Counting Maximal Seat Assignments that Obey Social Distancing

Presented by: George Spahn

Joint work with: Doron Zeilberger

Proctoring

- No two students may sit adjacent horizontally or vertically
- Perhaps the students sit randomly without violating the rules
- Will we run out of space?
- What density can we expect?

x		x		
				x
	x		x	
x		x		x

Definitions

- Input: The dimensions of the grid of seats $r \times c$
- Input: A set of forbidden patterns, S
- A seating assignment can be represented as an $r \times c$ matrix of 0s and 1s (1s represent occupied seats)
- An assignment is said to be maximal if it satisfies 2 properties:
  - None of the forbidden patterns are present.
  - Changing any 0 to a 1 causes a forbidden pattern to be present.

Example

Not Maximal

1	0	0	0	0
0	0	1	0	1
0	1	0	1	0
1	0	1	0	1

8 1s

Maximal

1	0	0	1	0
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0

7 1s

Questions

- Given r,c, and S:
- What is the maximum and minimum density of a maximal configuration?
- How many maximal configurations are there?
- If I were to select a maximal configuration uniformly at random, what is the expected density?

Max Density

1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1

- Density $= 0.5$
- What we might hope for when proctoring :D
- Will the min density be similar?

Min Density

		1				1	
1				1			
			1				1
	1				1		

- How would you feel if the students ended up like this?
- Density $= 0.25$
- Can we go smaller?

Min Density

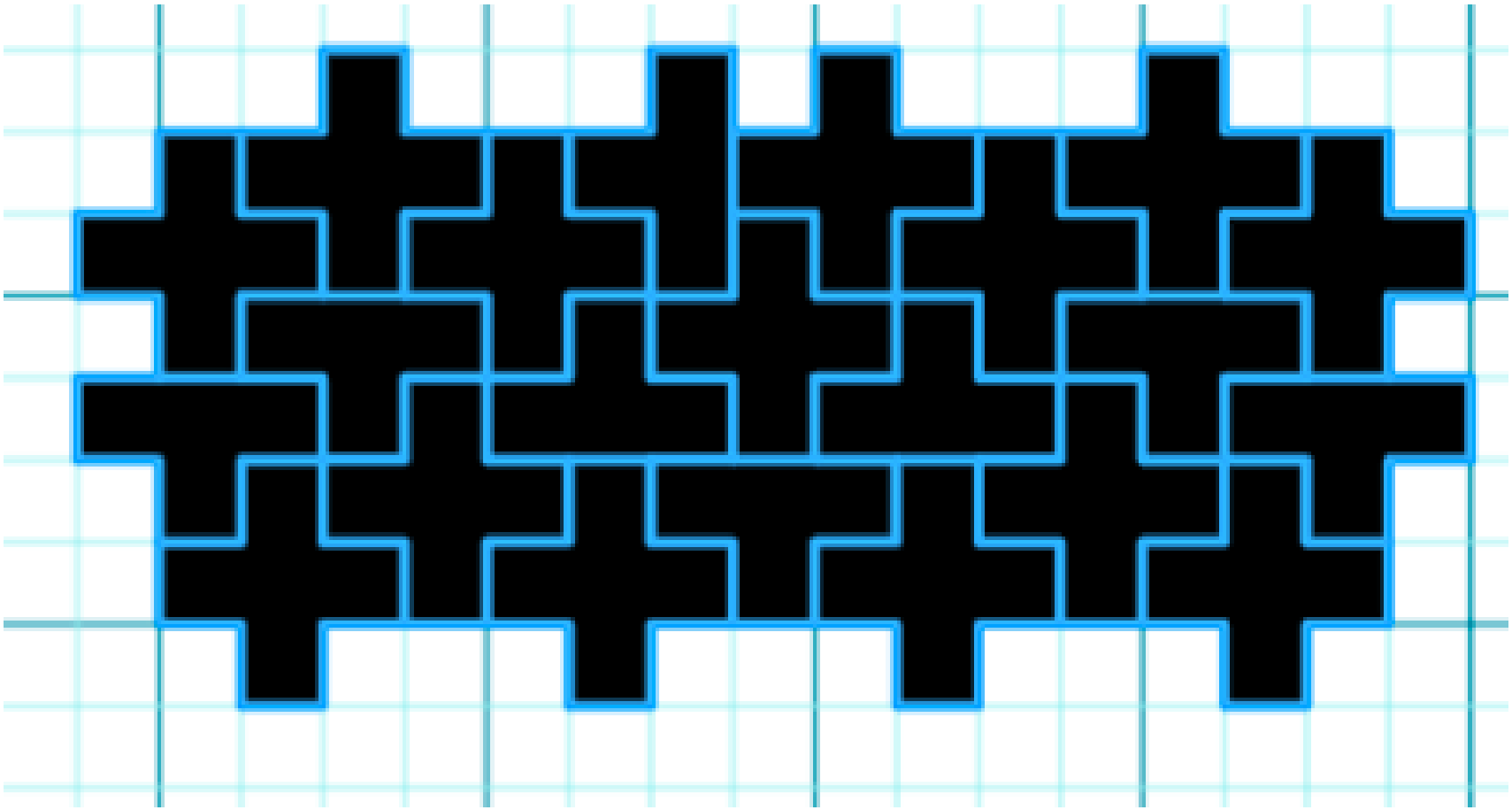
			0				
		0	1	0			
			0				

					0		
			0	0	1	0	
		0	1	0	0	0	
			0	0	0	1	0
			0	1	0	0	
				0			

Min Density

- Limiting min density $= 0.2$
- If we limit ourselves to 4 rows the density can never drop below 0.25
- With 5 rows we require at least 21 columns to get density below 0.25
- With 6 rows we require 15 columns.

Min Density 6x15



Credit to Julius Sun

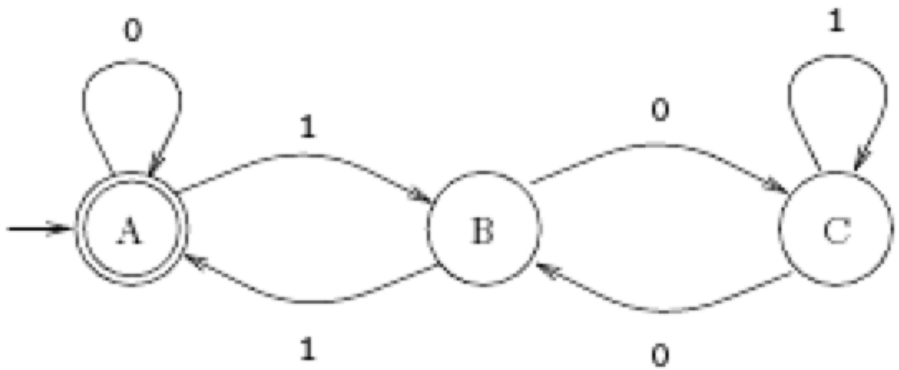
Automatic Counting of Maximal Assignments

- The general case of a r by c grid is a bit too hard
- We can fix r and consider the sequence counting the number of maximal assignments as a function of the number of columns
- An assignment is just a sequence of columns that satisfies some local conditions...

Finite State Machine!

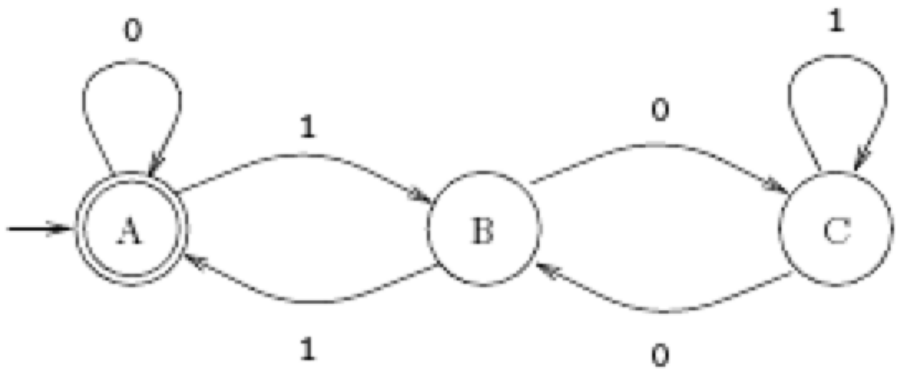
- A finite state machine (FSM) is a directed graph.
- The edges (transitions) are labeled with symbols from an alphabet
- The vertices (states) are used to store information as we read a sequence of symbols
- Some of the states are labelled as ACCEPT states
- If a sequence of symbols leads to an ACCEPT state, than the word formed by that sequence of symbols is accepted

Example



Checks whether a binary number is ...

Example



Checks whether a binary number is ... divisible by 3

Finite State Machine!

- For us, the symbols are possible columns
- 2^r symbols in our alphabet
- A r by c maximal assignment will be an accepted word of length c

When to REJECT

There are two ways that a seating assignment can fail to be maximal:

- There are two adjacent 1s
- There is a 0 with no adjacent 1

A 0 with an adjacent 1 is said to be a satisfied 0. If we encounter an unsatisfied 0 we should REJECT!

Detecting 2 adjacent 1s

- As the state machine reads in each column, it needs to be on the lookout for two adjacent 1s
- Reject if two adjacent 1s in same column
- Also store the contents of the previous column in the state.
- If a 1 would be adjacent to a 1 that we haven't read yet, we can reject later.

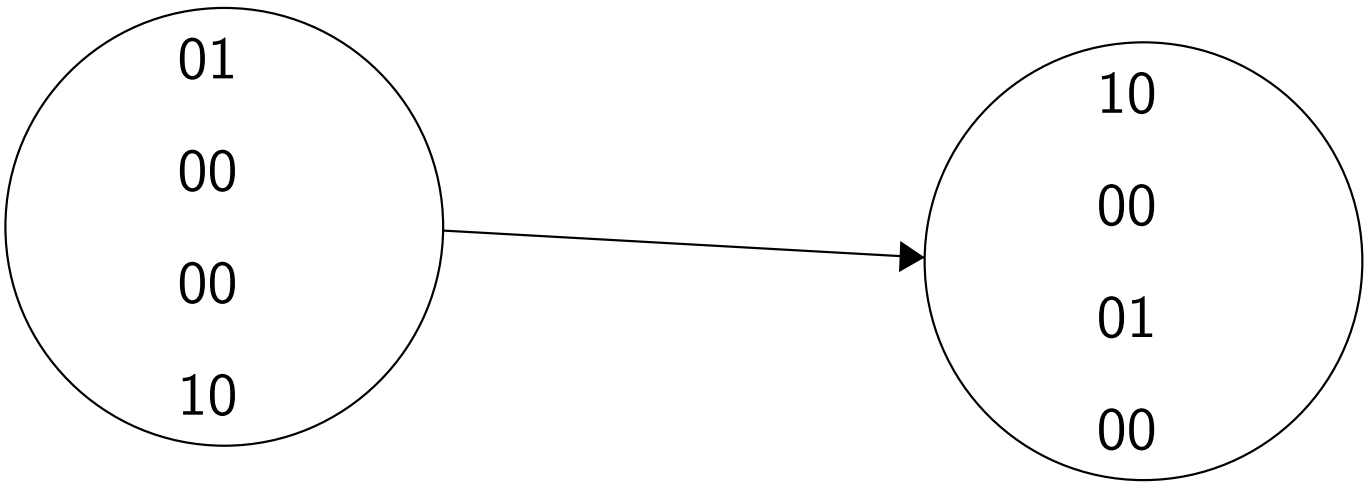
Detecting unsatisfied 0s

- We don't have enough information to determine whether a 0 in the current column is satisfied.
- Instead check that each 0 in the previous column is satisfied
- Need to store the previous TWO columns in the state.
- Total of 2^{2r} states, one for each possible contents of the previous two columns

Example

0	1	
0	0	
0	0	
1	0	

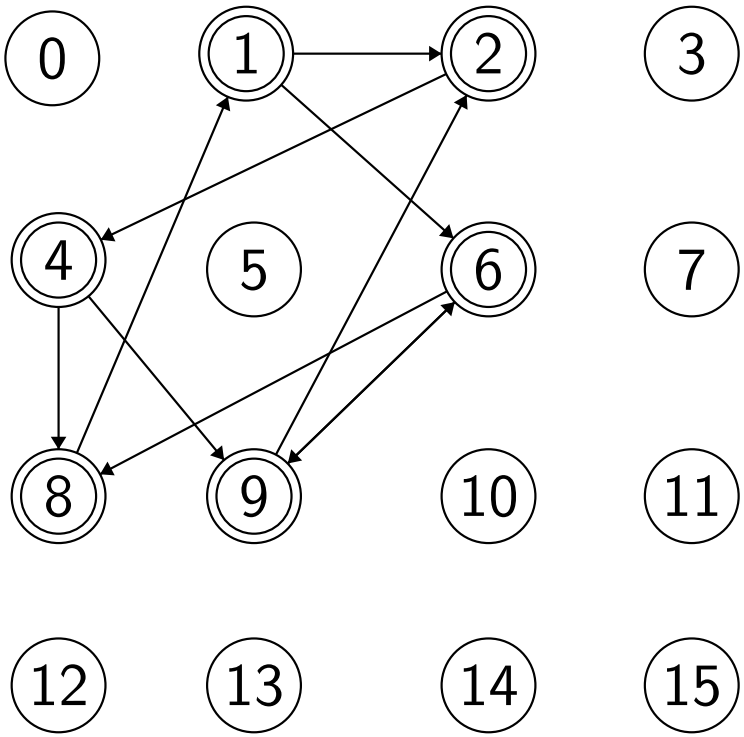
Example



Which states are ACCEPT states?

- If we reach the end of input, should we ACCEPT or REJECT?
- Still need to check 0s in most recent column!
- If all those 0s are satisfied, then ACCEPT

$r = 2$



Initialization is omitted

Counting Paths

- Now let's just say there is 1 big ACCEPT state that all the ACCEPT states transition to.
- Each maximal assignment corresponds to a path from the START state to the ACCEPT state.
- The number of maximal assignments with c columns is thus the number of paths from START to ACCEPT with length c!

Transition Matrix

- We can count paths using matrices!
- Let M be the adjacency matrix of the state machine
  - Each state becomes a row and column of the matrix
  - A valid transition from state i to j is represented by a 1 in the $[i, j]$ entry of M
  - All other entries are 0
- $M^2[i, j]$ now counts the number of paths from i to j of length 2
- $M^c[i, j]$ now counts the number of paths from i to j of length c

$r = 2$ sequence

- We now can compute the sequence giving the number of maximal assignments
- For $r = 2$: 2, 2, 4, 6, 10, 16, 26, 42, 68, 110, 178, 288, 466, 754, ...
- Notice anything about this sequence?

$r = 2$ sequence

- We now can compute the sequence giving the number of maximal assignments
- For $r = 2$: 2, 2, 4, 6, 10, 16, 26, 42, 68, 110, 178, 288, 466, 754, ...
- Notice anything about this sequence?
- Twice the Fibonacci sequence!

Why?

No valid assignment ends with the 11 or 00 columns...

...	1
...	0

Why?

No valid assignment ends with the 11 or 00 columns...

...	1
...	0

...	1	0
...	0	1

...	1	0	0
...	0	0	1

$r = 3, 4$

- Maximal assignments with 3 rows:
- 2, 4, 10, 18, 38, 78, 156, 320, 654, ...
- Maximal assignments with 4 rows:
- 3, 6, 18, 42, 108, 274, 692, 1754, 4442, ...
- A157049, A157050 is the OEIS

A157049

Number of n X 3 0..4 arrays with each element equal to the number its horizontal and vertical zero neighbors.

- Again looking at grids with 3 rows.
- Here the entries are integers from 0 to 4
- Each 0 is adjacent to 0 0s. (Seem Familiar?)
- Each other grid location is uniquely determined by how many 0s it is next to.
- Why "maximal"?

Maximal Independent Sets in the Grid Graph

- Recall from graph theory that an Independent Set of vertices in a graph is a subset of the vertices with no adjacencies.
- Max Independent Set denoted $\alpha(G)$
- In general computing $\alpha(G)$ is hard, for an n-vertex graph the best known algorithm takes time $O(1.2^n)$

Generating Function

- Using this method we can get generating functions for these sequences without too much extra work. The sequence:

$$f(n) = M^n[1, 2]$$

has the generating function:

$$F(x) = \sum f(n)x^n$$

A system of Equations

1. For each node, the number of ways to get there in n steps is equal to the sum of the ways to get to the preceding nodes in $n-1$ steps.
2. Let $F_i(x)$ be the generating function for the number of ways to reach state i from START in n steps.
3. Then
$$F_i(x) = \sum_j F_j(x) \cdot x$$
where the sum is taken over preceding nodes
4. Big system of equations is great for Maple!

Shortcut

- Ignoring matrices for a second...

$$F(x) = \sum M^n x^n \tag{1}$$
$$= \sum (Mx)^n \tag{2}$$
$$= \frac{1}{1 - Mx} \tag{3}$$
$$\approx (I - Mx)^{-1} \tag{4}$$

- This matrix, N, contains all of our desired generating functions!
- $N[1, 2]$ gives the generating function for the number of paths from START to ACCEPT.

Results

- Here is the generating function for $r = 3$:
- $$-\frac{2x^6 - x^5 + x^4 - x^3 - x^2 - x - 1}{x^5 + x^4 - 3x^3 - x^2 - x + 1}$$
- In the OEIS this appears as an "empirical generating function".
- Now we have a rigorous proof!

Back to Density

- What if want to compute the average density over all these maximal assignments?
- Modify the transition matrix M.
- Previously it had entries either 1 or 0 indicating edges in the graph.
- Now replace the ones with powers of z.
- z^t will indicate that the corresponding transition added t 1s to the assignment.

Density Polynomials

- Previously $M^n[1,2]$ counted the number of maximal assignments with n columns.
- Now it is a polynomial in z.
- The coefficient of z^k gives the number of maximal assignments with n columns and k total 1s.

Example

For 3x3 assignments we get the polynomial:

$$g(z) = z^5 + z^4 + 8z^3$$

	1		1		1			1	1		
1		1		1		1					1
	1		1		1			1		1	

The average density is 0.37. One way to compute this is:

Example

For 3x3 assignments we get the polynomial:

$$g(z) = z^5 + z^4 + 8z^3$$

	1		1		1			1	1								
1		1		1		1					1						1
	1		1		1			1		1				1			

The average density is 0.37. One way to compute this is:

$$\frac{g'(1)}{9g(1)}$$

Bivariate Generating Functions

- We can also include z in the generating function!
- The coefficient of $x^j z^k$ now gives the number of maximal assignments with j columns and k total 1s.
- For 3 rows: $$-\frac{2x^6z^5 - x^5z^4 + x^4z^3 + x^3z^4 - 2x^3z^3 - x^2z^2 - xz^2 - 1}{x^5z^4 + 2x^4z^4 - x^4z^3 + x^3z^4 - 4x^3z^3 - x^2z^3 - xz + 1}$$
- Maple can extract coefficient polynomials using Taylor series!

Limiting Density

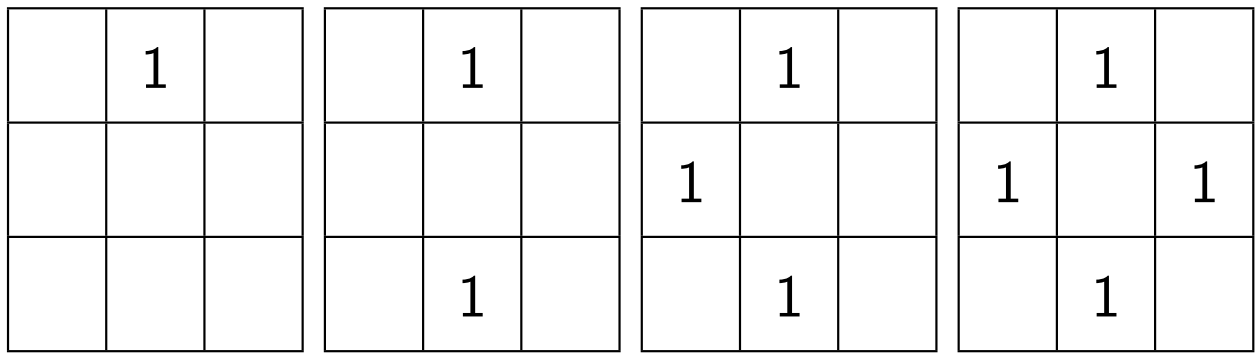
- We can look at the roots of the denominator of the generating function to get asymptotics.
- We can compute the limiting average density over all maximal assignments as the number of columns goes to infinity.
- For $r = 3$ we compute $d = 0.352...$
- For $r = 4$ we compute $d = 0.347...$
- For $r = 5$ we compute $d = 0.342...$
- Only slightly smaller than the 3 by 3 case, 0.37.

Other Distributions

- So far we have computed the average density assuming a uniform distribution on maximal assignments.
- A priori there is no reason that students filtering into a classroom would obey that distribution.
- We can use simulation to look at other models where seats are assigned sequentially.

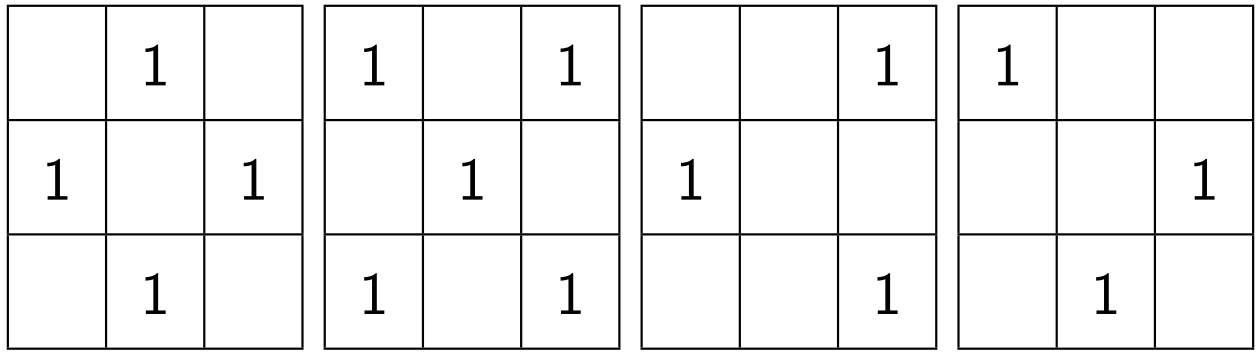
Random Sequential Adsorption

- In this model each student sequentially picks a seat uniformly at random from the set of available seats.



- For 3x3 grids, the average density is now 0.44
- This is much higher than the 0.37 from the uniform distribution. Why?

3x3 density



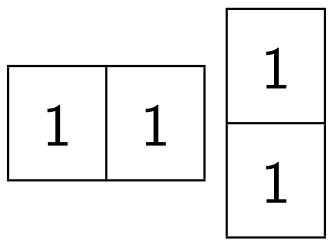
- Under the uniform distribution, the middle seat is taken only 1/10 of the time.
- Under sequential adsorption, it is taken 1/9 of the time by the first person to sit down, immediately forcing the maximum possible density!

Random Sequential Adsorption

- We ran a simulation with 100 by 3 matrices.
- The average density was 0.394
- This is notably higher than the limiting density under the uniform distribution, 0.352.

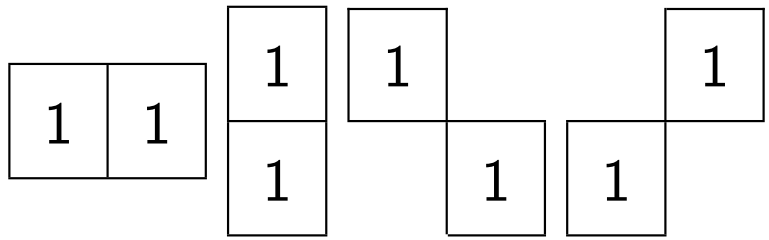
Generalizing S

- Recall that S is the set of violations.
- So far we have looked at the specific case where S has two elements: horizontal and vertical adjacencies
- We represent these violations as polyominoes



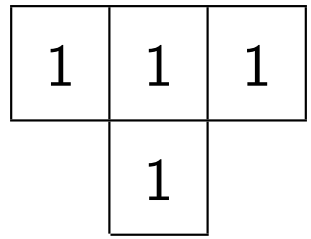
Non-Attacking Kings

If the seats are not allowed to be adjacent diagonally, we get the famous non-attacking kings problem.



Housing Developments

The paper that inspired this research was interested in avoiding the T-piece. "Packing density of combinatorial settlement planning models"



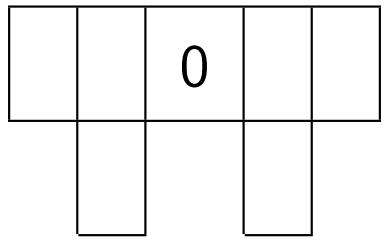
The idea is that you don't want any houses to be totally blocked from the sun (and there is no sun from the North)

Checking Arbitrary Patterns

- It is now not sufficient to only keep track of the previous two columns.
- Let W be the largest width of any polyomino.
- To avoid adding the polyomino we need to know the previous $W - 1$ columns.
- What about maximality?

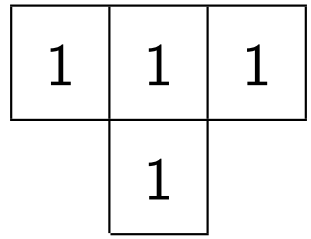
Ensuring Maximality

For the T, we have $W = 3$.



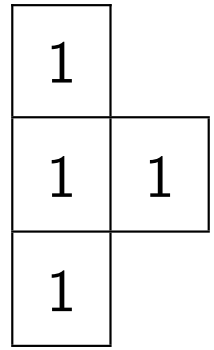
- If we store the previous 4 columns, then we can ensure that the 0s in the 3rd column are satisfied when adding the next column.
- In general we will have to store the previous $2W - 2$ columns.
- This gives a total of $2^{r(2W-2)}$ states.

Down Facing T



- Maximally avoiding the T with $r = 3$ gives the following sequence:
- 1, 1, 10, 19, 41, 105, 269, 651, 1560, ...
- Not yet in the OEIS
- Sadly need to make the code faster to compute the generating function, inverting the 189 x 189 matrix was taking too long.
- What if we rotate the T to make W smaller?

Right Facing T



- Maximally avoiding this T with $r = 4$ gives the following sequence:
- 1, 6, 19, 63, 208, 687, 2269, 7494, ...
- A189735 in the OEIS
- It satisfies

$$a(n) = 3a(n-1) + a(n-2)$$

- Open Problem: Give a combinatorial explanation

Further Generalizations

- Counting 0-1 matrices that satisfy a given set of properties is something that comes up often.
- The finite state machine approach is well suited to tackling many of these problems.
- I hope to abstract the code so that a user just need to implement a few of the functions to solve their own version of the problem.
- Just need a set of states, and a way to compute the transitions between those states

Further Generalizations

- The states contain some finite amount of information about the columns seen so far (such as the contents of the previous 10 columns).
- This finite information, plus the contents of the current column, must be enough to detect any error.
- The data needs to be hashable, so we can easily tell when two states are the same.
- The transition function can be built on top of whatever state system the user wants.
- My code + Maple will do the rest!

The End

Thank you for listening!