

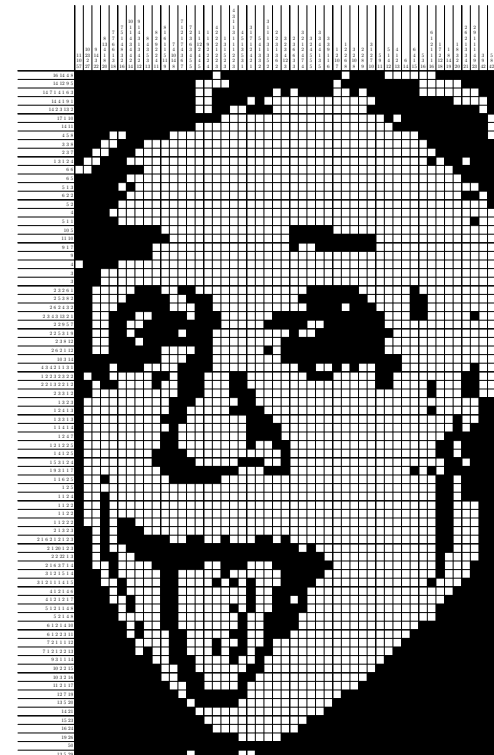
DT&N

Discrete Tomography and Nonograms

Walter Kosters, Universiteit Leiden

April 16, 2009; Helvoirt

www.liacs.nl/home/kosters/



When talking about **Japanese puzzles**, everyone thinks of **Sudoku**.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

When talking about **Japanese puzzles**, everyone thinks of **Sudoku**.

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

source: Wikipedia

But we will talk about **Nonograms** today.

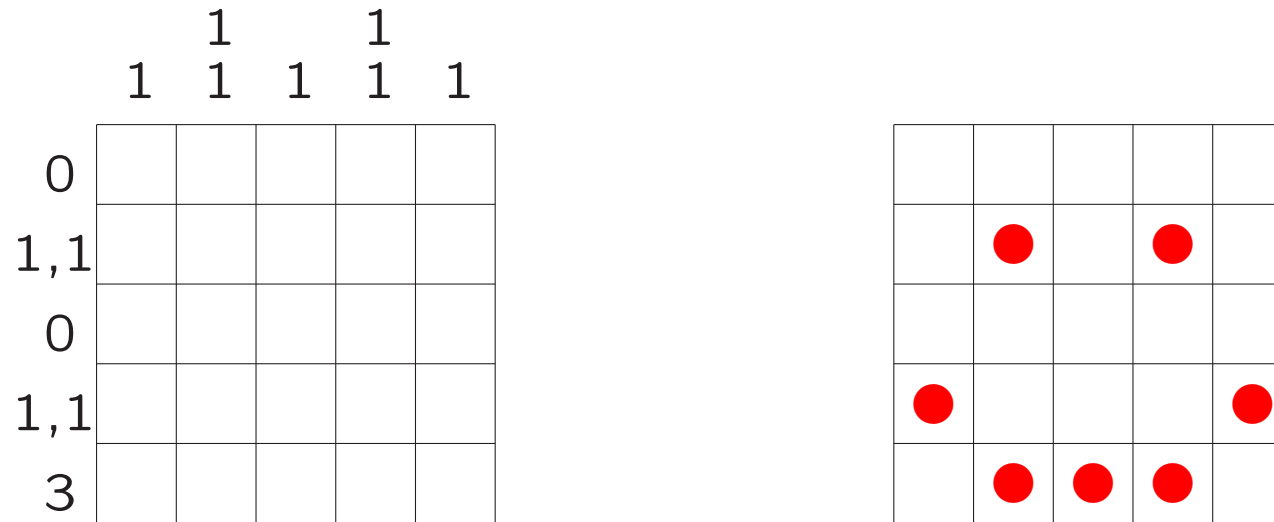
A **Nonogram** is a puzzle; a small example:

		1		1	
	1	1	1	1	1
0					
1,1					
0					
1,1					
3					

Next to each row and column we enumerate the lengths of consecutive series of **red** pixels.

Where are these **red** = black pixels?

The (unique) solution looks like this:



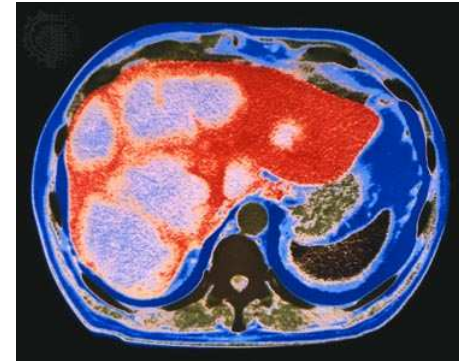
Next to each row and column we enumerate the lengths of consecutive series of **red** pixels — in order.

Why are scientists interested in Nonograms?

Tomography tries to solve the following problem:

How to reconstruct an object from **projections**?

Examples:



- Solve Nonograms
- How do we look like, given CT-scans? (Computerized Tomography = $CT \supseteq DT = \text{Discrete Tomography}$)
- Where are the “holes” in a diamond?

In **Discrete Tomography** we try to reconstruct an object from its projections.

An object “is” a finite subset of \mathbb{Z}^2 (so integer points in 2D space).

A **projection** gives *all* relevant “linesums” over lines parallel to a given line, e.g., all horizontal lines and all vertical lines (2 projections). It is also possible to use all lines through a given point.

In CT one typically has *many* projections, in DT a few.

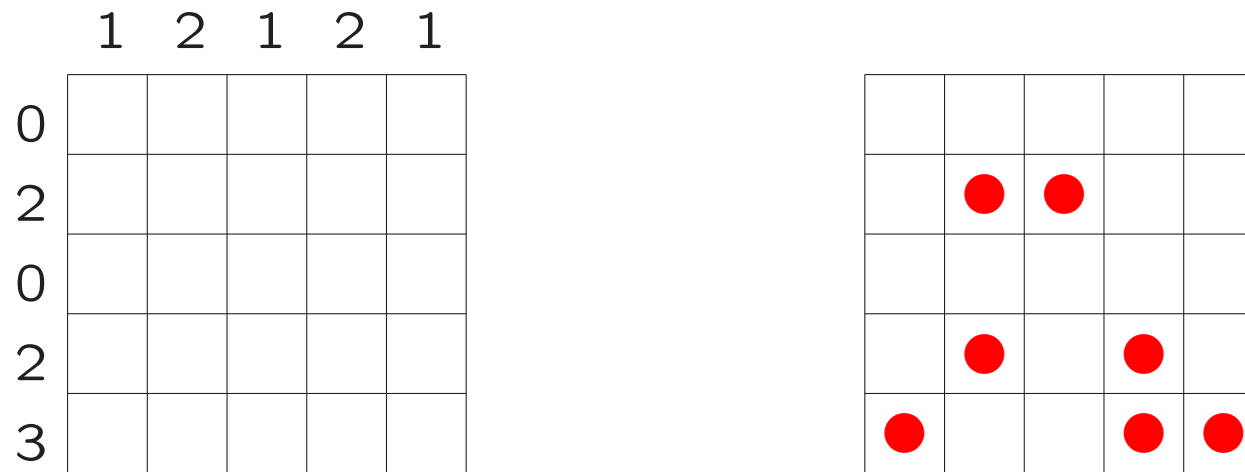
A small example of a Discrete Tomography problem:

	1	2	1	2	1
0					
2					
0					
2					
3					

Next to each row and column we give the (total) number of **red** pixels.

Where are these **red** = black pixels?

A (non-unique) solution looks like this:



Next to each row and column we give the (total) number of **red** pixels.

The problem can be defined more general:

Given an unknown function f on some domain D (discrete, or just some subset of \mathbf{R}^n), with a *discrete* range $\subseteq \mathbf{R}$, the task is to (approximately) reconstruct f , given sums (integrals) over certain subsets of D .

In our case, the range is $\{0, 1\} = \{\text{white, black}\}$.

General reference: G.T. Herman & A. Kuba, Discrete tomography, Foundations, algorithms and applications, Birkhäuser, 1999.

Usually we have three tasks:

Consistency Does an object with the given projection values (a “solution to the puzzle”) exist?

Uniqueness Suppose there is a solution. Does there exist another one?

Reconstruction Construct a solution.

All three problems, with horizontal and vertical projections, are solved in polynomial time by **Ryser's Theorem** from 1957.

But the problems for 3 or more projections are NP-hard!

$$2 \neq 3$$

If you were to use a flashlight in 2D, it would flicker like when throwing a stone into the water.

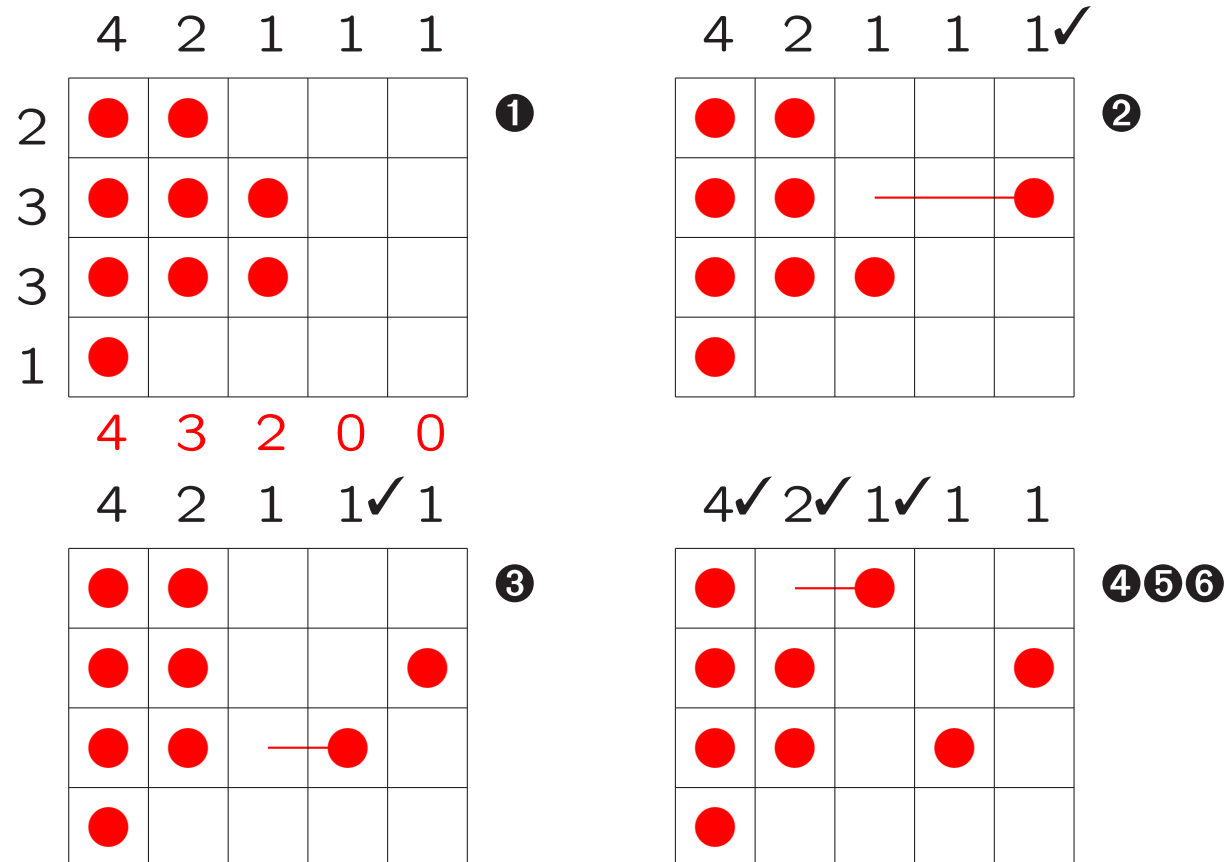
Ryser's Theorem Given two vectors $R = (r_1, \dots, r_m) \in \mathbb{N}_0^m$ (the row sums) and $S = (s_1, \dots, s_n) \in \mathbb{N}_0^n$ (the column sums) with the same total sum: $\sum_{i=1}^m r_i = \sum_{j=1}^n s_j$. Then there is a binary matrix $A = (a_{ij})$ with $\sum_{j=1}^n a_{ij} = r_i$ ($1 \leq i \leq m$) and $\sum_{i=1}^m a_{ij} = s_j$ ($1 \leq j \leq n$) if and only if $\sum_{j=\ell}^n s'_j \geq \sum_{j=\ell}^n \bar{s}_j$ ($2 \leq \ell \leq n$).

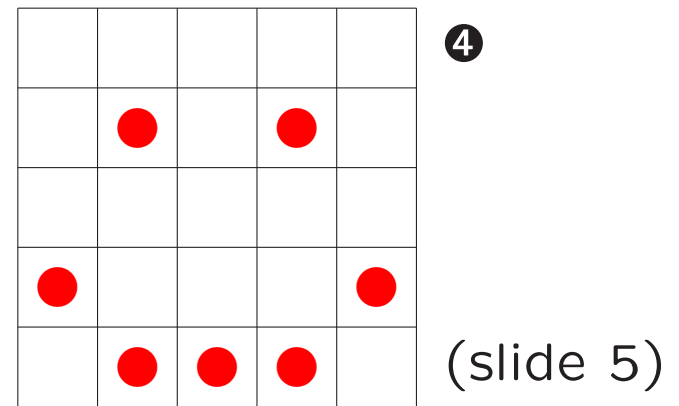
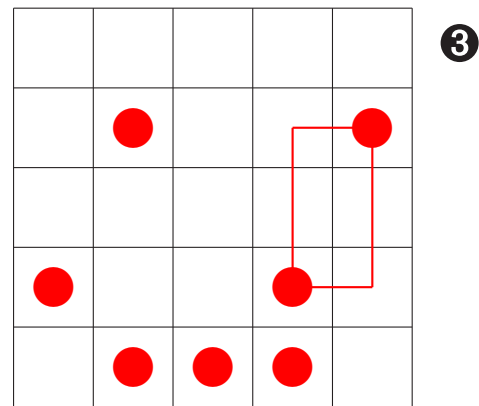
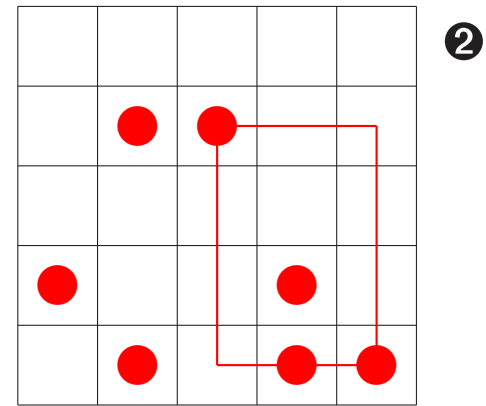
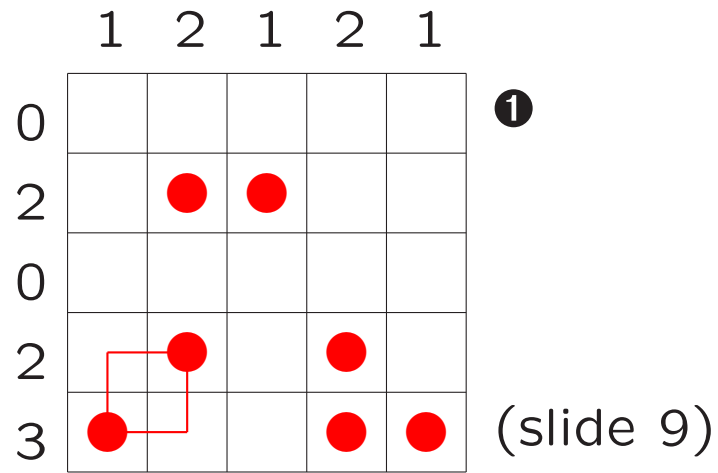
Here the s'_j are the (non-increasing) sorted s_j , and the \bar{s}_j are the column sums of *the* matrix with the r_i as row sums, and ones in the leftmost positions.

$$\begin{array}{c} 10 \\ 01 \end{array}$$

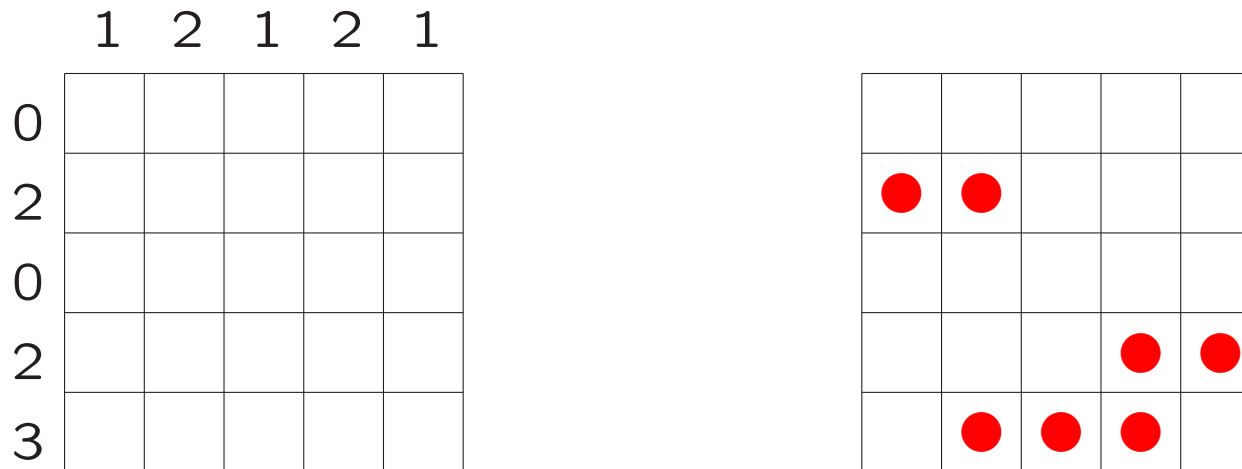
Furthermore, all solutions can be obtained from one another by a series of switchings with **switching components**.

Ryser's algorithm constructs a solution working backward from the last column. The column sums are already sorted in non-increasing order ($s = s'$ and \bar{s}):





In an *h*-convex object all rows must consist of consecutive black pixels: the rows have the “Nonogram property” .



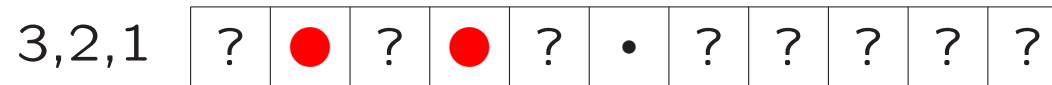
For *h*-convex objects the 2 projections Reconstruction problem is NP-complete . . .

Now back to Nonograms: how to solve them?

Most humans use **logic rules**, combined with **heuristics** like “interchange row reasoning and column reasoning”.

An example of a logic rule is: “if the number 3 is next to a row/column of width 5, the middle pixel must be red”. In this particular rule one looks at one row or column at a time.

Suppose you already know:

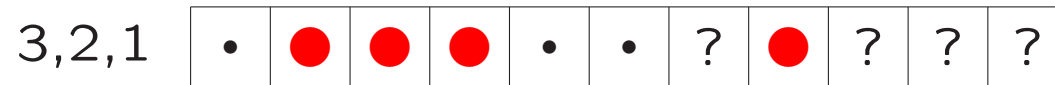


A • means a known white/empty pixel, a ● denotes a known filled pixel. The rest is still unknown.

Remember that we enumerate the lengths of consecutive series of **red** = black pixels — in order.

What can we conclude?

We conclude that for this row:



A • means a known white/empty pixel, a ● denotes a known filled pixel. The rest is still unknown.

So by examining a single row or column we can make progression.

How can a computer program draw such conclusions?

A first option is to use **brute-force**: try “all” possibilities. But a 5×5 Nonogram has

$$2^{25} = 2^{10} \cdot 2^{10} \cdot 2^5 = 1024 \cdot 1024 \cdot 32 \approx 32 \text{ million}$$

possible solutions! And the “ 80×50 Einstein” has $2^{4000} \approx 10^{1200}$ possibilities.

So ... no way! (But for small parts it might work.) We therefore first try some logic reasoning for a single **line** = row or column.

For a single line one can use **Dynamic Programming**.

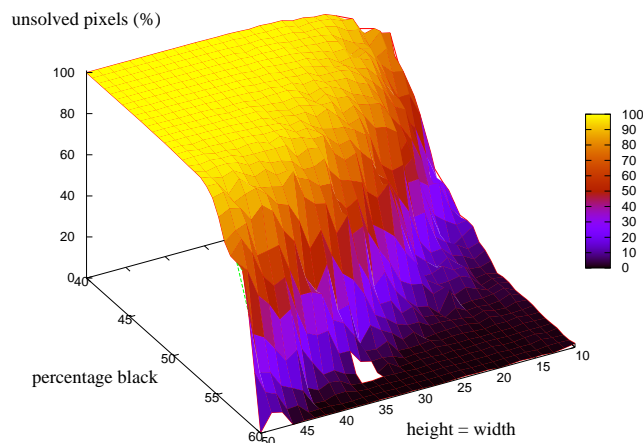
We want a string $s_1s_2\dots s_\ell$ over the alphabet $\Sigma \cup \{?\}$ to match a regular expression $d_1d_2\dots = \sigma_1\{a_1, b_1\}\sigma_2\{a_2, b_2\}\dots$ (so first between a_1 and b_1 times the character σ_1, \dots) in the following sense: $Fix(i, j)$ is true if and only if the prefix $s_1s_2\dots s_i$ can be made to match $d_1d_2\dots d_j$ by “fixing” ?’s to elements from Σ (e.g., $\Sigma = \{\bullet, \bullet\}$):

$$Fix(i, j) = \bigvee_{p = \max(i - b_j, A_{j-1}, L_i^{\sigma_j}(s))}^{\min(i - a_j, B_{j-1})} Fix(p, j - 1)$$

Here $A_j = \sum_{p=1}^j a_p$, $B_j = \sum_{p=1}^j b_p$ and $L_i^\sigma(s)$ is the largest index $h \leq i$ with $s_h \notin \{\sigma, ?\}$ if this exists (and 0 otherwise).

This polynomial time Dynamic Programming approach allows for efficient solving of most puzzles from newspapers. One can repeatedly apply the method to all rows and columns, thereby also introducing a difficulty measure.

See K.J. Batenburg & WAK, Solving Nonograms by combining relaxations, Pattern Recognition, 2009.



percentage unsolved pixels for randomly generated puzzles of different size, black percentage

How far can we get by looking at a single row/column?
Again, with • for a white pixel, and ● for a filled one:

		1		1	
	1	1	1	1	1
0					
1,1					
0					
1,1					
3					

•	•	•	•	•
•	●	•	●	•
•	•	•	•	•
?	?	•	?	?
?	?	●	?	?

But now we are stuck . . . unless we use rows and columns
together.

We have this:

		1		1	
	1	1	1	1	1
0					
1,1					
0					
1,1					
3					

	●		●	
<i>v</i>	<i>w</i>		?	?
<i>u</i>	<i>x</i>	●	?	?

Suppose that $u = \bullet$, then (column) v must be empty, and so (row) $w = \bullet$, and therefore (column) x must be empty. Contradiction (row)! So u must be empty.

The rest is simple.

The logic we used here has rules like “if this pixel is red, that pixel must be white”. This can be modeled through a 2-SAT problem, which happens to be solvable in polynomial time — in contrast with 3-SAT, which is NP-complete.

This adds another difficult measure.

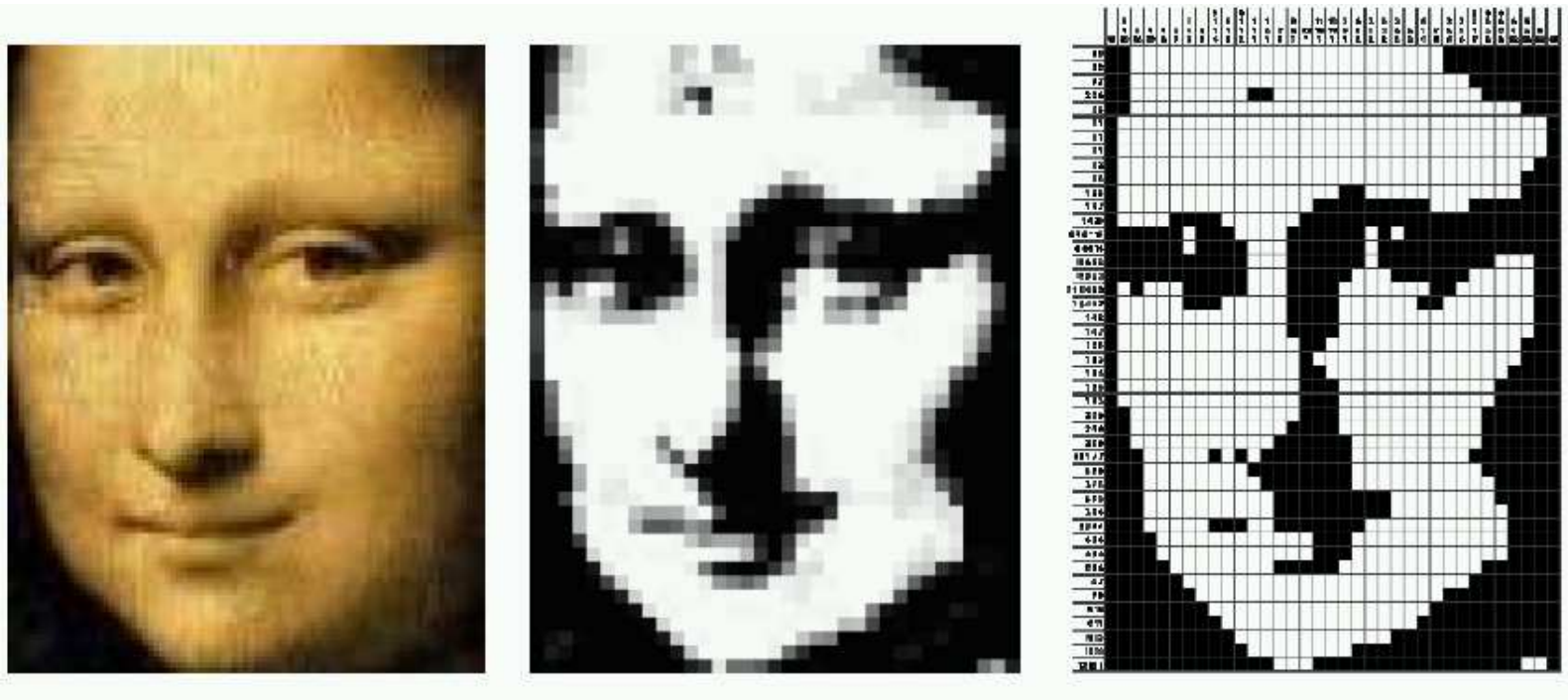
Solving a Nonogram in general is NP-complete.

As an illustration that this 2-SAT logic sometimes fails to catch everything:

			1		
	2	1	1	1	1
1	?		?		?
2	?	?	?	?	?
1	?		?		?
2	?	?	?	?	?
1	?		?		?

Partially solved 5×5 Nonogram, where the fact that pixel ○ must be white is hard to infer.

How to build = design your own Nonogram?



color picture

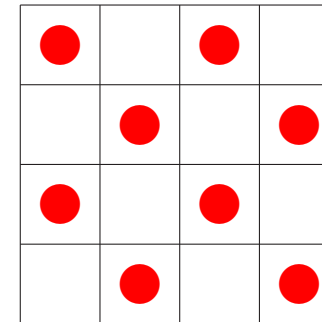
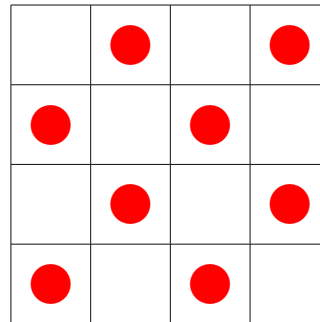
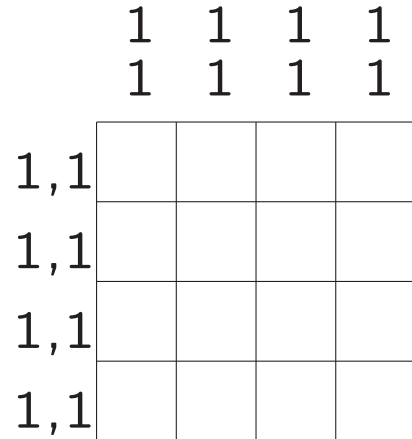
grey value picture

puzzle

<http://www.liacs.nl/home/kosters/nono/>

Remember that a *good* Nonogram should have a **unique** solution.

In general they have many different solutions with switching components!



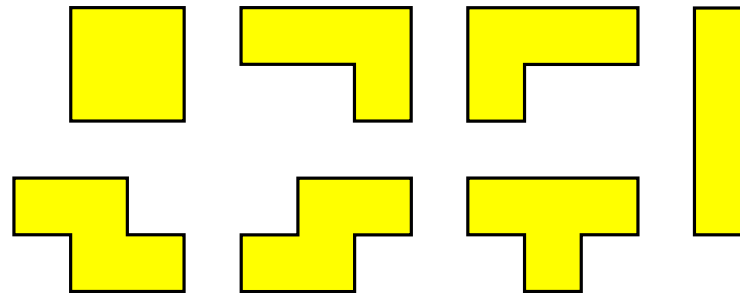
There are several interesting questions attached to a game like **Tetris**:

- How to play well? (AI — Artificial Intelligence)
- How hard is it? (complexity)
- What might happen?

It has been shown that certain Tetris-problems are **NP-complete** (joint work with researchers from MIT & HJH), that you can reach almost all configurations, but that not all problems are “decidable”.

Tetris: NP-complete?

The 7 Tetris-pieces:

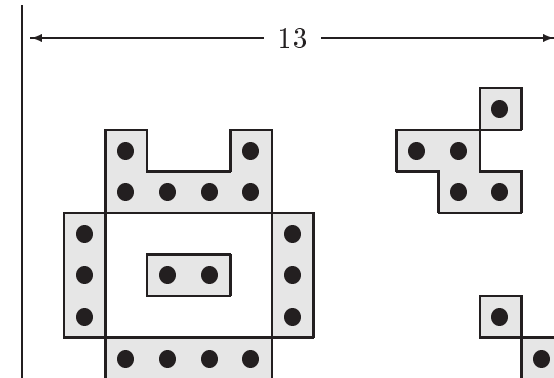


Random pieces fall down, and filled lines are cleared.
The question “Is it possible, given a finite ordered series of these pieces, to clear a partially filled game board?” is NP-complete.

If someone clears the board, this is easy to verify. If clearing is *not* possible however, up till now the only thing one can do to prove this is to check all possibilities, one by one!

Tetris: reachable?

An “arbitrary” configuration:



This figure can be made by dropping 276 suitable Tetris-pieces in the appropriate way, see

<http://www.liacs.nl/home/kosters/tetris/>

Claim: on a game board of odd width every configuration is reachable.