Programming, Problem Solving, and Algorithms

CPSC203, 2023 W2

Announcements

- Test 6 (last one!) will be from April 2nd – April 6th
  - Practice Test 6 will be released on PrairieLearn soon
  - Content covered: Transition Tables, Breadth-first search, Graphs, Sudoku Solver, Markov Chains

- Project 2 – reminder the deadline is March 29th

- Apologies for delay on some manual grading things, we're aiming to get to those soon.

Announcements

- Policy on “Retaking” an Examlet
  - Lowest Examlet score (including missed examlets) will be dropped!
  - Your best 5 of 6 examlets will be used
- Reminders of other (related) policies:
  - Best 10 of 11 Learning logs will be used
  - Best 10 of 11 POTW assignments will be used
  - Best 9 of 10 Labs will be used
  - Best 5 of 6 Examlets will be used
  - **No similar policy for Projects or the Final Exam!**

Announcements

- Reminder: there are course passing requirements

Passing requirements

- All students must satisfy ALL conditions to pass the course:
  1. Pass the Lab component with a grade of at least 50%,
  2. Pass the Test and Exam components (together) with a grade of at least 50%,
  3. Pass the Final Exam with a grade of at least 40%.

If students do not satisfy the appropriate requirements, the student will be assigned the **lower** of their earned course grade or, a maximum overall grade of 45 in the course.

Today's Plan...

1. Announcements! (10 mins)
2. Weekly Videos Review/Questions (20 mins)
3. Preparing for the Sudoku Solver (40 mins)

Slides from the Assigned Videos

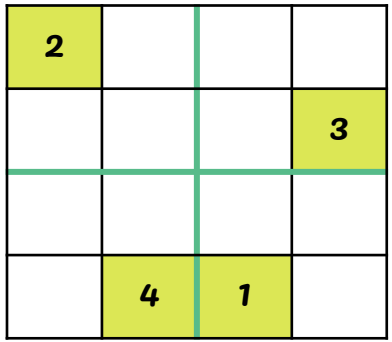
Representing Sudoku

A *representation* of a system is a model of the system that is useful in analysis.

A *state space* is a collection of all possible configurations of a physical system.

Each configuration is described using its representation, and is called a *state*.

How would you represent the game of Sudoku?



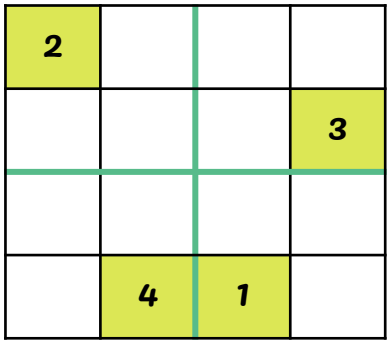
State Space Graphs

Define a graph where the set of vertices is _____.

And the set of edges consists of pairs (u,v) where _____

_____.

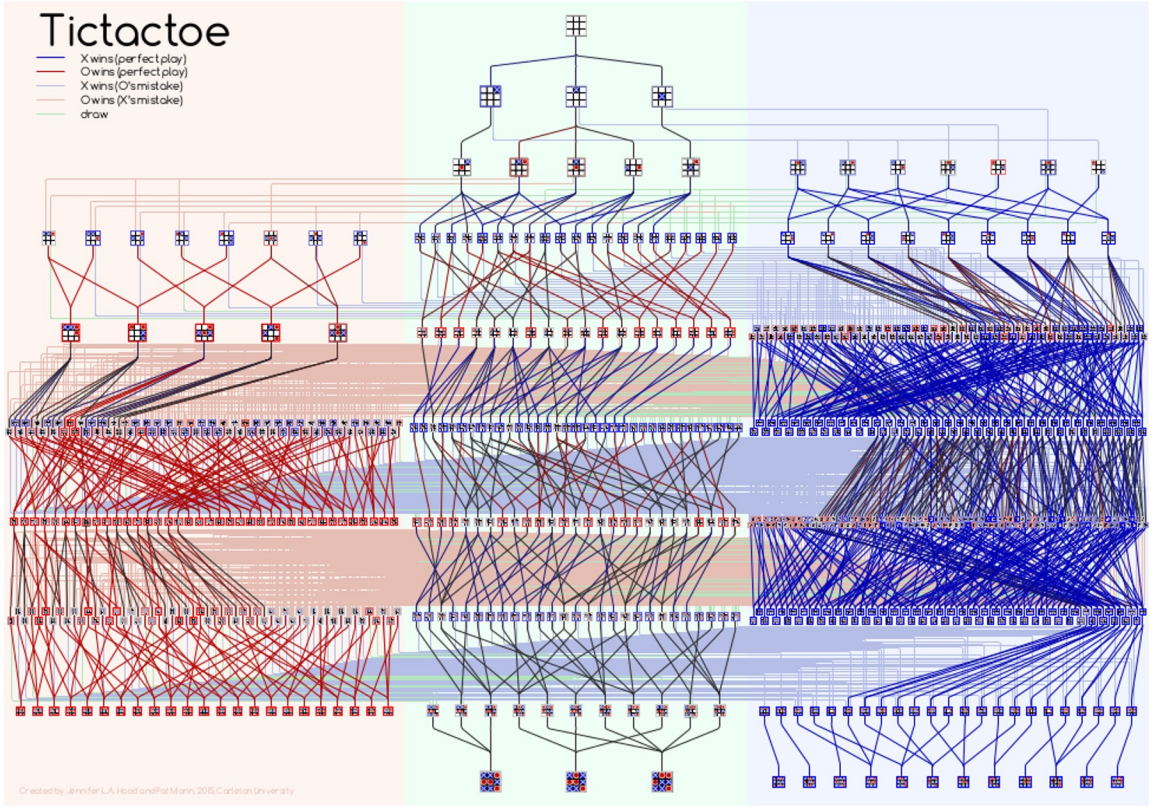
How many neighbors does this Sudoku puzzle state have?



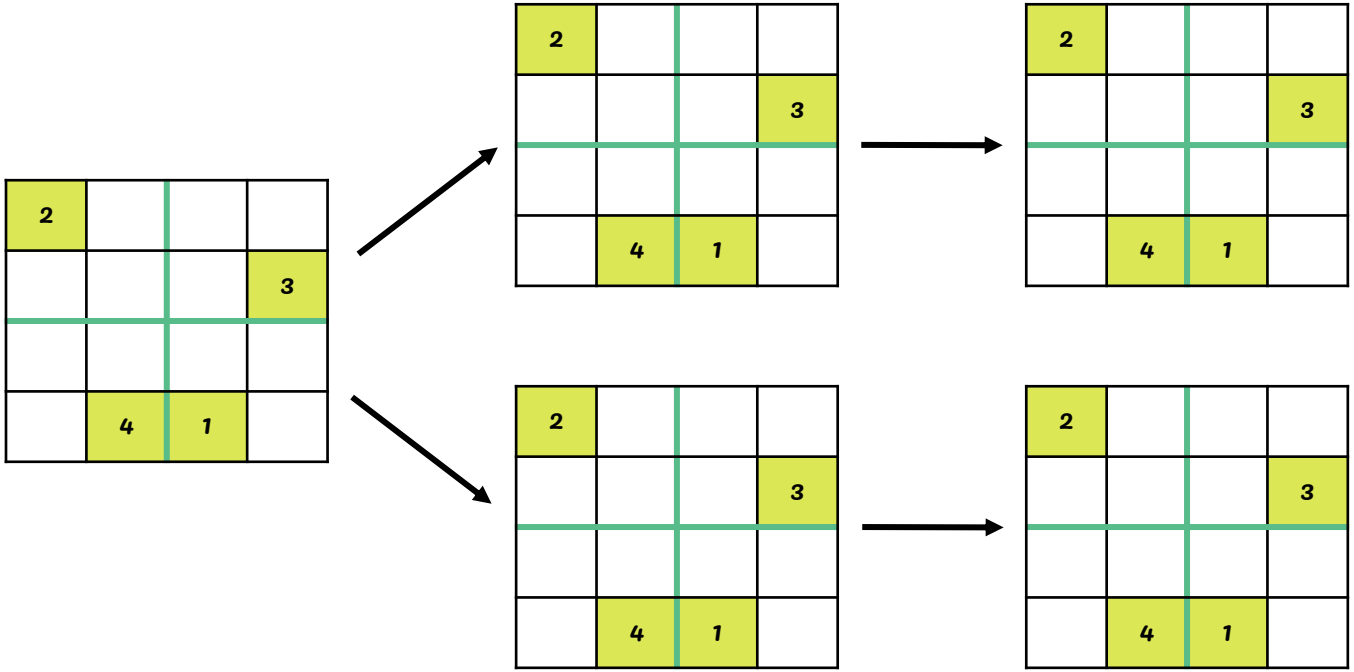
All neighbors:

Valid neighbors:

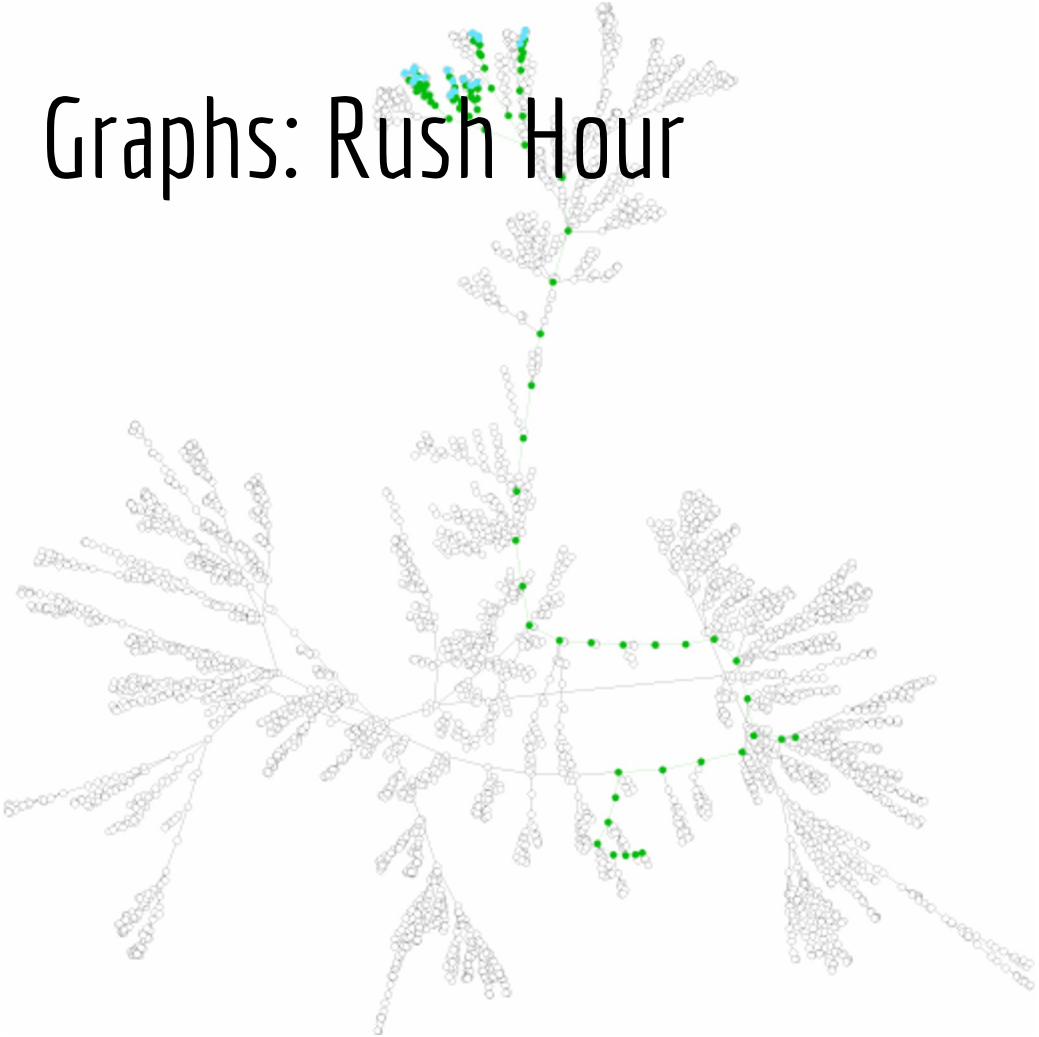
State Space Graphs - TicTacToe



Searching State Space Graphs



Graphs: Rush Hour



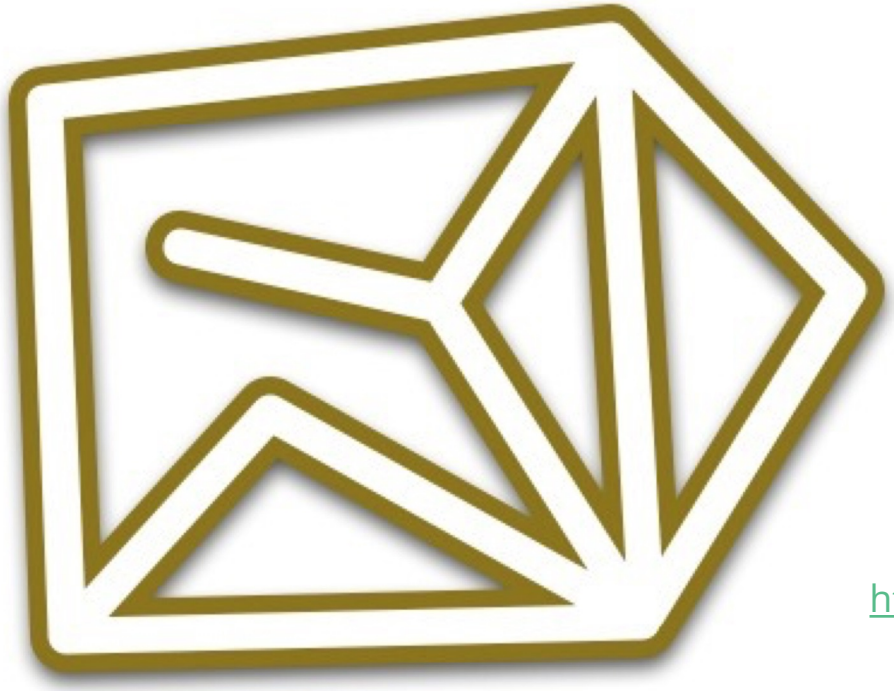
G = (V, E)

V: every vertex v is a board config

E: (u,v) means you can move from config u to config v

Path: sequence of vertices, connected by edges.

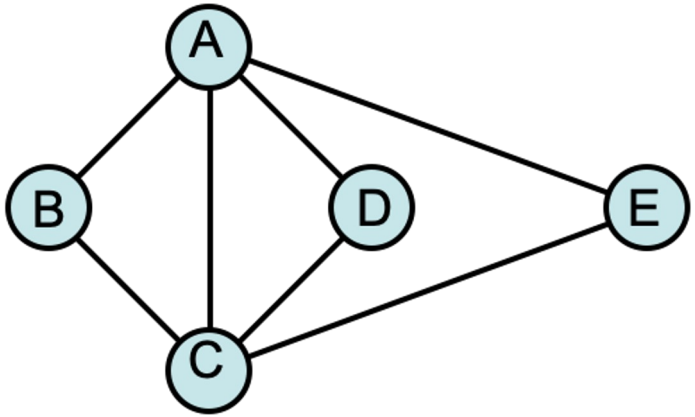
Depth First Search



Ariadne, Theseus, and the Minotaur

https://www.youtube.com/watch?v=8qrZ1clEp-Y

Depth First Search




```
Algorithm DFS(G,v)
        Input: graph G and start vertex v
        Output:  labeling of the edges of G in the
connected component of v as discovery edges and
back edges


setLabel(v, VISITED)
For all w in G.adjacentVertices(v)
    if getLabel(w) = UNVISITED
      setLabel((v,w),DISCOVERY)
      DFS(G,w)
    else if getLabel((v,w)) = UNEXPLORED
      setLabel(e,BACK)
```

Remember ADT: Stack?

Programmatic manifestation of _____.

ADT: Stack

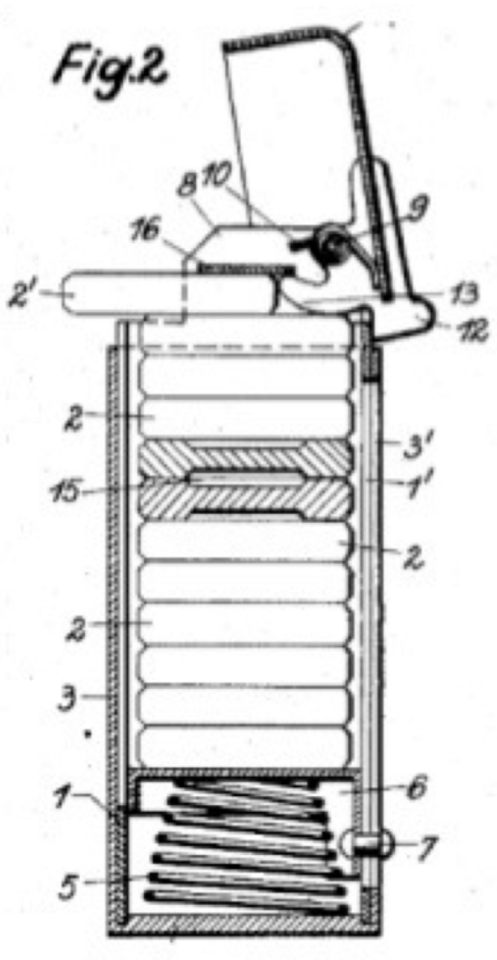
Insert -- push(data)

Remove -- pop() returns data

ADT: Deque (cuz python)

Insert -- append(data)

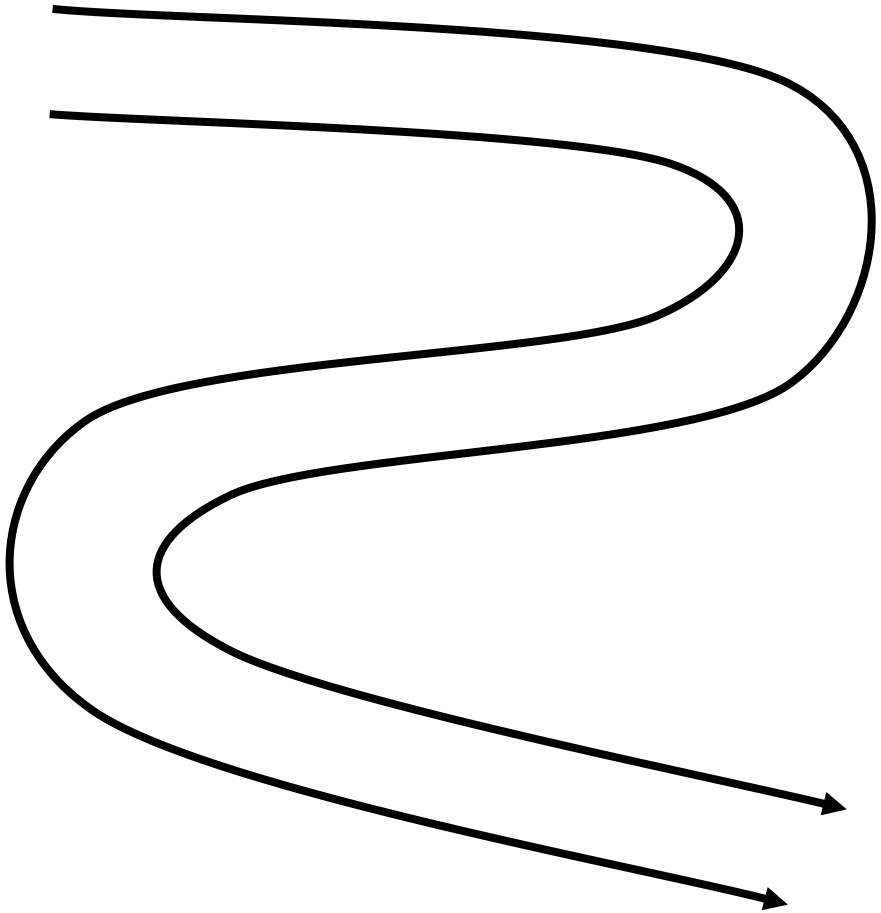
Remove -- pop()



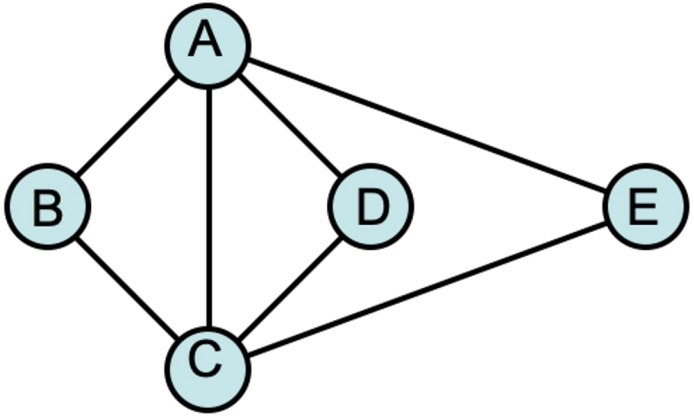
Putting it together

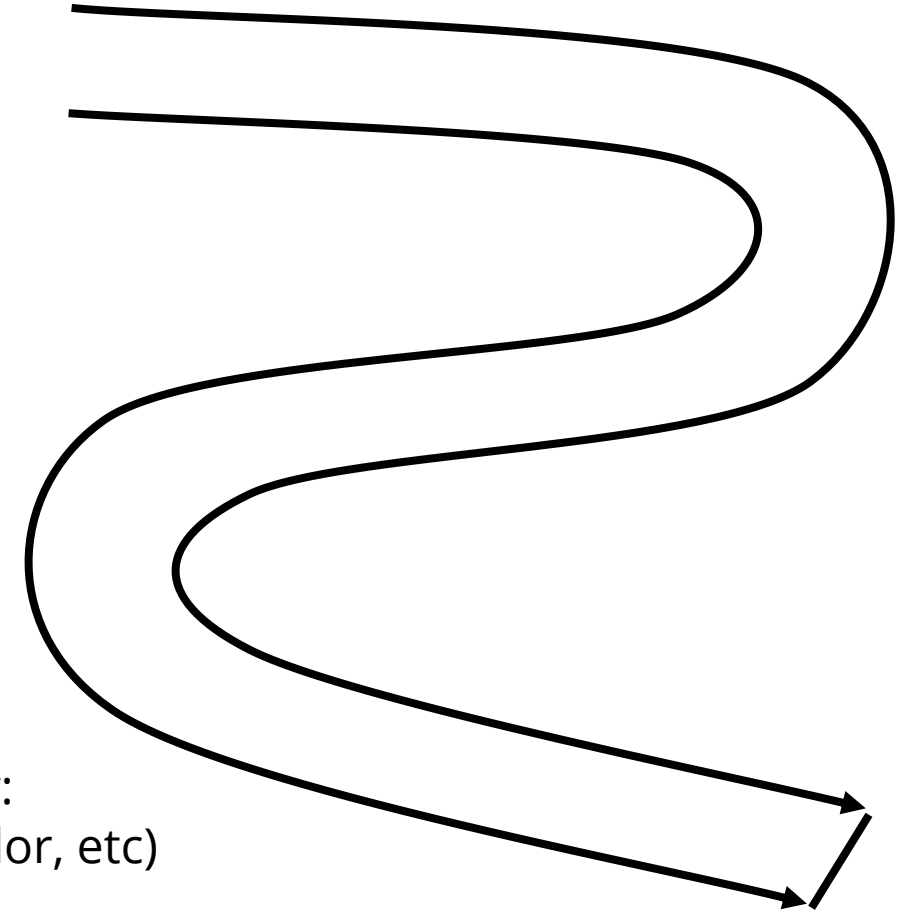
00	10	20	30	40	50	60	70	80	90
01	11	21	31	41	51	61	71	81	91
02	12	22	32	42	52	62	72	82	92
03	13	23	33	43	53	63	73	83	93
04	14	24	34	44	54	64	74	84	94
05	15	25	35	45	55	65	75	85	95

1) enqueue the center to start
2) while the queue is not empty:
    a) v = dequeue
    b) for each valid neighbor w, of v:
        i) color w
        ii) enqueue w



Iterative DFS





1) push the start
2) while the stack is not empty:
    a) v = pop()
    b) for each valid neighbor w, of v:
        i) process w (print, label, color, etc)
        ii) push(w)

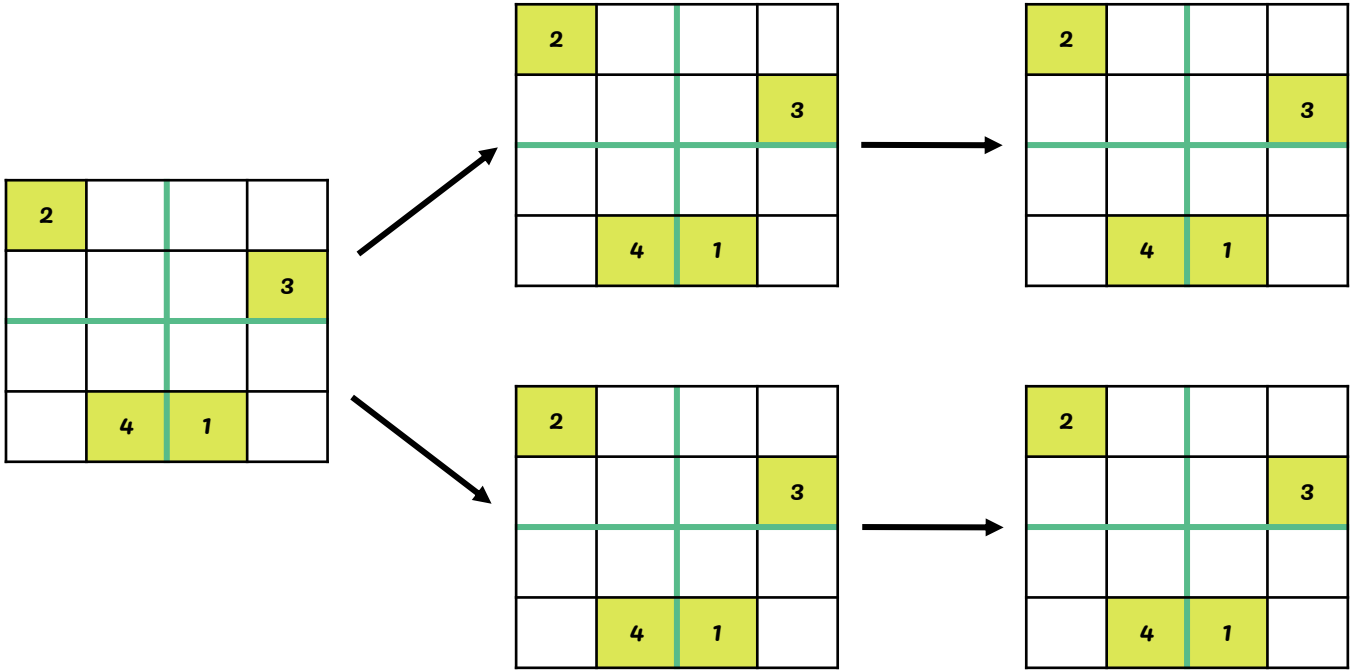
Overall Strategy:

Move forward through the states (board configurations) until you can’t go any farther.

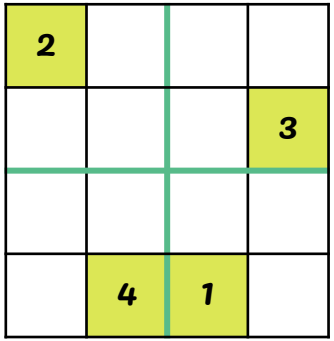
If the board is complete, you win!

If the board is not complete, then back up and try a new state in the most recent cell possible.

Searching State Space Graphs



Moving toward implementation:



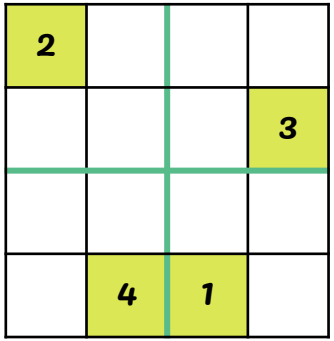
Need to be able to check whether a candidate entry is valid.

Suppose we have a variable `grid`, representing the board, and we want to place a value called `num`, in position `(x,y)`.

This code checks to see if num is valid. Which line checks row, which col?

```
63      if num in grid[x, :]:
64          return False
65      elif num in grid[:, y]:
66          return False
```

Moving toward implementation:



Need to be able to check whether a candidate entry is valid.

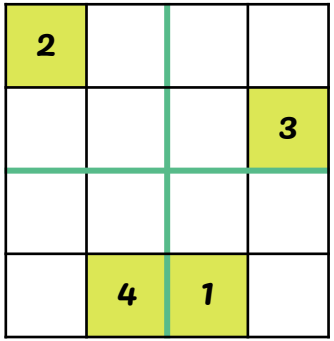
Suppose we have a variable `grid`, representing the board, and we want to place a value called `num`, in position `(x,y)`.

Region check?

EX: to query a region in a 2d numpy matrix, just define the bounds on the region and use `in`. In the above example, `2 in grid[0:2,0:2]` returns `True`.

New problem: define the region for given point `(x,y)` ?

New Problem:



Define the region for given point `(x,y)`

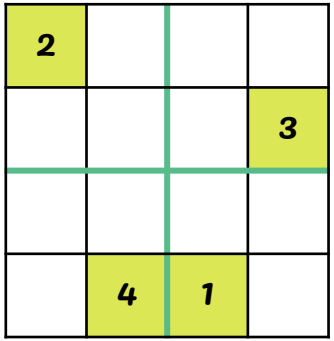
pt	region
(0,0)	[0:2,0:2]
(0,1)	
(1,0)	
(1,1)	

pt	region
(2,0)	[2:4,0:2]
(2,1)	
(3,0)	
(3,1)	

pt	region
(0,2)	[_:_,_:_]
(0,3)	
(1,2)	
(1,3)	

pt	region
(2,2)	[_:_,_:_]
(2,3)	
(3,2)	
(3,3)	

Now Generalize:



Goal: define the region for given point `(x,y)` in a r^2 x r^2 grid.

Here are some more examples:

(4, 8) in a 9x9 grid is in region [3:6, 6:9]

(22, 14) in a 25x25 grid is in region [20:25, 10:15]

(____,____) in a 100x100 grid is in region [____:____, ____:____]

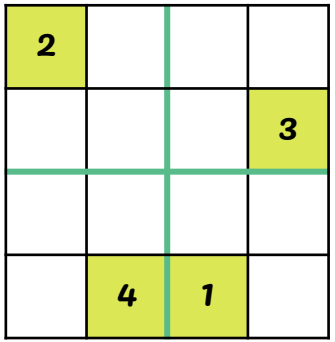
pt	region
(2,0)	[2:4,0:2]
(2,1)	
(3,0)	
(3,1)	

One last little thing:

We want to iterate over the 16 positions, but we need to refer to them by their `(x,y)` positions in the grid.

Write a function called `postup(p)` that takes a position `p` and returns `p`'s `(x,y)` coordinates in the grid.

Note: The upper left corner is position 0 and has coordinates `(0,0)`.)



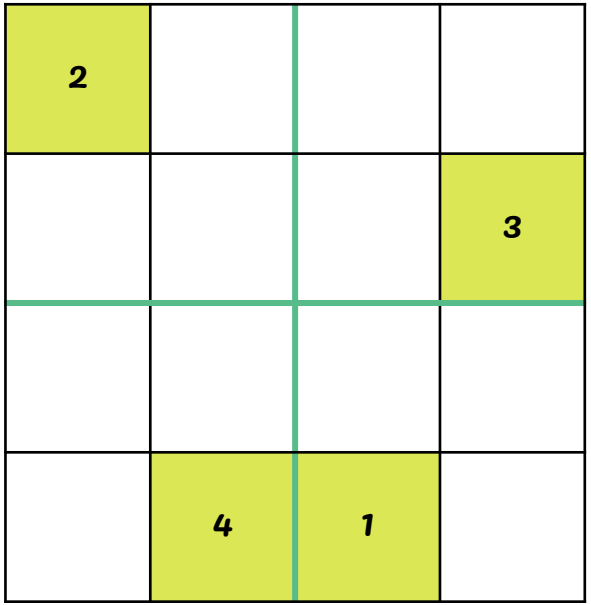
Demo:

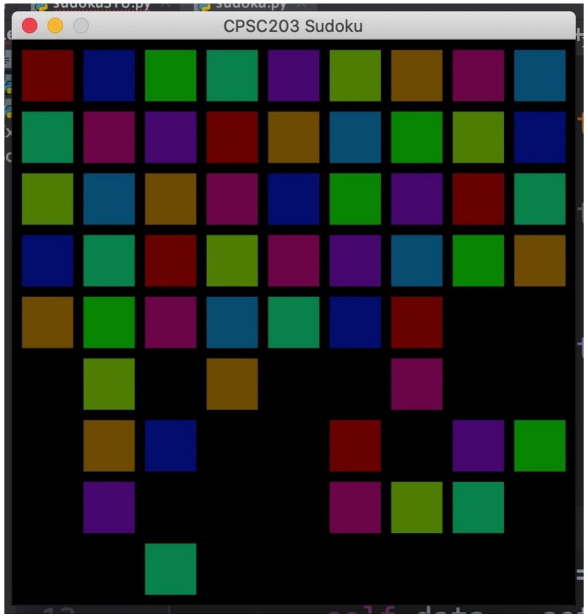
https://classroom.github.com/a/ZEaSSumI

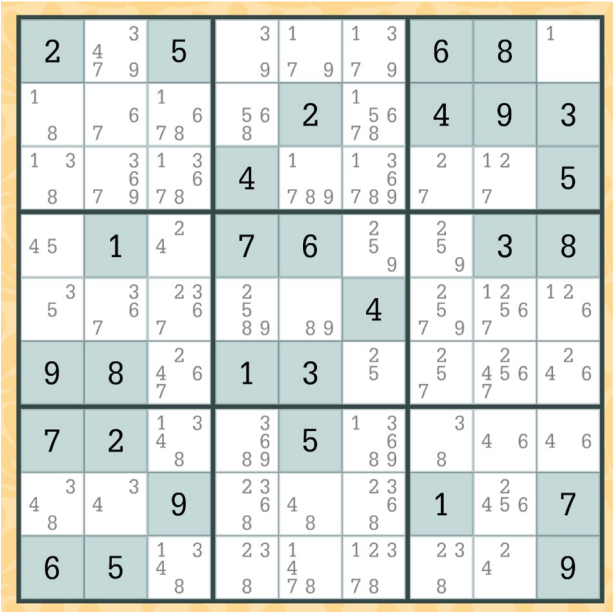
As we do the demo, be sure to write down new features of the code, and questions you have!

Sudoku, a closing thought...

Recall our algorithm for searching... could we be smarter?







Sudoku, another closing thought...

Is our solution to Sudoku tractable? (how fast does the state space grow, as we increase the board size?

3: 9x9

4: 16x16

5: 25x25

6: 36x36

Known to be NP-Complete --

Resources...

https://en.wikipedia.org/wiki/Mathematics_of_Sudoku (optional!)