Cutting Rectangles into Two Congruent Pieces

Robert Dougherty-Bliss and Natalya Ter-Saakov
(Dartmouth and Rutgers)

November 6th, 2025
Rutgers Experimental Mathematics Seminar

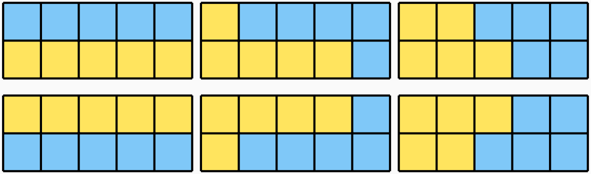
Joint work with Doron Zeilberger

Graham crackers; *Pour la Science*

Graham crackers

How many ways can you divide an $m \times n$ graham cracker into two congruent parts?

Here are some ways to do $2 \times n$:



The second row is a reflection of the first row.

We do not double-count symmetric arrangements!

Delahaye asked this in his March 2025 *Pour la Science* column.

His article covered:

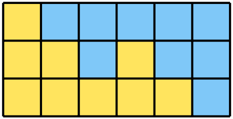
- $2 \times n$ (easy)
- $3 \times 2n$ (harder)

But $4 \times n$ was dismissed as *très difficile*.

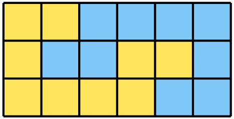
Delahaye counted ways to divide a $3 \times 2n$ rectangle and got $2^{n+1} - (n+1)$.

Delahaye counted ways to divide a $3 \times 2n$ rectangle and got $2^{n+1} - (n+1)$.

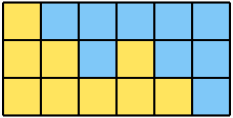
He categorized them as:



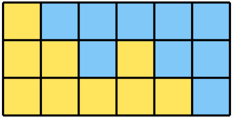
sans crochet



avec crochets

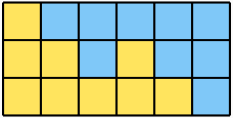


The divisions *sans crochet* can be further broken into sections by bottom row:



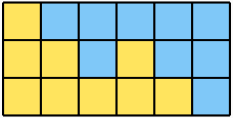
The divisions *sans crochet* can be further broken into sections by bottom row:

0. The entire bottom row is yellow



The divisions *sans crochet* can be further broken into sections by bottom row:

0. The entire bottom row is yellow
1. All but the last square of the bottom row is yellow



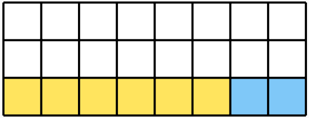
The divisions *sans crochet* can be further broken into sections by bottom row:

0. The entire bottom row is yellow
1. All but the last square of the bottom row is yellow
2. All but the last two squares of the bottom row are yellow

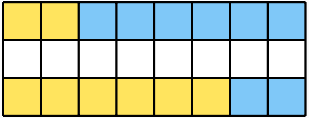
$\vdots$

n. The left half of the bottom row is yellow

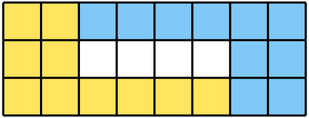
a. All but the last *a* squares of the bottom row are yellow.



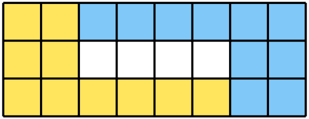
a. All but the last *a* squares of the bottom row are yellow.



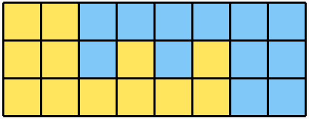
a. All but the last *a* squares of the bottom row are yellow.



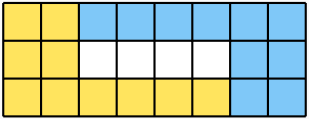
a. All but the last *a* squares of the bottom row are yellow.



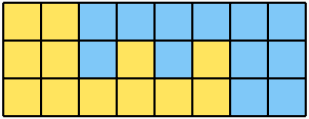
Choose the left half of the empty squares to decide the division: 2^{n-a} choices



a. All but the last *a* squares of the bottom row are yellow.



Choose the left half of the empty squares to decide the division: 2^{n-a} choices

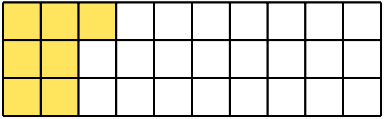


(Unless $a = 0$ which only has 2^{n-1} choices to account for symmetry)

The divisions *avec crochet* can be further broken into sections by the thickness of the left wall followed by the length of the overhang:

b. There are exactly *b* entirely yellow columns on the left-hand side.

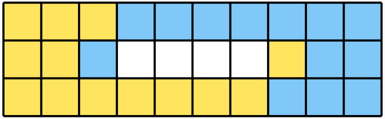
  c. There is an overhang of *c* squares.



The divisions *avec crochet* can be further broken into sections by the thickness of the left wall followed by the length of the overhang:

b. There are exactly *b* entirely yellow columns on the left-hand side.

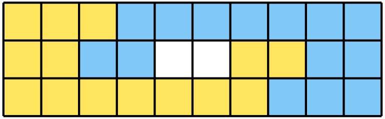
  c. There is an overhang of *c* squares.



The divisions *avec crochet* can be further broken into sections by the thickness of the left wall followed by the length of the overhang:

b. There are exactly *b* entirely yellow columns on the left-hand side.

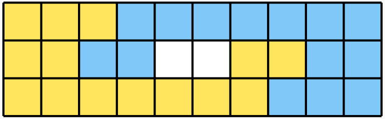
  c. There is an overhang of *c* squares.



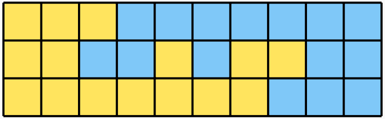
The divisions *avec crochet* can be further broken into sections by the thickness of the left wall followed by the length of the overhang:

b. There are exactly *b* entirely yellow columns on the left-hand side.

c. There is an overhang of *c* squares.



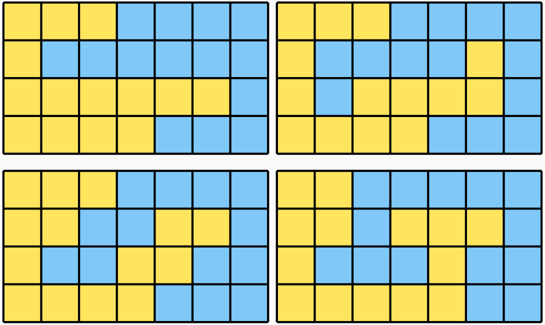
Choose the left half of the empty squares to decide the division: $2^{n-(b+c+1)}$ choices



Total count of divisions of a $3 \times 2n$ rectangle into two connected, congruent pieces:

$$2^{n-1} + \sum_{a=1}^{n} 2^{n-a} + \sum_{b=1}^{n-1} \sum_{c=1}^{n-b-1} 2^{n-(b+c+1)} = 2^{n+1} - (n+1)$$

Which of these should be counted as *avec crochet*?

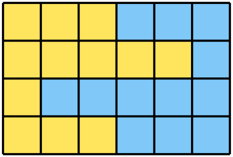


Decision problems

Definition

A *Graham matrix* is an $m \times n$ array of 0's and 1's such that:

- the 0's and 1's form exactly two simply connected, congruent regions (grid-connected)

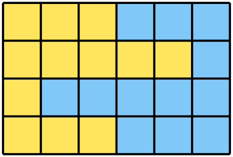


$$\rightarrow \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Definition

A *Graham matrix* is an $m \times n$ array of 0's and 1's such that:

- the 0's and 1's form exactly two simply connected, congruent regions (grid-connected)
- any combination of reflection, rotation, and swapping of 0's and 1's are considered equal.

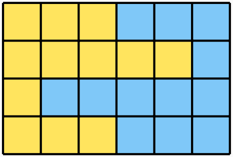


$$\rightarrow \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Definition

A *Graham matrix* is an $m \times n$ array of 0's and 1's such that:

- the 0's and 1's form exactly two simply connected, congruent regions (grid-connected)
- any combination of reflection, rotation, and swapping of 0's and 1's are considered equal.

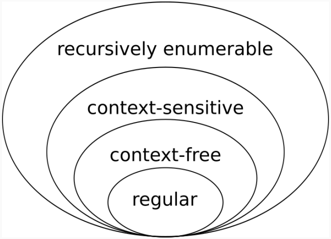


$$\rightarrow \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Main idea: Turn this into a *decision problem* that reads in one column at a time.

Parsers

Many combinatorial objects can be *parsed* by a computer program.



Chomsky's hierarchy

Certain kinds of programs imply generating function properties.

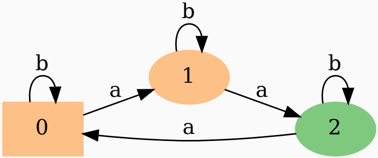
- Regular $\to$ rational generating functions
- Context-free $\to$ algebraic generating functions

Given a sufficiently restricted program, you get generating functions "for free."

Regular languages

The “most restrictive” kind of program is a *deterministic finite automata*.

It reads in symbols one at a time and performs a walk on a graph.



DFA that accepts strings of a’s and b’s with 2 $\pmod 3$ many a’s.

The object is accepted iff the walk terminates in an “accepting state.”

Languages accepted by a DFA are called *regular*.

If A is the adjacency matrix for the DFA, then

$$(I - xA)^{-1}$$

is matrix of generating functions that count walks.

This gives us an easy recipe to compute our generating function:

If A is the adjacency matrix for the DFA, then

$$(I - xA)^{-1}$$

is matrix of generating functions that count walks.

This gives us an easy recipe to compute our generating function:

1. Construct a DFA that recognizes our objects.

If A is the adjacency matrix for the DFA, then

$$(I - xA)^{-1}$$

is matrix of generating functions that count walks.

This gives us an easy recipe to compute our generating function:

1. Construct a DFA that recognizes our objects.
2. Compute the symbolic inverse $(I - xA)^{-1}$.

If A is the adjacency matrix for the DFA, then

$$(I - xA)^{-1}$$

is matrix of generating functions that count walks.

This gives us an easy recipe to compute our generating function:

1. Construct a DFA that recognizes our objects.
2. Compute the symbolic inverse $(I - xA)^{-1}$.
3. Sum up relevant entries for start and accepting states.

This is sometimes called the *transfer matrix method*.

Difficulties with the method

- It is not easy to tell if objects can be recognized with a DFA.

Difficulties with the method

- It is not easy to tell if objects can be recognized with a DFA.
- It is not easy to tell if a generic program can be turned into a DFA.

Difficulties with the method

- It is not easy to tell if objects can be recognized with a DFA.
- It is not easy to tell if a generic program can be turned into a DFA.
- Even if a DFA *exists*, it may be difficult to actually construct it.

Difficulties with the method

- It is not easy to tell if objects can be recognized with a DFA.
- It is not easy to tell if a generic program can be turned into a DFA.
- Even if a DFA *exists*, it may be difficult to actually construct it.
- The matrix inverse $(I - xA)^{-1}$ may be difficult to compute.

The Grammar

Graham matrices must follow the complement rule,

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

Graham matrices must follow the complement rule,

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

We build the left half of Graham matrix column by column, then use this rule to fill in the rest. Here are the possible columns:

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

Some adjacencies are easy to see:

$$\begin{bmatrix}0\\0\\0\\0\end{bmatrix}$$ can be followed by anything that starts and ends in a 0.

$$\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$$ can be followed by $$\begin{bmatrix}0\\1\\1\\0\end{bmatrix}$$ but not by $$\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$$

Some adjacencies are more complex:

$\begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$ looks like the 1's are disconnected. But $\begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ doesn't.

We track this by having two states corresponding to this column, one for disconnected 1's and one for disconnected 0's.

Valid starting states

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

Valid accepting states

If n is odd, the complement rule applied to the middle column gives

$$M_{i\left(\frac{n+1}{2}\right)} = 1 - M_{(n+1-i)\left(\frac{n+1}{2}\right)}$$

meaning that the middle column needs to be its own complement's reflection.

Valid accepting states

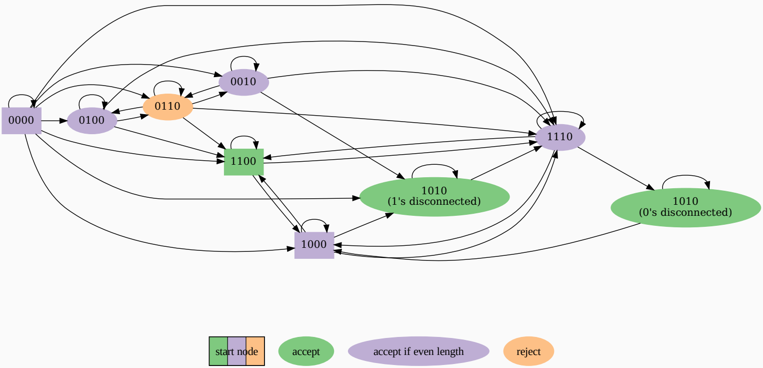
If n is odd, the complement rule applied to the middle column gives

$$M_{i\left(\frac{n+1}{2}\right)} = 1 - M_{(n+1-i)\left(\frac{n+1}{2}\right)}$$

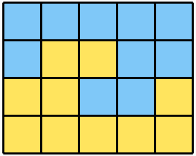
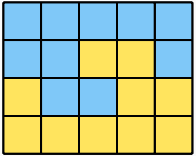
meaning that the middle column needs to be its own complement's reflection.
If n is even,

$$\left[\begin{array}{c:c} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{array}\right], \left[\begin{array}{c:c} 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 1 \end{array}\right], \left[\begin{array}{c:c} 0 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 1 \end{array}\right], \left[\begin{array}{c:c} 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{array}\right], \left[\begin{array}{c:c} 1 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{array}\right], \left[\begin{array}{c:c} 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{array}\right], \left[\begin{array}{c:c} 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 0 \end{array}\right], \left[\begin{array}{c:c} 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 0 \end{array}\right]$$

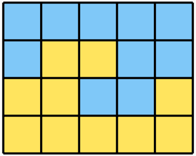
The DFA


0000
0100
0110
0010
1100
1000
1110
1010
(1's disconnected)
1010
(0's disconnected)
start node
accept
accept if even length
reject

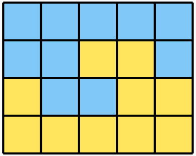
Gabriel Gendler pointed out to us that we counted both

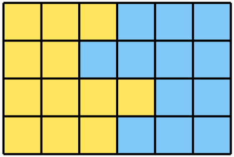
 and 

Gabriel Gendler pointed out to us that we counted both

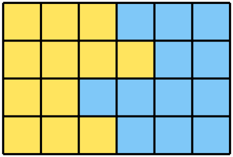


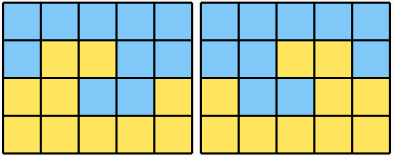
and



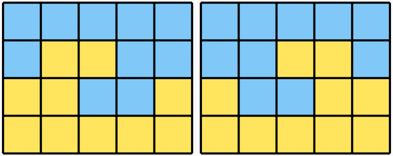


and



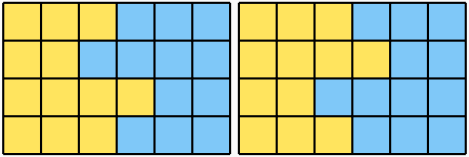


This double count only happens when the first column is $\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$.

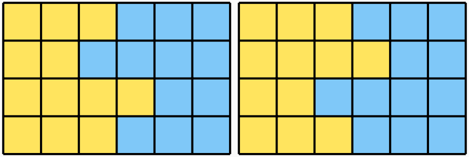


This double count only happens when the first column is $\begin{bmatrix}1\\1\\0\\0\end{bmatrix}$.

The fix: Create a second version of that state for starting only and declare that it can be followed by itself and $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$ but not $\begin{bmatrix}1\\1\\1\\0\end{bmatrix}$.



This double count only happens when the accepting state starts and ends in 0 (which can only happen if n is even).



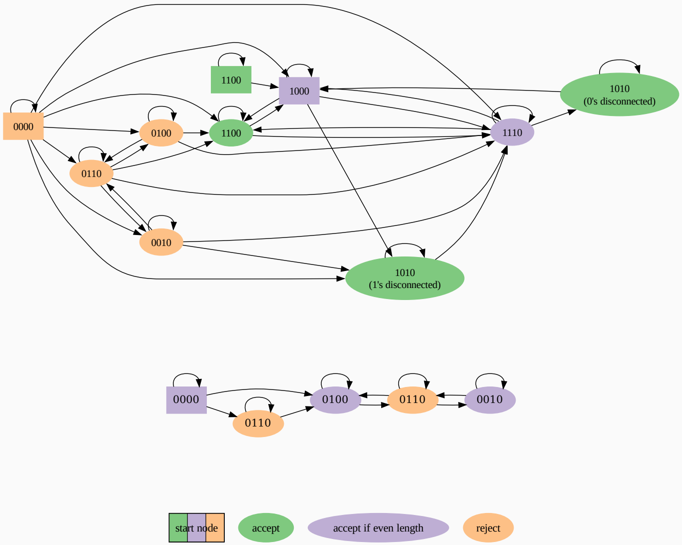
This double count only happens when the accepting state starts and ends in 0 (which can only happen if n is even).

The fix: Stop accepting on states that start and end with 0. Construct a second component to the DFA that includes only the states

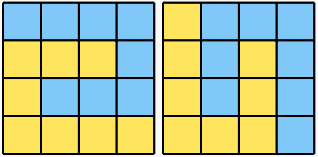
$$\begin{bmatrix}0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\\0\end{bmatrix},$$

and forces the second state to appear before the third one does.

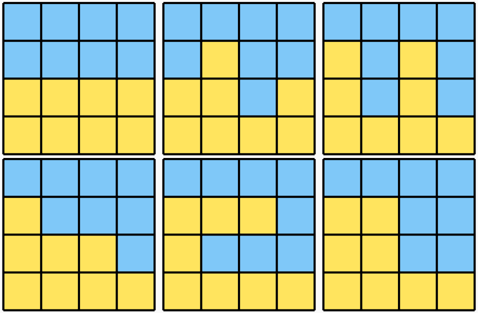
The Fixed DFA



Gabriel also pointed out that when $n = 4$, we double count reflections over $y = x$ such as



We counted all the squares by hand:



Generating functions / Generalizations

Computational results

Getting the grammar right is annoying, but the resulting adjacency matrix is not so big:

$$A = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{bmatrix}$$

The entries of $(I - xA)^{-1}$ are too big to show.

The LCM of the denominators is

$$\left(x^4 + 2x^3 - 3x^2 + 4x - 1\right)(x - 1)^2\left(x^3 + x^2 - 3x + 1\right).$$

This is not exactly the denominator of our generating function.

We need to swap $x \leftarrow x^2$:

The entries of $(I - xA)^{-1}$ are too big to show.

The LCM of the denominators is

$$\left(x^4 + 2x^3 - 3x^2 + 4x - 1\right)(x-1)^2\left(x^3 + x^2 - 3x + 1\right).$$

This is not exactly the denominator of our generating function.

We need to swap $x \leftarrow x^2$:

$$\left(x^4 - x^2 + 1\right)\left(x^4 + 3x^2 - 1\right)(x-1)^3(x+1)^3\left(x^4 + 2x^2 - 1\right).$$

The denominator of any derived generating function will divide this.

Theorem

The number $G_n := G(4, n)$ of $4 \times n$ Graham matrices has generating function

$$\sum_{n \geq 0} G_n x^n = \frac{p(x)}{(1-2x^2-x^4)(1-x)^2(1+x)(1-3x^2-x^4)},$$

where

Theorem

The number $G_n := G(4, n)$ of $4 \times n$ Graham matrices has generating function

$$\sum_{n\geq 0} G_n x^n = \frac{p(x)}{(1-2x^2-x^4)(1-x)^2(1+x)(1-3x^2-x^4)},$$

where

$$p(x) = x(1 + 2x - 4x^2 - 11x^3 + 11x^4 + 38x^5 - 35x^6 - 50x^7 + 50x^8 - 5x^9 + 5x^{10} + 24x^{11} - 24x^{12} + 6x^{13} - 6x^{14})$$

There are also implied recurrences, asymptotics, and so on.

Generalizing

Nothing we said was *that* special about four rows.

We believe that a DFA *exists* for any fixed number of rows.

But. . .

- It will have a lot of states.
- It is very difficult to generate by hand.
- It is unclear how far the symbolic inverses will go.

We hope to report back soon!