Wilf classification of subsets of four-letter patterns

Toufik Mansour
University of Haifa, Israel

July 17 – July 21, 2017

23rd Conference on Applications of Computer Algebra
Issue: Algorithmic Combinatoric

Joint works with
David Callan, Matthias Schork and Mark Shattuck

Table of contents

In the last decades, the problem of avoiding patterns in different combinatorial structures like permutations, coloured permutations, compositions, partitions, set partitions, etc. has been studied by many authors from many different point of views.

In this talk, we restrict to permutations and the problem of pattern avoidance for them.

Let $\mathcal{S}_n$ be the symmetric group of all permutations of $[n] \equiv \{1, \ldots, n\}$.

Let $\pi = \pi_1\pi_2\cdots\pi_n \in \mathcal{S}_n$ and $\tau = \tau_1\tau_2\cdots\tau_k \in \mathcal{S}_k$ be two permutations. We say that π contains τ if there exists a subsequence $1 \leq i_1 < i_2 < \cdots < i_k \leq n$ such that $\pi_{i_1}\pi_{i_2}\cdots\pi_{i_k}$ is order-isomorphic to τ, that is, $\pi_{i_a} < \pi_{i_b}$ if and only if $\tau_a < \tau_b$; in such a context τ is usually called a pattern.

Example: $\pi = 142653$ contains the pattern $\tau = 231$.

We say that π avoids τ, or is τ*-avoiding*, if such a subsequence does not exist. For example, 35412 avoids 123.

The set of all τ-avoiding permutations in $\mathcal{S}_n$ is denoted by $\mathcal{S}_n(\tau)$.

For an arbitrary finite collection of patterns T, we say that π avoids T if π avoids every pattern τ in T; the corresponding subset of $\mathcal{S}_n$ is denoted by $\mathcal{S}_n(T)$, i.e., $\mathcal{S}_n(T) = \bigcap_{\tau\in T}\mathcal{S}_n(\tau)$.

The sets of patterns T and T' belong to the same Wilf class (or are Wilf-equivalent) if and only if $|\mathcal{S}_n(T)| = |\mathcal{S}_n(T')|$ for all $n \geq 0$.

In 1985, Simion and Schmidt found the cardinality of $\mathcal{S}_n(T)$, where $T \subseteq \mathcal{S}_3$. Thus, the case of patterns of length three is well-known.

Let us turn to patterns of length four. For this case, much less is known and is seems hopeless to get an explicit formula for $|\mathcal{S}_n(T)|$ where $T \subseteq \mathcal{S}_4$ is arbitrary.

Already the case of avoiding exactly one pattern $\tau \in \mathcal{S}_4$ is not trivial. It was shown that there are three essentially different cases, namely $\mathcal{S}_n(\tau)$ where $\tau \in \{1342, 1234, 1324\}$. Since $|\mathcal{S}_7(1342)| = 2740$, $|\mathcal{S}_7(1234)| = 2761$ and $|\mathcal{S}_7(1324)| = 2762$, these three patterns comprise three different Wilf classes.

If we denote the number of symmetry classes and Wilf classes of subsets of k patterns in $\mathcal{S}_4$ by s_k and w_k, respectively, then this means that $w_1 = 3$ $(s_3 = 7)$.

Let us turn to subsets $T = \{\tau_1, \tau_2\}$ with exactly two patterns in $\mathcal{S}_4$. There do exist $\binom{24}{2} = 276$ such subsets T. Le established that these 276 subsets form $w_2 = 38$ distinct Wilf classes.

Thus, the aim of this talk is discuss how to determine w_k for $3 \leq k \leq 24$. Since the number of subsets of $\mathcal{S}_4$ containing at least 3 patterns is given by $\sum_{k=3}^{24} \binom{24}{k} = 16776915$, it seems to be impossible to reach by constructing explicit bijections between sets of permutations. The way out is to combine several software programs to do the work for us!!

This talk based on recent works of the speaker and his coauthors; David Callan, Mark Shattuck and Matthias Schork.

Main Results!!

Theorem, 2016(ManSch): Let w_k be the number of distinct Wilf classes of k patterns in $\mathcal{S}_4$. Then $w_{10} = 10624$, $w_{11} = 5857$, $w_{12} = 3044$, $w_{13} = 1546$, $w_{14} = 786$, $w_{15} = 393$, $w_{16} = 198$, $w_{17} = 105$, $w_{18} = 55$, $w_{19} = 28$, $w_{20} = 14$, $w_{21} = 8$, $w_{22} = 4$, $w_{23} = 2$, and $w_{24} = 1$.

Theorem, 2016(ManSch): We have $w_8 = 19002$ and $w_9 = 16293$.

Theorem, 2017(ManSch): We have $w_6 = 8438$ and $w_7 = 15392$.

Remark: all the classes are enumerated.

Theorem, 2017(CalManSha): The $\binom{24}{3} = 2024$ triples of 4-letter patterns split into 317 symmetry classes which split into 242 Wilf classes, 32 of which are large (a Wilf class is called large if it contains at least two symmetry classes, and small if it consists of a singleton symmetry class) and the large Wilf classes are all explicitly enumerated, where it is shown that each has an algebraic generating function.

Theorem, 2017(CalMan): Running the INSENC algorithm (regular insertion encoding) over all the 210 small Wilf classes determines the generating function for 126 of them.
(1) The remaining small classes that contain 1324 (34 classes) are all explicitly enumerated where it is shown that each has an algebraic generating function, but the generating function for $|S_n(4123, 4231, 4312)|$ is conjectured not to be differentiably finite (D-finite) and hence not algebraic (see thesis of Garrabrant).
(2) The remaining small classes that contain 1342 (40 classes) are all explicitly enumerated where it is shown that each has an algebraic generating function.
(3) All the other cases (10 classes) are under writing.

Keys/Hints for the proofs!!

The aim is to determine w_k for $k \in \{3, 6, 7, \ldots, 24\}$. Since the number of subsets of $\mathcal{S}_4$ is 2^{24}, it seems to be impossible to reach by using the action of the dihedral group on permutations and by constructing explicit bijections between sets of permutations.

The way out is to combine several software programs to do the work for us. Briefly, the line of argument will be as follows:

Let $T \subseteq \mathcal{S}_4$ be a set of patterns. We define $F_T(x)$ to be the generating function of the sequence $(|\mathcal{S}_n(T)|)_{n\geq 0}$, i.e.,

$$F_T(x) = \sum_{n\geq 0} |\mathcal{S}_n(T)|x^n.$$

If the generating function F_T can be found by the regular inserting encoding procedure INSENC, then we say that F_T is regular.

The key step to find w_k is to find the associated generating function F_T for all $T \subseteq \mathcal{S}_4$ with exactly k patterns (for regular generating functions using INSENC, for the non-regular ones by any method).

In fact, we only have to study the symmetry classes of subsets with exactly k patterns in $\mathcal{S}_4$; let us denote this set by $\mathcal{SC}_k$ and set $s_k = |\mathcal{SC}_k|$. However, as shown in the next table, there are still many symmetry classes to study.

k	s_k	k	s_k	k	s_k
12	343424	13, 11	316950	14, 10	249624
15, 9	166766	16, 8	94427	17, 7	44767
18, 6	17728	19, 5	5733	20, 4	1524
21, 3	317	22, 2	56	23, 1	7
24, 0	1				

Thus, we still need the help of a computer!

In the following five steps, we describe the software programs used in the successive steps to find the Wilf classes.

Step 1: Kuszmaul's Program to Generate All Symmetry Classes

We use the software of Kuszmaul to create all the symmetry classes $T \in \mathcal{SC}_k$ and the sequence $(|\mathcal{S}_n(T)|)_{n=1,\dots,16}$ for every such T.

For each $k \geq 0$, we denote the output file of Kuszmaul's program by OutFilKus(k); all these files can be found in my homepage.

The file OutFilKus(k) contains $2s_k$ lines. Each two consecutive lines are called a pair, denoted by (T, L), where $T \in \mathcal{SC}_k$, and $L \equiv L(T)$ is the sequence $(|\mathcal{S}_n(T)|)_{n=1,\dots,16}$.

In order to present our procedure, we introduce the following notation. Let $\mathcal{L}_k$ be the list of all different sequences L in the file OutFilKus(k), say $L_{k,1}, L_{k,2}, \ldots, L_{k,m_k}$ (with the same order of appearance in OutFilKus(k), from top to bottom); clearly, $m_k \leq s_k$.

For each sequence $L_{k,i} \in \mathcal{L}_k$, let $T_{k,i}^{(1)}, \ldots, T_{k,i}^{(m_{k,i})}$ be the list of all symmetry classes T (with the same order of appearance in the file OutFilKus(k), from top to bottom) such that $(T, L_{k,i})$ is a pair in OutFilKus(k).

This is essentially a rearrangement of the contents of OutFilKus(k) (but no sequence L appears more than once).

Note that for two symmetry classes $T_{k,i}^{(j)}$ and $T_{k,i'}^{(j')}$ with $i \neq i'$ one has $L_{k,i} \neq L_{k,i'}$, so they do not belong to the same Wilf class, implying $w_k \geq m_k$.

Note also that $T_{k,i}^{(j)}$ and $T_{k,i}^{(j')}$ have the same sequence $L_{k,i}$, meaning that $(|\mathcal{S}_n(T_{k,i}^{(j)})|)_{n=1,\dots,16} = (|\mathcal{S}_n(T_{k,i}^{(j')})|)_{n=1,\dots,16}$; thus, they *might* be in the same Wilf class.

Observe that, for $n = 1, 2, 3$, all permutations in $\mathcal{S}_n$ avoid all patterns in $\mathcal{S}_4$, implying $|\mathcal{S}_n(T)| = |\mathcal{S}_n| = n!$ for all $T \subseteq \mathcal{S}_4$. Since, by convention, $\mathcal{S}_0(T) = 1$, all sequences $(|\mathcal{S}_n(T)|)_{n\in\mathbb{N}}$ start with $(1, 1, 2, 6, \ldots)$.

Thus, the generating functions for all $T \subseteq \mathcal{S}_4$ can be written as

$$F_T(x) = 1 + x + 2x^2 + 6x^3 + \sum_{n\geq 4} |\mathcal{S}_n(T)|x^n.$$

Step 2: A Maple Program to Prepare the Input for INSENC

In order to sort the file OutFilKus(k), we use the C++ program OutMapCodMan(k) which operates as follows:

1. Add the initialization line

   'restart: read "FINLABEL.txt"; read "INSENC.txt";'

2. Then, for each $i = 1, \ldots, m_k$, add the ith sequence $L_{k,i} \in \mathcal{L}_k$ as 'L:=$L_{k,i}$;' and for all $j = 1, \ldots, m_{k,i}$, add 'T:=$T_{k,i}^{(j)}$: print(T); RWiAj:=insenc(T): AiAj:=rules2gf(RWiAj,x);'

In other words, the output file OutMapCodMan(k) contains the Maple code to operate the INSENC procedure on all symmetry classes $T \in \mathcal{SC}_k$.

In next table, we present the size of the file OutMapCodMan(k).

Table: Size of the file OutMapCodMan(k) in kilobyte.

k	**size**	k	**size**	k	**size**
6	2520	7	5890	8	11300
9	11300	10	28900	11	37700
12	41800	13	40500	14	33100
15	22800	16	13000	17	6370
18	2590	19	882	20	239
21	51	22	9	23	1
24	–				

Step 3: INSENC to Determine a List of Generating Functions

In order to cope with the huge amount of data contained in the files OutMapCodMan(k), we divide each file into smaller files OutMapCodMansmall(k,d). Then, we run each smaller file in Maple

(using the command 'read "OutMapCodMansmall(k,d)";').

For all symmetry classes T in OutMapCodMansmall(k,d), the package INSENC tries to determine the generating function $F_T(x) = \sum_{n\geq 0} |\mathcal{S}_n(T)| x^n$. The output is saved as Maple code OutMapCodMansmallmws(k,d) and as a TeX file OutMapCodMansmalltex(k,d).

The pdf compilation of all these TeX files can be found in HP-Man. In Table 2, the sizes of the pdf files OutMapCod-Manpdf(k) are shown as well as the running times; observe how the running time increases when *k* decreases.

Table: Running time of OutMapCodMan(k) in seconds and the size of the pdf file OutMapCodManpdf(k) in kilobyte.

k	**time**	**size**	k	**time**	**size**	k	**time**	**size**
10	247357	45600	11	214697	53900	12	193888	56300
13	142094	50500	14	69221	38900	15	46000	28920
16	21775	9060	17	7817	7620	18	2659	2980
19	759	980	20	172	287	21	32	84.5
22	4.8	39.5	23	0.6	29.3	24	–	–

Step 4: Notation and Generation of a List of Wilf Classes

Recall from the construction of the files OutFilKus(k) in Step 1 that if (T, L) and (T', L') are two pairs with $L \neq L'$, then $F_T \neq F_{T'}$, that is, T and T' do not belong to the same Wilf class. Hence, we need only to check whether for any two symmetry classes T and T' such that (T, L) and (T', L) are two pairs, the identity $F_T = F_{T'}$ holds true.

Here we define, for $1 \leq i \leq m_k$ and $1 \leq j \leq m_{k,i}$,

$$F_{k,i}^{(j)}(x) = \sum_{n \geq 0} |\mathcal{S}_n(T_{k,i}^{(j)})| x^n.$$

Step 5: Maple Program to Generate a Table of Wilf Classes

From each TeX file OutMapCodMantex(k) produced in Step 3 we generate a table of Wilf classes of subsets of exactly k patterns in $\mathcal{S}_4$. To do this in an efficient way, we write each rational generating function

$$F(x) = \frac{x^4 \sum_{i=0}^{m} p_i x^i}{\sum_{i=0}^{m'} q_i x^i}$$

as $seq(F) = p_0, p_1, \ldots, p_m / q_0, q_1, \ldots, q_{m'}$, which we call the *sequence form* of F. In the case $m' = 0$ and $q_0 = 1$, i.e., F is a polynomial, we represent it briefly as $seq(F) = p_0, p_1, \ldots, p_m$.

We scan the TeX file OutMapCodMantex(k) and produce a Maple code called OutMapCodTabmws(k), that does the following:

1. For each *different* generating function G, it writes a table entry containing
   1.1 the index i (counting the different generating functions),
   1.2 the sequence form of $G - 1 - x - 2x^2 - 6x^3$, namely $seq(G - 1 - x - 2x^2 - 6x^3)$.
   1.3 the number $n_{k,i}$ of symmetry classes it contains.
2. Prints the table.

Note that to avoid printing redundant information in the tables, we subtract the first 4 terms of the generating functions (which are the same for all symmetry classes).

This last step completes work done by the software part. In the case that for the k considered all generating functions were found by INSENC (i.e., $s_{k,i} = 0$ for all i), the above software-generated table lists the generating functions of all Wilf classes, hence determines w_k.

However, if INSENC failed to determine the generating function for some "bad" symmetry classes, one has to determine these generating functions by hand (here we need a magic).

(1) We computed the number of Wilf classes of subsets of k patterns in $\mathcal{S}_4$ ($k = 4, 5$ under writing). Since the number of subsets of k patterns is rather large, we used several software packages to do the main work.

(2) The two essential ingredients were Kuszmaul's algorithm to generate all symmetry classes and INSENC to determine the generating functions $\sum_{n\geq 0} |\mathcal{S}_n(T)|x^n$ for all symmetry classes T.

(3) Starting from $k = 23$ (the case $k = 24$ is trivial) and decreasing k, the running time increases. However, this only poses a practical problem. Far more serious is the problem that more and more symmetry classes exist for which INSENC fails to determine the generating function.

These cases have to be done by hand, which becomes increasingly cumbersome. For example, when $k \in \{13, \ldots, 23\}$ INSENC never fails,

k	*fails*	out of s_k	k	*fails*	out of s_k
12	1	343424	11	10	316950
10	48	249624	9	151	166766
8	335	94427	7	549	44767
6	659	17728	3	116	317

(4) For these cases, we used different methods such as scanning-element algorithm, left-right maxima, generating trees, Cell diagram, Block decompositions,.... with generating functions and kernel method.

Thanks

Thanks for attention