Fundamentals of Artificial Intelligence

Laboratory

Dr. Mauro Dragoni

Department of Information Engineering and Computer Science

Academic Year 2020/2021

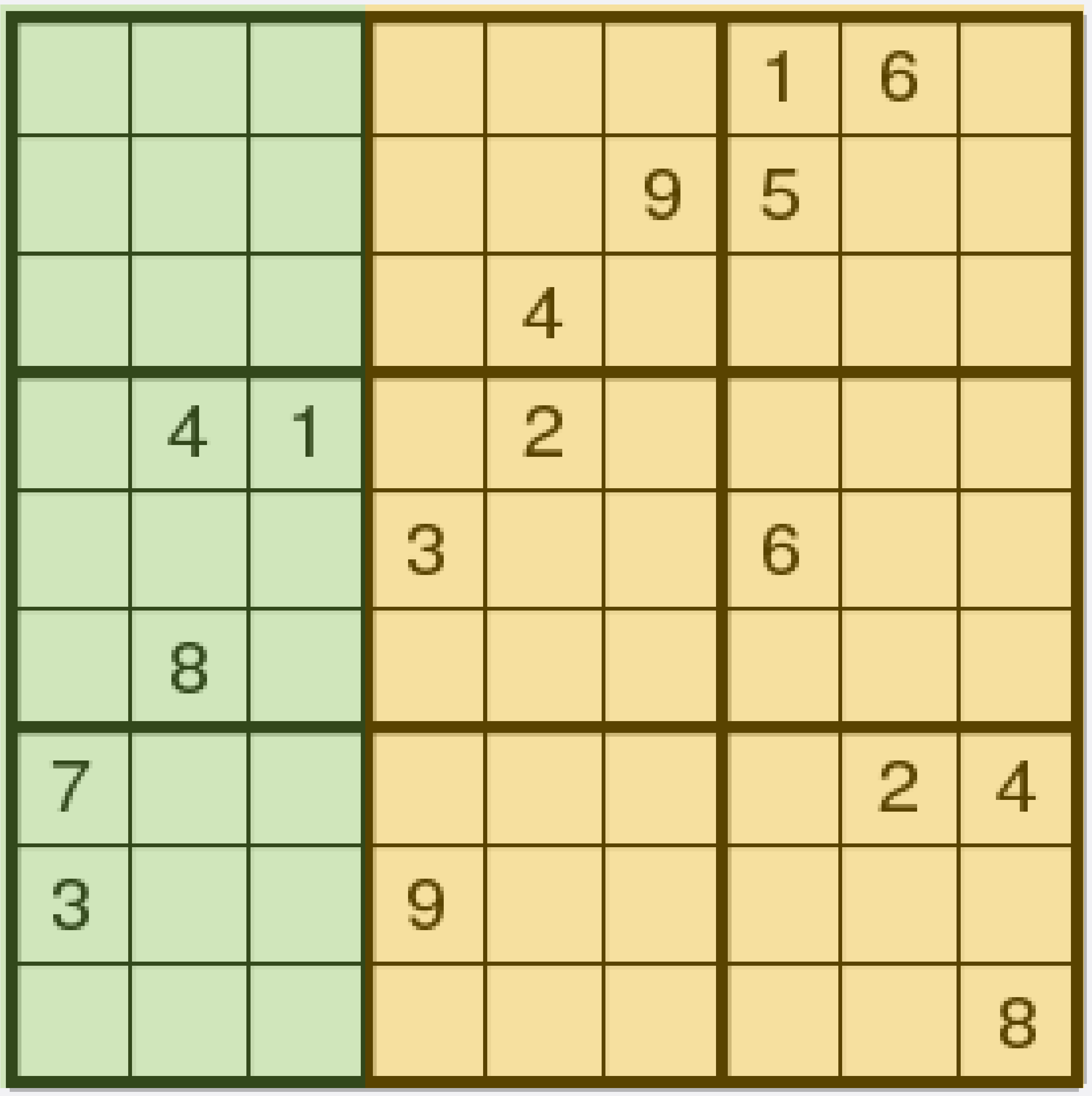
Exercise 3.14

- Design a genetic algorithm for solving a Sudoku puzzle.

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

Exercise 3.14 – some hints

- Crossover



Exercise 3.14 – some hints

- Mutation (sensorless)

{1,2,3,4,5,6,7,8,9}

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

Exercise 3.14 – some hints

- Mutation (partial observation - home square)

{1,2,3,4,5,6,7,8,9}

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

Exercise 3.14 – some hints

- Mutation (partial observation – row + home square)

{1,2,3,4,5,6,7,8,9}

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

Exercise 3.14 – some hints

- Mutation (partial observation – column + home square)

{1,2,3,4,5,6,7,8,9}

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

Exercise 3.14 – some hints

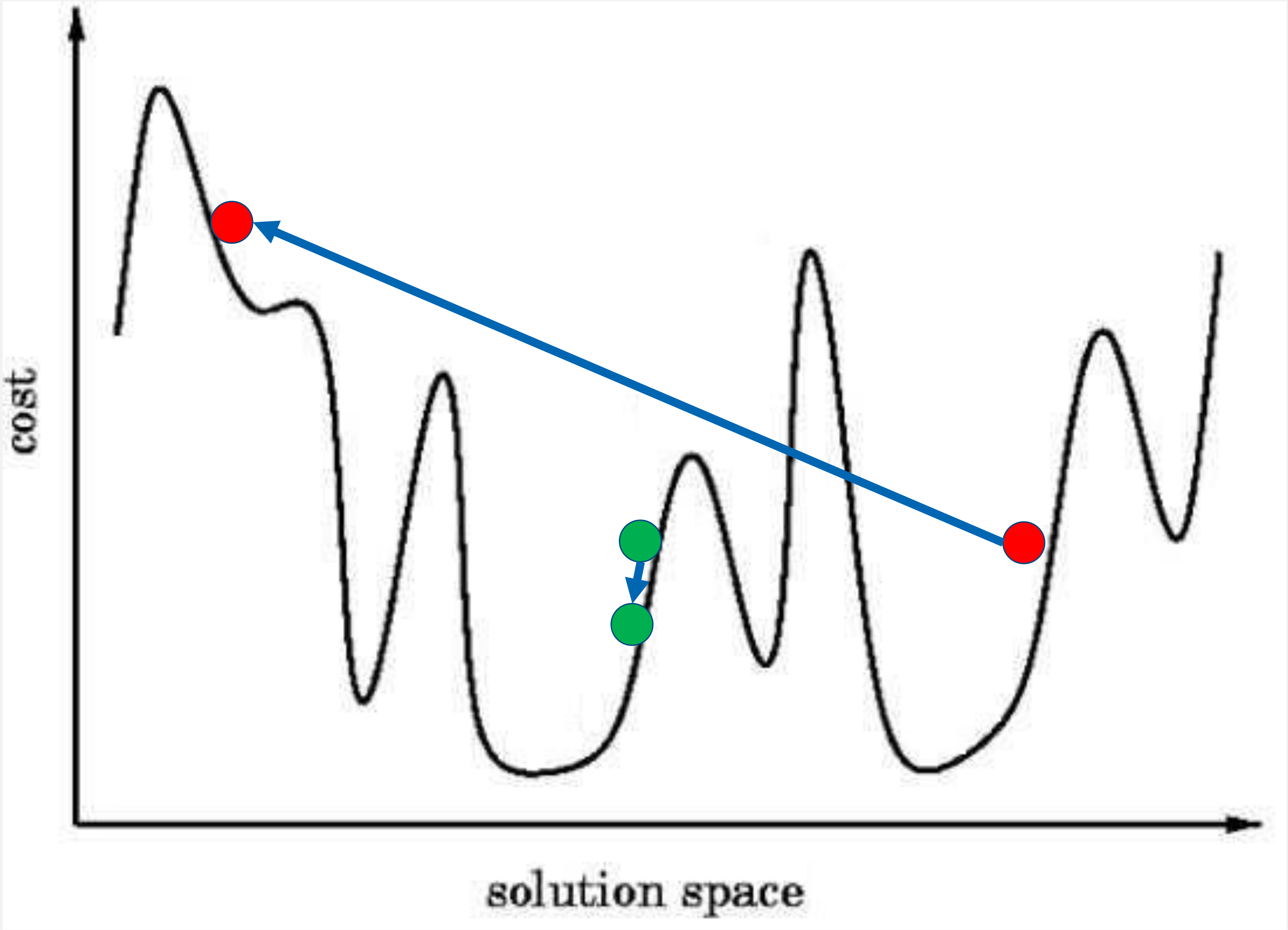
- Mutation (full field)

{1,2,3,4,5,6,7,8,9}

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

Exercise 3.14 – a last thing about GA

- How GAs walk on the solution space…



Before to start…

- A corpus for building (very) intelligent agents

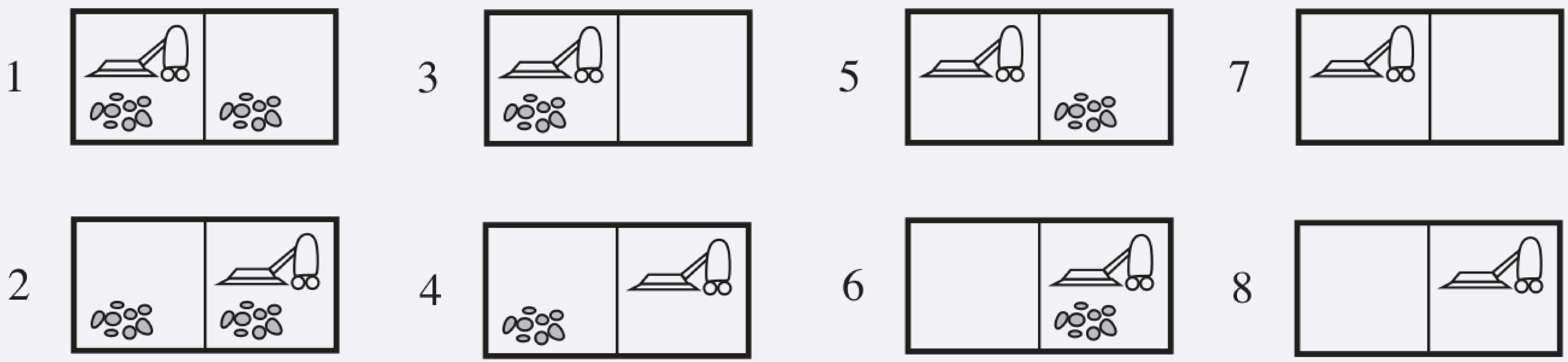
  The Abstraction and Reasoning Corpus (ARC)
  https://github.com/fchollet/ARC

- Reference paper:
  "On The Measure of Intelligence" - https://arxiv.org/abs/1911.01547

Exercise 4.1

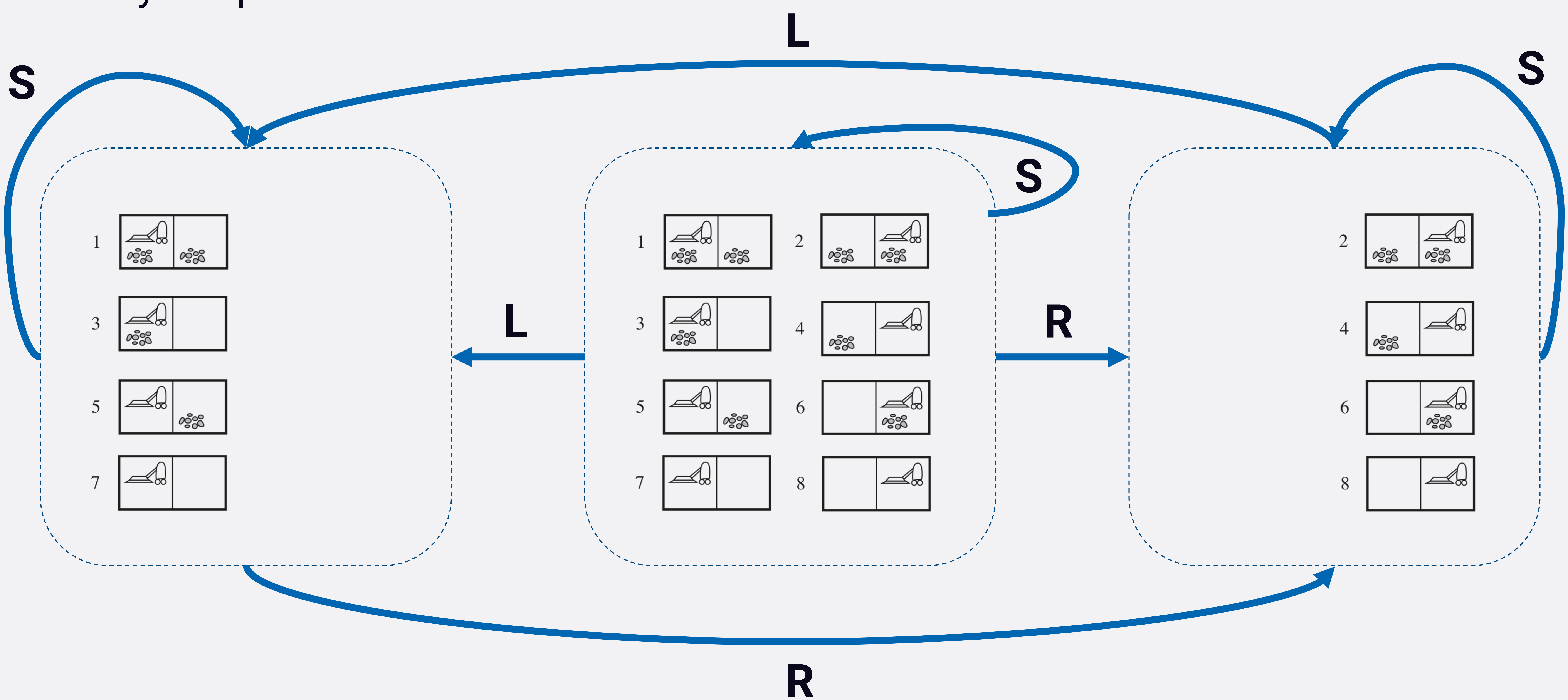
- Consider the sensorless version of the erratic vacuum world. Draw the belief-state space reachable from the initial belief state {1, 2, 3, 4, 5, 6, 7, 8}, and explain why the problem is unsolvable.

- Erratic vacuum world:
  - When applied to a dirty square the action cleans the square and sometimes cleans up dirt in an adjacent square, too.
  - When applied to a clean square the action sometimes deposits dirt on the carpet.



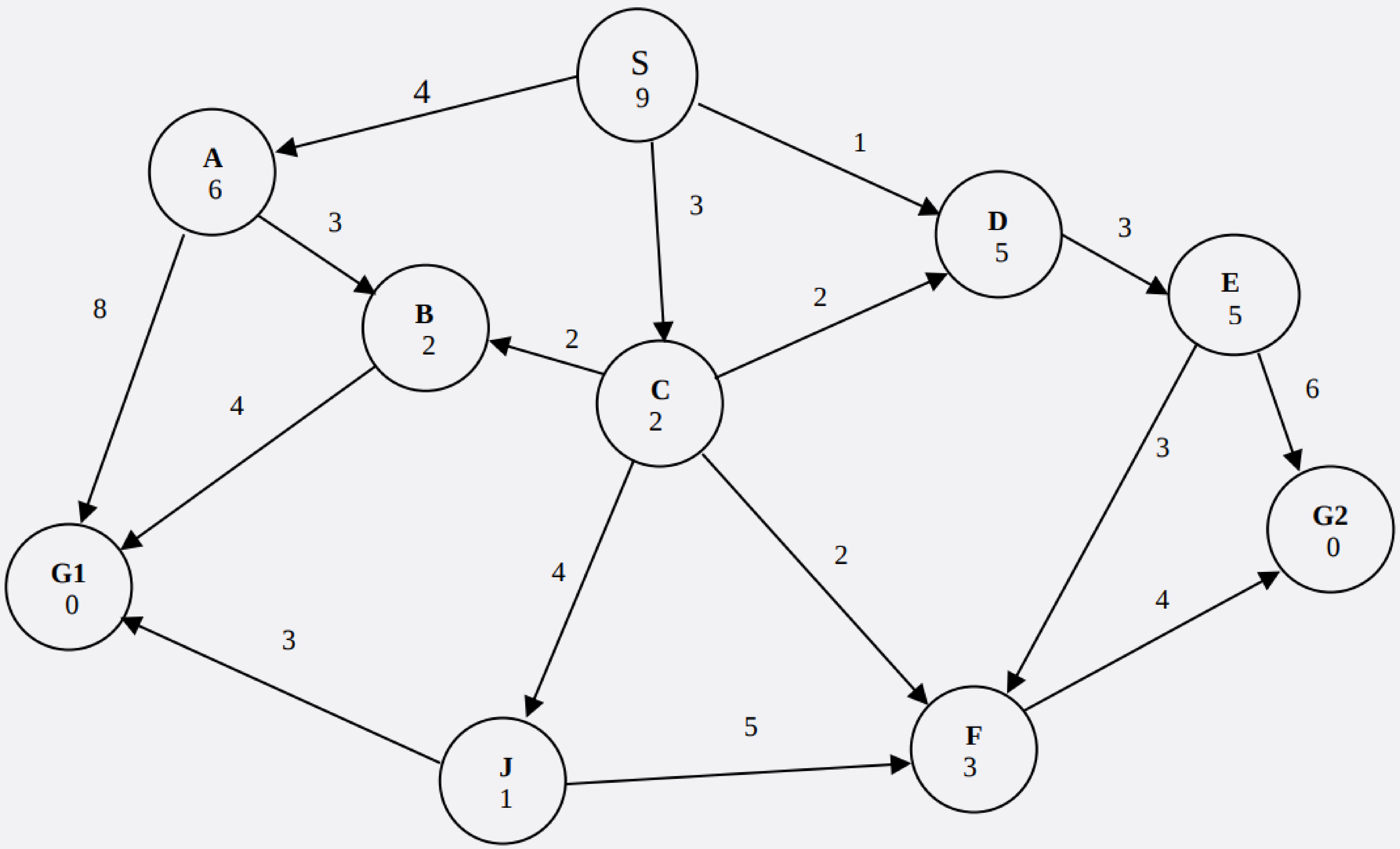
Exercise 4.1 - solution

- Consider the sensorless version of the erratic vacuum world. Draw the belief-state space reachable from the initial belief state {1, 2, 3, 4, 5, 6, 7, 8}, and explain why the problem is unsolvable.



Exercise 4.2

- Consider the search space below, where S is the start node and G1 and G2 satisfy the goal test. Arcs are labeled with the cost of traversing them and the estimated cost to a goal (i.e., the **h** function) is reported inside nodes (so lower scores are better). Apply **Greedy Best-First** and **A*** search strategies; indicate which goal state is reached (if any) and list, in order, all the states. In case of equal values, nodes should be removed from the list in alphabetical order.



Exercise 4.2 – Greedy best-first

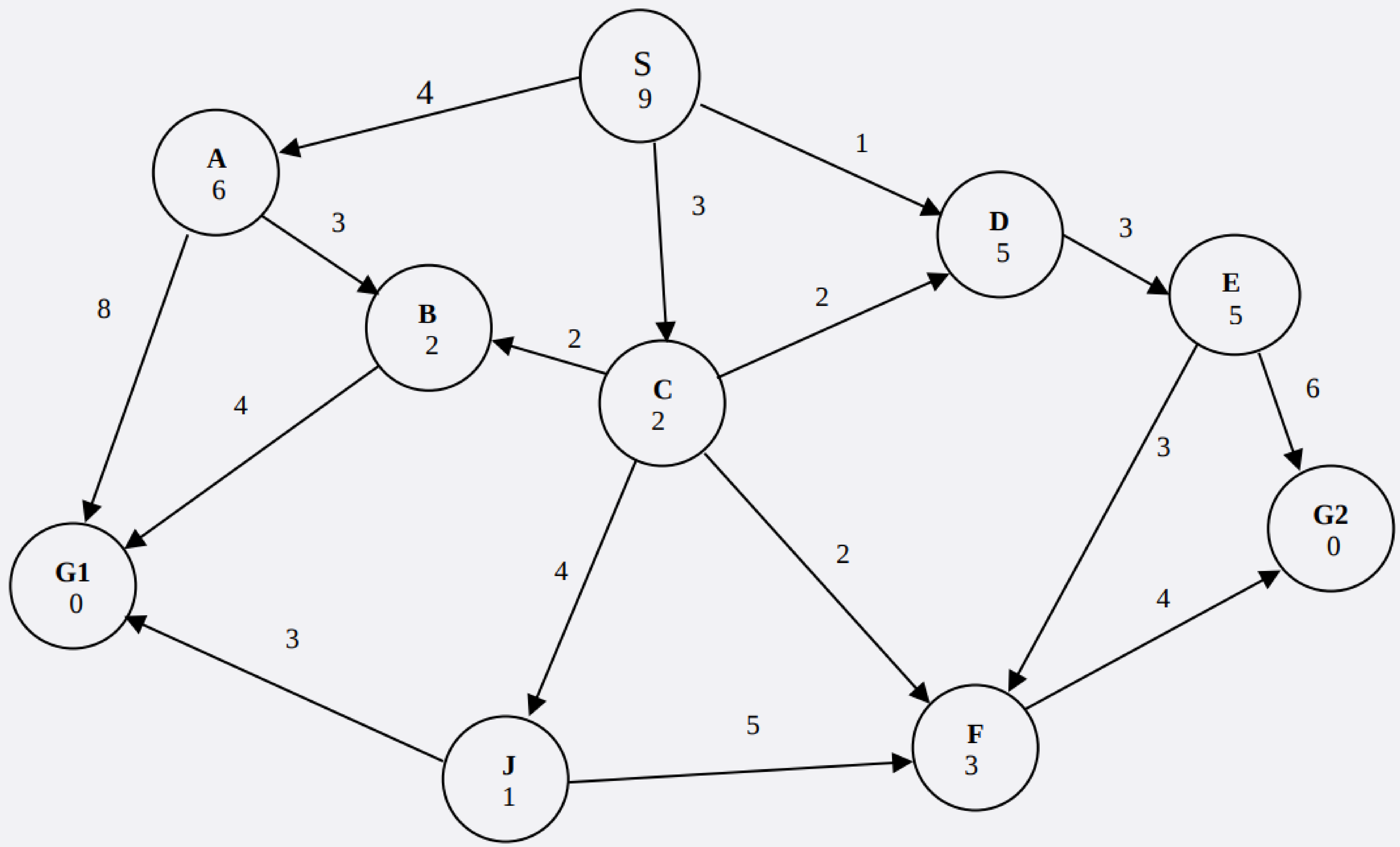
Step	Queue	Processed node	Children
1	S(9)	S(9)	A(6) C(2) D(5)
2	C(2) D(5) A(6)	C(2)	B(2) F(3) J(1)
3	J(1) B(2) F(3) D(5) A(6)	J(1)	F(3) G1(0)
4	G1(0) B(2) F(3) D(5) A(6)	G1(0)	

Exercise 4.2 – A*

Step	Queue	Processed node	Children
1	S(9)	S(9)	A(4+6) C(3+2) D(1+5)
2	C(5) D(6) A(10)	C(5)	B(5+2) F(5+3) D(5+5) J(7+1)
3	D(6) B(7) F(8) J(8) A(10)	D(6)	E(4+5)
4	B(7) F(8) J(8) E(9) A(10)	B(7)	G1(9+0)
5	F(8) J(8) E(9) G1(9) A(10)	F(8)	G2(9+0)
6	J(8) E(9) G1(9) G2(9) A(10)	J(8)	F(12+3) G1(10+0)
7	E(9) G1(9) G2(9) A(10)	E(9)	F(7+3) G2(10+0)
8	G1(9) G2(9) A(10)	G1(9)	

Exercise 4.2

- Is this **h** function admissible?

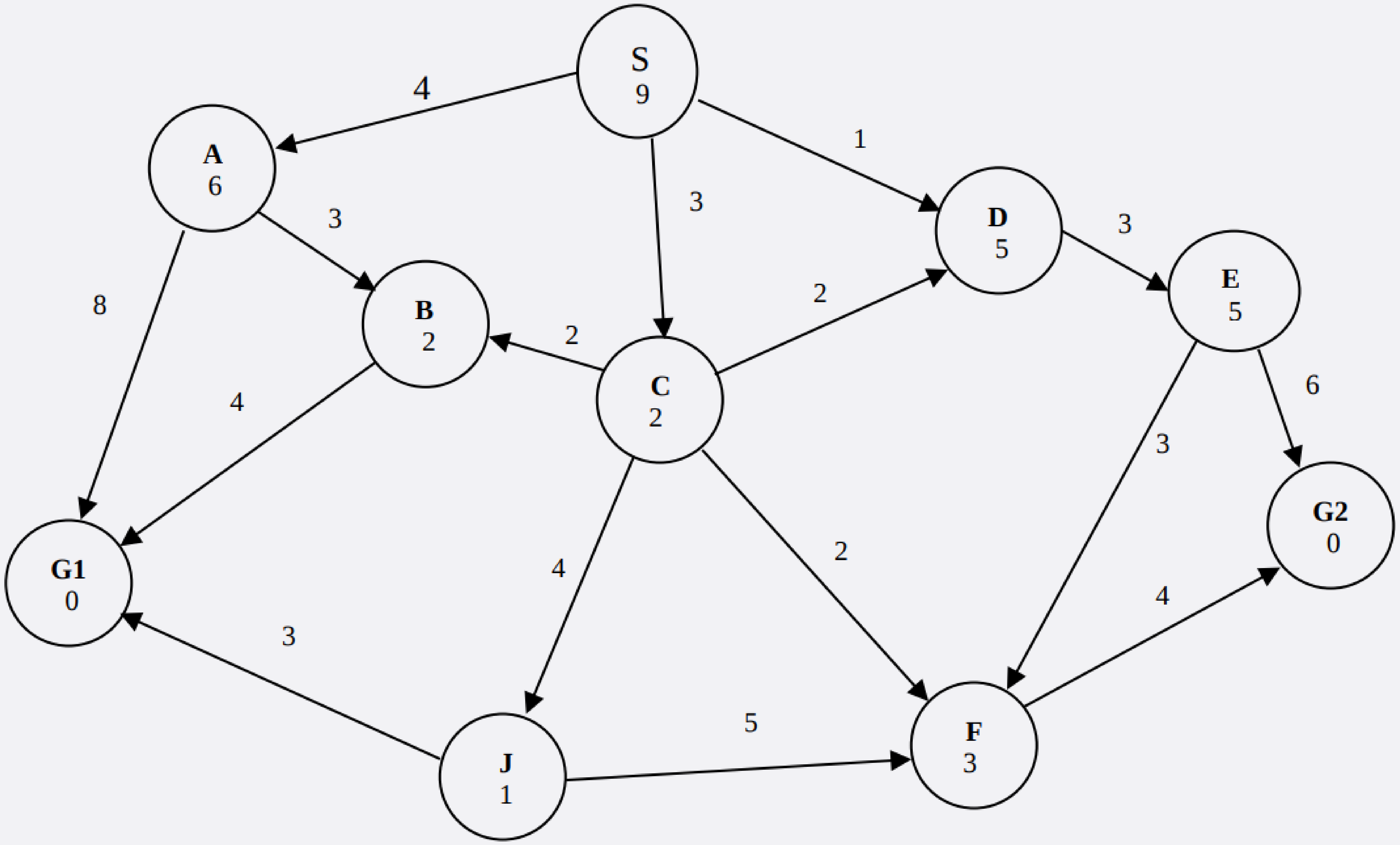


Exercise 4.2

- Is this **h** function admissible?

- This h function is admissible. For all nodes n, **h(n) <= h*(n)**, where h*(n) is the actual cost to reach the goal from node n.

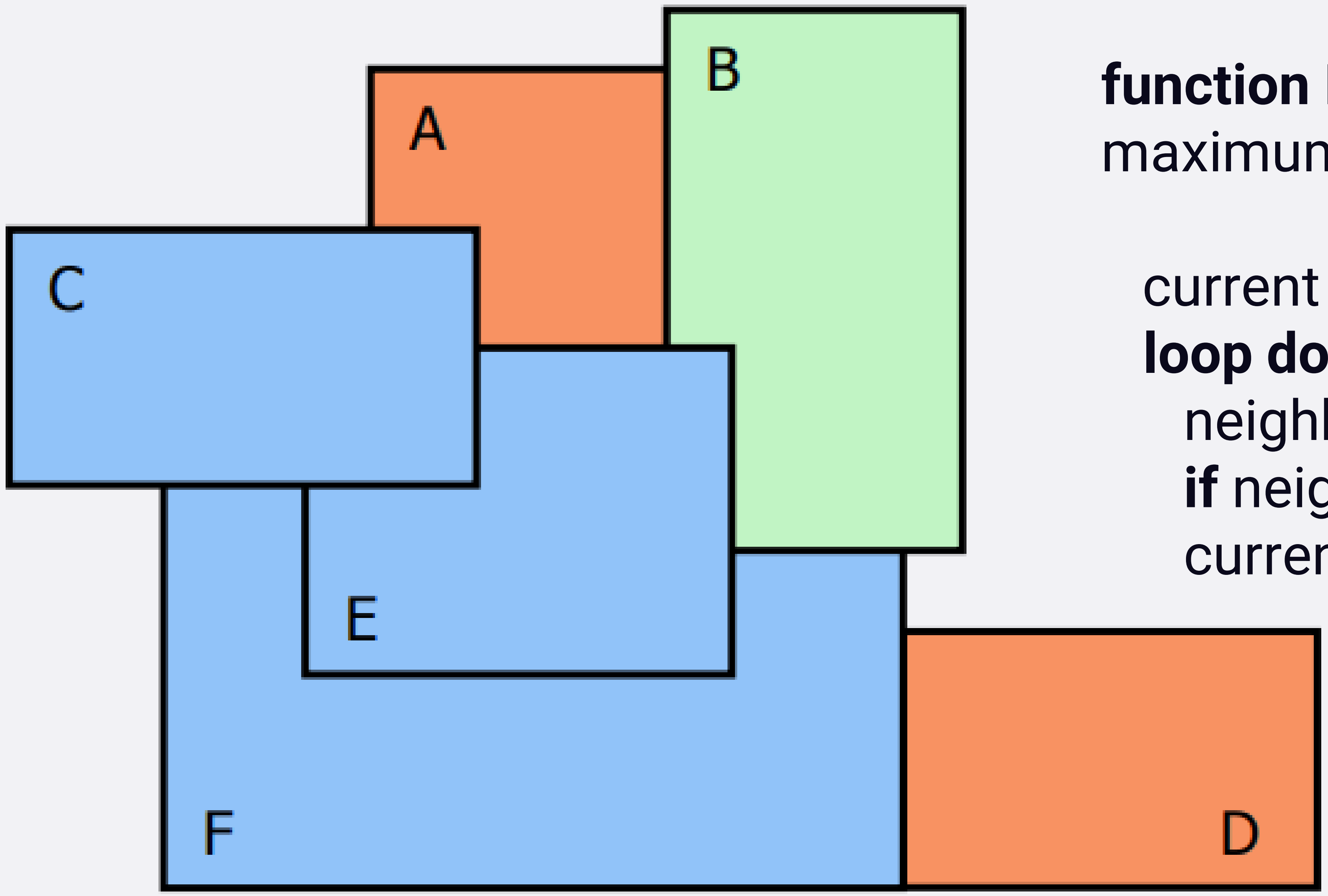
Node	h	h*
S	9	10
A	6	7
B	2	4
C	2	6
D	5	9
E	5	6
F	3	4
J	1	3



Exercise 4.3

- Graph coloring: start with random coloring of nodes and change color of one node to reduce the number of conflicts.

Step	A	B	C	D	E	F	# conflicts
1	r	g	b	r	b	b	3 {CE, CF, EF}

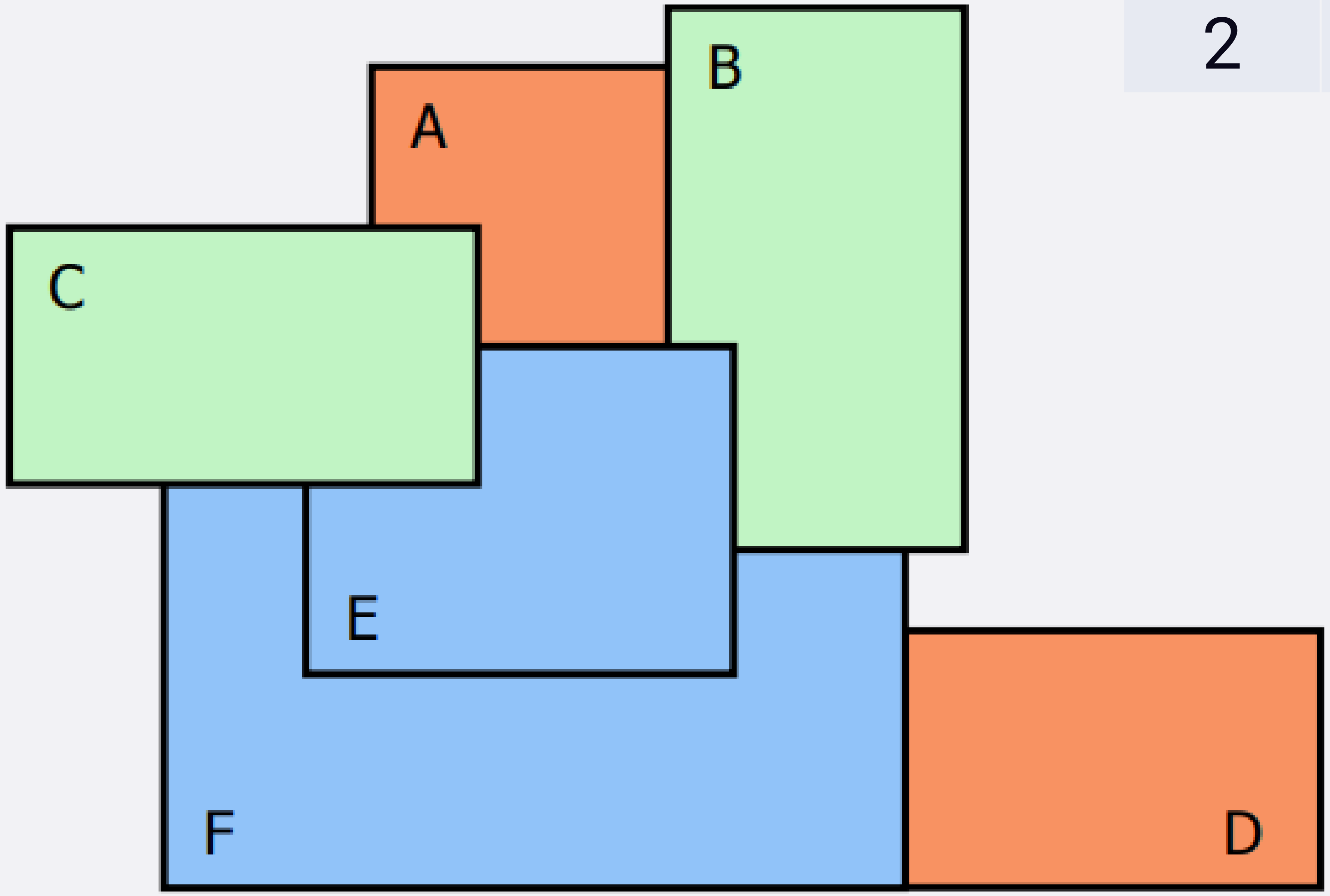


function HILL-CLIMBING (problem) **returns** a state that is a local maximum

current ← MAKE-NODE (problem.INITIAL-STATE)
loop do
neighbor ← a highest-valued successor of current
if neighbor.VALUE ≤ current.VALUE **then return** current.STATE
current ← neighbor

Exercise 4.3

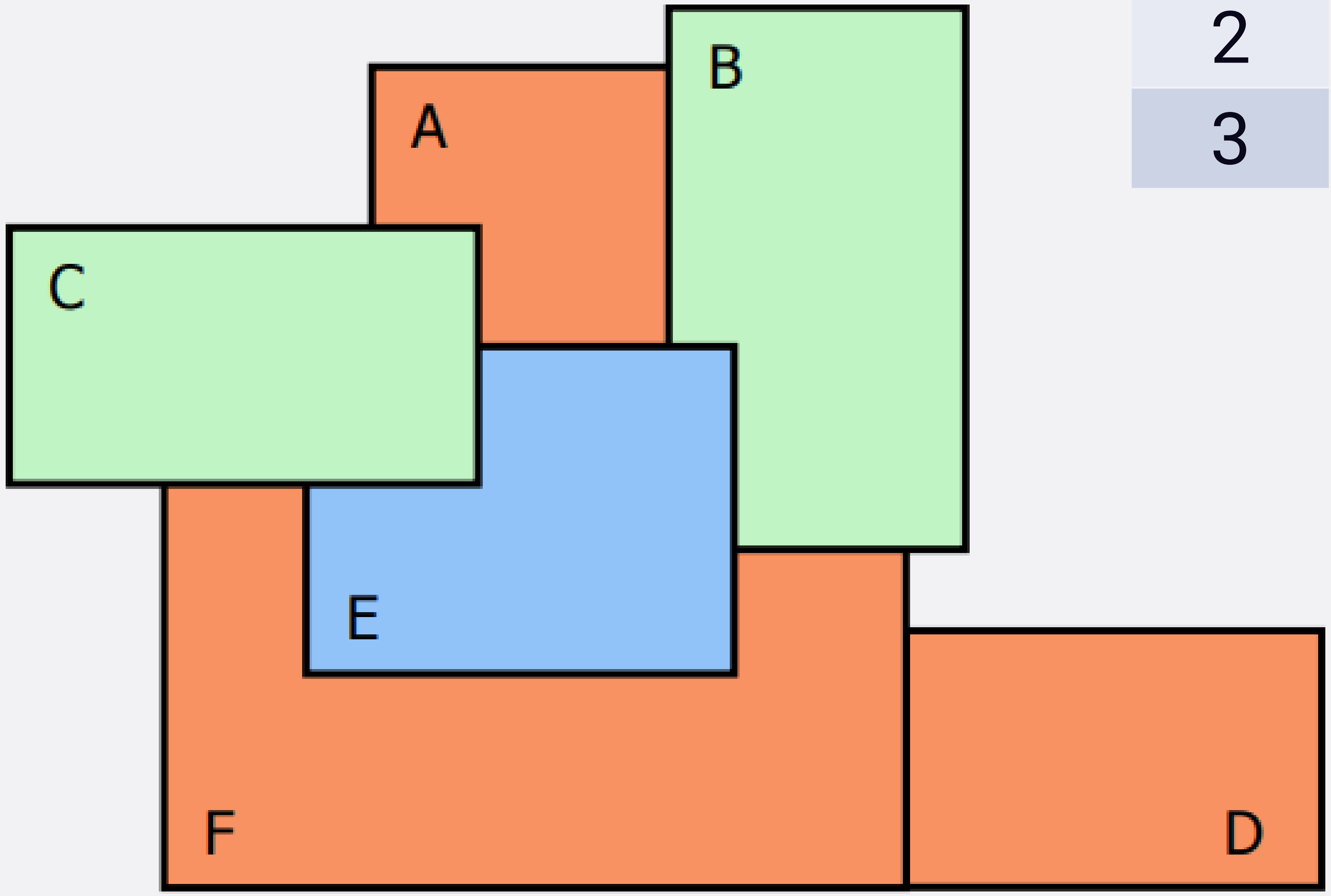
- Graph coloring: start with random coloring of nodes and change color of one node to reduce the number of conflicts.



Step	A	B	C	D	E	F	# conflicts
1	r	g	b	r	b	b	3 {CE, CF, EF}
2	r	g	**G**	r	b	b	1 {EF}

Exercise 4.3

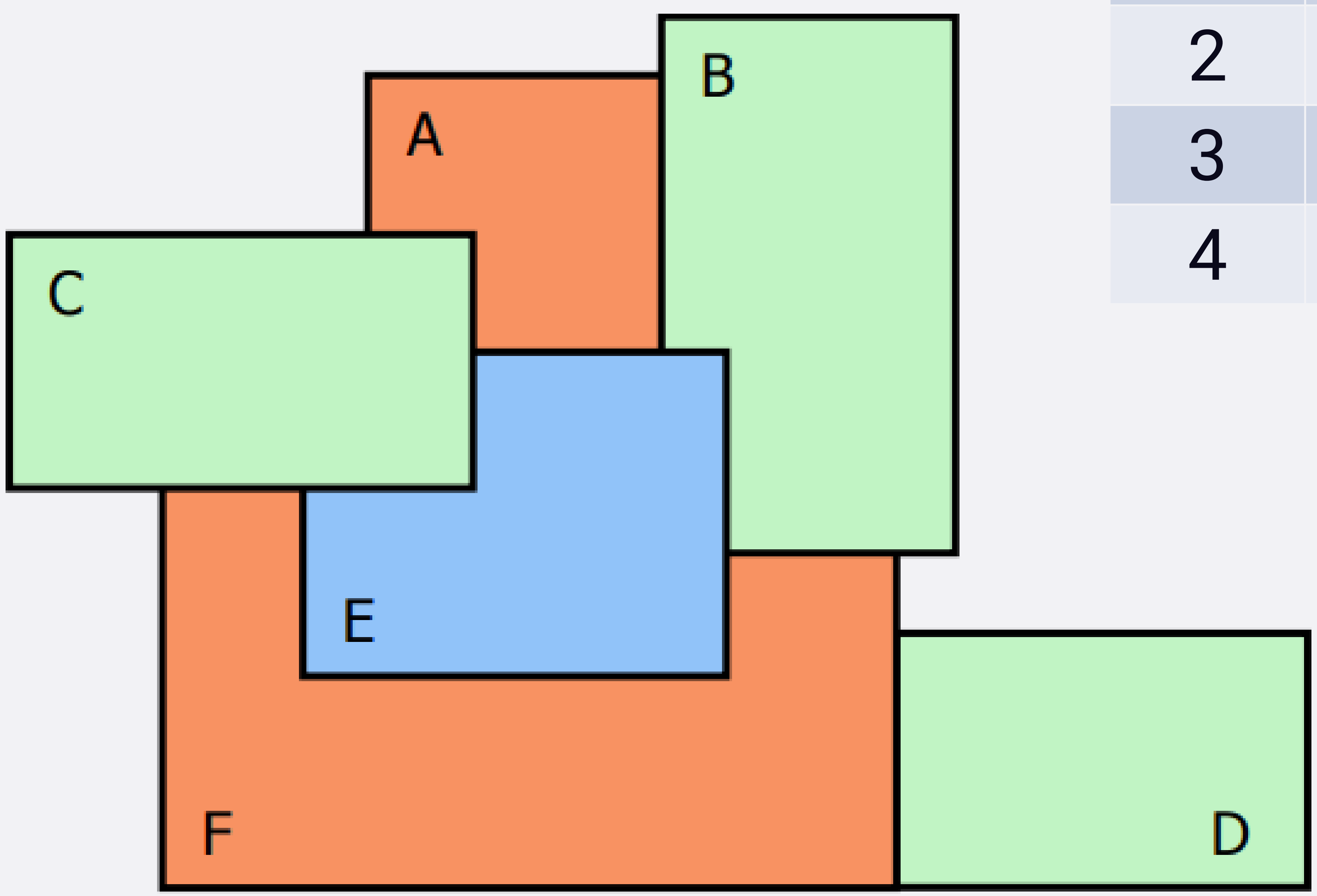
- Graph coloring: start with random coloring of nodes and change color of one node to reduce the number of conflicts.



Step	A	B	C	D	E	F	# conflicts
1	r	g	b	r	b	b	3 {CE, CF, EF}
2	r	g	**G**	r	b	b	1 {EF}
3	r	g	g	r	b	**R**	1 {DF}

Exercise 4.3

- Graph coloring: start with random coloring of nodes and change color of one node to reduce the number of conflicts.

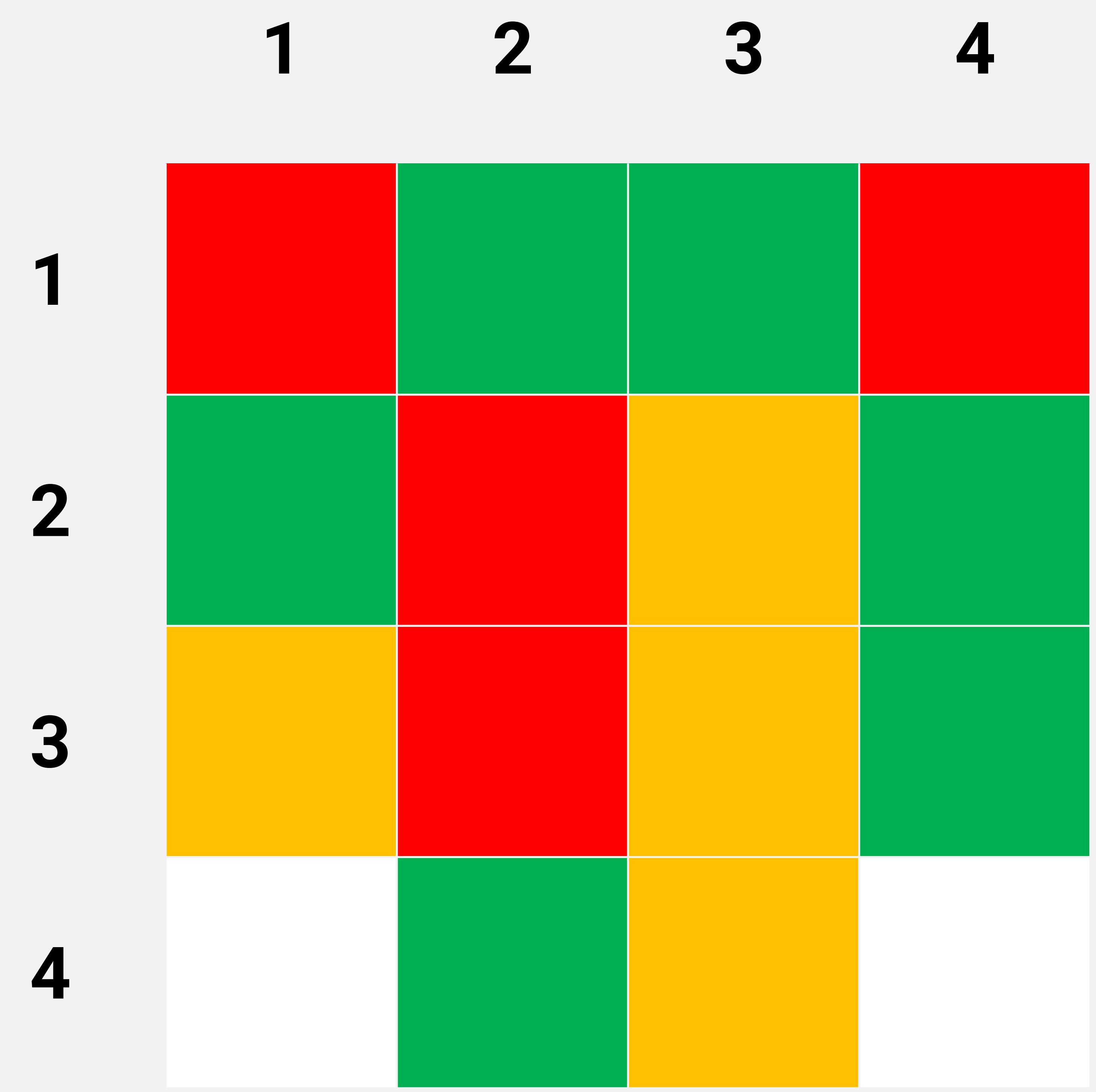


Step	A	B	C	D	E	F	# conflicts
1	r	g	b	r	b	b	3 {CE, CF, EF}
2	r	g	**G**	r	b	b	1 {EF}
3	r	g	g	r	b	**R**	1 {DF}
4	r	g	g	**G**	b	r	0

Exercise 4.4

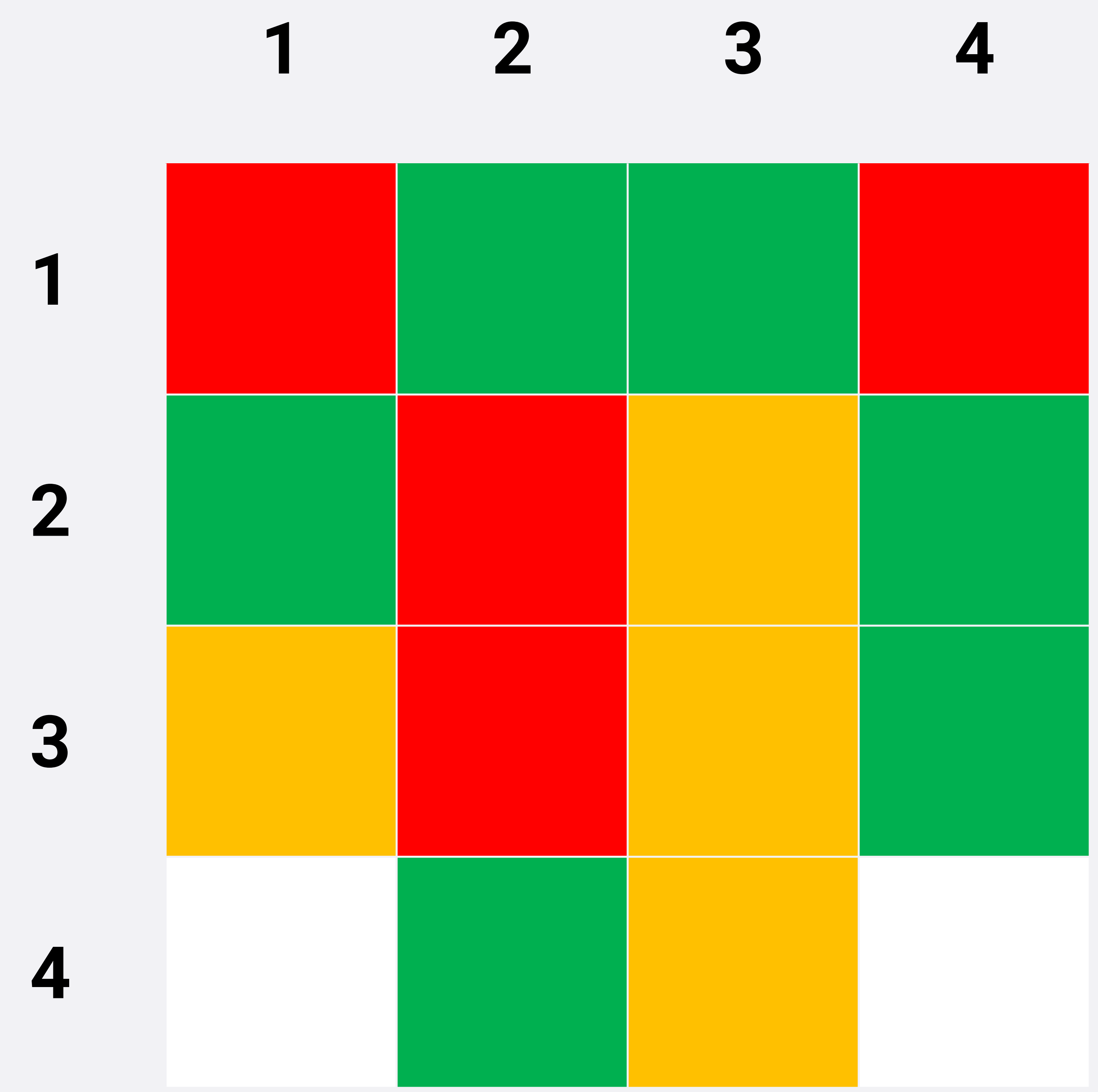
- Solve the Column Jump game

- The initial board has some configuration of colored balls with at least one empty space.
  The objective is to remove all but one ball from the board.
  Balls are removed when they are jumped according to the following rules. If a ball of one color jumps over a different colored ball to an empty space, the jumped ball is removed from the board.
  Additionally, if multiple balls of the same color are in a line, they can be jumped and removed together (by a different colored ball, provided that an empty space is on the other side of the line).



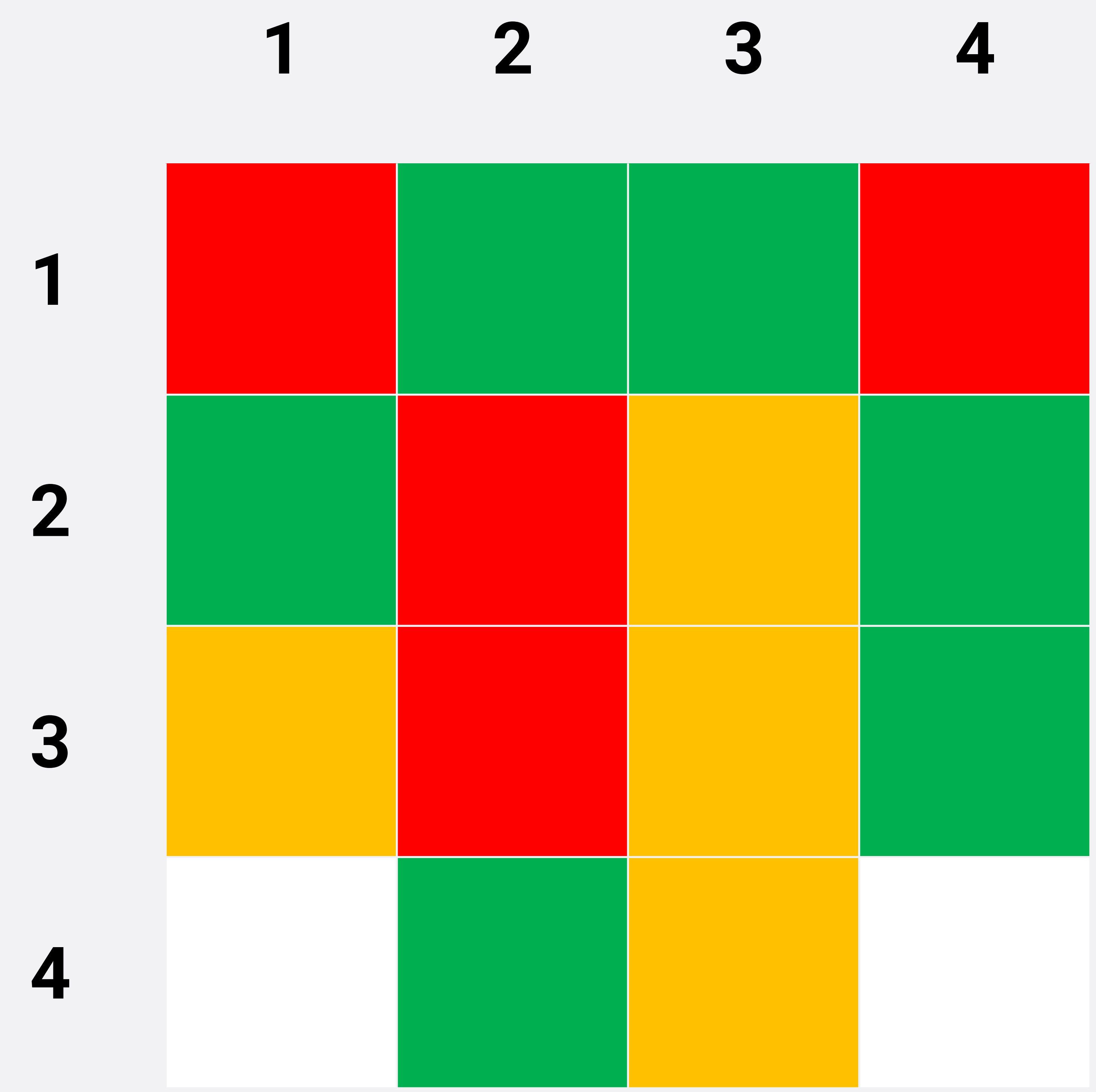
Exercise 4.4

- Provide possible heuristics for solving the game.



Exercise 4.4

- Color code:
  White (empty cells): 0
  Red: 1
  Green: 2
  Orange: 3

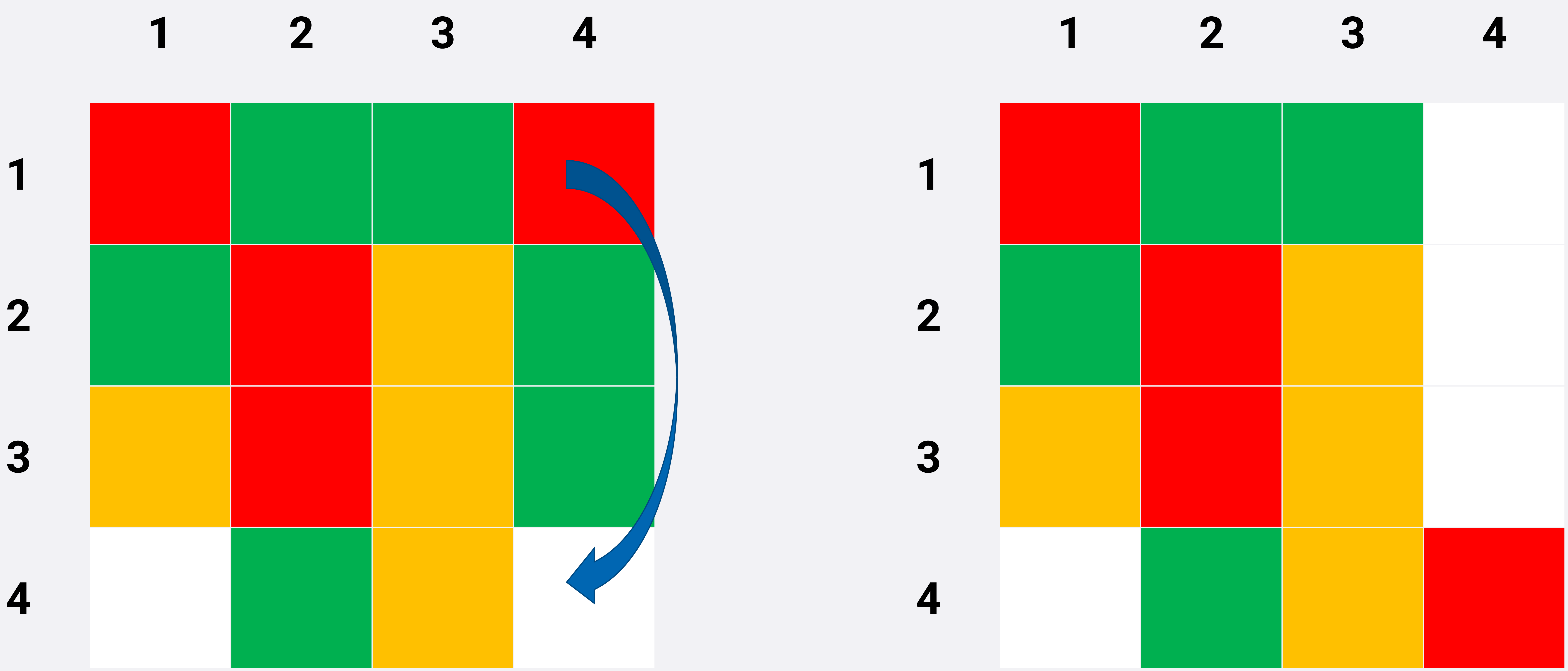


- Provide the series of moves required to solve the puzzle. A move is represented by two locations on the grid with (row, column) numbers.

- The grid locations follow this convention:
  (1,1) (1,2) (1,3) (1,4)
  (2,1) (2,2) (2,3) (2,4)
  (3,1) (3,2) (3,3) (3,4)
  (4,1) (4,2) (4,3) (4,4)

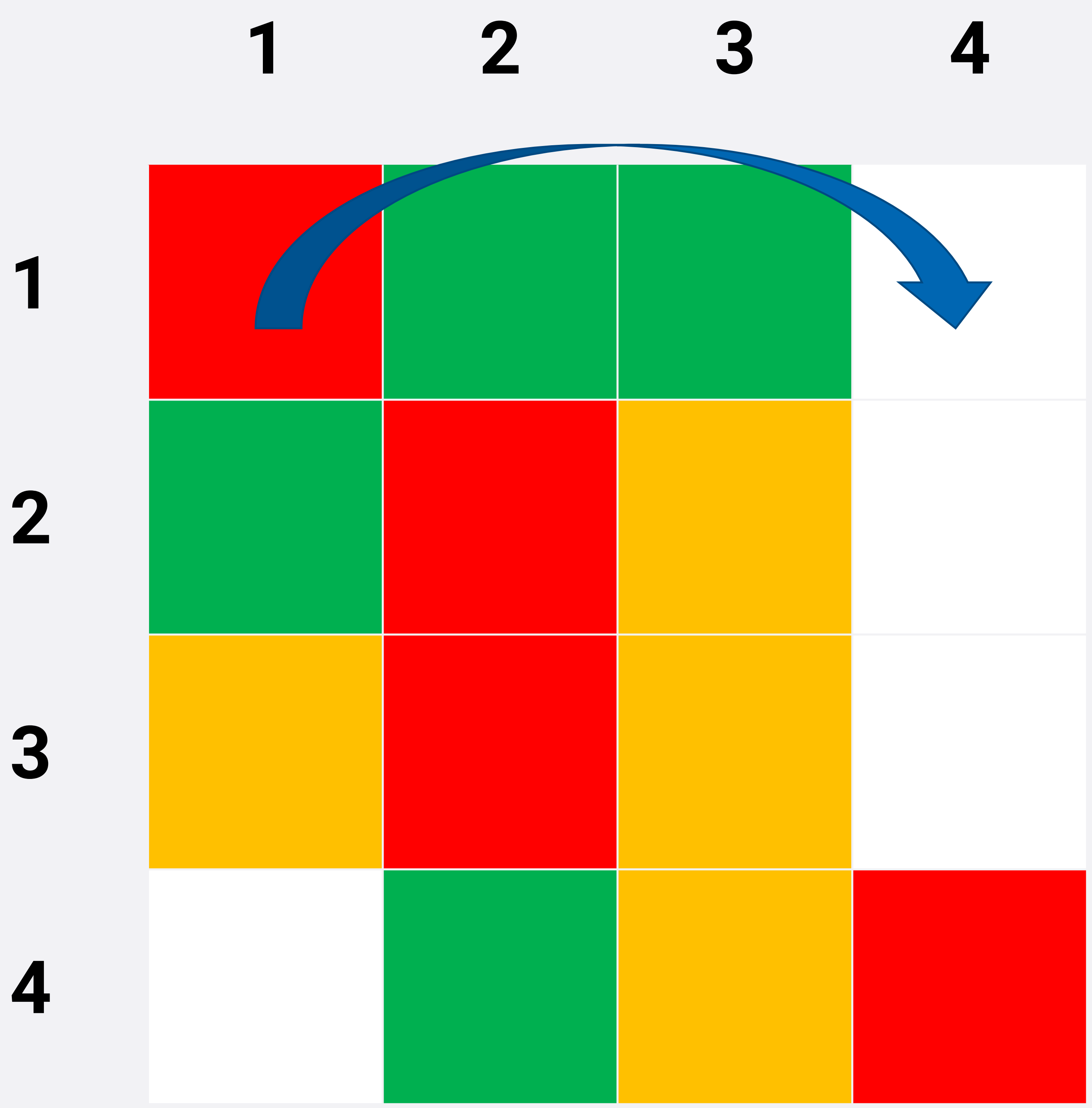
Exercise 4.4

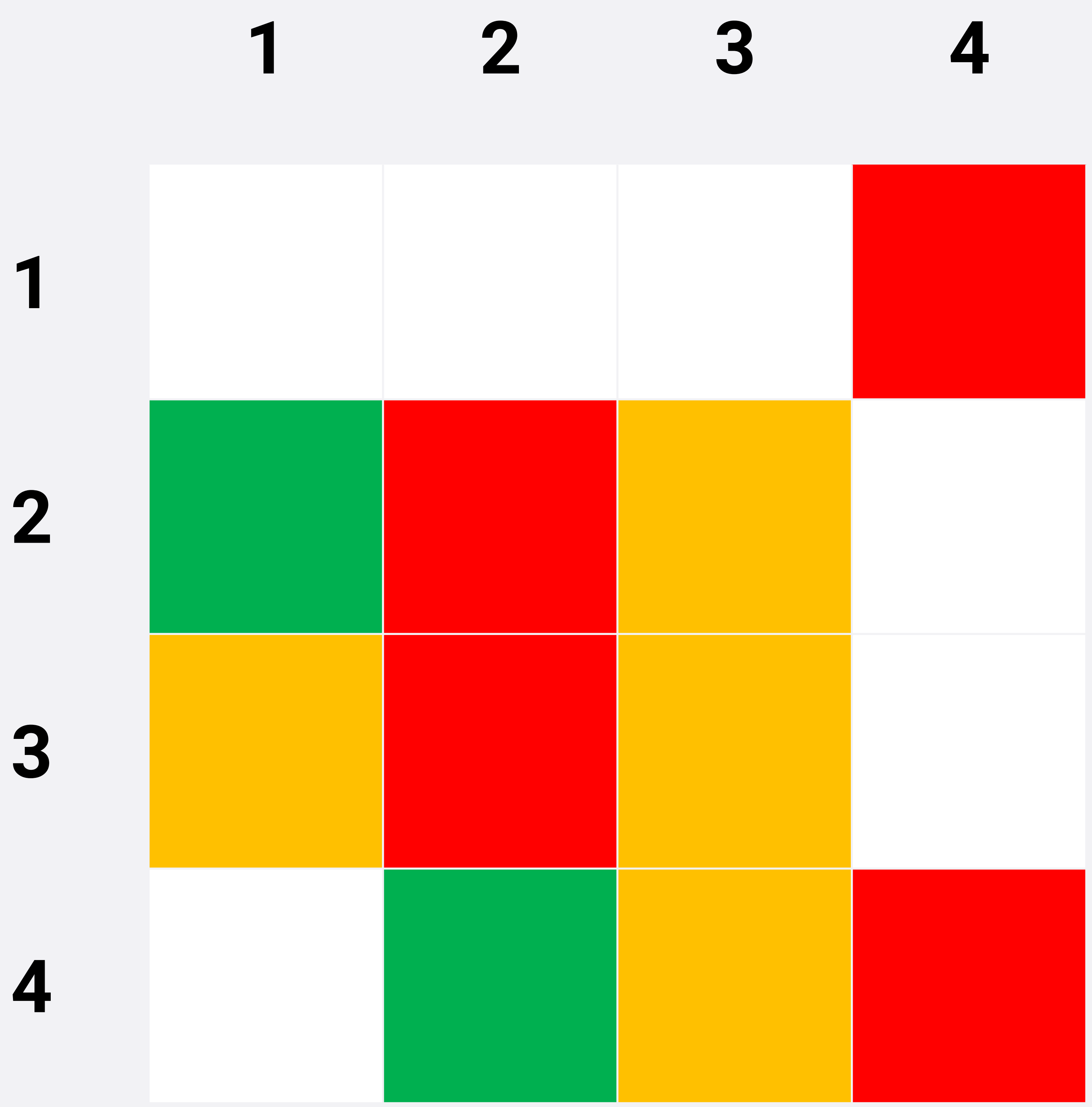
- By applying a “maximum-zero” heuristic – step 1



Exercise 4.4

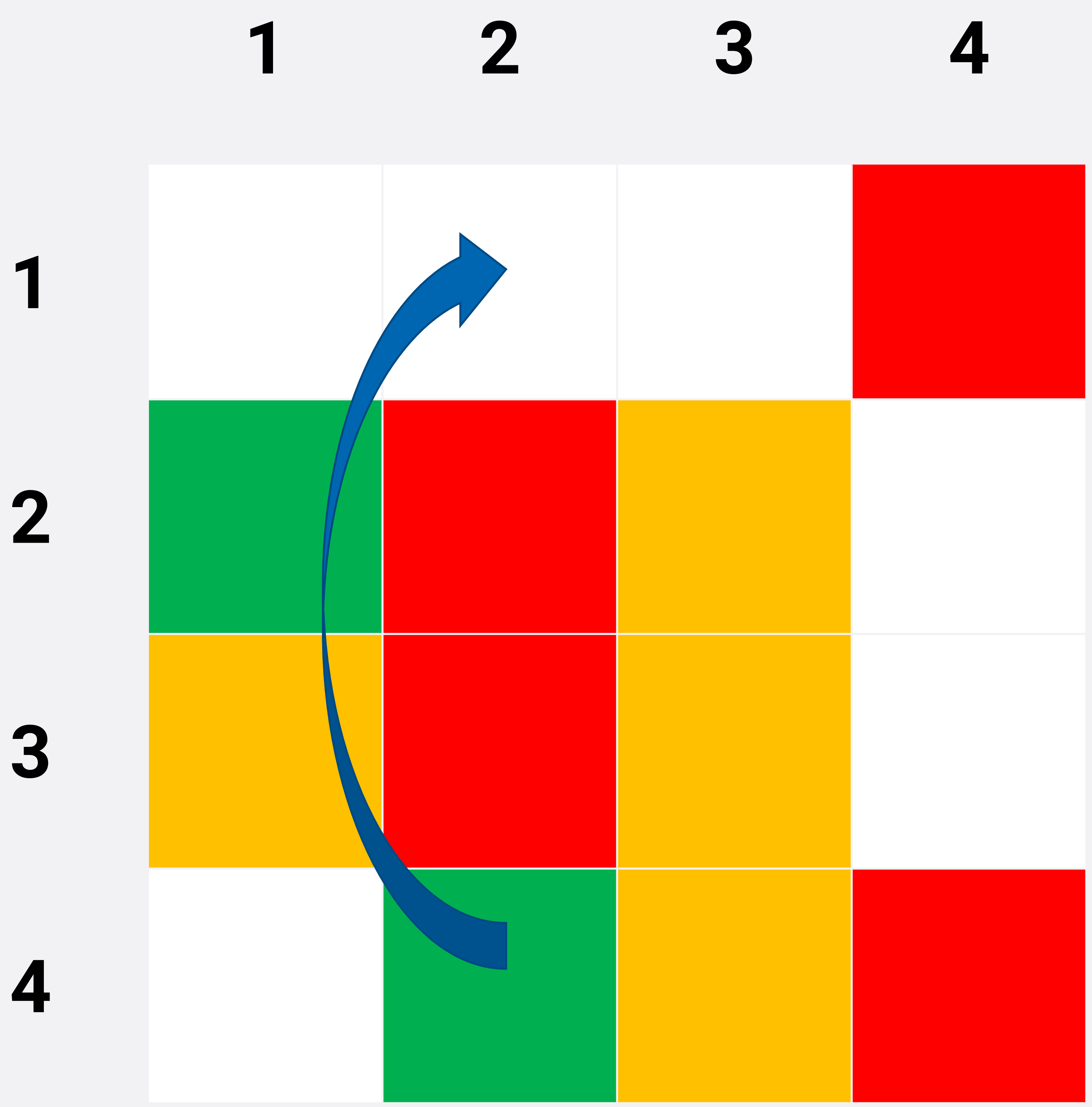
- By applying a "maximum-zero" heuristic – step 2

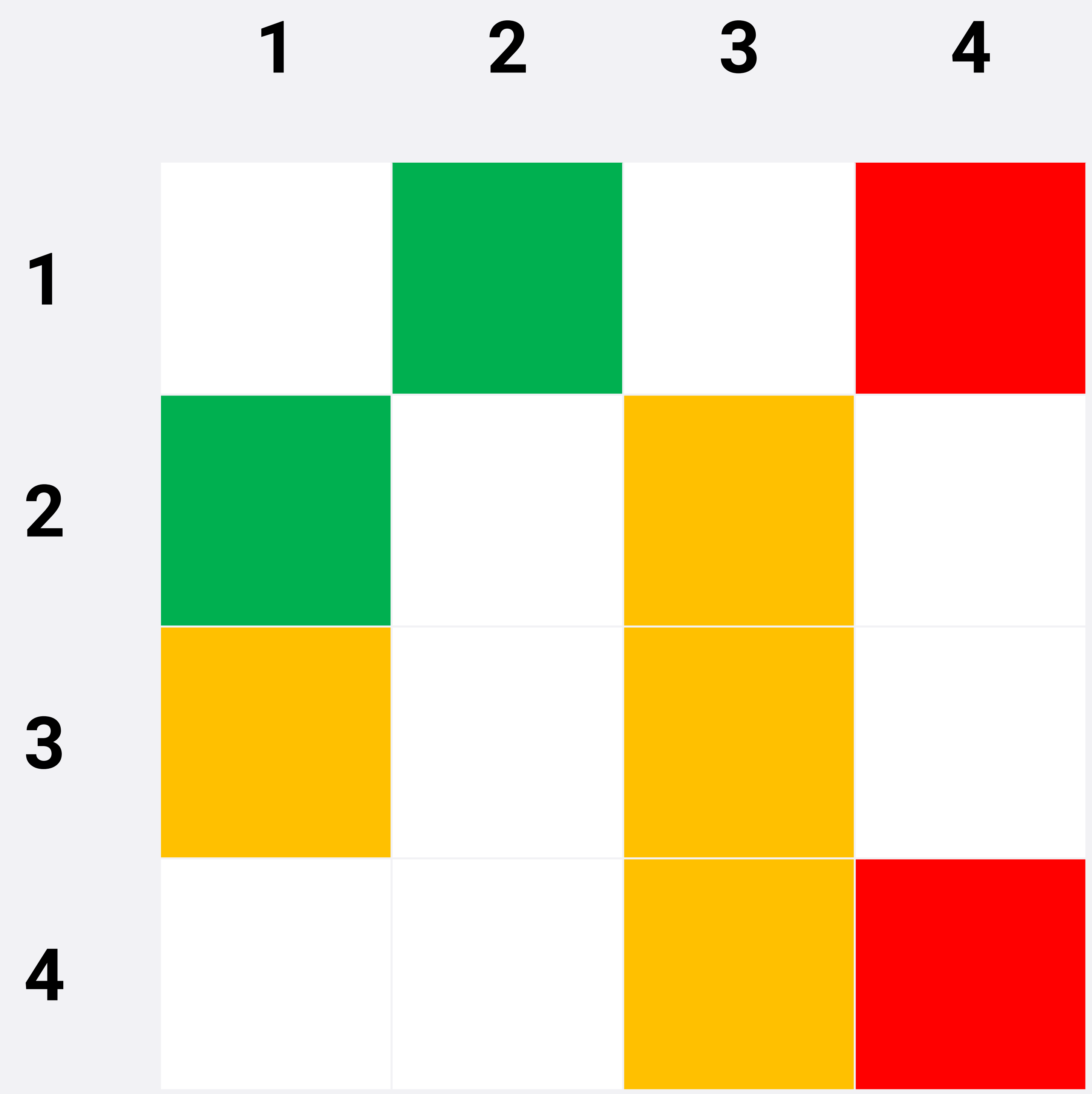




Exercise 4.4

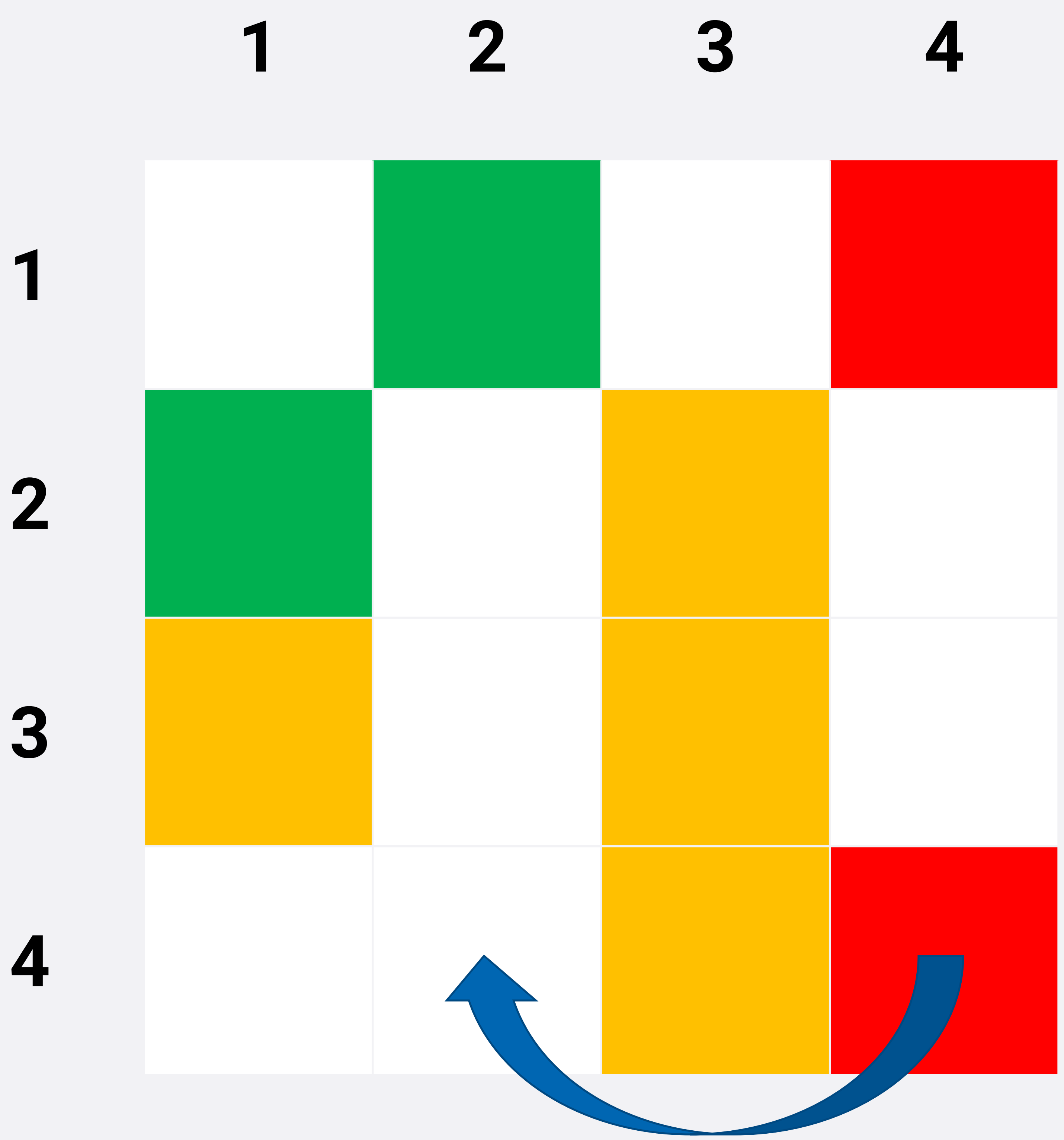
- By applying a “maximum-zero” heuristic – step 3

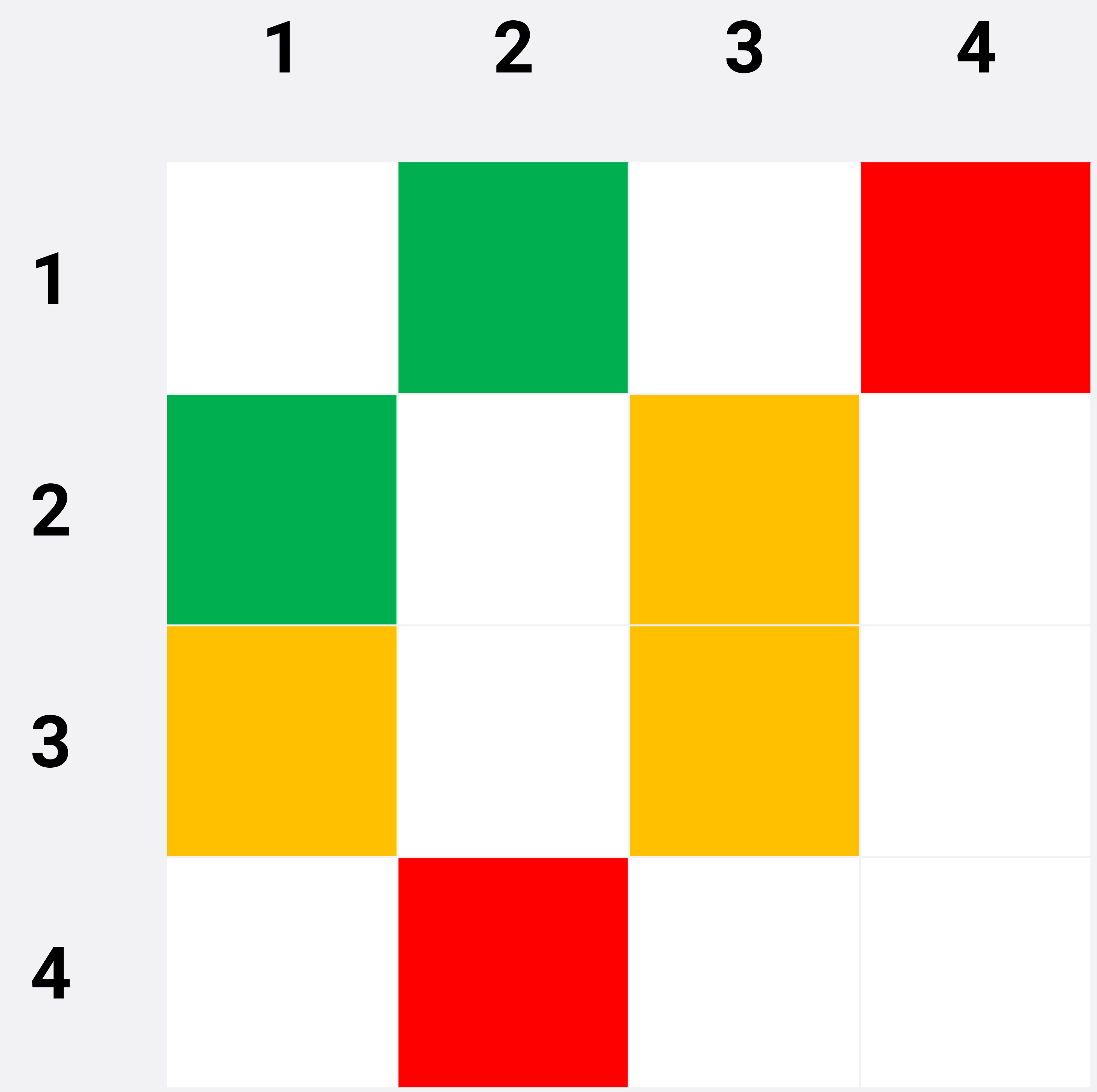




Exercise 4.4

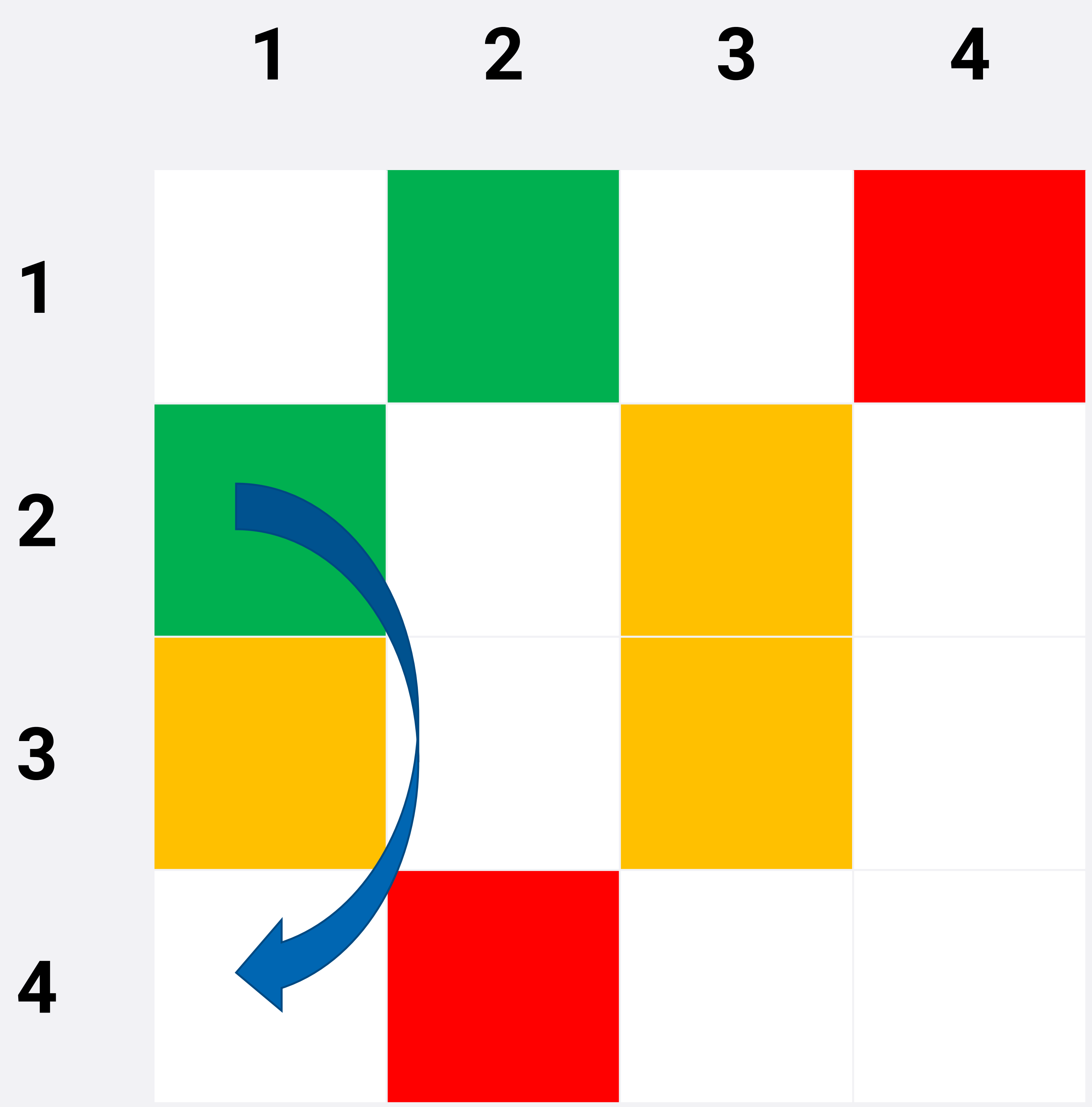
- By applying a “maximum-zero” heuristic – step 4

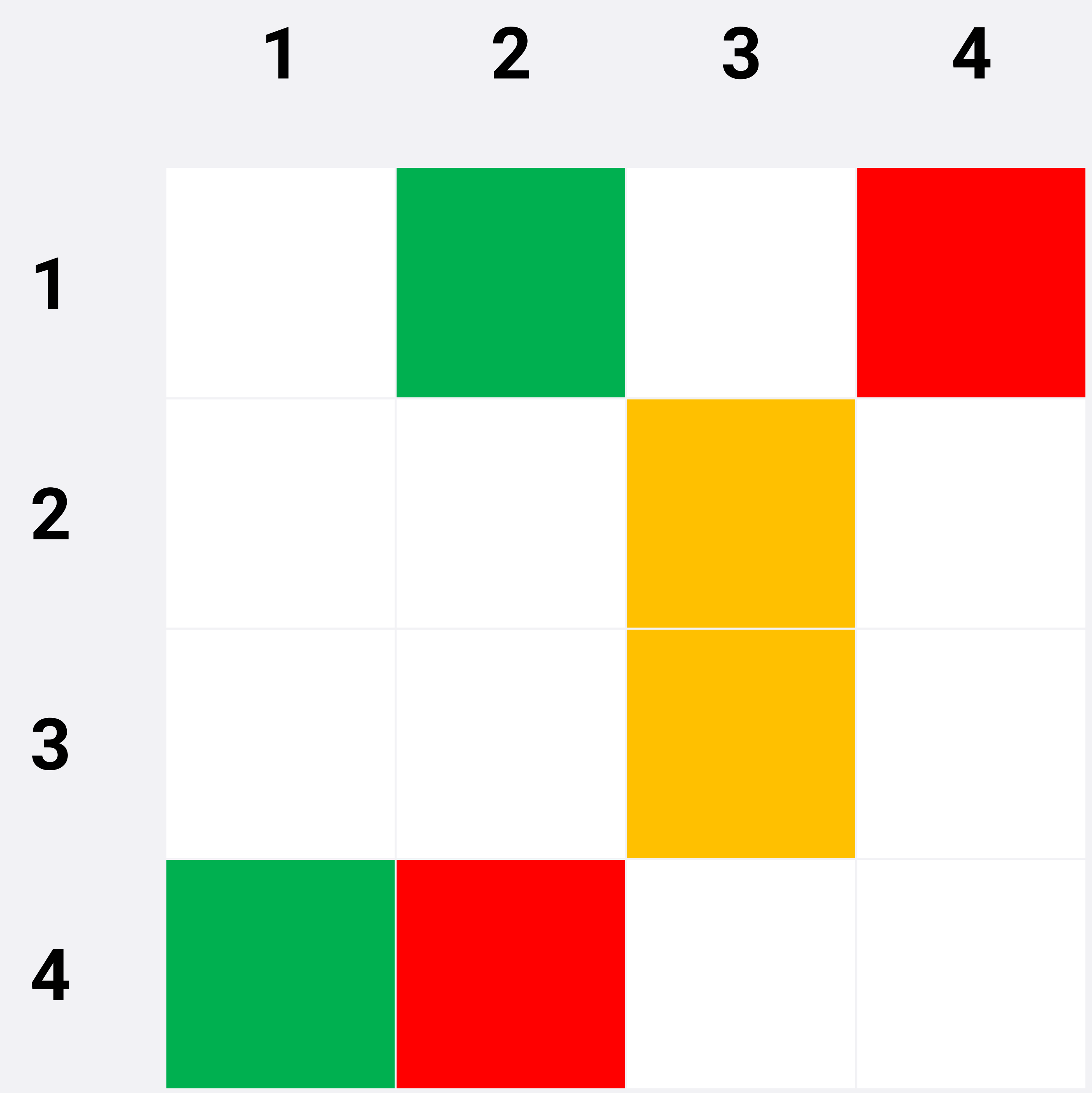




Exercise 4.4

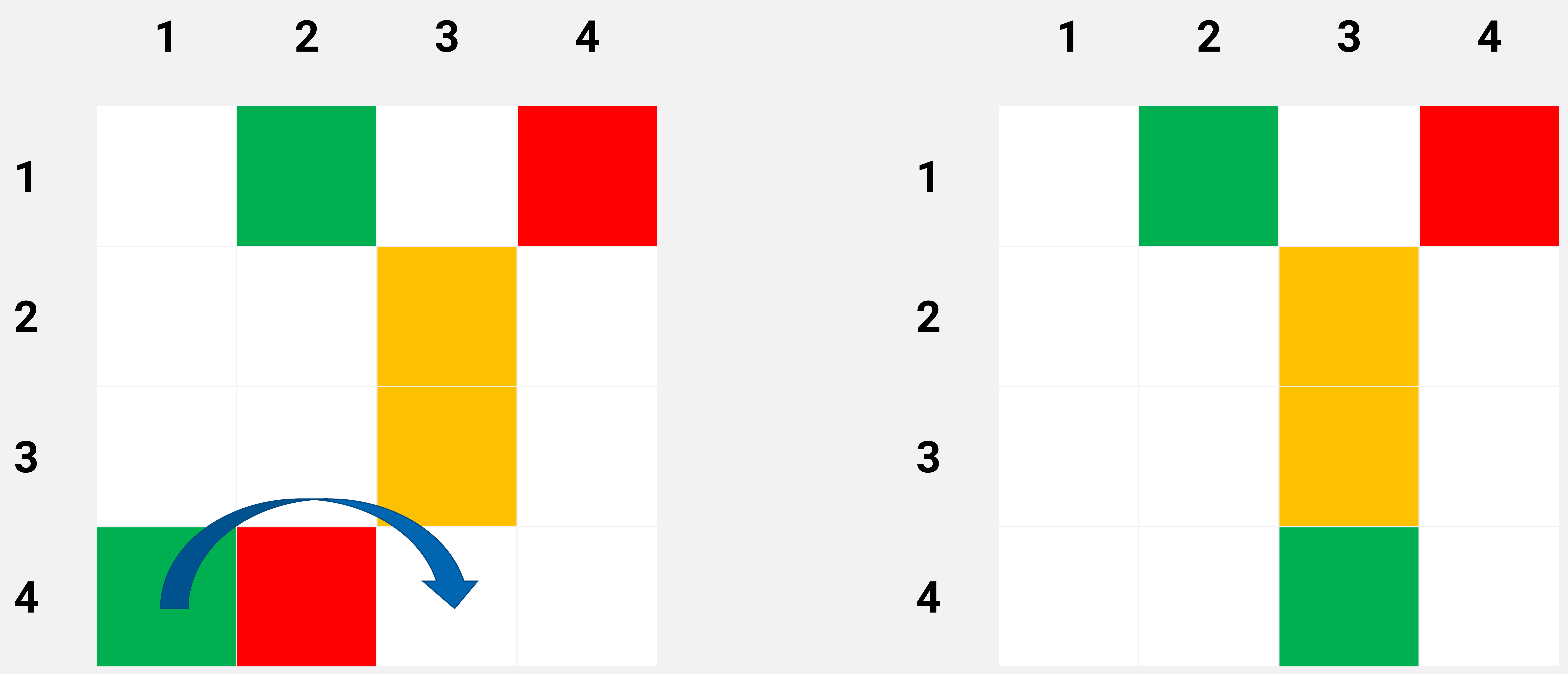
- By applying a “maximum-zero” heuristic – step 5





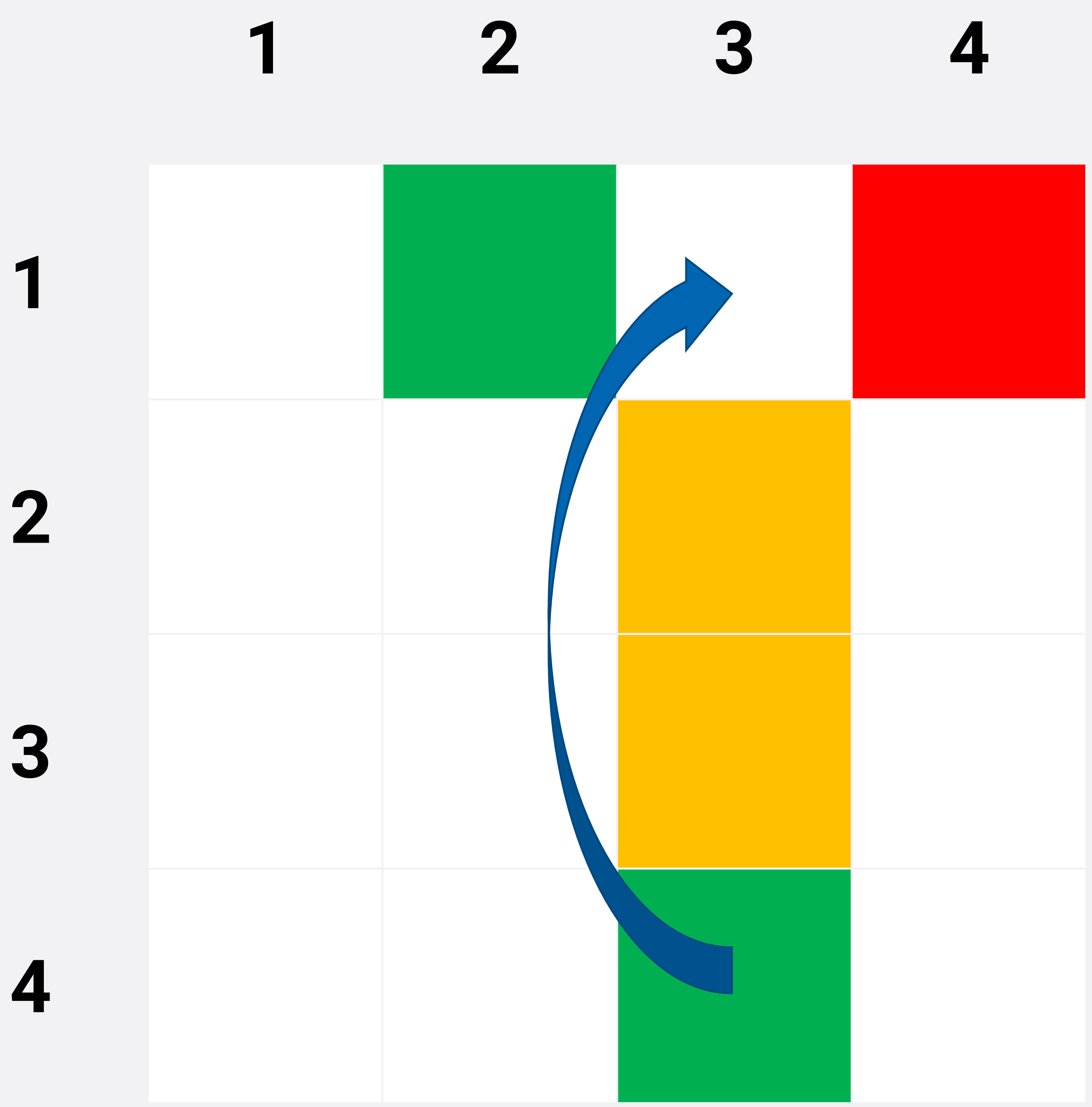
Exercise 4.4

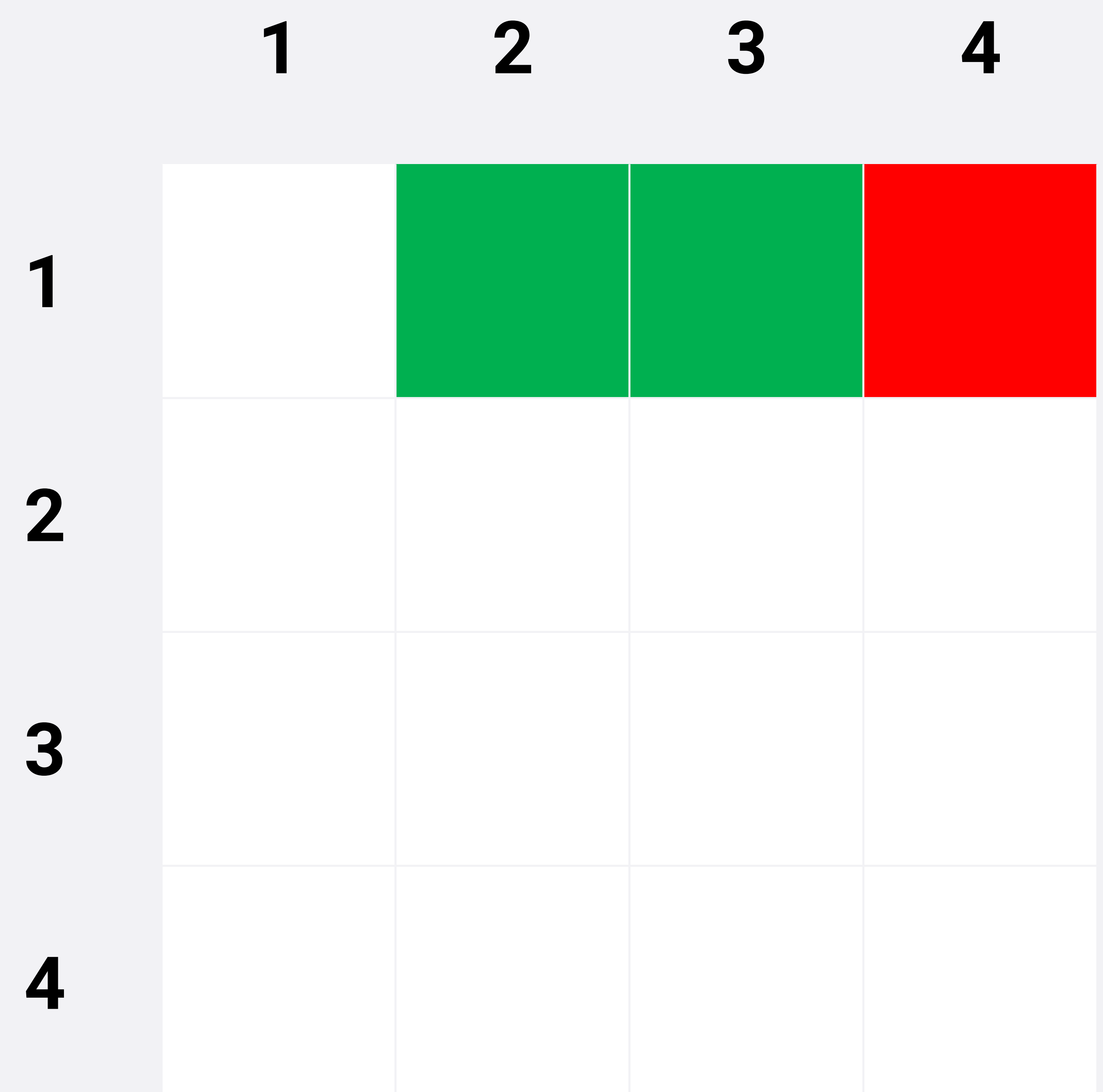
- By applying a “maximum-zero” heuristic – step 6



Exercise 4.4

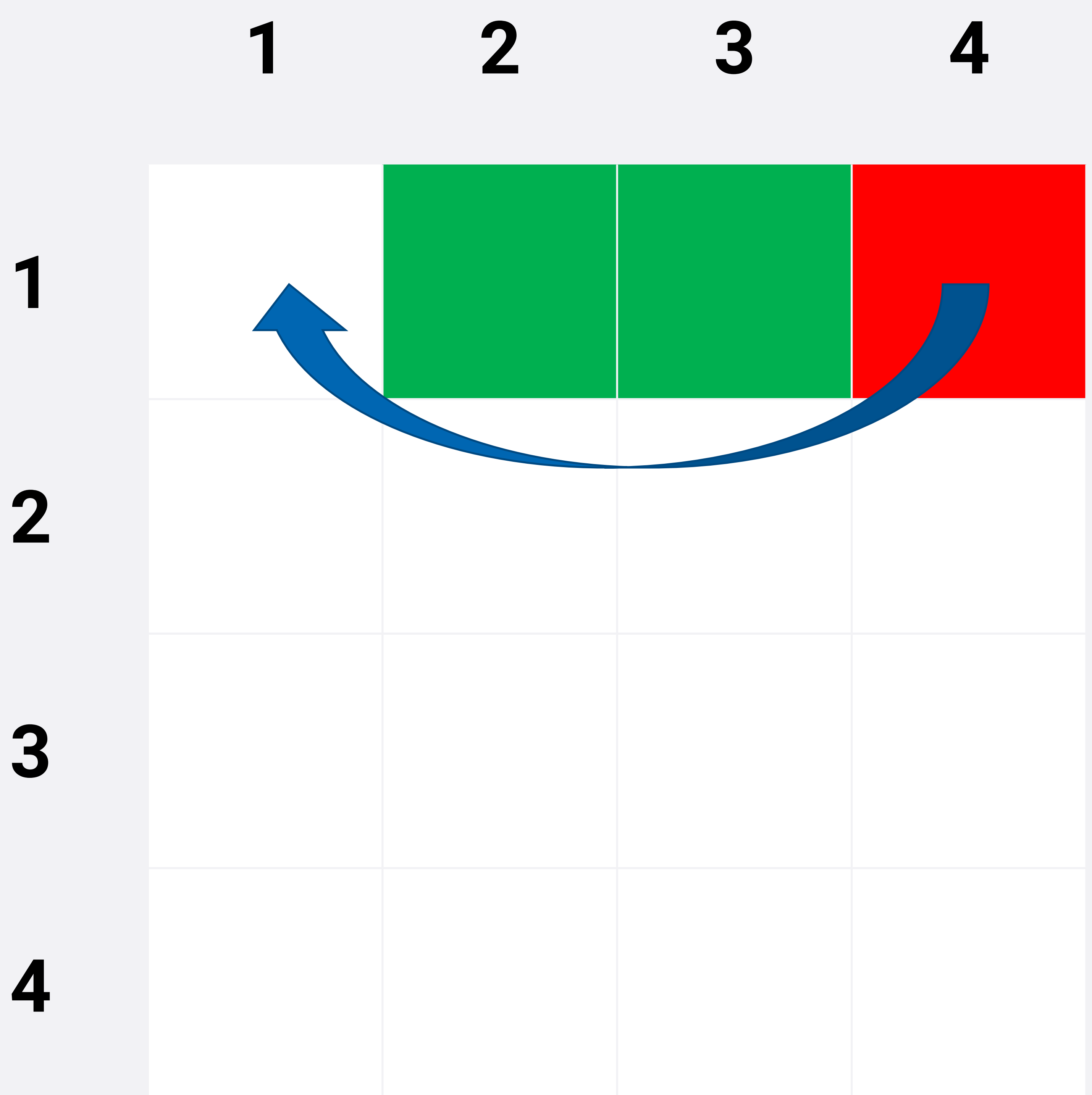
- By applying a “maximum-zero” heuristic – step 7

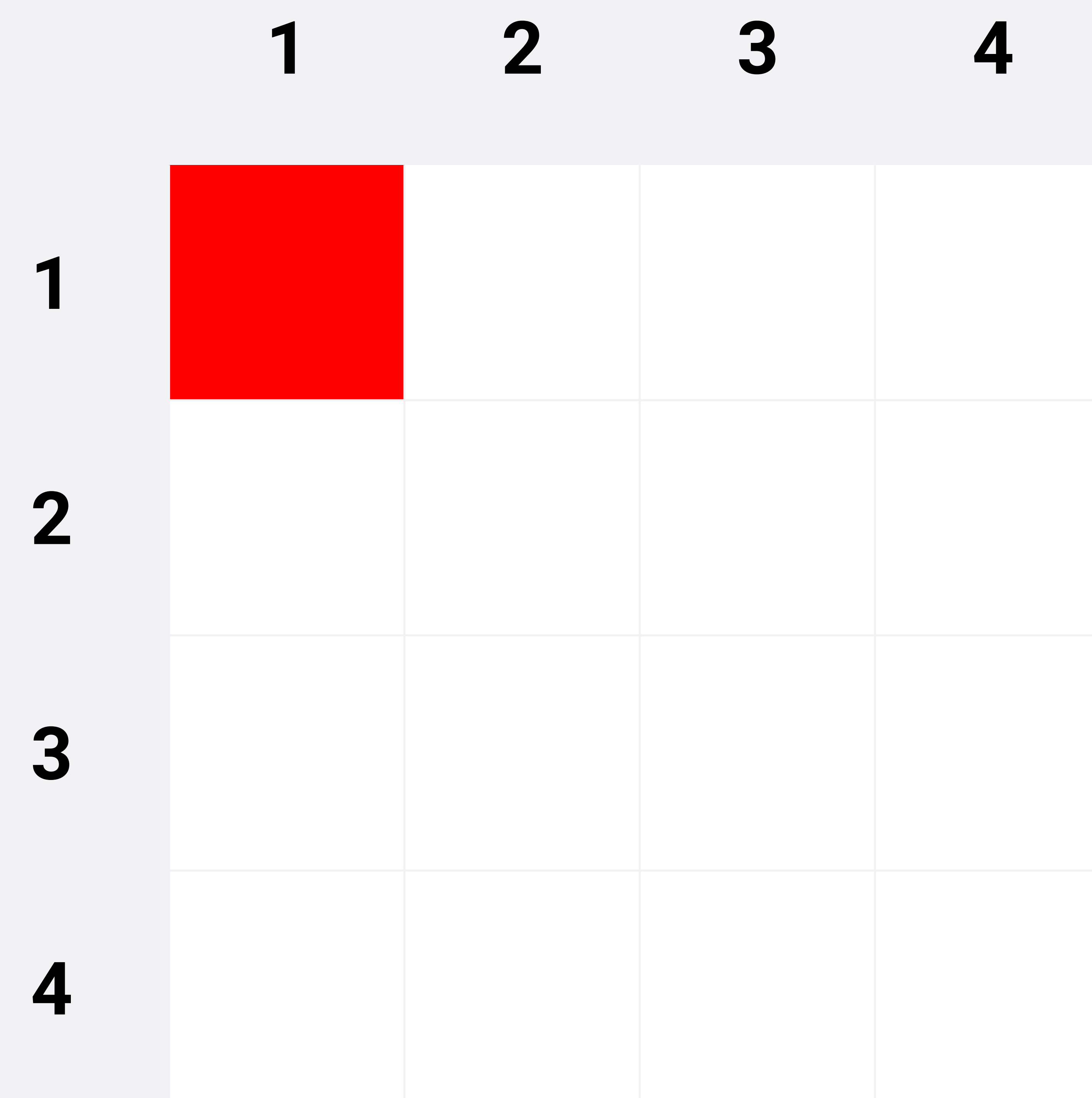




Exercise 4.4

- By applying a “maximum-zero” heuristic – step 8





Exercise 4.5 - Homework

- Color code:
  White (empty cells): 0
  Red: 1
  Green: 2
  Orange: 3
  Blue: 4

- Provide the series of moves required to solve the puzzle. A move is represented by two locations on the grid with (row, column) numbers.

- The grid locations follow this convention:
  (1,1) (1,2) (1,3) (1,4)
  (2,1) (2,2) (2,3) (2,4)
  (3,1) (3,2) (3,3) (3,4)
  (4,1) (4,2) (4,3) (4,4)