log^3 k$, thus only yielding a $f(k)n^{\Omega(k/\log^4 k)}$ lower bound for Inc_5-Pattern SCPPM. Note that the $\sqrt{k} \log^4 k$ factor on the text size vanishes in the function $f(k)$. Deciding is not harder than counting, so the same lower bound also applies to the counting case.

Finally, we give a general reduction from $\mathcal{C}$-Pattern #SCPPM to $\mathcal{C}$-Pattern #PPM for any permutation class $\mathcal{C}$. In contrast to the reductions above, we do not construct a single instance of $\mathcal{C}$-Pattern #PPM, but use several oracle calls to a presumed algorithm for $\mathcal{C}$-Pattern #PPM.

Lemma 4.5 (Berendsohn et al. [3])**.** *Let there be an algorithm that solves $\mathcal{C}$-Pattern #PPM in time $f(k)n^{\mathcal{O}(g(k))}$ for some functions f and g. Then $\mathcal{C}$-Pattern #SCPPM can be solved in time $h(k)n^{\mathcal{O}(g(k))}$ for some function h.*

These three lemmas imply theorem 4.1. Our main contribution is the reduction from SCPPM to Inc_5-SCPPM (lemma 4.4), the other reductions were already implicitly or explicitly proven in Berendsohn et al. [3].

4.1 PSI to SCPPM

The PSI problem [27] is defined as follows. The input consists of the graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ as well as a coloring $\chi\colon V_H \to V_G$ of vertices of H, using the vertices of G as colors. We have to decide whether there is a mapping $\phi\colon V_G \to V_H$ such that $\{u, v\} \in E_G$ if and only if $\{\phi(u), \phi(v)\} \in E_H$, and additionally, $\chi(\phi(v)) = v$ for all $v \in V_G$. In words, we have to embed G as a subgraph in H, but we partition the vertices of H (with χ) such that each vertex of G can only be mapped to a prescribed subset of vertices.

Let $k = |V_G|$ and $n = |V_H|$. Marx [27] showed that, assuming ETH, PSI cannot be solved in $f(k)n^{o(n \log n)}$ for any function f. Bringmann et al. [9] showed that this is true even if $|V_G| = |E_G|$.

In the SCPPM problem, we receive a k-permutation π, an n-permutation τ and a coloring $\chi\colon [n] \to [t]$, and we have to decide whether there is an order-preserving index mapping $\phi\colon [k] \to [n]$ from the indices of π to the indices of τ such that each color is hit at least once, i.e. $\chi(\phi([k])) = [t]$; in other words, $\chi \circ \phi$ is surjective.

Recall the lemma to be proven in this section:

Lemma 4.3 (Berendsohn et al. [3])**.** *An instance (G, H, χ) of PSI can be reduced to an instance (π, τ, χ') of SCPPM where $|\pi| \in \mathcal{O}(|E(G)|)$ and $|\tau| \in \mathcal{O}(|E(H)|)$ in polynomial time.*

Our proof of 4.3 has two parts. First, we present the polynomial-time construction of a SCPPM instance from a PSI instance. Then we show correctness of the construction, i.e. that either both instances are yes-instances or both are no-instances.

Construction. Let $G = (V_G, E_G)$, $H = (V_H, G_H)$ and $\chi\colon V_H \to V_G$ be an instance of PSI. We want to construct an instance (π, τ, χ') of SCPPM. To do this in a more intuitive way, we instead construct two point sets R_G and R_H and a coloring $\chi'\colon R_H \to [k]$, such that $(\mathrm{red}(R_G), \mathrm{red}(R_H), \chi'')$ is an input to SCPPM, where χ'' is constructed from χ' in the obvious way. The main idea is to build R_G to resemble the adjacency matrix M_G of G, and to build R_H to resemble the adjacency matrix M_H of H. Without loss of generality, we assume $V_G = [k]$ and $V_H = [n]$, and χ is monotonically increasing. We also assume χ is surjective (otherwise, (G, H, χ) is trivially a no-instance).

The two point sets R_G and R_H are both constructed the same way. Let $K \in \{G, H\}$ and let $m = |V_K|$. For $i \in [m]$, let

$$\begin{aligned} a_i^K &= (2m + (i-1)(m+2) + 1, 2i), \\ b_i^K &= (2m + i(m+2), 2i - 1), \\ c_i^K &= (2i, 2m + (i-1)(m+2) + 1), \text{ and} \\ d_i^K &= (2i - 1, 2m + i(m+2)). \end{aligned}$$

and for $i, j \in [m]$ with $\{i, j\} \in E_K$ or $i = j$, let

$$p_{i,j}^K = (2m + i(m+2) - j, 2m + j(m+2) - i).$$

Let $A_K = \{a_i^K \mid i \in [m]\}$, let B_K, C_K, D_K be defined accordingly and let $P_K = \{p_{i,j}^G \mid \{i, j\} \in E \text{ or } i = j \in [m]\}$. Let $R_K = A_K \cup B_K \cup C_K \cup D_K \cup P_K$.

The points $p_{i,j}^K$ correspond to the entries in the incidence matrix, while the other points are gadgets. The points in the i-th column are horizontally enclosed by a_i^K and b_i^K, so we call a_i^K and b_i^K the *bracketing points* of the i-th column. Similarly, the points in the i-th row are vertically enclosed by c_i^K and d_i^K, so we call c_i^K and d_i^K the bracketing points of the i-th row. Figure 4.1 shows an example.

It remains to define the coloring χ'. Let $\chi'\colon R_H \to [5k + 1]$ be defined as follows. For $i, j \in [n]$ with $i \neq j$, let

$$\begin{aligned} \chi'(a_i^H) &= \chi(i), \\ \chi'(b_i^H) &= k + \chi(i), \\ \chi'(c_i^H) &= 2k + \chi(i), \\ \chi'(d_i^H) &= 3k + \chi(i), \\ \chi'(p_{i,i}^H) &= 4k + \chi(i), \text{ and} \\ \chi'(p_{i,j}^H) &= 5k + 1. \end{aligned}$$

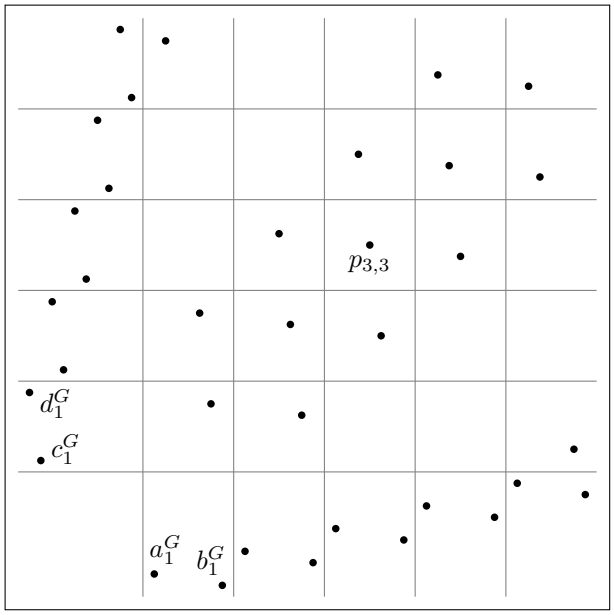

Figure 4.1: The point set R_G constructed from the graph $G = C_5$. Note that gaps have been added to draw the grid lines. Some points are exemplarily labeled.

The construction only requires polynomial time. Moreover, for $K \in \{G, H\}$, we have $|R_K| = 4|V_K| + |V_K| + |E_K| \in \mathcal{O}(|E_K|)$.

We proceed with the proof of correctness, i.e. we show that (G, H, χ) is a yes-instance of PSI if and only if (R_G, R_H, χ') is a yes-instance of SCPPM. We show the two directions of the equivalence separately.

Correctness ("only if"). Suppose (G, H, χ) is a yes-instance of PSI, and recall that we assumed $V_G = [k]$ and $V_H = [n]$. This means there is a mapping $\phi\colon [k] \to [n]$ such that $\{i, j\} \in E_G$ if and only if $\{\phi(i), \phi(k)\} \in E_H$ for all $i, j \in [k]$ and $\chi(\phi(i)) = i$ for all $i \in [k]$.

To show that (R_G, R_H, χ') is a yes-instance of SCPPM, we need to find a mapping $\psi\colon R_G \to R_H$ that preserves relative order and hits every color of R_H at least once. We define ψ as follows. For $i, j \in [k]$, let

$$\begin{aligned}
\psi(a_i^G) &= a_{\phi(i)}^H, \\
\psi(b_i^G) &= b_{\phi(i)}^H, \\
\psi(c_i^G) &= c_{\phi(i)}^H, \\
\psi(d_i^G) &= d_{\phi(i)}^H, \text{ and} \\
\psi(p_{i,j}^G) &= p_{\phi(i),\phi(j)}^H.
\end{aligned}$$

We now prove the two necessary properties of ψ.

Claim 4.6. *ψ preserves relative order.*

Proof. By assumption, χ is monotonically increasing. As $\chi(\phi(i)) = i$ for each $i \in [k]$, the mapping ϕ must also be monotonically increasing. Now, given that we constructed R_G and R_H in the same way, and considering the definition of ψ, it is easy to see that ϕ being monotonically increasing implies ψ preserves relative order. □

Claim 4.7. *ϕ hits every color of R_H at least once.*

Proof. We can assume G has an edge, otherwise the problem is trivial to solve. This means there is some point $p^G_{i,j}$, so ψ certainly hits the color $5k+1$.

Now let $i \in [k]$. By definition, we have

$$\begin{aligned}
\chi'(\psi(a^G_i)) &= \chi'(a^H_{\phi(i)}) = \chi(\phi(i)) = i,\\
\chi'(\psi(b^G_i)) &= \chi'(b^H_{\phi(i)}) = k + \chi(\phi(i)) = k + i,\\
\chi'(\psi(c^G_i)) &= \chi'(c^H_{\phi(i)}) = 2k + \chi(\phi(i)) = 2k + i,\\
\chi'(\psi(d^G_i)) &= \chi'(d^H_{\phi(i)}) = 3k + \chi(\phi(i)) = 3k + i, \text{ and}\\
\chi'(\psi(p^G_{i,i})) &= \chi'(p^H_{\phi(i),\phi(i)}) = 4k + \chi(\phi(i)) = 4k + i.
\end{aligned}$$

As such, ψ also hits all colors in $[5k]$. □

Correctness ("if") Now suppose (R_G, R_H, χ') is a yes-instance of SCPPM. Recall that this means there is a mapping $\psi\colon R_G \to R_H$ that preserves relative order and hits every color of R_H at least once. We have to construct a mapping $\phi\colon V_G \to V_H$ such that $\{i,j\} \in E_G$ if and only if $\{\phi(i), \phi(j)\}$ for each $i, j \in [k]$ and $\chi(\phi(i)) = i$ for all $i \in [k]$.

Before we construct ϕ, we prove some properties of ψ.

Claim 4.8. *$\psi(A_G \cup B_G) \subseteq A_H \cup B_H$ and $\psi(C_G \cup D_G) \subseteq C_H \cup D_H$ (and thus the preimage of P_H under ψ is a subset of P_G).*

Proof. Observe that $A_G \cup B_G$ consists of exactly the $2k$ lowest points in R_G. Moreover, $A_H \cup B_H$ consists of the $(2n)$ lowest points of R_H and contains exactly the points with colors in $[2k]$ (recall that χ is surjective). To hit all colors, ψ must therefore map $A_G \cup B_G$ to $A_H \cup B_H$. Essentially the same argument shows that $\psi(C_G \cup D_G) \subseteq C_H \cup D_H$.

□

Claim 4.9. *Let $P'_G = \{p^G_{i,i} \mid i \in [n]\} \subseteq P_G$ and $P'_H = \{p^H_{j,j} \mid j \in [n]\} \subseteq P_H$. Then the preimage of P'_H under ψ is exactly P'_G.*

Proof. Let P be the preimage of P'_H under ψ. We show that $P = P'_G$.

Observe that P'_H is an increasing point set (i.e. $\mathrm{red}(P'_H) \in \mathrm{Inc}_1$) and contains exactly the points of the k colors $4k+1, 4k+2, \ldots, 5k$. As such, P must also be increasing and contain at least k points. As $P'_H \subseteq P_H$, claim 4.8 implies that $P \subseteq P_G$. Indeed, the only increasing point set of size k in P_G is precisely P'_G. □

We now define $\phi\colon R_G \to R_H$ as the function that satisfies $\psi(p^G_{i,i}) = \psi(p^G_{\phi(i),\phi(i)})$. By claim 4.9, ϕ is well-defined. We now show that ϕ satisfies the three properties required by PSI.

Claim 4.10. *ϕ is monotonically increasing.*

Proof. Suppose it is not. Then there are $i, j \in [k]$ such that $i < j$ but $\phi(i) > \phi(j)$. This implies $p^G_{i,i}.x < p^G_{j,j}.x$, but also

$$\psi(p^G_{i,i}) = p^H_{\phi(i),\phi(i)}.x > p^H_{\phi(j),\phi(j)}.x = \psi(p^G_{j,j}),$$

by definition, which contradicts the assumption that ψ preserves relative order. □

Claim 4.11. *$\chi(\phi(i)) = i$ for each $i \in [k]$.*

Proof. We know that ψ hits all colors of R_H, in particular the colors $4k{+}1, 4k{+}2, \ldots, 5k$. Claim 4.9 implies that ψ maps the points $p^G_{i,i}$ to points of the colors $4k{+}1, 4k{+}2, \ldots, 5k$, i.e.

$$\begin{aligned}
&\{\chi'(\psi(p^G_{i,i})) \mid i \in [k]\} = \{4k+1, 4k+2, \ldots, 5k\} \\
&\implies \{\chi'(p^H_{\phi(i),\phi(i)}) \mid i \in [k]\} = \{4k+1, 4k+2, \ldots, 5k\} && \text{(by definition)} \\
&\implies \{4k + \chi(\phi(i)) \mid i \in [k]\} = \{4k+1, 4k+2, \ldots, 5k\} && \text{(by definition)} \\
&\implies \chi(\phi([k])) = [k],
\end{aligned}$$

i.e. $\chi \circ \phi$ is a bijection.

Further, χ is increasing (by assumption) and ϕ is increasing (by claim 4.10). As such, $\chi \circ \phi$ is increasing, implying that $\chi(\phi(i)) = i$ for each $i \in [k]$. □

Before we prove the last property (that ϕ maps edges of G to edges of H), we prove one more property of ψ.

Claim 4.12. *$\psi(a^G_i) = a^H_{\phi(i)}$ for all $i \in [k]$, and the same is true when replacing a with b, c, d.*

Proof. Consider the points $\psi(a^G_i)$ and $\psi(b^G_i)$. Both points are not in P_H by claim 4.8. As ψ preserves order, we have $\psi(a^G_i).y > \psi(b^G_i).y$ and $\psi(a^G_i), \psi(b^G_i)$ horizontally enclose the point $\psi(p^G_{i,i}) = p^G_{\phi(i),\phi(i)}$. The only pair of points in R_H that satisfies these conditions is $(a^H_{\phi(i)}, b^H_{\phi(i)})$. The argument for $\psi(c^G_i)$ and $\psi(d^G_i)$ is analogous. □

We finish the proof by showing that ϕ satisfies the last property required by PSI.

Claim 4.13. *For all $i, j \in [k]$, if $\{i, j\} \in E_G$, then $\{\phi(i), \phi(j)\} \in E_H$.*

Proof. Let $\{i, j\} \in E_G$ and consider the point $\psi(p_{i,j})$. We have

$$\begin{aligned}
&a^G_i.x < p^G_{i,j}.x < b^G_i.x, \text{ and} \\
&c^G_j.y < p^G_{i,j}.y < d^G_j.y,
\end{aligned}$$

implying

$$a^H_{\phi(i)}.x < \psi(p^G_{i,j}).x < b^H_{\phi(i)}.x, \text{ and}$$
$$c^H_{\phi(j)}.y < \psi(p^G_{i,j}).y < d^H_{\phi(j)}.y.$$

This means $\psi(p^G_{i,j}) = p^H_{\phi(i),\phi(j)} \in R_H$, which implies $\{\phi(i), \phi(j)\} \in E_H$. □

4.2 SCPPM to Inc_5-Pattern SCPPM

We prove lemma 4.4 (see below) in this section. The main idea is to reduce from an s-increasing pattern to a (roughly) $\sqrt{s}$-increasing pattern, while not increasing the size of the input too much. Getting down to 5-increasing patterns requires repeating this step roughly $\log\log k$ times, and additionally a slightly different reduction step.

We state the main lemma here and prove it in section 4.2.1.

Lemma 4.14. *An instance (π, τ, χ) of SCPPM where π is s-increasing can be reduced in polynomial time to another instance (π', τ', χ') where π' is $3\lceil\sqrt{s}\rceil$-increasing, $|\pi'| \leq 7|\pi|$ and $|\tau'| \leq 11\sqrt{s}n$.*

Using lemma 4.14, we cannot reduce SCPPM further than to Inc_{12}-Pattern SCPPM. To get down to 5-increasing patterns, we need a slightly different technique, which we present in section 4.2.2.

Lemma 4.15. *An instance (π, τ, ϕ) of SCPPM where π is s-increasing can be reduced in polynomial time to another instance (π', τ', ϕ') where π' is $(\lceil s/2\rceil + 2)$-increasing, $|\pi'| \leq 5|\pi|$ and $|\tau'| \leq 6|\tau|$.*

We now show how we can use these reduction steps to prove lemma 4.4.

Lemma 4.4. *An instance (π, τ, χ) of SCPPM can be reduced to an instance (π', τ', χ') where π' is 5-increasing, $|\pi'| \in \mathcal{O}(k\log^3 k)$ and $|\tau'| \in \mathcal{O}(n\sqrt{k}\log^4 k)$ where $n = |\tau|$ and $k = |\pi|$.*

Proof. We first apply $t = \lceil\log\log k\rceil$ times lemma 4.14. Let $(\pi, \tau, \chi) = (\pi_0, \tau_0, \chi_0)$ be an arbitrary instance of SCPPM, and let (π_i, τ_i, χ_i) be the result of applying lemma 4.14 to $(\pi_{i-1}, \tau_{i-1}, \chi_{i-1})$ for $i \in [t]$. Let s_i be minimal such that π_i is s_i-increasing.

Trivially, π is k-increasing, so $s_0 \leq k$. As $s_i \leq 4\sqrt{s_{i-1}}$ (as long as $s_{i-1} \geq 4$), we have

$$\begin{aligned} s_i &\leq 4\sqrt{s_{i-1}} \leq 4\sqrt{4\sqrt{s_{i-2}}} = 4^{1+1/2}s_{i-2}^{1/4} \\ &\leq \cdots < 4^2 s_0^{1/2^i} \leq 16k^{1/2^i}, \end{aligned}$$

and further

$$\begin{aligned}
&|s_t| \leq 16k^{1/2^t} \leq 16k^{1/(\log k)} \leq 32,\\
&|\pi_t| \leq 7|\pi_{t-1}| \leq \cdots \leq 7^t|\pi_0| = 7^t k \leq 8^t k \leq 8k\log^3 k,\\
&|\tau_t| \leq 11\sqrt{s_t}|\tau_{t-1}| \leq 11^2\sqrt{s_t}\sqrt{s_{t-1}}|\tau_{t-2}| \leq \cdots \leq 11^t|\tau_0|\prod_{i=1}^{t}\sqrt{s_i}\\
&\qquad \leq 11^t n\prod_{i=1}^{t} 16k^{1/2^{i+1}} < 16\cdot 11^t n\cdot k^{1/2} \leq 16n\sqrt{k}\log^4 k.
\end{aligned}$$

We now apply lemma 4.15 five times to obtain $(\pi_{t+i}, \tau_{t+i}, \chi_{t+i})$ for $i \in [5]$. Let s_{t+i} be minimal such that π_{t+i} is s_{t+i}-increasing. We have $s_{t+i+1} = 2\lceil s_{t+i}/2\rceil + 2$, and as such

$$s_t \leq 32 \implies s_{t+1} \leq 18 \implies s_{t+2} \leq 11 \implies s_{t+3} \leq 8 \implies s_{t+4} \leq 6 \implies s_{t+5} \leq 5.$$

Moreover, we have $|\pi_{t+5}| \leq 5^5|\pi_t| \in \mathcal{O}(k\log^3 k)$ and $|\tau_{t+5}| \leq 6^5|\tau_t| \in \mathcal{O}(n\sqrt{k}\log k)$. This concludes the proof. □

Observe that we cannot simply use lemma 4.15 $\Theta(\log n)$ times, as this would increase the pattern size too much.

4.2.1 Main reduction step

In this section, we prove

Lemma 4.14. *An instance (π, τ, χ) of SCPPM where π is s-increasing can be reduced in polynomial time to another instance (π', τ', χ') where π' is $3\lceil\sqrt{s}\rceil$-increasing, $|\pi'| \leq 7|\pi|$ and $|\tau'| \leq 11\sqrt{s}n$.*

Let $P = S_\pi$ and $T = S_\tau$. We assume we are given a coloring χ of the points in T, instead of a coloring of the indices of τ. As in the proof of lemma 4.3, instead of giving π', τ', χ', we will construct the two point sets P' and T' in general position and a coloring χ' of the points in T'. In the end, set $\pi' = \mathrm{red}(P')$ and $\tau' = \mathrm{red}(T')$, and modify χ' accordingly.

As π is s-increasing, we can partition P into $t = \lceil\sqrt{s}\rceil$ point sets $P_1, P_2, \ldots, P_t$, each $\lfloor\sqrt{s}\rfloor$-increasing.

We first sketch the general idea. We want to put each set P_i into its own "box", where the boxes are arranged diagonally from the bottom left to the top right. The result is a $\lfloor\sqrt{s}\rfloor$-increasing point set P''. The text will be *copied* t times, and the copies $T_1, T_2, \ldots, T_t$ arranged on the same diagonal. Let T'' be the resulting point set. Now if we have an order-preserving mapping ϕ from P to T, we can easily transform it into an order-preserving mapping ϕ'' from P'' to T'': if $p \in P_i$, we simply map p to the respective point in the i-th copy of $\phi(p)$.

However, the converse is not true. The problem is that we preserve the relative position of two points $p \in P_i$ and $q \in P_j$ only if $i = j$, but not if $i \neq j$. Basically,

with the construction as it stands, we only check whether the patterns P_i occur in T. What we want is to allow only mappings where we could take all mapped points in the t boxes of T'', stack all these boxes on top of each other and obtain a point set that is order-isomorphic to P.

We will achieve this by introducing certain gadgets left to all cells and certain gadgets below all cells, the exact workings of which will be explained in the correctness proof below. We now proceed with the formal construction, and later prove equivalence of the given and constructed instance. Recall that we need to construct the point sets P' and T' and the coloring χ' of T.

Construction of $\mathbf{P'}$. In preparation of the definition of P', we define *cells*, which include the boxes mentioned above and additional cells containing the gadgets. Let $k = |\pi|$, $\ell = 3k + 1$, and for $i, j \in \{0, 1, \ldots, t\}$,

$$C^{\mathrm{P}}_{i,j} = [i\ell + 1, (i+1)\ell] \times [j\ell + 1, (j+1)\ell].$$

For $i \in [t]$, the cell $C^{\mathrm{P}}_{i,i}$ is the i-th box containing the translation of P_i, the cells $C^{\mathrm{P}}_{0,i}$ contain the so-called *left* gadgets and the cells $C^{\mathrm{P}}_{i,0}$ contain the *bottom* gadgets. All other cells are empty.

Recall that we have partitioned P into t point sets $P_1, P_2, \ldots, P_t$, each of which is t-increasing. We map each point of P to 5 points in P', using the 5 mappings defined as follows for $p \in P_i$:

$$\begin{aligned}
\psi(p) &= (i\ell + 3p.x - 1, i\ell + 3p.y - 1) \in C^{\mathrm{P}}_{i,i},\\
\psi^{\mathrm{d}}(p) &= (3p.y - 2, i\ell + 3p.y - 2) \in C^{\mathrm{P}}_{0,i},\\
\psi^{\mathrm{u}}(p) &= (3p.y - 0, i\ell + 3p.y - 0) \in C^{\mathrm{P}}_{0,i},\\
\psi^{\ell}(p) &= (i\ell + 3p.x - 2, 3p.x - 2) \in C^{\mathrm{P}}_{i,0}, \text{ and}\\
\psi^{\mathrm{r}}(p) &= (i\ell + 3p.x - 0, 3p.x - 0) \in C^{\mathrm{P}}_{i,0}.
\end{aligned}$$

The mapping ψ moves the point to the appropriate box, the other mappings are used to construct the gadgets. Further, for each $j \in [t]$, we define the vertical and horizontal *anchor points*

$$\begin{aligned}
a^{\mathrm{v}}_j &= ((t+1)\ell + j - 1, j\ell), \text{ and}\\
a^{\mathrm{h}}_j &= (j\ell, (t+1)\ell + j - 1).
\end{aligned}$$

Finally, we set

$$P' = \psi(P) \cup \psi^{\mathrm{d}}(P) \cup \psi^{\mathrm{u}}(P) \cup \psi^{\mathrm{l}}(P) \cup \psi^{\mathrm{r}}(P) \cup \bigcup_{i=1}^{t} \{a^{\mathrm{v}}_i, a^{\mathrm{h}}_i\}.$$

Validity of $\mathbf{P'}$. We now show that P' contains at most $5k$ points, is in general position, and is $3\lceil\sqrt{n}\rceil$-increasing.

We map each of the k points from P to five points in P' and add the $2t$ anchor points. Thus, $|P'| = 5k + 2t \leq 7k$. It is easy to see that P' is in general position. Finally, for each $i \in [t]$, the sets $\psi^{\mathrm{d}}(P_i) \cup \psi^{\mathrm{u}}(P_i) \cup \{a_i^{\mathrm{h}}\}$ and $\psi^{\mathrm{l}}(P_i) \cup \psi^{\mathrm{r}}(P_i) \cup \{a_i^{\mathrm{v}}\}$ each are increasing. As mentioned in the informal introduction, $\psi(P)$ is $\lfloor\sqrt{s}\rfloor$-increasing. We conclude that P' consists of at most

$$\lfloor\sqrt{s}\rfloor + 2\lceil\sqrt{s}\rceil \leq 3\lceil\sqrt{s}\rceil$$

increasing sequences.

Construction of $\mathbf{T'}$. First, we define the cells as we did for P'. Let $n = |\tau|$, $m = 3n+1$ and $f = (t-1)m$. Let $g_0 = 0$, $g_i = f + (i-1)m$ for $i \in [t]$ and

$$C^{\mathrm{T}}_{i,j} = [g_i + 1, g_{i+1}] \times [g_j + 1, g_{j+1}]$$

for $i, j \in \{0, 1, \ldots, t\}$. The meaning of the cells corresponds to the meaning of the cells in P.

We use the $5t$ mappings defined as follows for $p \in T$ and $i \in [t]$:

$$\begin{aligned}
\omega_i(p) &= (f + im + 3p.x - 1, f + im + 3p.y - 1) \in C^{\mathrm{T}}_{i,i},\\
\omega_i^{\mathrm{d}}(p) &= (3p.y - 2, f + im + 3p.y - 2) \in C^{\mathrm{T}}_{0,i},\\
\omega_i^{\mathrm{u}}(p) &= (3p.y - 0, f + im + 3p.y - 0) \in C^{\mathrm{T}}_{0,i},\\
\omega_i^{\mathrm{l}}(p) &= (f + im + 3p.x - 2, 3p.x - 2) \in C^{\mathrm{T}}_{i,0}, \text{ and}\\
\omega_i^{\mathrm{r}}(p) &= (f + im + 3p.x - 0, 3p.x - 0) \in C^{\mathrm{T}}_{i,0}.
\end{aligned}$$

ω_i is responsible for the t copies, the other mappings again are used for the gadgets. The anchor points for T' are, for $j \in [t-1]$,

$$\begin{aligned}
a_j^{\mathrm{V}} &= (f + (t+1)m + j - 1, f + jm), \text{ and}\\
a_j^{\mathrm{H}} &= (f + jm, f + (t+1)m + j - 1).
\end{aligned}$$

Finally,

$$T' = \bigcup_{i=1}^{t} (\omega_i(T) \cup \omega_i^{\mathrm{d}}(T) \cup \omega_i^{\mathrm{u}}(T) \cup \omega_i^{\mathrm{l}}(T) \cup \omega_i^{\mathrm{r}}(T)) \cup \bigcup_{i=1}^{t-1} \{a_i^{\mathrm{V}}, a_i^{\mathrm{H}}\}.$$

Validity of $\mathbf{T'}$. We again observe that T' is in general position. The size of T' is $5t \cdot n + 2t \leq 5\lceil\sqrt{s}\rceil n + 2\lceil\sqrt{s}\rceil \leq 11\sqrt{s}n$. We remark that this can be somewhat improved when s is much smaller than n, but that would not affect our results in a meaningful way.

Construction of the coloring. Let $c \leq n$ be the number of colors of T. We now define $\chi' \colon T \to c + 2t - 1$.

Let $\chi'(\omega_i(p)) = \chi(p)$ for $p \in T$ and $i \in [t]$. For $j \in [t-1]$, let $\chi'(a_j^{\mathrm{V}}) = c + j$ and $\chi(a_j^{\mathrm{H}}) = c + t - 1 + j$. Now every anchor point has a unique color, distinct to all other points. We color the remaining (gadget) points with a single last color $c' = c + 2t - 1$. Figure 4.2 shows a diagram of the functions used here and in the correctness proof below. Figures 4.3 and 4.4 show example constructions of P' and T'.

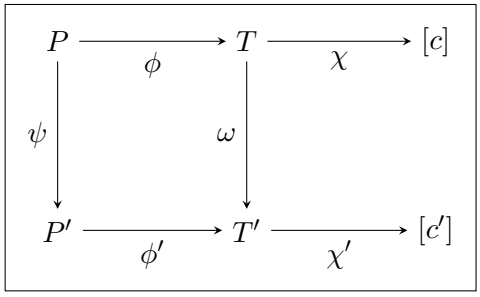

Figure 4.2: Diagram of the sets and functions (or families of functions) used in the reduction.

Before we proceed with the formal proof, let us informally describe how the construction solves the problem mentioned before, that the relative position of two points is lost if they are put into different boxes of P'. First, the anchor points ensure that each point in a grid cell $C_{i,j}^{\mathrm{P}}$ of P' has to be mapped into the corresponding grid cell $C_{i,j}^{\mathrm{T}}$ of T'. Now consider the two points $p \in P_2$ and $q \in P_1$ in fig. 4.3, and suppose that the mapping $\phi' \colon P' \to T'$ preserves relative positions.

We can imagine ϕ' as moving around the points of P'. As mentioned, the points have to stay in their cells. Also, each gadget of T' is a diagonal line, so we can only move the gadget points of P' on a diagonal line (indicated by the orange lines in fig. 4.3). Now the cyan lines indicate that we cannot move $\psi(p)$ and $\psi(q)$ such that $\psi(p)$ is "right of" $\psi(q)$ (relative to their respective lower left cell corner).

We now formally prove that the given instance $(\mathrm{red}(P), \mathrm{red}(T), \chi)$ and the constructed instance $(\mathrm{red}(P'), \mathrm{red}(T'), \chi')$ are equivalent.

Correctness ("only if"). Suppose that $(\mathrm{red}(P), \mathrm{red}(T), \chi)$ is a yes-instance. We show that $(\mathrm{red}(P'), \mathrm{red}(T'), \chi')$ is a yes-instance, too.

We know there is a function $\phi \colon P \to T$ that preserves relative order and hits every color in T (i.e. the colors $[c]$) at least once. We define $\phi' \colon P' \to T'$ as follows. Let $i \in [t]$

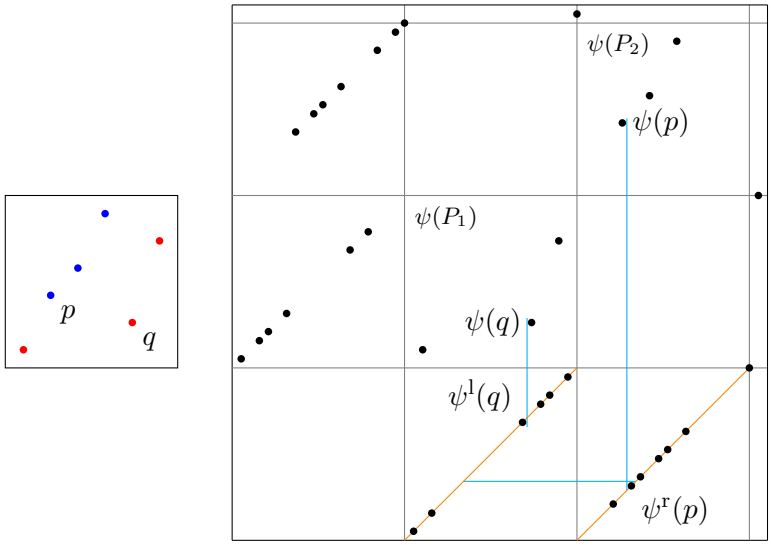

Figure 4.3: Example showing how P' (right) is constructed from P (left). The coloring of P indicates the partition into two 1-increasing subsets P_1 and P_2. The example is too small to use the reduction productively: P' is 5-increasing at best. The blue and orange lines indicate how relative position of two points in P is "preserved" in P'.

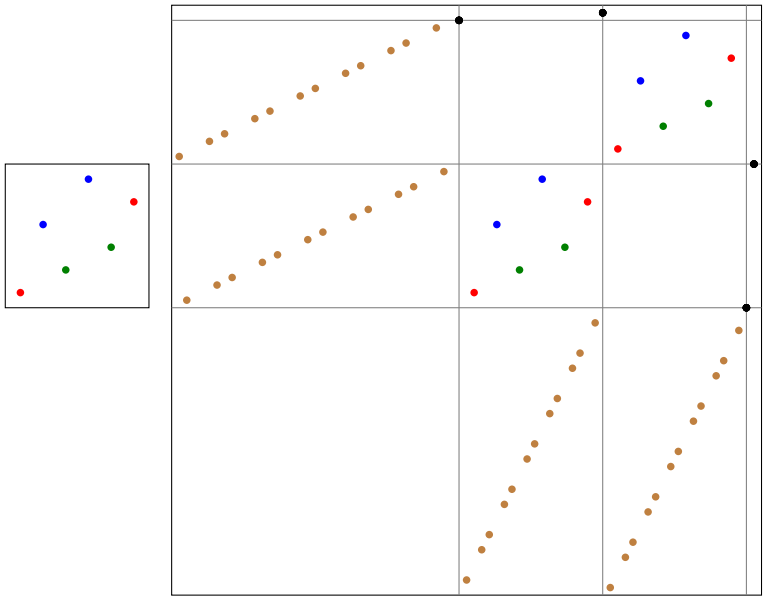

Figure 4.4: Example showing how T' (right) is constructed from T (left) if $t = 2$. The anchor points of T' are shown in black, although each of them has its own color in reality.

and $p \in P_i$. Let

$$\begin{aligned}
&\phi'(\psi(p)) = \omega_i(\phi(p)),\\
&\phi'(\psi^\alpha(p)) = \omega_i^\alpha(\phi(p)) \text{ for } \alpha \in \{\mathrm{d}, \mathrm{u}, \mathrm{l}, \mathrm{r}\},\\
&\phi(a_i^{\mathrm{h}}) = a_i^{\mathrm{H}} \text{ and}\\
&\phi(a_i^{\mathrm{v}}) = a_i^{\mathrm{V}}.
\end{aligned}$$

We have to show that ϕ' is (i) injective, (ii) preserves relative position and (iii) hits every color in T'. (i) follows directly from the definition of ϕ'. (iii) follows from the fact that ϕ hits every color in $[c]$. It remains to show (ii).

Claim 4.16. *ϕ' preserves relative position.*

Proof. Let $p, q \in P'$. First, suppose there is an i such that $p, q \in \psi(P_i)$ or $p, q \in \psi^\alpha(P_i)$ for some $\alpha \in \{\mathrm{d}, \mathrm{u}, \mathrm{l}, \mathrm{r}\}$. Then the definitions above imply that ϕ' preserves the relative position of p and q, as ϕ' moves p and q by the same offset. On the other hand, if there is no such i, then p and q are in different cells, and ϕ' will map them to corresponding cells in T', again preserving the relative position.

The remaining two cases concern the anchor points. If both p and q are anchor points, their relative position is clearly preserved. Suppose $p = a_i^{\mathrm{v}}$ and $q \in C_{j,k}^{\mathrm{P}}$. Then p is left of q, which is preserved by ϕ'. Moreover, the relative vertical position of p and q depends only on k, namely $p.y < q.y$ if and only if $i \leq k$. Now $\phi'(p).y a_i^{\mathrm{V}}.y < \phi'(q) \in C_{j,k}^{\mathrm{T}}$ if and only if $i \leq k$, so ϕ preserves the relative vertical position of p and q. The case that $p = a_i^{\mathrm{h}}$ can be proven similarly. □

Correctness ("if"). We now show if that $(\mathrm{red}(P'), \mathrm{red}(T'), \chi')$ is a yes-instance, then $(\mathrm{red}(P), \mathrm{red}(T), \chi)$ is a yes-instance, too.

Assume there is an order-preserving mapping $\phi' \colon P' \to T'$ that hits all $c + 2t - 1$ colors of T' at least once. We first prove some properties of ϕ'.

Claim 4.17. *$\phi'(a_i^{\mathrm{v}}) = a_i^{\mathrm{V}}$ and $\phi'(a_i^{\mathrm{h}}) = a_i^{\mathrm{H}}$ for each $i \in [t]$.*

Proof. Observe the points in $\bigcup_{i=1}^{t} \{a_i^{\mathrm{V}}, a_i^{\mathrm{H}}\}$ all have distinct colors and form the t rightmost (respectively topmost) points in T'. This means ϕ' must map to them the t rightmost (respectively topmost) points in P, which are indeed $\bigcup_{i=1}^{t} \{a_i^{\mathrm{v}}, a_i^{\mathrm{h}}\}$. The specific mappings we claimed follow from the definitions. □

Claim 4.18. *Let $p \in P'$. If $p \in C_{i,j}^{P}$, then $\phi'(p) \in C_{i,j}^{T}$.*

Proof. As mentioned in the informal description (see page 43), this is ensured by the anchor points. First, observe that by claim 4.17, ϕ' uses all the anchor points of T' to map the anchor points of P' to them. Let $p \in P' \cap C_{i,j}^{\mathrm{P}}$. As p is not an anchor point, p is mapped to a non-anchor point in T', so there are $i', j' \in \{0, 1, \ldots, t\}$ such that $p' := \phi'(p) \in C_{i',j'}^{\mathrm{T}}$. We show $i' = i$ and $j' = j$.

Suppose $i > 1$, then $a_i^{\mathrm{h}}.x < p.x$, thus $a_i^{\mathrm{H}}.x < p'.x$ by claim 4.17. This implies $i \leq i'$ (which is trivially true if $i = 1$). If $i < t$, then $p.x < a_{i+1}^{\mathrm{h}}.x$, thus $p'.x < a_{i+1}^{\mathrm{H}}.x$ by

claim 4.17. This now implies $i' \leq i$ (trivially true if $i = t$). By a similar argument, we can show $j' = j$. □

We are now ready to construct the mapping $\phi\colon P \to T$ that shows $(\mathrm{red}(P), \mathrm{red}(T), \chi)$ is a yes-instance. For each $p \in P_i$, let

$$\phi(p) = \left(\frac{\phi'(\psi(p)).x - f - im + 1}{3}, \frac{\phi'(\psi(p)).y - f - im + 1}{3} \right).$$

To provide an informal visualization, $\phi(P)$ corresponds to the points of T' in the cells $C^{\mathrm{T}}_{i,i}$ for $i \in [t]$, when we stack all these cells on top of each other and then scale the result. Note that it is not immediately clear that we do not stack two points on top of each other, i.e. that ϕ is injective.
We need to show that ϕ is well-defined (i.e. its image is a subset of T'), preserves relative order and hits each of the c colors of T. This implies that $(\mathrm{red}(P), \mathrm{red}(T), \chi)$ is a yes-instance, concluding the proof. We start with proving a stronger version of the first property, to be used in subsequent proves.

Claim 4.19. *Let $p \in P_i$. Then $\phi(p) \in T$ and $\phi'(\psi(p)) = \omega_i(\phi(p))$.*

Proof. By definition, we have $\psi(p) \in C^{\mathrm{P}}_{i,i}$. Claim 4.18 implies $\phi'(\psi(p)) \in C^{\mathrm{T}}_{i,i}$. As $C^{\mathrm{T}}_{i,i}$ only contains points in the image of ω_i, we have $\phi'(\psi(p)) = \omega_i(r)$ for some $r \in T$. It remains to show that $\phi(p) = r$. Indeed, by definition of ϕ, we have

$$\phi(p) = \left(\frac{\omega_i(r).x - f - im + 1}{3}, \frac{\omega_i(r).y - f - im + 1}{3} \right) = r.$$

□

Now we show that ϕ preserves relative order, which also implies that ϕ is injective, so we don't stack two points on top of each other in our informal visualization.

Claim 4.20. *ϕ preserves relative order.*

Proof. Let $p \in P_i$ and $q \in P_j$. By claim 4.19, there are $r, s \in T$ such that $\phi(p) = r$, $\phi'(\psi(p)) = \omega_i(r)$, $\phi(q) = s$ and $\phi'(\psi(q)) = \omega_j(s)$.

We need to prove that $p.x < q.x$ implies $\phi(p).x < \phi(p).x$ (ϕ preserves horizontal position) and $p.y < q.y$ implies $\phi(p).y < \phi(p).y$ (ϕ preserves vertical position).

Suppose $p.x < q.x$. First observe that

$$\begin{aligned} \psi^{\mathrm{r}}(p) \in C^{\mathrm{P}}_{i,0} &\implies \phi'(\psi^{\mathrm{r}}(p)) \in C^{\mathrm{T}}_{i,0}, \text{ and} \\ \psi^{\mathrm{l}}(q) \in C^{\mathrm{P}}_{j,0} &\implies \phi'(\psi^{\mathrm{l}}(q)) \in C^{\mathrm{T}}_{j,0}, \end{aligned}$$

implying $\phi'(\psi^{\mathrm{r}}(p)) = (f + im + a, a)$ and $\phi'(\psi^{\mathrm{l}}(q)) = (f + jm + b, b)$ for some $a, b \in [m]$. We further know that

$$\psi(p).x = il + 3p.x - 1 < il + 3p.x = \psi^{\mathrm{r}}(p).x \text{ and} \tag{1}$$

$$\psi^{\mathrm{r}}(p).y = 3p.x < 3q.x - 2 = \psi^{\mathrm{l}}(q).y, \text{ and} \tag{2}$$

$$\psi^{\mathrm{l}}(q).x = jl + 3q.x - 2 < jl + 3q.x - 1 = \psi(q).x. \tag{3}$$

As ϕ' preserves relative order and by the definition of r and s, this implies

$$(f + im + 3r.x - 1) = \omega_i(r).x < \phi'(\psi^{\mathrm{r}}(p)).x = f + im + a, \tag{1'}$$

$$a = \phi'(\psi^{\mathrm{r}}(p)).y < \phi'(\psi^{\mathrm{l}}(q).y).y = b, \text{ and} \tag{2'}$$

$$f + jm + b = \phi'(\psi^{\mathrm{l}}(q)).x < \omega_j(s).x = (f + jm + 3s.x - 1). \tag{3'}$$

We conclude that $3r.x - 1 < a < b < 3s.x - 1$ and thus thus $r.x < s.x$. It can be proven in essentially the same way that ϕ preserves relative vertical position. □

We now finish the correctness proof by showing the third property, that ϕ hits all colors of T.

Claim 4.21. $\chi(\phi(P)) = [c]$.

Proof. Let $i \in [c]$. We show that there is some $p \in P$ such that $\chi(\phi(p)) = i$.

As ϕ' hits all colors, we know $\chi'(\phi'(p')) = i$ for some $p' \in P'$. The coloring χ' only uses the colors $[c]$ on the set $\bigcup_{j=1}^{t} \omega_j(T) \subseteq \bigcup_{j=1}^{t} C^{\mathrm{T}}_{j,j}$, so $\phi'(p') \in C^{\mathrm{T}}_{j,j}$ for some $j \in [t]$, implying $p' \in C^{\mathrm{P}}_{j,j}$ by claim 4.18 and thus $p' = \psi(p)$ for some $p \in P_j$. By claim 4.19 $\phi'(\psi(p)) = \omega_j(\phi(p))$. Now by definition,

$$\chi(\phi(p)) = \chi'(\omega_j(\phi(p))) = \chi'(\phi'(\psi(p))) = \chi'(\phi'(p')) = i,$$

thus ϕ hits the color i. □

4.2.2 Secondary reduction step

We only sketch the proof of lemma 4.15 to avoid repetition.

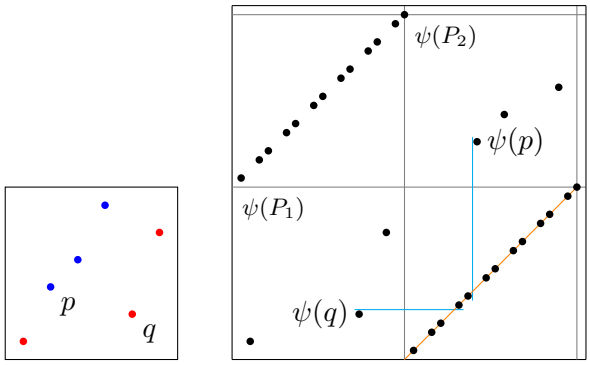

Figure 4.5: Example showing the construction of the pattern in the secondary reduction step. The coloring of P indicates the partition into two 1-increasing subsets P_1 and P_2.

Lemma 4.15. *An instance (π, τ, ϕ) of SCPPM where π is s-increasing can be reduced in polynomial time to another instance (π', τ', ϕ') where π' is $(\lceil s/2 \rceil + 2)$-increasing, $|\pi'| \leq 5|\pi|$ and $|\tau'| \leq 6|\tau|$.*

Proof sketch. We split P into *two* parts that are both $\lceil s/2 \rceil$-increasing and arrange them as we would in lemma 4.14, just that we invert the top right part and arrange the two gadgets as in fig. 4.5. We copy T once and arrange the two copies accordingly, again inverting the top right copy. Figure 4.5 shows how the vertical order of two points that we map into the two copies of T is preserved. We roughly want to prevent that p is moved too far down and that q is moved too far up. As P_2 is inverted in the new pattern, this means preventing p from moving too far *left*. Observe that the construction prevents this. □

4.3 $\mathcal{C}$-Pattern #SCPPM to $\mathcal{C}$-Pattern #PPM

We repeat the argument from Berendsohn et al. [3] here, and observe that it can be adapted to the restricted case that $\pi \in \mathcal{C}$.

Lemma 4.5 (Berendsohn et al. [3])**.** *Let there be an algorithm that solves* $\mathcal{C}$-Pattern #PPM *in time* $f(k)n^{\mathcal{O}(g(k))}$ *for some functions* f *and* g*. Then* $\mathcal{C}$-Pattern #SCPPM *can be solved in time* $h(k)n^{\mathcal{O}(g(k))}$ *for some function* h*.*

Proof. Let $\mathbb{A}$ be such an algorithm for $\mathcal{C}$-Pattern #PPM, and let π, τ, χ be an input of $\mathcal{C}$-Pattern #SCPPM, with $k = |\pi|$, $n = |\tau|$ and $\chi\colon [n] \to [t]$. We can assume that χ is surjective (otherwise, we can easily decrease t and modify it to be surjective). Further, if $t > k$, we cannot hit every color, so we simply reject. From now on, we assume $t \leq k$. For an index set $I \subseteq [t]$, we obtain the permutation τ_I by taking all elements from τ with a color in I and reducing the resulting sequence (alternatively, we delete all elements from τ with a color *not* in I and reduce the result). We make 2^t calls to $\mathbb{A}$, each with a different τ_I (but the same $\pi \in \mathcal{C}$). Let C_I be the result of $\mathbb{A}$ when called with input (π, τ_I). Recall that we want to compute the number C of occurrences of π in τ that hit *all* colors, and note that C_I is exactly the number of occurrence of π in τ that hit *only* the colors in I (but may not hit all of them). Now we simply use the inclusion-exclusion formula:

$$C = \sum_{I \subseteq [t]} (-1)^{k-|I|} C_I.$$

The running time is $2^k \cdot f(k)n^{\mathcal{O}(g(k))} = h(k)n^{\mathcal{O}(g(k))}$ with $h(k) = 2^k f(k)$. □

5 On-Line Permutation Pattern Matching

In this chapter, we investigate the *space complexity* of PPM in the on-line case. We will consider two variants of the problem. In π-On-Line-PPM we have to decide $\pi \preceq \tau$ when first we are given $n = |\tau|$ and then τ, one number at the time. In π-On-Line-Sequence-PPM, instead of τ, we are given a *sequence* ω of at most n distinct integers in $[n]$, terminated by a special *end-of-input* symbol, and have to decide $\pi \preceq \omega$. We write π-OLPPM and π-OLSPPM for short. We will only consider the deterministic case here. Jelínek et al. [20] have studied the nondeterministic variant of π-OLSPPM.

Our machine model in this chapter is the the unit-cost-RAM (although we can assume a word size of $\mathcal{O}(\log n)$ for our algorithms). The space an algorithm requires is measured in bits. Note that when talking about the problems π-OL(S)PPM, the number of bits in the input is $\Theta(n \log n)$ rather than just n. Also note that in the *offline* case, it makes no difference to the asymptotic space requirement whether the input text is given as a sequence or permutation, as reducing a sequence can be done in $\mathcal{O}(n)$ time and $\mathcal{O}(n \log n)$ bits of space using a table.

Trivially, if an algorithm solves π-OLSPPM, it also can be adapted to solve π-OLPPM with $\mathcal{O}(\log n)$ additional space and $\mathcal{O}(n)$ additional time (by using a counter to find the end of the input). Moreover, observe that π-OL(S)PPM and π^{C}-OL(S)PPM have the same complexity, as we can transform one problem to the other by just mapping each value v to $n + 1 - v$.

We first show that in the case that π is monotone, we only need $\mathcal{O}(\log n)$ space for both π-OLSPPM and π-OLPPM, using the well-known algorithm for the longest increasing subsequence.

Theorem 5.1. *Let π be monotone. Then π-OLSPPM can be solved in linear time and $\mathcal{O}(\log n)$ space.*

In all other cases, π-OLSPPM already needs linear space. Our lower bounds also hold if we allow a constant number of passes over the input (i.e. the whole input is given a constant number of times, sequentially).

Theorem 5.2. *Let π be non-monotone. Then π-OLSPPM cannot be solved using $o(n)$ bits of space, even if we allow a constant number of passes over the input.*

The two lemmas above are proven in section 5.2. For OLPPM, the situation is almost the same. The only exception are the non-monotone 3-permutations 132, 213, 231 and 312. If π is one of those, we can solve π-OLPPM using only $\mathcal{O}(\sqrt{n} \log n)$ bits of space, with the two algorithms presented in sections 5.3.1 and 5.3.3.

Theorem 5.3. *Let π be a non-monotone 3-permutation. Then π-OLPPM can be solved using $\mathcal{O}(\sqrt{n} \log n)$ bits of space.*

Finally, in section 5.3.4, we prove the mentioned lower bounds for OLPPM.

Theorem 5.4. *Let π be a non-monotone permutation of size at least 4. Then π-OLPPM cannot be solved using $o(n)$ bits of space, even if we allow a constant number of passes over the input.*

5.1 Communication complexity

To prove the lower bounds mentioned above, we will use tools from Communication Complexity [23, 25]. We briefly review this topic.

Let A and B be finite sets, and let $P \subseteq A \times B$. Suppose we have two players, Alice, who knows some $a \in A$, and Bob, who knows some $b \in B$. Alice and Bob each want to decide whether $(a, b) \in P$. However, as they both only hold part of the input, they usually have to exchange information. We are interested in the minimal amount of bits they have to send to each other to determine whether $(a, b) \in P$. On the other hand, we do not care how much time they need, and assume both may compute arbitrary functions. The minimal amount of bits Alice and Bob have to exchange in the worst case is called the *deterministic communication complexity* of P. For more details, we refer to Kushilevitz and Nisan [23].

We only consider one specific problem, namely the *set disjointness problem*. Let n be a positive integer known by both Alice and Bob and let $S, T \subseteq [n]$, where Alice knows S and Bob knows T. Alice and Bob have to decide whether $S \cap T \neq \emptyset$. For a fixed n, we call the set disjointness problem $\textsc{disj}_n$.

Lemma 5.5 (Kushilevitz and Nisan [23, Example 1.21])**.** *The deterministic communication complexity of* $\textsc{disj}_n$ *is* $n + 1$.

We will prove lower bounds for complexity of our on-line problems by reducing from $\textsc{disj}_n$. Let $m, n \in \mathbb{N}$ and let π be a permutation. Let the functions f_1, f_2, f_3, f_4 map subsets of $[n]$ to a sequences of integers in $[m]$ such that for all $S, T \subseteq [n]$, no integer appears more than once in the sequence $\omega = f_1(S)f_2(T)f_3(S)f_4(T)$, $|\omega| \leq m$, and $S \cap T \neq \emptyset \iff \pi \preceq \omega$. Then we say π-OLSPPM is m-$\textsc{disj}_n$*-hard*. If additionally, for all S, T, ω is a permutation (i.e., $|\omega| = m$), we say π-OLPPM is m-$\textsc{disj}_n$*-hard*.

If there is a $c \in \mathbb{N}$, such that π-OL(S)PPM is cn-$\textsc{disj}_n$-hard for each n, we say π-OL(S)PPM is DISJ-hard. The following lemma shows that this definition makes sense.

Lemma 5.6. *If π-OL(S)PPM is m-$\textsc{disj}_n$-hard, it is also m'-$\textsc{disj}_n$-hard for all $m' \geq m$*

Proof. The OLSPPM case is obvious. Assume π-OLPPM is m-$\textsc{disj}_n$-hard. We show that it is also $(m+1)$-$\textsc{disj}_n$-hard, the rest follows by induction. Without loss of generality, let $\pi(k-1) > \pi(k)$ with $k = |\pi|$. Let τ be an n-permutation, and let τ' by defined by appending $(n+1)$ to τ. If $\pi \preceq \tau$, then $\pi \preceq \tau'$. On the other hand, if $\pi \preceq \tau'$ via the index mapping $\phi\colon [k] \to [n+1]$, then $\tau(\phi(k)) < \tau(\phi(k-1)) \leq n$, so $\phi(k) < n+1$ and thus $\pi \preceq \tau$. This means we can simply append $(n+1)$ to the result of f_4 to show that π-OLPPM is $(m+1)$-$\textsc{disj}_n$-hard. □

Our hardness results will all hold even if we allow our algorithm a constant number of *passes* over the input. We say an on-line algorithm solves π-OL(S)PPM in k passes if it can determine $\pi \preceq \tau$ respectively $\pi \preceq \omega$ provided that the whole input is not just given once, but repeated k times.

Lemma 5.7. *Let π-OL(S)PPM be* DISJ*-hard. Then every on-line algorithm deciding π-OL(S)PPM in a constant number of passes needs $\Omega(m)$ space, where m is the length of the input sequence.*

Proof. Let $c \in \mathbb{N}$ such that π-OLSPPM is cn-DISJ_n-hard for each n. Suppose $\mathbb{A}$ is an algorithm that solves π-OL(S)PPM in $k \in \mathbb{N}_+$ passes, using $o(m)$ bits of space. Then there is some n such that $\mathbb{A}$ needs at most $n/2k$ bits of space on inputs of length at most cn. We will use $\mathbb{A}$ to solve DISJ_n using only n bits of communication, which contradicts lemma 5.5.

Let $S, T \subseteq [n]$ be an instance of DISJ_n. First, Alice computes $f_1(S)$ and $f_3(S)$, and Bob (independently) computes $f_2(T)$ and $f_4(T)$. Alice starts $\mathbb{A}$ and passes the number cn and the sequence $f_1(S)$ to it. When $\mathbb{A}$ has processed $f_1(S)$ completely and needs new input, Alice sends the intermediate state of $\mathbb{A}$ to Bob. Bob continues execution by passing $f_2(T)$, then sends the resulting state back to Alice. Alice then passes $f_3(S)$, sends the state to Bob, who passes $f_4(T)$ and then the *end-of-input* symbol (in the OLSPPM case) and sends the state back to Alive. Alice and Bob repeat this procedure $k-1$ additional times. After Bob sends the state for the last time, $\mathbb{A}$ has done k passes on the input sequence $f_1(S)f_2(T)f_3(S)f_4(T)$ (which is a valid input for π-OL(S)PPM), so it accepts or rejects. Both Alice and Bob now know whether S and T are disjoint.

The length of $f_1(S)f_2(T)f_3(S)f_4(T)$ is at most cn. As such, by assumption, $\mathbb{A}$ uses at most $M = n/2k$ space. We send $2k$ messages that contain intermediate states of the algorithm, so the total communication required is $2k \cdot M = n$. But, by lemma 5.5, the deterministic communication complexity of DISJ_n is $n+1$, a contradiction. □

Observe that if π-OL(S)PPM is DISJ-hard, then we can easily show that π^{R}-OL(S)PPM is also DISJ-hard by exchanging the functions f_1 with f_4 and f_2 with f_3 and reversing their outputs. Thus, as the complexity of π-OL(S)PPM and π^{C}-OL(S)PPM is the same, if π-OL(S)PPM is DISJ-hard then π'-OL(S)PPM is also DISJ-hard for every symmetry π' of π.

5.2 Complexity of π-OLSPPM

We first give the upper bound for finding monotone permutations. We can solve Id_k-OLSPPM using the well-known algorithm for the longest increasing subsequence problem due to Knuth and described by Fredman [15]. The original algorithm requires $\Theta(n \log n)$ space (where n is the length of the input sequence), but if we modify it to accept as soon as we found an increasing subsequence of length k, we only need $\mathcal{O}(\log n)$ space if k is a constant. The algorithm requires $\Theta(kn) = \Theta(n)$ time in the worst case. As noted in Section 5.1, the complexity of Id_k^{R}-OLSPPM is the same as the complexity of Id_k-OLSPPM. Theorem 5.1 follows:

Theorem 5.1. *Let π be monotone. Then π-OLSPPM can be solved in linear time and $\mathcal{O}(\log n)$ space.*

We now turn to non-monotone permutations.

Theorem 5.2. *Let π be non-monotone. Then π-OLSPPM cannot be solved using $o(n)$ bits of space, even if we allow a constant number of passes over the input.*

Proof. We show DISJ-hardness. Let $S, T \subseteq [n]$ be an instance of DISJ_n. We will only use f_1 and f_2 and set $f_3(S) = f_4(T) = \varepsilon$. First consider the k-permutations π such that $\pi(k) \notin \{1, k\}$. We will later reduce all other cases to this case.

Let $\ell = \pi^{-1}(1) \in [k-1]$ and $h = \pi^{-1}(k) \in [k-1]$. As π-OLSPPM has the same complexity as π^{C}-OLSPPM, we can assume $h < \ell$. For $i \in [n]$, we define ω_i to be an elevated copy of π without the last element if $i \in S$, and to be empty otherwise:

$$\omega_i = \begin{cases} ((i-1)k + \pi(1), (i-1)k + \pi(2), \ldots, (i-1)k + \pi(k-1)), & \text{if } i \in S, \text{ and} \\ \varepsilon, & \text{otherwise.} \end{cases}$$

Now let $f_1(S)$ be the concatenation of all ω_i with $i \in S$, in order of i. Let $f_2(T)$ be the sequence of values $(i-1)k + \pi(k-1)$ for $i \in [n]$ with $i \in T$. If $\pi(k-1) < \pi(k)$, we sort $f_2(T)$ in descending order, otherwise, in ascending order. We call the indices of ω_i *Alice's i-th block*, and we call $((i-1)k + \pi(k))$ *Bob's i-th element* and its index *Bob's i-th index*. Let $f_3(S) = f_4(T) = \varepsilon$ and let $\omega = f_1(S)f_2(T)$.
We now show that $\pi \preceq \omega$ if and only if S and T are not disjoint. Assume that $i \in S \cap T$. Then Alice's i-th block is nonempty and together with Bob's i-th element forms an occurrence of π.

Now assume $\pi \preceq \omega$ via the index mapping $\phi\colon [k] \to [n]$. The way we defined the order of Bob's elements, we cannot use more than one of Bob's indices in ϕ. This means that $i \in [k-1]$ has to be mapped into Alice's blocks. Assume h and ℓ are mapped into two different blocks. As $\pi(\ell) = 1 < k = \pi(h)$, and due to the ascending arrangements of Alice's blocks, we know $\phi(\ell) < \phi(h)$, which contradicts $h < \ell$. As h and ℓ are the indices of the highest and lowest value in π, all indices $i \in [k-1]$ must be mapped into one of Alice's blocks, say the j-th. As that block contains only k elements, the remaining index k must be mapped to Bob's j-th index. This implies $j \in S \cap T$.

Observe that the above proof also covers the case that $\pi(1) \notin \{1, k\}$, as DISJ-hardness of π-OLSPPM implies DISJ-hardness of π^{R}-OLSPPM.

Now consider a non-monotone k-permutation π such that $\{\pi(1), \pi(k)\} = \{1, k\}$. By symmetry, we can assume that $\pi(1) = 1$ and $\pi(k) = k$. Let ℓ be the maximal index such that $\pi(\ell) \neq \ell$ (if ℓ did not exist, π would be monotone). Let $\rho = \text{red}(\pi(1), \pi(2), \ldots, \pi(\ell))$. We now reduce ρ-OLSPPM to π-OLSPPM, the former of which requires $\Omega(n)$ space as shown in the first part of the proof.

As ℓ is maximal, we know that $\pi(\ell + i) = \ell + i$ for all $i \in [k - \ell]$. Let ω be a sequence of $m \leq n$ distinct integers in $[n]$, and let ω' be ω concatenated with the sequence $|\omega| + 1, |\omega| + 2, \ldots, |\omega| + (k - \ell)$. If $\rho \preceq \omega$, then $\pi \preceq \omega'$. If $\pi \preceq \omega'$ via the index mapping $\phi\colon [k] \to [m+k-\ell]$, then ϕ being strictly increasing implies $\phi([\ell]) = \phi([k-(k-\ell)]) \subseteq [m]$, thus $\rho \preceq \omega$. This concludes the reduction from ρ-OLSPPM to π-OLSPPM and shows that π cannot be solved using $o(n)$ space. □

5.3 Complexity of π-OLPPM

We turn to upper bounds on π-OLPPM. We will prove

Theorem 5.3. *Let π be a non-monotone 3-permutation. Then π-OLPPM can be solved using $\mathcal{O}(\sqrt{n}\log n)$ bits of space.*

There are 4 such permutations π, but there are two pairs of permutations that are complements of each other, so we really only have to consider two cases. We give an algorithm using $\mathcal{O}(\sqrt{n}\log n)$ bits of space for each of these cases. The two algorithms are similar in that they both partition the input into vertical strips of size roughly $\sqrt{n}$, and always store one complete strip in memory, along with data of logarithmic size for each strip already read.

5.3.1 An algorithm for 312-OLPPM

Lemma 5.8. *312-OLPPM can be solved in linear time using $\mathcal{O}(\sqrt{n}\log n)$ bits of space.*

We start with a simpler version of the algorithm that has running time $\Theta(n^2)$, and later show how it can be modified to have linear running time.

Let τ be the input permutation, and $n = |\tau|$. In the following, we assume we are given the respective points from S_τ (ordered by abscissa/index) instead of elements from $[n]$. We can easily simulate this setting by keeping a counter of the number of elements read so far. When τ does contain 312, we will not only accept but also return an occurrence of 312.

Algorithm. We partition the input into at most $\lceil\sqrt{n}\rceil$ strips of size at most $\lfloor\sqrt{n}\rfloor$. We always store all points in the current strip at once, as well as some (constant-size) data about each of the previous strips. The algorithm has three parts that run in parallel.

(1) Whenever we finished reading one strip, we do a brute-force search for an occurrence of 312 in that strip. If we find one, we return it.

(2) Now we want to detect occurrences of 312 where the first element is in one strip and the last two elements are in subsequent strips. Whenever we finish reading a strip S_i, we store the highest point $h_i \in S_i$. We also set $\ell_i \leftarrow h_i$. Now, whenever we read a point p in a later strip, if $p.y < \ell_i.y$, we set $\ell_i \leftarrow p$. If $\ell_i.y < p.y < h_i.y$, we return (h_i, ℓ_i, p).

Note that whenever we read a point $p \in S_j$, we check it against all ℓ_i, h_i for $i < j$. One could also think of this as "forking" a process P_i for each strip S_i, where P_i stores ℓ_i and h_i and processes each subsequent value.

(3) Lastly, we need to detect occurrences of 312 where the first two elements are in a single strip, and the third is in a later strip. After reading strip S_i, we consider all decreasing pairs $(p, q) \in S_i^2$, i.e. those where $p.x < q.x$ and $q.y < p.y$. We compute the highest value y such that $q.y < y < p.y$ and no element in S_i has value y. If y exists and $h_j.y < y$ for all $j < i$, we know that there must be some $k > i$ and some

point $r \in S_k$ with $r.y = y$. From now on, we ignore the rest of the algorithm and just read points from the input until we arrive at r. Then, we return (p, q, r). On the other hand, if y does not exist or $h_j.y < y$ for some $j < i$, we ignore (p, q).

When we arrive at the end of the input and we have not returned yet, we reject.

Required space. We store one strip at the time, as well as h_i, ℓ_i for each strip S_i. This amounts to $\mathcal{O}(\sqrt{n})$ space in total.

Correctness. The brute-force search (1) rejects or yields a correct occurrence of 312. For (2), suppose we read some $p \in S_j$ with $\ell_i.y < p.y < h_i.y$. We know that $h_i.y < \ell_i.y < p.y$, thus returning (h_i, ℓ_i, p) is correct.

For (3), suppose we have some $p, q \in S_i$ and $y \in [n]$ with $p.x < q.x$ and $q.y < y < p.y$ such that no point in S_i has value y and $h_j.y < y$ for all $j < i$, implying that no point in $S_1 \cup S_2 \cup \cdots \cup S_{i-1}$ has value y. Then some point $r \in S_j$ for some $j > i$ must have value y, and (p, q, r) form an occurrence of 312 in S_τ.

Completeness. Let $p \in S_i$, $q \in S_j$, $r \in S_k$ for $i \leq j \leq k$ be an occurrence of 312 in S_τ. If $i = j = k$, we find it by the brute-force search (1).

Assume $i < j \leq k$. By the time we read r, we know $h_i.x < \ell_i.x < r.x$ as well as $\ell_i.y \leq q.y < r.y < p.y \leq h_i.y$. Thus, we return the occurrence (h_i, ℓ_i, r) in (2), which may or may not be identical to (p, q, r).

Finally, assume $i = j < k$. Then p, q are considered as a pair in the third part of the algorithm. Let y be maximal such that $q.y < y < p.y$ and no point in S_i has value y. As $r \notin S_i$, y must exist and we know $r.y \leq y$. If $h_m.y < y$ for all $m < i$, the algorithm accepts (eventually) in (3). Assume $y \leq h_m.y$ for some $m < i$. Then (h_m, q, r) form an occurrence of 312 in S_τ, which is found by (2).

5.3.2 A linear-time algorithm for 312-OLPPM

We now show how to reduce the running time of the above algorithm to $\mathcal{O}(n)$ without raising the asymptotic space requirement. We will use an algorithm $\mathbb{A}$ as a subroutine. We first describe what exactly $\mathbb{A}$ computes and how we use it to solve 312-OLPPM, and later state $\mathbb{A}$ in pseudocode and prove its correctness and completeness.

Lemma 5.9. *There is an algorithm $\mathbb{A}$ that receives a point set $S \in [n] \times [n]$ of size m in general position, ordered by abscissa (index), and a value $k \in [n]$, and behaves as follows:*

(i) It finds an occurrence of 213, if $213 \preceq S$; or

(ii) otherwise finds an increasing pair $(p, q) \in S^2$ and a value $y \in [n]$ with $k < y$ and $p.y < y < q.y$ such that $r.y \neq y$ for all $r \in S$, if such a pair exists; or

(iii) otherwise rejects.

$\mathbb{A}$ needs $\mathcal{O}(m)$ time and $\mathcal{O}(m \log n)$ space.

$\mathbb{A}$ finds 213 instead of 312, but if we flip the input set S horizontally before applying $\mathbb{A}$, we obtain the algorithm $\mathbb{A}'$, which behaves as follows:

(i) It finds an occurrence of 312, if $312 \preceq S$; or

(ii) finds a decreasing pair $(p, q) \in S^2$ and a value $y \in [m]$ with $k < y$ and $q.y < y < p.y$ such that $r.y \neq y$ for all $r \in S$, if such a pair exists; or

(iii) otherwise rejects.

We now show how to use $\mathbb{A}'$ to improve the running time of the algorithm in section 5.3.1 to $\mathcal{O}(n)$.

Modification of the Algorithm. We first improve part (2) of the algorithm (without using $\mathbb{A}'$). We will maintain a linked list L of all relevant pairs (ℓ_i, h_i), sorted by $h_i.y$ in descending order. Suppose we are in strip S_k, and we read a point p. We find the *last* pair (ℓ_i, h_i) in L such that still $p.y < h_i$. If no such pair exists, we do nothing. If $\ell_i.y < p.y < h_i.y$, we return (h_i, ℓ_i, p). Otherwise, $p.y < \ell_i$. Let (ℓ_j, h_j) be the first entry in L. We set $\ell_j \leftarrow p.y$. Then, if $i \neq j$, we remove all entries from L that lie between (ℓ_j, h_j) and (ℓ_i, h_i), excluding (ℓ_j, h_j), but including (ℓ_i, h_i). Observe that the non-optimized version of the algorithm would have changed exactly ℓ_j *and* all the removed entries. After reading S_k completely, we find the highest point h_k and insert (h_k, h_k) into L at the proper position.

In each step where we look at m entries, we remove $m - 1$ entries. We insert $\mathcal{O}(\sqrt{n})$ entries into the list in total, one for each strip. The operations after reading a strip cost $\mathcal{O}(\sqrt{n})$ time. Thus, the total running time of this part is linear.

For the parts (1) and (3) of the algorithm for 312-OLPPM, we use $\mathbb{A}'$ with parameters S_k and $k = h_j.y$ where (ℓ_j, h_j) is the first entry of L, i.e. h_j is the highest point to the left of S_j. As S_k contains $\mathcal{O}(\sqrt{n})$ points, the running time of $\mathbb{A}'$ is $\mathcal{O}(\sqrt{n})$. We apply part (1) and (3) to every strip, so the total running time of these two parts is $\mathcal{O}(n)$.

Correctness and completeness. We call the modified parts (1'), (2') and (3'). By the correctness and completeness of $\mathbb{A}'$, (1') and (3') behave the same as (1) and (3). It is also evident that (2') only accepts if (2) would have accepted. It remains to show *completeness* of (2'), i.e., if (2) accepts, then (2') accepts, too.

Suppose that at some point in strip S_j, we read a point p that makes (2) accept. This means $\ell_i.y < p.y < h_i.y$ for some $i < j$. If (ℓ_i, h_i) is in L, then (2') accepts. Suppose this is not the case, i.e. (ℓ_i, h_i) is not in L. Then there must be some (ℓ_j, h_j) with $\ell_j.y < \ell_i.y < h_i.y < h_j.y$ that is still in L or was removed at a later time than (ℓ_i, h_i). By induction, we will find some occurrence of 312 with p as the third point.

We now describe and analyze $\mathbb{A}$. Observe that as the input set S is ordered by abscissa, and because we only care about relative position, we may assume that the i-th point in S has abscissa exactly i, i.e. $S \in [m \times n]$.

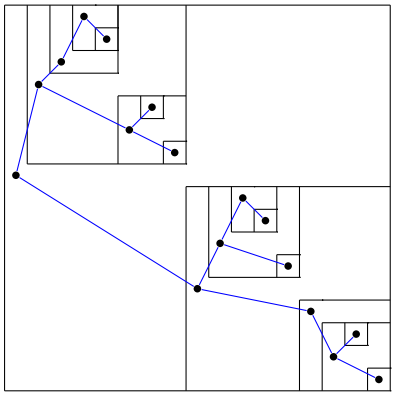

Figure 5.1: The tree-like structure of a 213-avoiding permutations. A point p divides the points right to it into two "boxes" (possibly empty) whose leftmost points can be seen as the children of p in a tree.

Algorithm. $\mathbb{A}$ is a recursive streaming algorithm in the following sense. We maintain a global variable i, initialized as $i \leftarrow 1$, that contains the abscissa of the point we use next (which, by assumption, is also the position in the ordered set S). The recursive calls *consume* the points p_i in order.

In algorithm 5.1 we present the two procedures used in the algorithm. CONSUME returns the point p_i and increments i if there are points left and the ordinate of p_i is in the given interval. FIND-213 is the main recursive procedure. It exploits the tree-like structure of 213-avoiding permutations, sketched in Figure fig. 5.1. The parameters ℓ and h specify the lower and upper bound on the ordinate of points it can consume.

In addition to i, we have another global variable T that may hold a triple of two points and a value, initialized with $\perp$. The algorithm first calls FIND-213$(1, n)$. After that, if we have not accepted in FIND-213 yet, we discard the result of the call and examine T. If $T \neq \perp$, we return $T = (p, q, y)$ as a solution for (ii). Otherwise, we reject.

Running time. Each recursive call requires constant time. As each recursive call that makes recursive calls itself consumes a value (without putting it back), there are at most $2n - 1$ recursive calls in total. Thus, the running time is linear.

Space requirements. Each recursive call needs $\mathcal{O}(\log n)$ space, and as there are $\mathcal{O}(m)$ recursive calls, the space requirement is $\mathcal{O}(m \log n)$.

Correctness and Completeness for (i). In the following, we will write p_C, r_C, h_C, ℓ_C when referring to the respective variables of a call C of FIND-213. We start with proving that $\mathbb{A}$ indeed determines whether $213 \preceq S$. We first need a technical property of FIND-213.

Claim 5.10. *Let C be a call of* FIND-213 *and let i' be the value of i when starting C. Then C and its sub-calls either accept at some point or precisely consume the points*

Algorithm 5.1 The procedures CONSUME and FIND-213

```
1:  function CONSUME(ℓ, h)
2:      if i = m + 1 or p_i.y ∉ [ℓ, h] then
3:          return ⊥
4:      else
5:          i ← i + 1
6:          return p_i
7:  function FIND-213(ℓ, h)
8:      if ℓ > h then
9:          return ⊥
10:     p ← CONSUME(ℓ, h)
11:     if p = ⊥ then
12:         return "gap"
13:     R_1 ← FIND-213(p.y + 1, h)
14:     R_2 ← FIND-213(ℓ, p.y − 1)
15:     r ← CONSUME(ℓ, h)
16:     if r ≠ ⊥ then
17:         Output "(p, p_{i−2}, r) is an occurrence of 213 in S"
18:         accept
19:     else if R_1 is "gap at y below q" then
20:         T ← (p, q, y)
21:     else if R_2 is "gap" then
22:         return "gap at (p.y − 1) below p"
23:     else if R_2 is "gap at y below q" then
24:         return "gap at y below p"
25:     else if R_1 is "gap" then
26:         return "gap"'
27:     else
28:         return ⊥
```

$p'_i, p_{i'+1}, \ldots, p_j$ *where* $p_k.y \in [\ell_C, h_C]$ *for* $k \in [i, j]$ *and* j *is maximal.*

Proof. We do induction on $h_C - \ell_C$. If $h_C - \ell_C < 0$, the claim is trivially true. Assume $\ell_C \leq h_C$ and that C and its sub-calls do not accept. By induction hypothesis, the calls in lines 13 and 14 consume only points with ordinate in $[\ell_C, h_C]$. As C does not accept, we know $r_C = \perp$, so $p_{j+1}.y \notin [\ell_C, h_C]$, where p_j is the last point consumed by C. □

We now prove correctness of $\mathbb{A}$ with respect to (i). Suppose $\mathbb{A}$ returns an occurrence of 213 in S. This means that there is a call C of FIND-213 that reaches line 18. We know that $r_C \neq \perp$, so $r_C.y \in [\ell_C, h_C]$. We further know that $r_C.y > p_C.y$, as otherwise the second call (line 14) would have consumed $r_C.y$ by claim 5.10. Moreover, that call must have consumed at least one point, otherwise the first call (line 13) would have consumed r_C. Thus, if i' is the value of i directly after line 15, then $q = p_{i'-2}$ was consumed by the second call which implies $q.y < p_C.y$. This proves that (p_C, q, r_C) is an occurrence of 213 in S.

We proceed with proving completeness of $\mathbb{A}$ with respect to (i). Suppose (p, q, r) is an occurrence of 213 in S. Fix r such that $r.x$ is minimal.

Claim 5.11. *We can assume* $q.x = r.x - 1$.

Proof. Suppose $q.x < r.x - 1$ and let $s \in S$ with $s.x = r.x - 1$. If $s.y < p.y$, then (p, s, r) is an occurrence of 213 in S, so we can set $q \leftarrow s$. If $p.y < s.y$, then p, q, s is an occurrence of 213 in S, violating minimality or $r.x$. □

Assume $q.x = r.x - 1$. We already proved that the algorithm is correct, i.e. it cannot accept before finding an occurrence of 213. By minimality of $r.x$, the algorithm must eventually consume q and r. Let C with parameters ℓ_C, h_C be the call such that $h_C - \ell_C$ is minimal with $\ell_C < q.y < p.y < r.y < h_C$. Minimality implies $q.y < p_C.y < r.y$.

Let i' be the value of i after the second recursive call (line 14) in C. We finish the proof by showing $i' = r.x$, which implies $r_C = p_{i'} = r$, so C will then return (p, q, r) as an occurrence of 213.

Claim 5.12. $i' = r.x$.

Proof. We prove by contradiction. First, suppose $r.x < i'$. Then the two recursive calls C_1 and C_2 that C makes must consume both q and r. As $q.y < p_C.y < r.y$, the first call C_1 must consume r and the second call C_2 must consume q. But $q.x < r.x$, so q is consumed before r, a contradiction.

Second, suppose $i' \leq q.x = r.x - 1$. We know $p_{i'}.y \notin [\ell_C, p.y]$ by claim 5.10, but $q.y \in [\ell_C, p.y]$, so in particular $i' < q.x$. We distinguish three possible cases, and prove that all three violate minimality of $r.x$.

- If $p_{i'}.y < \ell_C$, then $(p_C, p_{i'}, q)$ is an occurrence of 213.
- If $h_C < p_{i'}.y$, then $h_C < m$, so there must be some call C' such that $p_{C'}.x < p_C.x$ and $p_{C'}.y = h_C + 1$. Then $(p_{C'}, p_C, p_{i'})$ form an occurrence of 213.
- Finally, if $p_C.y < p_{i'}.y \leq h_C$, then the algorithm will return $(p_C, p_{i'-1}, p_{i'})$, which must be an occurrence of 213 by the correctness proof above.

□

Correctness and Completeness for (ii). In the following, we assume that $213 \not\preceq S$. As part (i) is correct, this means the algorithm cannot accept until the end. We now prove that the algorithm indeed satisfies (ii), i.e., given k, it finds a decreasing pair $(p, q) \in S^2$ and a value $y \in [n]$ with $k < y$ and $q.y < y < p.y$ such that $r.y \neq y$ for all $r \in S$ (if such a pair exists). We first prove a characterization of the possible return values of $\mathbb{A}$.

Claim 5.13. *Let C be a call of* Find-213*, let S_C be the set of points consumed by C and its sub-calls and let $Y_C = \{p.y \mid p \in S_C\}$. Assume C and its subcalls do not set T in line 20. Then*

(a) C returns "gap" if and only if $\ell_C \leq h_C$ and $Y_C = [\ell_C, y]$ for some $y \in [\ell_C - 1, h_C - 1]$ (say $[a, b] = \emptyset$ if $b < a$);

(b) if C returns "gap at y below q", then $q \in S_C$ and $y \in [\ell_C, q.y - 1] \setminus Y_C$; and

(c) if $Y_C \neq [\ell_C, y']$ for all $y' \in [\ell_C - 1, h_C]$, then C returns "gap at y below q" for some $q \in S_C$ and some $y \in [\ell_C, q.y - 1] \setminus Y_C$.

Proof. We prove the claim by structural induction on the recursion tree. Assume C makes no recursive calls. Then $Y_C = \emptyset$ and C returns "gap". Now assume C makes the two sub-calls C_1 and C_2. Let S_i be the set of points consumed by C_i and its sub-calls and let $Y_i = \{q.y \mid q \in S_i\}$ for $i \in \{1, 2\}$. For convenience, we say *a gap message* when referring to an unspecified message of the form "gap" or "gap at y below q". First, we show (b) and the "only if" direction of (a).

- Suppose C returns "gap" in line 12. Then $p_C = \perp$ and thus $Y_C = \emptyset$.
- Suppose C returns "gap at $(p_C.y - 1)$ below p_C in line 22. Then C_2 returned "gap" and thus $Y_2 \subseteq [\ell_{C_2}, h_{C_2} - 1] = [\ell_C, p_C.y - 2]$, so $(p_C.y - 1) \notin Y_C$.
- Suppose C returns "gap at y below p_C" in line 24. Then C' returned "gap at y below q" for some y, so $q \in S_2 \subseteq S_C$ and $y \in [\ell_{C_2}, q.y - 1] \setminus Y_1$, which together implies $y \in [\ell_C, p_C.y - 1] \setminus Y_C$.
- Suppose C returns "gap" in line 26. Then C_1 returned "gap", so $\ell_{C_1} \leq h_{C_1}$, implying $p_C.y < h_C$, and $Y_1 = [p_C.y + 1, y]$ for some $y \in [p_C.y, h_C - 1]$. As we have not returned yet, we know in particular that C_2 has not returned a gap message, so $Y_2 = [\ell_C, p_C.y - 1]$. Now $Y_C = Y_1 \cup \{p_C.y\} \cup Y_2 = [\ell_C, y]$.

We now show the "if" direction of (a). Suppose $\ell_C \leq h_C$ and $Y_C = [\ell_C, y]$ for some $y \in [\ell_C - 1, h_C - 1]$. If $Y_C = \emptyset$, then $p_C = \perp$, so C returns "gap". Assume $Y_C \neq \emptyset$. As $p_C.y \in Y_C$, we know $y \geq p_C.y$, so $Y_1 = [p_C.y + 1, y]$ and $Y_2 = [\ell, p_C.y - 1]$. This means C_2 cannot return a gap message, and C_1 returns "gap", so C returns "gap" in line 26.

Finally, we show (c). Suppose $Y_C \neq [\ell_C, y']$ for all $y' \in [\ell_C - 1, h_C]$. We know C does not set T, so C_1 cannot return a message of the form "gap at y' below q", and thus $Y_1 = [p_C.y + 1, y]$ for some $y \in [p_C.y, j]$. This means $Y_2 \neq [\ell_C, p_C.y - 1] = [\ell_{C_2}, h_{C_2}]$ (otherwise $Y_C = [\ell_C, y]$), so C_2 returns a gap message. Suppose C_2 returns "gap". Then p_C returns "gap at $(p_C.y - 1)$ below p_C" and we know $h_{C_2} = p_C.y - 1 \notin Y_2$. Now suppose C_2 returns "gap at y below q", then C returns "gap at y below p_C". Moreover, we know that $q \in S_2$ and $y \in [\ell_{C_2}, q.y - 1] \subseteq [\ell_C, p_C.y - 1]$. □

We are now ready to prove correctness of $\mathbb{A}$ with respect to (ii). Suppose the algorithm asserts (p, q, y) is a solution for (ii). At some point, say in call C, the variable T must be set in line 20. This means the first call of C (line 13) must have returned "gap at y below q", which by claim 5.13 (b) implies that (p, q, y) is a solution for (ii).

We now prove completeness of $\mathbb{A}$ with respect to (ii). Suppose $p, q \in S$ and $y \in [m]$ such that (p, q, y) is a solution for (ii). We need to show that T is set to some value in line 20 at some point. Suppose this is not the case.

As $213 \not\preceq S$, we have $r_C = \perp$ in every call C. This means p is consumed in line 10 in some call C. Let $Q = \{r \in S \mid p.x < r.x \leq q.x\}$ and let C_1 be the first call C makes (in line 13, with parameters $\ell_{C_1} = p.y + 1$ and $h_{C_1} = h_C$). We claim that all points in Q are consumed by C_1 or its sub-calls. If this is not the case, there is some $r \in Q$ with $r.y < p.y$ or $h_C < r.y$. In the former case, (p, r, q) form an occurrence of 213 in S, a contradiction. In the latter case, we know $h_C < m$, so there is some $s \in S$ with $s.x < p.x$ and $s.y = h_C$, which means (s, p, q) form an occurrence of 213 in S, again a contradiction. By Claim 5.13 (c), the call C_1 returns "gap at y' below q'" for some y' and q', which causes C to set T. This concludes the correctness and completeness proof of $\mathbb{A}$.

5.3.3 An algorithm for 213-OLPPM

Lemma 5.14. *213-OLPPM can be solved using $\mathcal{O}(\sqrt{n} \log n)$ space.*

Note that, in contrast to the algorithm from lemma 5.8, the algorithm presented in this section does not return an occurrence when it accepts.

Algorithm. Similar to the algorithm of lemma 5.8, we partition the input into $t = \lceil \sqrt{n} \rceil$ vertical strips $S_1, S_2, \ldots, S_t$ of roughly equal size. We again assume we are given the points of S_τ, ordered by abscissa. Here we consider four way ways an occurrence of 213 could be distributed over the strips. The following four parts run in parallel.

(1) Again, when we finished reading strip S_i, we first search for 213 in S_i using the algorithm $\mathbb{A}$ from section 5.3.2 (alternatively, we could just use brute force, which increases the running time, but does not affect the space requirements).

(2) To find occurrences (p, q, r) of 213 where $p, q \in S_i$, we maintain a value $L \in [n]$, initially set to n. After we finished reading S_i, we find the lowest point $p \in S_i$ such that p, q are a decreasing pair for some $q \in S_i$, i.e. $p.x < q.x$ and $q.y < p.y$. If there is such a p and $p.y < L$, we update $L \leftarrow p.y$. Whenever we read a point p, if $p.y > L$, we accept immediately.

(3) To find occurrences (p, q, r) with $q, r \in S_i$ for some i, we use a more sophisticated technique. We will need to store two points $\ell_i, h_i \in [n]^2 \cup \{\perp\}$ and a counter $k_i \in \{0, 1, \ldots, h_i - \ell_i - 1\}$ for each strip S_i.

Suppose we just read S_i. We first define (but do not store)

$$I_i = \{(p, q) \in S_i^2 \mid p.x < q.x, \exists r \in S_\tau \setminus S_i : p.y < r.y < q.y\},$$

the set of increasing pairs in S_i where some point in a different strip lies vertically between the two points.

Observe that we can check if $(p, q) \in I_i$ using only S_i, as we know exactly which values occur in S_i, and, therefore, which occur in the other strips. We can also enumerate the pairs in I_i one-by-one in $\mathcal{O}(n)$ time and using $\mathcal{O}(\log n)$ additional space.

If $I_i = \emptyset$, we set $\ell_i = h_i = k_i = \perp$ and ignore the rest of this case. Otherwise, we set ℓ_i to be the lowest point such that there is a pair $(\ell_i, q) \in I_i$ and set h_i to be the highest point such that there is a pair $(p, h_i) \in I_i$. We initialize k_i to be the number of points $p \in S_i$ where $\ell_i.y < p.y < h_i.y$. From now on, whenever we read a point p with $\ell_i.y < p.y < h_i.y$ in a later strip, we increment k_i by one. Finally, after reading the whole input, we accept if $k_i < h_i - \ell_i - 1$ (recall that we do this for each $i \in [n]$).

(4) Finally, we consider occurrences where all three points are in different strips. We use a "counter" technique similar to (3). After reading S_i, we compute the lowest point ℓ'_i. Then, we initialize h'_i to be the highest point in S_i to the right of ℓ'_i and initialize the counter k'_i to the number points in S_i above and to the right of ℓ'_i. In subsequent strips, whenever we read a point p with $\ell'_i.y < p.y$, we increment k'_i by one. Then, if $h'_i.y < p.y$, we set $h'_i \leftarrow p$. Finally, after reading the whole input, we accept if $k'_i < h'_i.y - \ell'_i.y$.

If, at the end of the input, we have not accepted yet due to (1-4), we reject.

Required space. For each strip S_i, we need to store the points ℓ_i, h_i, ℓ'_i and h'_i and the values $k_i, k'_i \in [n]$. Additionally, we store $L \in [n]$ and the $\mathcal{O}(\sqrt{n})$ points in S_i. The other operations such as the enumeration of increasing/decreasing pairs only need $\mathcal{O}(\log n)$ space. Thus, the maximal space requirement is $\mathcal{O}(\sqrt{n} \log n)$.

Running time. For each strip, (1) needs $\mathcal{O}(\sqrt{n})$ time for algorithm $\mathbb{A}$. (2) requires constant time per point. Thus, (1) and (2) amount to a total time of $\mathcal{O}(n)$.

In (3) and (4), we enumerate the $\mathcal{O}((\sqrt{n})^2) = \mathcal{O}(n)$ increasing and decreasing pairs, needing constant time for each such pair. Moreover, we need to compare each newly read point with the variables $\ell_i, h_i, \ell'_i, h'_i$ in the previous strips, thus spending $\mathcal{O}(\sqrt{n})$ time per point. The overall running time is thus $\mathcal{O}(n^{3/2})$.

For the proofs of correctness and completeness, we write $L^{(i)}$ for the value of L after processing strip S_i. On the other hand, we only consider the final values of k_i and h'_i.

Correctness. We show that if the algorithm accepts, then τ contains 213.

Suppose the algorithm accepts the input τ. If this was due to finding 213 in a single strip (via part (1), i.e. algorithm $\mathbb{A}$), then τ clearly contains 213. If part (2) of the algorithm finds a point $r \in S_i$ with $r.y < L^{(i-1)}$, then, for some $j < i$, the strip S_j contains a decreasing pair of points (p, q) with $p.y = L^{(i-1)} < r.y$, thus p, q, r form an occurrence of 213 in τ.

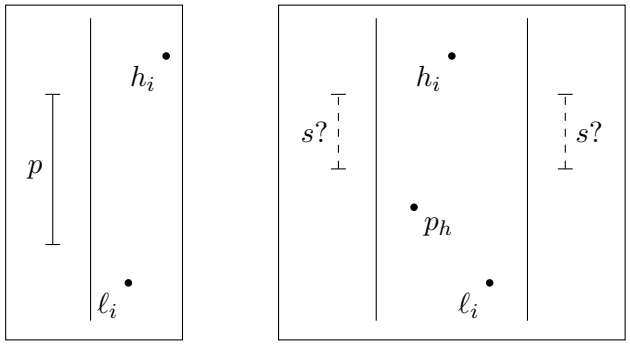

Figure 5.2: Illustration of the correctness of part (2) of the algorithm.

Now consider the case that (3) accepts, i.e. we have $k_i < h_i - \ell_i - 1$ for some $i \in [n]$ in the end. Then k_i is the number of points p in the strips $S_i, S_{i+1}, \ldots, S_n$ with $\ell_i.y < p.y < h_i.y$. As there are $h_i.y - \ell_i.y - 1$ such points in the input S_τ, and $k_i < h_i.y - \ell_i.y - 1$, there must be at least one such point $p \in S_j$ for some $j < i$.

By definition, there are $q_l, p_h \in S_i$ such that $(\ell_i, q_l), (p_h, h_i) \in I_i$. If $\ell_i.x < h_i.x$, then p, ℓ_i, h_i form an occurrence of 213 in τ (see fig. 5.2, left). Otherwise, $p_h.x < h_i.x < \ell_i.x$. By definition of I_i, there is some $s \in S_\tau \setminus S_i$ with $p_h.y < s.y < h_i.y$ (see fig. 5.2, right). If $s \in S_j$ with $j < i$, then s, p_h, h_i form an occurrence of 213. On the other hand, if $s \in S_j$ for some $i < j$, then p_h, ℓ_i, s form an occurrence of 213, as $\ell_i.y < p_h.y$ by minimality of $\ell_i.y$.

Finally, suppose (4) accepts, i.e. $k'_i < h'_i.y - \ell'_i.y$ holds for some $i \in [n]$ in the end. k'_i is the number of points to the right and above ℓ'_i (including h'_i). There are exactly $h'_i.y - \ell'_i.y - 1$ points $p \in S_\tau$ with $\ell'_i.y < p.y < h'_i.y$, but only $k'_i - 1$ such points that are also to the right of ℓ'_i, so there must be at least one point q with $q.x < \ell'_i.x < h'_i.x$ and $\ell'_i.y < q.y < h'_i.y$. Thus (q, ℓ'_i, h'_i) form an occurrence of 213 in S_τ.

Completeness. We show that if $213 \preceq \tau$, then the algorithm accepts.

Suppose that S_τ contains three points p, q, r that form an occurrence of 213. If there is some i such that $p, q, r \in S_i$, part (1) will find (p, q, r).

Suppose $p, q \in S_i$ and $r \notin S_i$ for some i. We show that then (2) accepts. Let (p', q') be decreasing pair in S_i such that $p'.y$ is minimal. Let $r \in S_j$. As $i \leq (j-1)$, by definition of L and minimality of $p'.y$, we know $L^{(j-1)} \leq p'.y \leq p.y < r.y$. As such, (2) accepts when the algorithm reads r and notices $L < r.y$.

Suppose $p \notin S_i$ and $q, r \in S_i$ for some i. We show that then (3) accepts. We have $\ell_i.y \leq q.y < p.y < r.y \leq h_i.y$. As $p \in S_j$ for some $j < i$, the number k_i of points s with $\ell_i.y < s.y < h_i.y$ to the right of S_i is less than the $h_i.y - \ell_i.y - 1$ (the total number of such points in S_τ). Thus, (3) accepts.

Finally, suppose p, q and r are in different strips. We show that then (4) accepts. Let $q \in S_i$. Then $\ell'_i.y < q.y < p.y < r.y < h'_i.y$. As $p \in S_j$ for some $j < i$, there the number k'_i of points s to the right of ℓ'_i which satisfy $\ell'_i.y < s.y$ (and, by definition, $s.y \leq h'_i.y$) is at most $h'_i.y - \ell'_i.y - 1$. Thus, we accept.

This concludes the proof of correctness and completeness of the algorithm for 213-OLPPM.

We remark that the algorithm for 213-OLPPM (Lemma 5.14) has two major disadvantages compared with the algorithm for 312-OLPPM (Lemma 5.8). First, the latter finds an occurrence of 312 if it exists, while the former cannot find an occurrence of 213 in all cases. Second, it does not seem like the running time of the former algorithm can be improved below $\Theta(n^{3/2})$, as it heavily relies on updating each of up to $2\sqrt{n}-2$ counters for each newly read value.

As noted in the introduction, π-OLPPM and π^{C}-OLPPM have essentially the same complexity, so Lemma 5.8 and Lemma 5.14 imply Theorem 5.3.

5.3.4 Lower bounds for π-OLPPM

We now establish linear lower bounds for finding the remaining non-monotone permutations of size at least 4. We will do so by proving the lower bounds for all 4-permutations, and then show that π-OLPPM for an arbitrary non-monotone permutation π of size at least 5 can be reduced to that case. Recall that we only have to consider one π out of every symmetry class when proving DISJ-hardness. In the following, we prove that π-OLPPM is DISJ-hard for $\pi \in \{4231, 4213, 4132, 4123, 4312, 3142, 2143\}$.

Lemma 5.15. *Let $\pi \in \{4231, 4213, 4132, 4123\}$. Then π-OLPPM is DISJ-hard.*

Proof. Let $S, T \subseteq [n]$. We define $f_1(S) = (a_1, a_2, \ldots, a_{2n})$, where for $I \in [n]$,

$$(a_{2i-1,2i}) = \begin{cases} (4(i-1)+\pi(1), 4(i-1)+\pi(2)), & \text{if } i \in S, \\ (4(i-1)+\pi(2), 4(i-1)+\pi(1)), & \text{otherwise.} \end{cases}$$

For $i \in [n]$, let $d(i) = i$ if $\pi = 4231$, and let $d(i) = n+1-i$ otherwise. Note that always $d(d(i)) = i$. We define $f_2(T) = (b_1, b_2, \ldots, b_n)$, where for $i \in [n]$,

$$(b_{2i-1}, b_{2i}) = \begin{cases} (4(d(i)-1)+\pi(3), 4(d(i)-1)+\pi(4)), & \text{if } d(i) \in T, \\ (4(d(i)-1)+\pi(4), 4(d(i)-1)+\pi(3)), & \text{otherwise.} \end{cases}$$

We set $f_3(S) = f_4(T) = \varepsilon$. Observe that the sequence $\tau = f_1(S)f_2(T)$ contains each value in $[4n]$ exactly once, i.e. σ is a $4n$-permutation. Figure 5.3 shows examples of the construction.

We need to prove that $\pi \preceq \tau$ if and only if $S \cap T \neq \emptyset$. Suppose there is some $i \in S \cap T$. As $d(d(i)) = i \in T$, the sequence $(a_{2i-1}, a_{2i}, b_{2d(i)-1}, b_{2d(i)})$ is a subsequence of τ, implying $\pi \preceq \tau$.

Now suppose $\pi \preceq \tau$ via the index mapping $\phi\colon [4] \to [4n]$. We need to show that $S \cap T \neq \emptyset$.

We first show that $\phi(3) > 2n$. Suppose that is not the case. Then, by definition of $f_1(S)$, there is at most one index $i < \phi(3)$ such that $\tau(i) > \tau(\phi(3))$, namely $i = (\phi(3)-1)$. As $\pi(1) > \pi(3)$, this implies $\phi(1) = i-1$, so $i-1 = \phi(1) < \phi(2) < \phi(3) = i$, a contradiction.

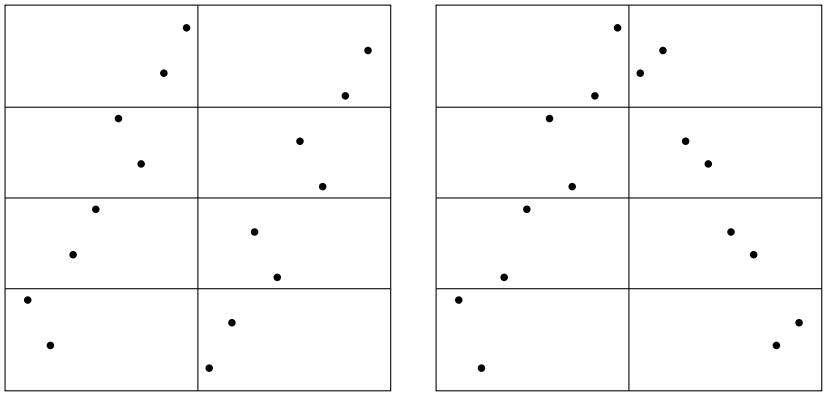

Figure 5.3: Construction of $\tau = f_1(S)f_2(S)$ for $\pi = 4231$ (left) and for $\pi = 4132$ (right) with $n = 4$, $S = \{1, 3\}$ and $T = \{2, 3\}$

Second, we show that $\phi(2) \leq 2n$. Assume this is not the case and $\pi = 4231$. As $d(i) = i$, there is at most one index $i > \phi(2)$ with $\tau(i) < \tau(\phi(2))$, namely $i = \phi(2) + 1$. As above, this is a contradiction. In the case that $\pi \neq 4231$, we have $\pi(2) < \pi(4)$ and $d(i)$ is decreasing, which similarly implies a contradiction.

We now have $\phi(1) < \phi(2) \leq 2n < \phi(3) < \phi(4)$. As $\pi(1) > \pi(2)$, we know that $\tau(\phi(1)) > \tau(\phi(2))$. This means $\phi(1) = 2i - 1$ and $\phi(2) = 2i$ for some $i \in S$, as $f_1(S)$ contains no other decreasing value pairs. Consequently, we have $\tau(\phi(j)) = 4(i-1)+\pi(j)$ for $j \in [1, 2]$.

To conclude the proof, we show that $\tau(\phi(j)) = 4(i - 1) + \pi(j)$ for $j \in [3, 4]$, which implies $i \in T$ and thus $i \in S \cap T$. We distinguish three cases.

- Let $\pi = 4231$. We have $\pi(2) < \pi(3) < \pi(1)$, thus

$$4(i - 1) + 2 = \tau(\phi(2)) < \tau(\phi(3)) < \tau(\phi(1)) = 4(i - 1) + 4,$$

  implying that indeed $\tau(\phi(3)) = 4(i - 1) + 3$. Now $\tau(\phi(3)), \tau(\phi(4))$ must be a decreasing pair, which leaves only $4(i - 1) + 1$ as a possible value for $\tau(\phi(4))$.

- Let $\pi = 4213$. By a similar argument as above, $\pi(2) < \pi(4) < \pi(1)$ implies that $\tau(\phi(4)) = 4(i - 1) + 3$. Now $\tau(\phi(3)), \tau(\phi(4))$ must be an *increasing* pair, which implies $\tau(\phi(3)) = 4(i - 1) + 1$.

- Finally, let $\pi \in \{4123, 4123\}$. Now $\pi(2) < \pi(3) < \pi(1)$ and $\pi(2) < \pi(4) < \pi(1)$, directly implying that $\tau(\phi(3)) = 4(i-1)+\pi(3)$ and $\tau(\phi(4)) = 4(i-1)+\pi(4)$, thus $i \in T$.

□

Lemma 5.16. 4312-OLPPM *is* DISJ*-hard.*

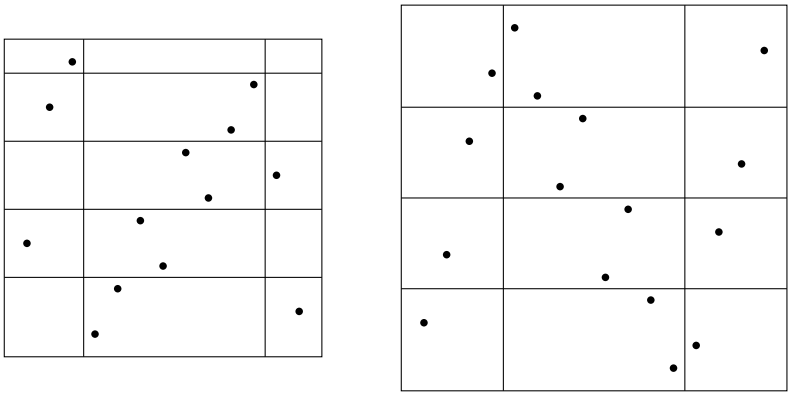

Figure 5.4: Construction of $\tau = f_1(S)f_2(S)f_3(T)$ for $\pi = 4312$ (left) and for $\pi = 3142$ (right) with $n = 4$, $S = \{1, 3\}$ and $T = \{2, 3\}$

Proof. Let $S, T \subseteq [n]$. Let $f_1(S)$ consist of the values $3(i-1)+2$ for $i \notin S$ in ascending order, and the value $3n+1$ at the end. Let $f_2(T) = b_1 b_2 \dots b_{2n}$, where

$$(b_{2i-1}, b_{2i}) = \begin{cases} (3(i-1)+3, 3(i-1)+1), & \text{if } i \in T, \\ (3(i-1)+1, 3(i-1)+3), & \text{otherwise.} \end{cases}$$

Finally, let $f_3(S)$ contain the values $3(i-1)+2$ for $i \in S$, in *descending* order, and let $f_4(T) = \varepsilon$. The sequence $\tau = f_1(S)f_2(T)f_3(S)$ is a $(3n+1)$-permutation. Figure 5.4 (left) shows an example of the construction. We claim that $4231 \preceq \tau$ if and only if $S \cap T \neq \emptyset$.

Suppose $i \in S \cap T$. Then $b_{2i-1} = 3(i-1)+3$, $b_{2i} = 3(i-1)+1$ and the value $3(i-1)+2$ occurs in $f_3(S)$. The value $3n+1$ always occurs in $f_1(S)$. These four values together form an occurrence of 4312 in τ.

Now suppose 4312 occurs in τ via the index mapping $\phi\colon [4] \to [n]$. Let $k_1 = |f_1(S)|$ and $k_2 = |f_2(T)|$. First, as $f_1(S)$ is increasing and $f_3(S)$ is decreasing, we know $k_1 < \phi(2)$ and $\phi(3) \leq k_2$. Second, we claim that $\phi(1) \leq k_1$. Assume this is not the case. Then $k_1 < \phi(1) < \phi(2) < \phi(3) < k_2$, so the descending sequence $(\tau(\phi(1)), \tau(\phi(2)), \tau(\phi(3))$ is a subsequence of $f_2(T)$. However, $f_2(S)$ is 2-increasing, a contradiction. Third, $k_2 < \phi(4)$, as $f_2(T)$ does not contain the pattern 312. In conclusion, we have $\phi(1) \leq k_1 < \phi(2) < \phi(3) \leq k_2 < \phi(4)$.

By definition of $f_2(T)$ and using $\tau(\phi(2)) > \tau(\phi(3))$, we know that there is some $i \in T$ such that $\tau(\phi(2)) = 3(i-1)+3$ and $\tau(\phi(3)) = 3(i-1)+1$. Thus $\tau(\phi(4)) = 3(i-1)+2$. This value occurring in $f_3(S)$ implies that $i \in S$. □

Lemma 5.17. 3142-OLPPM *and* 2143-OLPPM *are* DISJ*-hard.*

Proof. Let $S, T \subseteq [n]$. Let $f_1(S) = a_1, a_2, \dots, a_n$ and $f_3(S) = c_1, c_2, \dots, c_n$, where for

$i \in [n]$,

$$a_i = \begin{cases} 4(i-1)+\pi(1), & \text{if } i \in S, \\ 4(i-1)+\pi(4), & \text{otherwise.} \end{cases}$$

$$c_i = \begin{cases} 4(i-1)+\pi(4), & \text{if } i \in S, \\ 4(i-1)+\pi(1), & \text{otherwise.} \end{cases}$$

Let $f_2(S) = b_1 b_2 \dots b_{2n}$ where for $i \in [n]$,

$$(b_{2i-1}, b_{2i}) = \begin{cases} (4(n-i)+1, 4(n-i)+4), & \text{if } i \in T, \\ (4(n-i)+4, 4(n-i)+1), & \text{otherwise.} \end{cases}$$

Let $f_4(T) = \varepsilon$. The sequence $\tau = f_1(S)f_2(T)f_3(S)$ is a $4n$-permutation. Figure 5.4 (right) shows an example of the construction.
We show that $\pi \preceq \tau$ if and only if $S \cap T \neq \emptyset$. Suppose there is some $i \in S \cap T$. Then $(a_i, b_{2i-1}, b_{2i}, c_i)$ is an occurrence of π in τ.

Now suppose $\pi \preceq \tau$ via the index mapping $\phi\colon [4] \to [4n]$. Both $f_1(S)$ and $f_3(S)$ are increasing. As $\pi(1) > \pi(2)$ and $\pi(3) > \pi(4)$, we have $n < \phi(2) \leq \phi(3) \leq 3n$. Moreover, we know $(\phi(2), \phi(3)) = (n+2i-1, n+2i)$ for some $i \in T$, as there are no other increasing pairs in $f_2(T)$. This means $\tau(\phi(2)) = 4(i-1)+1$ and $\tau(\phi(3)) = 4(i-1)+4$. Now, as $\pi(1)$ and $\pi(4)$ are between $\pi(2) = 1$ ad $\pi(3) = 4$, we have $\tau(\phi(1)) = 4(i-1)+\pi(1)$ and $\tau(\phi(4)) = 4(i-1)+\pi(4)$, which is only possible if $i \in S$. □

Having completed all cases of π-OLPPM with $|\pi| = 4$, we now restate and prove theorem 5.4.

Theorem 5.4. *Let π be a non-monotone permutation of size at least 4. Then π-OLPPM cannot be solved using $o(n)$ bits of space, even if we allow a constant number of passes over the input.*

Proof. Let π be a non-monotone k-permutation, where $k \geq 4$. We show that π-OLPPM is DISJ-hard by induction on k. If $k = 4$, one of the lemmas 5.15 to 5.17 apply by symmetry.

Now let $k > 4$. Let $\pi' = \mathrm{red}(\pi(1), \pi(2), \dots, \pi(k-1))$. Without loss of generality, assume that $\pi(k-1) > \pi(k)$.

By induction hypothesis, π'-OLPPM is DISJ-hard with some $c \in \mathbb{N}$. Let $n \in \mathbb{N}_+$ and let $m = cn$. Then π'-OLPPM is m-DISJ$_n$-hard, say with parameters f_1', f_2', f_3', f_4'. For $i \in [4]$, we construct f_i from f_i' by doubling every value in the sequence, and additionally appending the sequence $1, 3, \dots, 2n+1$ to the result of f_4'. Let $\tau = f_1(S)f_2(T)f_3(S)f_4(T)$ and let $\tau' = f_1'(S)f_2'(T)f_3'(S)f_4'(T)$. We claim that $\pi \preceq \tau$ if and only if $\pi' \preceq \tau'$. Assuming this claim holds, given $S, T \subseteq [n]$, we have

$$\begin{aligned} &\pi \preceq f_1(S)f_2(T)f_3(S)f_4(T) \\ &\iff \pi' \preceq f_1'(S)f_2'(T)f_3'(S)f_4'(T) && \text{(our claim)} \\ &\iff S \cap T \neq \emptyset && \text{(hardness of } \pi'\text{-OLPPM).} \end{aligned}$$

As $f_1(S)f_2(T)f_3(S)f_4(T)$ has length $2m+1$, this shows π-OLPPM is $(3cn+1)$-DISJ$_n$-hard for every n, and thus is DISJ-hard.

It remains to show the claim that $\pi \preceq \tau$ if and only if $\pi' \preceq \tau'$. Observe that $\tau(i) = 2\tau'(i)$ for $i \in [n]$ and $\tau(n+i) = 2i-1$ for $i \in [n+1]$.

Suppose $\pi' \preceq \tau'$ via the index mapping $\phi\colon [k-1] \to [n]$. Let $i \in [n]$ such that $\pi(i) = \pi(k)+1$ (recall that $\pi(k) < k$), and let $j \in [2n-1]$ such that $\tau(j) = 2\tau'(\phi(i))-1$. Then, $j > n$, and $\phi(1), \phi(2), \ldots, \phi(n), j$ are the indices of a subsequence of τ that is order-isomorphic to π.

Now suppose $\pi \preceq \tau$ via the index mapping $\phi\colon [k] \to [2n+1]$. As $\pi(k-1) > \pi(k)$, and the last $n+1$ values of τ' are increasing, we know that $\phi(k-1) \leq n$. Now the first $k-1$ values of π (order-isomorphic to π') are a subsequence of the first n values of τ (order-isomorphic to τ'), so $\pi' \preceq \tau'$. □

6 Conclusion

In this thesis, we investigated the complexity of the permutation pattern matching problem. In chapter 3, we considered the performance of *treewidth-based* algorithms. We were able to show that for almost every principal class $\mathrm{Av}(\sigma)$, the restriction to $\mathrm{Av}(\sigma)$-Pattern PPM does not make these algorithms significantly faster, as those classes contain permutations with incidence graphs of almost linear treewidth. Our results strengthen the dichotomy of Jelínek and Kynčl [19] between polynomial-solvable and NP-hard cases of $\mathrm{Av}(\sigma)$-Pattern PPM by proving that the treewidth-based algorithms cannot be faster than $n^{\Omega(\sqrt{k})}$ in the latter case. On the other hand, we were able to give an $n^{\mathcal{O}(\sqrt{k})}$ upper bound on the complexity of Mon_2-Pattern PPM. Although we have determined the maximal treewidth of almost all principal permutation classes up to a logarithmic factor, there are still several cases where the maximal treewidth is unknown.

Open question 6.1. *What is the maximal treewidth of the remaining principal classes? Are there cases were it is neither* $\Omega(n/\log n)$ *nor* $\Theta(\sqrt{n})$*?*

Another interesting question is whether the treewidth-based algorithms are optimal.

Open question 6.2. *Is there a permutation class* $\mathcal{C}$ *such that* $\mathcal{C}$-Pattern #PPM *or* $\mathcal{C}$-Pattern PPM *can be solved in substantially less time than* $n^{\Theta(\mathrm{tw}(G_\pi))}$*?*

In chapter 4, we gave a general lower bound on the complexity of the counting version of PPM. Again, the restricted problem $\mathrm{Av}(\sigma)$-Pattern #PPM is almost as hard as PPM, for almost all avoided patterns σ. However, there are much more cases open than in the treewidth question. Additionally, the lower bound does not apply to the decision problem PPM.

Open question 6.3. *For which permutations* π *can* $\mathrm{Av}(\pi)$-Pattern #PPM *be solved in* $n^{\mathcal{O}(k^c)}$ *time for some* $c < 1$*?*

Finally, in chapter 5, we considered the space-requirement of on-line algorithms deciding PPM with fixed pattern π. It was already known that this problem can be solved in logarithmic space if π is monotone. Otherwise, if π has length at least 4, we showed that we need almost linear space. If π has size 3 (and is not monotone), the space complexity depends on the exact formulation of the problem. If the input is a *sequence* of length n that may omit some numbers in $[n]$, then we again need almost linear space. On the other hand, if the input is a permutation, we can solve the problem using only $\mathcal{O}(\sqrt{n}\log n)$ bits of space. We leave open the question whether that is optimal.

Open question 6.4. *Is there an algorithm for* 312*-OLPPM or* 213*-OLPPM that uses* $o(\sqrt{n})$ *space?*

Acknowledgments

The author would like to thank Walter G. Berendsohn and Alireza Selahi for their helpful comments and suggestions, and especially László Kozma for his support throughout the writing of this thesis.

Bibliography

[1] Shlomo Ahal and Yuri Rabinovich, *On complexity of the subpattern problem*, SIAM Journal on Discrete Mathematics **22** (2008), no. 2, 629–649.

[2] Michael H. Albert, Robert E. L. Aldred, Mike D. Atkinson, and Derek A. Holton, *Algorithms for pattern involvement in permutations*, Algorithms and Computation (Peter Eades and Tadao Takaoka, eds.), Springer Berlin Heidelberg, 2001, pp. 355–367.

[3] Benjamin Aram Berendsohn, László Kozma, and Dániel Marx, *Finding and counting permutations via CSPs*, 14th International Symposium on Parameterized and Exact Computation (IPEC), to appear.

[4] Sebastian Berndt, *Computing tree width: From theory to practice and back*, Sailing Routes in the World of Computation (Florin Manea, Russell G. Miller, and Dirk Nowotka, eds.), Springer International Publishing, Cham, 2018, pp. 81–88.

[5] Frank Bernhart and Paul C. Kainen, *The book thickness of a graph*, Journal of Combinatorial Theory, Series B **27** (1979), no. 3, 320–331.

[6] Hans L. Bodlaender, *Dynamic programming on graphs with bounded treewidth*, Automata, Languages and Programming (Timo Lepistö and Arto Salomaa, eds.), Springer Berlin Heidelberg, 1988, pp. 105–118.

[7] Hans L. Bodlaender, *A partial k-arboretum of graphs with bounded treewidth*, Theoretical Computer Science **209** (1998), no. 1, 1–45.

[8] Prosenjit Bose, Jonathan F. Buss, and Anna Lubiw, *Pattern matching for permutations*, Information Processing Letters **65** (1998), no. 5, 277–283.

[9] Karl Bringmann, László Kozma, Shay Moran, and N. S. Narayanaswamy, *Hitting Set for hypergraphs of low VC-dimension*, arXiv e-prints (2015), arXiv:1512.00481.

[10] Marie-Louise Bruner and Martin Lackner, *A fast algorithm for permutation pattern matching based on alternating runs*, Algorithmica **75** (2016), no. 1, 84–117.

[11] Miklós Bóna, *A survey of stack-sorting disciplines*, The Electronic Jorunal of Combinatorics **9** (2003).

[12] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to algorithms*, 2nd ed ed., MIT Press, Cambridge, Mass., 2001.

[13] Marek Cygan, Lukasz Kowalik, and Arkadiusz Socala, *Improving TSP tours using dynamic programming over tree decompositions*, 25th Annual European Symposium on Algorithms (Kirk Pruhs and Christian Sohler, eds.), Leibniz International Proceedings in Informatics (LIPIcs), vol. 87, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017, pp. 30:1–30:14.

[14] P. Erdős and G. Szekeres, *A combinatorial problem in geometry*, Compositio Mathematica **2** (1935), 463–470.

[15] Michael L. Fredman, *On computing the length of longest increasing subsequences*, Discrete Mathematics **11** (1975), no. 1, 29–35.

[16] Sylvain Guillemot and Dániel Marx, *Finding small patterns in permutations in linear time*, Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms (Philadelphia, PA, USA), SODA '14, Society for Industrial and Applied Mathematics, 2014, pp. 82–101.

[17] Louis Ibarra, *Finding pattern matchings for permutations*, Information Processing Letters **61** (1997), no. 6, 293–295.

[18] Russell Impagliazzo and Ramamohan Paturi, *On the complexity of k-sat*, Journal of Computer and System Sciences **62** (2001), no. 2, 367–375.

[19] Vít Jelínek and Jan Kynčl, *Hardness of permutation pattern matching*, Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (Philadelphia, PA, USA), Society for Industrial and Applied Mathematics, 2017, pp. 387–396.

[20] Vít Jelínek, Michal Opler, and Pavel Valtr, *Generalized coloring of permutations*, 26th Annual European Symposium on Algorithms (Yossi Azar, Hannah Bast, and Grzegorz Herman, eds.), Leibniz International Proceedings in Informatics (LIPIcs), vol. 112, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018, pp. 50:1–50:14.

[21] Donald E. Knuth, *The art of computer programming, volume 1: Fundamental algorithms*, Addison-Wesley, 1968.

[22] László Kozma, *Faster and simpler algorithms for finding large patterns in permutations*, arXiv e-prints (2019), arXiv:1902.08809.

[23] Eval Kushilevitz and Noam Nisan, *Communication complexity*, 1. publ. ed., Cambridge Univ. Press, Cambridge, 1997.

[24] Richard J. Lipton and Robert Endre Tarjan, *A separator theorem for planar graphs*, SIAM Journal on Applied Mathematics **36** (1979), no. 2, 177–189.

[25] László Lovász, *Paths, flows and VLSI layout*, ch. Communication Complexity: a survey, pp. 235–265, Springer, 1990.

[26] Adam Marcus and Gábor Tardos, *Excluded permutation matrices and the stanley–wilf conjecture*, Journal of Combinatorial Theory, Series A **107** (2004), no. 1, 153–160.

[27] Dániel Marx, *Can you beat treewidth?*, Theory of Computing **6** (2010), no. 5, 85–112.

[28] Vaughan R. Pratt, *Computing permutations with double-ended queues, parallel stacks and parallel queues*, Proceedings of the Fifth Annual ACM Symposium on Theory of Computing (New York, NY, USA), ACM, 1973, pp. 268–277.

[29] Vincent Vatter, *Small permutation classes*, Proceedings of the London Mathematical Society **103** (2011), no. 5, 879–921.

[30] Vincent Vatter, *Permutation classes*, arXiv e-prints (2014), arXiv:1409.5159.