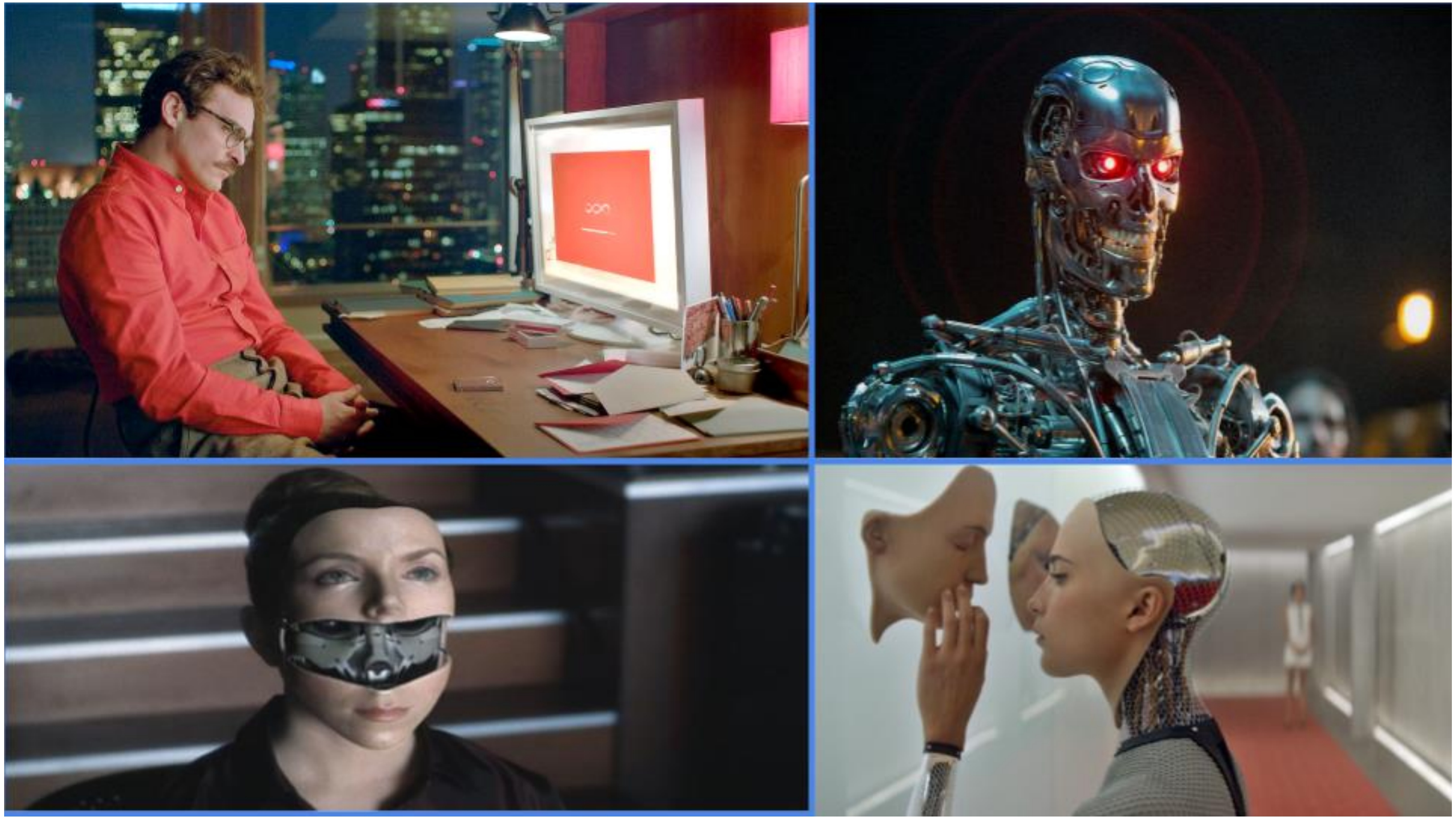
AI lecture 3-2

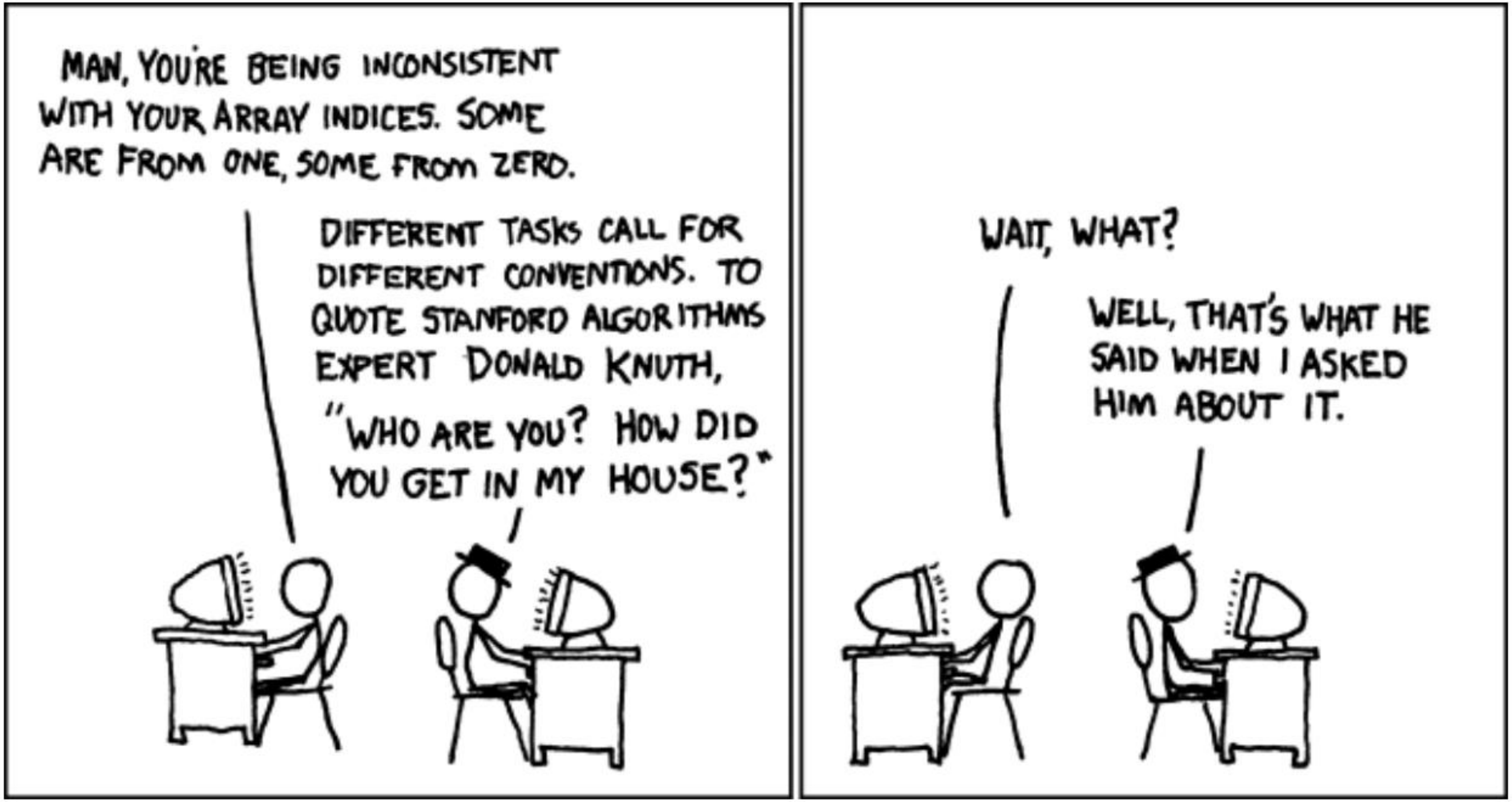
intro AI

solving by searching

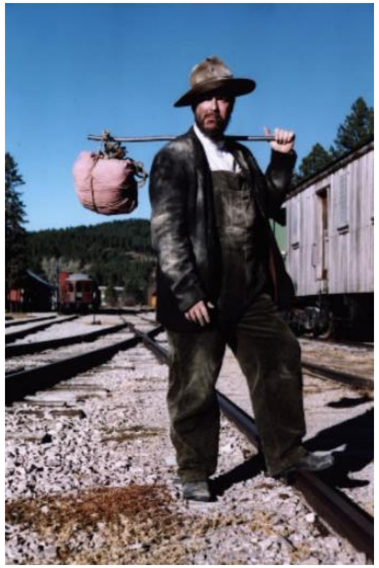


Contents

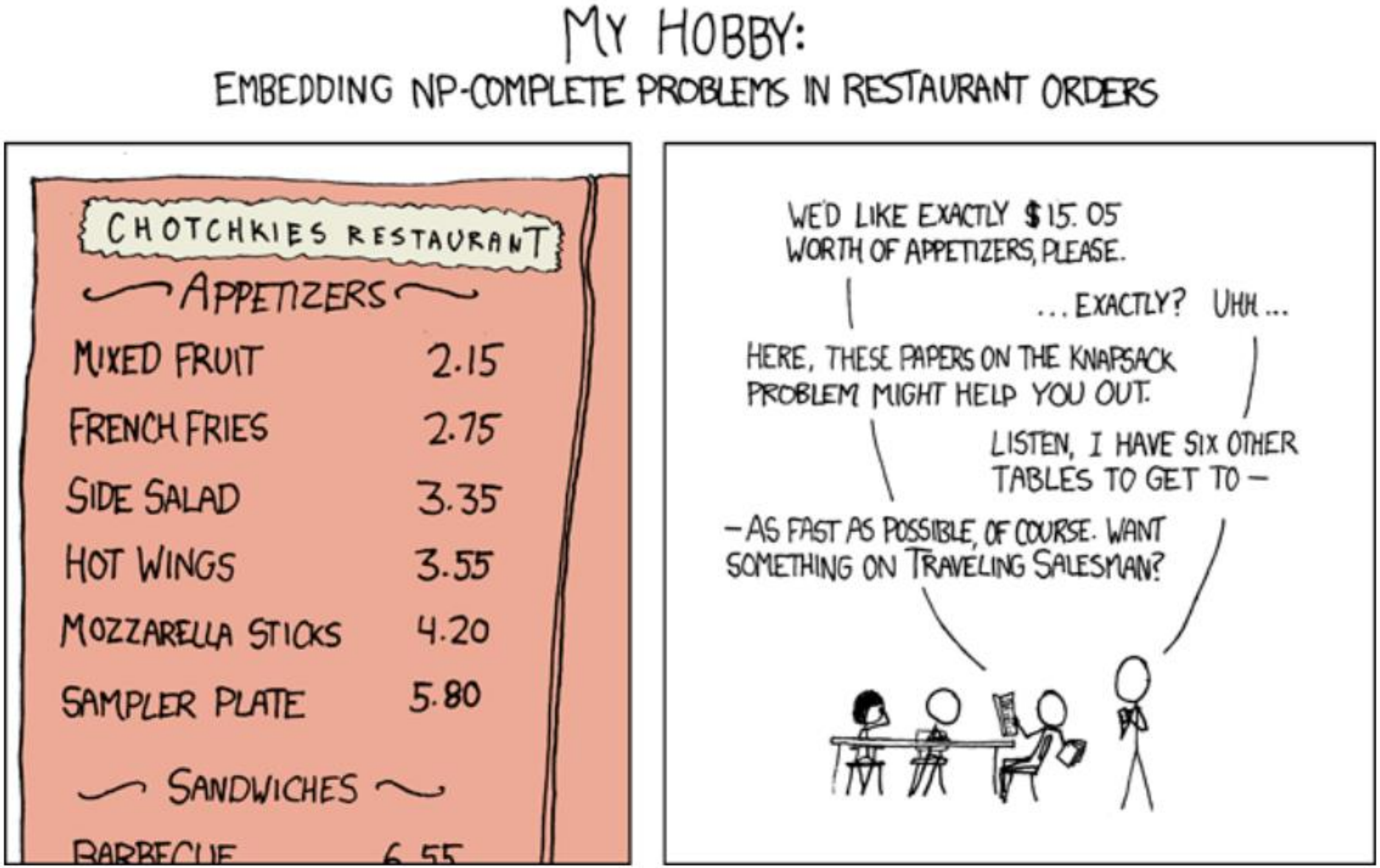
- **Solving Sudoku with Algorithm X**


MAN, YOU'RE BEING INCONSISTENT WITH YOUR ARRAY INDICES. SOME ARE FROM ONE, SOME FROM ZERO.
DIFFERENT TASKS CALL FOR DIFFERENT CONVENTIONS. TO QUOTE STANFORD ALGORITHMS EXPERT DONALD KNUTH,
"WHO ARE YOU? HOW DID YOU GET IN MY HOUSE?"
WAIT, WHAT?
WELL, THAT'S WHAT HE SAID WHEN I ASKED HIM ABOUT IT.

Reducing the problem



- two strategies:
  - remain within the problem-domain and work to a solution
  - reduce the problem to a more **general problem** for which you know there is a strategy (to obtain a solution)
- reducing a problem: problem A is reducible to problem B if an algorithm for solving problem B could also be used to solve problem A
  - two problems are isomorph (have the same form) if their (abstract) structure is similar
  - all NP-complete problems are isomorph: if you could solve one in P-time, you can solve them all
  - example: Knapsack was shown to be NP-complete by reducing Exact Cover to Knapsack
  - see wiki ‘List of NP complete problems’


MY HOBBY:
EMBEDDING NP-COMPLETE PROBLEMS IN RESTAURANT ORDERS
CHOTCHKIES RESTAURANT
APPETIZERS
MIXED FRUIT 2.15
FRENCH FRIES 2.75
SIDE SALAD 3.35
HOT WINGS 3.55
MOZZARELLA STICKS 4.20
SAMPLER PLATE 5.80
SANDWICHES
WE'D LIKE EXACTLY $15.05 WORTH OF APPETIZERS, PLEASE.
...EXACTLY? UHH...
HERE, THESE PAPERS ON THE KNAPSACK PROBLEM MIGHT HELP YOU OUT.
LISTEN, I HAVE SIX OTHER TABLES TO GET TO –
– AS FAST AS POSSIBLE, OF COURSE. WANT SOMETHING ON TRAVELING SALESMAN?

Exact cover is like a partition

Exact cover problem

- set S = {1, 2, 3, 4, 5}
- a collection of subsets of set S
  - A = {1, 4, 5}
  - B = {1, 4}
  - C = {1, 5}
  - D = {3, 5}
  - E = {2}
- is there a selection of subsets such that every element in S exists in ***exactly one*** of these selected subsets?

Represented as a (binary) matrix

	1	2	3	4	5
A		1		1	1
B	1			1	
C	1				1
D			1		1
E		1			

{B, D, E} is an exact cover

Knuth's Algorithm X

‘cover’ means: delete from matrix A

‘overlap’ means: share a 1 in the same column

repeat:

1. select a col c (having *lowest* number of 1’s)
2. select a row r having a 1 in col c
    1. cover row r and include row r in the partial solution
    2. cover all rows that overlap with row r
    3. cover all cols that have a 1 in row r
3. if matrix *A* has no cols left, a solution is found (an exact cover)
4. backtrack if matrix A has col c without a 1

Algorithm X example

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

Algorithm X example

- select a col c (having lowest number of 1’s)

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

Algorithm X example

- select a col c (having lowest number of 1’s)
- select a row r having a 1 in col c
  - cover row r and include row r in the partial solution

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

Algorithm X example

- cover all rows that overlap with row r

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

Algorithm X example

- cover all cols that have a 1 in row r

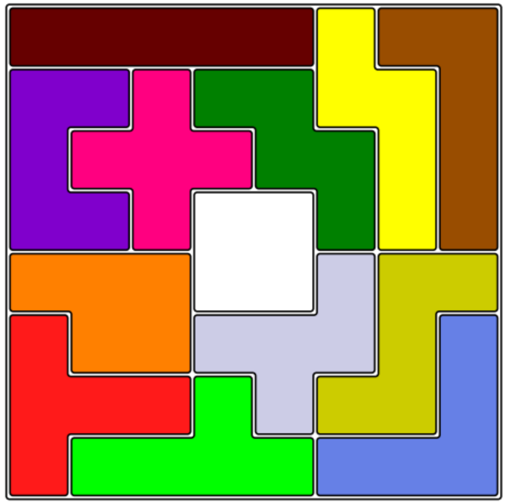
0	0	1	1	0	0
1	1	0	0	0	0
0	1	0	1	0	0
0	0	1	0	0	1
1	0	0	0	0	0
0	0	0	1	1	0

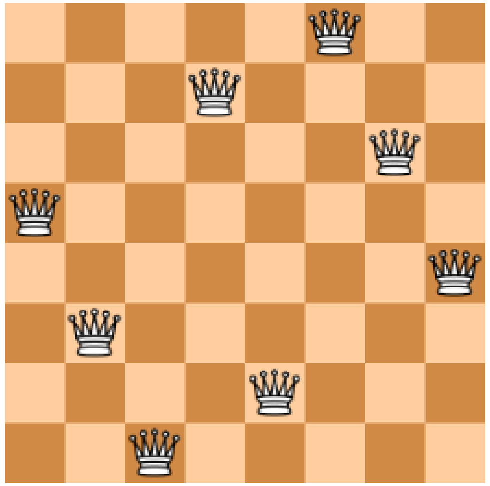
- continue with reduced matrix
- see a detailed example: [link]

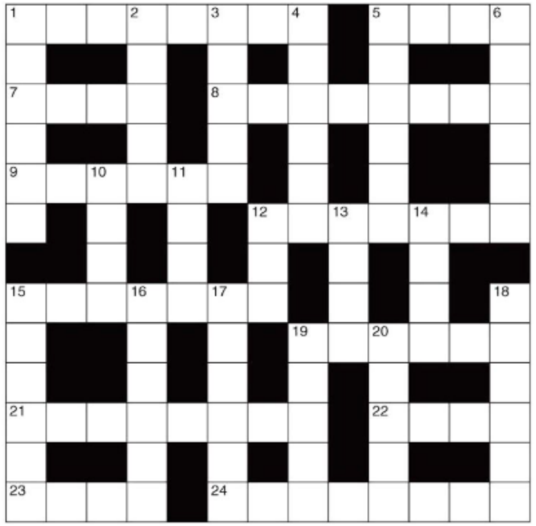
Complexity algorithm X

- let n = number of rows = number of columns
- space $O(n^2)$ (if we do it in-place)
- how many pairwise comparisons?
  - $n(n-1)/2$
  - so for 1000 rows about 500.000
- time $O(n^2)$ comparisons, each takes $O(n)$, so $O(n^3)$
- in practice, matrix is sparse, so time complexity is *much* better

Many CSP's can by solved using Alg-X/DL



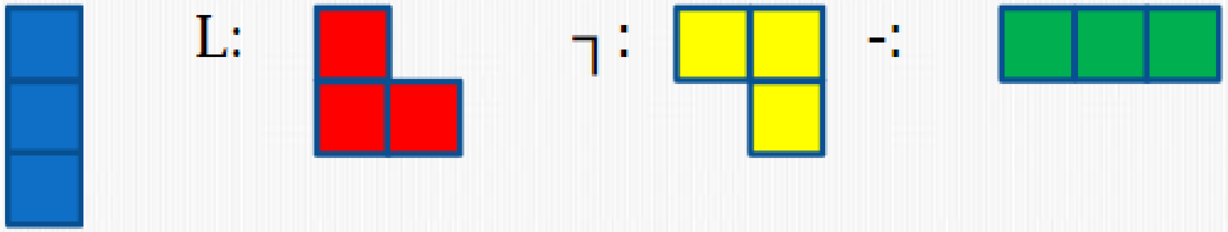




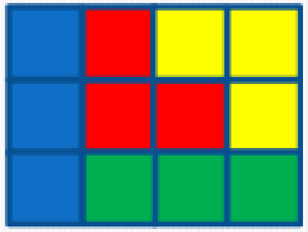
Translating puzzle to matrix

- Let's do a simpler puzzle
  - 4 triominoes

    I: L: ┐: -:

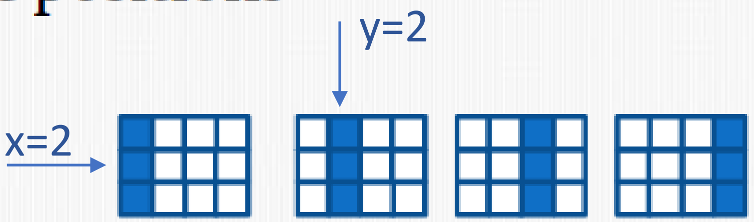


  - No rotation or flip
  - Rectangle 3 x 4
  - Possible solution:



Translating puzzle to exact cover problem

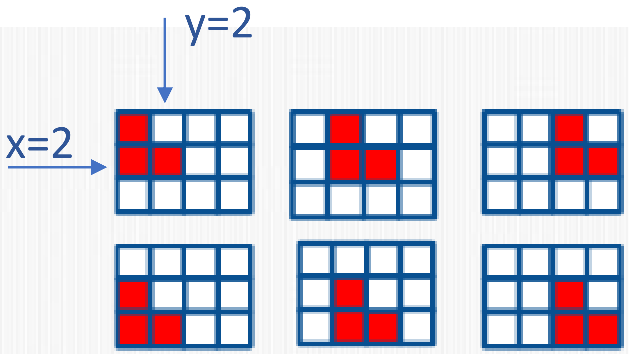
- 16 columns:
  - 4 columns for the pieces
  - 12 columns for the positions
- Rows:
  - 4 I placements



I	L	˥	-	(1,1)	(1,2)	(1,3)	(1,4)	(2,1)	(2,2)	(2,3)	(2,4)	(3,1)	(3,2)	(3,3)	(3,4)
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0
1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1

Translating puzzle to exact cover problem

- Rows
  - 6 L placements

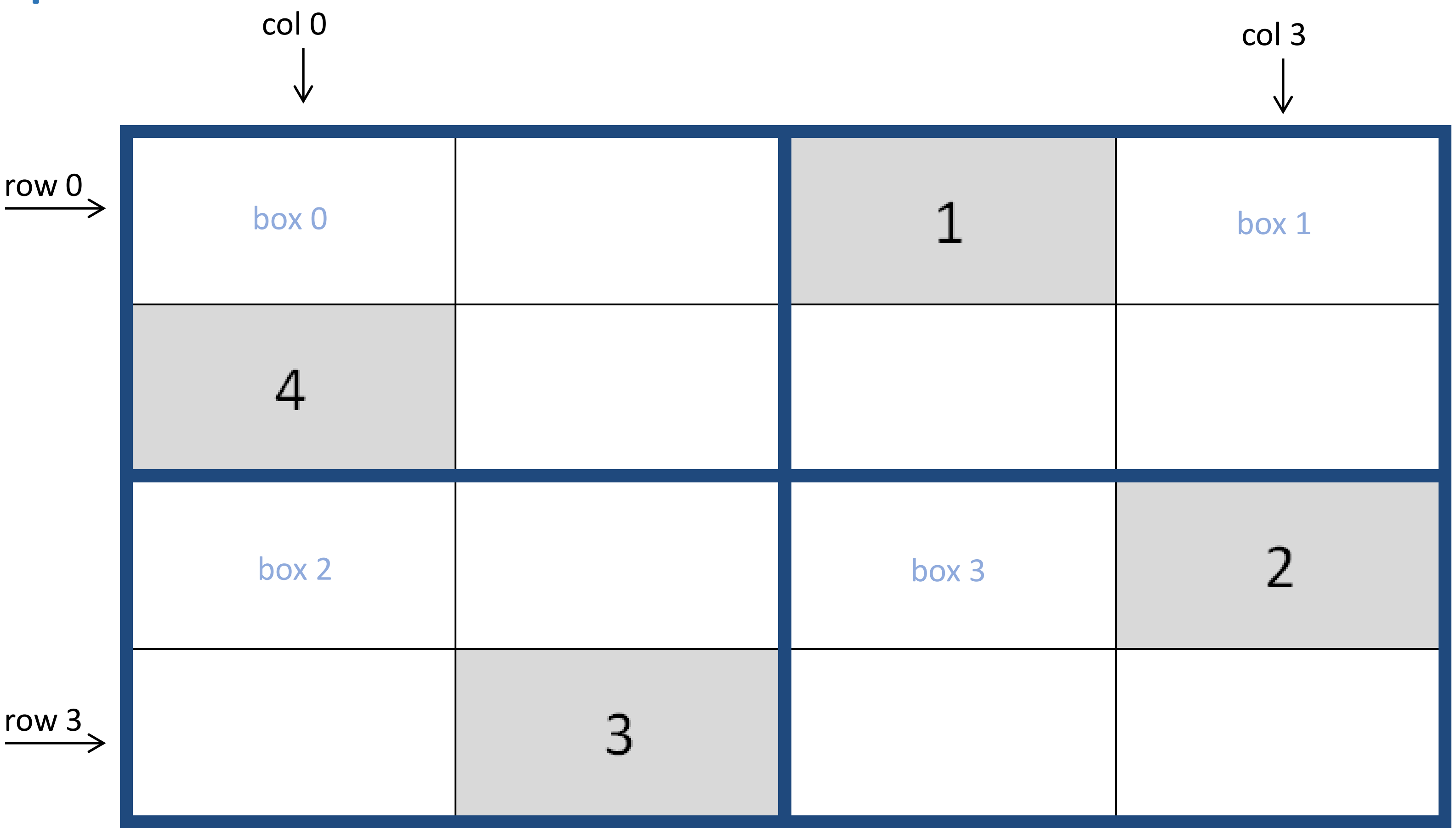


I	L	┐	-	(1,1)	(1,2)	(1,3)	(1,4)	(2,1)	(2,2)	(2,3)	(2,4)	(3,1)	(3,2)	(3,3)	(3,4)
0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	1	0	0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0
0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	0
0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1

Reduce Sudoku

- can we reduce Sudoku to an exact cover problem?
  - exact cover: is there a selection of subsets such that every element in S exists in *exactly one* of the selected subsets?
- idea:
  - let rows represent possible cell-assignments
  - let columns represent the four constraints
  - solution = selection of rows that form exact cover
- Sudoku constrains:
  - every cell has exact one value
  - in every box: a value may only appear once
  - in every row: a value may only appear once
  - in every column: a value may only appear once

Example: 4 x 4 Sudoku



Rows are assignments

- with 4x4 there are 16 cells
- 4 numbers gives 16*4 = 64 possible cell-assignments
- 64 cell assignments are represented by 64 rows

Columns are constraints

- 64 constraints are represented by 64 columns:
  - every cell has exact one value (16 constraints)
  - in every box: a value may only appear once (16 constraints)
  - in every row: a value may only appear once (16 constraints)
  - in every column: a value may only appear once (16 constraints)

- R3C4: a '1' means <u>unique cell </u>number is assigned to cell (3,4)
- R3#7: a '1' means <u>row</u> 3 has value 7 assigned
- C3#7: a '1' means <u>col</u> 3 has value 7 assigned
- B3#7: a '1' means <u>box</u> 3 has value 7 assigned

The matrix

		cells																rows																columns																boxes																
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	
		R0C0	R0C1	R0C2	R0C3	R1C0	R1C1	R1C2	R1C3	R2C0	R2C1	R2C2	R2C3	R3C0	R3C1	R3C2	R3C3	R0#1	R0#2	R0#3	R0#4	R1#1	R1#2	R1#3	R1#4	R2#1	R2#2	R2#3	R2#4	R3#1	R3#2	R3#3	R3#4	C0#1	C0#2	C0#3	C0#4	C1#1	C1#2	C1#3	C1#4	C2#1	C2#2	C2#3	C2#4	C3#1	C3#2	C3#3	C3#4	B0#1	B0#2	B0#3	B0#4	B1#1	B1#2	B1#3	B1#4	B2#1	B2#2	B2#3	B2#4	B3#1	B3#2	B3#3	B3#4	
0	R0C0#1	1																1																1																1																4
1	R0C0#2	1																	1																1																1															4
2	R0C0#3	1																		1																1																1														4
3	R0C0#4	1																			1																1																1													4
4	R0C1#1		1															1																				1												1																4
5	R0C1#2		1																1																				1												1															4
6	R0C1#3		1																	1																				1												1														4
7	R0C1#4		1																		1																				1												1													4
8	R0C2#1			1														1																								1												1												4
9	R0C2#2			1															1																								1												1											4
10	R0C2#3			1																1																								1												1										4
11	R0C2#4			1																	1																								1												1									4
12	R0C3#1				1													1																												1								1												4
13	R0C3#2				1														1																												1								1											4
14	R0C3#3				1															1																												1								1										4
15	R0C3#4				1																1																												1								1									4
16	R1C0#1					1																1												1																1																4
17	R1C0#2					1																	1												1																1															4
18	R1C0#3					1																		1												1																1														4
19	R1C0#4					1																			1												1																1													4
20	R1C1#1						1															1																1												1																4
21	R1C1#2						1																1																1												1															4
22	R1C1#3						1																	1																1												1														4
23	R1C1#4						1																		1																1												1													4
24	R1C2#1							1														1																				1												1												4
25	R1C2#2							1															1																				1												1											4
26	R1C2#3							1																1																				1												1										4
27	R1C2#4							1																	1																				1												1									4
28	R1C3#1								1													1																								1								1												4
29	R1C3#2								1														1																								1								1											4
30	R1C3#3								1															1																								1								1										4
31	R1C3#4								1																1																								1								1									4
32	R2C0#1									1																1								1																								1								4
33	R2C0#2									1																	1								1																								1							4
34	R2C0#3									1																		1								1																								1						4
35	R2C0#4									1																			1								1																								1					4
36	R2C1#1										1															1												1																				1								4
37	R2C1#2										1																1												1																				1							4
38	R2C1#3										1																	1												1																				1						4
39	R2C1#4										1																		1												1																				1					4
40	R2C2#1											1														1																1																				1				4
41	R2C2#2											1															1																1																				1			4
42	R2C2#3											1																1																1																				1		4
43	R2C2#4											1																	1																1																				1	4
44	R2C3#1												1													1																				1																1				4
45	R2C3#2												1														1																				1																1			4
46	R2C3#3												1															1																				1																1		4
47	R2C3#4												1																1																				1																1	4
48	R3C0#1													1																1				1																								1								4
49	R3C0#2													1																	1				1																								1							4
50	R3C0#3													1																		1				1																								1						4
51	R3C0#4													1																			1				1																								1					4
52	R3C1#1														1															1								1																				1								4
53	R3C1#2														1																1								1																				1							4
54	R3C1#3														1																	1								1																				1						4
55	R3C1#4														1																		1								1																				1					4
56	R3C2#1															1														1												1																				1				4
57	R3C2#2															1															1												1																				1			4
58	R3C2#3															1																1												1																				1		4
59	R3C2#4															1																	1												1																				1	4
60	R3C3#1																1													1																1																1				4
61	R3C3#2																1														1																1																1			4
62	R3C3#3																1															1																1																1		4
63	R3C3#4																1																1																1																1	4
		4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	

suppose there is a clue: row1, col1 = 1

Example: 4 x 4 Sudoku

	col 0			col 3
row 0	box 0		1	box 1
	4			
	box 2		box 3	2
row 3		3		

given the clues, cover rows (dark green)
finally: the red cells (rows) form an exact cover

		cells																rows																columns																boxes															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
		R0C0	R0C1	R0C2	R0C3	R1C0	R1C1	R1C2	R1C3	R2C0	R2C1	R2C2	R2C3	R3C0	R3C1	R3C2	R3C3	R0#1	R0#2	R0#3	R0#4	R1#1	R1#2	R1#3	R1#4	R2#1	R2#2	R2#3	R2#4	R3#1	R3#2	R3#3	R3#4	C0#1	C0#2	C0#3	C0#4	C1#1	C1#2	C1#3	C1#4	C2#1	C2#2	C2#3	C2#4	C3#1	C3#2	C3#3	C3#4	B0#1	B0#2	B0#3	B0#4	B1#1	B1#2	B1#3	B1#4	B2#1	B2#2	B2#3	B2#4	B3#1	B3#2	B3#3	B3#4
0	R0C0#1	1																1																1																		1															
1	R0C0#2	1																		1																1																1														
2	R0C0#3	1																		1																1																1													
3	R0C0#4	1																				1																1																1												
4	R0C1#1		1															1																				1												1															
5	R0C1#2		1																1																				1												1														
6	R0C1#3		1																		1																				1												1													
7	R0C1#4		1																				1																				1												1												
8	R0C2#1			1														1																								1												1											
9	R0C2#2			1																1																								1												1										
10	R0C2#3			1																		1																								1												1									
11	R0C2#4			1																				1																								1												1								
12	R0C3#1				1													1																												1												1											
13	R0C3#2				1															1																												1												1										
14	R0C3#3				1																	1																														1												1									
15	R0C3#4				1																			1																																	1											1								
16	R1C0#1					1																1												1																1															
17	R1C0#2					1																		1												1																1														
18	R1C0#3					1																				1												1																1													
19	R1C0#4					1																						1												1																1												
20	R1C1#1						1															1																1												1															
21	R1C1#2						1																	1																1												1														
22	R1C1#3						1																			1																1												1													
23	R1C1#4						1																					1																1												1												
24	R1C2#1							1													1																				1												1											
25	R1C2#2							1															1																				1												1										
26	R1C2#3							1																	1																				1												1									
27	R1C2#4							1																			1																				1												1								
28	R1C3#1								1													1																								1												1											
29	R1C3#2								1															1																								1												1										
30	R1C3#3								1																	1																								1												1									
31	R1C3#4								1																			1																								1												1								
32	R2C0#1									1																1								1																								1							
33	R2C0#2									1																		1								1																								1						
34	R2C0#3									1																				1								1																								1					
35	R2C0#4									1																						1								1																								1				
36	R2C1#1										1															1												1																				1							
37	R2C1#2										1																	1												1																				1						
38	R2C1#3										1																			1												1																				1					
39	R2C1#4										1																					1												1																				1				
40	R2C2#1											1													1																1																				1			
41	R2C2#2											1															1																1																				1		
42	R2C2#3											1																	1																1																				1	
43	R2C2#4											1																			1																1																				1
44	R2C3#1												1													1																				1																				1			
45	R2C3#2												1															1																				1																				1		
46	R2C3#3												1																	1																				1																				1	
47	R2C3#4												1																			1																				1																				1
48	R3C0#1													1																	1				1																								1							
49	R3C0#2													1																			1				1																								1						
50	R3C0#3													1																					1				1																								1					
51	R3C0#4													1																							1				1																								1				
52	R3C1#1														1															1								1																				1							
53	R3C1#2														1																	1								1																				1						
54	R3C1#3														1																			1								1																				1					
55	R3C1#4														1																					1								1																				1				
56	R3C2#1															1													1												1																				1			
57	R3C2#2															1															1												1																				1		
58	R3C2#3															1																	1												1																				1	
59	R3C2#4															1																			1												1																				1
60	R3C3#1																1													1																1																1			
61	R3C3#2																1															1																1																1		
62	R3C3#3																1																	1																1																1	
63	R3C3#4																1																			1																1																1

3	2	1	4
4	1	2	3
1	4	3	2
2	3	4	1

Solving 9 x 9 Sudoku in Python

- what data structures do we need?
  - nr of rows = 9×9×9=729
  - nr of cols = 81+81+81+81=324
  - a read-only matrix (list of lists: rows x cols and the transpose cols x rows)
  - a list row_valid to make a row valid or invalid
  - a list col_valid to make a row valid or invalid

Solving 3 x 3 Sudoku in Python

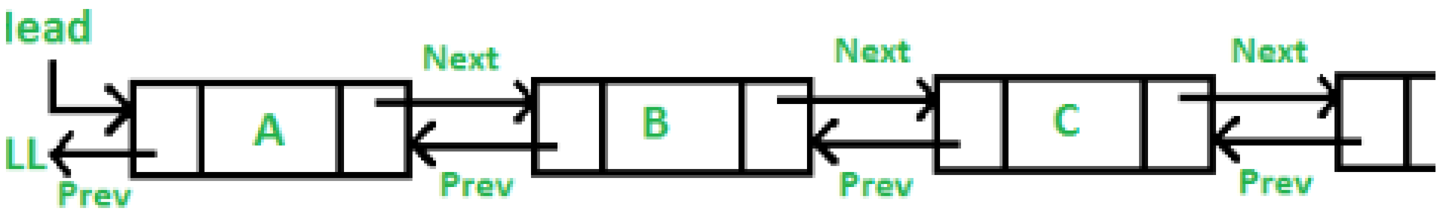
- what functions do we need?
  - a function that finds an exact cover using algorithm X
- a helper function that:
  - initializes the matrix with 0’s and 1’s
  - parses a string to clues (problem is given as a string)
  - puts clues in the matrix
  - displays the solution 9x9
  - given a selected row, makes appropriate cols and rows invalid

Operations that we'll do often

- given col c find all rows having col c = 1
- given row r find all cols having row r = 1
  - note: these operations are symmetrical
- cover a row or col
- uncover a row or col
- find a col with the least number of ones

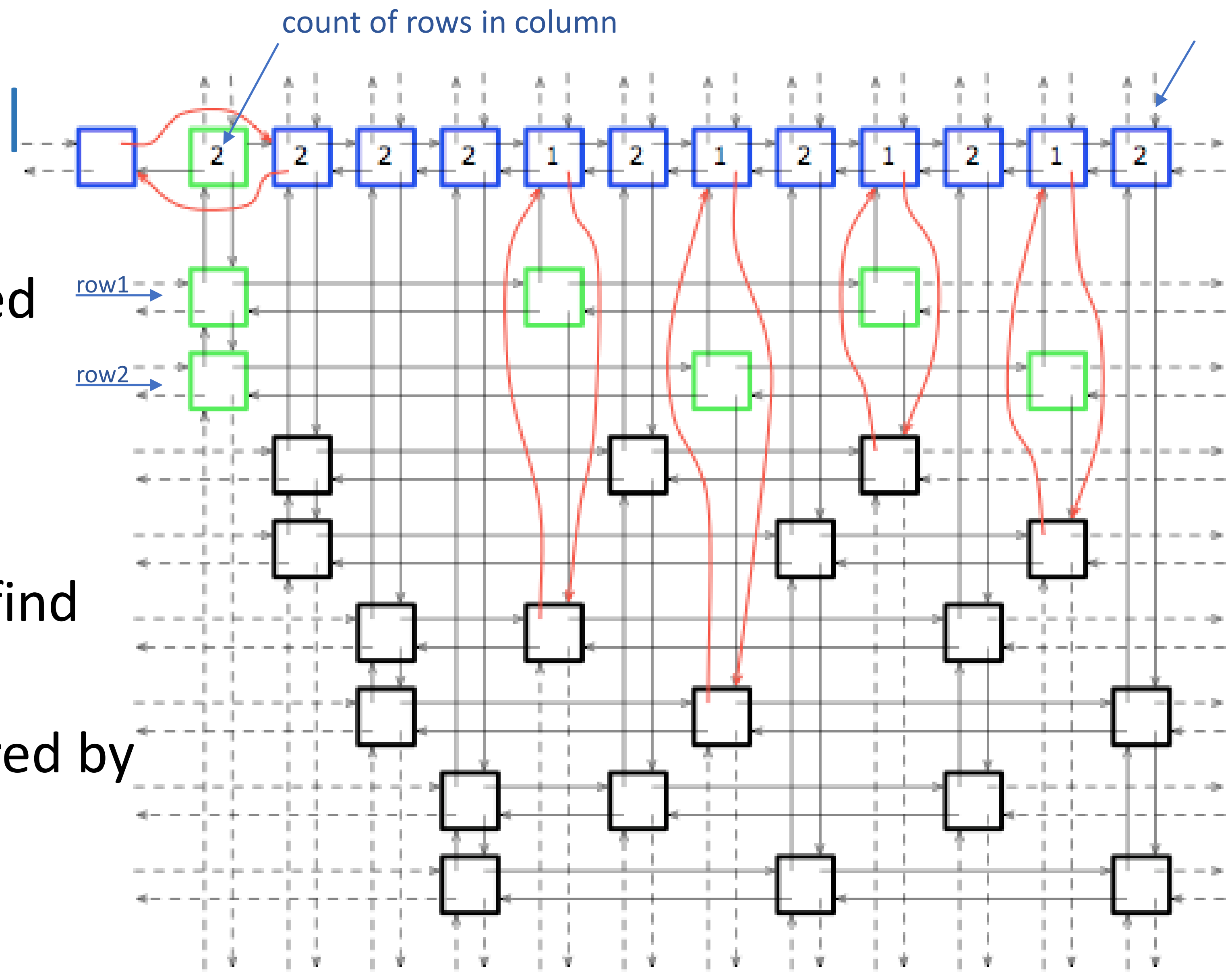
Implementation using 'Dancing Links'

- idea: if rows and cols and are doubly linked lists it's fast to find all 1's in rows and cols
  - each "1" in the matrix is represented by a node of linked list
- note: it's then easy to remove and later restore (undo-delete) a node!
- what makes Dancing Links efficient is that as long as you store a pointer to the column header, you can easily uncover a col
- can be done very efficient in C/C++ or Go



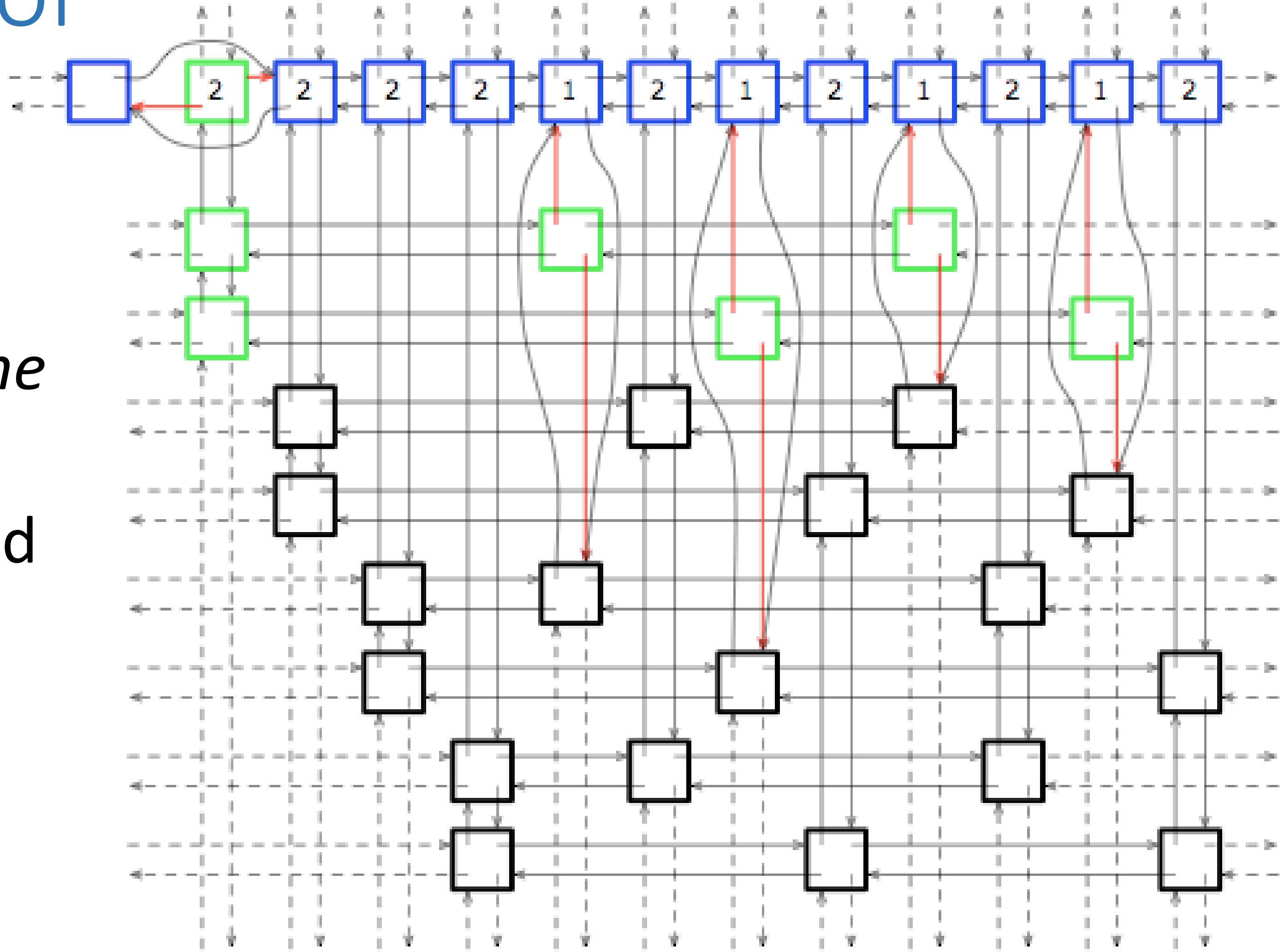
“Covering” a col

- red arrows are modified links
- first col is covered by modifying links
- then move down and find rows 1 and 2
- rows 1 and 2 are covered by traversing right and modifying links



“Un-covering” a col

- un-covering is the reverse operation of covering
- for this we need to *save the old links*
- red arrows are the restored links



Donald Knuth quotes

- Programming is the **art of telling** another human being what one wants the computer to do.
- Programs are meant to be **read by humans** and only incidentally for computers to execute.
- Computer **programming is an art**, because it applies accumulated knowledge to the world, because it requires skill and ingenuity, and especially because it produces objects of beauty. A programmer who subconsciously views himself as an artist will enjoy what he does and will do it better.
- When you write a program, think of it primarily as a work of literature. You're trying to write something that human beings are going to read. Don't think of it primarily as something a computer is going to follow. The more effective you are at **making your program readable**, the more effective it's going to be: You'll understand it today, you'll understand it next week, and your successors who are going to maintain and modify it will understand it.