Permutation Pattern matching in (213, 231)-avoiding permutations

Both Emerite Neou, Romeo Rizzi, Stéphane Vialette

Permutation Pattern matching in (213, 231)-avoiding permutations

Both Emerite Neou*[1] Romeo Rizzi[2] and Stéphane Vialette[1]

[1] Université Paris-Est, LIGM (UMR 8049), CNRS, UPEM, ESIEE Paris, ENPC, F-77454, Marne-la-Valle, France
{neou,vialette}@univ-mlv.fr

[2] Department of Computer Science, Università degli Studi di Verona, Italy
romeo.rizzi@univr.it

Abstract. Given permutations σ of size k and π of size n with $k < n$, the *permutation pattern matching* problem is to decide whether σ occurs π as an order-isomorphic subsequence. We give a linear-time algorithm in case both π and σ avoid the two size-3 permutations 213 and 231. For the special case where only σ avoids 213 and 231, we present a $O(max(kn^2, n^2 \log\log n)$-time algorithm. We extend our research to bivincular patterns that avoid 213 and 231 and present a $O(kn^4)$-time algorithm. Finally we look at the related problem of the longest subsequence which avoids 213 and 231.

1 Introduction

A permutation σ is said to *occur* in another permutation π (or π *contains* σ), denoted $\sigma \preceq \pi$, if there exists a subsequence of elements of π that has the same relative order as σ. Otherwise, π is said to *avoid* the permutation σ. For example a permutation contains the permutation 123 (resp. 321) if it has an increasing (resp. a decreasing) subsequence of size 3. Similarly, 213 occurs in 6152347, but 231 does not occurs in 6152347. During the last decade, the study of patterns in permutations has become a very active area of research [15] and a whole annual conference (Permutation Patterns) focuses on patterns in permutations.

We consider here the so-called *permutation pattern matching* problem (also sometimes referred to as the *pattern involvement problem*): Given two permutations σ of size k and π of size n, the problem is to decide whether $\sigma \preceq \pi$ (the problem is attributed to Wilf in [5]). The permutation pattern matching is known to be **NP**-hard [5]. It is however polynomial-time solvable by brute-force enumeration if σ has bounded size. Improvements to this algorithm were presented in [3] and [1], the latter describing a $O(n^{0.47k+o(k)})$-time algorithm. Bruner and Lackner [8] gave a fixed-parameter algorithm solving the permutation pattern matching problem with an exponential worst-case runtime of $O(1.52^n.n.k)$. This is an improvement upon the $O(k\binom{n}{k})$ runtime required by brute-force search

* On a Co-tutelle Agreement with the Department of Mathematics of the University of Trento

without imposing restrictions on σ and π, where one has to enumerate all the $\binom{n}{k}$ different subsequences of size k in π and test if one of them has the same relative order as σ. Guillemot and Marx [11] showed that the permutation pattern matching problem is solvable in $2^{O(k^2 \log k)}$, and hence is fixed-parameter tractable with respect to the length k of the pattern σ (standard parameterization). However, Mach proved that the permutation containment problem under the standard parameterization does not have a polynomial size kernel (assuming $\mathbf{NP} \not\subseteq \mathbf{coNP/poly}$ [16].

A few particular cases of the permutation pattern matching problem have been attacked successfully. The case of increasing patterns is solvable in $O(n \log \log k)$-time [9], improving the previous 30-year bound of $O(n \log k)$. Furthermore, the patterns 132, 213, 231, 312 can all be handled in linear-time by stack sorting algorithms. Any pattern of size 4 can be detected in $O(n \log n)$ time [3]. Algorithmic issues for 321-avoiding patterns matching for permutations have been investigated in [12] and more recently in [2]. The permutation pattern matching problem is also solvable in polynomial-time for separable patterns [13, 5] (see also [6] for the longest common subsequence problem and related ones on separable permutations). Separable permutations are permutations that do not have an occurrence of 2413 nor 3142, and they are enumerated by the Schröder numbers. (Notice that the separable permutations include as a special case the stack-sortable permutations, which avoid the pattern 231.) The major negative result is given in [14]. They prove that the permutation pattern matching problem is **NP**-complete when σ has no decreasing subsequence of length 3 and π has no decreasing subsequence of length 4. This provides the first known example of the permutation pattern matching problem being hard when one or both of π and σ are restricted to a proper hereditary class of permutations.

There exist many generalisations of patterns that are worth considering in the context of algorithmic issues in pattern matching (see [15] for an up-to-date survey). *Vincular patterns*, also called *generalized patterns*, resemble (classical) patterns with the additional constraint that some of the elements in a occurrence must be consecutive in positions. Of particular importance in our context, Bruner and Lackner [8] proved that deciding whether a vincular pattern σ of size k occurs in a longer permutation π is $W[1]$-complete for parameter k; for an up-to-date survey of the $W[1]$ class and related material, see [10]. *Bivincular patterns* generalize classical patterns even further than vincular patterns by adding a constraint on values.

We focus in this paper on pattern matching issues for *wedge permutations* (*i.e.*, those permutations that avoid both 213 and 231). This paper is organized as follows. In Section 2 the needed definitions are presented. Section 3 is devoted to presenting an online linear-time algorithm in case both permutations are wedge permutations, whereas Section 4 focuses on the case where only the pattern is a wedge permutation. In Section 5 we give a polynomial-time algorithm for a bivincular wedge permutation pattern. In Section 6 we consider the problem of finding the longest wedge permutation pattern in permutations.

2 Definitions

A *permutation* of size n is a linear ordering of an ordered set of size n. When writing the permutation we omit the $<$ sign, thus writing the permutation as a word: $\pi = \pi_1\pi_2 \dots \pi_n$, whose elements are distinct and usually consist of the integers $1, 2, \dots, n$.

We need to consider both the natural order of the set and the linear order given by the permutation. When talking about an element, we refer to the relative position in the natural order of the set as its value and we refer to the relative position in the order given by the permutation as its position. We write $\pi[i]$ to indicate the value of the element at position i (*i.e.* π_i). For example, in the permutation $\pi = 51342$, The element 1 is at position 2 and the element at position 1 has for value 5. Conveniently, we let $\pi[i : j]$ stand for $\pi_i\pi_{i+1} \dots \pi_j$, $\pi[: j]$ stand for $\pi[1 : j]$ and $\pi[i :]$ stand for $\pi[i : n]$.

It is convenient to use a geometric representation of permutations to ease the understanding of algorithms. The geometric representation corresponds to the set of points with coordinate $(i, \pi[i])$ (see Figure 1).

When an element $\pi[\alpha]$ is smaller (resp. larger) than an element $\pi[\beta]$ by value (in the natural order of the set), we say that $\pi[\alpha]$ is below (resp. above) $\pi[\beta]$, as a reference to the geometric representation of the permutation. Besides, the element with the largest value is called the topmost element and the element with the smallest value is called the bottommost element. For example, in the permutation $\pi = 51342$, 2 is below 4, the topmost element is 5 and the bottommost element is 1.

When an element $\pi[\alpha]$ is smaller (resp. larger) than an element $\pi[\beta]$ by position (in the order given by the permutation), we say that $\pi[\alpha]$ is on the left of (resp. on the right of) $\pi[\beta]$. Besides, the element with the largest position is called the rightmost element and the element with the smallest position is called the leftmost element. For example, in the permutation $\pi = 51342$, 5 is on the left of 3, the leftmost element is 5 and the rightmost element is 2.

A permutation σ is said to *occur* in the permutation π, written $\sigma \preceq \pi$, if there exists a subsequence of (not necessarily consecutive) elements of π that has the same relative order as σ. Such subsequence is called an *occurrence* of σ in π. Otherwise, π is said to *avoid* the permutation σ. For example, the permutation $\pi = 391867452$ has an occurrence of the pattern $\sigma = 51342$, as can be seen in the highlighted subsequence of $\pi = \mathbf{391867}\mathbf{4}52$ (or $\pi = \mathbf{3918674}\mathbf{5}\mathbf{2}$ or $\pi = \mathbf{3918674}5\mathbf{2}$ or $\pi = \mathbf{391}8674\mathbf{52}$). Since the permutation $\pi = 391867452$ contains no increasing subsequence of size four, π avoids 1234.

For an occurrence s of σ in π, we say that $\pi[j]$ correspond $\sigma[i]$ in the occurrence or that $\sigma[i]$ is matched to $\pi[j]$ in the occurrence if and only if $\pi[j] = s[i]$. Geometrically, π has an occurrence of σ if there exists a set of points in π whose relative positions are the same as those of σ (see Figure 1).

A *right-to-left maximum* (RLMax, for short) of π is an element that does not have any element above it and to its right (see Figure 2). Formally, $\pi[i]$ is a RLMax if and only if $\pi[i]$ is topmost element of $\pi[i :]$. Similarly $\pi[i]$ is a

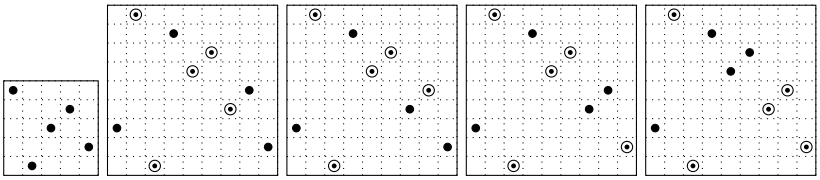

Fig. 1: The pattern $\sigma = 51342$ and four occurrences of σ in 391867452.

right-to-left minimum (RLMin, for short) if and only if $\pi[i]$ is the bottommost element of $\pi[i:]$.

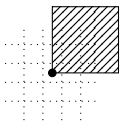

Fig. 2: The element is a RLMax if and only if the hashed area is empty.

A *bivincular permutation pattern* $\widetilde{\sigma}$ of size k is a permutation written in two-line notation (the top row is $12 \ldots k$ and the bottom row is a permutation $\sigma_1\sigma_2 \ldots \sigma_k$) with overlined elements in the top row and underlined elements in the bottom row. We have the following conditions on the top and bottom rows of σ, as seen in [15] in Definition 1.4.1:

- If the bottom line of $\widetilde{\sigma}$ contains $\underline{\sigma_i\sigma_{i+1} \ldots \sigma_j}$ then the elements corresponding to $\sigma_i\sigma_{i+1} \ldots \sigma_j$ in a occurrence of σ in π must be adjacent, whereas there is no adjacency condition for non-underlined consecutive elements. Moreover if the bottom row of $\widetilde{\sigma}$ begins with ${}_{\llcorner}\sigma_1$ then any occurrence of $\widetilde{\sigma}$ in a permutation π must begin with the leftmost element of π, and if the bottom row of σ ends with $\sigma_{k\lrcorner}$ then any occurrence of $\widetilde{\sigma}$ in a permutation π must end with the rightmost element of π.
- If the top line of $\widetilde{\sigma}$ contains $\overline{i\, i+1 \ldots j}$ then the elements corresponding to $i, i+1, \ldots, j$ in an occurrence of σ in π must be adjacent in values, whereas there is no value adjacency restriction for non-overlined elements. Moreover, if the top row of $\widetilde{\sigma}$ begins with ${}^{\ulcorner}1$ then any occurrence of $\widetilde{\sigma}$ in a permutation π must contain the bottommost element of π, and if top row of σ ends with $k^{\urcorner}$ then any occurrence of $\widetilde{\sigma}$ in a permutation π must contain the topmost element of π.

Given a bivincular permutation pattern $\widetilde{\sigma}$, σ refers to the bottom line of $\widetilde{\sigma}$ without any element underlined. For example, let $\widetilde{\sigma} = {}^{1\overline{234}^{\urcorner}}_{\llcorner 21\underline{43}}$, in 3217845, **32**17**84**5 is an occurrence of $\widetilde{\sigma}$ but 3**217**84**5** is not and $\sigma = \overline{2143}$. The best general reference for bivincular pattern is [15].

Geometrically, we represent underlined and overlined elements by *forbidden areas*. If two elements are overlined then we draw a horizontal forbidden area

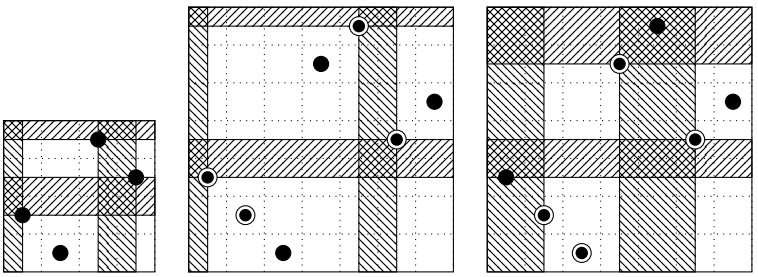

Fig. 3: From left to right, the bivincular pattern $\widetilde{\sigma} = \begin{smallmatrix} 1\overline{234}\urcorner \\ \llcorner \underline{21}\underline{43} \end{smallmatrix}$, an occurrence of $\widetilde{\sigma}$ in 3216745, an occurrence of σ in 3216745 but not an occurrence of $\widetilde{\sigma}$ because the points $(1, 3)$ and $(5, 7)$ are in the forbidden areas.

between those two points. In an occurrence, we also draw the forbidden area between the matching of two overlined elements. This area must be empty to have a valid occurrence of the bivincular pattern. If the area is empty then there is no element between them when reading the permutation from bottom to top, in other words the matching elements are consecutive in value. If two elements are underlined then we draw a vertical forbidden area. (See Figure 3).

3 Both π and σ are wedge permutation

This section is devoted to present a fast algorithm for deciding if $\sigma \preceq \pi$ in case both π and σ are wedge permutations. We begin with an easy and folk but crucial structure lemma.

Lemma 1. *The first element of any wedge permutations must be either the bottommost or the topmost element.*

Proof. Any other initial element would serve as a '2' in either a 231 or 213 with 1 and n as the '1' and '3' resp.. □

Corollary 1. *π is a wedge permutation of size n if and only if for $1 \leq i \leq n$, $\pi[i]$ is a RLMax or a RLMin.*

As a consequence, a wedge permutation can be partitioned into three sets: the set of RLMax elements, the set of RLMin elements and the rightmost element which can be both a RLMax element and a RLMin element. The partition gives a bijection between the set of wedge permutations of size n with elements $1, \ldots, n$ and the set of binary words of size $n - 1$. The word w which corresponds to π is the word where each letter at position i represents whether $\pi[i]$ is a RLMax element or a RLMin element. We call this bijection B.

As a consequence, every element of a wedge permutation can be partitioned into three sets: the set of RLMax elements, the set of RLMin elements and the rightmost element which can be both a RLMax element and a RLMin element.

Remark 1. The set of RLMax elements is a decreasing subsequence, indeed each element is left above the next RLMax element by definition of a RLMax element. In the same fashion, the set of RLMin elements is an increasing subsequence. See Figure 4.

Remark 2. We can draw a horizontally line (passing thought the rightmost element) on a wedge permutations such that the upper half contains only a decreasing subsequence and the lower half contains only an increasing subsequence. This shape the permutation as a $>$, hence the name of wedge permutations. See Figure 4.

For any permutation, to figure out whether or not an element is a RLMax element or a RLMin element, one has to run the whole permutation from left to right starting from the position of the element, we give a corollary of lemma 1, to guest it in constant time for a wedge permutation.

Corollary 2. *Let π be a wedge permutation and $1 \leq i < n$. Then,*

1. *$\pi[i]$ is a RLMin element if and only if $\pi[i] < \pi[i+1]$*
2. *$\pi[i]$ is a RLMax element if and only if $\pi[i] > \pi[i+1]$*

Now that we had highlight some structural point of a wedge permutation, we can use this structure to solve the permutation pattern matching. More precisely, we use the bijection on binary word to solve this problem, thank to the following lemma.

Lemma 2. *Let π and σ be two wedge permutations. Then, π has an occurrence of σ if and only if there exists a subsequence t of π such $B(t) = B(\sigma)$.*

Proof. The forward direction is obvious. We prove the backward direction by induction on the size of σ: if $B(t) = B(\sigma)$ then t is an occurrence of σ. The base case is a pattern of size 2. Suppose that $\sigma = 12$ and thus $B(\sigma) = \text{RLMin}$. Let $t = \pi_{i_1}\pi_{i_2}$, $i_1 < i_2$, be a subsequence of π such that $B(t) = \text{RLMin}$, this reduces to saying that $\pi_{i_1} < \pi_{i_2}$, and hence that t is an occurrence of $\sigma = 12$ in π. A similar argument shows that the lemma holds for $\sigma = 21$. Now, assume that the lemma is true for all patterns up to size $k \geq 2$. Let σ be a wedge permutation of size $k+1$ and let t be a subsequence of π of size $k+1$ such that $B(t) = B(\sigma)$. As $B(t)[2:] = B(\sigma)[2:]$ by the induction hypothesis, it follows that $t[2:]$ is an occurrence of $\sigma[2:]$. Moreover $B(t)[1] = B(\sigma)[1]$ thus $t[1]$ and $\sigma[1]$ are both either the bottommost or the topmost element of their respective sequences. Therefore, t is an occurrence of σ in π. □

We are now ready to solve the permutation pattern matching problem in case both π and σ are wedge permutations.

Proposition 1. *Let π and σ be two wedge permutations. One can decide whether π has an occurrence of σ in linear time.*

Proof. According to Lemma 1 the problem reduces to deciding whether $B(\sigma)$ occurs as a subsequence in $B(\pi)$. This can be solved in linear time with a straightforward greedy approach as follows. We read both $B(\sigma)$ and $B(\pi)$ from left to right; when two letters are equal, we match them, and move to the next in both $B(\sigma)$ and $B(\pi)$ (if such letters exist); otherwise we stay at the same letter of $B(\sigma)$ but move to the next one in $B(\pi)$ (if it exists); we accept exactly when all letters of B(sigma) have been matched. □

Thanks to Corollary 2, we do not need to compute the words $B(\sigma)$ and $B(\pi)$ before running the greedy algorithm. This computation can be done at the same time as running the algorithm, thus, giving an on-line algorithm.

4 Only σ is a wedge permutation

This section focuses on the permutation pattern matching problem in case only the pattern σ is a wedge permutation. We need to consider a specific decomposition of σ into *factors*: we split the permutation into maximal sequences of consecutive RLMin and RLMax elements, respectively called a RLMin factor and a RLMax factor. This corresponds to splitting the permutation between every pair of RLMin-RLMax and RLMax-RLMin elements (see Figure 4). For the special case of a wedge permutation, this also corresponds to split the permutation into maximal sequences of elements consecutive in value. We label the factors from right to left. Note that the rightmost element can be both a RLMin or a RLMax, we take the convention that it is as the same type as the element before it for the factorization, such that the first factor (the rightmost factor) always contains at least two elements. For example, $\sigma = 123984765$ is split as $123-98-4-765$. Hence $\sigma = \text{factor}(4)\ \text{factor}(3)\ \text{factor}(2)\ \text{factor}(1)$ with $\text{factor}(4) = 123$, $\text{factor}(3) = 98$, $\text{factor}(2) = 4$ and $\text{factor}(1) = 765$.

Remark 3. A factor is either an increasing or a decreasing sequence of elements.

To lighten a figure of an occurrence and to help the comprehension, we represent an occurrence in π (or a part of it) by any rectangle $((A_x, A_y), (B_x, B_y))$ that contains at least all the points of the elements of the occurrence, where A is the bottom left corner and B is the top right corner. Formally the rectangle $((A_x, A_y), (B_x, B_y))$ is the set of points in $\pi[A_x : B_x]$ where each element is in $[A_y, B_y]$. Especially we say that a rectangle is minimal if and only if it is the smallest rectangle that contains all of the elements of the occurrence. Moreover, the minimal rectangle of an occurrence is $((lm, bm), (rm, tm))$ where lm stands for the position of the leftmost element, bm stands for the value of the bottommost element, rm stands for the position of the rightmost element and tm stands for the value of the topmost element. See Figure 5.

Remark 4. A rectangle contains an occurrence of a RLMin (resp. RLMax) factor if and only if the rectangle contains an increasing (resp. decreasing) subsequence of same size or larger than the size of the factor.

We give a property of a wedge permutation that allows us to decide whether a rectangle contains an occurrence of the wedge permutation: given a rectangle we show that we can spilt the rectangle in two smaller rectangles. The splitting transforms the problem in deciding whether or not each of the two rectangles contains part of the wedge permutation. Intuitively, the problems on the two rectangles are easier than the problem on the original rectangle.

Lemma 3. *There exists an occurrence of a wedge permutation starting with a leftmost RLmin (resp. RLmax) factor if and only if there exists two rectangles R_1 and R_2, such that R_1 is left below (resp. left above) R_2, R_1 contains an occurrence of the leftmost factor and R_2 contains an occurrence of the rest.*

Proof. This is true as the leftmost factor is left bellow (resp. left above) the rest of the wedge permutation.

We show how to compute the permutation pattern matching using this lemma. Note that the algorithm given in the proof is not the best we can do in respect to the time and space complexity, but it helps to understand the best version.

Lemma 4. *Let σ be a wedge permutation of size k and π a permutation of size n. One can decide in polynomial time and space whether π contains an occurrence σ.*

Proof. We introduce a set of values needed in computing the permutation patten matching. Let $LIS_\pi(((j, lb), (j', up)))$ (resp. $LDS_\pi(((j, lb), (j', up)))$) be the size of the longest increasing (resp. decreasing) subsequence in $((j, lb), (j', up))$. For all the rectangles $LIS_\pi(((j, lb), (j', up)))$ and $LDS_\pi(((j, lb), (j', up)))$ are computed in $O(n^2 \log(\log(n)))$ time (see [4]). $LIS_\pi(((j, lb), (j', up)))$ (resp. LDS_π $(((j, lb), (j', up))))$ allow us to decide the existence of an occurrence of any LRMin (resp. LRMax) factor in the rectangle $((j, lb), (j', ub))$ in constant time if we precomputed it for all the rectangles.

We solve the problem of deciding whether a rectangle $((j, lb), (n, ub))$ contains an occurrence of $\text{factor}(i) \dots \text{factor}(i')$, More formally:

$$PM_\sigma^\pi(i, i', ((j, lb), (n, ub))) = \begin{cases} True & \text{if and only if } ((j, lb), (n, ub)) \text{ contains} \\ & \text{an occurrence of } \text{factor}(i) \dots \text{factor}(i') \\ False & \text{Otherwise} \end{cases}$$

By definition π contains an occurrence of σ if and only if $PM_\sigma^\pi(\ell, 1, ((1, 1), (n, n)))$ is true where ℓ is the number of factors in σ. We show how to compute $PM_\sigma^\pi(i, i', ((j, lb), (n, up)))$ recursively.

- If the permutation is reduced to a factor then one has to decide if the rectangle contains an increasing or a decreasing factor with the same size or larger than the size of the unique factor. This case happens when $i = i'$:

- if $factor(i)$ is a LRMin factor:

$$PM_\sigma^\pi(i,i,((j,lb),(n,up))) =$$

$$\begin{cases} True & |factor(i)| \leq LIS_\pi(((j,lb),(n,up))) \\ False & \text{Otherwise} \end{cases}$$

- if $factor(i)$ is a LRMin factor:

$$PM_\sigma^\pi(i,i,((j,lb),(n,up))) =$$

$$\begin{cases} True & |factor(i)| \leq LDS_\pi(((j,lb),(n,up))) \\ False & \text{Otherwise} \end{cases}$$

– Otherwise, assuming that factor(i) is a LRMin (resp. LRMax) factor, we need to decide if there exists a spitting of $((j,lb),(n,up))$ into two rectangle R_1 and R_2 such that R_1 is left below (resp above) R_2, R_1 contains an occurrence of factor(i) and R_2 contains an occurrence of factor($i-1$)...factor(i'). A direct strategy is to try every pair of rectangles where one rectangle is left below (resp. above) the other and to decide whether the first one contains a occurrence of factor(i) and the second one contains an occurrence of factor($i-1$)...factor(i'). Note that to reduce the number of pair of rectangles to test, it is better to consider the biggest rectangles possible as long as the positions of the two rectangles are respected. Indeed if R' is contained in R and R' contains an occurrence then R also contains this occurrence, so we do not need to find an occurrence in both R and R' but only in R. Especially we can always consider that R_1 has for left bottom corner (j,lb) (resp. right top corner (j,up)) and that R_2 has for right top corner (n,up) (resp. right bottom corner (n,lb)) and R_1 is next to R_2. More formally:
  - if $factor(i)$ is a LRMin factor:

$$PM_\sigma^\pi(i,i',((j,lb),(n,up))) =$$

$$\bigvee_{\substack{j \leq j' < n \\ up' < up}} PM_\sigma^\pi(i,i,((j,lb),(j',up'))) \wedge PM_\sigma^\pi(i-1,i',((j'+1,up'+1),(n,up)))$$

  - if $factor(i)$ is a LRMax factor:

$$PM_\sigma^\pi(i,i',((j,lb),(n,up))) =$$

$$\bigvee_{\substack{j \leq j' < n \\ lb' > lb}} PM_\sigma^\pi(i,i,((j,lb'),(j',up))) \wedge PM_\sigma^\pi(i-1,i',((j'+1,lb),(n,lb'-1)))$$

Remark that by modifying the definition of $PM_\sigma^\pi(i,i',((j,lb),(n,up)))$ we can reduce the computational time of the algorithm. Instead of asking for any occurrence in $((j,lb),(n,up)))$, we ask for the occurrences that start at $\pi[j]$. As any element in the pattern is a RLMax or a RLMin, the value of $\pi[j]$ can be used has a top or a bottom edge of a rectangle. So we can remove one argument from the problem, reducing the maximal number of different problems that we have to compute. □

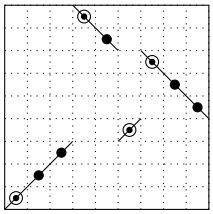

Fig. 4: The wedge permutation 123984765. Every line represents a factor, every circled point represents the leftmost element of each factor.

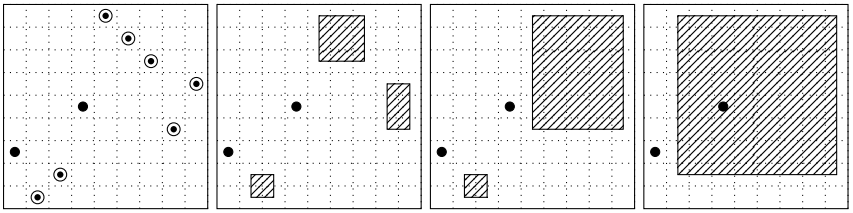

Fig. 5: From left ro right, an occurrence of a permutation the same occurrence where each occurrence of a factor is represented by the minimal rectangle, the same occurrence where the occurrence of the first two factor are represented by the minimal rectangle and the same occurrence represented by its minimal rectangle.

We extend our research to reduce the number of pairs of rectangles that we have to test. Informally, given that the leftmost factor is a RLMin (resp. RLMax) factor and given that the left top (resp. bottom left) corner of the second rectangle is fixed and that the second rectangle contains the occurrence of the rest of the permutation, it is in the interest of finding an occurrence in the whole rectangle that the bottom (resp. top) edge of the second rectangle is the highest (resp. lowest) possible. Indeed, the first rectangle will have a highest top (resp. a lowest bottom) edge so it will contains more elements which is better to find an occurrence. The next lemma indicates which element is the topmost and the bottommost, and the next one and its corollary formalise what said above. But first we introduce another notation.

We introduce the notation LMEp(s) (which stands for the left most element position): Suppose that s is a subsequence of S, LMEp(s) is the position of the leftmost element of s in S. Thus for every factor, LMEp(factor(j)) stands for the position in σ of the leftmost element of factor(j). From the above example, LMEp(factor(4)) = 1, LMEp(factor(3)) = 4, LMEp(factor(2)) = 6 and LMEp(factor(1)) = 7.

Lemma 5. *Given a wedge permutation σ, if* factor(i) *is a RLMin (resp. RLMax) factor the topmost (resp. bottommost) element of* factor(i)...factor(1) *is the leftmost element of* factor($i-1$).

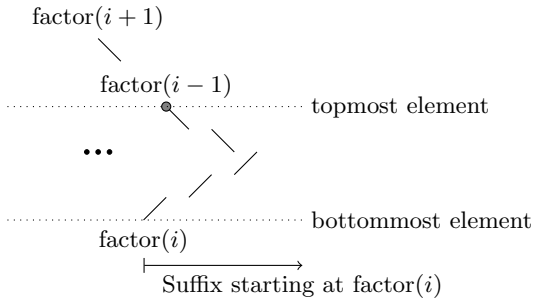

Fig. 6: The topmost element of the suffix starting at factor(i) is the leftmost element of factor($i-1$) (represented by the grey dot). $\mathrm{LM}^{\pi}_{\sigma}(i,j)$ is the smallest value of the matching of the topmost element (the grey dot) in all the occurrences of the suffix starting at LMEp(factor(i)) in $\pi[j:]$.

Proof. This corollary states that, given a wedge permutation if the permutation starts with a RLMin (resp. RLMax) element then the topmost (resp. bottommost) element of this permutation is the first RLMax (resp. RLMin) element (see Figure 6). This is easy to see from the shape of a wedge permutation. Formally, the RLMin elements are below the RLMax element, thus the topmost element must be the first RLMax element, which is the leftmost element of factor($i-1$). □

$\mathrm{S}^{\pi}_{\sigma}(i,j)$ as the set of all subsequences s of $\pi[j:]$ that start at $\pi[j]$ and that are occurrences of factor(i)...factor(1).

Lemma 6. *Let σ be a wedge permutation and* factor(i) *be an RLMin (resp. RLMax) factor σ. Let π be a permutation and s a subsequence of π such that $s \in \mathrm{S}^{\pi}_{\sigma}(i,j)$ and s minimizes (resp. maximizes) the matching of the leftmost element of* factor($i-1$). *Let s' be a subsequence of π such that $s \in \mathrm{S}^{\pi}_{\sigma}(i,j)$ and let $t = t's'$ be a subsequence of π that extends s' on the right. Assume t is an occurrence of* factor($i+1$)...factor(1) *such that the leftmost element of* factor(i) *is matched to $\pi[j]$. Then the subsequence $t's$ is also an occurrence of* factor($i+1$)...factor(1) *such that the leftmost element of* factor(i) *is matched to $\pi[j]$.*

Informally, this lemma states that we can replace the rectangle of an occurrence in $\mathrm{S}^{\pi}_{\sigma}(i,j)$ by another rectangle of an occurrence in $\mathrm{S}^{\pi}_{\sigma}(i,j)$ such that the second rectangle shares the same edges as the first one, except for the top edge which is lower. Especially the second rectangle can be chosen as the one with the lowest top edge in all the rectangles sharing the right, bottom and left edges. More formally, given any occurrence of factor($i+1$) factor(i) ... factor(1), where factor(i) is a RLMin (resp. RLMax) factor, we can replace the part of the occurrence where factor(i) ... factor(1) occurs, by any occurrence that minimises (resp. maximises) the leftmost element of factor($i-1$). Indeed the leftmost ele-

ment of factor$(i-1)$ is the topmost (resp. bottommost) element of factor(i) ... factor(1) (see Figure 6).

Proof. Let us consider the case where factor(i) is a RLMin factor. By definition s is an occurrence of σ[LMEp(factor(i)) :]. First of all, remark that t' is an occurrence of factor$(i+1)$. So to prove that $t's$ is an occurrence of factor$(i+1)\dots$factor(1) we need to prove that the elements of t' are above the elements of s. Since $t's'$ is an occurrence of factor$(i+1)\dots$factor(1) it follows that the elements of t' are above the elements of s'. Moreover the topmost element of s is below (or equal to) the topmost element of s' thus the elements of s are below the elements of t'. We use a symmetric argument if factor(i) is a RLMax factor. □

Corollary 3. *Let σ be a permutation,* factor(i) *be a RLMin (resp. RLMax) factor and s be a subsequence of π such that $s \in \mathrm{S}^{\pi}_{\sigma}(i,j)$ and that minimizes (resp. maximizes) the matching of the leftmost element of* factor$(i-1)$*. The following statements are equivalent:*

- *There exists an occurrence of σ in π where the leftmost element of* factor(i) *is matched to $\pi[j]$.*
- *There exists an occurrence t of* factor$(\ell)\dots$factor$(i+1)$ *in $\pi[:j-1]$ such that ts is an occurrence of σ in π with the leftmost element of* factor(i) *is matched to $\pi[j]$.*

Proof. This corollary takes a step further from the previous one, as it states that if there is no occurrence of σ in π where the leftmost element of factor(i) is matched to $\pi[j]$ and such that the leftmost element of factor$(i-1)$ is minimized (resp. maximized) then there does not exist any occurrence at all. The backward direction is trivial and the forward direction follows the Lemma 6. □

Proposition 2. *Let σ be a wedge permutation of size k and π a permutation of size n. One can decide in $O(max(kn^2, n^2\log(\log(n)))$ time and $O(n^3)$ space if π contains an occurrence σ.*

Proof. The algorithm follows the previous one, but instead of return true or false, assuming that factor(i) is a RLMin (resp. RLMax) factor the algorithm return the value of the top (resp. bottom) edge. So that once we compute the problem for R_2 we can use this value has the top (resp. bottom) edge for the rectangle R_1.

We first introduce a set of values needed in the problems. Let $LIS_\pi(j, j', bound)$ (resp. $LDS_\pi(j, j', bound)$) be the longest increasing (resp. decreasing) subsequence in $\pi[j:j']$ starting at $\pi[j]$, with every element of this subsequence being smaller (resp. larger) than *bound*. LIS_π and LDS_π can be computed in $O(n^2\log(\log(n)))$ time (see [4]). $LIS_\pi(j, j', bound)$ (resp. $LDS_\pi(j, j', bound)$) allow us to decide the existence of an occurrence of any LRMin (resp. LRMax) factor in the rectangle $((j, \pi[j]), (j', bound))$ (resp. $((j, bound), (j', \pi[j]))$.

Given a RLMin (resp. RLMax) factor factor(i) of σ and a position j in π, we want the value of the top (resp. bottom) edge of any rectangle with

left bottom (resp. with left top) corner $(j, \pi[j])$ containing an occurrence of factor$(i)\dots$factor(1) starting at $\pi[j]$ and which minimizes (resp. maximizes) its top (resp. bottom) edge or a value which indicate that no occurrence exists in this rectangle. More formally:

– If factor(i) is RLMin factor.

$$PM_\sigma^\pi(i,j) =$$

$$\begin{cases} \text{The top edge of any rectangle with left bottom corner } (j,\pi[j]) \\ \text{with right edge } n \text{ containing an occurrence of} \\ \text{factor}(i)\dots\text{factor}(1) \text{ starting at } \pi[j] \text{ which minimize the top edge} \\ \\ \text{Or } \infty \text{ if no occurrence exists} \end{cases}$$

– If factor(i) is RLMax factor.

$$PM_\sigma^\pi(i,j) =$$

$$\begin{cases} \text{The bottom edge of any rectangle with left top corner } (j,\pi[j]) \\ \text{with right edge } n \text{ containing an occurrence of} \\ \text{factor}(i)\dots\text{factor}(1) \text{ starting at } \pi[j] \text{ which maximize its bottom edge} \\ \\ \text{Or } 0 \text{ if no occurrence exists} \end{cases}$$

By definition, there exists an occurrence of σ in π if and only if there exists a $1 \leq j \leq n$ such that $PM_\sigma^\pi(\ell, j) \neq 0$ and $PM_\sigma^\pi(1, \ell, j) \neq \infty$ with ℓ the number of factors in σ.

We show how to compute recursively those values.

– If the permutation is reduced to a RLMin (resp. RLMax) factor then one has to decide the top (resp. lower) edge of a rectangle that contains an increasing or a decreasing factor with the same size of larger than the size of the unique factor starting at $\pi[j]$. This case happens when $i = 1$:
  - If factor(i) is RLMin factor.

$$PM_\sigma^\pi(1,j) = \min\{\pi[j'] \mid \text{if } |factor(1)| \leq LIS_\pi(j, j', \pi[j'])\}_{j' \geq j} \cup \{\infty\}$$

  - If factor(i) is RLMax factor.

$$PM_\sigma^\pi(1,j) = \min\{\pi[j'] \mid \text{if } |factor(1)| \leq LDS_\pi(j, j', \pi[j'])\}_{j' \geq j} \cup \{0\}$$

– Otherwise, if factor(i) is a RLMin (resp. RLMax) factor, we split the rectangle into two rectangles R_1 and R_2 such that R_1 is left below R_2, R_1 contains an occurrence of factor(i) starting at j and R_2 contains an occurrence of factor$(i-1)\dots factor(1)$ starting at $j' > j$. As said before, we only need to test the pair of rectangles where the rectangle R_2 has

for left edge j' and has the highest bottom (resp. lowest top) edge. So for R_2 we want a rectangle starting at j' which contains an occurrence of factor$(i-1)\dots factor(i')$ starting at j' with maximize the bottom (resp. minimize the top) edge. Moreover we also want to know the value of the bottom (resp. top) edge to use it as a bound to R_1 so that that R_1 is below R_2, in other words we want $PM_\sigma^\pi(i-1,j')$. As before R_1 has to be biggest possible and to ensure that R_1 is left below (resp. above) R_2, R_1 must have for top right corner $(j'-1, PM_\sigma^\pi(i-1,j')-1)$ (resp. bottom right corner $(j'-1, PM_\sigma^\pi(i-1,j')+1)$). Finally in all the "correct" pairs of rectangles, we need the value of the top (resp. bottom) edge of a rectangle minimizing the top edge (resp. maximizing the bottom edge) containing the pair of rectangles. Note that the top (resp. bottom) edge has for value $\pi[j']$ (the matching of LMEp(factor$(i-1)$)). Formally:

- If factor(i) is RLMin factor:

$$PM_\sigma^\pi(i,j) =$$

$\min\{\infty\} \cup \{\pi[j'] \mid b = PM_\sigma^\pi(i-1,j')$ is not ∞ and $|factor(i)| \leq LIS_\pi(j, j'-1, b-1)\}_{j'<j}$

- If factor(i) is RLMax factor:

$$PM_\sigma^\pi(i,j) =$$

$\min\{\infty\} \cup \{\pi[j'] \mid b = PM_\sigma^\pi(i-1,j')$ is not ∞ and $|factor(i)| \leq LDS_\pi(j, j'-1, b+1)\}_{j'<j}$

The number of factors is bounded by k. Every instance of LIS_π and LDS_π can be computed in $O(n^2 \log(\log(n)))$ (see [4]) and takes $O(n^3)$ space. There are n base cases that can be computed in $O(n)$ time, thus computing every base case takes $O(n^2)$ time. There are kn different instances of AF and each one of them takes $O(n)$ time to compute, thus computing every instance of AF takes $O(kn^2)$ time. There are kn different instances of LM and each one of them takes $O(n)$, thus computing every LM takes $O(kn^2)$ time. Thus computing all the values takes $O(max(kn^2, n^2 \log(\log(n))))$ time. Every value takes $O(1)$ space, thus the problem takes $O(kn^2)$ space but LIS_π and LDS_π takes $O(n^3)$ space. □

5 Bivincular wedge permutation patterns

This section is devoted to the pattern matching problem with bivincular wedge permutation pattern. Recall that a bivincular pattern generalises a permutation pattern by being able to force elements in an occurrence to be consecutive in value or/and in position. Intuitively we cannot use the previous algorithm, as the restrictions on position and value are not managed.

Given a bivincular permutation pattern $\widetilde{\sigma}$, we let $\widetilde{\sigma}[i:]$ refers to the bivincular permutation pattern which has for bottom line the bottom line of $\widetilde{\sigma}$ where we remove the elements before i and has for top line the elements of $\sigma[i:]$ where m

and $m+1$ are overlined if and only if m and $m+1$ are in $\sigma[i:]$, and m and $m+1$ are overlined in $\widetilde{\sigma}$. For example given $\widetilde{\sigma} = {}^{1\,2\overline{3\,4}\urcorner}_{\llcorner 2\underline{1\,4\,3}}$, $\sigma = 2143$ and $\widetilde{\sigma}[2:] = {}^{2\overline{3\,4}\urcorner}_{\underline{1\,4\,3}}$.

We describe other structures properties of wedge permutations needed to solve the problem.

Lemma 7. *Let σ be a wedge permutation.*

- *If $\sigma[i]$ is a RLMin element and is not the rightmost element, then $\sigma[i]+1$ is to the right of $\sigma[i]$.*
- *If $\sigma[i]$ is a RLMax element and is not the rightmost element, then $\sigma[i]-1$ is to the right of $\sigma[i]$.*

Proof. For the first point, by contradiction, $\sigma[i]+1$ would serve as a '2', $\sigma[i]$ would serve as a '1' and $\sigma[i+1]$ would serve as a '3' in an occurrence of 213. For the second point, by contradiction, $\sigma[i]-1$ would serve as a '2', $\sigma[i]$ would serve as a '3' and $\sigma[i+1]$ would serve as a '1' in an occurrence of 231. □

Lemma 8. *Let σ be a wedge permutation.*

- *If m and $m+1$ are both RLMin elements then any element between m and $m+1$ (if any) is a RLMax element.*
- *If m and $m-1$ are both RLmax elements then any element between m and $m-1$ (if any) is a RLMin element.*

Proof. For the first point, by contradiction, let $\sigma[\alpha] = m$ and $\sigma[\beta] = m+1$ and suppose that there exists $\alpha < \gamma < \beta$, such that $\sigma[\gamma]$ is a RLMin element. We know that RLMin are strictly increasing so $\sigma[\alpha] < \sigma[\gamma] < \sigma[\beta]$, which contradict the fact that $\sigma[\beta] = \sigma[\alpha]+1$. □

Proposition 3. *Let $\widetilde{\sigma}$ be a bivincular wedge permutation pattern of size k and π a permutation of size n. One can decide in $O(kn^4)$ time and $O(kn^3)$ space if π contains an occurrence $\widetilde{\sigma}$.*

Proof. Consider the following problem: Given a lower bound lb, a upper bound ub, a position i of σ and a position j of π, we want to know if there exists an occurrence of $\widetilde{\sigma}[i:]$ in $\pi[j:]$ with every element of the occurrence is in $[\mathrm{lb}, \mathrm{ub}]$ and starting at $\pi[j]$, In other words given the rectangle with bottom left edge (j, lb) and with top right edge (n, ub) contains an occurrence of $\widetilde{\sigma}[i:]$ starting at $\pi[j]$, More formally:

$$PM_{\pi}^{\widetilde{\sigma}}(\mathrm{lb}, \mathrm{ub}, i, j) =$$

$$\begin{cases} true & \text{if } \pi[j:] \text{ has an occurrence of the bivincular pattern } \widetilde{\sigma}[i:] \\ & \text{with every element of the occurrence in } [\mathrm{lb}, \mathrm{ub}] \\ & \text{and starting at } \pi[j] \\ & \text{and if } \overline{\sigma[j](\sigma[j]+1)} \text{ or } \sigma[j]^{\urcorner} \text{ appear in } \widetilde{\sigma} \text{ then } \pi[j] = \mathrm{ub} \\ & \text{and if } \overline{(\sigma[j]-1)\sigma[j]} \text{ or } {}^{\ulcorner}\pi[j] \text{ appear in } \widetilde{\sigma} \text{ then } \sigma[j] = \mathrm{lb} \\ false & \text{otherwise} \end{cases}$$

Clearly if $\llcorner\sigma[1]$ does not appear in $\widetilde{\sigma}$ then π contains an occurrence of $\widetilde{\sigma}$ if and only if $\bigcup_{0<j} PM_{\pi}^{\widetilde{\sigma}}(1, n, 1, j)$ is true. If $\llcorner\sigma[1]$ appear in σ then π contains an occurrence of the bivincular wedge permutation pattern σ if and only if $PM_{\pi}^{\widetilde{\sigma}}(1, n, 1, 1)$.

We show how to compute recursively those values. Informally, to find an occurrence of $\widetilde{\sigma}[i:]$ in $\pi[j:]$, given that $\sigma[i]$ is a RLMin element, we need to find a rectangle R in π such that R contains an occurrence of $\widetilde{\sigma}[i+1:]$ and R is right above $\pi[j]$. Moreover

- If $\underline{\sigma[i]\sigma[i+1]}$ then we require that $\pi[j]$ and R are next to each other horizontally and that the occurrence in R starts at the left edge of R. In other words R has for left edge $\pi[j]+1$.
- If $\overline{\sigma[i](\sigma[i]+1)}$ then we require that $\pi[j]$ and R are next to each other vertically and that the minimal (resp. maximal) element in the occurrence is on the bottom (resp. top) edge of R. In other words R has for bottom edge $\sigma[i]+1$.

Note that the parameters lb, ub and j corresponding to the occurrence of $\widetilde{\sigma}[i+1:]$ are respectively, the bottom edge, the top edge and the left edge of the rectangle R. The case where $\sigma[i]$ is a RLMax element can be dealt with symmetry. More formally:

BASE:

$$PM_{\pi}^{\widetilde{\sigma}}(\mathrm{lb}, \mathrm{ub}, k, j) = \begin{cases} true & \text{if } \pi[j] \in [\mathrm{lb}, \mathrm{ub}] \\ & \text{and if } \sigma[k]\rfloor \text{ appears in } \widetilde{\sigma} \text{ then } j = n \\ & \text{and if } \llcorner\sigma[k] \text{ appears in } \widetilde{\sigma} \text{ then } j = 1 \\ & \text{and if } \sigma[k]\urcorner \text{ appears in } \widetilde{\sigma} \text{ then } \pi[j] = \mathrm{ub} = n \\ & \text{and if } \ulcorner\sigma[k] \text{ appears in } \widetilde{\sigma} \text{ then } \pi[j] = \mathrm{lb} = 1 \\ & \text{and if } \overline{(\sigma[k]-1)\sigma[k]} \text{ appears in } \widetilde{\sigma} \text{ then } \pi[j] = \mathrm{lb} \\ & \text{and if } \overline{(\sigma[k]\sigma[k]+1)} \text{ appears in } \widetilde{\sigma} \text{ then } \pi[j] = \mathrm{ub} \\ false & \text{otherwise} \end{cases}$$

The base case finds an occurrence for the rightmost element of the pattern. If the rightmost element does not have any restriction on positions and on values, then $PM_{\pi}^{\widetilde{\sigma}}(\mathrm{lb}, \mathrm{ub}, k, j)$ is true if and only if $\sigma[k]$ is matched to $\pi[j]$. This is true if $\pi[j] \in [\mathrm{lb}, \mathrm{ub}]$. If $\sigma[k]\rfloor$ appears in $\widetilde{\sigma}$ then $\sigma[k]$ must be matched to the rightmost element of π thus j must be n. If $\sigma[k]\urcorner$ appears in $\widetilde{\sigma}$ then $\sigma[k]$ must be matched to the topmost element which is n. If $\ulcorner\sigma[k]$ appears in $\widetilde{\sigma}$ then $\sigma[k]$ must be matched to the bottommost element which is 1. If $\overline{(\sigma[k]-1)\sigma[k]}$ appears in $\widetilde{\sigma}$ then the matching element of $\sigma[k]$ and $\sigma[k]-1$ must be consecutive in value, by recursion the value of the element matching $\sigma[k]-1$ will be recorded in lb and by adding 1 to it thus $\sigma[k]$ must be matched to lb. If $\overline{(\sigma[k]\sigma[k]+1)}$ appears in $\widetilde{\sigma}$ then the element matching $\sigma[k]$ and $\sigma[k]+1$ must be consecutive in value, by recursion the value of the element matching $\sigma[k]+1$ will be recorded in ub and by removing 1 to it thus $\sigma[k]$ must be matched to ub.

STEP:

In the following, we use the notation $\bigcup$ to represent the disjunction. We consider 3 cases for the problem $PM_\pi^{\widetilde{\sigma}}(\mathrm{lb}, \mathrm{ub}, i, j)$:

– If $\pi[j] \notin [\mathrm{lb}, \mathrm{ub}]$ then:

$$PM_\pi^{\widetilde{\sigma}}(\mathrm{lb}, \mathrm{ub}, i, j) = false$$

which is immediate from the definition.

– If $\pi[j] \in [\mathrm{lb}, \mathrm{ub}]$ and $\sigma[i]$ is a RLMin element then:

$$PM_\pi^{\widetilde{\sigma}}(\mathrm{lb}, \mathrm{ub}, i, j) = \begin{cases} \bigcup_{\ell > j} PM_\pi^{\widetilde{\sigma}}(\pi[j]+1, \mathrm{ub}, i+1, \ell) & \text{if } \sigma[i] \text{ is not underlined} \\ & \text{and } \sigma[i] \text{ is not overlined} \\ \bigcup_{\ell > j} PM_\pi^{\widetilde{\sigma}}(\pi[j]+1, \mathrm{ub}, i+1, \ell) & \text{if } \sigma[i] \text{ is not underlined} \\ & \text{and } \overline{(\sigma[i]-1)\sigma[i]} \text{ or } \ulcorner\sigma[i] \\ & \text{appears in } \widetilde{\sigma} \\ & \text{and } \pi[j] = \mathrm{lb} \\ PM_\pi^{\widetilde{\sigma}}(\pi[j]+1, \mathrm{ub}, i+1, j+1) & \text{if } \underline{\sigma[i]\sigma[i+1]} \\ & \text{appears in } \widetilde{\sigma} \\ & \text{and } \sigma[i] \text{ is not overlined} \\ PM_\pi^{\widetilde{\sigma}}(\pi[j]+1, \mathrm{ub}, i+1, j+1) & \text{if } \underline{\sigma[i]\sigma[i+1]} \\ & \text{and } \overline{(\sigma[i]-1)\sigma[i]} \text{ or } \ulcorner\sigma[i] \\ & \text{appear in } \widetilde{\sigma} \\ & \text{and } \pi[j] = \mathrm{lb} \\ false & \text{otherwise} \end{cases}$$

Remark that $\sigma[i]$ can be matched to $\pi[j]$ because $\pi[j] \in [\mathrm{lb}, \mathrm{ub}]$. Thus if $\pi[j+1:]$ has an occurrence of $\widetilde{\sigma}[i+1:]$ with every element of the occurrence in $[\pi[j]+1, \mathrm{ub}]$ then $\pi[j:]$ has an occurrence $\widetilde{\sigma}[i:]$. To decide the latter condition, it is enough to know if there exists ℓ, $j < \ell$ such that $PM_\pi^{\widetilde{\sigma}}(\pi[j]+1, \mathrm{ub}, i+1, \ell)$ is true. The first case corresponds to an occurrence without restriction on position and on value. The second case asks for the matching of $\sigma[i]-1$ and $\sigma[i]$ to be consecutive in value, but the matching of $\sigma[i]-1$ is $lb-1$ thus we want $\pi[j] = \mathrm{lb}$. The third case asks for the matching of $\sigma[i]$ and $\sigma[i+1]$ to be consecutive in positions, thus the matching of $\sigma[i+1]$ must be $\pi[j+1]$. The fourth case is a union of the second and third case.

– If $\pi[j] \in [\mathrm{lb}, \mathrm{ub}]$ and $\sigma[i]$ is a RLMax element then:

$$PM_\pi^{\widetilde{\sigma}}(\mathrm{lb},\mathrm{ub},i,j) = \begin{cases} \bigcup_{\ell>j} PM_\pi^{\widetilde{\sigma}}(\mathrm{lb},\pi[j]-1,i+1,\ell) & \text{if } \sigma[i] \text{ is not underlined} \\ & \text{and } \sigma[i] \text{ is not overlined} \\ \bigcup_{\ell>j} PM_\pi^{\widetilde{\sigma}}(\mathrm{lb},\pi[j]-1,i+1,\ell) & \text{if } \sigma[i] \text{ is not underlined} \\ & \text{and } \overline{\sigma[i](\sigma[i]+1)} \text{ or } \sigma[i]^{\urcorner} \\ & \text{appear in } \widetilde{\sigma} \\ & \text{and } \pi[j] = \mathrm{ub} \\ PM_\pi^{\widetilde{\sigma}}(\mathrm{lb},\pi[j]-1,i+1,j+1) & \text{if } \underline{\sigma[i]\sigma[i+1]} \\ & \text{appears in } \widetilde{\sigma} \\ & \text{and } \sigma[i] \text{ is not overlined} \\ PM_\pi^{\widetilde{\sigma}}(\mathrm{lb},\pi[j]-1,i+1,j+1) & \text{if } \underline{\sigma[i]\sigma[i+1]} \\ & \text{and } \overline{\sigma[i](\sigma[i]+1)} \text{ or } \sigma[i]^{\urcorner} \\ & \text{appear in } \widetilde{\sigma} \\ & \text{and } \pi[j] = \mathrm{ub} \\ false & \text{otherwise} \end{cases}$$

The same remark as the last case holds.

Remark that there are constraints that we do not concern given in $PM_\pi^{\widetilde{\sigma}}(*,*,i,*)$. Especially in the case where $\sigma[i]$ is a RLMin the case where $\underline{\sigma[i-1]\sigma[i]}, \overline{\sigma[i](\sigma[i]+1)}$ and every combination of the constraints appear in $\widetilde{\sigma}$ are not in the conditions. $\underline{\sigma[i-1]\sigma[i]}$ is not in the conditions because it is up to $PM_\pi^{\widetilde{\sigma}}(*,*,i-1,*)$ to ensure that the elements corresponding to elements at position $i-1$ and i in an occurrence are next to each other in position. $\overline{\sigma[i](\sigma[i]+1)}$ is not in the conditions because $\sigma[i]+1$ is on the right of $\sigma[i]$ (see Lemma 7) and thus will be taken care during the calls of $PM_\pi^{\widetilde{\sigma}}(*,*,i',*)$ for some $i < i'$. In the same fashion, when $\sigma[i]$ is a RLMax $\underline{\sigma[i-1]\sigma[i]}$ do not appear in the conditions for the exact same reason and $\overline{(\sigma[i]-1)\sigma[i]}$ because $\sigma[i]-1$ is on the right of $\sigma[i]$.

Clearly the problem return true if π contains an occurrence of $\widetilde{\sigma}$ if there is no constraint on position and on value. We now discus how the position and value constraints are taken into account so that the algorithm returns true if and only if π has an occurrence of $\widetilde{\sigma}$.

Position Constraint. There are 3 types of position constraints that can be added by underlined elements.

- If ${}_{\llcorner}\sigma[1]$ appears in $\widetilde{\sigma}$ then the leftmost element of σ must be matched to the leftmost element of π ($\sigma[1]$ is matched to $\pi[1]$ in a occurrence of σ in π). This constraint is satisfied by requiring that the occurrence starts at the leftmost element of π: if $PM_\pi^{\widetilde{\sigma}}(1,n,1,1)$ is true.
- If $\sigma[k]_{\lrcorner}$ appears in $\widetilde{\sigma}$ then the rightmost element σ must be matched the rightmost element of π ($\sigma[k]$ is matched to $\pi[n]$ on a occurrence of σ in π). This constraint is checked in the base case.

– If $\overline{\sigma[i]\sigma[i+1]}$ appears in $\widetilde{\sigma}$ then the positions of the element matching $\sigma[i]$ and $\sigma[i+1]$ must be consecutive. In other words, if $\sigma[i]$ is matched to $\pi[j]$ then $\sigma[i+1]$ must be matched to $\pi[j+1]$. We ensure this restriction by recursion by requiring that the matching of $\sigma[i+1:]$ starts at position $j+1$.

Value Constraint. There are 3 types of value constraints that can be added by overlined elements.

– If $\overline{\sigma[i]}$ appears in $\widetilde{\sigma}$ (and thus $\sigma[i] = 1$) then the bottommost element of σ must be matched to the bottommost element of π.
  - If $\sigma[i]$ is a RLMin element, then remark that:
    * Every problem $PM_{\pi}^{\widetilde{\sigma}}(\mathrm{lb}, *, i, *)$ is true only if $\sigma[i]$ is matched to the element with value lb (by recursion). So it is enough to require that $\mathrm{lb} = 1$.
    * The recursive calls for $\sigma[1]$, ..., $\sigma[i-1]$ do not change the lower bound as $\sigma[i]$ is the leftmost RLMin element. Indeed if not, then there exists a RLMin element $\sigma[i']$, $i' < i$ and by the shape of a wedge permutation $\sigma[i'] < \sigma[i]$ which is not possible as $\sigma[i]$ must be the bottommost element. As a consequence $\sigma[1]$, ..., $\sigma[i-1]$ are RLMax elements. Finally the recursive calls for RLMax elements do not change the lower bound.
    * The first call of the problem has for parameter $\mathrm{lb} = 1$.

    So $PM_{\pi}^{\widetilde{\sigma}}(*, *, i, *)$ return true only if $\sigma[i]$ is matched to 1.
  - If $\sigma[i]$ is a RLMax element then $i = k$ ($\sigma[i]$ is the rightmost element). Thus every $PM_{\pi}^{\widetilde{\sigma}}(*, *, i, *)$ is a base case and is true only if $\sigma[i]$ is matched to 1.

– If $\sigma[i]\overline{\ }$ appears in $\widetilde{\sigma}$ (and thus $\sigma[i] = k$) then the topmost element of σ must be matched to the topmost element of π.
  - If $\sigma[i]$ is a RLMax element, then remark that:
    * Every problem $PM_{\pi}^{\widetilde{\sigma}}(*, \mathrm{ub}, i, *)$ is true only if $\sigma[i]$ is matched to the element with value ub (by recursion). So it is enough to require that $\mathrm{ub} = n$.
    * The recursive calls for $\sigma[1]$, ..., $\sigma[i-1]$ do not change the upper bound as $\sigma[i]$ is the leftmost RLMax element. Indeed if not, then there exists a RLMax element $\sigma[i']$, $i' < i$ and by the shape of a wedge permutation $\sigma[i'] > \sigma[i]$ which is not possible as $\sigma[i]$ must be the topmost element. As a consequence $\sigma[1]$, ..., $\sigma[i-1]$ are RLMin elements. Finally the recursive calls for RLMin elements do not change the upper bound.
    * The first call of the problem has for parameter $\mathrm{ub} = n$.

    So $PM_{\pi}^{\widetilde{\sigma}}(*, *, i, *)$ return true only if $\sigma[i]$ is matched to n.
  - If $\sigma[i]$ is a RLMin element then $i = k$ ($\sigma[i]$ is the rightmost element). Thus every $PM_{\pi}^{\widetilde{\sigma}}(*, *, i, *)$ is a base case and is true if $\sigma[i]$ is matched to n.

– If $\overline{\sigma[i]\sigma[i']}$ appears in $\widetilde{\sigma}$, (which implies that $\sigma[i'] = \sigma[i] + 1$) then if $\sigma[i]$ is matched to $\pi[j]$ then $\sigma[i']$ must be matched to $\pi[j] + 1$ or f $\sigma[i']$ is matched to $\pi[j]$ then $\sigma[i]$ must be matched to $\pi[j] - 1$.

- The cases $\sigma[i]$ is a RLMax element, $\sigma[i']$ is a RLMin element and $i < i'$ or $\sigma[i]$ is a RLMin element, $\sigma[i']$ is a RLMax element and $i' < i$ are not possible. Indeed $\sigma[i]$ is the topmost element of $\sigma[i :]$ thus $\sigma[i] > \sigma[i']$ which is in contradiction with $\sigma[i'] = \sigma[i] + 1$.
- If $\sigma[i]$ is a RLMin element, $\sigma[i']$ is a RLMax element and $i < i'$ (remark that this case is symmetric to the case where $\sigma[i]$ is a RLMax element, $\sigma[i']$ is a RLMin element and $i' < i$), then remark that
  - $i' = k$, indeed if i' is not the right most element, there would exist an element between $\sigma[i]$ and $\sigma[i']$. So every recursive call $PM_\pi^{\tilde{\sigma}}(\text{lb}, *, i', *)$ is solved as a base case. So $PM_\pi^{\tilde{\sigma}}(\text{lb}, *, i', *)$ is true only if $\sigma[i']$ is matched to the element with lb.
  - The recursive calls for $\sigma[i+1]$, ..., $\sigma[i'-1]$ do not change the lower bound. Indeed $\sigma[i]$ is the rightmost RLMin. So $\sigma[i+1]$, $\sigma[i+2]$, ..., $\sigma[i'-1]$ are RLMax elements. Moreover recursive calls for a RLMax do not change the lower bound.
  - $PM_\pi^{\tilde{\sigma}}(\text{lb}, *, i, *)$ sets the lb to $\pi[j]+1$ and matches $\sigma[i]$ to $\pi[j]$.

  So $\sigma[i]$ is matched to $\pi[j]$ and $\sigma[i']$ is matched to $\pi[j]+1$.
- If $\sigma[i]$ is a RLMin element and $\sigma[i']$ is a RLMin element then first remark:
  - From lemma 8, we have that $i < i'$.
  - Every recursive call $PM_\pi^{\tilde{\sigma}}(\text{lb}, *, i', *)$ is true only if $\sigma[i']$ is matched to the element with value lb.
  - The recursive calls for $\sigma[i+1]$, ..., $\sigma[i'-1]$ do not change the lower bound. Indeed from lemma 8, $\sigma[i+1]$, ..., $\sigma[i'-1]$ are RLMax elements. Finally the recursive calls for RLMax elements do not change the lower bound.
  - $PM_\pi^{\tilde{\sigma}}(*, *, i, *)$ sets lb to $\pi[j]+1$ and matches $\sigma[i]$ to $\pi[j]$.

  So $\sigma[i]$ is matched to $\pi[j]$ and $\sigma[i']$ is matched to $\pi[j]+1$.
- If $\sigma[i]$ is a RLMax element and $\sigma[i']$ is a RLMax element then first remark:
  - From lemma 8, we have that $i' < i$.
  - Every recursive call $PM_\pi^{\tilde{\sigma}}(*, \text{ub}, i, *)$ is true only if $\sigma[i]$ is matched to the element with value ub.
  - The recursive calls for $\sigma[i+1]$, ..., $\sigma[i'-1]$ do not change the upper bound. Indeed from lemma 8, $\sigma[i+1]$, ..., $\sigma[i'-1]$ are RLMin elements. Finally the recursive calls for RLMin elements do not change the upper bound.
  - $PM_\pi^{\tilde{\sigma}}(*, *, i', *)$ sets ub to $\pi[j]-1$ and matches $\sigma[i']$ to $\pi[j]$.

  So $\sigma[i']$ is matched to $\pi[j]$ and $\sigma[i]$ is matched to $\pi[j]-1$.

There are n^3 base cases that can be computed in constant time. There are kn^3 different cases. Each case takes up to $O(n)$ time to compute. Thus computing all the cases take $O(kn^4)$ time. Each case take $O(1)$ space, thus we need $O(kn^3)$ space. □

6 Computing the longest wedge permutation pattern

This section is focused on a problem related to the permutation pattern matching problem. Given a set of permutations, one must compute the longest permutation which occurs in each permutation in the set. This problem is known to be NP-Hard for an arbitrary size of the set even when all the permutations of this set are separable permutations (See [7]). We show how to compute the longest wedge permutation occurring in a set. We do not hope that this problem is solvable in polynomial time if the size of the set is not fixed. Indeed the size of the set appears in the exponent in the complexity of the algorithm. Thus we focus on the cases where only one or two permutations are given in set.

Conveniently, we say that a subsequence is a wedge subsequence if and only if the permutation represented by the subsequence is a wedge permutation.

We start with the easiest case where we are given just one input permutation. We need the set of RLMax elements and the set of RLMin elements. $A(\pi) = \{i|\pi[i] \text{ is a RLMin element}\} \cup \{n\}$ and $D(\pi) = \{i|\pi[i] \text{ is a RLMax element}\} \cup \{n\}$.

Proposition 4. *If s_i is the longest increasing subsequence with last element at position f in π and s_d is the longest decreasing subsequence with last element at position f in π then $s_i \cup s_d$ is a longest wedge subsequence with last element at position f in π.*

Proof. Let us first prove that we have a wedge subsequence. s_i is an increasing subsequence with values below or equal $\pi[f]$ and s_d is an decreasing subsequence with values above or equal $\pi[f]$, so $s_i \cup s_d$ is a wedge subsequence. Let us prove that this is a longest. Let s be a wedge subsequence with its rightmost element at position f in π such that $|s| > |s_i \cup s_d|$, first note that $A(s)$ is also an increasing subsequence with its rightmost at position f in π and $D(s)$ is also an decreasing subsequence with its rightmost at position f in π, then as $|s| > |s_i \cup s_d|$ then either $|A(s)| > |s_i|$ or $|D(s)| > |s_d|$, which is in contradiction with the definition of s_i and s_d.

Proposition 5. *Let π be a permutation. One can compute the longest wedge subsequence that can occur in π in $O(n \log(\log(n)))$ time and in $O(n)$ space.*

Proof (of Proposition 5). Proposition 4 leads to an algorithm where one computes the longest increasing and decreasing subsequence ending at every possible position and then finds the maximum sum of longest increasing and decreasing subsequence ending at the same position. Computing the longest increasing subsequences and the longest decreasing subsequences can be done in $O(n \log(\log(n)))$ time and $O(n)$ space (see [4]), then finding the maximum can be done in linear time. □

We now consider the case where the input is composed of two permutations.

Proposition 6. *Given two permutations π_1 of size n_1 and π_2 of size n_2, one can compute the longest common wedge subsequence in $O(n_1^3 n_2^3)$ time and space.*

Proof. Consider the following problem that computes the longest wedge subsequence common to π_1 and π_2: Given two permutations π_1 and π_2, we define $\mathrm{LCS}^{\pi_2,\mathrm{lb}_2,\mathrm{ub}_2}_{\pi_1,\mathrm{lb}_1,\mathrm{ub}_1}(i_1, i_2)$

$= \max \{|s| \mid s$ occurs $\pi_1[i_1 :]$ with every element of the occurrence in $[\mathrm{lb}_1, \mathrm{ub}_1]$ and s occurs $\pi_2[i_2 :]$ with every element of the occurrence in $[\mathrm{lb}_2, \mathrm{ub}_2]$ and s is a wedge subsequence $\}$

We show how to solve this problem by dynamic programming.

BASE:

$$\mathrm{LCS}^{\pi_2,\mathrm{lb}_2,\mathrm{ub}_2}_{\pi_1,\mathrm{lb}_1,\mathrm{ub}_1}(n_1, n_2) = \begin{cases} 1 & \text{if } \mathrm{lb}_1 \le \pi_1[n_1] \le \mathrm{ub}_1 \\ & \text{and } \mathrm{lb}_2 \le \pi_2[n_2] \le \mathrm{ub}_2 \\ 0 & \text{otherwise} \end{cases}$$

STEP:

$$\mathrm{LCS}^{\pi_2,\mathrm{lb}_2,\mathrm{ub}_2}_{\pi_1,\mathrm{lb}_1,\mathrm{ub}_1}(i_1, i_2) = \max \begin{cases} \mathrm{LCS}^{\pi_2,\mathrm{lb}_2,\mathrm{ub}_2}_{\pi_1,\mathrm{lb}_1,\mathrm{ub}_1}(i_1, i_2+1) \\ \mathrm{LCS}^{\pi_2,\mathrm{lb}_2,\mathrm{ub}_2}_{\pi_1,\mathrm{lb}_1,\mathrm{ub}_1}(i_1+1, i_2) \\ \mathrm{M}^{\pi_2,\mathrm{lb}_2,\mathrm{ub}_2}_{\pi_1,\mathrm{lb}_1,\mathrm{ub}_1}(i_1, i_2) \end{cases}$$

with

$$\mathrm{M}^{\pi_2,\mathrm{lb}_2,\mathrm{ub}_2}_{\pi_1,\mathrm{lb}_1,\mathrm{ub}_1}(i_1, i_2) = \begin{cases} 1 + \mathrm{LCS}^{\pi_2,\pi_2[i_2]+1,\mathrm{ub}_2}_{\pi_1,\pi_1[i_1]+1,\mathrm{ub}_1}(i_1+1, i_2+1) & \pi_1[i_1] < \mathrm{lb}_1 \\ & \text{and } \pi_2[i_2] < \mathrm{lb}_2 \\ 1 + \mathrm{LCS}^{\pi_2,\mathrm{lb}_2,\pi_2[i_2]-1}_{\pi_1,\mathrm{lb}_1,\pi_1[i_1]-1}(i_1+1, i_2+1) & \pi_1[i_1] > \mathrm{ub}_1 \\ & \text{and } \pi_2[i_2] > \mathrm{ub}_2 \\ 0 & \text{otherwise} \end{cases}$$

The solution to the problem relies on the fact that the longest wedge subsequence is found either by considering the problem with $\pi_1[i_1 :]$ and $\pi_2[i_2+1 :]$ or by considering the problem with $\pi_1[i_1+1 :]$ and $\pi_2[i_2 :]$ or by matching $\pi_1[i_1]$ and $\pi_2[i_2]$ and adding to the solution the LCS for $\pi_1[i_1+1 :]$ and $\pi_2[i_2+1 :]$ which is compatible, meaning that if we consider the current elements to correspond to a RLMin (resp. RLMax) element in the longest wedge subsequence then we consider only the solution with elements above (below) $\pi_1[i_1]$ for the occurrence in $\pi_1[i_1+1 :]$ and $\pi_2[i_2]$ for the occurrence in $\pi_2[i_2+1 :]$.

These relations lead to a $O(|\pi_1|^3|\pi_2|^3)$ time and $O(|\pi_1|^3|\pi_2|^3)$ space algorithm. Indeed there are $|\pi_1|^3|\pi_2|^3$ possible cases for the problem and each case is solved in constant time. □

References

1. S. Ahal and Y. Rabinovich. On complexity of the subpattern problem. *SIAM Journal on Discrete Mathematics*, 22(2):629–649, 2008.
2. M. H. Albert, M.-L. Lackner, M. Lackner, and V. Vatter. The Complexity of Pattern Matching for 321-Avoiding and Skew-Merged Permutations. *ArXiv e-prints*, October 2015.
3. M.H. Albert, R.E.L. Aldred, M.D. Atkinson, and D.A. Holton. Algorithms for pattern involvement in permutations. In *Proc. International Symposium on Algorithms and Computation (ISAAC)*, volume 2223 of *Lecture Notes in Computer Science*, pages 355–366, 2001.
4. Sergei Bespamyatnikh and Michael Segal. Enumerating longest increasing subsequences and patience sorting, 2000.
5. P. Bose, J.F.Buss, and A. Lubiw. Pattern matching for permutations. *Information Processing Letters*, 65(5):277–283, 1998.
6. M. Bouvel, D. Rossin, and S. Vialette. Longest common separable pattern between permutations. In B. Ma and K. Zhang, editors, *Proc. Symposium on Combinatorial Pattern Matching (CPM'07), London, Ontario, Canada*, volume 4580 of *Lecture Notes in Computer Science*, pages 316–327, 2007.
7. M. Bouvel, D. Rossin, and S. Vialette. Longest Common Separable Pattern between Permutations. *ArXiv Mathematics e-prints*, February 2007.
8. Marie Louise Bruner and Martin Lackner. A fast algorithm for permutation pattern matching based on alternating runs.
9. M. Crochemore and E. Porat. Fast computation of a longest increasing subsequence and application. *Information and Compututation*, 208(9):1054–1059, 2010.
10. R.G. Downey and M. Fellows. *Fundamentals of Parameterized Complexity*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2013.
11. S. Guillemot and D. Marx. Finding small patterns in permutations in linear time. In C. Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Portland, Oregon, USA*, pages 82–101. SIAM, 2014.
12. S. Guillemot and S. Vialette. Pattern matching for 321-avoiding permutations. In Y. Dong, D.-Z. Du, and O. Ibarra, editors, *Proc. 20-th International Symposium on Algorithms and Computation (ISAAC), Hawaii, USA*, volume 5878 of *LNCS*, page 10641073. Springer, 2009.
13. L. Ibarra. Finding pattern matchings for permutations. *Information Processing Letters*, 61(6):293–295, 1997.
14. V. Jelínek and J. Kyncl. Hardness of permutation pattern matching. *CoRR*, abs/1608.00529, 2016.
15. S. Kitaev. *Patterns in Permutations and Words*. Springer-Verlag, 2013.
16. L. Mach. *Parameterized complexity : permutation patterns, graph arrangements, and matroid parameters*. PhD thesis, University of Warwick, UK, 2015.