*Discrete Mathematics and Theoretical Computer Science* . . . . . .
. . . . . **vol. 27:1, Permutation Patterns 2024 #1 (2025)**

# *Sorting inversion sequences*

Toufik Mansour[1] Howard Skogman[2] Rebecca Smith[3]

[1] *Department of Mathematics, University of Haifa, Israel*
[2] *Excellus BlueCross BlueShield, Rochester, NY, USA*
[3] *Department of Mathematics, SUNY Brockport, NY, USA*

*revisions 31st July 2024, 7th Jan. 2025, 14th Jan. 2025; accepted 29th Jan. 2025.*

We consider the avoidance of patterns in inversion sequences that relate sorting via sorting machines including data structures such as stacks and pop stacks. Such machines have been studied under a variety of additional constraints and generalizations, some of which we apply here. We give the classification of several classes of sortable inversion sequences in terms of pattern avoidance. We are able to provide an exact enumeration of some of the sortable classes in question using both classical approaches and a more recent strategy utilizing generating trees.

**Keywords:** inversion sequence, permutation pattern, sorting algorithm, pop stack, generating tree

## 1 Introduction

An *inversion* in a permutation $\pi = \pi_1\pi_2\pi_3\cdots\pi_n$ is a pair of entries $\pi_i, \pi_j$ where $i < j$, but $\pi_i > \pi_j$. An *inversion sequence* of length $n$ is a word $e = e_1 \dots e_n$ which satisfies for each $i \in \{1, \dots, n\}$ the inequalities $0 \le e_i \le i-1$. These sequences correspond to permutations via a variant of the Lehmer code Laisant (1888); Lehmer (1960) by mapping each entry $\pi_j$ of the permutation $\pi$ to $e_{j-1}$ where $e_{j-1}$ is the number of inversions in $\pi$ where $\pi_j$ is the second, smaller entry of the pair. The set of inversion sequences (respectively the set of inversion sequences of length $n$) is denoted $\mathbf{I}$ (respectively $\mathbf{I}_n$).

**Example 1.1.** *The permutation* $\pi = 51743862$ *corresponds to the inversion sequence* $e = 01023026$.

The study of pattern avoidance in inversion sequences began with the work of Corteel, Martinez, Savage, and Weselcouch 2016 as well as that of Shattuck and the first author 2015. Since then, extensive work has been done in this area, including Lin (2020); Lin and Kim (2021); Yan and Lin (2020–2021) to list a few. Recently, Kotsireas, Yıldırım, and the first author 2024 introduced an algorithmic technique involving generating trees to enumerate many pattern classes of inversion sequences (also see Callan et al. (2023); Callan and Mansour (2023) and the references therein). We utilize this algorithm in Section 4 to obtain a generating function for the number of inversion sequences sortable by two particular machines and give the succession rules to build the generating tree for a third machine.

There remain many open problems in characterizing and enumerating the inversion sequences that avoid a class consisting of a small number of relatively short (length less than five) patterns. Our interest is in the characterization and enumeration of inversion sequences sortable by stacks and pop stacks. We also study the characterization of inversion sequences sortable by data structures obtained by generalizing or further restricting the pop stacks.

ISSN 1365–8050 

A stack is a last-in, first-out sorting device with push and pop operations. Knuth 1969 showed the permutation $\pi$ can be sorted by a stack (that is, by applying push and pop operations to the sequence $\pi_1, \ldots, \pi_n$ one can output the identity permutation $1, \ldots, n$) if and only if $\pi$ avoids the permutation 231.

**Definition 1.2.** *A permutation $\pi = \pi_1\pi_2\ldots\pi_n \in S_n$ is said to* contain *a permutation $\sigma = \sigma_1\sigma_2\ldots\sigma_k$ if there exist indices $1 \leq \alpha_1 < \alpha_2 < \ldots < \alpha_k \leq n$ such that $\pi_{\alpha_i} < \pi_{\alpha_j}$ if and only if $\sigma_i < \sigma_j$. Otherwise, we say $\pi$* avoids *$\sigma$.*

*These definitions of* containment *and* avoidance *have been naturally extended to words (and specifically inversion sequences) with the added allowance of entries being equal to one another.*

**Example 1.3.** *The permutation $\pi = 241563$ contains $312$ since the $4, 1, 3$ appear in the same relative order as $3, 1, 2$. However, $\pi$ avoids $321$ since there is no decreasing subsequence of length three in $\pi$.*

*Similarly, the inversion sequence $\tau = 0021104$ contains $1002$ from $2, 1, 1, 4$, but avoids $201$.*

Describing the permutations that are sortable by machines consisting of one or more data structures (such as stacks or pop stacks) is a point of interest in the field. To this end, it is often possible to describe the sortable machine in terms of its *basis*.

**Definition 1.4.** *A* permutation class *is a downset of permutations under the containment order. Every permutation class can be specified by the set of minimal permutations which are* not *in the class called its* basis*. For a set $B$ of permutations, we denote by $\mathrm{Av}(B)$ the class of permutations which do not contain any element of $B$.*

In the same vein, for any set of words $R$, denote by $\mathbf{I}(R)$ the set of inversion sequences in $\mathbf{I}$ that avoid every word in $R$. Similarly, denote by $\mathbf{I}_n(R)$ the set of inversion sequences of length $n$ that avoid every word in $R$.

As mentioned above, the stack-sortable permutations are the class $\mathrm{Av}(231)$.

A restricted version of a stack that has received a fair amount of attention is a pop stack, first introduced by Avis and Newborn 1981. A pop stack supports the same push and pop operations as a standard stack, but whenever a pop operation occurs, every entry in the pop stack is popped immediately. One foundational result is that the permutations that can be sorted by a single pop stack are exactly those that avoid 231 and 312. Some later works involving pop stacks include Asinowski et al. (2019); Atkinson and Sack (1999); Cerbai (2021); Claesson and Guðmundsson (2019); Defant and Williams (2022); Elder and Goh (2021); Hong (2022); Pudwell and Smith (2019); Smith and Vatter (2009).

Atkinson 1998 explored a variation of a stack called an $(r, s)$-stack which extends the traditional push and pop operations. Specifically, the first index of an $(r, s)$ stack allowed one to push an entry not only to the top of the stack but to any of the top $r$ spots. Similarly, the second index expanded the options of the pop operation to allow one to pop any of the top $s$ entries from the stack. Thus a $(1, 1)$-stack is simply a traditional stack. Atkinson was able to characterize these permutations when one of $r, s$ was fixed at 1, as well as in the case when $r = s = 2$. Natural variations on stacks arise when studying inversion sequences that are similar in flavor to these generalized stacks.

Elder 2006 considered restricting the depth of a stack, that is the maximum number of entries allowed in a stack at any given time, in the context of generating permutations with two stacks in series. Elder later continued this study with Lee and Rechnitzer 2015 and then extended to sorting and pop stack sorting with Goh 2018; 2021. Although this work focused on a permutation sorting (or generating) machine consisting of two stacks in series, we will consider a similar restriction for a single stack or pop stack when sorting inversion sequences.

Note that natural sorting machines consisting of stacks and/or pop stacks do not give any advantage in sorting a sequence by inserting extra entries that might force a stack or pop stack to be popped at an earlier stage. As such, the sortable words from any such machine form a class and can be completely characterized by classical pattern avoidance.

## 2 Stack sorting inversion sequences

We give the standard optimal stack sorting algorithm extended to words in Algorithm 1. Note that a word is considered sorted if all of its entries appear in weakly increasing order.

**Algorithm 1** Stack Sorting Algorithm

$w = w_1 w_2 \cdots w_n$ is a word of length $[n]$.
$S$ is a stack.
$O$ is an empty array.
$i = 1$.
**procedure** STACKSORT($w, n$)
  **while** $i < n + 1$ **do**
    **if** $S$ is empty **then**
      push $w_i$ to the top of $S$
      $i = i + 1$
    **else if** $w_i \leq$ top element of $S$ **then**
      push $w_i$ to the top of $S$
      $i = i + 1$
    **else**
      pop the top entry of $S$ to the end of $O$
    **end if**
  **end while**
**return** $O$
**end procedure**

For a word $w$ to be stack sortable, $w$ must avoid $120$, the same as avoiding $231$ which is required for permutations to be sortable. The repetition allowed in words does not change this requirement from being both necessary and sufficient and so the proof does not change either. It is included below for completeness.

**Proposition 2.1.** *The words that are stack sortable are exactly the words that avoid* $120$.

**Proof:** The $2$ of a $120$ pattern forces the smaller entry $1$ to the output before the $0$ even enters the stack which creates an inversion with the $1, 0$. Hence words containing a $120$ pattern are not stack sortable.

Next consider a word $w$ that is not stack sortable. The output $S(w)$ must contain an inversion, say an entry $b > a$ that appears before $a$. For $a$ to not have exited the stack before $b$, we know $a$ appears after $b$ in $w$. Furthermore, for $b$ to have been forced out of the stack before $a$ entered, there must have been a larger entry $c$ that appeared after $b$, but before $a$ to force $b$ out before $a$ could enter the stack. Thus $w$ contains a $120$ pattern $bca$. □

As inversion sequences are words with added restrictions, Proposition 2.1 also gives us a basis for the stack sortable inversion sequences.

**Corollary 2.2.** *The inversion sequences that are stack sortable are exactly those that avoid* $120$.

The enumeration of these sortable inversion sequences has been studied, but thus far remains open. See (Mansour and Shattuck, 2015, Section 4) and Corteel et al. (2016) for initial study and further work on avoiding $120$ in Yan and Lin (2020–2021).

# 3 Pop stack sorting inversion sequences

The optimal pop stack sorting algorithm is mostly the same procedure as that for stacks, but when a pop stack is popped, all entries must exit the stack as described in Algorithm 2. Note that the end result of this algorithm is a reversal of each maximal weakly descending run of the input sequence.

**Algorithm 2** Pop Stack Sorting Algorithm

$w = w_1 w_2 \cdots w_n$ is a word of length $[n]$.
$PS$ is a pop stack.
$O$ is an empty array.
$i = 1$.
**procedure** POPSTACKSORT($w, n$)
  **while** $i < n + 1$ **do**
    **if** $PS$ is empty **then**
      push $w_i$ to the top of $PS$
      $i = i + 1$
    **else if** $w_i \leq$ top element of $PS$ **then**
      push $w_i$ to the top of $PS$
      $i = i + 1$
    **else**
      **while** $PS$ is nonempty **do**
        pop the top entry of $PS$ to the end of $O$
      **end while**
    **end if**
  **end while**
**return** $O$
**end procedure**

Note that $120, 201$ are the patterns whose avoidance completely determines the pop stack sortability of a permutation. However, due to the repetition allowed in words, a copy of a larger element already in the stack can force the entire stack to be popped while there is still a copy of a smaller entry appearing later in the permutation.

We note that Cerbai 2021, Theorem 5.2 found the basis for sortable Cayley words in the context of a *hare pop stack* which is simply a pop stack that only allows entries to appear in weakly decreasing order from top to bottom (which is a necessary requirement for sorting with a single stack). This basis is the

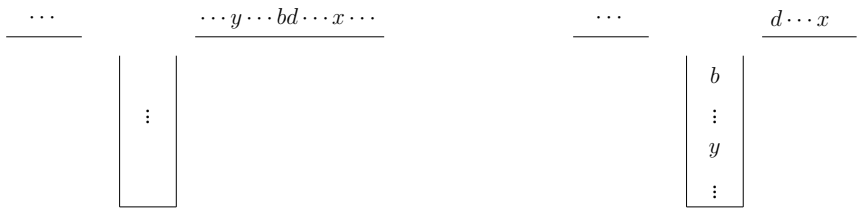


**Fig. 1:** Moving to the state where $y$ will be forced to exit the pop stack before $x$ enters

same as the one in Theorem 3.1 below. Cerbai's argument could also be extended to sorting general words by a pop stack as we have done here.

**Theorem 3.1.** *The words that are pop stack sortable are exactly the words that avoid* $120, 201,$ *and* $1010$.

**Proof:** One can check that none of $120, 201,$ and $1010$ are pop stack sortable. As such, a word that contains any of these patterns cannot be sorted by a pop stack.

Conversely, suppose $w$ is not pop stack sortable. Then the image of $w$ under the pop stack operator, which we will denote by $PS(w)$, must contain an inversion $(y, x)$. That is, $y > x$ and $y$ appears before $x$ in $PS(w)$. Consider what happens to force the entry $y$ to the output before $x$.

Using Algorithm 2, there must be a point when $y$ is in the stack and an entry $d$ that appears (immediately) after a smaller entry $b$ in $w$. Because $b < d$ and $b$ appears on the top of the pop stack when $d$ is the next entry of $w$, the pop stack must be popped before $d$ can enter. As $x$ appears after $y$ in $PS(w)$, it must be that $x$ is still in the input at this state. These entries (if they are all distinct) are shown before entering the pop stack (left) and in the state where $b$ just entered the pop stack (right) in Figure 1.

If $y = b$, then $x$ appears after $d$ in $w$. Thus $bdx$ forms a $120$ pattern in $w$.

Otherwise, there exists an entry $y > b$ that is also forced out before $d$ enters the pop stack but before $x$. Thus, $y$ appears before $b$ in $w$. Either we can assume $x = d$ (that is, $y > d$) or if $x < d$, then $x$ appears after $d$ in $w$. In the former case, $ybd$ is a $201$ pattern in $w$. In the latter case, $x < d$ appears after $d$, so consider the pattern $ybdx$ in $w$ where we are reduced to the options $d \geq y > x, b$. Thus, $ybdx$ is one of the following patterns $1010, 1020, 2021, 2120, 2031, 2130$. Of these patterns, only $1010$ does not contain one of the smaller forbidden patterns $120, 201$. □

As inversion sequences are words with added restrictions, Theorem 3.1 also gives us a basis for the pop stack sortable inversion sequences.

**Corollary 3.2.** *The inversion sequences that are pop stack sortable are those in* $\mathbf{I}(120, 201, 1010)$.

The pop stack sortable permutations are also known as *layered* permutations which are permutations of the form $\pi = \sigma_1\sigma_2 \cdots \sigma_k$ where each *layer* $\sigma_i$ is a maximal length contiguous decreasing sequence such that all of the entries of $\sigma_i$ are less than all of the entries of $\sigma_{i+1}$. For example, the permutation $\pi = 543216987$ is a layered permutation with layers $\sigma_1 = 54321, \sigma_2 = 6, \sigma_3 = 987$ as shown on the left in Figure 2.

We can define layered words similarly as follows.

**Definition 3.3.** *A* layered *word* $w$ *is a word* $w = \tau_1\tau_2 \cdots \tau_k$ *where each* $\tau_i$ *is*

1. *maximal in length,*


**Fig. 2:** The layered permutation $\pi = 543216987$ and the layered word $w = 211033275$.

2. *weakly decreasing, and*

3. *such that the last (weakly smallest) entry of $\tau_{i+1}$ is at least as large as the first (weakly largest) entry of $\tau_i$ for all $i$.*

For example, the word $w = 211033275$ is a layered word with layers $2110, 332, 75$ shown on the right in Figure 2.

It can then be seen that these layered words are exactly the pop stack sortable words.

**Proposition 3.4.** *Layered words are exactly the words that avoid* $120, 201, 1010$.

**Proof:** Suppose $w$ is layered. Then $w$ cannot contain $120$ because if not, the layer containing the $0$ of the pattern must appear after the layer containing the $1$ which violates the third condition. The same argument can be made for the $1$ of the $201$ pattern appearing in a later consecutive decreasing pattern than that of the $2$ and once again for the second $0$ of the $1010$ pattern appearing in a later consecutive decreasing pattern than that of the first $1$.

Now suppose $w$ avoids $120, 201, 1010$. Then $w$ is pop stack sortable. Consider the output from sorting $w$ broken into segments based on the pops. Reversing those segments reverts the word back to $w$. In addition, according to the algorithm, these breaks occur exactly at the strict ascents in $w$. The segments then are weakly decreasing so that the smallest entry of any segment is at least as large as the largest entry of the previous segment for $w$ to be sortable and also maximal in length. Hence $w$ is layered. □

In Section 4.2, we are able to describe the generating tree for layered inversion sequences. There are also some partial results we can obtain combinatorially based on the number of layers (which is also the number of pops needed to sort the sequence). Recall that the Eulerian numbers $E(n, k)$ count the number of permutations of length $n$ with $k - 1$ descents.

**Proposition 3.5.** *The number of pop stack sortable inversion sequences of length $n$ with $k = 1, 2$ layers are counted by $E(n, k)$.*

**Proof:** In fact, the pop sortable inversion sequences of length $n$ with $k = 1, 2$ layers correspond exactly to the permutations of length $n$ with $k - 1$ descents by the variant of Lehmer code used to create inversion sequences. In the trivial case of $k = 1$, a permutation of length $n$ with $k - 1 = 0$ descents is the identity permutation which is the only permutation of length $n$ to have an inversion sequence with one layer, namely the inversion sequence consisting of all $0$s.

Now consider the case when $k = 2$. A permutation $\pi$ with exactly one descent, say at index $i$, will have an inversion sequence that begins with $i$ 0s which will be the first layer. Since the permutation $\pi$ is such that $\pi_{i+1}$ is the smaller entry in an inversion with at least $\pi_i$, the corresponding inversion sequence now has a positive entry in position $i + 1$. As the remainder of the entries of $\pi$ are all in increasing order, each subsequent entry of $\pi$ can be the smaller entry of at most as many inversions as the previous entry. Thus, the rest of the inversion sequence is in weakly decreasing order meaning that the entries from position $i + 1$ to position $n$ form the second and final layer. □

As not all inversion sequences are pop stack sortable, this correspondence can clearly not continue indefinitely. Indeed, a permutation with even $k - 1 = 2$ descents is not guaranteed a corresponding pop stack sortable inversion sequence. For example, the permutation $\pi = 3214$ has the inversion sequence $e(\pi) = 0120$, which cannot be sorted by a pop stack.

At the other extreme, only one inversion sequence of any given length $n \geq 1$ can have $n$ layers, namely the increasing sequence $012 \ldots (n-1)$. Reducing the number of layers by one to $n-1$ layers gives a set of inversion sequences enumerated by the Tetrahedral numbers, sequence number A000292 in OEIS Sloane (2025).

**Proposition 3.6.** *The number of pop stack sortable inversion sequences of length $n \geq 1$ with $n - 1$ layers is $\binom{n+1}{3}$.*

**Proof:** A pop stack sortable inversion sequence $e$ of length $n \geq 1$ with $n - 1$ layers will have exactly one entry $e_i$ as the second entry of a layer where $i \in \{2, 3, 4, \ldots, n\}$. It must be the case that $e_m = m - 1$ for all $m < i$ for those entries to have all started new layers. Note that $e_i \geq e_{i-2}$ as otherwise $e_{i-2}, e_{i-1}, e_i$ would form a forbidden 120 pattern in $e$. This means $e_i = e_{i-1} = i - 2$ or $e_i = e_{i-2} = i - 3$. Further, we know $e_i, e_{i+1}, e_{i+2}, \ldots, e_n$ are in strictly increasing order as they are all in different layers.

In the case that $e_i = i - 2$, since we must have $i - 1 \leq e_{i+1} < e_{i+2} < \cdots < e_n \leq n - 1$, we can select up to one of these later entries to be two greater than the previous entry. Consider positions $\{2, 3, 4, \ldots, n, n + 1\}$ where we pick two positions; the first to be the position of the second entry in the layer of length two and the second to be the position of the entry that is two more (as opposed to one more) than the previous. In the case that $n + 1$ is one of the selected options, then we do not have any entry that is two more than the one before it. Hence there are $\binom{n}{2}$ such inversion sequences.

Otherwise, $e_i = i - 3$ and so $i \geq 3$. Now because $i - 2 \leq e_{i+1} < e_{i+2} < \cdots < e_n \leq n - 1$, up to two of these later entries could be two greater than the previous entry or one of these later entries could be three greater than the previous entry. For the cases where no entry is three more than the previous entry, we have $\binom{n-2}{1} + \binom{n-2}{2} + \binom{n-2}{3}$ ways to select the second entry of the length two layer and the entries that are two greater than the entry they follow. In the case where we have one entry that is three more than the previous entry, there are $\binom{n-2}{2}$ ways to select this entry and the entry that was the second in the longest layer. This gives us a total of $\binom{n-2}{1} + 2\binom{n-2}{2} + \binom{n-2}{3} = \binom{n}{3}$ via a generalization of Pascal's Identity.

By another application of Pascal's Identity, there are $\binom{n}{2} + \binom{n}{3} = \binom{n+1}{3}$ pop stack sortable inversion sequences of length $n \geq 1$ with $n - 1$ layers. □

One last special case is when the sortable inversion sequence contains only 0s and 1s and thus only needs to avoid 1010, thereby having at most three layers. The number of such sequences is the number of compositions of $n$ into at most four parts because we must begin with a 0 and can only alternate values at most three times after that. These numbers (with an offset of one) are better known as the "Cake numbers" Yaglom and Yaglom (1987), sequence number A000125 in OEIS Sloane (2025), which

count the maximum number of slices of a cylindrical cake one can have after making $n-1$ planar cuts perpendicular to the top circular surface.

**Proposition 3.7.** *The number of pop stack sortable inversion sequences of length $n$ only containing $0,1$ is the $(n-1)$st Cake number: $\binom{n-1}{0}+\binom{n-1}{1}+\binom{n-1}{2}+\binom{n-1}{3}$.*

We note that Cerbai 2021 also considered a *tortoise pop stack* for sorting Cayley words that had an additional pattern restriction of not allowing a $00$ pattern to appear in the pop stack. (Restricting the content of stacks in terms of pattern avoidance was initially considered in Atkinson et al. (2002); Smith (2014); Cerbai et al. (2020).) Cerbai's characterization of sortable Cayley words can be easily extended to all words, so we omit the proof here.

**Proposition 3.8.** *The words (and thus inversion sequences) that are tortoise pop stack sortable are those which avoid $120, 201, 110,$ and $100$.*

Given this classification, the generating function for the enumeration of tortoise pop stack sortable inversion sequences is given in Theorem 12 in recent work by Callan and Mansour 2023.

## 3.1 Restricting pop stack depth

When extending the notion of the depth of a stack to words, we could continue to use the definition in terms of the number of elements allowed in the stack as introduced by Elder 2006. This could be more practical in some applications. However, given the repetition that distinguishes words from permutations and considering the storage in terms of distinct elements to remember, it also makes sense to consider another definition that is still consistent with the original definition when applied to permutations. The content of the stack could be stored as a limited number (the depth) of lists of positions in the stack where these positions are counted from the bottom. We consider the latter notion as given in the following definition.

**Definition 3.9.** *The* depth *of a stack (or pop stack) is the number of entries with distinct values allowed in the stack (or pop stack) at any stage.*

For example, a stack of depth three under this definition would still only allow three entries of a permutation to be in the stack at one time. However, this same restricted stack could allow arbitrarily many entries of a word in the stack, provided that no more than three distinct letters were represented at any time. Using this definition, we begin by characterizing words and specifically inversion sequences sortable by pop stacks of limited depth.

**Proposition 3.10.** *A word $w$ is sortable by a pop stack of depth $k$ if and only if $w$ avoids $120, 201, 1010$ and also $k(k-1)\cdots 10$.*

**Proof:** As before any word $w$ must avoid $120, 201, 1010$ to be pop stack sortable. Additionally, a decreasing sequence that is too long to fit in a limited depth pop stack is also not sortable.

Conversely, if $w$ is not pop stack sortable by a pop stack of depth $k$ and avoids $120, 201, 1010$, then the problem is a result of the limitation of the pop stack's capacity. Hence $w$ must contain the $k(k-1)\cdots 10$ pattern. □

**Corollary 3.11.** *Inversion sequences sortable by a pop stack of depth $k$ are exactly those in $\mathbf{I}(120, 201, 1010, k(k-1)\cdots 10)$.*

In the case of depth one, the number of sortable inversion sequences is the number of weakly increasing inversion sequences which are known to be enumerated by the Catalan numbers due to Martinez and Savage 2018. The proof shows these inversion sequences to correspond to the 213 avoiding permutations, which were enumerated by Simion and Schmidt 1985.

**Theorem 3.12.** *(Martinez and Savage) The number of inversion sequences of length $n$ which avoid $10$ and thus are sortable by a pop stack of depth $1$ is the $n$th Catalan number $C_n = \frac{\binom{2n}{n}}{n+1}$.*

For a pop stack of depth two, the sortable inversion sequences would need to avoid $120, 201, 210, 1010$. The first few terms of the enumeration sequence are: $1, 1, 2, 6, 23, 100, 471, 2349$. We define the succession rules for the generating tree for these inversion sequences (using the algorithm introduced in Kotsireas et al. (2024)) in Section 4.3. However, it is hard to get an explicit formula by translating this generating tree to a system of equations and it is even more difficult for larger values of $k$.

Another approach is to count the inversion sequences recursively directly, but this too is not as nice as we would hope. One recursive characterization is given below.

A weakly decreasing word of length $n$ has depth $k$ if it has the form $a_1 \dots a_1 a_2 \dots a_2 \dots a_{k+1} \dots a_{k+1}$ where for each $i$, $a_i > a_{i+1} \geq 0$ (so there are $k$ descents). Let $WD_k(n, a)$ be the set of all such words whose first value is $a$ and whose depth is at most $k$.

**Lemma 3.13.** *For all $n, k \in \mathbb{Z}^+$ with $k < n$, we have*

$$|WD_k(n, a)| = 1 + \sum_{j=1}^{k} \binom{n-1}{j} \binom{a}{j}. \tag{1}$$

Note that we assume $\binom{n}{k} = 0$ if $k > n$. The result follows from choosing $j$ locations for a descent, then $j$ values for the weakly decreasing sequence, and one for the sequence with no descents.

We say a word $w$ of length $n$ is a layered weakly decreasing word of depth at most $k$ if $w$ can be split into a sequence of subwords (in the factor sense) $w_1, w_2, ...$ where each $w_i \in WD_k(m_i, a_i)$ with $\sum_i m_i = n$, and for each $i$, $a_i \geq 0$, and $min(w_{i+1}) \geq max(w_i)$ (i.e. the largest (first) value of each word is bounded by the smallest value of the succeeding word). Further, we consider the words of this type which are also inversion sequences, i.e. $max(w_i) \leq \sum_{j=1}^{i-1} m_j$. Let $LDI_k(n)$ be the set of all such layered depth at most $k$ inversion sequences of length $n$.

**Lemma 3.14.** *The set of sortable inversion sequences by a depth $k$ pop stack is exactly $LDI_k(n)$.*

The proof is simply to note that when the stack is popped, there are at most $k$ distinct values, and they must be less than any later sequence values.

Let $w_1 w_2 ... w_t \in LDI_k(n)$, where $w_1, w_2, ..., w_t$ are the subwords as defined above with each value of $w_{i+1}$ bounded below by the largest value in $w_i$. Subtracting $a_i$ from every value in $w_{i+1}$ yields an element of $WD_k(m_i, a_{i+1} - a_i)$.

As a result we get the following formula.

**Proposition 3.15.** *For all $n, k > 0$,*

$$|LDI_k(n)| = \sum_{\substack{m_1+m_2+...m_t=n, \\ m_i \geq 0}} \quad \sum_{a_1, a_2, ... a_t}^{*} |WD_k(m_i, a_i)|, \tag{2}$$

*where* $a_1 = 0$*, and* $\sum^*$ *means for all* $j > 1$ *to sum on values of* $a_j$ *such that* $0 \leq a_j \leq \left( \sum_{h=1}^{j-1} m_h - a_h \right)$.

Since the first word in an element of $LDI_k(n)$ must be a string of zeros, we let $LDI_k(n, m_1)$ denote the elements of $LDI_k(n)$ that begin with $m_1$ zeros followed by a non-zero value. Further, let $LDI_k(n, m_1, a_2)$ be the subset of these layered decreasing inversion sequences which begin with $m_1$ zeros, and have first non-zero value $a_2$ (and $a_2 \leq m_1$). We now decompose our sequence of words as $\{\overline{0}_{m_1}, w_2, \hat{w}\}$ where $\hat{w} = \{w_3, ...w_t\}$. Now $w_2 \in WD_k(m_2, a_2)$ for some $0 \leq m_2 \leq n - m_1$, $a_2 > 0$. If we subtract $a_2$ from all the elements in $\hat{w}$ the result is in $LDI(n - m_2 - m_1, m_1 + m_2 - a_2, a)$ for some $a$ with $a \leq m_1 + m_2 - a_2$. Note that the number of starting zeros guarantees that adding $a_2$ to all of these entries will still yield an element of $LDI_k(n)$.

Hence we get the following formula.

**Proposition 3.16.** *For all* $n, k, m_1 \in \mathbb{Z}^+$ *with* $m_1 < n$*,*

$$|LDI_k(n, m_1)| = \sum_{m_2=0}^{n-m_1} \sum_{a_2 \leq m_1} |WD_k(m_2, a_2)| \sum_{a \leq m_1+m_2-a_2} |LDI_k(n - m_1 - m_2, m_1 + m_2 - a_2, a)|. \tag{3}$$

Equations 2 and 3 are rather difficult to use for computation. However they are presented to give some insight into directly enumerating these inversion sequences.

For completeness, we give some characterizations of words sortable by stacks of depth $k$.

**Proposition 3.17.** *A word* $w$ *is sortable by a stack of depth* $k$ *if and only if* $w$ *avoids* $120$ *and also* $k(k-1)\cdots 10$*.*

**Corollary 3.18.** *The number of words of length* $n$ *on an alphabet* $[k]$ *sortable by a stack of depth* $1$*, i.e. avoiding* $10$ *is* $\binom{n+k-1}{k-1}$.

**Proof:** These weakly increasing words can be thought of as weak compositions of $n$ into $k$ parts where each part represents the next largest letter. □

The depth 2 stack sortable words are enumerated by Burstein 1988 in the context of avoiding $120, 210$.

**Theorem 3.19.** *(Burstein) The number of words of length* $n$ *on an alphabet* $[k]$ *avoiding* $120, 210$ *and thus sortable by a stack of depth* $2$ *is given by*

$$\frac{1-(-1)^k}{2} + 2^n \sum_{i=0}^{\lfloor \frac{k-2}{2} \rfloor} \binom{n+k-3-2i}{n-1}.$$

Finally, we note that Kotsireas, Yıldırım, and the first author 2024 give a functional equation for the inversion sequences of length $n$ avoiding $120, 210$ and thereby sortable by a stack of depth 2.

## *3.2 Generalizing pop stacks*

We can also expand the ability of our sorting machine using a similarly defined version of Atkinson's 1998 extension of push and pop operations. Specifically, we will define a $(r, 1)$-pop stack that is also modified to take into account the repetition of letters in inversion sequences (and more generally words). In Atkinson's

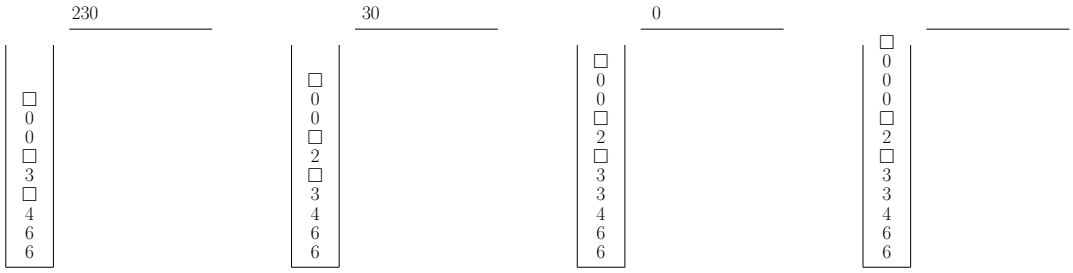


**Fig. 3:** A $(3, 1)$-stack shown at several stages.

work, an $(r, 1)$-stack would have allowed for push operations to push entries into the stack in any of the top $r$ positions. We can again naturally redefine generalized stacks for words in a way that is consistent with Atkinson's work on permutations.

**Definition 3.20.** *Let an $(r, 1)$-stack allow the push operation to push an entry to any of the top $r$ "value positions" in the stack where consecutive entries of the same value in the stack are grouped together.*

Note that the above definition could be extended to the broader notion of an $(r, s)$ stack where we allow the pop operation to be defined in terms of these same value positions.

**Example 3.21.** *Suppose a $(3, 1)$-stack contained the values $0, 0, 3, 4, 6, 6$ as shown in Figure 3 with the remainder of the word being $230$. The available positions for new entries to enter the stack at each stage are indicated by the $\square$s.*

Varying the restrictions of depth and generalizations of the push operation as described above, we get a family of stacks and a family of pop stacks. There are some variations that no longer give us new machines though. Specifically, if our stack or pop stack has limited depth, having the option to push entries to (or pop entries from) spaces that do not exist does add any sorting power.

**Proposition 3.22.** *For a $(r, 1)$-stack or $(r, 1)$-pop stack of depth $d$, if $r > d$, the machine sorts the same words as if $r = d$.*

As such, we will restrict $r \leq d$.

We will first consider the case where $r = d = 2$.

The $(2, 1)$-pop stack of depth two has the nice characteristic where it sorts words (and permutations) that avoid three patterns of length three.

**Theorem 3.23.** *A $(2, 1)$-pop stack of depth two sorts exactly the words that avoid $120, 201, 210$.*

**Proof:** Because the $(2, 1)$-pop-stack we are using only has depth two, any pattern of length three with three distinct values will have at least the first entry popped out before the third entry can enter. For all of $120, 201, 210$, this means the larger first entry is popped out before the smaller last entry enters the stack. Thus no word containing $120, 201, 210$ is sortable by a $(2, 1)$-pop stack of depth two.

Now consider a word $w$ that cannot be sorted by a $(2, 1)$-pop stack of depth two. If a larger entry was forced out before a smaller entry could be pushed into the stack due to the limited depth, there are two ways this could have happened. One is that two distinct larger entries appeared before a smaller entry, that is $w$ contains a $120$ or $210$ pattern. Otherwise, we have a large entry that is below a small entry in the pop

stack, and our third entry waiting to enter is between the small and large entry in value. Because of the $(2,1)$ generalization, in theory, the first two entries could appear in either order in $w$ to form either a $021$ or a $201$ pattern. However, in the case of a $021$ pattern, the $0$ entry could be popped out before pushing the $2$ into the pop stack, so only the $201$ pattern presents a problem. □

Note: Algorithm 3 gives a procedure for sorting with a $(2,1)$-pop stack of depth 2. Because there are only two possible values in $PS2$, it would be equivalent to simply push $w_i$ to the bottom of $PS2$ if it matched the larger value in the $(2,1)$-pop stack of depth 2 rather than keep track of how many smaller entries there are. However, we present the algorithm in this form for the option of generalization if desired.

**Theorem 3.24.** *Algorithm 3 is optimal for sorting words with an $(r,1)$-pop stack of depth 2.*

**Proof:** By way of contradiction, let $w$ be a sortable word that is not sorted by our algorithm on a $(2,1)$-pop stack of depth 2. Specifically, $S(w)$ must contain a descent, say $w_j w_k$. Because Algorithm 3 never allows a descent in the pop stack, $w_j$ must have been the last entry popped from one pop operation and $w_k$ must have been the first entry popped from the next pop operation. The entries $w_j$ and $w_k$ must have appeared in the same relative order (forming an inversion) in $w$. The only way to remove this inversion would be for both entries to be in the pop stack at the same time.

Notice Algorithm 3 only pops the pop stack in two cases. In the first case, the pop stack already contained two distinct values that are different from the next input value. In this case, $w_j$ is one of the larger entries in the pop stack. Also, $w_j$ is the first entry in $w$ from the all of entries of the same value popped at the same time as $w_j$ because a new entry only enters in the pop stack if it is smaller than the other value represented or the new entry matches a value already there and is placed above it in the pop stack. In fact, $w_j$ must be pushed into an empty pop stack as we are not inserting any unmatched larger entries into the pop stack after a smaller entry. Thus there must be a smaller entry $w_m$ popped at the same time as $w_j$ and was inserted into the pop stack after $w_j$. Notice $w_j w_m w_k$ form a $201$ or $210$ pattern in $w$ which is forbidden.

Otherwise, when $w_j$ is popped from the pop stack, only values matching $w_j$ can be in the pop stack. For the next entry $w_i$, not to enter the pop stack, we must have $w_j < w_i$. In this case, $w_j w_i w_k$ form a $120$ pattern in $w$ which is also forbidden. □

The enumeration problem for this $(2,1)$-pop stack of depth 2 can be considered for permutations, words, and specifically inversion sequences. The permutations of length $n$ avoiding $231, 312, 321$, i.e. $120, 201, 210$, are known as the *layered permutations* where each layer can have size one or two. Note that layered permutations are exactly the permutations that avoid $231, 312$. Furthermore, the restriction of avoiding $321$ forces each layer (consecutive decreasing subsequence) to have size at most two. These restricted layered permutations are enumerated by the Fibonacci numbers as shown by Simion and Schmidt 1985. As a consequence of their result and Theorem 3.23 we have the following corollary.

**Corollary 3.25.** *(Simion and Schmidt) The number of permutations of length $n$ sortable by a $(2,1)$-pop stack of depth 2 is the $(n+1)^{st}$ Fibonacci number $F_{n+1}$.*

Burstein 1988, Theorem 5.1 enumerated the words of length $n$ over an alphabet $[k]$ avoiding $120, 201, 210$.

**Theorem 3.26.** *(Burstein) The generating function for the number of words of length $n$ over an alphabet $[k]$ sortable by a $(2,1)$-pop stack of depth 2 is given by the coefficient of $x^n y^k$ in $F(x,y)$ where*

$$F(x,y) = \frac{(1-x)(1-2x) - ((1-x)(1-2x) + x^2)y}{(1-x)(1-2x) - (1-x)(2-3x)y + (1-2x)y^2}.$$

**Algorithm 3** $(2,1)$-Pop Stack of Depth 2 Sorting Algorithm

$w = w_1 w_2 \cdots w_n$ is a word of length $n$.
$PS2$ is a $(2,1)$-Pop Stack of Depth 2 .
$O$ is an empty array.
$i = 1$ is the index of the next entry in the input.
$s = 0$ is the counter for the number of copies of the smallest value in the $(2,1)$-Pop Stack of Depth 2.
**procedure** POPSTACKSORT2$(w, n)$
    **while** $i < n + 1$ **do**
        **if** $PS2$ is empty **then**
            push $w_i$ to the top of $PS2$
            $i = i + 1$
            $s = 1$
        **else if** $w_i =$ the top/smallest element(s) in $PS2$ **then**
            push $w_i$ to the top of $PS2$
            $i = i + 1$
            $s = s + 1$
        **else if** $w_i =$ the larger element(s) in $PS2$ **then**
            push $w_i$ to the $(s + 1)$st position of $PS2$
            $i = i + 1$
        **else if** there are already two distinct values in $PS2$ **then**
            **while** $PS2$ is nonempty **do**
                pop the top entry of $PS2$ to the end of $O$
            **end while**
        **else if** $p_i <$ top element of $PS2$ **then**
            push $p_i$ to the top of $PS2$
            $i = i + 1$
            $s = 1$
        **else**
            **while** $PS2$ is nonempty **do**
                pop the top entry of $PS2$ to the end of $O$
            **end while**
        **end if**
    **end while**
**return** $O$
**end procedure**

The case for the enumeration of sortable inversion sequences by a $(2,1)$-pop stack of depth 2 (in the form of avoidance of $120, 201, 210$ or otherwise) does not appear in the literature prior to this work, but can be solved using generating trees as demonstrated in Section 4.1. Specifically, the generating function for these sortable inversion sequences is given in Theorem 4.3.

# 4 The generating tree method

## *4.1 A formula for the generating function for the number of inversion sequences of length $n$ that avoid $120, 201, 210$*

To enumerate the inversion sequences of length $n$ avoiding $120, 201, 210$, i.e. the inversion sequences sortable by a $(2,1)$-pop stack of depth 2, we use the generating tree method shown by Kotsireas, Yıldırım, and the first author 2024. See the recent paper of Pantone 2024 for another nice description of using succession rules to enumerate pattern avoiding inversion sequences.

Let $\mathbf{A} = \cup_{n=0}^{\infty} \mathbf{I}_n(120, 201, 210)$. We construct a pattern-avoidance tree $\mathbf{T}$ for the class of pattern-avoiding inversion sequences $\mathbf{A}$ as follows.

1. The root is $0$ (inversion sequence with one letter), that is, $0 \in \mathbf{T}$ at level $1$.
2. Recursively construct the nodes at level $n+1$ of tree $\mathbf{T}$ from the nodes at level $n$ by inserting a new letter at the end of the inversion sequence. That is, the children of $e = e_0 \cdots e_n \in \mathbf{I}_n \cap \mathbf{A}$ are the inversion sequences $e^* = e_0 \cdots e_n j$ with $j = 0, 1, \ldots, n+1$ where $e^* \in \mathbf{I}_{n+1} \cap \mathbf{A}$.

Now, we relabel the vertices of the tree $\mathbf{T}$ as follows. Define $\mathbf{T}(e)$ to be the subtree consisting of the inversion sequence $e$ as the root and its descendants in $\mathbf{T}$. We say that $e$ is *equivalent* to $e'$, denoted by $e \sim e'$, if and only if $\mathbf{T}(e) \cong \mathbf{T}(e')$ (in the sense of plane trees). Define an order on the nodes of any tree to be from top to bottom, and within a level from left to right. Denote by $\mathbf{T}'$ the tree $\mathbf{T}$ where we have replaced, in order, each node $e \in T$ by the first node $e' \in \mathbf{T}$ (from top to bottom and within a level from left to right) in $\mathbf{T}$ such that $\mathbf{T}(e) \cong \mathbf{T}(e')$ assuming that such a node $e'$ exists.

**Definition 4.1.** *A* succession rule*, denoted $x \rightsquigarrow y_1, y_2, \ldots, y_k$, will indicate that the set $\{y_1, y_2, \ldots, y_k\}$ will be such that the generating subtrees rooted at $y_1, y_2, \ldots, y_k$ are equivalent to those rooted at $z_1, z_2, \ldots, z_k$ where $\{z_1, z_2, \ldots, z_k\}$ is the complete set of children of $x$.*

Note that exponents are used in two ways below based on context. Specifically, when describing succession rules, an exponent of $j$ on a child indicates the number of copies of that child the parent has. For example, in Lemma 4.2, $b_{m,j}$ has $j$ children of the form $b_{m+2-j,1}$. However, when describing the individual nodes that make up the tree, the exponent refers to copies of the letter in the word (in particular, inversion sequence). Again, referring to Lemma 4.2, we have $a_m$ is the word made up of exactly $m$ 0s.

**Lemma 4.2.** *The generating tree $\mathbf{T}'$ is given by root $0$ and the following succession rules*

$$\begin{aligned} a_m &\rightsquigarrow a_{m+1}, b_{m,1}, \ldots, b_{m,m}, \\ b_{m,j} &\rightsquigarrow (b_{m+2-j,1})^j, b_{m+1,j}, b_{m+1-j,1}, \ldots, b_{m+1-j,m+1-j}, \end{aligned}$$

*where $a_m = 0^m$ and $b_{m,j} = a_m j$ for $1 \le j \le m$.*

**Proof:** We label the inversion sequence $0 \in \mathbf{I}_0$ by $a_1$. Thus, $a_1 \rightsquigarrow a_2, b_{1,1}$. More generally, by the definitions, the children of $a_m \in \mathbf{T}'$ are $a_m0, a_m1, \ldots, a_mm$, which can also be denoted $a_{m+1}, b_{m,1}, \ldots, b_{m,m}$, respectively. Thus, the rule $a_m \rightsquigarrow a_{m+1}, b_{m,1}, \ldots, b_{m,m}$ holds.

Also, the children of $b_{m,j} \in \mathbf{T}'$ are $b_{m,j}0, b_{m,j}1, \ldots, b_{m,j}(m+1)$. Notice $\mathbf{T}(b_{m,j}k) \cong \mathbf{T}(b_{m+2-j,1})$ with $k = 0, 1, \ldots, j-1$. To show this, we map any inversion sequence $\pi = 0^m jk\pi' \in \mathbf{I}_n(120, 201, 210)$ to $0^{m+2-j}1\pi''$, where $\pi''$ is obtained from $\pi'$ (there are no letters in $\pi'$ belonging to the set $\{0, \ldots, k-1, k+1, \ldots, j-1\}$) by replacing each letter $x \geq j$ by $x+1-j$ and replacing the letter $k$ by 0. Hence, we see that $\pi \in \mathbf{I}_n(120, 201, 210)$ if and only if $0^{m+2-j}1\pi'' \in I_{n+2-j}(120, 201, 210)$.

As none of our patterns have repeated letters, it is not hard to see $\mathbf{T}(b_{m,j}j) \cong \mathbf{T}(b_{m+1,j})$. Here we map any inversion sequence $0^m jj\pi' \in \mathbf{I}_n$ to $0^{m+1}j\pi' \in I_n$ with $1 \leq j \leq m$, so this map respects the rule of avoiding $120, 201, 210$.

In the last cases when $k = j+1, j+2, \ldots, m+1$, we have a similar mapping as with the smaller $k$ values. Specifically, $\mathbf{T}(b_{m,j}k) \cong \mathbf{T}(b_{m+1-j,k-j})$. To see this, again, let $\pi = 0^m jk\pi' \in I_n(120, 201, 210)$. Since $\pi$ avoids 120, we see that each letter of $\pi'$ is at least $j$. Let $\pi''$ be the obtained sequence from $\pi'$ by decreasing each nonzero letter of $\pi'$ by $j$. Then the map from $\pi = 0^m jk\pi' \in \mathbf{I}_n(120, 201, 210)$ to $0^{m+1-j}(k-j)\pi'' \in \mathbf{I}_{n-j}(120, 201, 210)$ is a bijection. Hence, we have the following rule $b_{m,j} \rightsquigarrow (b_{m+2-j,1})^j, b_{m+1,j}, b_{m+1-j,1}, \ldots, b_{m+1-j,m+1-j}$, which completes the proof. □

Define $A_m(x)$ (respectively, $B_{m,j}(x)$) to be the generating function for the number of nodes at level $n \geq 1$ for the subtree of $\mathbf{T}(B; a_m)$ (respectively, $\mathbf{T}(B; b_{m,j})$), where the root stays at level 1. Thus, by Lemma 4.2, we have

$$A_m(x) = x + xA_{m+1}(x) + x\sum_{j=1}^{m} B_{m,j}(x), \tag{4}$$

$$B_{m,j}(x) = x + jxB_{m+2-j,1}(x) + xB_{m+1,j}(x) + x\sum_{k=1}^{m+1-j} B_{m+1-j,k}(x), \tag{5}$$

for all $1 \leq j \leq m$.

In order to solve the above recurrence relations, we define

$$A(v) = \sum_{m\geq 1} A_m(x)v^{m-1},$$

$$B(v,u) = \sum_{m\geq 1}\sum_{j=1}^{m} B_{m,j}(x)u^{m-j}v^{m-1}, \text{ and}$$

$$C(v) = \sum_{m\geq 1} B_{m,1}(x)v^{m-1}.$$

Thus Equations (4)-(5) can be written as

$$A(v) = \frac{x}{1-v} + \frac{x}{v}(A(v) - A(0)) + xB(v,1), \tag{6}$$

$$B(v,u) = \frac{x}{(1-v)(1-uv)} + \frac{x}{uv(1-v)^2}(C(vu) - C(0)) + \frac{x}{uv}(B(v,u) - B(v,0)) + \frac{x}{1-v}B(uv,1), \tag{7}$$

$$C(v) = \frac{x}{1-v} + \frac{2x}{v}(C(v) - C(0)) + xB(v,1), \tag{8}$$

where Equation (8) is the translation of (5) with $j = 1$.

To solve the system from Equations (7)-(8), we make the following guess based on the first terms of the generating functions $B(v,1)$, $C(v)$, and $A(v)$:

$$B(v,1) = \frac{1}{(1-v)^2}C(v) - \frac{v}{(1-v)^2}A(v). \tag{9}$$

Next, we solve the system from Equations (7)-(9), which satisfies the original system from Equations (7)-(8). By substituting the expression of $B(v,1)$ from Equation (9) into Equation (7), and solving for $C(v)$, we obtain

$$C(v) = \frac{v^3 - 2v^2 + 2vx + v - x}{vx}A(v) + \frac{(1-v)^2}{v}A(0) + v - 1. \tag{10}$$

From here, use Equation (9) and Equation (10) to rewrite Equation (8) as

$$\begin{aligned} &\frac{2x^2 - 3x(x+1)v + (5x+1)v^2 - 2(x+1)v^3 + v^4}{v^2 x}A(v) \\ &= \frac{2x - (3x+1)v + 2(x+1)v^2 - v^3}{v^2}A(0) - \frac{2x}{v}C(0) - \frac{(v-2x)(v-1)}{v}. \end{aligned} \tag{11}$$

Let $K(v) = 2x^2 - 3x(x+1)v + (5x+1)v^2 - 2(x+1)v^3 + v^4$ be the kernel of this equation. Note that for the kernel equation $K(v) = 0$ there are four roots, say $v_1, v_2, v_3, v_4$, where

$$\begin{aligned} v_1 &= 1 + \frac{-1+\sqrt{5}}{2}x + \frac{-5+4\sqrt{5}}{5}x^2 + \cdots, \\ v_2 &= 1 + \frac{-1-\sqrt{5}}{2}x + \frac{-5-4\sqrt{5}}{5}x^2 + \cdots, \\ v_3 &= 2x + 2x^2 + 10x^3 + \cdots, \\ v_4 &= x - x^3 - x^4 + \cdots. \end{aligned}$$

By taking Equation (11) with either $v = v_3$ or $v = v_4$, we obtain a system of equations in $A(0)$ and $C(0)$.

We solve this system, obtaining

$$\begin{aligned}
A(0) &= -\frac{(v_1+v_2-2x-1)v_1v_2}{v_1v_2(v_1+v_2)-2(1+x)v_1v_2+2x},\\
C(0) &= -\frac{v_1v_2(v_2-1)(v_1-1)-(v_1^2v_2+v_1v_2^2-2v_1^2-3v_1v_2-2v_2^2+2v_1+2v_2)x}{2x(v_1v_2(v_1+v_2)-2(1+x)v_1v_2+2x)}\\
&\quad+\frac{2(v_1v_2-2v_1-2v_2+2)x^2}{2x(v_1v_2(v_1+v_2)-2(1+x)v_1v_2+2x)}..
\end{aligned}$$

Using the expressions of $A(0), C(0)$, we obtain an explicit formula for $A(v)$ from Equation (11). We can then obtain an explicit formula for $C(v)$ from Equation (10). Then we can use that formula to get an explicit formula for $B(v,1)$ from Equation (9). Finally, we have the information we need to obtain an explicit formula for $B(v,u)$ from Equation (7). We omit the presentations of the expressions $A(v), C(v), B(v,1), B(v,u)$ because they are very lengthy. However, these expressions satisfy the system of Equations (6)-(8). Hence, we can state the following result.

**Theorem 4.3.** *The generating function for the number of inversion sequences in* $\mathbf{I}_n$ *that avoid* $120, 201, 210$ *and thus are sortable by a* $(2,1)$*-pop stack of depth* $2$ *is given by*

$$\begin{aligned}
A(0) &= -\frac{(v_1+v_2-2x-1)v_1v_2}{v_1v_2(v_1+v_2)-2(1+x)v_1v_2+2x}\\
&= x+2x^2+6x^3+23x^4+101x^5+484x^6+2468x^7+13166x^8+72630x^9+411076x^{10}\\
&+2374188x^{11}+13938018x^{12}+82932254x^{13}+499031324x^{14}+3031610924x^{15}\\
&+18568429963x^{16}+114541486785x^{17}+710973143614x^{18}+4437415155234x^{19}\\
&+27831038618735x^{20}+175318861863701x^{21}+1108762012137252x^{22}\\
&+7037137177329268x^{23}+44808588430903068x^{24}+\cdots.
\end{aligned}$$

## *4.2 The generating tree for the pop stack sortable inversion sequences*

As in the previous case, one can show that the generating tree (based on the algorithm given in Kotsireas et al. (2024)) of $I_n(120, 201, 1010)$, that is, of the pop stack sortable inversion sequences, is given by a root $a_1$ and the rules

$$\begin{aligned}
a_m &\rightsquigarrow a_{m+1}, b_{m,1}, \ldots, b_{m,m},\\
b_{m,j} &\rightsquigarrow c_{m+1-j,1}, \ldots, c_{m,j}, b_{m+1,j}, b_{m+1-j,1}, \ldots, b_{m+1-j,m+1-j},\\
c_{m,j} &\rightsquigarrow c_{m+2-j,1}, \ldots, c_{m+1,j}, a_{m+3-j}, b_{m+2-j,1}, \ldots, b_{m+2-j,m+2-j},
\end{aligned}$$

where $a_m = 0^m$, $b_{m,j} = 0^m j$ and $c_{m,j} = 0^m j(j-1)$.

For any sequence of nodes $f_{m,j}$, we define $F_{m,j}(x)$ as the generating functions for the number of nodes

in the subtrees $\mathcal{T}(B; f_{m,j})$, where $f \in \{a, b, c\}, F \in \{A, B, C\}$. Define

$$A(v) = \sum_{m\geq 1} A_m(x)v^{m-1},$$
$$B(v,u) = \sum_{m\geq 1}\sum_{i=1}^{m} B_{m,i}v^{m-1}u^{m-i},$$
$$C(v,u) = \sum_{m\geq 1}\sum_{i=1}^{m} C_{m,i}v^{m-1}u^{m-i}.$$

Then the generating function for the case 120,201,1010 is $\frac{x}{1-x} + xB(x,1)$, where $B(x,v)$ satisfies

$$\begin{aligned}
A(v) &= \frac{x}{1-v} + \frac{x}{v}(A(v) - A(0)) + xB(v,1),\\
B(v,u) &= \frac{x}{(1-v)(1-uv)} + \frac{x}{uv}(B(v,u) - B(v,0)) + \frac{x}{1-v}C(v,u) + \frac{x}{1-v}B(vu,1),\\
C(v,u) &= \frac{x}{(1-v)(1-uv)} + \frac{x}{uv(1-v)}(C(v,u) - C(v,0))\\
&\quad + \frac{x}{u^2v^2(1-v)}(A(uv) - A(0) - uv\frac{\partial}{\partial v}A(v)\mid_{v=0}) + \frac{x}{uv(1-v)}(B(vu,1) - B(0,0)).
\end{aligned}$$

By applying this system 20 times starting from $A(x,v) = B(x,v,u) = C(x,v,u) = 0$, we obtain the first 20 coefficients of $A(0)$ as

$$\begin{aligned}
A(0) &= x + 2x^2 + 6x^3 + 23x^4 + 101x^5 + 485x^6 + 2488x^7 + 13414x^8 + 75126x^9 + 433546x^{10}\\
&\quad + 2563335x^{11} + 15461646x^{12} + 94835817x^{13} + 589997530x^{14} + 3715451178x^{15}\\
&\quad + 23645541066x^{16} + 151874732111x^{17} + 983428159871x^{18}\\
&\quad + 6413887925931x^{19} + 42100271440339x^{20} + \cdots .
\end{aligned}$$

## 4.3 Inversion sequences sortable by a pop stack of depth two

Recall from Corollary 3.11 that the inversion sequences sorted by a pop stack of depth two are precisely those which avoid $P = \{120, 201, 210, 1010\}$. We can define the generating tree $T_P$ for inversion sequences that avoid $P$ as having a root $a_1$ and satisfying the following rules:

$$\begin{aligned}
a_m &\rightsquigarrow a_{m+1}, b_{m,1}, \ldots, b_{m,m},\\
b_{m,j} &\rightsquigarrow (c_{m+1-j})^j, b_{m+1,j}, b_{m+1-j,1}, \ldots, b_{m+1-j,m+1-j}, \quad 1 \leq j \leq m,\\
c_m &\rightsquigarrow c_{m+1}, a_{m+2}, b_{m+1,1}, \ldots, b_{m+1,m+1},
\end{aligned}$$

where $a_m = 0^m, b_{m,j} = 0^m j$, and $c_m = 0^m 10$. We label the inversion sequence 0 by $a_1$, so the root of the tree $T_P$ is indeed $a_1$. Now, let us show that the succession rules hold in $T_P$. Based on the algorithm introduced in [17] (for more examples and proofs, see Callan et al. (2023); Callan and Mansour (2023)), we have the following:

- The children of $a_m$ in $T_P$ are $a_m j$ with $j = 0, 1, \ldots, m$, which are equal to $a_{m+1}, b_{m,1}, \ldots, b_{m,m}$. Hence, the first succession rule holds.

- The children of $b_{m,j}$ in $T_P$ are $0^m ji$ with $i = 0, 1, \ldots, m+1$. But, note that the set of inversion sequences of the form $0^m ji\pi'$ with $0 \leq i \leq j-1$ that avoid $P$ can be mapped bijectively to the set of inversion sequences of the form $0^{m+1-j}10\pi''$ that avoid $P$ (by mapping each letter $i$ to 0 and each letter $s \geq j$ to $s+1-j$ and writing the first initial $0^m$ as $0^{m+1-j}$). Hence, $T_P(0^m ji) \cong T_P(c_{m+1+j})$, for all $i = 0, 1, \ldots, j-1$.

  Now, let us look at the remaining children of $b_{m,j}$ in $T_P$, which are $0^m ji$ with $i = j, j+1, j+2, \ldots, m+1$. Clearly, an inversion sequence $0^m jj\pi'$ avoids $P$ if and only if the inversion sequence $0^{m+1}j\pi'$ avoids $P$, so $T_P(0^m jj) \cong T_P(0^{m+1}j)$. Next consider the inversion sequence $\pi = 0^m ji\pi'$ with $j+1 \leq i \leq m+1$. Notice that $\pi$ avoids $P$ if and only if $0^{m+1-j}(i-j)\pi''$ avoids $P$ (by mapping each letter $s \geq j$ to $s-j$ and writing the initial $0^m j$ as $0^{m+1-j}$). Hence, the children of $b_{m,j}$ in $T_P$ are $(c_{m+1-j})^j, b_{m+1,j}, b_{m+1-j,1}, \ldots, b_{m+1-j,m+1-j}$, which proves the second succession rule.

- The children of $c_m$ in $T_P$ are $0^m 100, 0^m 101, \ldots, 0^m 10(m+2)$. Clearly, an inversion sequence $0^m 100\pi'$ avoids $P$ if and only if $0^{m+1}10\pi'$ avoids $P$. Also, an inversion sequence $0^m 10i\pi'$ avoids $P$ if and only if $0^{m+2}\pi''$ avoids $P$ (here $\pi''$ is obtained by mapping each letter $s$ in $i\pi'$ to $s-1$). Thus the children of $c_m$ in $T_P$ are $c_{m+1}, a_{m+2}, b_{m+1,1}, \ldots, b_{m+1,m+1}$, which shows that the third succession rule holds.

# 5 Open Problems

There are many directions left to continue the study of sortable inversion sequences. The most obvious question is the elusive enumeration problem for the inversion sequences of length $n$ which avoid 120. Already interesting as a standalone pattern avoidance question, the fact that these inversion sequences are exactly those sortable by the classic stack sorting algorithm by Knuth 1969 increases the intrigue. The first author and Shattuck 2015, Section 4 obtained a recurrence relation for these inversion sequences yielding a functional equation in three variables (sequence number A263778 in OEIS Sloane (2025)). We note that the construction method used here was later generalized by Testart 2024 as part of his work to complete the enumeration of 22 open cases of pattern-avoiding inversion sequences.

**Open Question 5.1.** *Is there a nicer combinatorial formula for the number of inversion sequences of length $n$ are stack sortable, i.e. avoid* $120$*?*

Next, while we have classified the pop stack sortable inversion sequences and words and provided a generating tree for the inversion sequence case in Section 4.2, the following combinatorial enumeration is also still open.

**Open Question 5.2.** *How many inversion sequences of length $n$ are pop stack sortable, i.e. avoid* $120, 201, 1010$*?*

And the cases restricting pop stacks to particular depths in an analogous way to Elder (2006); Elder and Goh (2018, 2021); Elder et al. (2015) and/or the expansion in allowable pushes analogous to Atkinson (1998) can be interesting for words and inversion sequences.

**Open Question 5.3.** *Classify and enumerate the inversion sequences of length $n$ that are $(r, 1)$-pop stack sortable for pop stacks of depth $k$ for $r \geq 2$ and/or $k \geq 2$.*

Another avenue of exploration is to consider other sorting machines or algorithms. For instance, the authors considered a sorting algorithm on a single stack prioritizing outputting the next correct entry of the sequence until no other moves were available and then either returning the elements in order to be sorted again Mansour et al. (2019a) or reversing the output for the next pass Mansour et al. (2019b). In particular, the 2-reverse-pass sortable permutation class has a reasonably small basis, namely $\{2413, 2431, 23154\} = \{1302, 1320, 12043\}$ that may prove tractable.

**Open Question 5.4.** *How many inversion sequences of length $n$ are 2-reverse-pass sortable?*

## Acknowledgements

The authors are grateful for the helpful comments from the referees.

## References

Andrei Asinowski, Cyril Banderier, Sara Billey, Benjamin Hackl, and Svante Linusson. Pop-stack sorting and its image: permutations with overlapping runs. *Acta Math. Univ. Comenian. (N.S.)*, 88(3):395–402, 2019. ISSN 0862-9544,1336-0310.

Mike D. Atkinson. Generalized stack permutations. *Combin. Probab. Comput.*, 7:239–246, 1998.

Mike D. Atkinson and Jörg-Rüdiger Sack. Pop-stacks in parallel. *Inform. Process. Lett.*, 70:63–67, 1999.

Mike D. Atkinson, Maximillian M. Murphy, and Nik Ruškuc. Sorting with two ordered stacks in series. *Theoret. Comput. Sci.*, 289(1):205–223, 2002. ISSN 0304-3975.

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

Alexander Burstein. *Enumeration of Words with Forbidden Patterns*. PhD thesis, University of Pennsylvania, 1988.

David Callan and Toufik Mansour. Inversion sequences avoiding quadruple length-3 patterns. *Integers*, 23:Paper No. A78, 64, 2023. ISSN 1553-1732.

David Callan, Vít Jelínek, and Toufik Mansour. Inversion sequences avoiding a triple of patterns of 3 letters. *Electron. J. Combin.*, 30(3):Paper No. 3.19, 39, 2023. ISSN 1077-8926. doi: 10.37236/11603.

Giulio Cerbai. Sorting Cayley permutations with pattern-avoiding machines. *Australas. J. Combin.*, 80: 322–341, 2021. ISSN 1034-4942,2202-3518.

Giulio Cerbai, Anders Claesson, and Luca Ferrari. Stack sorting with restricted stacks. *J. Combin. Theory Ser. A*, 173:105230, 19, 2020. ISSN 0097-3165,1096-0899.

Anders Claesson and Bjarki Ágúst Guðmundsson. Enumerating permutations sortable by $k$ passes through a pop-stack. *Adv. in Appl. Math.*, 108:79–96, 2019. ISSN 0196-8858,1090-2074.

Sylvie Corteel, Megan A. Martinez, Carla D. Savage, and Michael Weselcouch. Patterns in inversion sequences I. *Discrete Math. Theor. Comput. Sci.*, 18(2):Paper No. 2, 21, 2016. ISSN 1365-8050.

Colin Defant and Nathan Williams. Crystal pop-stack sorting and type $A$ crystal lattices. *European J. Combin.*, 103:Paper No. 103514, 19, 2022. ISSN 0195-6698.

Murray Elder. Permutations generated by a stack of depth 2 and an infinite stack in series. *Electron. J. Combin.*, 13:Research paper 68, 12 pp., 2006.

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

Murray Elder and Yoong Kuan Goh. $k$-pop stack sortable permutations and 2-avoidance. *Electron. J. Combin.*, 28(1):Paper No. 1.54, 15, 2021.

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

Letong Hong. The pop-stack-sorting operator on Tamari lattices. *Adv. in Appl. Math.*, 139:Paper No. 102362, 13, 2022. ISSN 0196-8858.

Donald E. Knuth. *The art of computer programming. Volume 1*. Addison-Wesley Publishing Co., Reading, Mass., 1969. Fundamental Algorithms.

Ilias Kotsireas, Toufik Mansour, and Gökhan Yıldı rım. An algorithmic approach based on generating trees for enumerating pattern-avoiding inversion sequences. *J. Symbolic Comput.*, 120:Paper No. 102231, 18, 2024. ISSN 0747-7171,1095-855X.

Charles-Ange Laisant. Sur la numération factorielle, application aux permutations. *Bull. Soc. Math. France*, 16:176–183, 1888.

Derrick H. Lehmer. Teaching combinatorial tricks to a computer. In *Proc. Sympos. Appl. Math., Vol. 10*, pages 179–193. American Mathematical Society, Providence, R.I., 1960.

Zhicong Lin. Patterns of relation triples in inversion and ascent sequences. *Theoret. Comput. Sci.*, 804: 115–125, 2020. ISSN 0304-3975,1879-2294.

Zhicong Lin and Dongsu Kim. Refined restricted inversion sequences. *Ann. Comb.*, 25(4):849–875, 2021. ISSN 0218-0006,0219-3094.

Toufik Mansour and Mark Shattuck. Pattern avoidance in inversion sequences. *Pure Math. Appl. (PU.M.A.)*, 25(2):157–176, 2015. ISSN 1218-4586,1788-800X.

Toufik Mansour, Howard Skogman, and Rebecca Smith. Passing through a stack $k$ times. *Discrete Math. Algorithms Appl.*, 11(1):1950003, 22, 2019a. ISSN 1793-8309.

Toufik Mansour, Howard Skogman, and Rebecca Smith. Passing through a stack $k$ times with reversals. *European J. Combin.*, 81:309–327, 2019b. ISSN 0195-6698.

Megan Martinez and Carla Savage. Patterns in inversion sequences II: inversion sequences avoiding triples of relations. *J. Integer Seq.*, 21(2):Art. 18.2.2, 44, 2018.

Jay Pantone. The enumeration of inversion sequences avoiding the patterns 201 and 210. *Enumer. Comb. Appl.*, 4(4):Paper No. S2R25, 12, 2024. ISSN 2710-2335.

Lara Pudwell and Rebecca Smith. Two-stack-sorting with pop stacks. *Australas. J. Combin.*, 74:179–195, 2019. ISSN 1034-4942.

Rodica Simion and Frank W. Schmidt. Restricted permutations. *European J. Combin.*, 6(4):383–406, 1985. ISSN 0195-6698.

Neil J. A. Sloane. The Online Encyclopedia of Integer Sequences. `http://oeis.org`, 2025.

Rebecca Smith. Two stacks in series: A decreasing stack followed by an increasing stack. *Ann. Comb.*, 18:359–363, 2014.

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

Benjamin Testart. Completing the enumeration of inversion sequences avoiding one or two patterns of length 3. arXiv:2407.07701, 2024. URL `https://arxiv.org/abs/2407.07701`.

Akiva M. Yaglom and Isaak M. Yaglom. *Challenging Mathematical Problems with Elementary Solutions*, volume 1. Dover Publications Inc., New York, 1987.

Chunyan Yan and Zhicong Lin. Inversion sequences avoiding pairs of patterns. *Discrete Math. Theor. Comput. Sci.*, 22(1):Paper No. 23, 35, 2020–2021. ISSN 1365-8050.