04-10-2023

CS051A

INTRO TO COMPUTER SCIENCE WITH TOPICS IN AI

20: Informed and adversarial search

Alexandra Papoutsaki

she/her/hers

Lectures

Zilong Ye

he/him/his

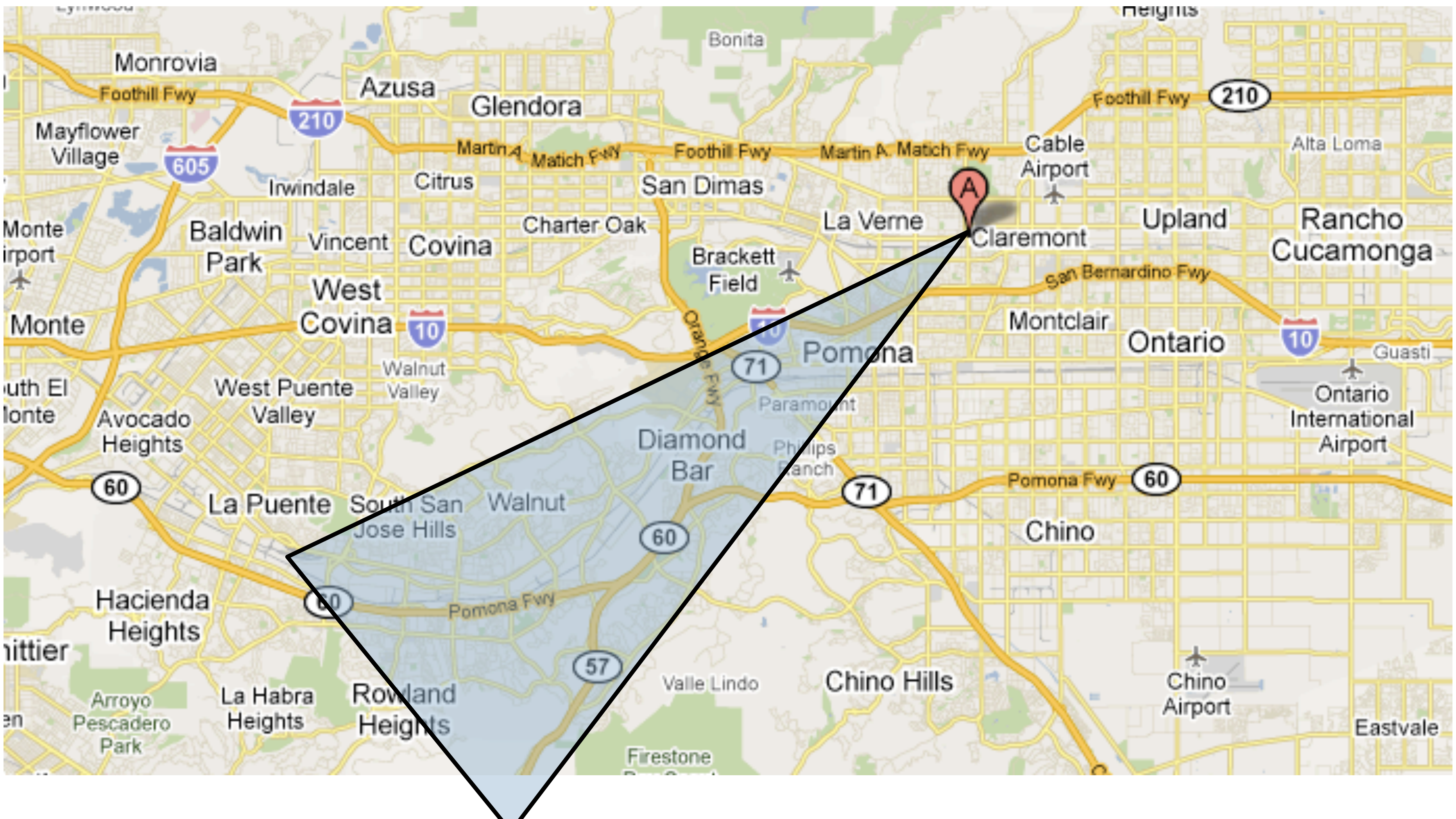
Labs

TODAY'S LECTURE IN A NUTSHELL

Lecture 20: Informed and adversarial search

- Informed search
- Adversarial search

We'd like to bias search towards the actual solution



Informed search

- Order to_visit based on some knowledge of the world that estimates how "good" a state is
    - h(n) is called an evaluation function

- Best-first search
    - rank to_visit based on h(n)
    - take the most desirable state in to_visit first
    - different approaches depending on how we define h(n)

Heuristic function: h(n)

- An estimate of how close the node is to a goal
- Uses domain-specific knowledge!
- Examples
  - Map path finding?
    - straight-line distance from the node to the goal (“as the crow flies”)
  - 8-puzzle?
    - how many tiles are out of place
    - sum of the “distances” of the out of place tiles
  - Foxes and Chickens?
    - number of passengers on the starting bank

Two heuristics

2	8	3
1	6	4
	7	5

1	2	3
8	6	4
	7	5

6	2	3
8		4
7	1	5

Which state is better?

1	2	3
8		4
7	6	5

GOAL

Two heuristics

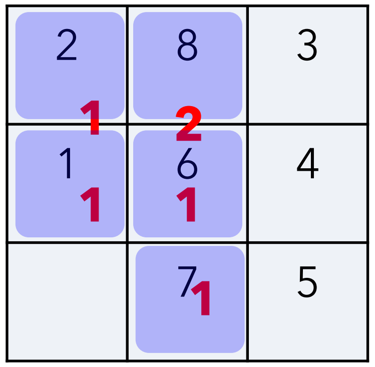
2	8	3
1	6	4
	7	5

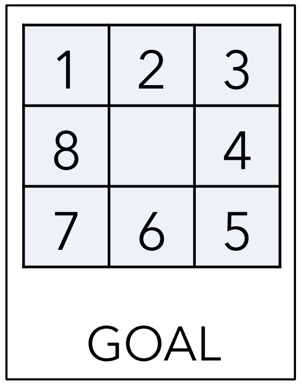
1	2	3
8		4
7	6	5

GOAL

What is the “distance” of the tiles that are out of place?

Two heuristics

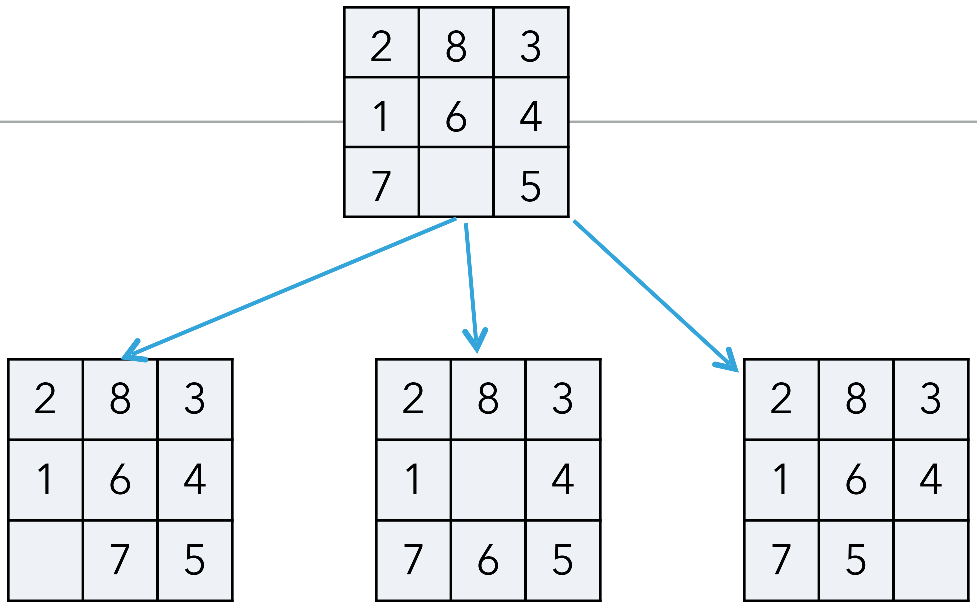




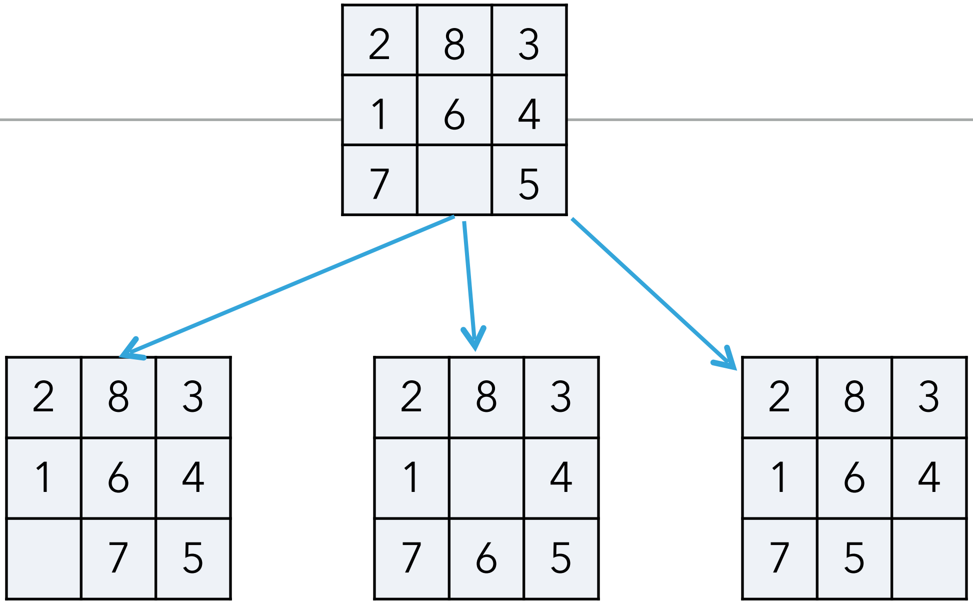
6

2	8	3
1	6	4
7		5

Next states?

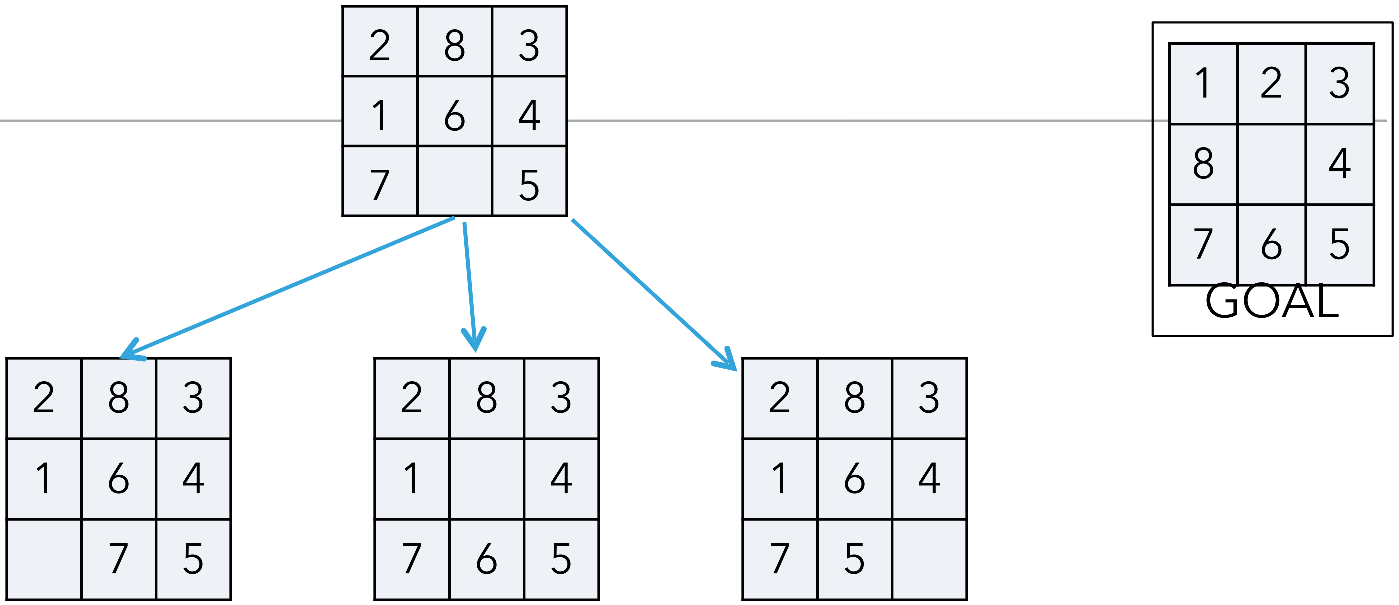


Which would you do?



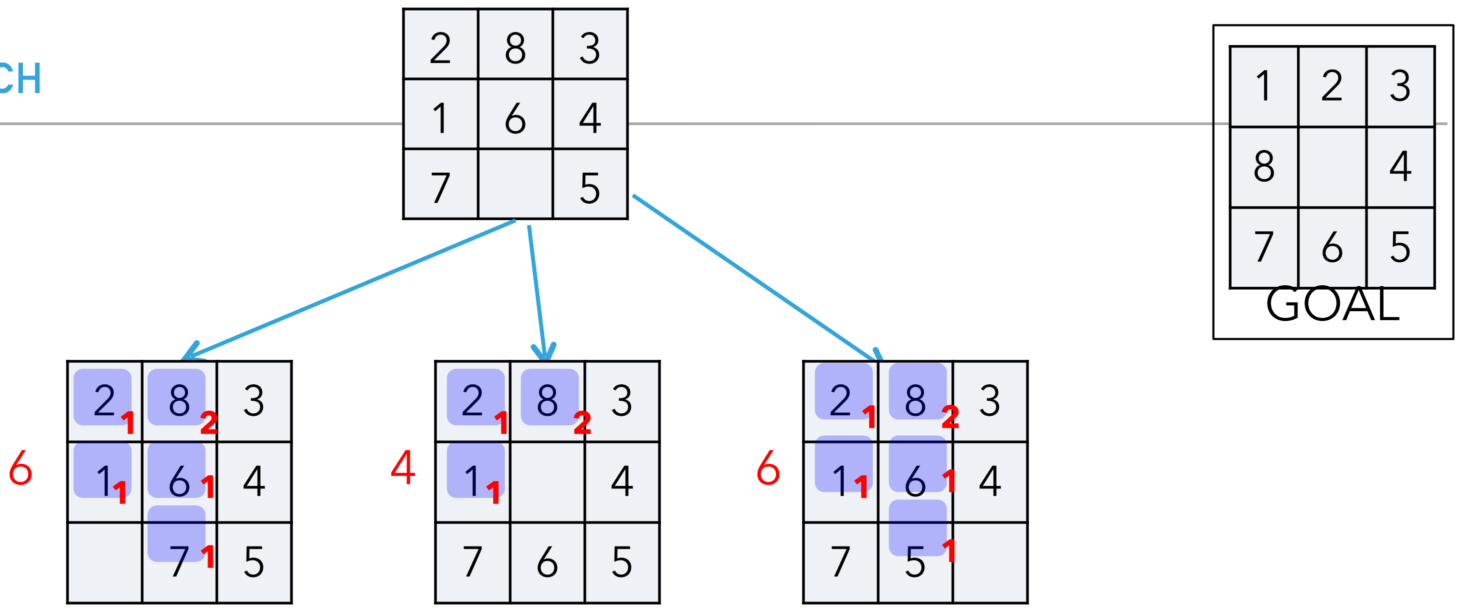
Which would DFS choose?

Completely depends on how next states are generated.
Not an “intelligent” decision!



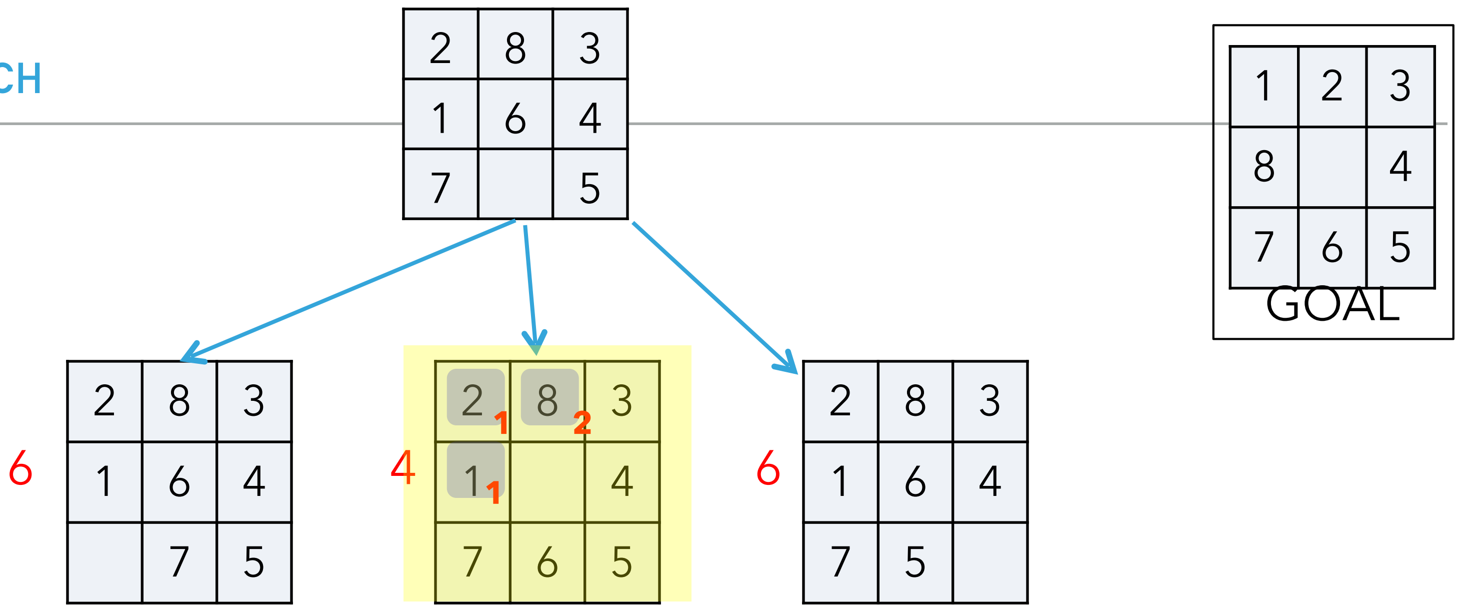
Best first search: distance of tiles?

INFORMED SEARCH

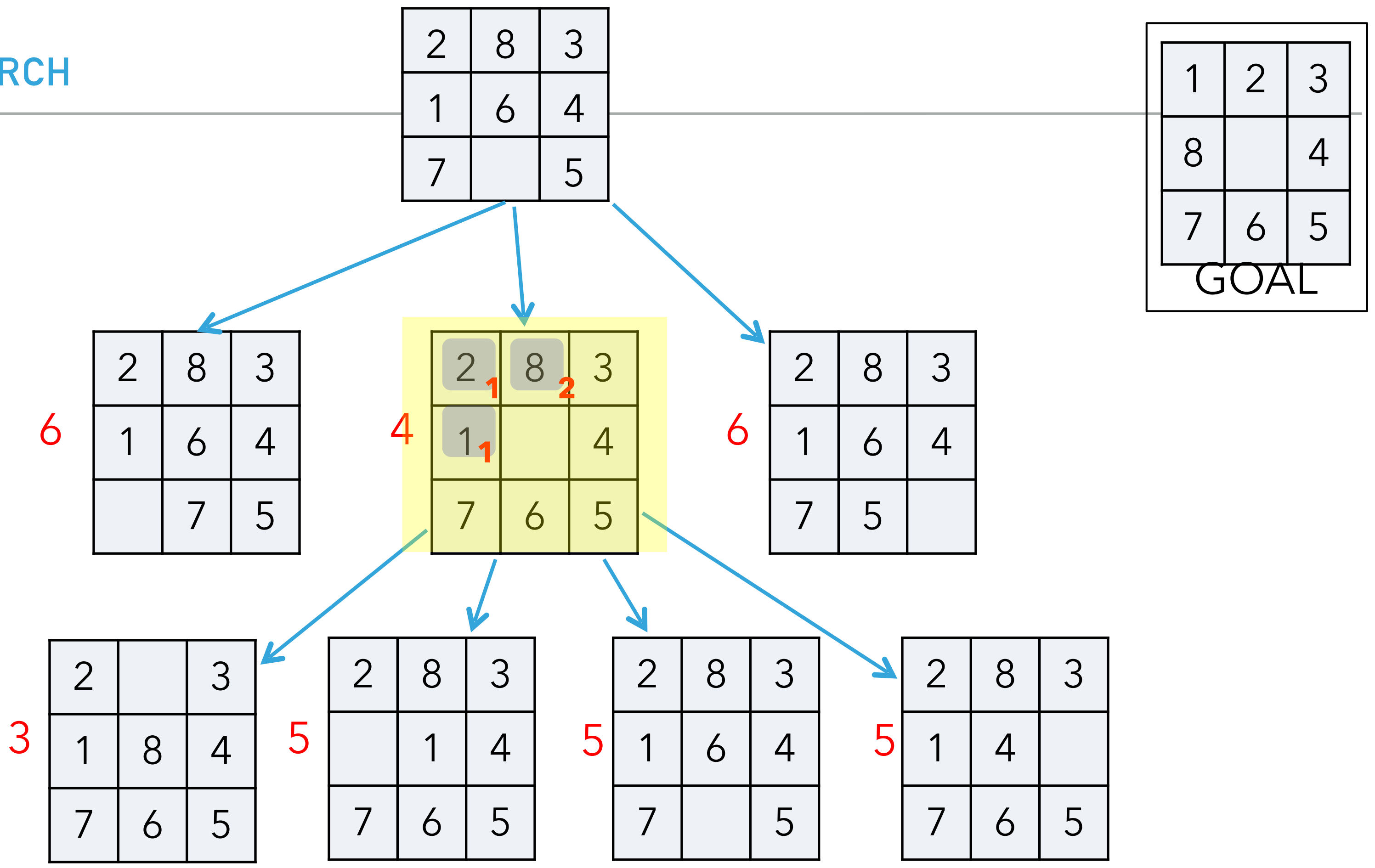


Which next for best first search?

INFORMED SEARCH

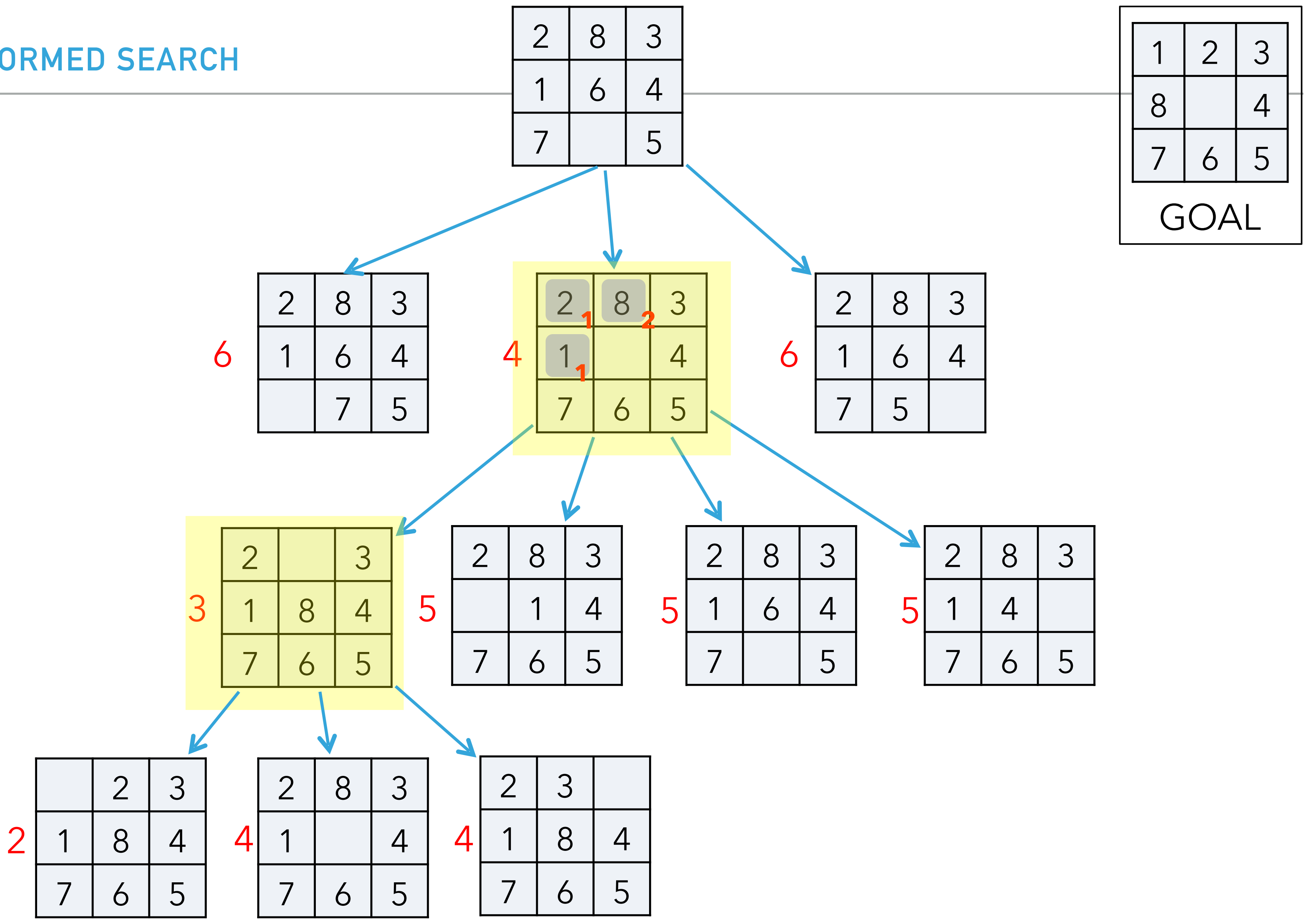


Which next for best first search?

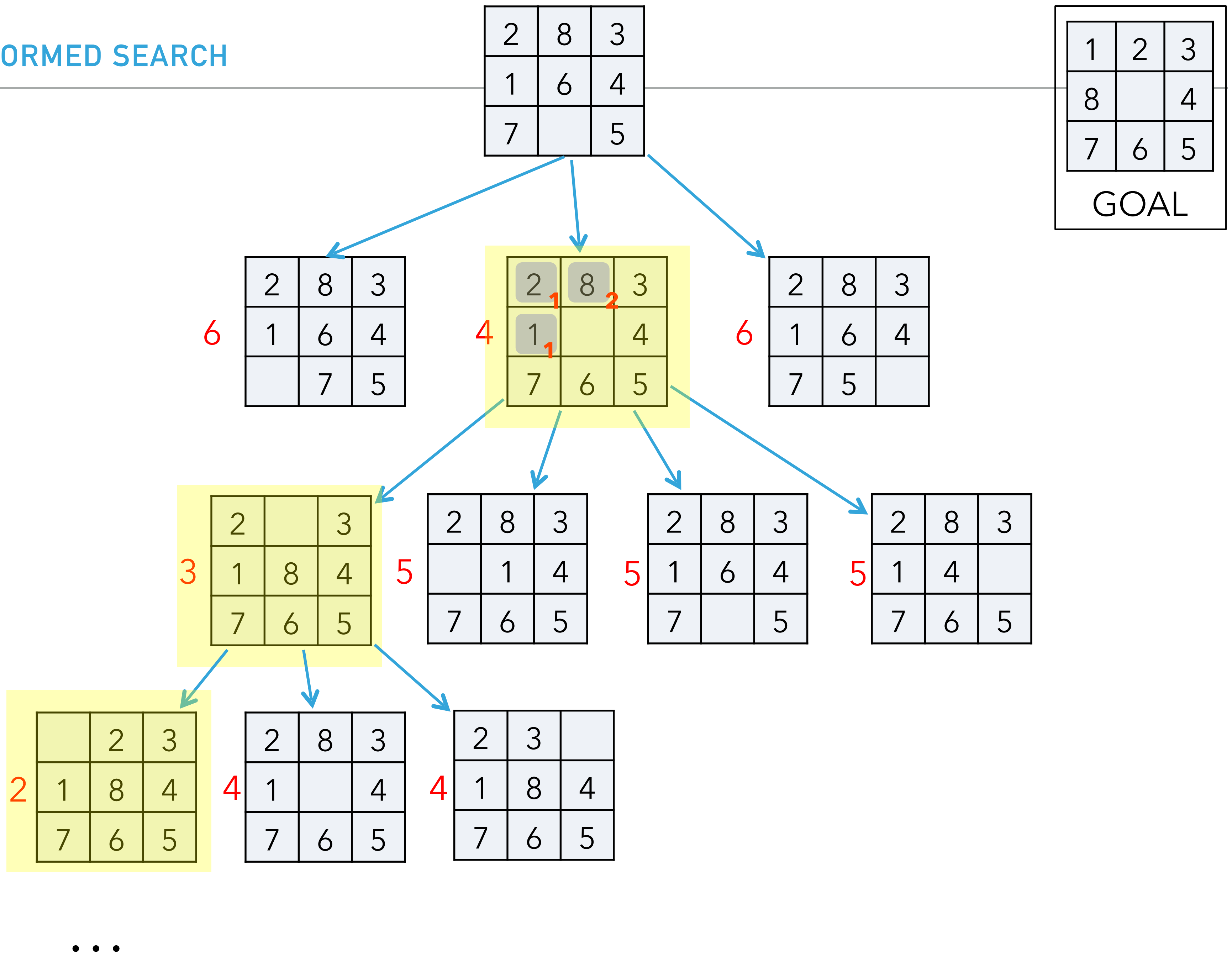


Which next for best first search?

INFORMED SEARCH



INFORMED SEARCH



Informed search

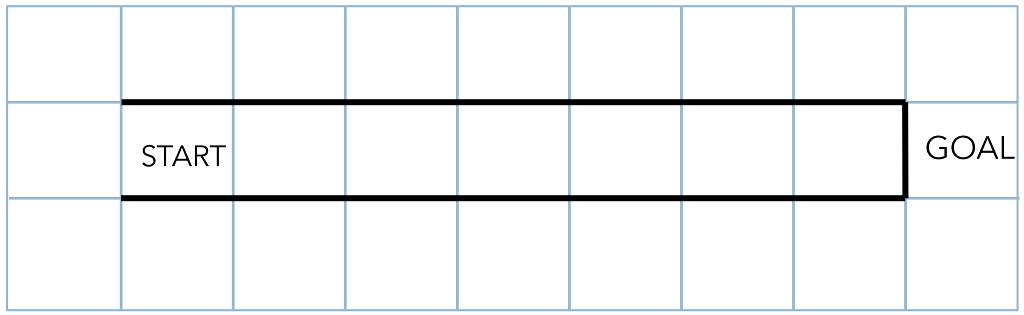
- Best first search is called an “informed” search algorithm
- Why wouldn’t we always use an informed algorithm?
  - Coming up with good heuristics can be hard for some problems.
  - There is computational overhead (both in calculating the heuristic and in keeping track of the next “best” state).

Informed search

- Any other problems/concerns about best first search?
  - Only as good as the heuristic function.

Informed search

- Any other problems/concerns about best first search?
  - Only as good as the heuristic function.

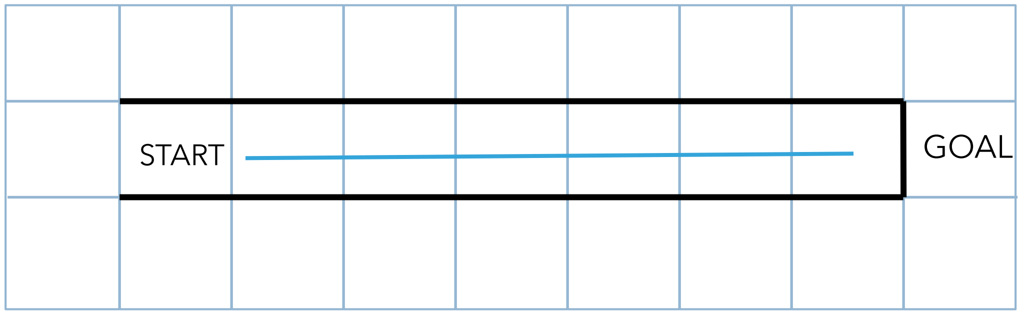


Best first search using distance as the crow flies as heuristic

What would the search do?

Informed search

- Any other problems/concerns about best first search?
  - Only as good as the heuristic function.

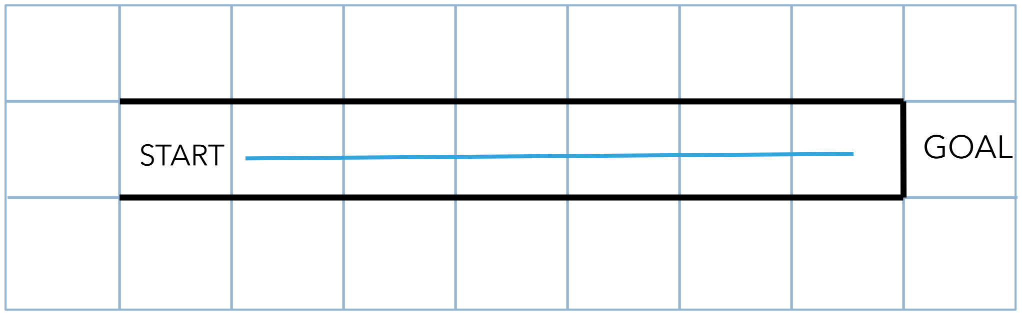


Best first search using distance as the crow flies as heuristic

What is the problem?

Informed search

- Any other problems/concerns about best first search?
  - Only as good as the heuristic function.



Best first search using distance as the crow flies as heuristic

Doesn't take into account how far it has come.

Best first search is a "greedy" algorithm

Informed search

- Best first search is called an “informed” search algorithm
- There are many other informed search algorithms:
  - A* search (and variants)
  - Theta*
  - Beam search

Sudoku

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

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Sudoku

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

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

INFORMED SEARCH

Sudoku

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

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

How can we pose this as a search problem?

State

Start state

Goal state

State space/transitions

Sudoku

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

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

How can we pose this as a search problem?

State: 9 x 9 grid with 1-9 or empty

Start state:

Goal state:

State space/transitions

Sudoku

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

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Sudoku

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

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

How many next states?
What are they?

Sudoku

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

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

1, 6, 7, 9

Sudoku

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

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

1, 6, 7, 9

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Sudoku

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

Generate next states:

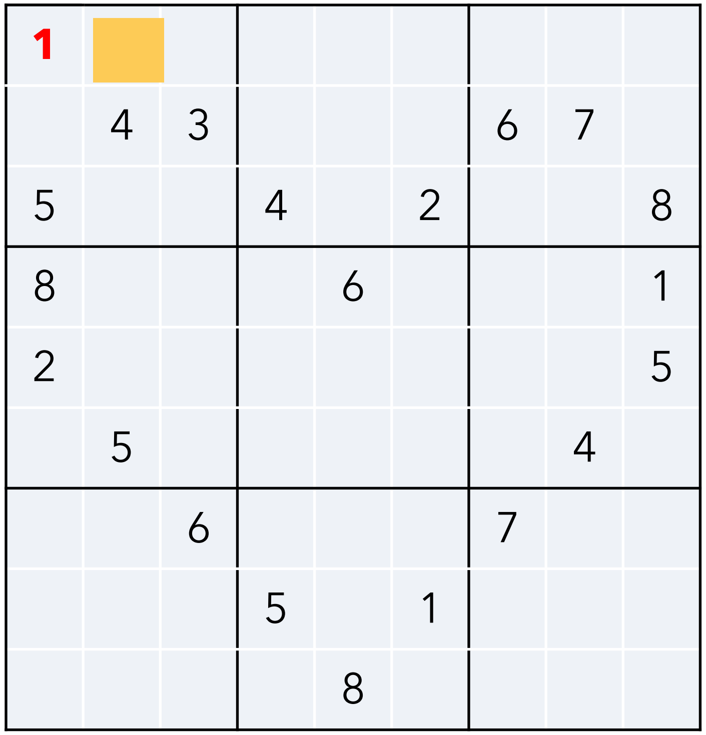
- pick an open entry
- try all possible numbers that meet constraints

How many next states?

What are they?

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Sudoku



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

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

2, 6, 7, 8, 9

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Sudoku

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

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

[✓ 2], 6, 7, 8, 9

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Sudoku

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

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

2, 6, 7, 8, 9

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Sudoku

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

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

How many next states?

What are they?

Sudoku

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

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

7, 8, 9

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Sudoku

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

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

[✓ 7], 8, 9

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Sudoku

1	2	**7**						
🟨	4	3				6	7	
5			4		2			8
8				6				1
2								5
	5						4	
		6				7		
			5		1			
				8				

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

How many next states?

What are they?

Sudoku

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

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

9

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Sudoku

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

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

How many next states?

What are they?

Sudoku

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

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

6

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Sudoku

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

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

Now what?

Sudoku

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

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

Now what?

Try another branch, i.e. go back to a place where we had a decision and try a different one

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Sudoku

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

Generate next states:

- pick an open entry
- try all possible numbers that meet constraints

7, [✓ 8], 9

- Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

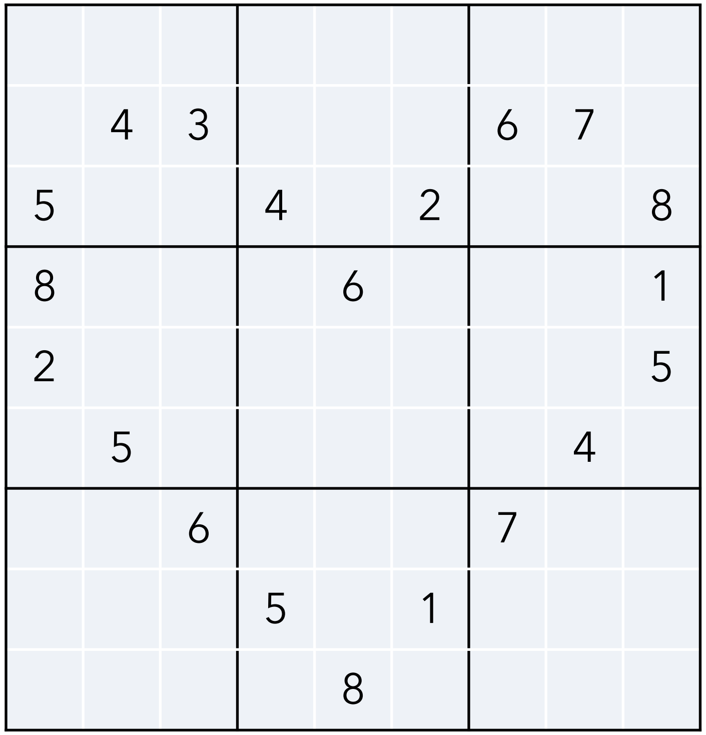
Best first Sudoku search

- DFS and BFS will choose entries (and numbers within those entries) randomly.
- Is that how people do it?
- How do you do it?
- Heuristics for best first search?

Best first Sudoku search

- DFS and BFS will choose entries (and numbers within those entries) randomly.
- Pick the entry that is MOST constrained
- People often try and find entries where only one option exists and only fill it in that way (very little search)

Representing the Sudoku board



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

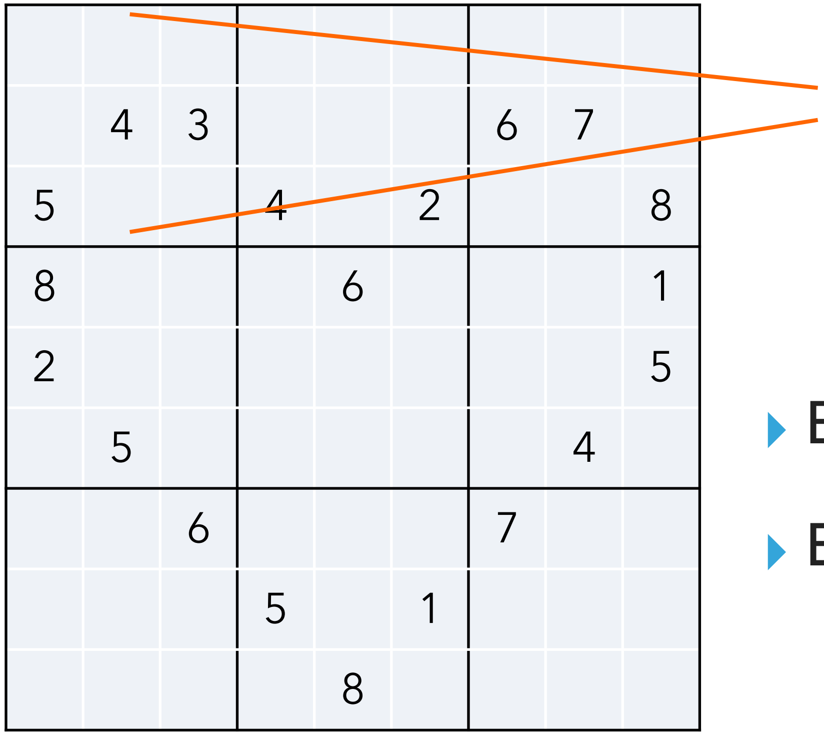
SudokuEntry: Class to keep track of an entry on the sudoku board.

Keep track of whether or not a value has been placed in that entry (fixed).

If one hasn't, keep track of what numbers are still valid.

- Board is a matrix (list of lists)

Representing the Sudoku board



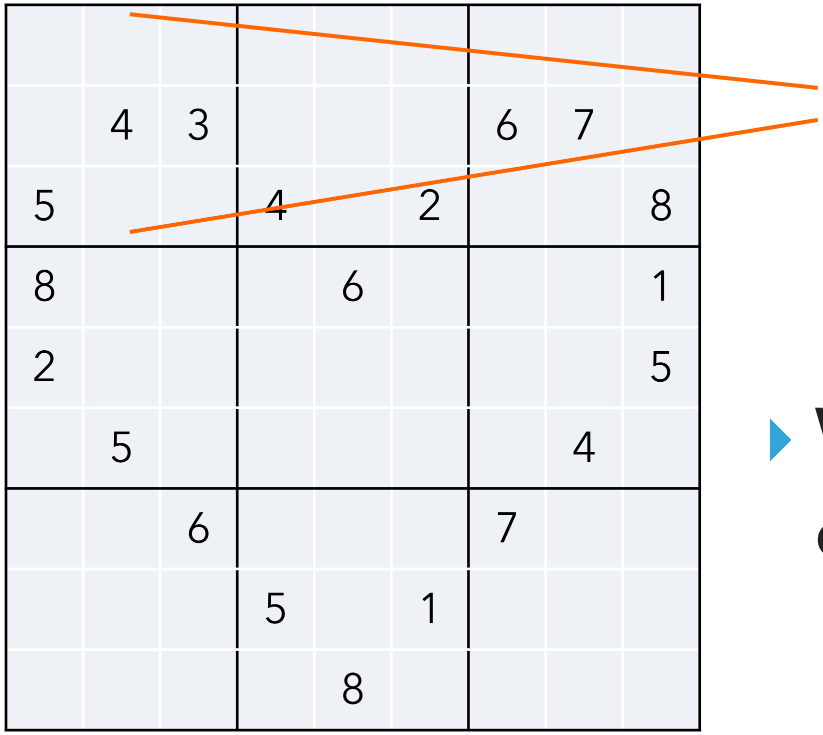
	4	3				6	7	
5			4		2			8
8				6				1
2								5
	5						4	
		6				7		
			5		1			
				8				

```
[1, 6, 7, 9], [1, 2, 6, 7, 8, 9], [1, 2, 7, 8, 9],
[1, 9],        4,                  3,
5,             [1, 6, 7, 9],       [1, 7, 9]
```

- Board will consist of 81 SudokuEntrys
- Each quadrant will have 9 entries.

- Board is a matrix (list of lists)
- Each entry is a SudokuEntry:
  - "fixed" if it has a number place
  - if not, then keeps a list of values still available

Representing the Sudoku board



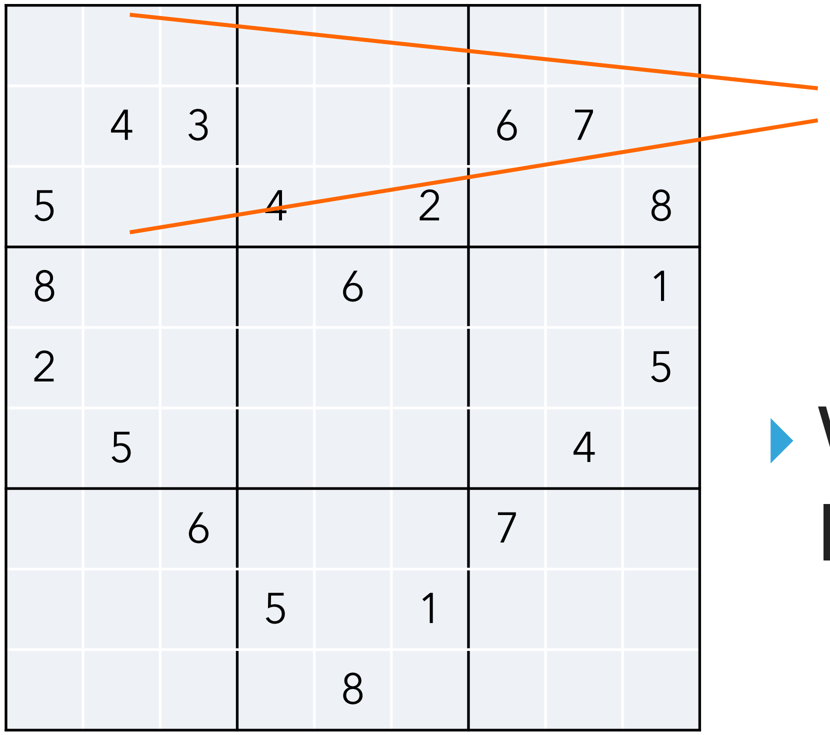
	4	3				6	7	
5			4		2			8
8				6				1
2								5
	5						4	
		6				7		
			5		1			
				8				

[1, 6, 7, 9], [1, 2, 6, 7, 8, 9], [1, 2, 7, 8, 9],
[1, 9], 4, 3,
5, [1, 6, 7, 9], [1, 7, 9]

- Which one is the most constrained of the ones above?

- Board is a matrix (list of lists)
- Each entry is a SudokuEntry:
  - "fixed" if it has a number place
  - if not, then keeps a list of values still available

Representing the Sudoku board

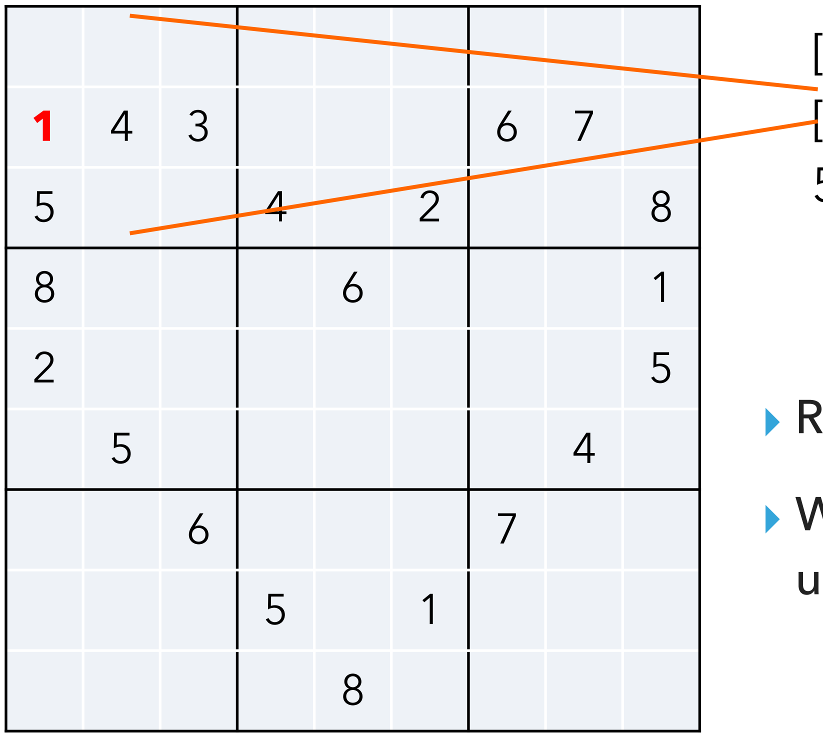


```
[1, 6, 7, 9], [1, 2, 6, 7, 8, 9], [1, 2, 7, 8, 9],
[1, 9],        4,                 3,
5,             [1, 6, 7, 9],      [1, 7, 9]
```

- What would the state look like if we pick 1?

- Board is a matrix (list of lists)
- Each entry is a SudokuEntry:
    - "fixed" if it has a number place
    - if not, then keeps a list of values still available

Representing the Sudoku board

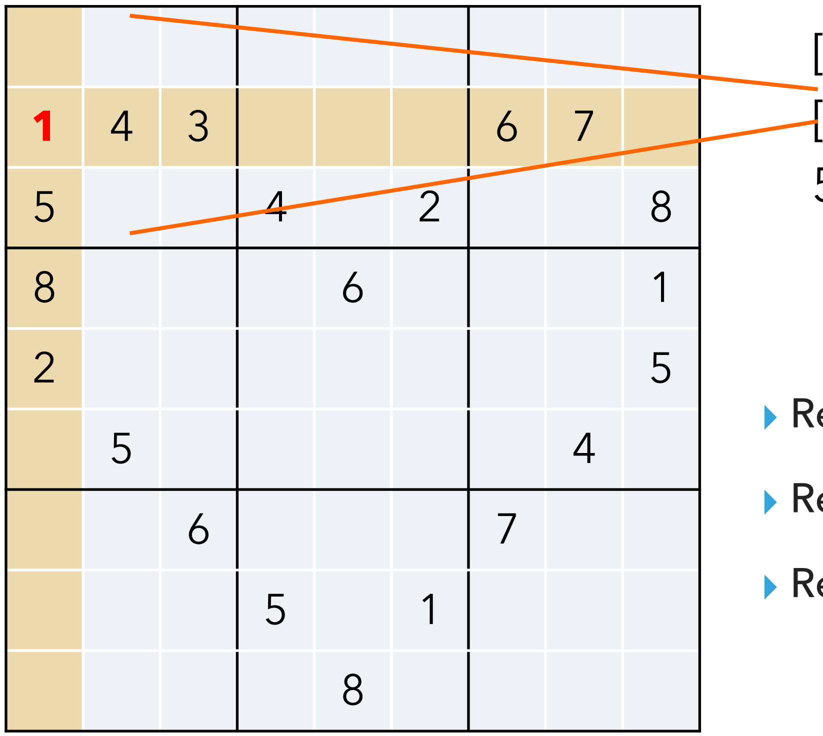


[6, 7, 9], [2, 6, 7, 8, 9], [2, 7, 8, 9],
[9], 4, 3,
5, [6, 7, 9], [7, 9]

- Remove 1 from all entries in the quadrant.
- What other parts of the board need to be updated?

- Board is a matrix (list of lists)
- Each entry is a SudokuEntry:
  - "fixed" if it has a number place
  - if not, then keeps a list of values still available

Representing the Sudoku board



[6, 7, 9], [2, 6, 7, 8, 9], [2, 7, 8, 9],
[9], 4, 3,
5, [6, 7, 9], [7, 9]

- Remove 1 from all entries in the quadrant
- Remove 1 from all entries in the same column
- Remove 1 from all entries in the same row

- Board is a matrix (list of lists)
- Each entry is a SudokuEntry:
    - "fixed" if it has a number place
    - if not, then keeps a list of values still available

TODAY'S LECTURE IN A NUTSHELL

Lecture 20: Informed and adversarial search

- Informed search
- Adversarial search

Why should we study games?

- Clear success criteria
- Important historically for AI
- Fun ☺
- Good application of search
    - hard problems (chess 35100 states in search space, 1040 legal states)
- Real-world problems fit this model
    - game theory (economics)
    - multi-agent problems

Types of games: game properties

- single-player vs. 2-player vs. multiplayer
- Fully observable (perfect information) vs. partially observable
- Discrete vs. continuous
- real-time vs. turn-based
- deterministic vs. non-deterministic (chance)

Strategic thinking =? Intelligence

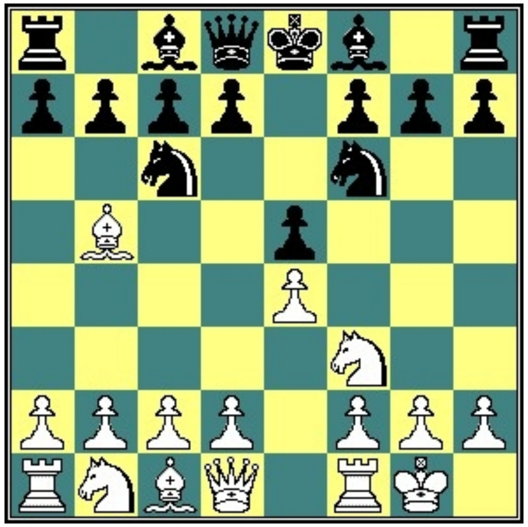
- Important question: Is strategic thinking the same as intelligence?
- For reasons previously stated, two-player games have been a focus of AI since its inception…

Strategic thinking =? Intelligence

- Humans and computers have different relative strengths in these games:

humans

?



computers

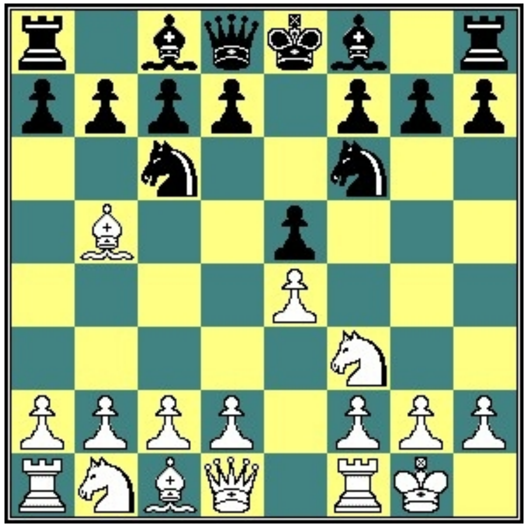
?

Strategic thinking =? Intelligence

- Humans and computers have different relative strengths in these games:

humans

good at evaluating the strength of a board for a player



computers

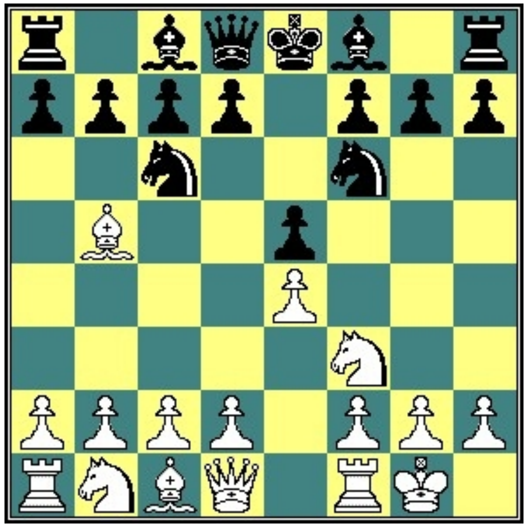
good at looking ahead in the game to find winning combinations of moves

Strategic thinking =? Intelligence

- How could you figure out how humans approach playing chess?

humans

good at evaluating the strength of a board for a player



How humans play games…

- In 1940s, psychologist de Grout performed a seminal experiment in which chess boards were shown to novice and expert players…
- experts could reconstruct the layouts perfectly
- novice players did far worse…

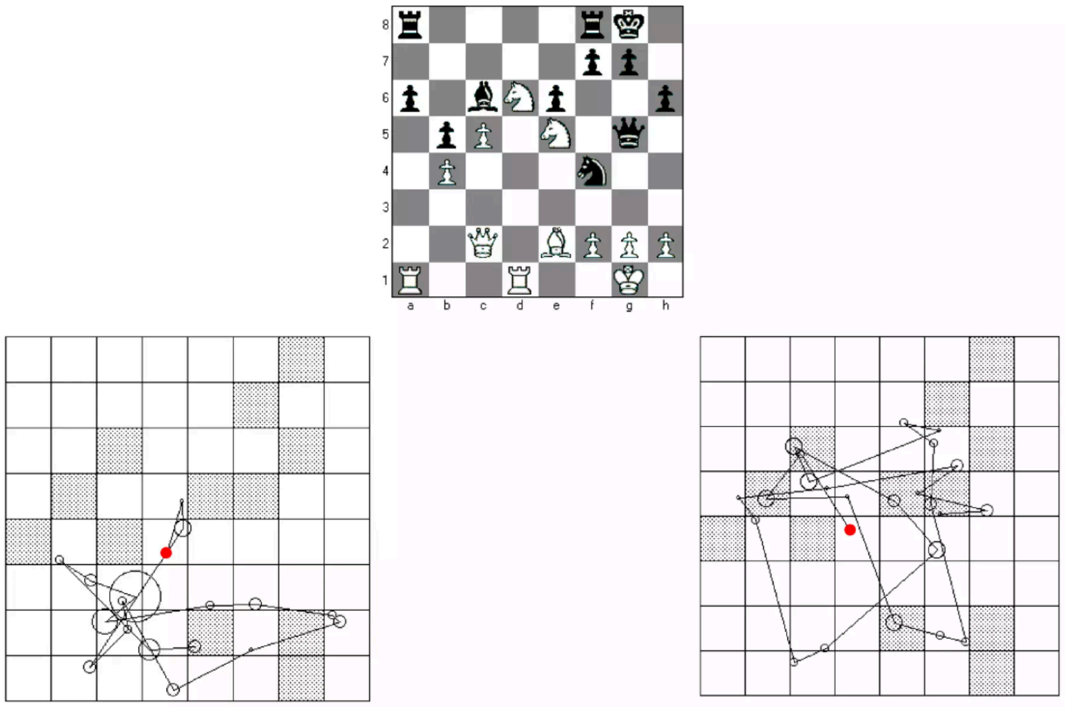
https://www.wired.com/2010/11/the-cognitive-cost-of-expertise/

ADVERSARIAL SEARCH

How humans play games…

- Random chess positions (not legal ones) were then shown to the two groups.
- experts and novices did just as badly at reconstructing them!
- If experts had enhanced memories, then the location shouldn't matter: a pawn was still a pawn.
  - Perception based on "chunking".

How humans play games…



Novice	Fixation duration	Master
310 msec	*mean*	260 msec
140 msec	*sd*	100 msec

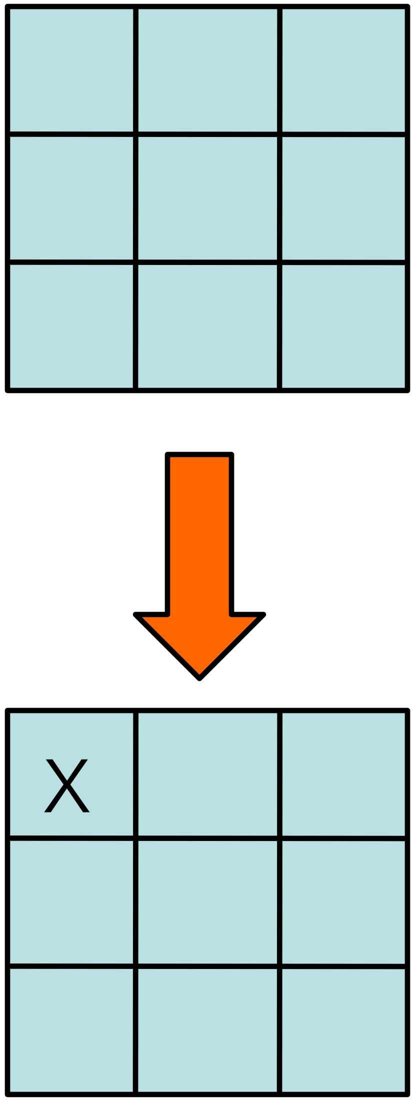
https://www.slideshare.net/kcfe/what-does-research-into-chess-expertise-tell-us-about-education-gobet

Back to tic tac toe

- If we want to write a program to play tic tac toe, what question are we trying to answer?
- Given a state (i.e. board configuration), what move should we make!

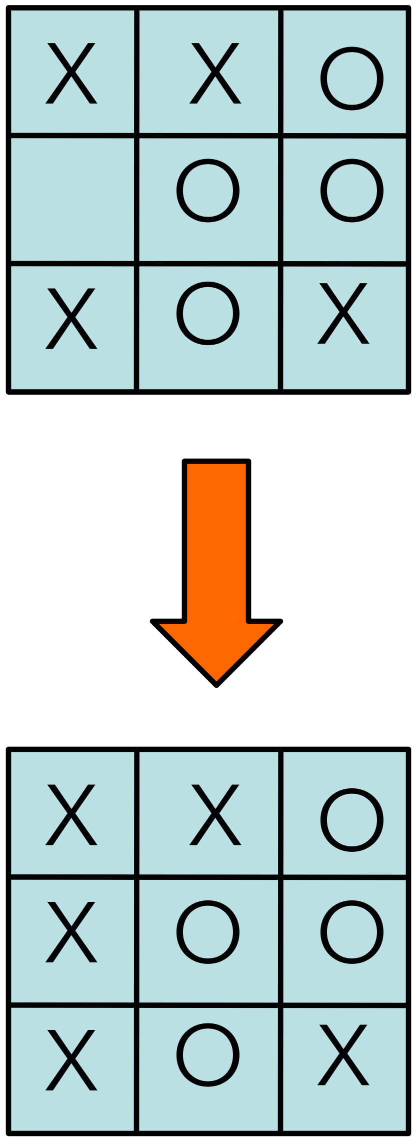
ADVERSARIAL SEARCH

Tic tac toe as search

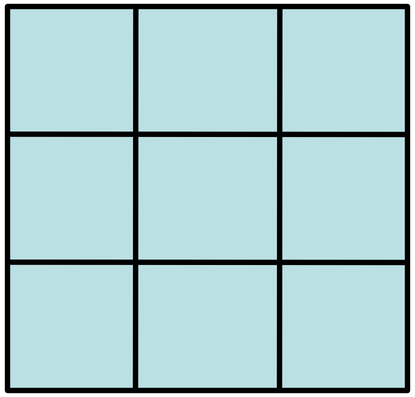


ADVERSARIAL SEARCH

Tic tac toe as search

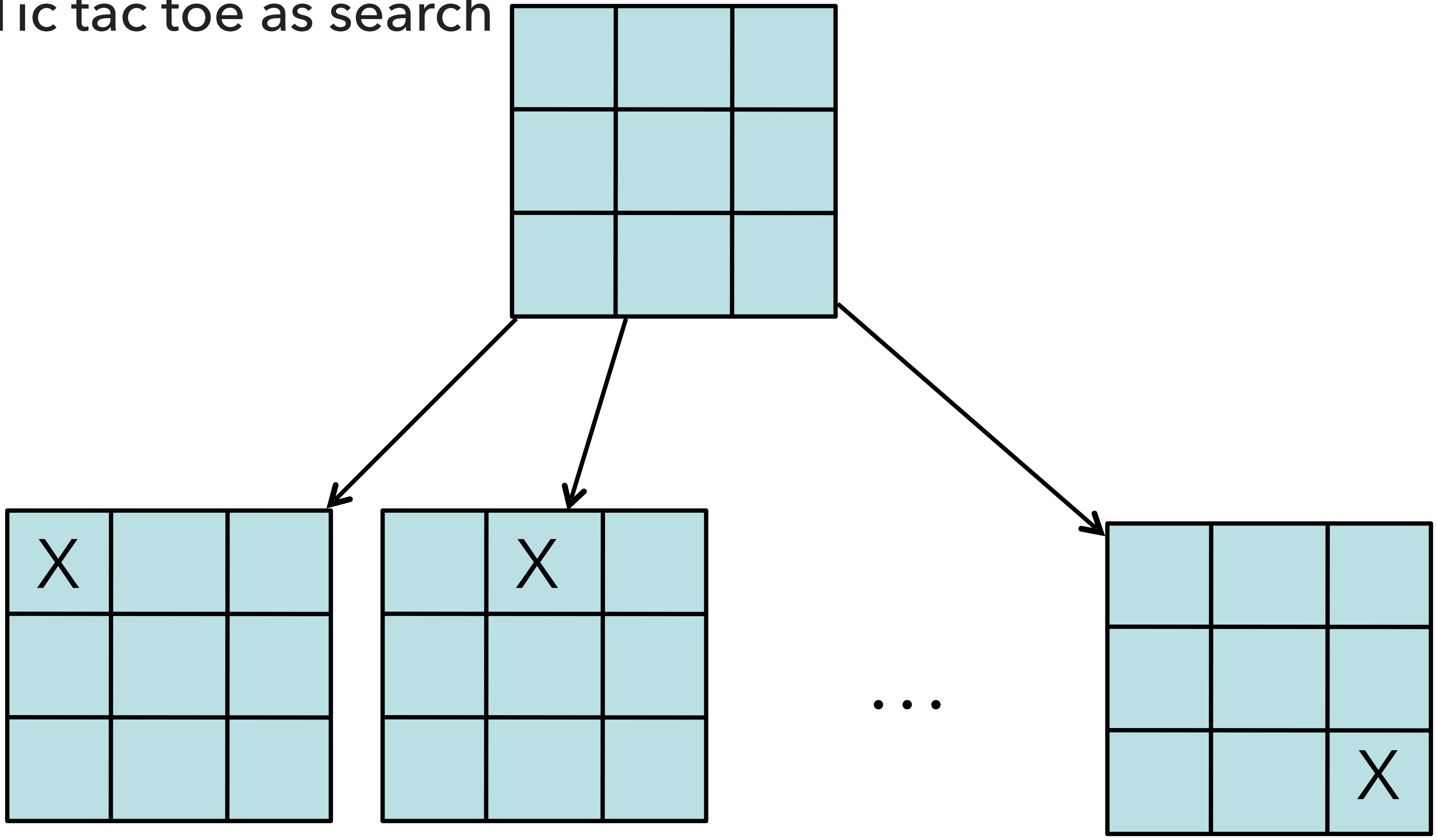


Tic tac toe as search

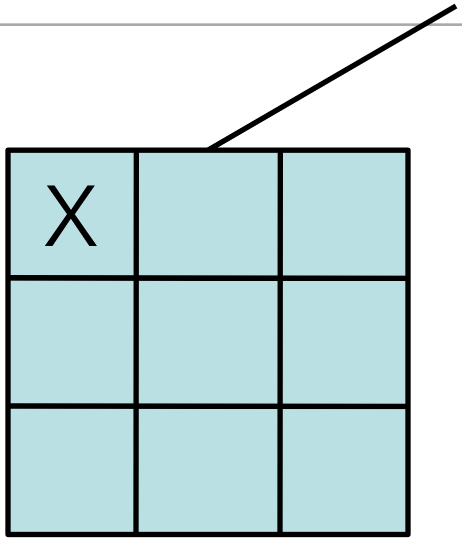


If we want to write a program to play tic tac toe, what question are we trying to answer?

Tic tac toe as search



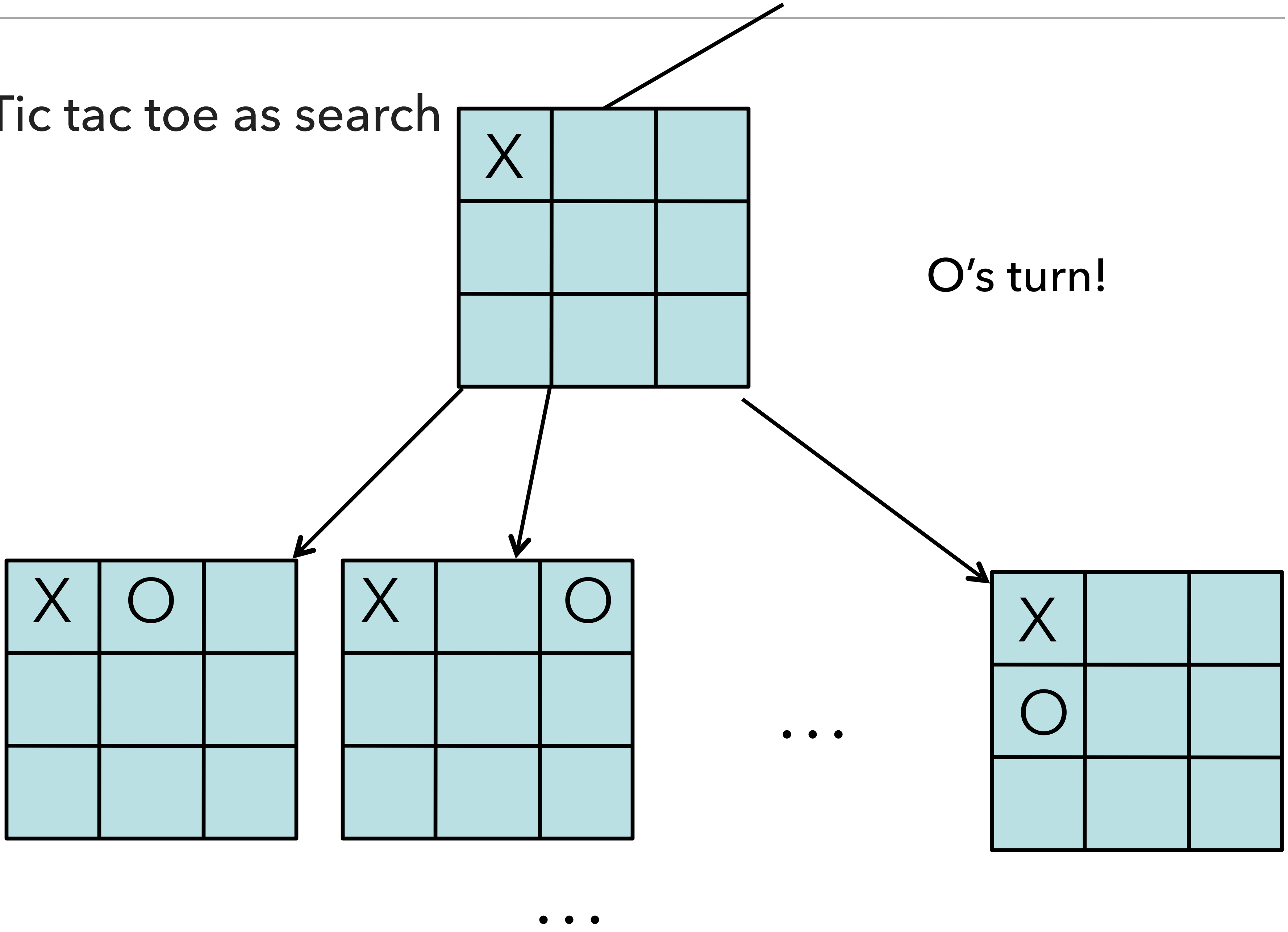
Tic tac toe as search



Now what?

ADVERSARIAL SEARCH

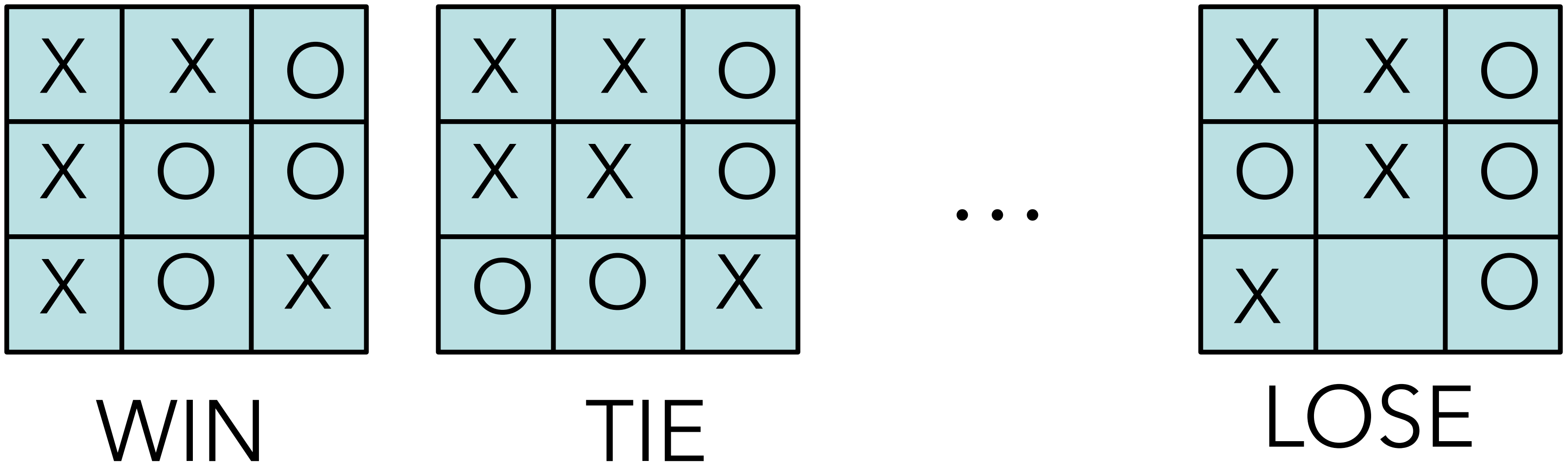
Tic tac toe as search



ADVERSARIAL SEARCH

Tic tac toe as search

Eventually, we'll get to a leaf



How does this help us?

Try and make moves that move us towards a win, i.e. where there are leaves with a WIN.

ADVERSARIAL SEARCH

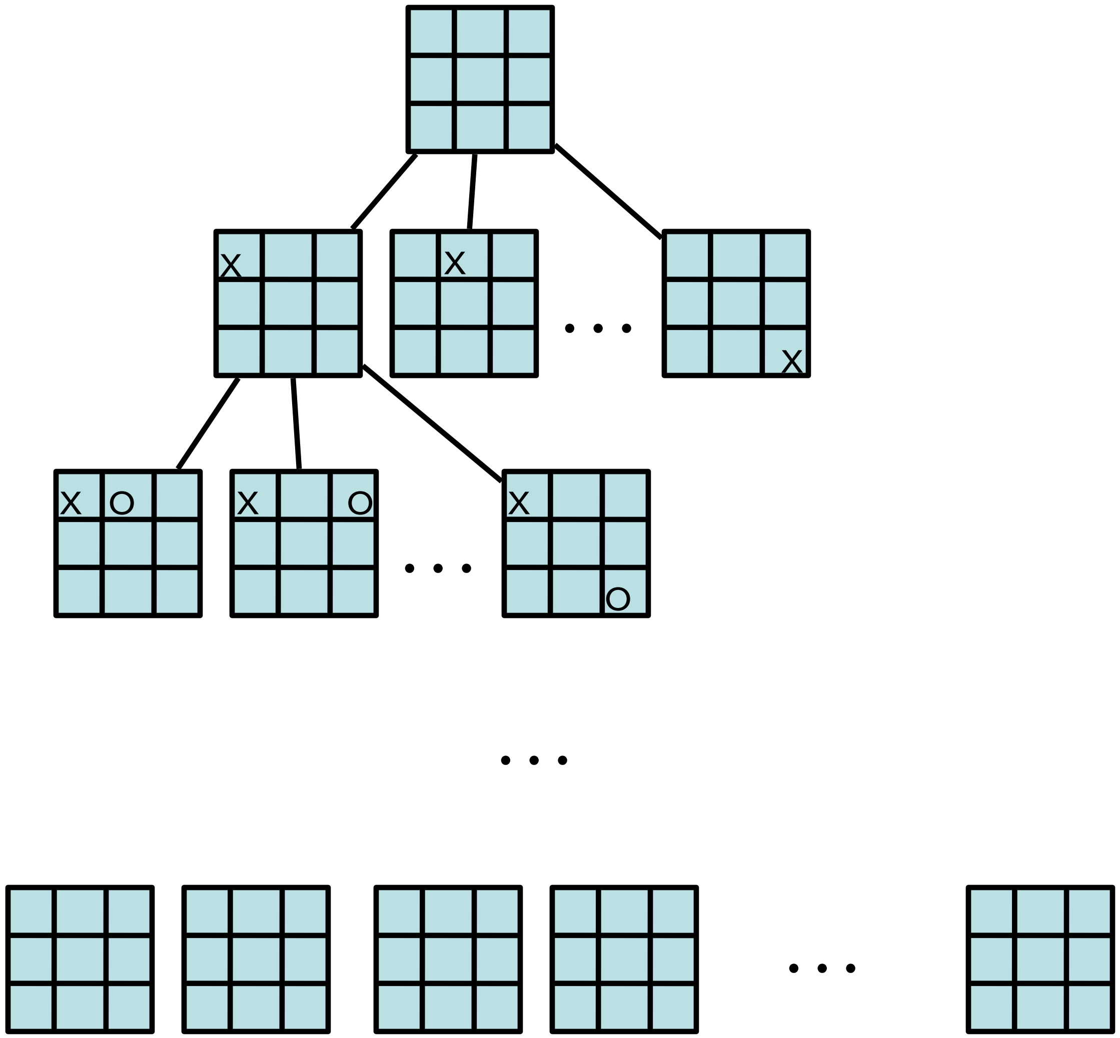
Tic tac toe as search

X's turn

O's turn

X's turn

…

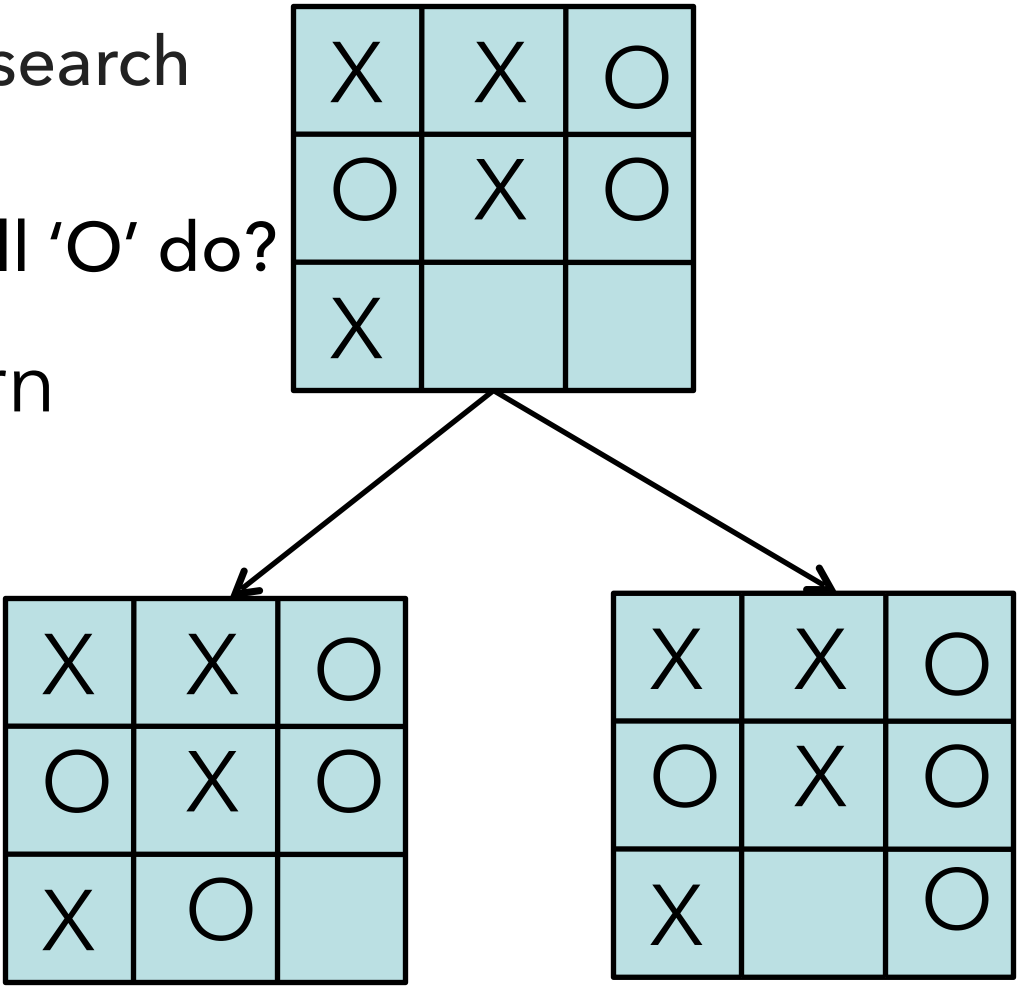


Problem: we don't know what O will do

Tic tac toe as search

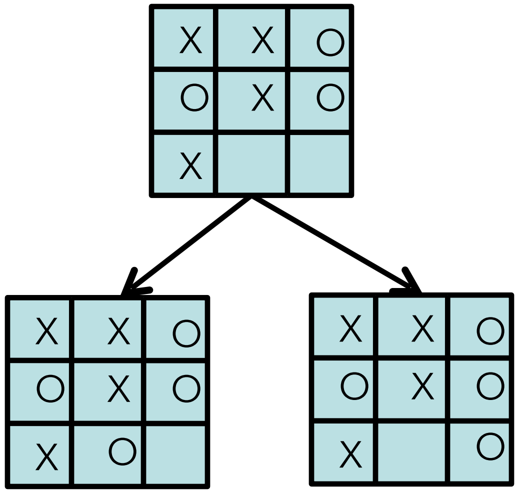
I'm X, what will 'O' do?

O's turn



Minimizing risk

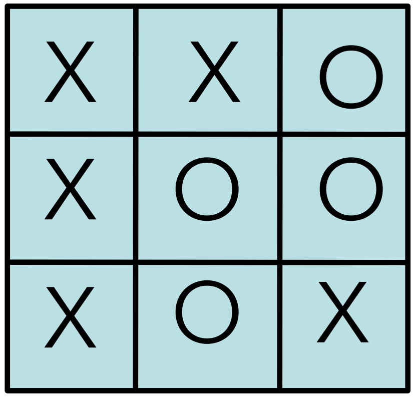
- The computer doesn't know what move O (the opponent) will make.
- It can assume that it will try and make the best move possible.
- Even if O actually makes a different move, we're no worse off. Why?



Optimal strategy

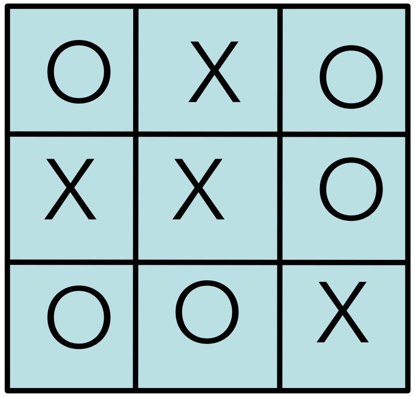
- An optimal strategy is one that is at least as good as any other, no matter what the opponent does.
  - If there's a way to force the win, it will
  - Will only lose if there's no other option

Defining a scoring function



WIN

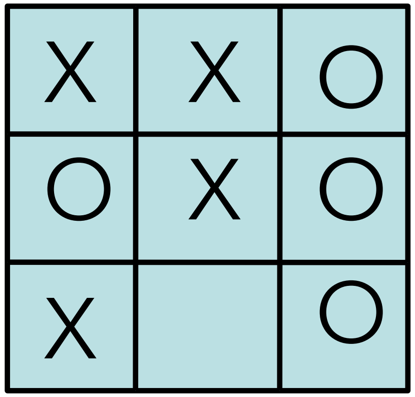
+1



TIE

0

. . .



LOSE

-1

- Idea:
  - define a function that gives us a "score" for how good each state is
  - higher scores mean better

Defining a scoring function

Our (X) turn

X	X	O
	O	O
X	O	X

What should be the score of this state?

Defining a scoring function

Our (X) turn

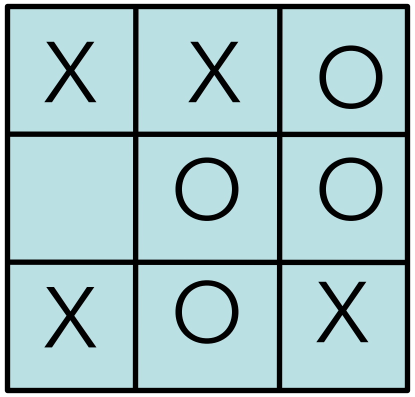
X	X	O
	O	O
X	O	X

What should be the score of this state?

+1: we can get to a win

Defining a scoring function

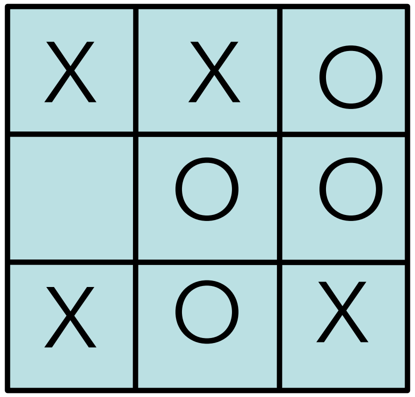
Opponent's (O) turn



What should be the score of this state?

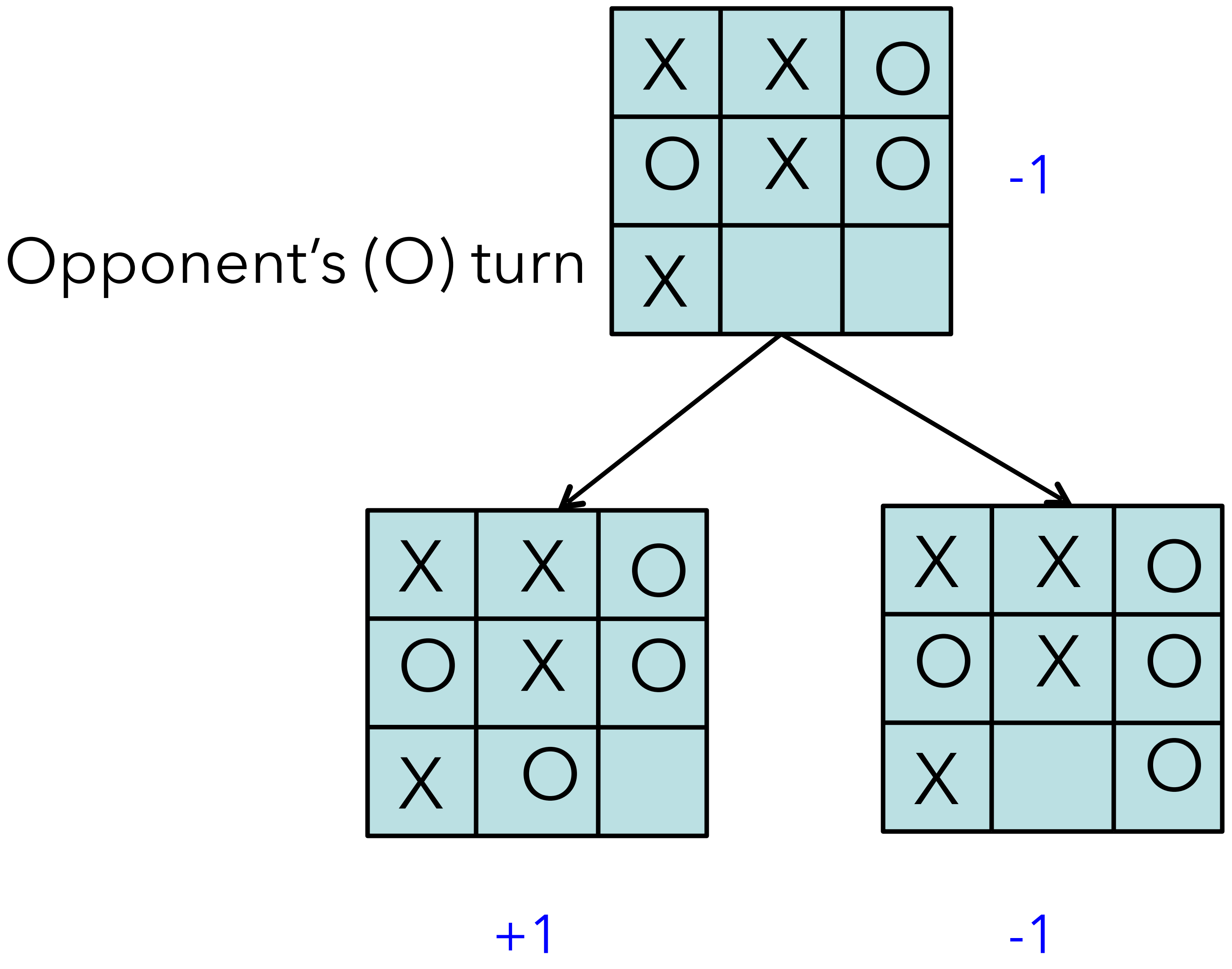
Defining a scoring function

Opponent's (O) turn



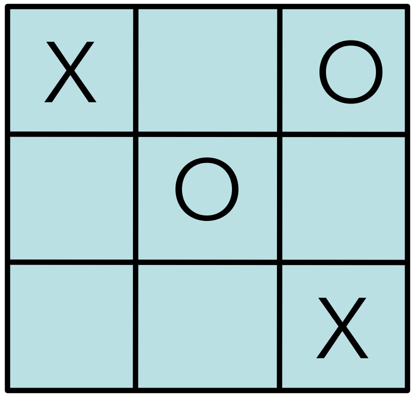
What should be the score of this state?

-1: opponent can get to a win


Opponent's (O) turn
X X O
O X O
X
-1
X X O
O X O
X O
+1
X X O
O X O
X O
-1

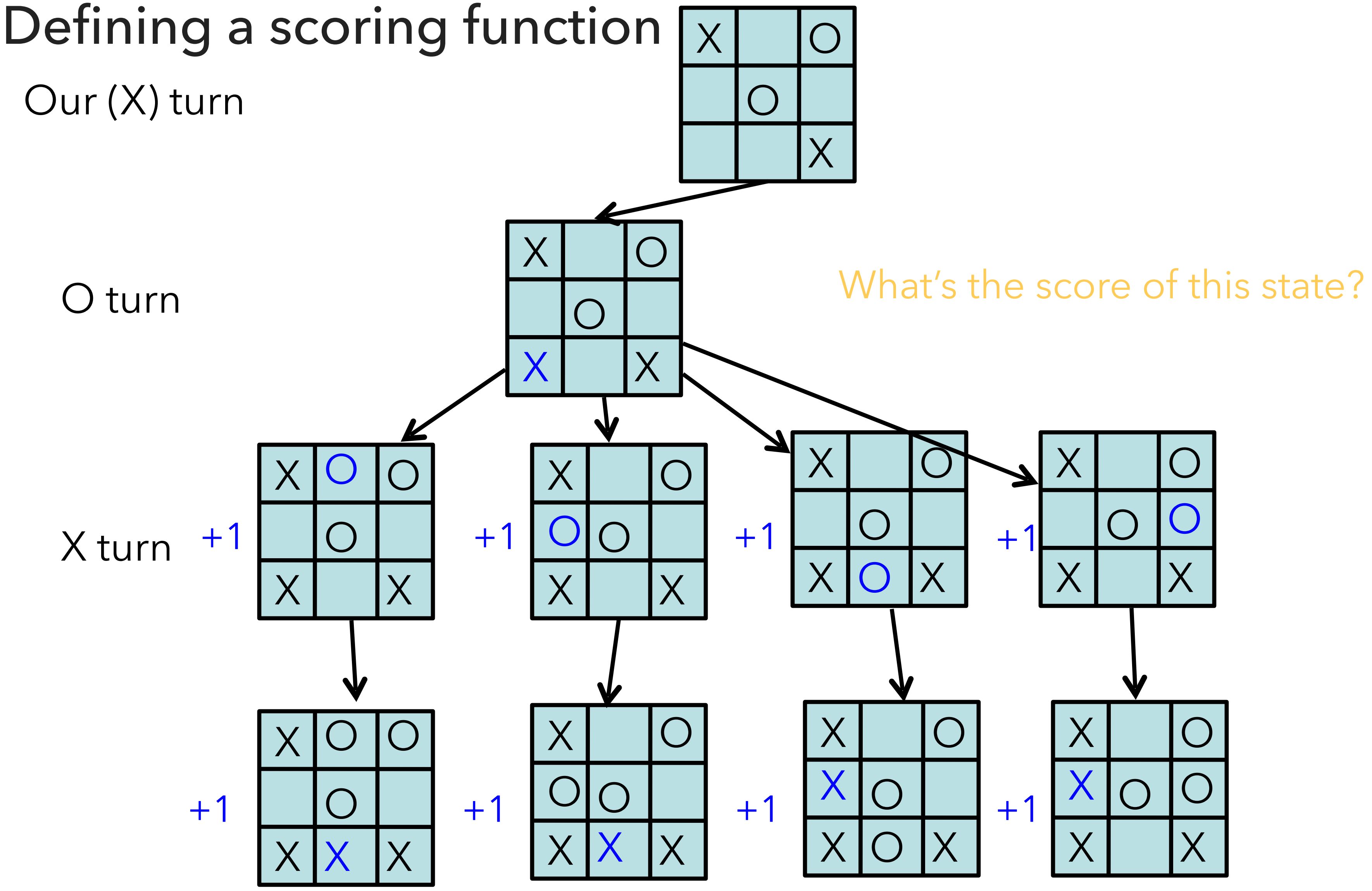
Defining a scoring function

Our (X) turn



What should be the score of this state?

Defining a scoring function

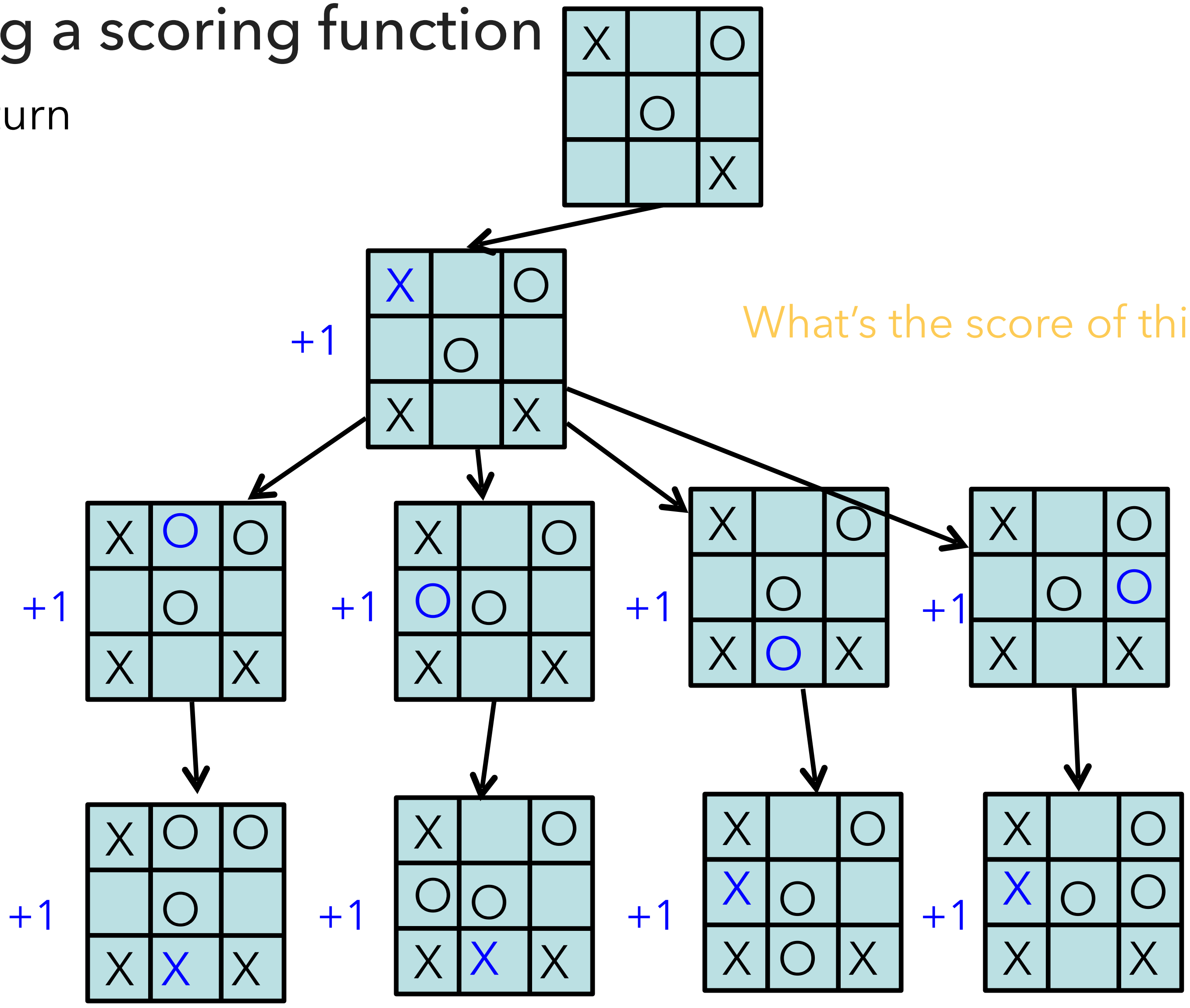


Defining a scoring function

Our (X) turn

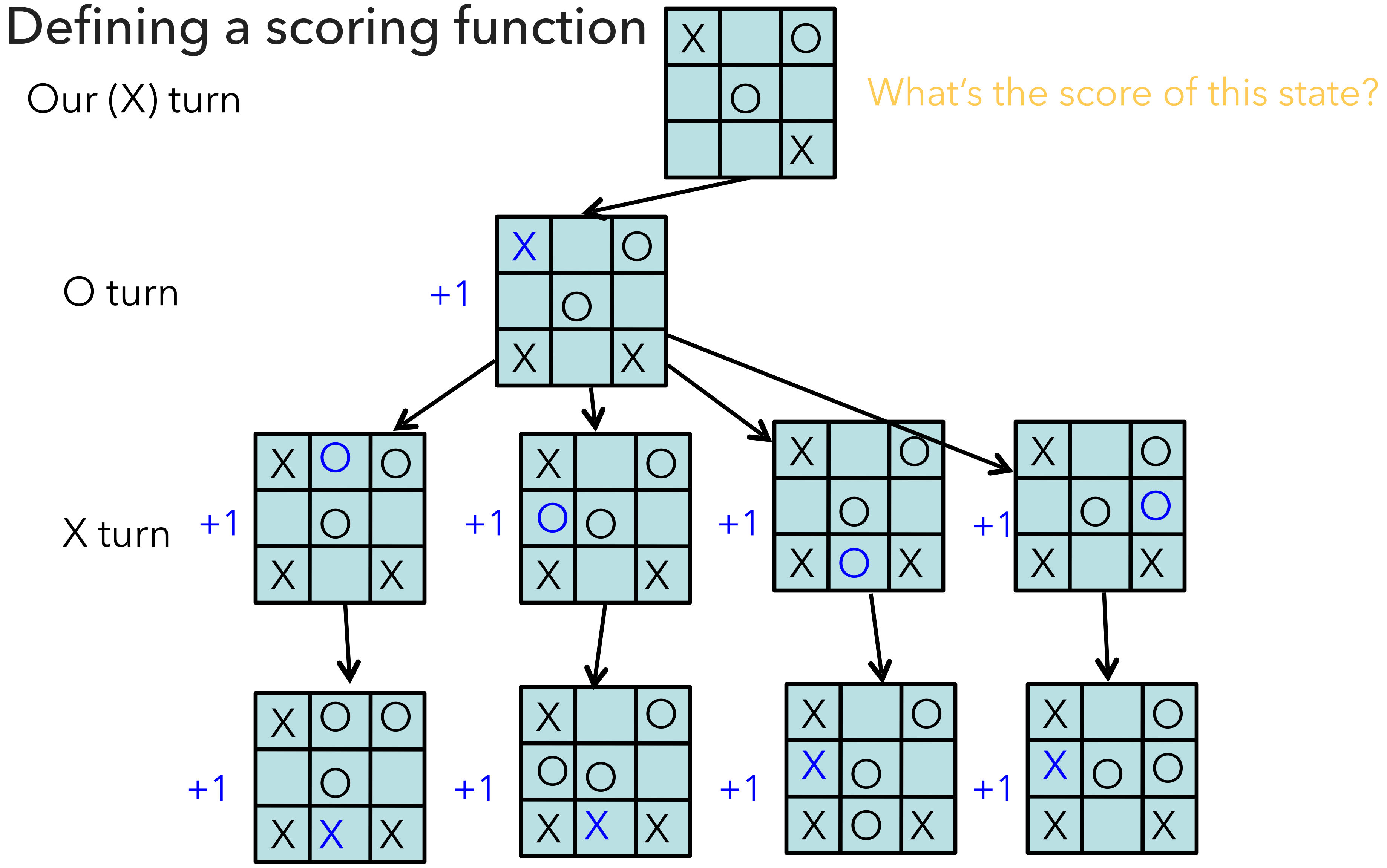
O turn

X turn

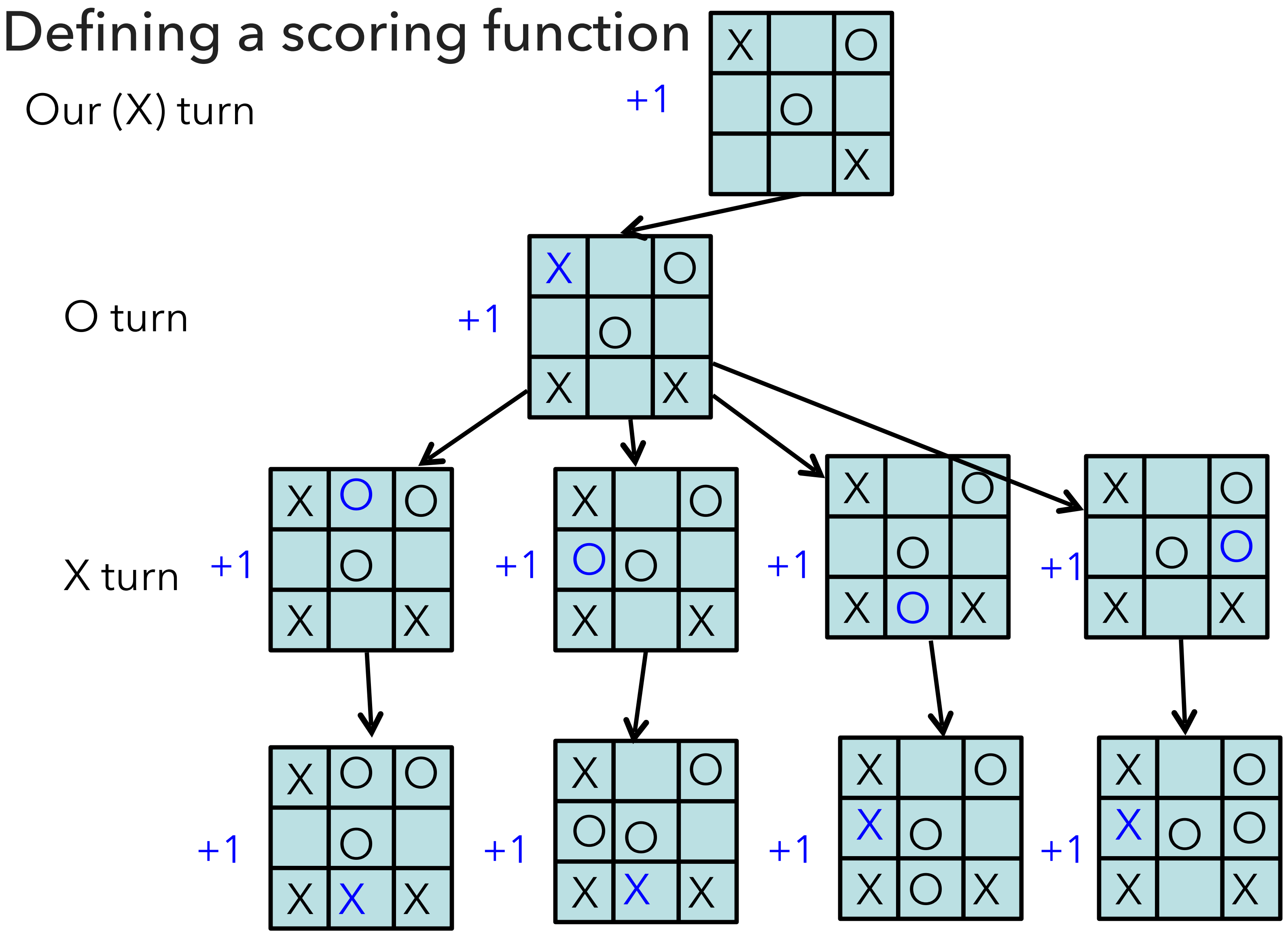


What's the score of this state?

Defining a scoring function



Defining a scoring function


Our (X) turn
O turn
X turn
+1
+1
+1
+1
+1
+1
+1
+1
+1
+1
X
O
O
X