Counting pop-stacked permutations in polynomial time

Anders Claesson
Division of Mathematics
The Science Institute
University of Iceland
akc@hi.is

Bjarki Ágúst Guðmundsson
Department of Computer Science
Reykjavík University
Reykjavík, Iceland
bjarkig@ru.is

Jay Pantone
Department of Mathematical and Statistical Sciences
Marquette University
Milwaukee, WI, USA
jay.pantone@marquette.edu

Abstract

Permutations that can be sorted greedily by one or more stacks having various constraints have been studied by a number of authors. A *pop-stack* is a greedy stack that must empty all entries whenever popped. Permutations in the image of the pop-stack operator are said to be *pop-stacked*. Asinowki, Banderier, Billey, Hackl and Linusson recently investigated these permutations and calculated their number up to length 16. We give a polynomial-time algorithm to count pop-stacked permutations up to a fixed length and we use it to compute the first 1000 terms of the corresponding counting sequence. With the 1000 terms we apply a pair of computational methods to prove some negative results concerning the nature of the generating function for pop-stacked permutations and to empirically predict the asymptotic behavior of the counting sequence using differential approximation.

1 Pop-stacked permutations

The abstract data type known as a *stack* has two operations: *push* adds an element to the top of the stack; *pop* removes the top element from the stack. A *pop-stack* is a variation of this introduced by Avis and Newborn [3] in which the pop operation empties the entire stack.

MSC 2010: 05A15, 65Q30, 68R05

Let $\pi = a_1a_2\dots a_n$ be a permutation of $[n] = \{1, 2, \dots, n\}$. An *ascending run of length* d in π is a maximal sequence of consecutive ascending letters $a_i < a_{i+1} < \cdots < a_{i+d-1}$, and a *descending run* is defined similarly. For instance, the ascending runs of $\pi = 617849235$ are 6, 178, 49 and 235; its descending runs are 61, 7, 84, 92, 3 and 5.

Let $P(\pi)$ be the result of greedily sorting π using a pop-stack subject to the constraint that elements on the pop-stack are increasing when read from the top to the bottom of the stack. In other words, if we factor π into its descending runs $\pi = D_1D_2\dots D_m$, then $P(\pi)$ is obtained by reversing each of those runs: $P(\pi) = D_1^rD_2^r\dots D_m^r$. For instance, using vertical bars as a visual cue to separate adjacent descending runs in the input permutations as well as to separate their images in the output permutations, $P(5321|764) = 1235|467$ and $P(61|7|84|92|3|5) = 16|7|48|29|3|5$. A permutation π is said to be sortable by a pop-stack if $P(\pi)$ is the identity permutation. More generally, π is said to be sortable by k passes through a pop-stack if $P^k(\pi)$ is the identity permutation. Several types of sorting networks involving pop-stacks have been studied: Atkinson and Sack [2] considered two pop-stacks in parallel, Pudwell and Smith [12] investigated two pop-stacks in series, while Smith and Vatter [15] took a mixed approach, placing a pop-stack in series with an ordinary stack. Claesson and Guðmundsson [4] later showed that the generating function for the number of permutations of $[n]$ that are sortable by k passes through a pop-stack is always rational.

Asinowski et al. [1] defined that σ is *pop-stacked* if $\sigma = P(\pi)$ for some permutation π, and gave the following theorem.

Theorem 1 (Asinowski et al. [1])**.** *A permutation is pop-stacked if and only if for each pair* (R_i, R_{i+1}) *of its adjacent ascending runs* $\min R_i < \max R_{i+1}$.

They further showed that the generating function for pop-stacked permutations of $[n]$ with exactly k ascending runs is rational for each k. Enumerating pop-stacked permutations without this restriction is, however, an open problem. Asinowski et al. initiated an investigation into this by calculating the number of pop-stacked permutations of length $n = 1, \dots, 16$, adding the resulting sequence to the OEIS [14] as A307030. In the following section, we give an efficient algorithm for counting pop-stacked permutations, expanding the sequence up to $n = 1000$. While the algorithm and the augmented sequence could give additional insight into the structure of pop-stacked permutations, finding a generating function or a closed form solution to their enumeration remains an open problem. Section 3 gives experimental data in this direction.

2 Polynomial-time counting algorithm

A *ballot*, alternatively known as an ordered set partition, is a collection of pairwise disjoint nonempty sets, referred to as blocks, where the blocks are assigned some total ordering. Any permutation can be seen as a ballot by decomposing it into its ascending runs. The permutation $\pi = 617849235$ would then be viewed as the ballot $\{6\}\{1,7,8\}\{4,9\}\{2,3,5\}$. Conversely, a ballot $B_1B_2\dots B_k$

represents a permutation in this manner if, and only if, $\max B_i > \min B_{i+1}$ for each i in $[k-1]$. Thus, the ballots corresponding to pop-stacked permutations are precisely those such that

$$\max B_i > \min B_{i+1} \quad \text{and} \quad \min B_i < \max B_{i+1},$$

where the latter inequality comes from Theorem 1. In other words, the intervals between the smallest and largest elements of each pair of adjacent blocks overlap,

$$[\min B_i, \max B_i] \cap [\min B_{i+1}, \max B_{i+1}] \neq \emptyset,$$

and we call these ballots *overlapping*; here, $[a,b]$ denotes the interval $\{a, a+1, \dots, b\}$. Let $F[U]$ be the set of overlapping ballots whose underlying set is U. As an example,

$$F[\{1,2,3\}] = \big\{\{1,2,3\}, \{2\}\{1,3\}, \{1,3\}\{2\}\big\}.$$

Let $F_{c,d}[U]$ denote the subset of $F[U]$ whose last block, B, is such that $c = \min B$ and $d = \max B$. Clearly, if $c > d$ then $F_{c,d}[U] = \emptyset$. Also,

$$F[U] \;=\; \bigcup_{c,d\,\in\, U} F_{c,d}[U].$$

Fix values of c and d and consider a ballot $B_1 B_2 \dots B_k$ in $F_{c,d}[U]$. If $c = \min U$ and $d = \max U$, then one possibility is that $k = 1$ and there is a single block consisting of all elements of U. Assume now that $k \geq 2$ so that we are in the more typical case when there are two or more blocks. By definition, the last block, B_k, satisfies $c = \min B_k$ and $d = \max B_k$, or expressed differently $\{c,d\} \subseteq B_k \subseteq [c,d]$. Let $a = \min B_{k-1}$ and $b = \max B_{k-1}$. The blocks B_{k-1} and B_k overlap if, and only if, $a < d$ and $b > c$. Thus

$$F_{c,d}[U] \;=\; \chi_{c,d}[U] \;\cup \bigcup_{\substack{\{c,d\}\,\subseteq\, B \,\subseteq\, [c,d] \\ a,b\,\in\, U\setminus B \\ a<d \,\wedge\, b>c}} F_{a,b}[U \setminus B]\, B, \tag{1}$$

where $F_{a,b}[U \setminus B]B$ is the set $\big\{wB : w \in F_{a,b}[U \setminus B]\big\}$, and

$$\chi_{c,d}[U] = \begin{cases} \{U\} & \text{if } c = \min U \text{ and } d = \max U, \\ \emptyset & \text{otherwise.} \end{cases}$$

We now turn to counting. Let $f(n)$ be the number of overlapping ballots of $[n]$. That is, $f(n) = |F[n]|$ in which $F[n]$ is short for $F[\{1,\dots,n\}]$. Also, let $f_{c,d}(n) = |F_{c,d}[n]|$. If $c > d$ then $f_{c,d}(n) = 0$. Otherwise we shall use the recursive decomposition (1) and do case analysis based on whether c and d are the same or two distinct elements.

If $c = d$, then the last block consists of a single point. In terms of (1) the ballot is written $w\{c\}$, where $w \in F_{a,b}[[n] \setminus \{c\}]$ and $a < c < b$, i.e., w contains the values $\{1, 2, \dots, c-1, c+1, \dots, n\}$. Let $s(w)$ denote the ballot resulting from subtracting 1 from each value of w greater than c, so that $s(w) \in F_{a,b}[n-1]$. There is then a clear bijection

$$F_{c,c}[n] \to \bigcup_{\substack{a \in [1,c-1] \\ b \in [c,n-1]}} F_{a,b}[n-1] \quad \text{defined by} \quad w\{c\} \mapsto s(w).$$

Thus, the number of ballots in $F_{c,c}[n]$ is

$$\sum_{a=1}^{c-1} \sum_{b=c}^{n-1} f_{a,b}(n-1).$$

Now suppose that $c < d$ and consider the number of blocks in the ballot. The case where the ballot consists of a single block only occurs when $c = 1$ and $d = n$ and in that case the ballot is unique.

If the ballot contains more than one block we may write it as wB and let $\ell = |B| - 2$. There are $\binom{d-c-1}{\ell}$ ways to choose B. Similar to the case where $c = d$, subtracting from each value of w so as to fill in the gaps after B has been removed gives a bijection mapping w to $F_{a,b}[n-\ell-2]$, where $a \leq d-\ell-2$ and $b \geq c$. Thus, the number of such ballots is

$$\sum_{\ell=0}^{d-c-1} \binom{d-c-1}{\ell} \sum_{a=1}^{d-\ell-2} \sum_{b=c}^{n-\ell-2} f_{a,b}(n-\ell-2).$$

Taking all this together, we have that

$$\begin{aligned} f_{c,d}(n) = {} & [c = 1 \wedge d = n] \\ & + [c = d] \sum_{a=1}^{c-1} \sum_{b=c}^{n} f_{a,b}(n-1) \\ & + [c < d] \sum_{\ell=0}^{d-c-1} \binom{d-c-1}{\ell} \sum_{a=1}^{d-\ell-2} \sum_{b=c}^{n-\ell-2} f_{a,b}(n-\ell-2). \end{aligned} \tag{2}$$

Here $[p]$ is the Iverson bracket: it converts the proposition p into 1 if p is satisfied, and 0 otherwise. Further, $f(n) = \sum_{a=1}^{n} \sum_{b=a}^{n} f_{a,b}(n)$.

Recurrence (2) can be augmented to count overlapping ballots with a specific number of blocks, or, equivalently, pop-stacked permutations with a specific number of ascending runs. Let $f_{c,d}(n,k)$ denote the number of overlapping ballots of $[n]$ with exactly k blocks. Then we have $f(n,k) = \sum_{a=1}^{n} \sum_{b=a}^{n} f_{a,b}(n,k)$ and

$$\begin{aligned} f_{c,d}(n,k) = {} & [c = 1 \wedge d = n \wedge k = 1] \\ & + [c = d] \sum_{a=1}^{c-1} \sum_{b=c}^{n} f_{a,b}(n-1,k-1) \\ & + [c < d] \sum_{\ell=0}^{d-c-1} \binom{d-c-1}{\ell} \sum_{a=1}^{d-\ell-2} \sum_{b=c}^{n-\ell-2} f_{a,b}(n-\ell-2,k-1). \end{aligned} \tag{3}$$

Note that there are two locations in the recurrence (2) where we have a plain two-dimensional sum over f, that is $\sum_{a=\star}^{\star}\sum_{b=\star}^{\star} f_{a,b}(\star)$, where $\star$ are fixed and not dependent on a, b or each other. We simplify these two-dimensional sums using "prefix sums". Let

$$g_{c,d}(n) = \sum_{a=1}^{c}\sum_{b=1}^{d} f_{a,b}(n)$$

In particular, $g_{c,d}(n) = 0$ if $c = 0$ or $d = 0$. Note that

$$g_{c,d}(n) = f_{c,d}(n) + g_{c-1,d}(n) + g_{c,d-1}(n) - g_{c-1,d-1}(n). \tag{4}$$

Also noting that[1]

$$\sum_{a=p}^{q}\sum_{b=r}^{s} f_{a,b}(n) = g_{q,s}(n) - g_{p-1,s}(n) - g_{q,r-1}(n) + g_{p-1,r-1}(n),$$

we can now simplify Equation (3) to

$$\begin{aligned} f_{c,d}(n) &= [c = 1 \wedge d = n] \\ &\quad + [c = d]\, \Delta_{c-1,n,c-1}(n-1) \\ &\quad + [c < d] \sum_{\ell=0}^{d-c-1} \binom{d-c-1}{\ell} \Delta_{d-2-\ell,n-2-\ell,c-1}(n-2-\ell) \end{aligned} \tag{5}$$

where $\Delta_{u,v,w}(n) = g_{u,v}(n) - g_{u,w}(n)$. We further have $f(n) = g_{n,n}(n)$. The same simplification can also be applied to the recurrence for counting by blocks.

Say we wanted to compute $f(n)$ for all $1 \le n \le N$. We can precompute binomial coefficients $\binom{n}{k}$ for all $0 \le k \le n \le N$ using the recurrence $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$. Then, using dynamic programming we can compute $f_{c,d}(n)$, $g_{c,d}(n)$ and $f(n)$ using Recurrences (4) and (5) for all $1 \le c$, $d \le n \le N$ in $O(N^4)$ time using $O(N^3)$ memory. When counting by blocks this is $O(N^5)$ time, but $O(N^3)$ memory is still sufficient.

This assumes that all arithmetic operations are $O(1)$. In reality, some of the numbers are on the order of $N!$. This means that multiprecision arithmetic has to be used, which slows down the computation considerably. One way to speed this up is to choose a set of relatively small primes whose product is greater than $N!$. For each prime p, the above computation is then carried out in the finite field $\mathbf{F}_p$. This can be done in parallel, as the computation for different primes is independent. The values of $f(n)$, which are guaranteed to be at most $N!$ for all $n \le N$, are then recovered using the Chinese Remainder Theorem.

This was used to calculate the number of pop-stacked permutations of each length up to $N = 1000$. With 286 distinct primes just under 10^9, and one CPU

[1] Intuitively, this can be thought of as finding the volume under the rectangle with vertices at (p, r) and (q, s), where $g_{c,d}$ gives the volume under the rectangle with vertices at $(1, 1)$ and (c, d). Starting with the rectangle corresponding to $g_{q,s}$, we remove the rectangles corresponding to $g_{p-1,s}$ and $g_{q,r-1}$. This removed the rectangle corresponding to $g_{p-1,r-1}$ twice, so we add it once back.

core per prime, the computation took just under an hour to complete, with each core using 3.8GiB of RAM. In a similar manner the number of pop-stacked permutations of each length up to $N = 300$ grouped by number of ascending runs were computed. Table 1 gives the number of pop-stacked permutations of each length up to $N = 45$, but the complete results, along with the code used to generate the results, can be found on GitHub [5].

3 Experimental analysis

With the first 1000 terms of the counting sequence of pop-stacked permutations now calculated, we turn to a pair of experimental techniques for an empirical analysis: *automated fitting* and *differential approximation*. Given initial terms of a counting sequence, the first of these methods searches for a generating function whose power series expansion matches the sequence, while the second predicts the asymptotic growth of the sequence.

For the counting sequence at hand, automated fitting does not conjecture a generating function, giving instead several (rigorous) negative results, while differential approximation gives very precise estimates of the asymptotic behavior.

3.1 Automated fitting for pop-stacked permutations

Let $a_0, a_1, \ldots$ be a counting sequence and $F(x) = \sum_{n \geq 0} a_n x^n$ its generating function. If $F(x)$ is a rational function, then we can write $F(x) = p(x)/q(x)$ for relatively prime polynomials $p(x), q(x) \in \mathbb{Q}[x]$; equivalently,

$$q(x)F(x) - p(x) = 0. \tag{6}$$

Conversely, suppose we are given only some initial terms $a_0, a_1, \ldots, a_n$ of a counting sequence and want to determine whether the generating function $F(x)$ of the unknown counting sequence is rational. If $F(x)$ is rational with $\max(\deg(p(x)), \deg(q(x))) = d$, then using p_i and q_i to denote the coefficients of x^i in $p(x)$ and $q(x)$ respectively, we can write Equation (6) as

$$(q_0 + q_1 x + \cdots + q_d x^d)(a_0 + a_1 x + \cdots + a_n x^n) - (p_0 + p_1 x + \cdots + p_d x^d) = 0. \tag{7}$$

Expanding the left-hand side gives a polynomial in x, and the coefficients of $x^0, x^1, \ldots, x^n$ must all equal 0. We thus have a system of $n+1$ equations in the $2d+2$ unknowns $p_0, \ldots, p_d, q_0, \ldots, q_d$. A generic system of this form is likely to have non-trivial solutions when $n \leq 2d$, and so when initial terms up to a_n are known, it is only productive to consider d such that $2d < n$.

If this system has no non-trivial solution, then we are guaranteed that $F(x)$ is not rational with numerator and denominator of degree at most d. If the system does have a non-trivial solution, then it is possible, though far from guaranteed, that

$$F(x) = \frac{p_0 + p_1 x + \cdots + p_d x^d}{q_0 + q_1 x + \cdots + q_d x^d}.$$

n	$f(n)$
1	1
2	1
3	3
4	11
5	49
6	263
7	1653
8	11877
9	95991
10	862047
11	8516221
12	91782159
13	1071601285
14	13473914281
15	181517350571
16	2608383775171
17	39824825088809
18	643813226048935
19	10986188094959045
20	197337931571468445
21	3721889002400665951
22	73539326922210382215
23	1519081379788242418149
24	32743555520207058219615
25	735189675389014372317381
26	17167470189102029106503457
27	416297325393961581614919699
28	10468759109047048511785181499
29	272663345523662949571086535201
30	7346518362495550669587951987399
31	204539324291355079758576427320853
32	5878416448467628215599958670190869
33	174223945386975482728912851110751431
34	5320106374135453888563313157982976111
35	167232974698164950641578719412434688845
36	5407019929661274797886581276653666104943
37	179677314965899717327756420597568210468933
38	6132116544121046402686046213590718114272089
39	214787281796488809444762543177377466419782267
40	7716175695131570964771559074490172330993576115
41	284131588386675257705011846785657928372695002841
42	10717718945463416620327720805595647805635809236711
43	413908527884993695909526722330319436067536797304549
44	16356508568742954048255540186930772843919017766669517
45	661053598808034620660440013405109251647269697650963759

Table 1: The number of pop-stacked permutations of each length up to $N = 45$.

The larger the difference between n and $2d$, the more confident that one can be in such a conjecture. Empirically, this is like using the first $2d$ known terms to guess the rational generating function and the remaining $n-2d$ as confirmation.

Automated fitting can be extended beyond the realm of rational generating functions. A generating function $F(x)$ is called *algebraic* if there are polynomials $P_0(x), \ldots, P_m(x) \in \mathbb{Q}[x]$ such that

$$P_m(x)F^m(x) + \cdots + P_1(x)F(x) + P_0(x) = 0,$$

called *differentially finite* (or *D-finite*) if there are polynomials $P_0(x), \ldots, P_k(x), Q(x) \in \mathbb{Q}[x]$ such that

$$P_k(x)F^{(k)}(x) + \cdots + P_1(x)F'(x) + P_0(x)F(x) + Q(x) = 0,$$

and called *differentially algebraic* (or *D-algebraic*) if there exists a $(k+2)$-variate polynomial P with coefficients in $\mathbb{Q}$ such that

$$P(x, F(x), F'(x), \ldots, F^{(k)}(x)) = 0.$$

To determine whether a generating function $F(x)$ is algebraic given some initial terms, an equation similar to (7) can be set up assuming each $P_i(x)$ has degree at most d, giving a linear system with n equations and $(m+1)(d+1)$ unknowns. In the D-finite case, the system has $(k+2)(d+1)$ unknowns. The D-algebraic case requires further assumptions about form—the ideas are similar, but not worth elaborating upon here. There are various software packages that perform fitting of this kind, including `Gfun` [13] in Maple, `Guess` [10] in Mathematica, and `Guess` [8] in FriCAS. We have used a different package, `GuessFunc`, written by the third author.

We applied automated fitting to the counting sequence of pop-stacked permutations up to length 1000, and found no conjectured rational, algebraic, D-finite, or D-algebraic form for the unknown generating function $F(x)$. From this we can conclude rigorously that, for example,

- ◇ If $F(x)$ is rational, then either the degree of the denominator or the degree of the numerator is at least 500.
- ◇ If $F(x)$ is algebraic, then the degree of algebraicity m and the maximum degree of polynomial coefficient $d = \max(\deg(P_0(x)), \ldots, \deg(P_m(x)))$ must satisfy $(m+1)(d+1) > 1000$.
- ◇ If $F(x)$ is D-finite, then the differential order k and the maximum degree of polynomial coefficient $d = \max(\deg(Q(x)), \deg(P_0(x)), \ldots, \deg(P_k(x)))$ must satisfy $(k+2)(d+1) > 1000$.

A similar negative result could be written for the D-algebraic case, although it would require further explanation of the structure of the corresponding search space.

One can also apply various transformations to the generating function before initiating the automated fitting procedure. In addition to trying to find a fit for the ordinary generating function $F(x) = \sum_{n\geq 0} a_n x^n$, we also attempted to find a fit for the exponential generating function $\sum_{n\geq 0}(a_n/n!)x^n$, the reciprocal $1/F(x)$, the compositional inverse $F(x)^{\langle -1\rangle}$, and also several combinations of these transformations. No results were found.

3.2 Automated fitting for pop-stacked permutations with a fixed number of ascending runs

Let $F_k(x)$ denote the power series for those pop-stacked permutations with precisely k ascending runs. Asinowski et al. [1] showed that these permutations are in bijection with words from a regular language that is recognized by a certain deterministic finite automaton (DFA) $\mathcal{A}_k$, proving that $F_k(x)$ is rational. Furthermore, a system of linear equations can be derived from this DFA, whose solution gives $F_k(x)$. Deriving $F_k(x)$ in this way is only practical for small values of k, however, as the number of states in $\mathcal{A}_k$ grows exponentially with k.

As mentioned earlier, Recurrence (3) permits the fast computation of the counting sequence for pop-stacked permutations with a fixed number of ascending runs. This, along with the techniques of automated fitting gives rise to a different approach for finding $F_k(x)$, albeit heuristically[2].

Using the counting sequence for pop-stacked permutations of length at most 300 with a fixed number of ascending runs, we were able to find a rational fit for each $F_k(x)$ for $k \leq 24$. We were further able to verify that the rational fits were exact for $k \leq 6$ by using the previously mentioned method based on Asinowski et al. [1]. The first four generating functions follow.

$$\begin{aligned}
F_1(x) &= \frac{x}{1-x},\\
F_2(x) &= \frac{2x^3}{(1-2x)(1-x)^2},\\
F_3(x) &= \frac{2x^4(1+3x-6x^2)}{(1-3x)(1-2x)^2(1-3x)^3},\\
F_4(x) &= \frac{2x^6(21-74x+5x^2+180x^3-144x^4)}{(1-4x)(1-3x)^2(1-2x)^3(1-x)^4}.
\end{aligned}$$

Based on this data, which can be found in full on GitHub [5], we pose the following conjecture.

Conjecture 2. *For all k, the rational generating function $F_k(x)$ can be written as*

$$F_k(x) = N_k(x) \Big/ \prod_{i=1}^{k} (1-ix)^{k-i+1},$$

where $N_k(x)$ is a polynomial of degree $k(k+1)/2$, the same degree as the conjectured denominator.

[2] Given enough terms of the sequence, automated fitting will find $F_k(x)$. The number of terms required is the sum of the degrees of the numerator and denominator of $F_k(x)$, which is not known. An upper bound is twice the number of states in $\mathcal{A}_k$, which is exponential.

3.3 Differential approximation

Differential approximation empirically estimates the asymptotic growth of a counting sequence based on its initial terms by using linear differential equations to model the unknown generating function and studying the complex singularities of solutions of those linear differential equations. Here we will only present the results of this analysis—for information about how differential approximation works we refer the reader to [6, 7].

The cornerstone of analytic combinatorics is the observation that the asymptotic behavior of a counting sequence is intimately connected to the singularities of its generating function when treated as a complex function. For example, the location of the singularities closest to the origin (the *dominant* singularities) determine the largest exponential growth factors of the counting sequence, and the nature of those singularities determines the sub-exponential behavior.

The output of differential approximation is an estimate of the location and nature (specifically, the critical exponent) of all singularities of the unknown generating function based on the given known initial terms. Typically, although not always, the dominant singularity is predicted with the highest precision, with the precision of the estimates of other singularities decreasing as distance from the origin increases. Obviously such an analysis is only experimental, but in practice the estimates given by differential approximation are incredibly accurate. In tests where the true singularity structure of a generating function is independently known, the estimates from differential approximation are rarely off by more than the last decimal place (Kahn [11] provides some related empirical analysis.)

The counting sequence of pop-stacked permutations grows superexponentially [1], implying that its generating function has a singularity at the origin. Accordingly, we use differential approximation to analyze the exponential generating function. It predicts a number of singularities on the positive real axis, located at the values below.

1.113439041736727043761661526918083240141390165833449466152700785053219911270 . . .
2.417184228722564007388473547672885752580057534770845001690528350200102151036 . . .
3.076673197412146436807595671137309181422151285506943038305240180949212077913 . . .
3.527590791728018755531106354662725269743465863978439496914729951030934478987 . . .
3.872438162423457670453537298789680569472671309363632792004917259462379566078 . . .
4.152519207830100565666605055176411745894938982832118599384868016797119166567 . . .
4.388766437824164163366758081274636520883940965171626205159043874261749420137 . . .
4.593300493040369902037314403433340137408669134838327397901215132095535249496 . . .
4.773787732301263733990448984231076188826829730174328444872240429327757789160 . . .
4.93539355029443080528699130532727322201728351298582403913
5.08176797057144544489527338196678922218609719159
5.215588012778242472294262722856995906
5.453200964209036692
5.55979961612
5.659669

Each of these singularities is predicted to have critical exponent -1, making them simple poles. The topmost 9 estimates have been truncated to fit on the page. In reality, they are given to many more decimal places—nearly 800 for the dominant singularity. More precise estimates could be obtained if desired. These results suggest that the exponential generating function may posses an infinite number of singularities. If true, this would imply the non-D-finiteness of both the ordinary and exponential generating functions.

Differential approximation also predicts several complex pairs of singularities, also simple poles, of which we will list just a few.

$$\begin{aligned}
&0.427938097544072724299159137354094602963785449752185713425477735405948 9934\ldots\\
&\quad\pm 3.60125951342747821372945513235678991468782821094074923509880159005527 87045\ldots i\\
&1.80793192245255330456527156504385531865084517865786934122477869708107 74117\ldots\\
&\quad\pm 4.04623498761068877028974574411286457634903048503441957438805928710461 30995\ldots i\\
&2.50839987173696627276872491933149454763814647478804617699208846228748 45896\ldots\\
&\quad\pm 4.24168001603923292919409692042505451403821499822723942133725953064298 64967\ldots i
\end{aligned}$$

The dominant pole at $\mu \approx 1.11343904$ implies that the exponential growth rate of the counting sequence is

$$\mu^{-1} \approx 0.8981183185746869695116759646856448\ldots,$$

implying that the asymptotic behavior of the number of pop-stacked permutations is

$$a_n \sim C \cdot n! \cdot (0.898118\ldots)^n.$$

Differential approximation does not provide an estimate for the constant C but this can be obtained numerically given the extremely accurate estimate for μ. We find that

$$C \approx 0.6956885490706357679957031687241101565741983507216179232324\ldots$$

giving the final asymptotic approximation

$$a_n \sim (0.695688\ldots) \cdot n! \cdot (0.898118\ldots)^n.$$

Full decimal values for the approximated singularities and constants can also be found on GitHub [5].

Acknowledgements. Computations were performed on the Garpur cluster [9], a joint project between the University of Iceland and the University of Reykjavik funded by the Icelandic Centre for Research. We thank them for the use of their resources.

References

[1] Andrei Asinowski, Cyril Banderier, Sara Billey, Benjamin Hackl, and Svante Linusson. Pop-stack sorting and its image: Permutations with over-

lapping runs. *Acta Mathematica Universitatis Comenianae*, 88(3):395–402, 2019.

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

[3] David Avis and Monroe Newborn. On pop-stacks in series. *Utilitas Math*, 19(129-140):410, 1981.

[4] Anders Claesson and Bjarki Ágúst Guðmundsson. Enumerating permutations sortable by k passes through a pop-stack. *Adv. in Appl. Math.*, 108:79–96, 2019.

[5] Anders Claesson, Bjarki Ágúst Guðmundsson, and Jay Pantone. Enumerating the pop-stacked permutations. `https://github.com/SuprDewd/pop-stacked-perms`.

[6] Anthony J Guttmann. Asymptotic analysis of power-series expansions. *Phase transitions and critical phenomena*, 13:1–234, 1989.

[7] Anthony J Guttmann and Iwan Jensen. Series analysis. In *Polygons, polyominoes and polycubes*, volume 775 of *Lecture Notes in Phys.*, pages 181–202. Springer, Dordrecht, 2009.

[8] Waldemar Hebisch and Martin Rubey. Extended rate, more GFUN. *J. Symbolic Comput.*, 46(8):889–903, 2011.

[9] Garpur cluster. IHPC - Icelandic High Performance Computer - University of Iceland and Reykjavik University, 2019.

[10] Manuel Kauers. Guess: A Mathematica package for guessing multivariate recurrence equations. *Research Institute for Symbolic Computation.*

[11] M. A. H. Khan. High-order differential approximants. *J. Comput. Appl. Math.*, 149(2):457 – 468, 2002.

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

[13] Bruno Salvy and Paul Zimmermann. GFUN: A Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Trans. Math. Softw.*, 20(2):163–177, June 1994.

[14] Neil J. A. Sloane. The Online Encyclopedia of Integer Sequences. `https://oeis.org`, 2019.

[15] Rebecca Smith and Vincent Vatter. A stack and a pop stack in series. *Australas. J. Combin.*, 58:157–171, 2014.