AI-m of the Game

Workshop #1: Introduction to AI Games

Project Ignite Instructors:
Brandon Wang, Brandon Wei

Project Introduction

What is “AI-m of the Game”?

Project Advisors (PAs)

Brandon Wang

- CMU Junior
- ECE + Robotics
- Previously PA for Robotics Lab
- Likes: math, video games, manga, biking

Brandon Wei

- CMU Sophomore
- ECE
- Previously PA for VR Game
- Likes: tennis, video games

Project Participants

- Abhay
- Anirudh
- Philip
- Rushil
- Subham
- Varun
- Victor
- Vikram
- Vishaal

Give us your name, grade, where you're from, and a like / interest / fact / why you want to be here

AI-m of the Game Overview

Students will implement a variety of AI algorithms to beat simple 2-player classical competitive games

- Examples: Nim, Tic-Tac-Toe, Connect Four, checkers, chess, Battleships, etc.
- We'll discuss (not implement) more complex methods (eg. neural nets, Q-learning)

Workshop Structure

Workshops #1-4

- Discuss multiple game structures and basic AI techniques
- Take-away game, Tic-Tac-Toe, Nim, Sim, and Connect Four
- Walk you step-by-step

Workshops #5-10

- Choose, build, then “solve” your own game!
  - should be hard eg. checkers, Go, chess
- Do your own research
- Advisors will only give guidance and deadlines

Key Links for this Course

- Meeting Minutes: https://tinyurl.com/y5e365wn
  - workshop schedule and notes
  - links to resources for learning and reviewing
  - let me know if you can't access and edit it
- GitHub: https://github.com/project-ignite-2021-ai
  - upload/download all of the code
- CMU Canvas
  - upload/download all of the non-code files (eg. slides)

AI Games

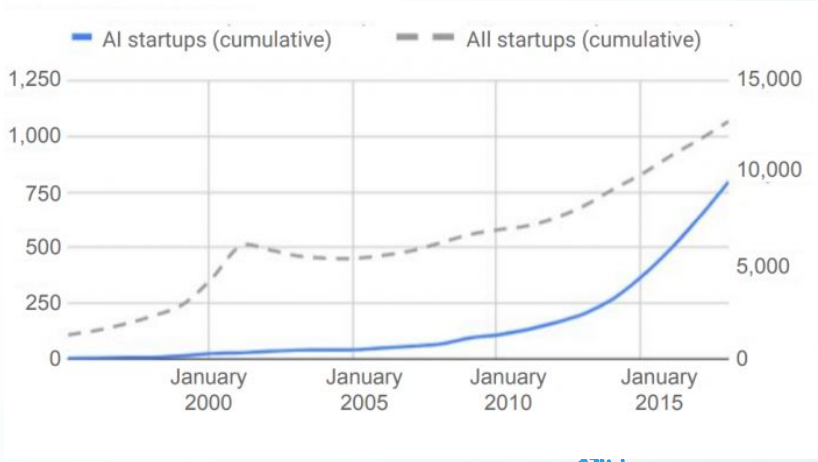
What is AI? What are “games”?

AI (Artificial Intelligence)

- AI = a program that thinks intelligently and rationally
  - used to replace natural intelligence (thinking like humans/animals)
- traditional AI problems/tasks
  - reasoning (eg. decide if it's going to rain tomorrow)
  - planning (eg. path from City A to City B)
  - processing (eg. language)
  - identifying (eg. differentiate b/w a dog and a cat)
- involves boolean logic, probability, statistics, mathematical optimizations, and data structures

AI Motivations

- growing incredibly fast!!
- versatile - used in countless products and industries
- the “skill of the century”
  - used by every major company, and pays a lot!
- the “perfect” program
  - incredibly smart, fast, accurate, and predictive



“Game”

- game = environment where 2+ players (agents) try to maximize their own reward (utility)
  - Eg. Fortnite, checkers, poker, race, basketball, hide-and-seek, group project, Human Knot
  - Not Eg. Mario, Pac-Man, sudoku, Secret Santa, teaching
- There are a lot of kinds of games, so we’ll need classifications for them

Game Types: Simultaneous vs Sequential

Simultaneous

players make their moves at the exact same time

- Eg. RPS, prisoner's dilemma, "Shotgun"

Sequential

players switch b/w taking turns to make their moves

- Eg. Go Fish, Tic-Tac-Toe, Blackjack, Mario Party, RPG

Real-Time

players make their moves whenever they want to

- Eg. Smash Bros, Fall Guys, LOL, race

Game Types: Zero-Sum vs General-Sum

Zero-Sum Games

- opposite utilities that sum to 0 (or another constant)
- adversarial/competitive
  - I win if you lose!
- Eg. RPS, Tic-Tac-Toe, chess, football

General-Sum Games

- independent utilities that don't remain the same sum
- may be cooperative, competitive, indifferent, or shift alliances
- Eg. prisoner's dilemma, Blackjack, Overcooked, elections

Game Types: Zero-Sum vs General-Sum

Zero-Sum: RPS

Player 2

Player 1	Rock	Paper	Scissors
Rock	0, 0	-1, 1	1, -1
Paper	1, -1	0, 0	-1, 1
Scissors	-1, 1	1, -1	0, 0

General-Sum: Prisoner's Dilemma

Player 2

Player 1	Silent	Testify
Silent	-1, -1	-5, 0
Testify	0, -5	-3, -3

Other Game Types

- fully observable (ie. perfect information) vs partially observable
  - chess vs Battleships
- deterministic vs stochastic (ie. random)
  - choosing a piece to move vs rolling a die
- discrete vs continuous
  - Pong (arcade game) vs ping-pong

So what games are *we* learning?

combinatorial games: 2-player sequential zero-sum fully-observable discrete deterministic games

- a subset of 2-player abstract strategy games
- Eg. Tic-Tac-Toe, dots-and-boxes, Go, checkers, chess, Chinese checkers, Xiangqi, shogi, Hex, mancala, Othello, Blokus, Connect Four, Nine Men's Morris, Chopsticks
- Not Eg. Pong, Scrabble, Blackjack, Jenga, Uno, Mahjong, Chutes and Ladders, Pictionary, Yugioh, RPS, Old Maid, Monopoly, Go Fish, Guess Who, darts, matching games

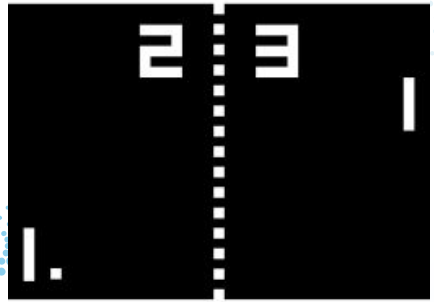
Game Strategies

How will our AI agent beat our opponent?

Consider these Simple 2P Games

Pong (arcade game)

move a paddle to stop a ball from getting past you



100-meter Race

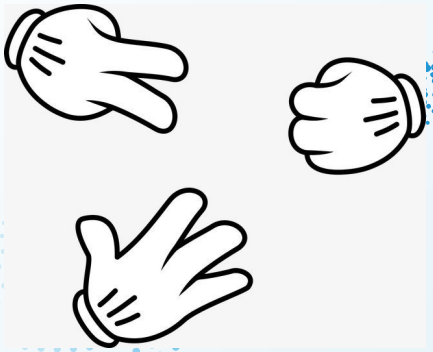
whoever crosses the finish line first wins!

RPS

Rock beats Scissors

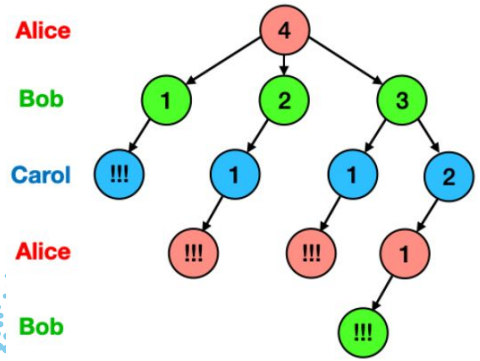
Paper beats Rock

Scissors beats Paper



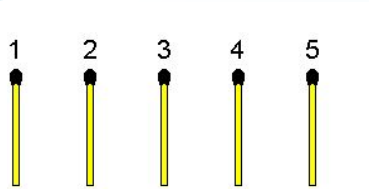
Consider these Strategy 2P Games

During-Each-Turn-A-Person-Says-A-Number-Between-1-and-6-That's-Less-Than-The-Previous-Person's-Number-And-The-Person-Who-Cannot-Say-Any-Such-Number-Win



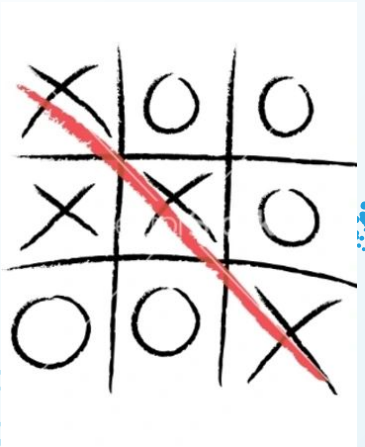
Takeaway Game

whoever draws the last “stick” wins



Tic-Tac-Toe

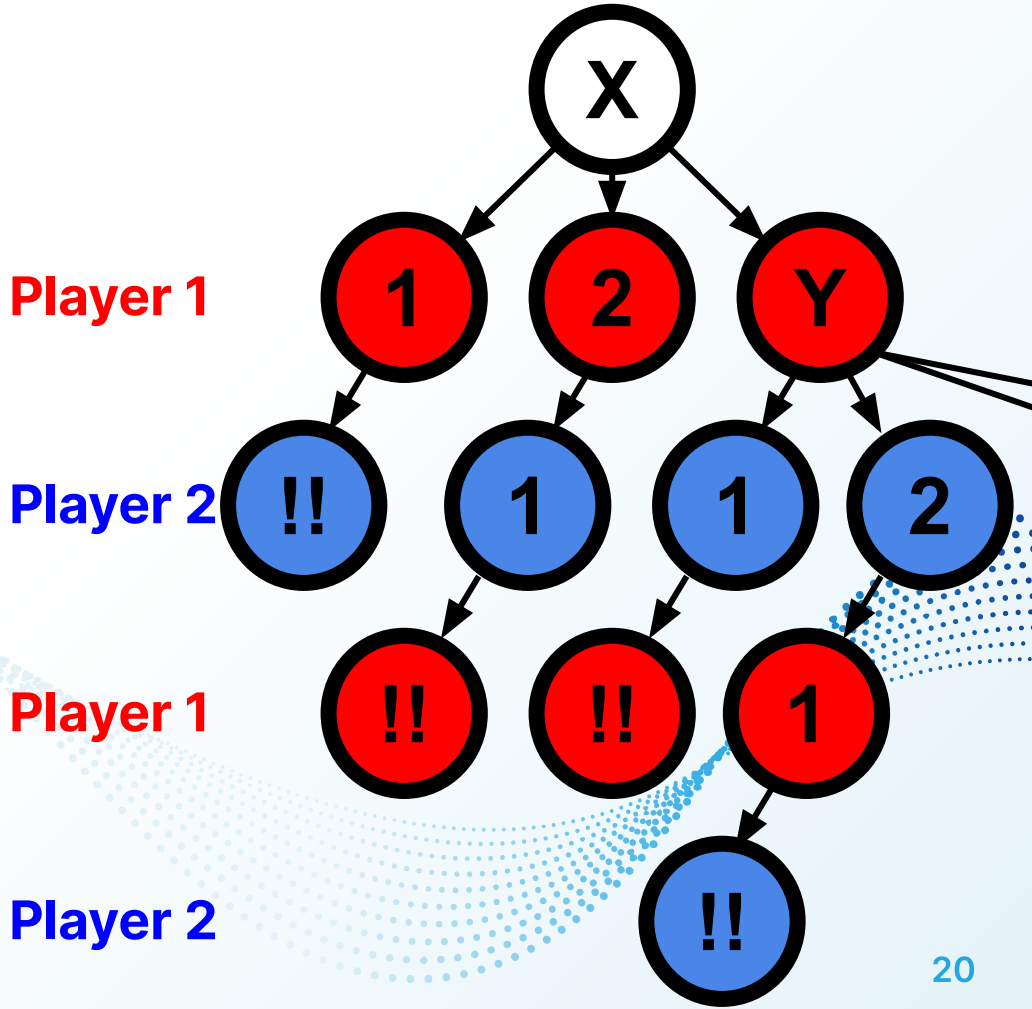
whoever makes 3 in a row wins



During-Each-...-Wins

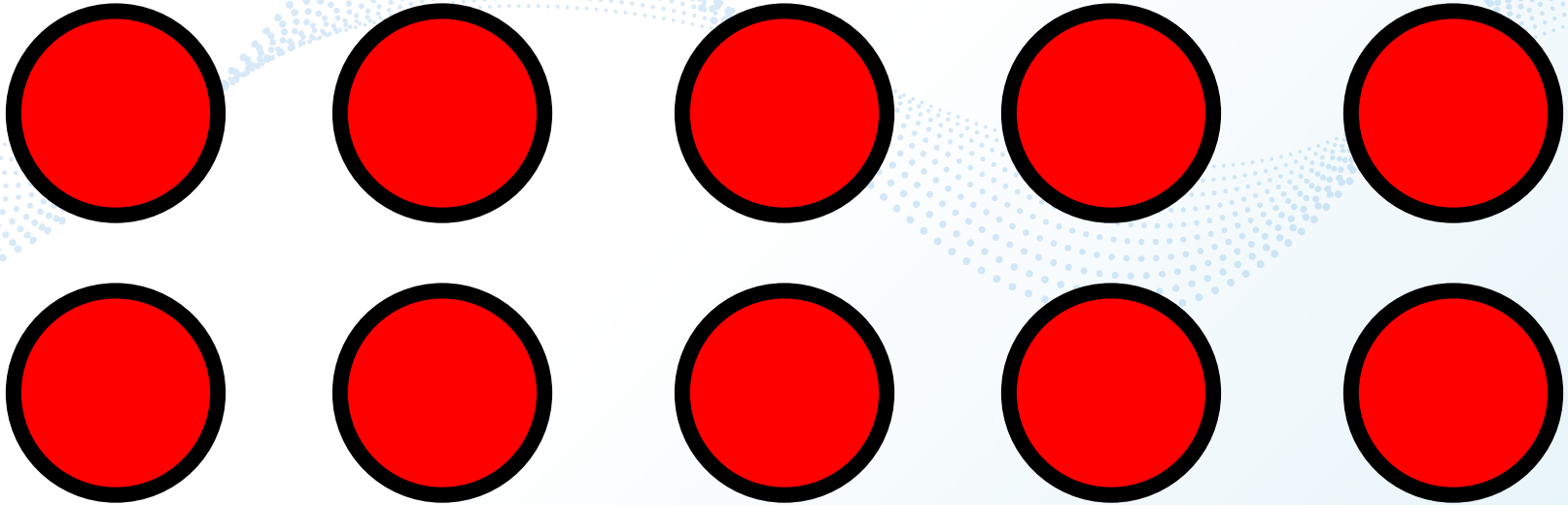
Proposition: If opponent says ‘2’, you are forced to say ‘1’, and your opponent wins

Strategy: Always say ‘2’ yourself!



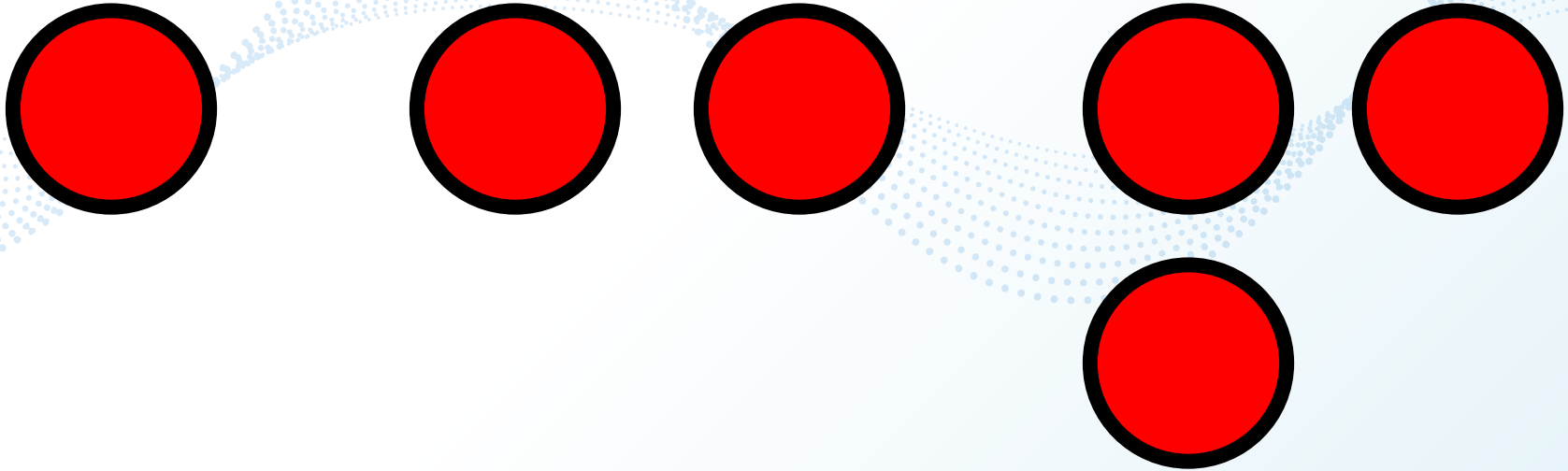
Takeaway Game

Take 1, 2, or 3 dots. The winner is whoever takes the last dot!



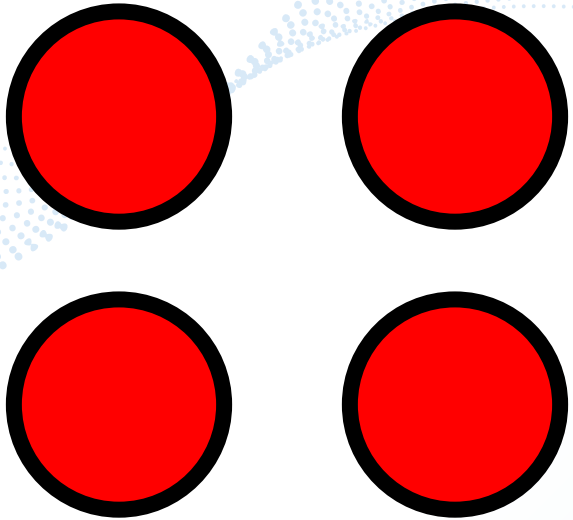
Takeaway Game

Begin with 1,2,3 dots



Takeaway Game

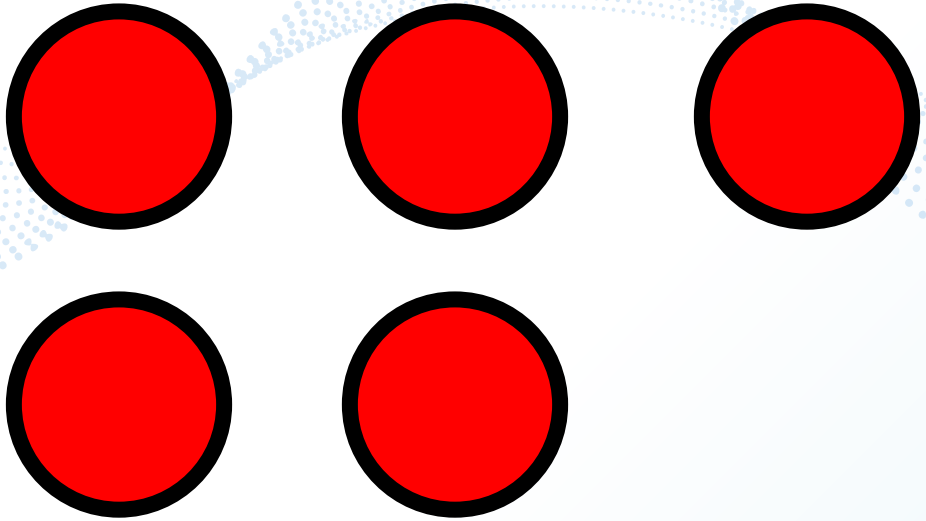
Begin with 4 dots



“Zugzwang”

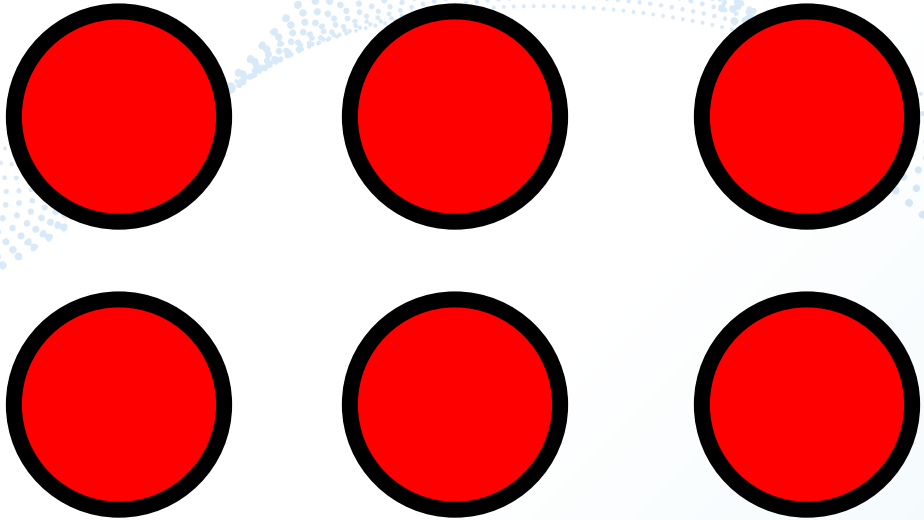
Takeaway Game

Begin with 5 dots



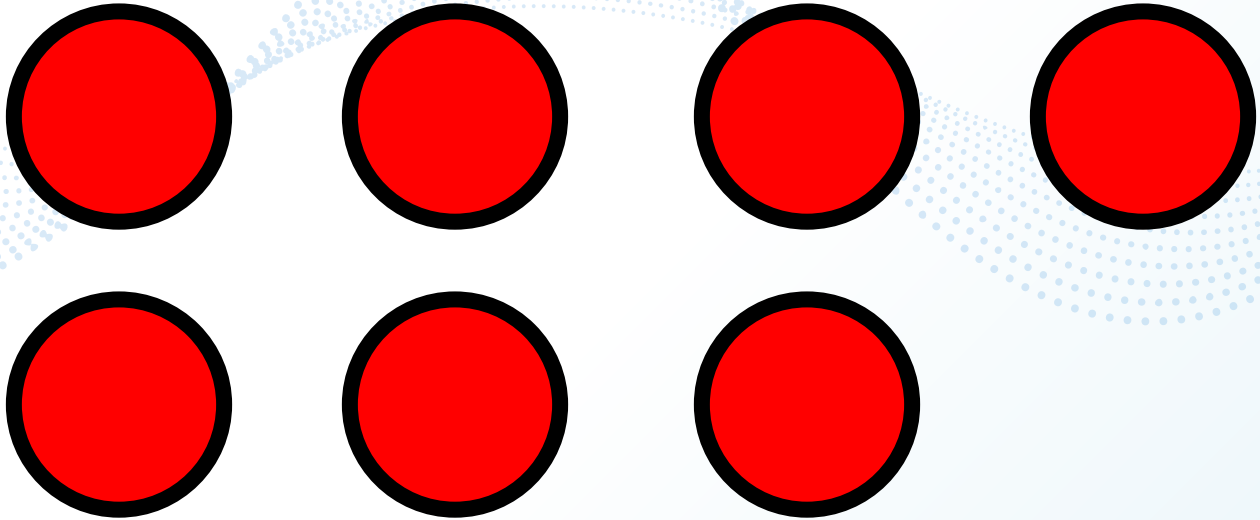
Takeaway Game

Begin with 6 dots



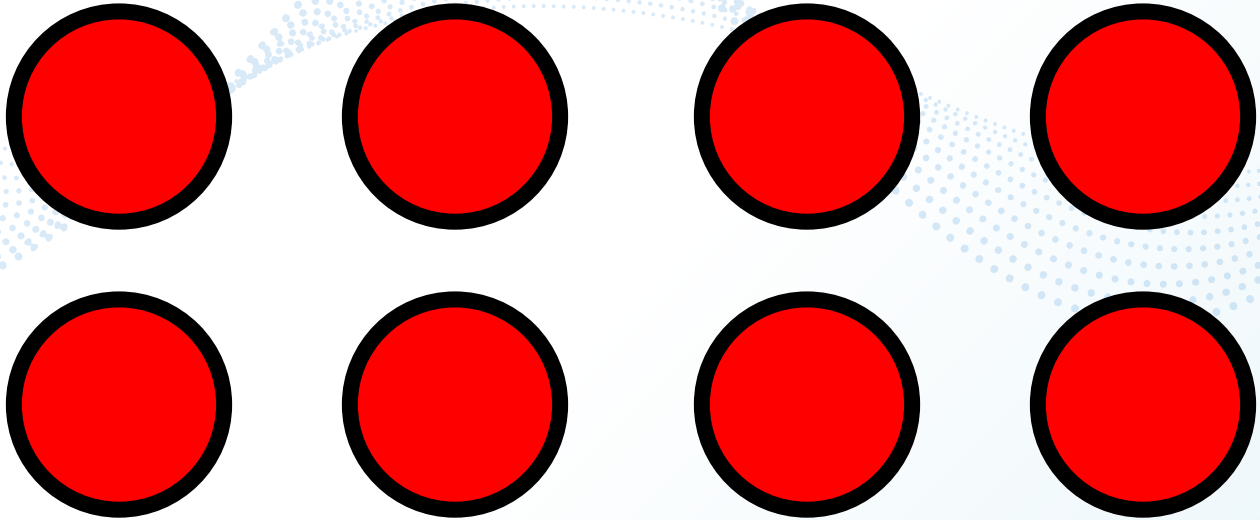
Takeaway Game

Begin with 7 dots



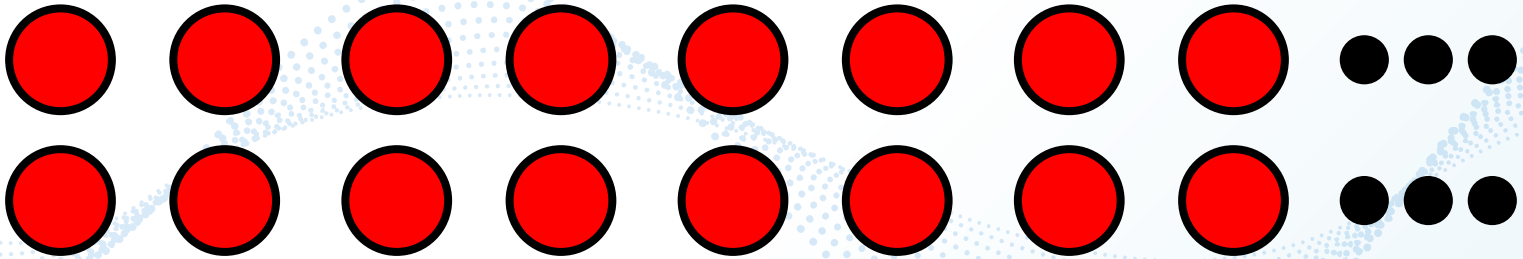
Takeaway Game

Begin with 8 dots

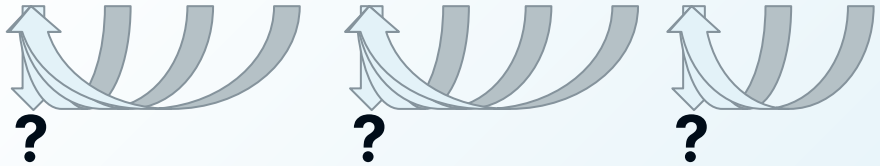


Takeaway Game

Begin with X dots



X	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Outcome	W	W	W	L	W	W	W	L	W	W	W	L	W	W

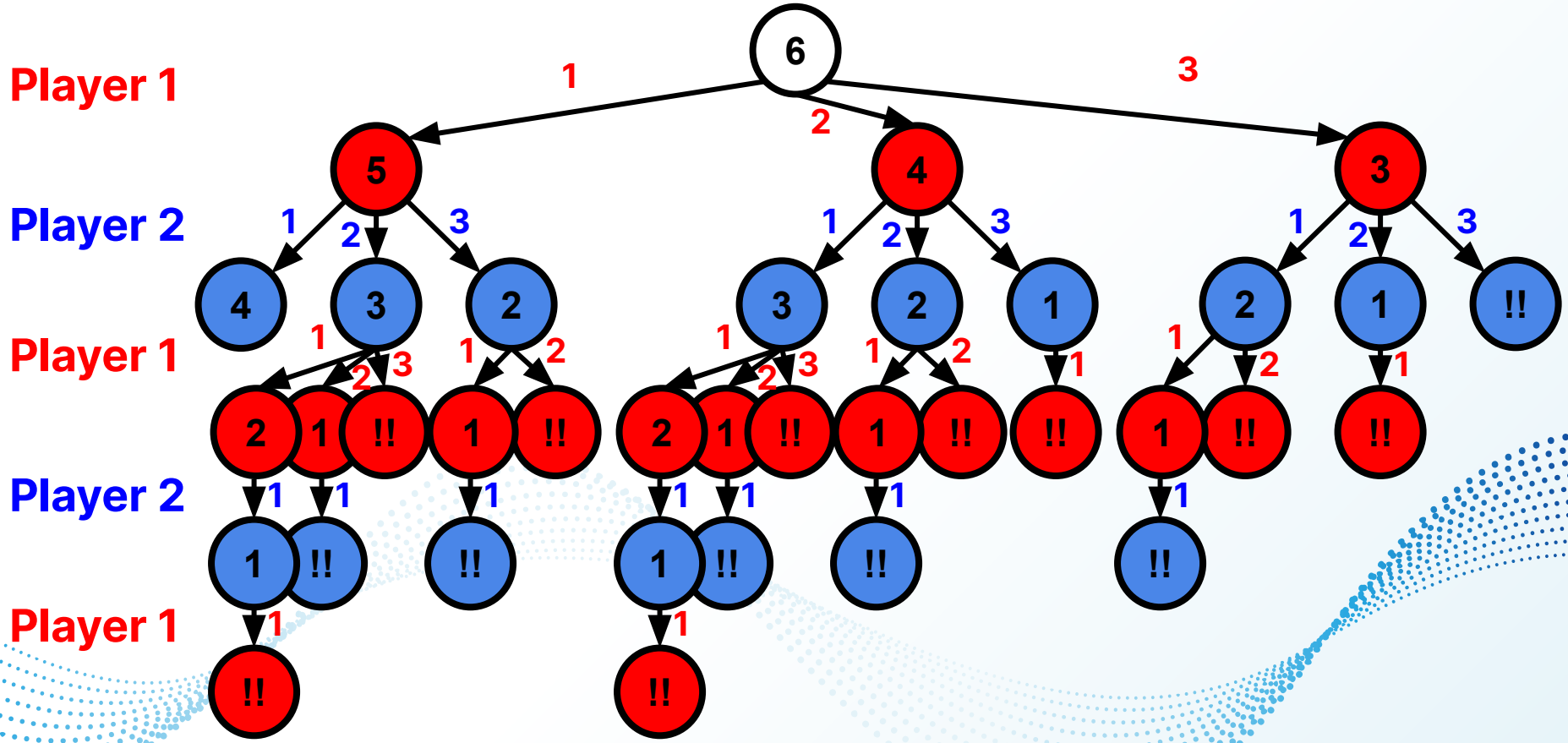


Takeaway Game (1,2,3)

Proposition: if you have a multiple of 4 sticks at the start of your turn, you lose the game

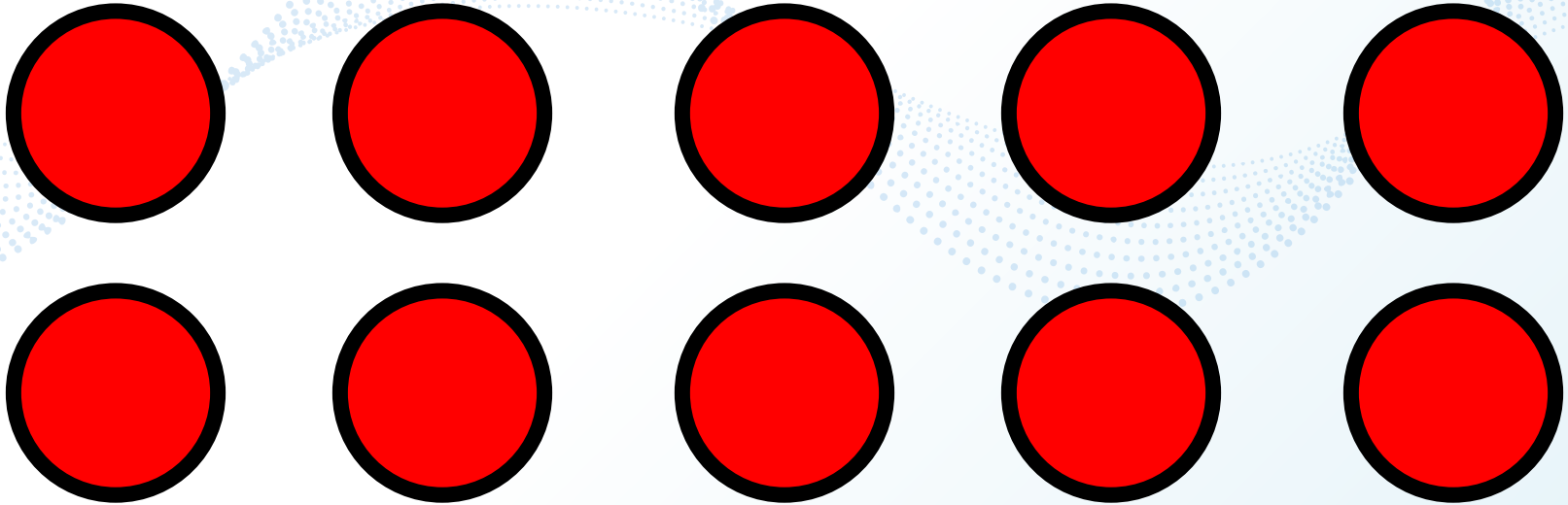
Strategy: each turn, leave your opponent with a multiple of 4 sticks

X	1	2	3	4	5	6	7	8	...
Outcome	W	W	W	L	W	W	W	L	...


Player 1
Player 2
Player 1
Player 2
Player 1
6
5
4
3
4
3
2
3
2
1
2
1
!!
2
1
!!
1
!!
2
1
!!
1
!!
!!
1
!!
!!
1
!!
!!
!!
!!

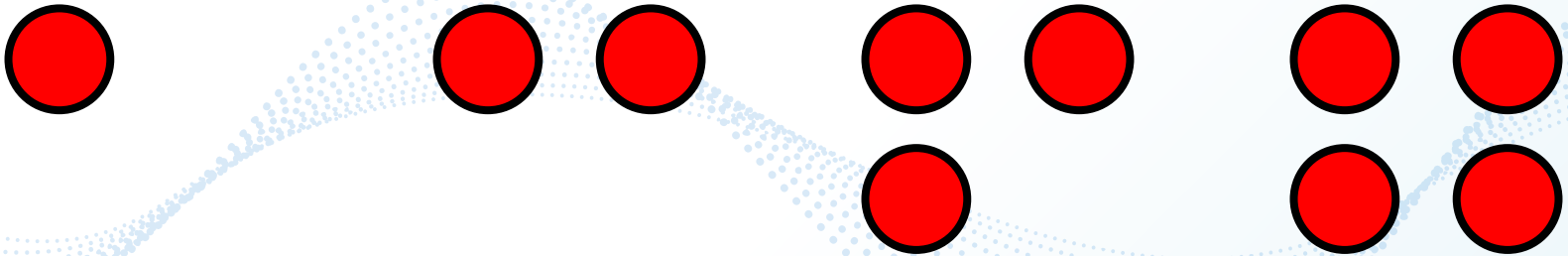
Takeaway Game

Take 1, 2, or ***4*** dots. The winner is whoever takes the last dot!



Takeaway Game

Begin with 1,2,3,4 dots



X	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Outcome	W	W	L	W										

Takeaway Game

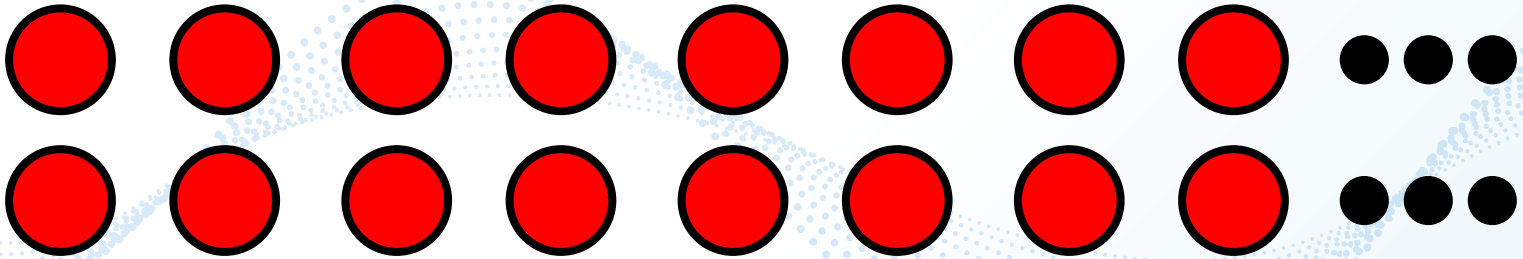
Begin with 5,6 dots

X	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Outcome	W	W	L	W	W	L								

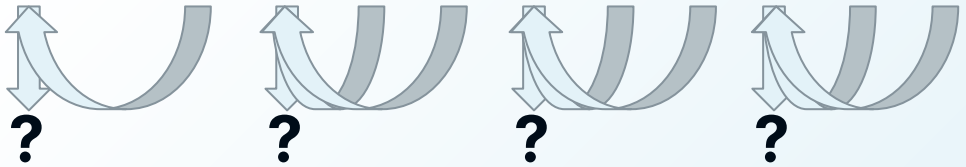
?

Takeaway Game

Begin with X dots



X	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Outcome	W	W	L	W	W	L	W	W	L	W	W	L	W	W



Takeaway Game (any range)

Proposition: if you make a move that puts the opponent in a losing position, you win

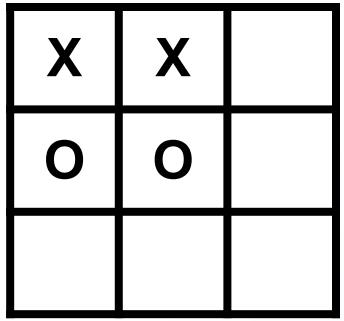
Strategy: backtrack from the end states (ie. X=0) to find the losing and winning positions

X	1	2	3	4	5	6	7	8	...
Outcome	W	W	L	W	W	L	W	W	...

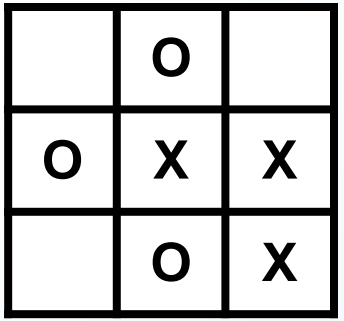
Tic-Tac-Toe

Proposition: if you make a move that puts the opponent in a losing position, you win

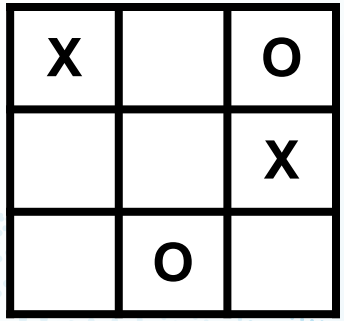
Strategy: well, it's complicated (learn next workshop!)



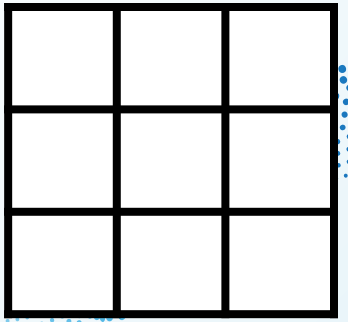
O to move



O to move



O to move



O to move

Programming Game Strategies

We know how to beat our opponent. Let’s code it!

Python

Python = an easy-to-learn widely-used functional programming language

- ignores the complexities of argument and variable types, at the cost of being slower to compile and run
- 100s of 1000s of public Python modules/libraries for various purposes (eg. game development, machine learning, data analysis)

Let’s review some Python concepts

Assignments

No need for variable types!

```
x = 3
str = “Hello, World!”
A,B,C = [‘A’,True,10]
```

Operations

The same as other languages

```
x = 3
x += (x**2) % (x - 2/2)
print(x) # this is 4
```

Conditionals

Use English words

```
if x >= 0 or x == -1:
     print(“Valid”)
elif x < 0 and x != -1:
     print(“Invalid”)
```

Lists/Tuples/Sets

Learn the differences!

```
l = [1, 2, 3, 4]
t = (1, 2, 3, 4)
s = {1, 2, 3, 4}
```

Loops

2 kinds: while and for loops

```
x = 0
while x < 100:
     x = x + 2
for i in range(1,11):
     print(i)
```

Functions

The syntax is slightly different, but should be familiar

```
def sumdif(x, y):
     return [x+y, x-y]
sum, diff = sumdif(5, 3)
```

Python Checklist

- Download any version of Python3 (preferably 3.6+)
  - https://www.python.org/downloads/
- Get a (good) Python IDE
  - VSCode, IDLE, PyCharm, Sublime, Atom
  - do ***NOT*** use a text editor or command prompt!
- Get pip, which allows you to install Python libraries
  - https://pip.pypa.io/en/stable/installing/
  - on your Terminal/Command Prompt, run:

```
pip install pygame
```

Solve the Takeaway Game

Download takeaway.py from GitHub

- Briefly scan through the file to understand the rules/instructions/structure
- Try playing it for yourself by running the code
- Complete the FixedRangeStrat() function
- Complete the BacktrackStrat() function
- Try running each one by setting the Player1Strategy and/or Player2Strategy by un/commenting them
  - set sticks and pickup_range in main()

One More Game: Nim

A general takeaway game with multiple piles!

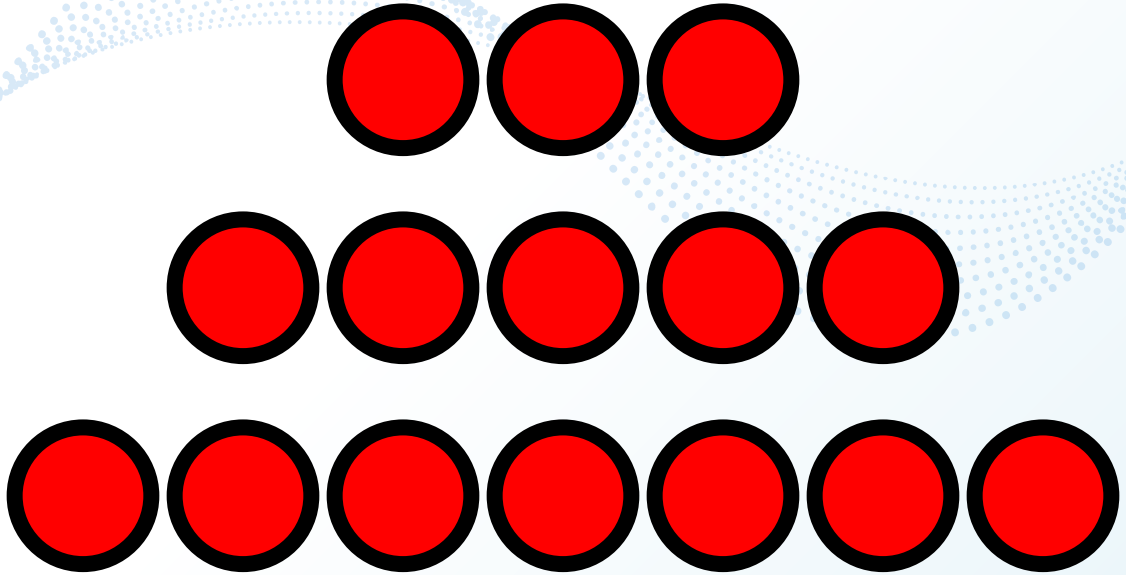
Nim

Take any number of dots (≥1) from any pile. The winner is whoever takes the last dot!

Pile 0

Pile 1

Pile 2



Nim
Strategy 1

backtrack from the end states (ie. piles=[0,0,0]) to find the losing and winning positions

- harder to code
- very messy
- very, very slow

Piles	[1,0,0]	[2,0,0]	[3,0,0]	[3,0,1]	
Outcome	W	W	W	L	...

Nim
Strategy 2

Proposition: if the Nim Sum (ie. column sum of 1s in binary form, modulo 2) is 0, you're losing

Strategy: make a move that makes the Nim Sum equal to 0, putting your opponent in a losing position

Piles = [3,5,7]

3: 0 1 1
5: 1 0 1
7: 1 1 1
Nim Sum: 0 0 1

Taking 1 from Pile 0 is winning:

Piles = [2,5,7]

2: 0 1 0
5: 1 0 1
7: 1 1 1
Nim Sum: 0 0 0

For a better explanation: https://plus.maths.org/content/play-win-nim

Solve Nim

Download nim.py from GitHub

- Briefly scan through the file to understand the rules/instructions/structure
- Try playing it for yourself by running the code
- Complete the NimSumStrat() function
- Try running it by setting the Player1Strategy and/or Player2Strategy by un/commenting them
  - for [3,5,7], Player 1 should always win
  - for [3,5,6], Player 2 should always win

To-do

Things to do by the next workshop:

- Complete takeaway.py and nim.py, or look at our sol'ns
- Get comfortable with Python (if you found these two assignments difficult): Tutorial/Reference
- Think about (or look up) the best way to "solve" Tic-Tac-Toe from any position

Thanks for coming!