INFORMED SEARCH

David Kauchak
CS30 – Spring 2015

Admin

Assignment 8… how did it go?

Assignment 9
- Out later today
- Due Sunday at 11:59 pm
- Lab tomorrow is a work session. Make sure to read the handout and ideally start before then.

Office hours this week are posted on course web page

Schedule

Midterm next Tuesday (4/14)
- In-class
- Will focus on material since the second midterm up through today's class
- Can use 2 pages of notes (like last time)
- I'll post practice problems (as soon as I can ☺)

Review session: Monday, 5-6pm (location TBA)

No lab next week (4/15)

Pre-registration: We'll talk about on Thursday

Other search problems

What problems have you seen that could be posed as search problems?

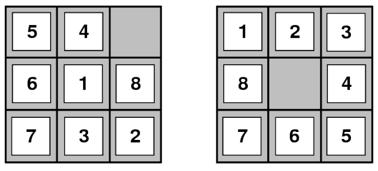
What is the state?

Start state

Goal state

State-space/transition between states

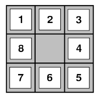
8-puzzle



Start State Goal State

8-puzzle

goal



Goal State

state representation?

start state?

state-space/transitions?

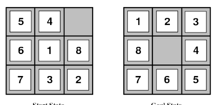
8-puzzle

state:

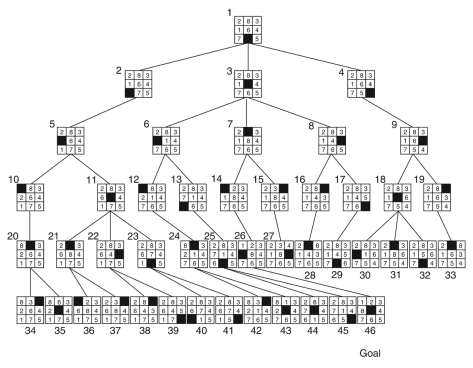
- all 3 x 3 configurations of the tiles on the board

transitions between states:

- Move Blank Square Left, Right, Up or Down.
- This is a more efficient encoding than moving each of the 8 distinct tiles



Start State Goal State



Cryptarithmetic

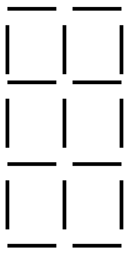
Find an assignment of digits (0, ..., 9) to letters so that a given arithmetic expression is true. examples:

SEND + MORE = MONEY

```
 FORTY
+  TEN
+  TEN
 -----
 SIXTY
F=2, O=9, R=7, etc.
```

Remove 5 Sticks

Given the following configuration of sticks, remove exactly 5 sticks in such a way that the remaining configuration forms exactly 3 squares.



Water Jug Problem

Given a full 5-gallon jug and a full 2-gallon jug, fill the 2-gallon jug with exactly one gallon of water.

Water Jug Problem

State = (x,y), where x is the number of gallons of water in the 5-gallon jug and y is # of gallons in the 2-gallon jug

Initial State = (5,2)

Goal State = (*,1), where * means any amount

Operator table

Name	Cond.	Transition	Effect
Empty5	–	(x,y)→(0,y)	Empty 5-gal. jug
Empty2	–	(x,y)→(x,0)	Empty 2-gal. jug
2to5	$x \leq 3$	(x,2)→(x+2,0)	Pour 2-gal. into 5-gal.
5to2	$x \geq 2$	(x,0)→(x-2,2)	Pour 5-gal. into 2-gal.
5to2part	$y < 2$	(1,y)→(0,y+1)	Pour partial 5-gal. into 2-gal.

8-puzzle revisited

How hard is this problem?

1	3	8
4		7
6	5	2

8-puzzle revisited

The average depth of a solution for an 8-puzzle is 22 moves

An exhaustive search requires searching ~3^{22} = 3.1 x 10^{10} states

- BFS: 10 terabytes of memory
- DFS: 8 hours (assuming one million nodes/second)

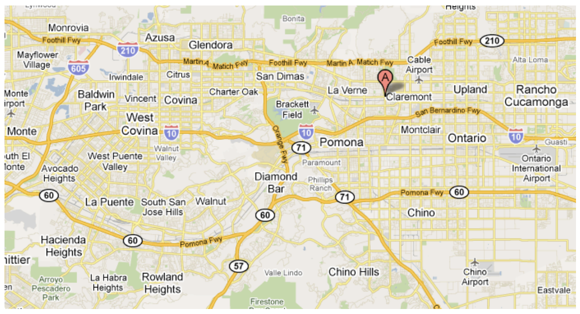
Can we do better?

Is DFS and BFS intelligent?

1	3	8
4		7
6	5	2

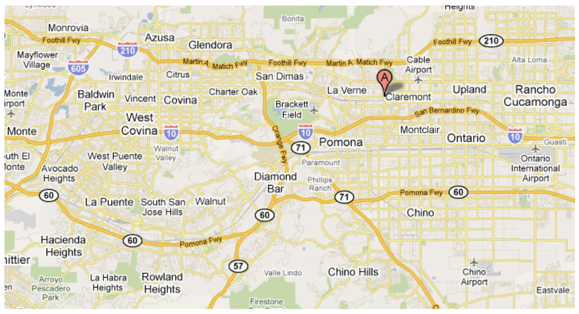
from: Claremont to:Rowland Heights

How do you think google maps does it?



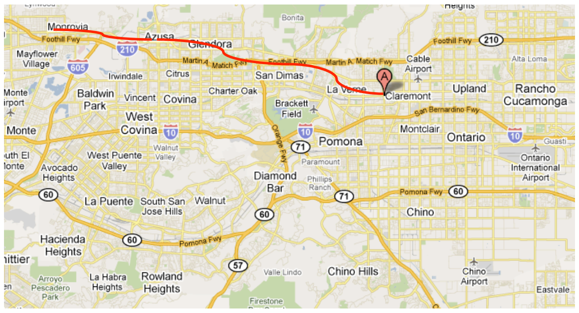
from: Claremont to:Rowland Heights

What would the search algorithms do?



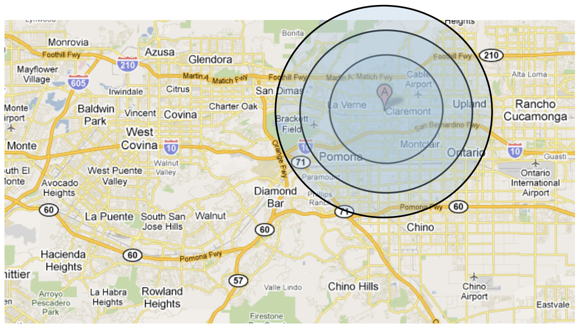
from: Claremont to:Rowland Heights

DFS



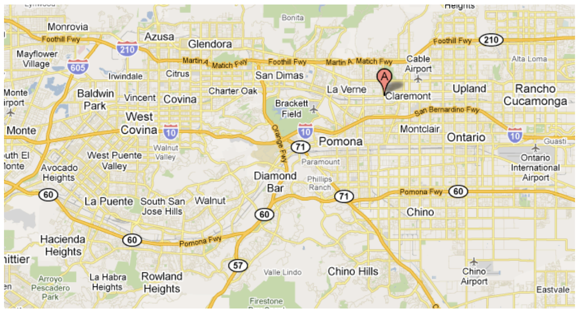
from: Claremont to:Rowland Heights

BFS



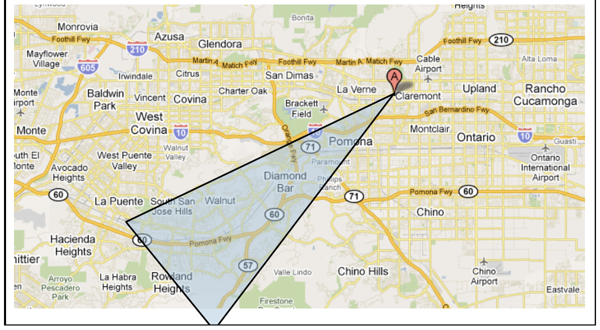
from: Claremont to: Rowland Heights

Ideas?



from: Claremont to: Rowland Heights

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

Merriam-Webster's Online Dictionary

Heuristic (pron. \hyu-ˈris-tik\): adj. [from Greek *heuriskein* to discover.] involving or serving as an aid to learning, discovery, or problem-solving by experimental and especially trial-and-error methods

The Free On-line Dictionary of Computing (2/19/13)

heuristic 1. Of or relating to a usually speculative formulation serving as a guide in the investigation or solution of a problem: "The historian discovers the past by the judicious use of such a heuristic device as the 'ideal type'" (Karl J. Weintraub).

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
- Missionaries and cannibals?
  - number of people on the starting bank

Two heuristics

2	8	3
1	6	4
	7	5

1	2	3
8	6	4
	7	5

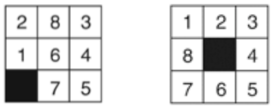
6	2	3
8		4
7	1	5

Which state is better?

1	2	3
8		4
7	6	5

GOAL

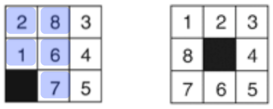
Two heuristics



Goal

How many tiles are out of place?

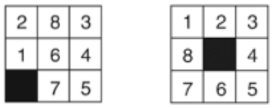
Two heuristics



Goal

5

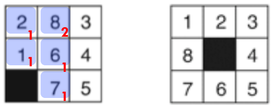
Two heuristics



Goal

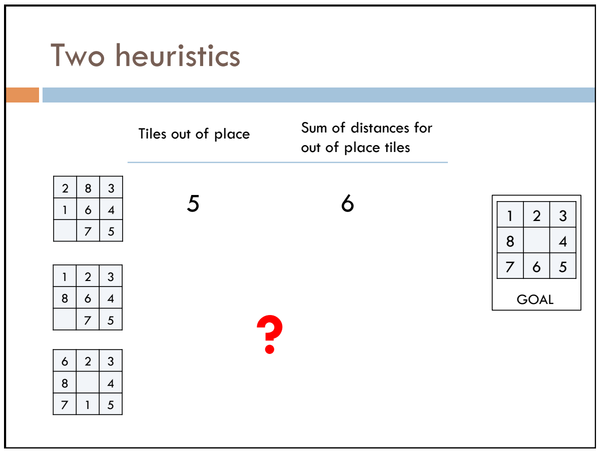
What is the “distance” of the tiles that are out of place?

Two heuristics



Goal

6



Two heuristics

2	8	3
1	6	4
	7	5

1	2	3
8	6	4
	7	5

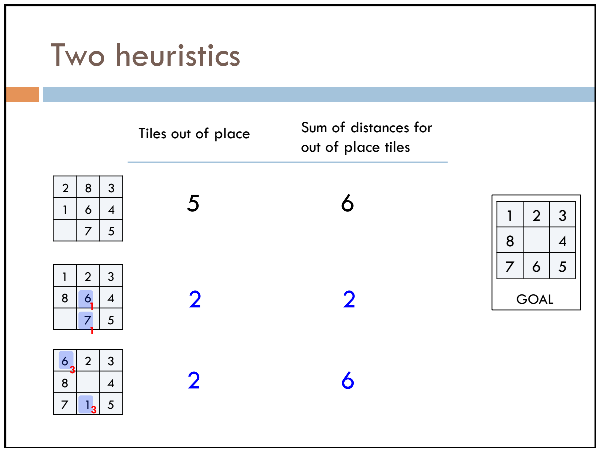
6	2	3
8		4
7	1	5

Tiles out of place	Sum of distances for out of place tiles
5	6

?

1	2	3
8		4
7	6	5

GOAL



Two heuristics

2	8	3
1	6	4
	7	5

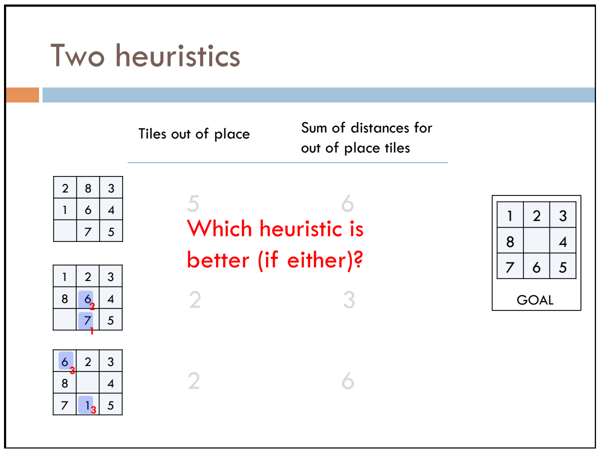
1	2	3
8	6_{1}	4
	7_{1}	5

6_{3}	2	3
8		4
7	1_{3}	5

Tiles out of place	Sum of distances for out of place tiles
5	6
2	2
2	6

1	2	3
8		4
7	6	5

GOAL



Two heuristics

2	8	3
1	6	4
	7	5

1	2	3
8	6_{2}	4
	7_{1}	5

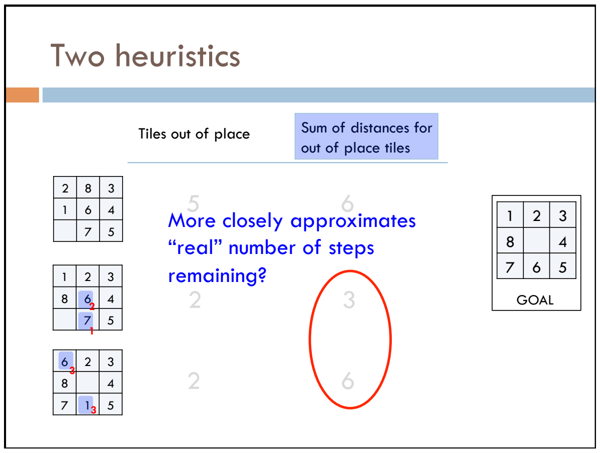
6_{3}	2	3
8		4
7	1_{3}	5

Tiles out of place	Sum of distances for out of place tiles
5	6
2	3
2	6

Which heuristic is better (if either)?

1	2	3
8		4
7	6	5

GOAL



Two heuristics

2	8	3
1	6	4
	7	5

1	2	3
8	6_{2}	4
	7_{1}	5

6_{3}	2	3
8		4
7	1_{3}	5

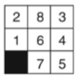
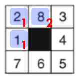
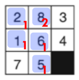
Tiles out of place	Sum of distances for out of place tiles
5	6
2	3
2	6

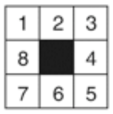
More closely approximates “real” number of steps remaining?

1	2	3
8		4
7	6	5

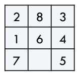
GOAL

Two heuristics

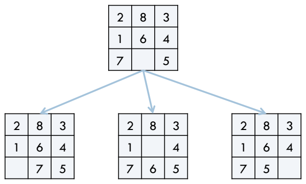
	Tiles out of place	Sum of distances for out of place tiles
	5	6
	3	4
	5	6



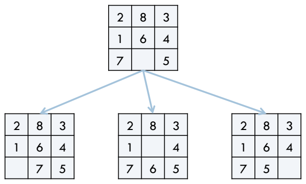
Goal



Next states?

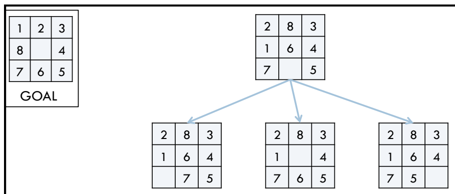


Which would you do?

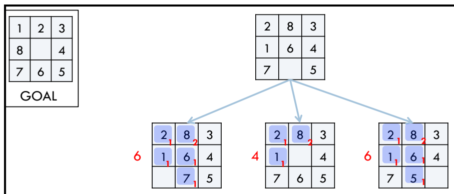


Which would DFS choose

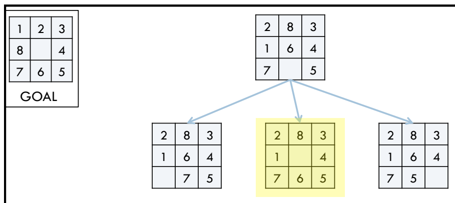
Completely depends on how next states are generated.
Not an "intelligent" decision!



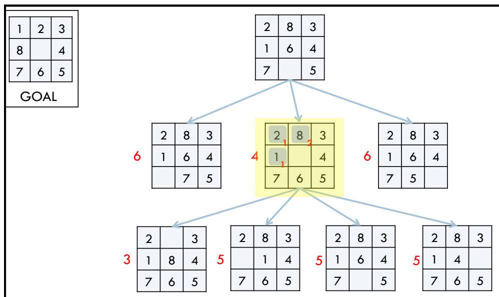
Best first search: out of place tiles?



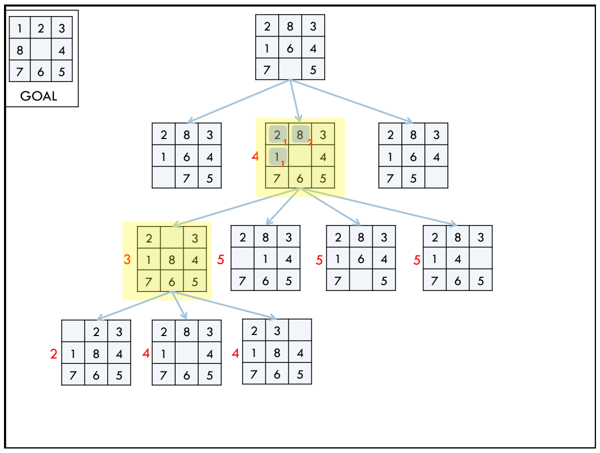
Best first search: distance of tiles?

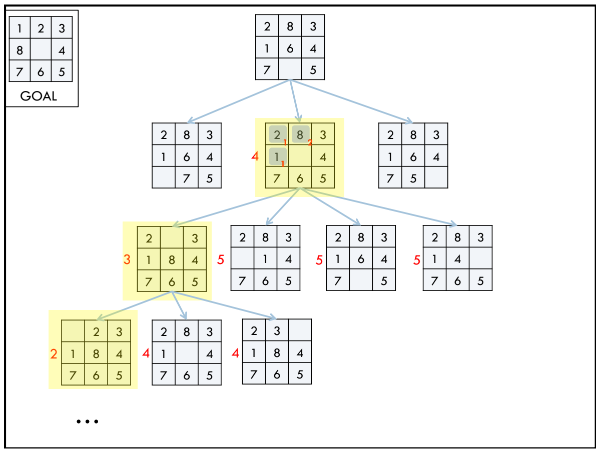


Next states?



Which next for best first search?





Informed search algorithms

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)

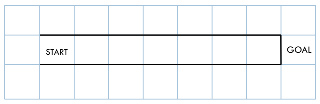
Informed search algorithms

Any other problems/concerns about best first search?

Informed search algorithms

Any other problems/concerns about best first search?

- Only as good as the heuristic function



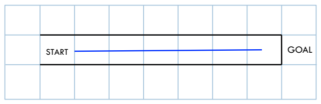
Best first search using distance as the crow flies as heuristic

What would the search do?

Informed search algorithms

Any other problems/concerns about best first search?

- Only as good as the heuristic function



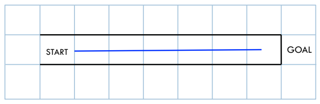
Best first search using distance as the crow flies as heuristic

What is the problem?

Informed search algorithms

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's come.
Best first search is a "greedy" algorithm

Informed search algorithms

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)

Sudoku

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

How can we pose this as a search problem?

State

Start state

Goal state

State space/transitions

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				

How can we pose this as a search problem?

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

Start state:

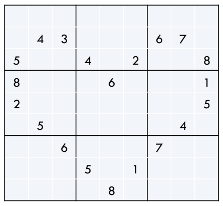
Goal state:

State space/transitions

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

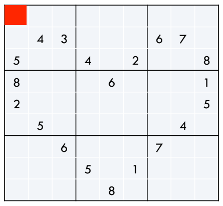


	4	3				6	7	
5			4		2			8
8				6				1
2								5
	5						4	
		6				7		
			5		1			
				8				

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



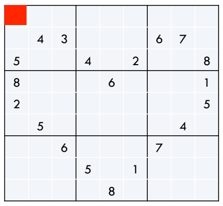
🟥								
	4	3				6	7	
5			4		2			8
8				6				1
2								5
	5						4	
		6				7		
			5		1			
				8				

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



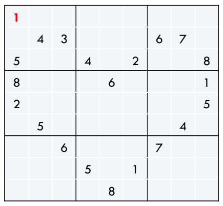
🟥								
	4	3				6	7	
5			4		2			8
8				6				1
2								5
	5						4	
		6				7		
			5		1			
				8				

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

[✓ 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	**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

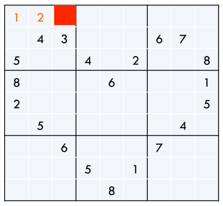
1	2	🟥						
	4	3				6	7	
5			4		2			8
8				6				1
2								5
	5						4	
		6				7		
			5		1			
				8				

Generate next states:
- pick an open entry
- try all possible numbers that meet constraints

What are the next states?

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

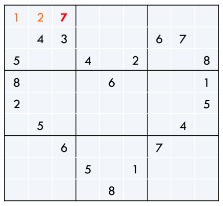


Generate next states:
- pick an open entry
- try all possible numbers that meet constraints

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

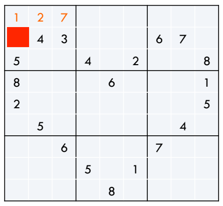


Generate next states:
- pick an open entry
- try all possible numbers that meet constraints

[✓ 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

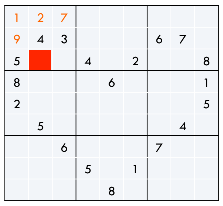


Generate next states:
- pick an open entry
- try all possible numbers that meet constraints

7, 8, 9

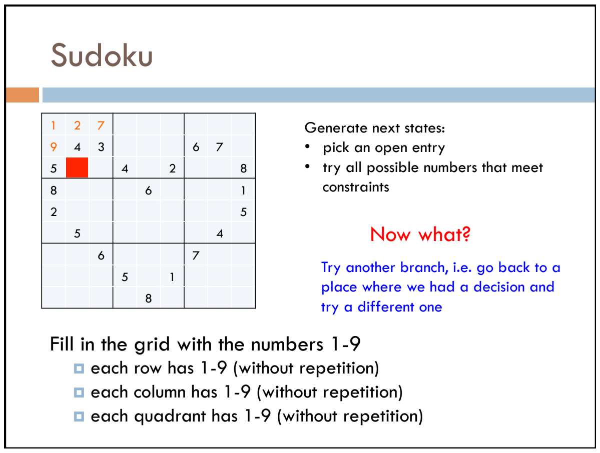
Fill in the grid with the numbers 1-9
- each row has 1-9 (without repetition)
- each column has 1-9 (without repetition)
- each quadrant has 1-9 (without repetition)

Sudoku



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

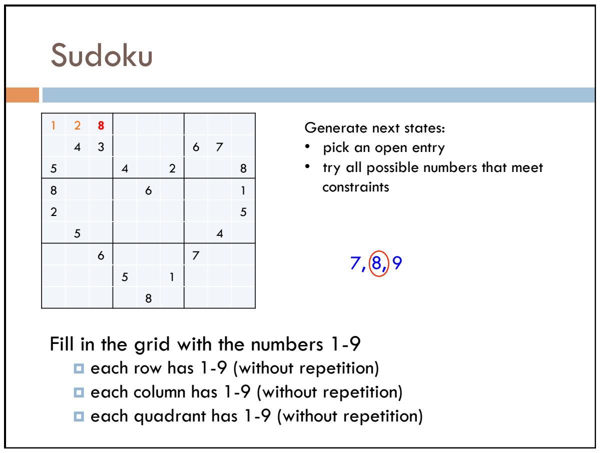
1	2	7						
9	4	3				6	7	
5	🟥		4		2			8
8				6				1
2								5
	5						4	
		6				7		
			5		1			
				8				

Generate next states:
- pick an open entry
- try all possible numbers that meet constraints

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)



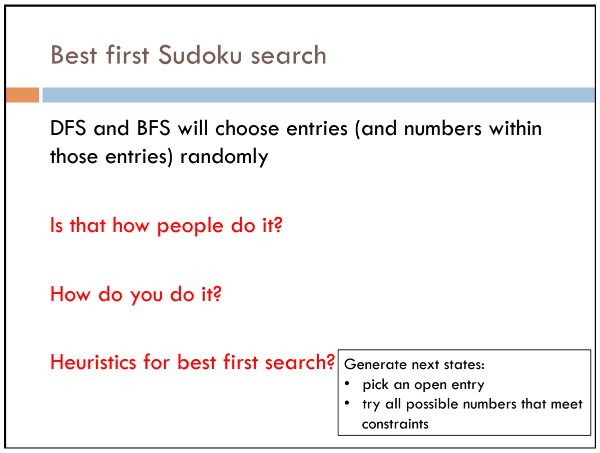
Sudoku

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

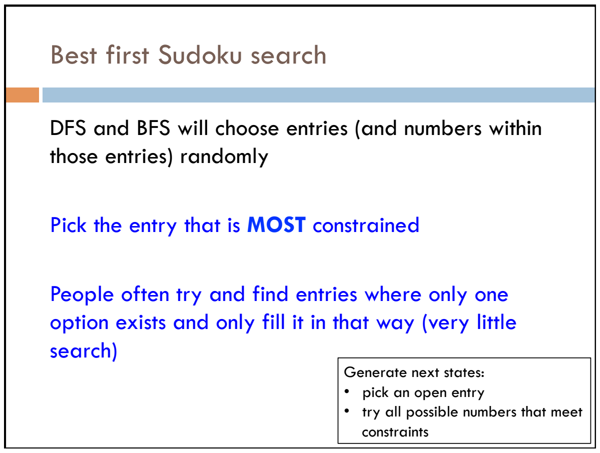
DFS and BFS will choose entries (and numbers within those entries) randomly

Is that how people do it?

How do you do it?

Heuristics for best first search?

Generate next states:
- pick an open entry
- try all possible numbers that meet constraints



Best first Sudoku search

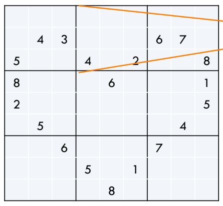
DFS and BFS will choose entries (and numbers within those entries) randomly

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)

Generate next states:
- pick an open entry
- try all possible numbers that meet constraints

Representing the Sudoku board

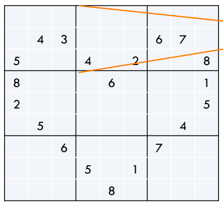


[1, 6, 7, 9], [1, 2, 6, 7, 8, 9], [1, 2, 7, 8, 9],
[1, 9], 4, 3,
5, [1, 6, 7, 9], [1, 7, 9]

Which is the most constrained (of the ones above)?

- Board is a matrix (list of lists)
- Each entry is *either*:
  - a number (if we've filled in the space already, either during search or as part of the starting state)
  - a list of numbers that are valid to put in that entry if it hasn't been filled in yet

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]

Which is the most constrained (of the ones above)?

- Board is a matrix (list of lists)
- Each entry is *either*:
  - a number (if we've filled in the space already, either during search or as part of the starting state)
  - a list of numbers that are valid to put in that entry if it hasn't been filled in yet