Programming, Problem Solving, and Algorithms

CPSC203, 2019 W1

Announcements

Project 2 is released. Due 11:59p, Nov 7.

“Problem of the Day” continues!

Today:

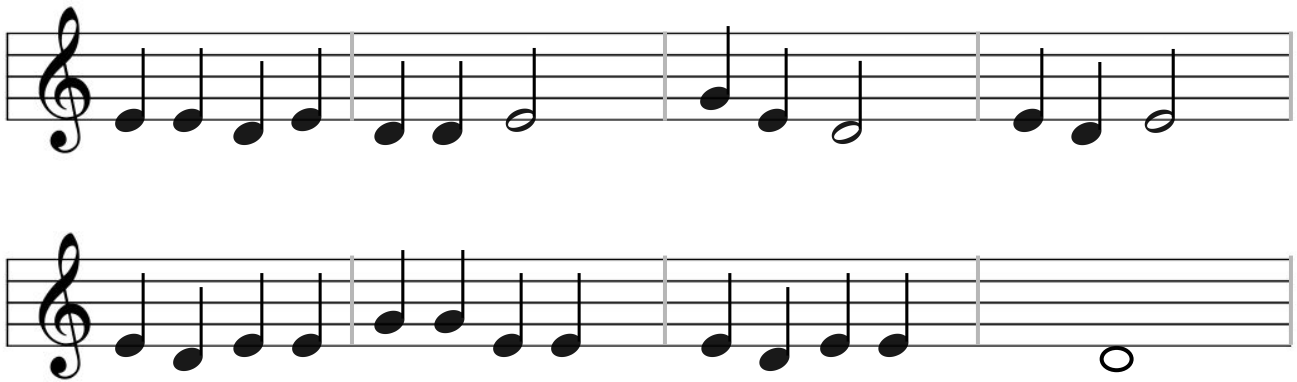
Markov Chains Fin

State Space Search

Representation

Implementation

Building a Song



1. Randomly choose a start note and put it in a list
2. for 25 notes, in the rhythm of MHaLL
    a. Generate a new note
    b. Put the new note in the list
3. play the list of notes

	C	D	E	G
C	0	1.0	0	0
D	0.3	0.3	0.4	0
E	0	0.45	0.45	0.1
G	0	0	0.5	0.5

The Technical Details

You have just learned about a particular type of random process called a *Markov Chain*.

We modelled it using a *transition table*, or a *finite state machine*, and we used it as the basis for an algorithm to generate music.

Now let's look at some code!

https://github.students.cs.ubc.ca/cpsc203-2019w-t1/LecMHALL/

Other Applications

PageRank: Google's first search algorithm

- Some pages are likely to "follow" (be linked from) others.
- Rank of page is based on the probability that you'll be there at any moment

Natural Language Processing

- Some words are more likely to follow others.
- "I just ate the whole desert" probably has a misspelling.
- "For dinner I ___ … " next word is probably "ate"

DNA matching

Chemical reaction simulation

Many others…

But I thought we were talking about graphs...

	C	D	E	G
C	0	1.0	0	0
D	0.3	0.3	0.4	0
E	0	0.45	0.45	0.1
G	0	0	0.5	0.5

http://setosa.io/markov

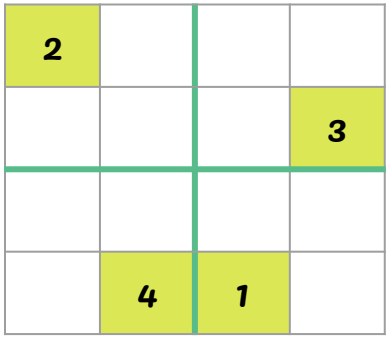
Representing Sudoku

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?



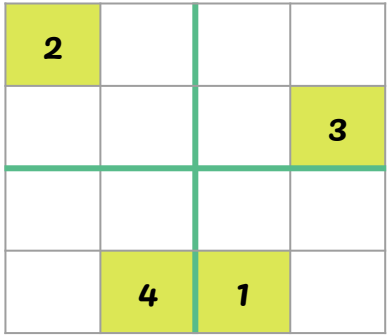
State Space Graphs

Define a graph where the set of vertices is _____.

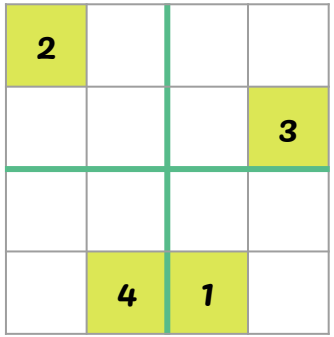
And the set of edges consists of pairs (u,v) where _____

_____.

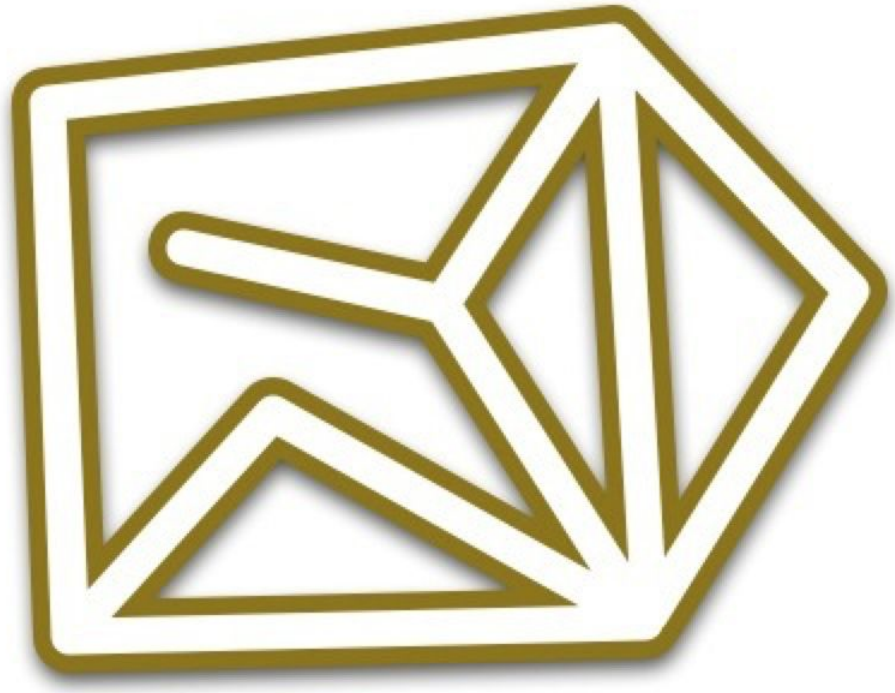
How many neighbors does this Sudoku puzzle state have?



Searching State Space Graphs

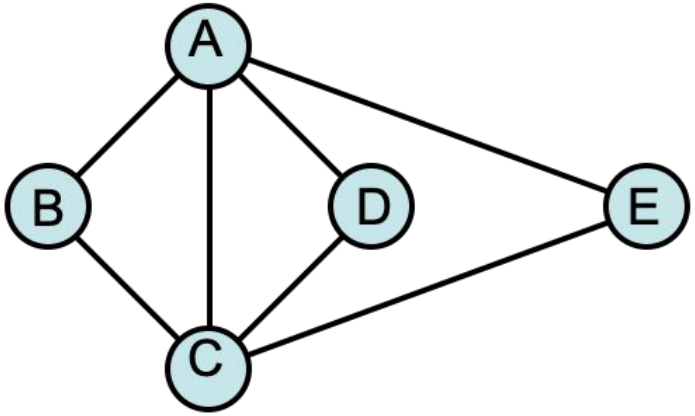


Depth First Search



Ariadne, Theseus, and the Minotaur

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)
```

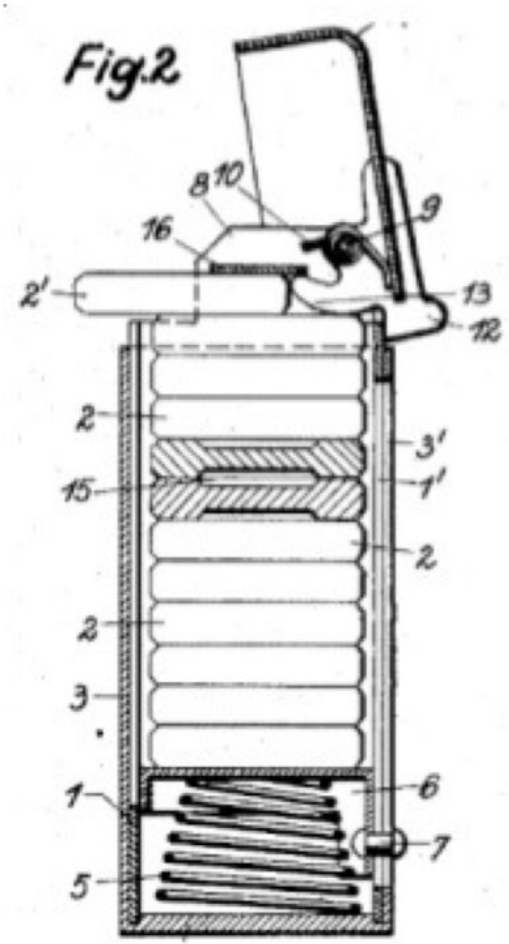
A new ADT: Stack

Programmatic manifestation of _____.

ADT: Stack

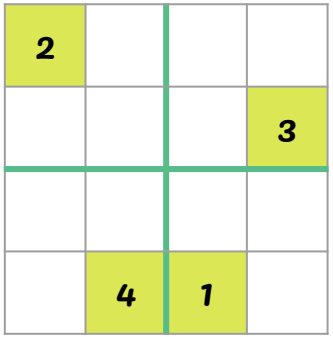
Insert -- push(data)

Remove -- pop() returns data



Recursion: An abstract Stack

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)`.

Row check:

Column check:

Region check:

POTD #28 Thu

https://github.students.cs.ubc.ca/cpsc203-2019w-t1/potd28

Describe any snags you run into:

1. Line ___: _____
2. Line ___: _____
3. Line ___: _____
4. Line ___: _____
5. Line ___: _____

ToDo for next class...

POTD: Continue every weekday! Submit to repo.

Reading: TLACS Ch 10 & 12 (lists and dictionaries)

References:

https://brilliant.org/wiki/markov-chains/

https://medium.com/@eightlimbed/counting-on-pythons-defaultdict-b652204780bd