ÖAW AUSTRIAN ACADEMY OF SCIENCES

RICAM – JOHANN RADON INSTITUTE FOR COMPUTATIONAL AND APPLIED MATHEMATICS

A348456(4) = 7157114189

M. Kauers, C. Koutschan, G. Spahn

RICAM-Report 2022-23

WWW.RICAM.OEAW.AC.AT

ALTENBERGER STRASSE 69 | 4040 LINZ, AUSTRIA
T: +43 732 2468-5222 | F: +43 732 2468-5212 | OFFICE@RICAM.OEAW.AC.AT
IBAN: AT541100000262650519 | BIC: BKAUATWW | VAT: ATU37116303

RICAM

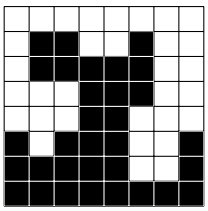
$A348456(4) = 7157114189$

MANUEL KAUERS *, CHRISTOPH KOUTSCHAN, AND GEORGE SPAHN

Abstract. We compute the next few terms of the OEIS sequence A348456 and provide guessed equations for the generating functions of some sequences in its context.

1. Introduction

In a guest lecture on April 28, 2022, in Doron Zeilberger's famous experimental mathematics seminar at Rutgers University [11], Neil Sloane talked about some of his favorite entries of the OEIS. One of the entries he highlighted in his lecture was A348456. The nth term of this sequence is defined as the number of ways to dissect a $2n \times 2n$ chessboard into two polyominoes each of area $2n^2$. Here is one of the solutions for $n = 4$:



Finding such dissections can be viewed as a combinatorial version of gerrymandering, and it has thus been suggested to call the sequence the gerrymander sequence. When Sloane gave his lecture, only the first three terms of the gerrymander sequence were known (they are 2, 70, 80518), and he declared, perhaps exaggerating a bit, that he considers the next term of this sequence as the "most wanted number" in the whole OEIS. This statement motivated Zeilberger to offer a donation of \$100 to the OEIS in honor of the person who first manages to compute this most wanted number. In this paper, we explain how we computed not only the next term of A348456, but in fact the next four terms. They are 7157114189, 49852157614583644, 28289358593043414725944353, and 13350565794230803711864568885437321 62, respectively.

We employ the transfer matrix method, a classical technique in enumerative combinatorics whose general idea is nicely explained in Sect. 4.7 of Stanley's textbook [9]. In Sect. 2, we explain how the problem at hand is translated into the setting of this method. Besides the computation of specific terms, the transfer matrix method also allows to derive structural information about the generating function for boards of rectangular shapes $m \times n$ when m is fixed and n varies. The case $m = 3$ was proposed by Knuth as a Monthly problem a few years ago [8]. It turns out that if a_n is the number of ways to break a $3 \times 2n$ board into two connected components of the same size (A167242), then

$$\sum_{n=0}^{\infty} a_n x^n = \frac{1+\sqrt{1-4x}}{(\sqrt{1-4x}+x)^2}\frac{1}{\sqrt{1-4x}} - \frac{1-x^2+2x^3}{(1-x)^3}.$$

It is not a coincidence but a consequence of a theorem of Furstenberg [2] that the generating function is algebraic. In principle, it can be computed by combining the transfer matrix method with the method of creative telescoping [10, 1, 5]. However, this quickly becomes expensive when m increases. In Sect. 4, we report on some computations we did in this direction.

2. The Transfer Matrix

The transfer-matrix method was invented in the context of statistical mechanics [6, 7], in order to express the partition function of a statistical model in a simpler and more succinct form than its plain definition as a multi-dimensional sum. The method is applicable whenever the mechanical system can be decomposed into a sequence of N subsystems, each of them interacting only with the previous and the next one. Let ℓ denote the number of states that each of these subsystems can have, and $m_{i,j}(k)$ a "statistical weight"

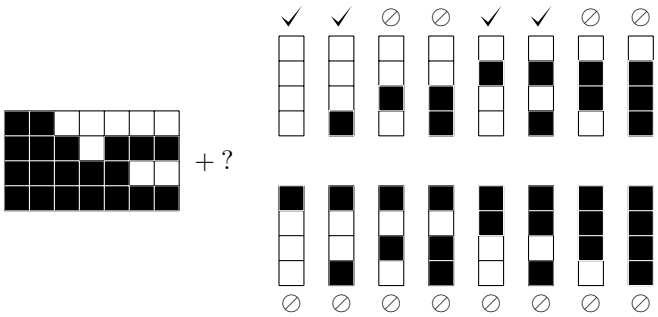
* Supported by the Austrian FWF grant P31571-N32.

that is associated with state i of subsystem $k-1$ being next to state j of subsystem k. The relation between these two adjacent subsystems is then described by the *transfer matrix* $M(k) = \big(m_{i,j}(k)\big)_{1\le i,j\le \ell}$, and the partition function of the whole system can be written in the form

$$v_{\text{init}}^{\top} \cdot \left(\prod_{k=1}^{N} M(k)\right) \cdot v_{\text{final}},$$

where v_{init} and v_{final} are vectors of dimension ℓ.

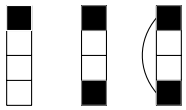
Recall that we are interested in counting the number of ways how an $m \times n$ grid can be divided into two (or mode generally, q) connected regions, each of which is represented by a different color. We can apply the transfer-matrix method to this gerrymandering problem by decomposing the grid into a sequence of columns that are added one after the other. In each step, not all of the 2^m (resp. q^m) potential columns can be added, because some would violate the rules, e.g., by creating two disconnected regions of the same color:



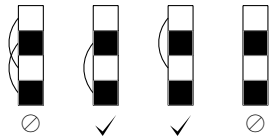
Clearly, the information how the squares in the right-most column are colored is not sufficient to decide which columns can come next. For example, we need to know that the two black squares in the last column belong to the same connected region, otherwise we could not add a column with only white squares.

Thus, in order to decide which columns can be added and to tell whether the completed grid has the desired number of connected regions, we introduce *states* for remembering connectivity information. More precisely, a state is described by a tuple (c, P), where $c \in \{0,1\}^{2n}$ encodes the content of the last column (the colors white and black are now represented by the numbers 0 and 1, respectively), and where $P = \{P_1, \dots, P_k\}$ is a partition of $\{1, \dots, 2n\}$ that indicates the connectness of same-color squares in that column. All squares at positions given by P_j are assumed to belong to the same connected region; in particular, they must have the same color, that is $|\{c_i \mid i \in P_j\}| = 1$ for all $1 \le j \le k$.

For example, $\big((0,0,1,0), \{\{1,2\},\{3,4\}\}\big)$ or $\big((1,1,0,0), \{\{1,2\},\{3\},\{4\}\}\big)$ are not valid state descriptions because the first violates the same-color condition, while the latter claims that the squares at positions 3 and 4 belong to different regions although they are obviously connected. In contrast, $\big((1,0,0,0), \{\{1\},\{2,3,4\}\}\big)$, $\big((1,0,0,1), \{\{1\},\{2,3\},\{4\}\}\big)$, or $\big((1,0,0,1), \{\{1,4\},\{2,3\}\}\big)$ are valid state descriptions, which we depict graphically as follows (connections that happen in previous columns are symbolized by an arc):



But even if we stick to the above rules, there are still many tuples (c, P) that describe *impossible* states, for example $\big((0,1,0,1), \{\{1,3\},\{2,4\}\}\big)$. Connecting 1 with 3 and 2 with 4 produces arcs that cross each other, meaning that it is not possible to achieve this connectivity by extending this column to the left. Similarly, but somewhat dual, we can discard *uninteresting* states, i.e., states that could possibly be reached, but which represent "hopeless" situations that will never allow us to complete the grid in a satisfactory manner. For example, the state $\big((0,1,0,1), \{\{1\},\{2\},\{3\},\{4\}\}\big)$ is uninteresting, because all four squares are declared to belong to different regions, and it is impossible to connect 1 with 3 and 2 with 4 by adding more columns to the right. Hence, for the column $c = (0,1,0,1)$ we consider only two out of four possible states:



There is one subtle issue one still has to take care of: if the current column equals $(0, \ldots, 0)$ or $(1, \ldots, 1)$, then we need to store the information whether the other color has not yet appeared (in which case this column can be followed by any other column), or whether the other color has appeared previously (in which case this column can only be followed by more copies of the same column). In our graphical notation, we decorate the latter of these two states by a prime.

Let us denote the set of states that is constructed according to the above rules, by L. The number $\ell = |L|$ of states grows (at least) exponentially with n, since each of the 2^{2n} possible columns appears in at least one state. For example, for $n = 2$ we have 16 different columns but 26 states in total; they are explicitly enumerated in Figure 1. The number of states for $1 \leq n \leq 7$ is given as follows:

n	1	2	3	4	5	6	7
ℓ	6	26	154	1026	7222	52650	393878

For constructing the transfer matrix M, we need to determine for each state into which other states it can transition by adding another column. Note that for this purpose it does not matter in which particular column of the grid we are: in each step we can use the same matrix M (this is in contrast to the general situation sketched at the beginning of this section). The rows and columns of the transfer matrix are indexed by the states; hence we obtain an $\ell \times \ell$ matrix. Let $s = (c, P)$ be any of the ℓ states and let $c' \in \{0, 1\}^{2n}$ be an arbitrary column. If attaching c' to the state s would violate the connectivity requirements, then the matrix entries at positions (s, s') are set to 0, where s' is any state having c'.

But what should be put as matrix entry when a transition is actually possible? Here another condition has to be considered that so far has not been taken care of: the two regions must have the same area. Since this requirement can only be checked at the very end when the whole grid is filled, we need to propagate information about the number of squares of either color through the whole computation. For this purpose, we introduce a "catalytic" variable x that counts the number of white squares that have been used so far. In each transition this counter has to be increased accordingly. Hence, if state $s = (c, P)$ admits attaching column c' to it (the new connectivity information P' is then uniquely determined), yielding a new state $s' = (c', P')$, then the matrix entry at position (s, s') is set to $x^{\#_0(c')}$, where $\#_0(c')$ denotes the number of 0's in c'.

Now that we have constructed the transfer matrix M, it remains to consider the start and the end of this process. We start the grid with a single column. There cannot be any additional connectivity information other than what can be seen in this column. We define a start vector v_{init}, which is indexed by the states and hence ℓ-dimensional, such that its entry at position $s = (c, P)$ equals $x^{\#_0(c)}$ if the parts of P correspond exactly to the consecutive runs of entries of the same color in c (in other words: if the graphical representation of s has no arcs (and no prime!)), and 0 otherwise.

When we have filled the grid up to the last column, we have to decide which states are "accept" states. Clearly, this is the case for states $s = (c, P)$ such that $|P| \leq 2$. Since we just want to add up the results of all acceptable states, we define a vector v_{final} that is 1 at accept states and 0 otherwise.

Having all this at hand—the transfer matrix M, the start vector v_{init}, and the end vector v_{final}—one can compute

$$p(x) = v_{\text{init}}^{\top} \cdot M^{2n-1} \cdot v_{\text{final}}, \tag{1}$$

which is a polynomial in x. The coefficient of x^k in p gives the number of ways how the $2n \times 2n$ grid can be divided into a white polyominoe of k squares and a black polyominoe consisting of $4n^2 - k$ squares. This means that for sequence A348456 we need to extract the coefficient of x^{2n^2}.

3. Optimizations

In this section, we discuss some optimizations that one can employ when implementing the method described in Section 2. For our purposes we have used the computer algebra systems Maple and Mathematica.

First, it is clear that in (1) one should not compute M^{2n-1} explicitly, using expensive matrix multiplications. Instead, it is more efficient to exploit the associativity of matrix multiplication and perform only

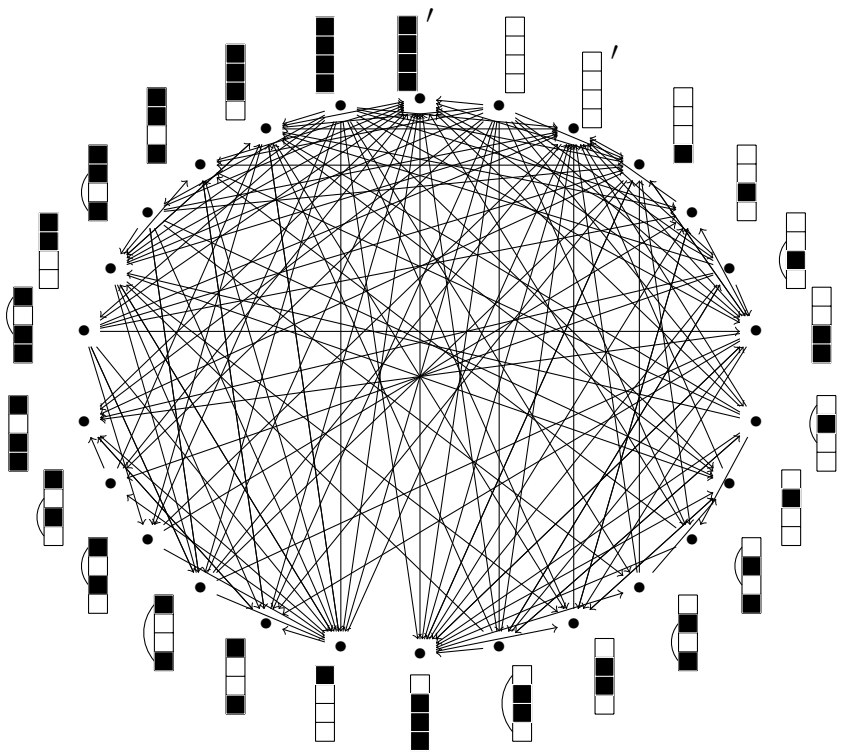


FIGURE 1. All 26 states for a grid with 4 rows and their possible transitions; note that each state can also be followed by itself—these loops are not depicted.

(cheap) matrix-vector multiplications:

$$p(x) = (\cdots((v_{\text{init}}^{\top} \cdot M) \cdot M) \cdots) \cdot v_{\text{final}}.$$

Then we observe that the transfer matrix is very sparse: clearly, in each row we can have at most 2^{2n} nonzero entries (because this is the number of different grid columns we can add), but actually there are much fewer, since many of these columns are not admissible (e.g., because they disconnect existing regions). For example, for $n = 5$ in the case of an 10×10 grid, 98.4% of the matrix entries are zeros. For $n = 7$ the matrix has 393878^2, i.e. about 115 billion entries, which poses memory challenges when one does not have a supercomputer at hand. In Mathematica, one can use the command `SparseArray` to store such matrices in a memory-efficient way, which has the additional advantage that it also speeds up the matrix-vector multiplications.

The next observation concerns the structure of the transfer matrix: since each specific column of the matrix is responsible for producing the counting polynomial for a specific label (c, P), by matrix multiplication, all entries of this column of M must either be 0 or $x^{\#_0(c)}$. No other powers of x can occur in the same column. It is more efficient to work with the $\{0,1\}$-matrix $M' = M\big|_{x\to 1}$ and store the x-powers in a separate diagonal matrix

$$X = \operatorname{diag}\left(\left(x^{\#_0(c)}\right)_{(c,P)\in L}\right)$$

such that $M = M' \cdot X$. Computing $v_{\text{init}}^{\top} \cdot M$ has the disadvantage that large intermediate expressions are produced that can only be combined after expansion. This is avoided by computing $\left(v_{\text{init}}^{\top} \cdot M'\right) \cdot X$.

We know that the result is a polynomial $p(x)$ of degree $4n^2$, whose coefficient of x^{2n^2} we wish to extract. Hence, the whole computation can be done modulo x^{2n^2+1}, which does not change the coefficient of x^{2n^2}, but which reduces the size of intermediate expressions. Alternatively, one can apply the evaluation-interpolation technique combined with the Chinese remainder theorem. Not only do we know a priori the degree of $p(x)$, we also know that it is palindromic: $p(x) = x^{4n^2} p(1/x)$. Therefore $p(x)$ can be interpolated by using only $2n^2 + 1$ evaluation points.

The following table illustrates the effect of different strategies on the runtime of the computation (in the case of $n = 4$):

transfer matrix		mult. in $\mathbb{Z}[x]$	in $\mathbb{Z}[x]/(x^{2n^2+1})$	eval-int. + CRT
dense	M	23.1 s	22.3 s	69.6 s
dense	$M'X$	9.1 s	8.3 s	89.7 s
sparse	M	16.3 s	15.3 s	1.8 s
sparse	$M'X$	1.9 s	2.0 s	0.6 s

The next table shows computation times for $4 \leq n \leq 7$. Note that we used parallelization for some of the tasks, but the timings are given in CPU time. As a curiosity, we realized that when computing $v_{\text{init}}^\top \cdot M$ (row vector times matrix), Mathematica takes about twice as much time as when computing the equivalent product $M^\top \cdot v_{\text{init}}$ (transposed matrix times column vector).

	build M	matrix-vector multiplications		
		mult. in $\mathbb{Z}[x]$	in $\mathbb{Z}[x]/(x^{2n^2+1})$	eval-int. + CRT
$n = 4$	10 s	1.9 s	2.0 s	0.6 s
$n = 5$	5 min	56 s	54 s	21 s
$n = 6$	5 h	49 min	44 min	10 min
$n = 7$	5 d	29 h	25 h	6 h

4. Further Results

While the gerrymandering problem for a rectangular board of size $m \times n$ is symmetric in m and n, the cost of the transfer matrix method is highly asymmetric. As explained above, the cost depends exponentially on the side length that determines the transfer matrix but only polynomially on the side length that appears in the exponent. Because of this discrepancy, slim rectangular boards are somewhat easier to handle than boards that are quadratic or close to quadratic. For small values of m, it is not too hard to let n grow into the hundreds or even thousands.

For fixed m and varying n, we are interested in the number of solutions to the gerrymandering problem for a board of size $m \times n$. Obviously there is no solution when both m and n are odd. Therefore, for fixed and even m, we define a_n as the number of solutions for a board of size $m \times n$, and for fixed odd m, we define a_n as the number of solutions for a board of size $m \times 2n$. For certain vectors $v_{\text{init}}, v_{\text{final}}$ and a certain matrix M whose entries are polynomials in x, we then have $a_n = [x^{nm/2}](v_{\text{init}}^\top M^n v_{\text{final}})$ if m is even and $a_n = [x^{nm}](v_{\text{init}}^\top M^{2n} v_{\text{final}})$ if m is odd. We know from linear algebra that the entries of a matrix power M^n are C-finite sequences with respect to n, i.e., they satisfy linear recurrences with constant coefficients, or in other words, their generating functions are rational. An explicit formula is given in Thm. 4.7.2 of [9]: for a fixed $\ell \times \ell$-matrix M and any $i, j \in \{1, \ldots, \ell\}$, the generating function of the sequence appearing in the (i, j)th entry of M^n is

$$(-1)^{i+j} \frac{\det(I_\ell - t\,M)^{[j,i]}}{\det(I_\ell - t\,M)},$$

where the exponent $[j, i]$ indicates the removal of the jth row and the ith column of the matrix $I_\ell - tM$. The generating function for a sequence defined as $v_{\text{init}}^\top M^n v_{\text{final}}$ is just a certain linear combination of such rational functions.

In particular, the rational generating function for the sequence $(v_{\text{init}}^\top M^n v_{\text{final}})_{n=0}^\infty$ (or $(v_{\text{init}}^\top M^{2n} v_{\text{final}})_{n=0}^\infty$, if m is odd) can be explicitly computed from the vectors $v_{\text{init}}, v_{\text{final}}$, and the matrix M. At least in principle. In practice, for large matrices M involving a symbolic parameter x, computing the determinant of $I_\ell - t\,M$, which involves an additional symbolic parameter t, can be a hassle. We have managed to compute the rational expressions using evaluation/interpolation techniques for $m = 3, \ldots, 7$. The computation is non-rigorous in so far as the number of evaluation points was only determined experimentally. In the table below, we summarize their sizes:

m	1	2	3	4	5	6	7
$\deg_t$ of numerator	2	4	7	17	36	75	203
$\deg_x$ of numerator	2	4	23	34	190	236	1425
monomials in numerator	3	9	76	194	1955	4312	55218
$\deg_t$ of denominator	2	4	7	17	36	75	203
$\deg_x$ of denominator	2	4	22	34	188	235	1422
monomials in denominator	4	9	76	194	1935	4310	55188

.

The cases $m = 1$ and $m = 2$ are quite simple, for example, for $m = 2$ the generating function is

$$\frac{-t^4x^4 - t^3x^4 - 2t^3x^3 - t^3x^2 + 4t^2x^2 - tx^2 + 2tx - t + 1}{(t-1)^2\,(tx^2-1)^2}$$
$$= 1 + (x^2 + 2x + 1)t + (x^4 + 4x^3 + 4x^2 + 4x + 1)t^2 + (x^6 + 6x^5 + 6x^4 + 6x^3 + 6x^2 + 6x + 1)t^3 + \cdots$$

For $m = 3$, the expression is already too big to fit in a line, and as can be seen in the table, the sizes increase significantly with respect to m.

The number of colors affects the growth of the expressions even more significantly. We have considered a variant of the problem where besides black cells and white cells we also have gray cells. The question is then how many ways there are to dissect the $m \times n$ grid into three connected regions, with prescribed areas for each color. The corresponding rational generating function contains three variables: one marking the length n of the board, one marking the area of black cells, and one marking the area of white cells. (There is no need for a variable marking the area of the gray cells because we know that the areas of the three colors must sum to mn.) We were only able to construct the rational generating function for the case $m = 3$. Its numerator has degree 29 in t and degree 32 in x_1 and x_2, it altogether consists of 7939 monomials. The denominator has degree 28 in t and degree 30 in x_1 and x_2, and it consists of 7412 monomials.

Returning to the case of two colors, it remains to discuss the coefficient extraction operator. If $a(x, t)$ is a rational generating function in x and t, viewed as power series in t whose coefficients are polynomials in x, extracting the coefficient of $x^{\alpha n}$ (with $\alpha = m/2$ or $\alpha = m$ depending on whether m is even or odd) from the nth term of the series is the same as extracting the coefficient of x^{-1} from $x^{-1}a(x, t/x^\alpha)$. This can be done by creative telescoping [10, 1, 5], as follows: using computer algebra, we can compute polynomials $p_0(t), \ldots, p_r(t)$ which only depend on t as well as a rational function $b(x, t)$ such that

$$p_0(t)a(x, t/x^\alpha) + \cdots + p_r(t)\frac{d^r}{dt^r}a(x, t/x^\alpha) = \frac{d}{dx}b(x, t).$$

Applying $[x^{-1}]$ on both sides, we get zero on the right, because the derivative of a rational function cannot have a residue. On the left, observe that taking the residue with respect to x commutes with the polynomials $p_i(t)$ and the derivations in t. We can thus find a homogeneous linear differential equation for the series $a(t) := [x^{-1}]\big(x^{-1}a(x, t/x^{m/2})\big)$. At least in principle.

In practice, using Koutschan's implementation [4], this works nicely for $m = 3$ (A167242), where we obtain a computer proof of Knuth's result mentioned in the introduction, and for $m = 4$ (A167247), where we find a differential equation of order 5 and polynomial coefficients of degree 208. Using guessing [3], we can also find a differential equation of order 2 with polynomial coefficients of degree 96 as well as a polynomial equation of degree 2 with polynomial coefficients of degree 51. The correctness of these guessed equations can be proved using the large differential equation constructed by creative telescoping. For $m \geq 5$, we have not been able to find equations either by creative telescoping or by guessing, although Furstenberger's theorem guarantees their existence. Following Zeilberger's example, the first named author (M.K.) will therefore offer a donation of 100€ to the OEIS in honor of the person who first manages to find a differential equation for some $m \geq 5$.

References

[1] Frédéric Chyzak. The ABC of creative telescoping. Habilitation a diriger des recherches (HDR), 2014. `http://specfun.inria.fr/chyzak/Chyzak-2014-ABC.pdf`.

[2] Harry Furstenberg. Algebraic functions over finite fields. *Journal of Algebra*, 7:271–277, 1967.

[3] Manuel Kauers. Guessing handbook. Technical Report 09-07, RISC Report Series, Johannes Kepler University, Linz, Austria, 2009. http://www.risc.jku.at/research/combinat/software/Guess/.

[4] Christoph Koutschan. HolonomicFunctions (user's guide). Technical Report 10-01, RISC Report Series, Johannes Kepler University, Linz, Austria, 2010. https://risc.jku.at/sw/holonomicfunctions/.

[5] Christoph Koutschan. Creative telescoping for holonomic functions. In Carsten Schneider and Johannes Blümlein, editors, *Computer Algebra in Quantum Field Theory: Integration, Summation and Special Functions*, Texts & Monographs in Symbolic Computation, pages 171–194. Springer, Wien, 2013.

[6] H. A. Kramers and G. H. Wannier. Statistics of the two-dimensional ferromagnet. Part I. *Phys. Rev.*, 60:252–262, 1941.

[7] H. A. Kramers and G. H. Wannier. Statistics of the two-dimensional ferromagnet. Part II. *Phys. Rev.*, 60:263–276, 1941.

[8] Donald E. Knuth (proposer) and Editors (solver). Balanced tilince of a rectangle with three rows. problem 11929. *American Mathematical Monthly*, 125:566–568, 2018.

[9] Richard P. Stanley. *Enumerative Combinatorics, Volume 2*. Cambridge Studies in Advanced Mathematics 62. Cambridge University Press, 1999.

[10] Doron Zeilberger. The method of creative telescoping. *Journal of Symbolic Computation*, 11:195–204, 1991.

[11] Doron Zeilberger and Robert Dougherty-Bliss. Rutgers experimental mathematics seminar. https://sites.math.rutgers.edu/ zeilberg/expmath/, season 2022.

Manuel Kauers, Institute for Algebra, J. Kepler University Linz, Austria

Email address: manuel.kauers@jku.at

Christoph Koutschan, Radon Institute for Computational and Applied Mathematics, Austrian Academy of Sciences, Austria

Email address: christoph.koutschan@oeaw.ac.at

George Spahn, Department of Mathematics, Rutgers University (New Brunswick)

Email address: geosp98@gmail.com