Cutting 4 by n grids into two congruent pieces

Robert Dougherty-Bliss, Natalya Ter-Saakov, and Doron Zeilberger

October 13, 2025

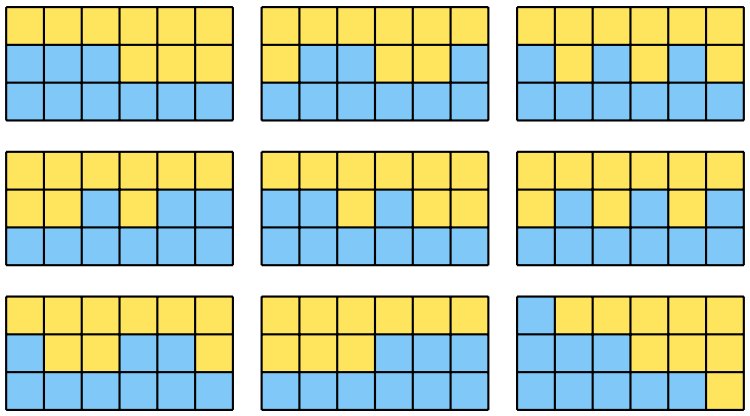
Pour Jean-Paul Delahaye, avec admiration

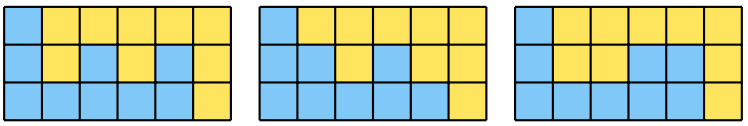
Introduction

In the March 2025 issue of *Pour La Science* (the French analogue of *Scientific American*), in his delightful monthly column [D], Jean-Paul Delahaye described in detail the solution of the following counting problem.

In how many ways can you cut a $3\times 2n$ *rectangle consisting of* $6n$ *unit squares (in other words a* $3 \times 2n$ *(colorless) checkerboard) into two* ***connected, congruent pieces?***

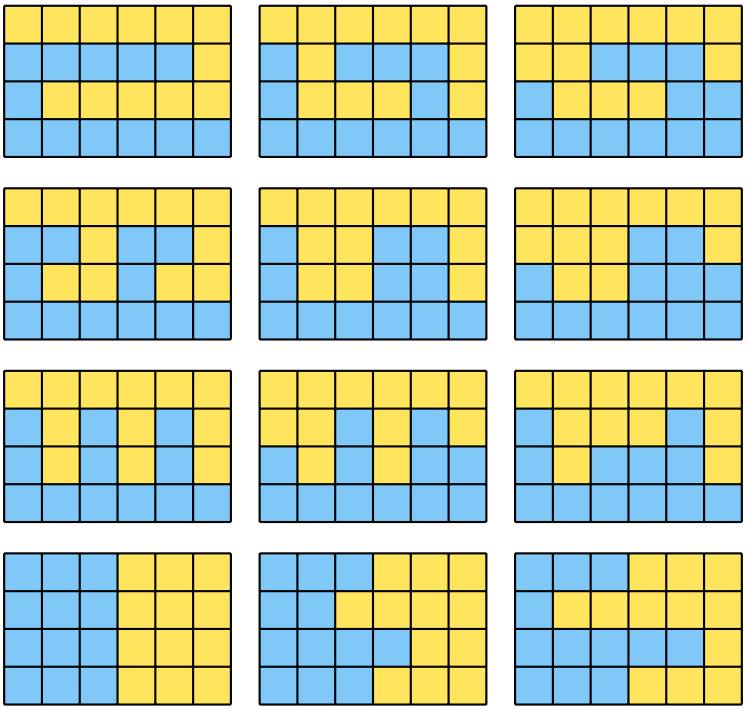
In collaboration with his *épouse*, Martine Raison, Delahaye used human ingenuity to prove that this number is given **explicitly** by the beautiful formula $2^{n+1} - n - 1$. Here are all twelve ways to divide 3×6 rectangles into two congruent, connected parts:





We asked ourselves, what about a $4 \times n$ checkerboard?

Here are twelve such ways to split a 4×6 rectangle into two congruent parts:



Now human ingenuity is still *necessary*, but (most probably) not *sufficient*. In collaboration with our beloved silicon friends we proved the following deep theorem.

Theorem 1. *Let c_n be the number of ways of cutting a $4 \times n$ grid into two (connected) congruent pieces. The (ordinary) generating function of c_n is*

$$\sum_{n=0}^{\infty} c_n x^n = \frac{x\left(2x^8 - 4x^7 + 8x^6 - 7x^5 + 3x^4 + 2x^3 - 5x^2 + x + 1\right)}{(x-1)^2\left(x^4 + 3x^2 - 1\right)\left(x^4 + 2x^2 - 1\right)}.$$

The coefficients have the (approximate) asymptotic expansion

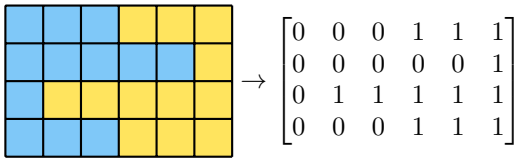
$$c_n \sim 1.93104(1 + 0.08417(-1)^n)(1.8174)^n.$$

For the sake of the OEIS the first 30 terms are:

$$1, 3, 5, 14, 22, 54, 84, 197, 305, 696, 1075, 2410, 3716, 8231, 12676,$$
$$27844, 42843, 93558, 143865, 312859, 480868, 1042624, 1602002,$$
$$3466064, 5324385, 11501987, 17665729, 38119718, 58540246, 126217718$$

To our delight, this sequence was not (as of Sept. 4, 2025) in the OEIS. We hope to submit it soon.

Our colorful rectangles are very pretty, but it is more convenient to represent such *decoupages* as $4 \times n$ matrices with entries 0 and 1, where the value at position ij indicates to which part that entry belongs. For example:



$$\rightarrow \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Definition 1. A $m \times n$ matrix M with entries in $\{0, 1\}$ that satisfies the rule

$$M_{ij} = 1 - M_{(m-i)(n-j)} \tag{1}$$

such that the 0's and 1's form exactly two connected components (using north-south-east-west adjacency) is called a *Graham matrix*. (Note that this rule uses *zero* indexing.)

Why Graham? For one, to honor the great Ron Graham, who advanced discrete mathematics in a substantial way; for another, because the problem is something like breaking apart a graham cracker fairly.

Note that if we drop the condition of the two pieces being congruent, and consider only square matrices, then we have the (computationally) challenging problem of the *Gerrymander* sequence [Sl], beautifully treated in [KKS] and extended in [GJ].

How did we find this beautiful generating function?

We asked our computer to show us all these creatures up to $n = 12$. After some thought, we convinced ourselves that our matrices could be recognized by reading them one column at a time; in other words, that they could be recognized by a *finite state machine*, or a *regular grammar*.

A finite state machine (also called a *deterministic finite automata*) is an object which detects words of a language. It reads in words one "symbol" at a time,

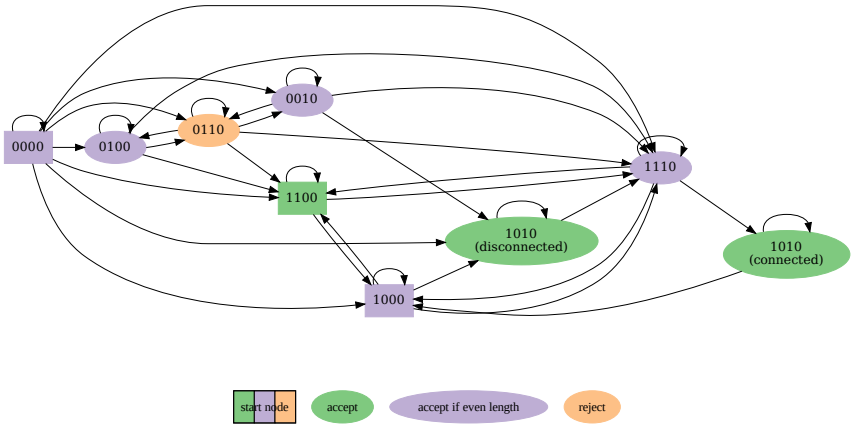


Figure 1: Finite state machine constructed to recognize $4 \times n$ Graham matrices. Rectangular states are start states, purple states accept if the string has an even number of symbols, and green states accept no matter what.

and uses these symbols to determine a walk on a finite graph. Some states in this graph are marked as *accepting*, and all others are *rejecting*. A string of symbols is accepted by the machine if the walk it determines ends in an accepted state.

In principle, it is not hard to see that our arrays can be recognized by a finite state machine. We can track connectivity data and update it whenever a column is read in, and we only need to consider the first half of the array thanks to rule (1). When we have read in the first half, we can merge with the implied connetivity data of the second half, and making sure everything works. The difficult part is actually *constructing* this finite state machine. In this problem it turns out that there are some shortcuts which make the machine a more reasonable size. If you are in a hurry, see Figure 1.

The motivation behind constructing a finite state machine is the *transfer matrix method* ([St], section 4.7). If a finite state machine has adjacency matrix T (sometimes called a transfer matrix), then the *symbolic inverse* $(I - xT)^{-1}$ contains the generating functions which count walks between different vertices of the state machine. The generating function we want is

$$\sum_{\substack{\text{start state } i \\ \text{accepting state } j}} [(I - xT)^{-1}]_{ij}.$$

We will now actually describe the grammar.

The grammar

A matrix that follows the complement rule (1)

$$M_{ij} = 1 - M_{(4-i)(n-j)}$$

is determined by its left half. To avoid double-counting symmetric arrangements, we need to stipulate some conditions.

1. The left-half of the bottom row of M is all 0's.
2. The first column of M has at least as many 0's as 1's.

We can enforce these conditions by a combination of rotations and reflections. First, force a 0 in the bottom left corner by rotating or reflecting. Then, there are two cases:

1. If the bottom-right corner is a 0, then the entire bottom row must be 0 to satisfy the connectivity rule. Then, either the first or last column of M satisfies the second condition, and we can force it to be the first column with a reflection.
2. If the bottom-right corner is a 1, then the top-left corner is a 0, and the first column is all 0's. The bottom and top rows consist of a run of 0's followed by a run of 1's. If the run of 0's in the bottom row is less than half of the row, then the run of 0's in the top row is at least half the row by rule (1). Reflect if necessary to make this happen in the bottom row.

With these stipulations, there are only $2^3 = 8$ possible column vectors on the left-hand side:

$$\begin{bmatrix}0\\0\\0\\0\end{bmatrix}\ \begin{bmatrix}0\\0\\1\\0\end{bmatrix}\ \begin{bmatrix}0\\1\\0\\0\end{bmatrix}\ \begin{bmatrix}0\\1\\1\\0\end{bmatrix}\ \begin{bmatrix}1\\0\\0\\0\end{bmatrix}\ \begin{bmatrix}1\\0\\1\\0\end{bmatrix}\ \begin{bmatrix}1\\1\\0\\0\end{bmatrix}\ \begin{bmatrix}1\\1\\1\\0\end{bmatrix}$$

According to the stipulations, the possible initial columns are $(0,0,0,0)$, $(1,0,0,0)$, and $(1,1,0,0)$. These columns all start with the 0's and 1's connected. The way we generate edges in Figure 1 is to look at all possible next columns and track connectivity information. Almost all edges are "trivial," meaning that the connectivity information does not change. For example, here are four potential

edges determined which begin at $(1,1,0,0)$:

$$\begin{bmatrix}1\\1\\0\\0\end{bmatrix} \to^{?} \begin{bmatrix}1\\1\\0\\0\end{bmatrix} \text{— yes; values remain connected}$$

$$\begin{bmatrix}1\\1\\0\\0\end{bmatrix} \to^{?} \begin{bmatrix}0\\0\\1\\0\end{bmatrix} \text{— no; 1's are disconnected}$$

$$\begin{bmatrix}1\\1\\0\\0\end{bmatrix} \to^{?} \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \text{— yes; values remain connected}$$

$$\begin{bmatrix}1\\1\\0\\0\end{bmatrix} \to^{?} \begin{bmatrix}1\\1\\1\\0\end{bmatrix} \text{— yes; values remain connected}$$

The only time that connectivity information is changed is when we encounter the column $(1,0,1,0)$. *Sometimes* when we see this column, new connected components spawn. For example, in the edge

$$\begin{bmatrix}1\\0\\0\\0\end{bmatrix} \to \begin{bmatrix}1\\0\\1\\0\end{bmatrix},$$

we get new 0 and 1 connected components. Accordingly, there are two states in Figure 1 for this column—one where the values are still connected, and one where they are disconnected.

This explains all of the edges. But which states should be accepting?

The basic condition is that, when the second half of the matrix is appended, the connected components need to "line up." If an even number of columns have been read in, then the next column of the final can be determined from the current column by applying rule (1). It is routine to check whether the connected components line up, even in the disconnected states, because the connectivity information also follows rule (1).

If an *odd* number of columns have been read in, then the last column is the "middle" column, and rule (1) must leave it fixed. The only columns for which this is true are $(1,1,0,0)$ and $(1,0,1,0)$. In this case, the first column in the right-half is determined by the *predecessor* of one of these columns. It is routine

to check that no matter what the predecessor is, the resulting matrix consists of exactly two connected components.

We save the last word in this section for the lonely column $(0,1,1,0)$, the only column which is *always* rejected.

Maple packages and asymptotics

We have constructed our finite state machine in a Maple package `Decoupage.txt` that can be gotten from

`https://sites.math.rutgers.edu/~zeilberg/tokhniot/Decoupage.txt` .

There are many ways to order the vertices of a finite state machine. We have chosen, essentially, an arbitrary one. Our adjacency matrix is can be obtained with `TransferMatrixMethods[transfer_matrix]`. It is as follows:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} .$$

The generating function matrix $(I - xM)^{-1}$ is too large to display here, but the LCM of the denominators of its entries is

$$(x-1)^2(x^2-x+1)(x^2+3x-1)(x^2+2x-1).$$

Using Maple's `RootOf` representation in concert with partial fraction decomposition, we can determine the asymptotic behavior of c_n starting from its generating function

$$\frac{x\left(2x^8-4x^7+8x^6-7x^5+3x^4+2x^3-5x^2+x+1\right)}{(x-1)^2\left(x^4+3x^2-1\right)\left(x^4+2x^2-1\right)}.$$

The smallest poles of this rational function are z and $-z$, where

$$z^{-1} = \frac{2}{\sqrt{2\sqrt{13}-6}} = 1.817354022\ldots$$

The asymptotic formula is

$$c_n \sim \frac{89z^2 - 92z + 218 - 86z^{-1}}{234}(-z)^{-n} + \frac{89z^2 + 92z + 218 + 86z^{-1}}{234}z^{-n}$$
$$\approx 1.93104(1 + 0.08417(-1)^n)z^{-n}.$$

The front of this article

`https://sites.math.rutgers.edu/~zeilberg/mamarim/mamarimhtml/decoupage.html`
contains some sample output files.

To see all $4 \times n$ Graham matrices for $1 \leq n \leq 12$ see the output file:

`https://sites.math.rutgers.edu/~zeilberg/tokhniot/oDecoupage1.txt` .

Future work

What about the analogous problem of $5 \times 2n$, $6 \times n$, $7 \times 2n$, $8 \times n$ checkerboards? As we mentioned earlier, it is not hard to see that for any *fixed* m, a finite state machine can recognize the matrices of size $m \times n$. It is another matter to actually *construct* these machines.

We plan to follow-up with this question in a separate publication where computers do everything—discover the grammar, process it, compute the generating function, and so on. But even computers can only go so far, and we are sure that humankind, and perhaps even computerkind will **never** have the generating function for the sequence counting $100 \times n$ Graham matrices.

References

[D] Jean-Paul Delahaye, *Combinatoire pour les rectangles*, *Logique & Calcul*, **Pour La Science** No. **569**, Mars 2025.

[GJ] Anthony J. Guttmann and Iwan Jensen, *The gerrymander sequence, or A348456*, arXiv:2211.14482 [math.CO], 2022. `https://arxiv.org/abs/2211.14482` .

[KKS] Manuel Kauers, Christoph Koutschan, and George Spahn, *How Does the Gerrymander Sequence Continue?*, J. Int. Seq., **25** (2022), Article 22.9.7. `https://cs.uwaterloo.ca/journals/JIS/VOL25/Kauers/kauers6.html`. arxiv version: `https://arxiv.org/abs/2209.01787` .

[Sl] Neil Sloane, OEIS sequence A348456 `https://oeis.org/A348456`.

[St] Richard P. Stanley, *"Enumerative Combinatorics"*, Volume 1, Cambridge University Press, First Edition, Wadsworth and Brooks/Cole, 1986.

Robert Dougherty-Bliss, Department of Mathematics, Dartmouth College, 29 N. Main Street, 6188 Kemeny Hall, Hanover NH 03755-3551 . Email: `robert dot w dot bliss at gmail dot com`

Natalya Ter-Saakov, Department of Mathematics, Rutgers University (New Brunswick), Hill Center-Busch Campus, 110 Frelinghuysen Rd., Piscataway, NJ 08854-8019, USA.
Email: `nt399 at rutgers dot edu` .

Doron Zeilberger, Department of Mathematics, Rutgers University (New Brunswick), Hill Center-Busch Campus, 110 Frelinghuysen Rd., Piscataway, NJ 08854-8019, USA.
Email: `DoronZeil at gmail dot com` .