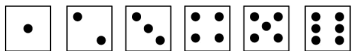


Attacking Die

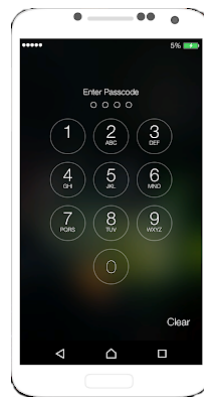


1	X					
2	X	X				
3	X	X	X			
4	X	X	X	X		
5	X	X	X	X	X	
6	X	X	X	X	X	X

```

03003802 996CB7BA 0EG0161B G0021C06
BA7CE203 G0030200 01208600 37D1AD00
1B7125G0 024FG002 53D03C00 AD722500
1BD03C00 887525C1 01A07700 37D14D00
B7125G0 024FG002 53D03C00 AD722500
BD03C00 887525C1 4F553F 5341424:
F4F3D41 4242434E 3D4A6 6469204:
16C2F4F 553D4553 414 4F3D414
425604 00312E30 0424 0003424
003042 4CC 024E4E4F 00B1D3:
254F1 21 309 8833B0CC 2957EE
3ECAA CB3E8EF DF038D7F A14217
2AA4D 04143B75 4F571C83 535C0:
7DED9 B57C659E C820EE07 FA49F
196DB 7D7F743D 9A36DD29 454E0
014D 410800C8 9A54E072 5A14C

```

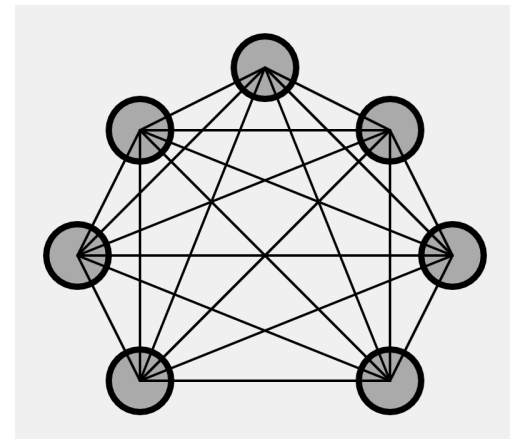
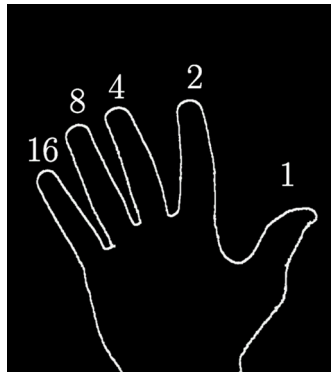
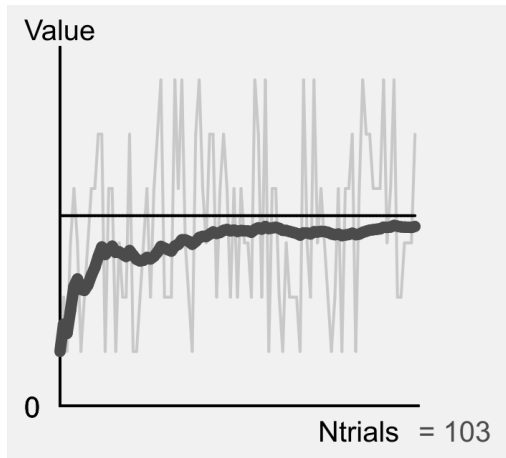


Crack the PIN
guess pin: 610

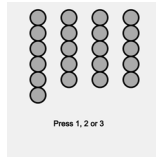
A match!

Progress bar (match may occur before scan is complete):

A Computer-Science Enriched Curriculum for Discrete Math

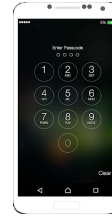


1. Games

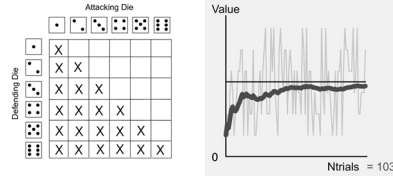


2. Counting/Combinatorics

- With application to cybersecurity

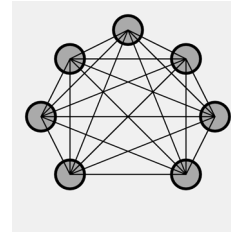


3. Probability



4. Connectivity

- Related to Graph/Network Theory



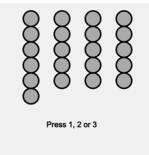
5. Iteration and recursion

$$N! = 1 \cdot 2 \cdot 3 \cdots (N - 1) \cdot N$$

6. Cryptography

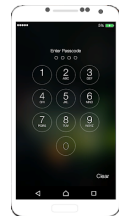
```
03003802 996CB7BA 0EG0161B G0021C06
BA7CE203 G0030200 01208600 37D14D00
1B7125G0 024FG002 53D03C00 AD722500
1BD03C00 887525C1 01A07700 37D14D00
B7125G0 024FG002 53D03C00 AD722500
BD03C00 887525C1 4F553300 53414240
F4F3D41 4242434E 3D4A6000 64659204
16C2F4F 553D4553 41424200 4F3D414
425604 00312E30 0A242400 0003424
003042 4C000000 024E4E4F 00B1D30
1254F1 21424209 8B33B0CC 2957EB0
3ECAA CB3588EF DF038D7F A14217
2AA4D 04143B75 4F571C83 535C00
7DED9 B57C659E C820EE07 FA49F
96DB 7D7F743D 9A36DD29 454E0
014D 410800C8 9A54E072 5A14C
```

1. Games



2. Counting/Combinatorics

- With application to cybersecurity

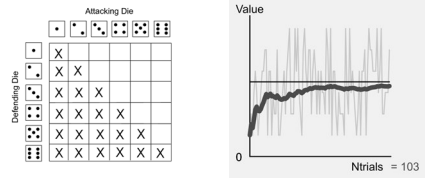


Crack the PIN
guess pin: 610

A match!

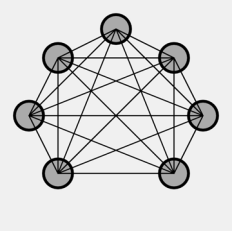
Progress bar (match may occur before scan is complete):

3. Probability



4. Connectivity

- Related to Graph/Network Theory



5. Iteration and recursion

$$N! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot (N - 1) \cdot N$$

6. Cryptography

```

03003802 996CB7BA 0EG0161B G0021C06
BA7CE203 G0030200 01208600 37D14D00
1B7125G0 024FG002 53D03C00 AD722500
1BD03C00 887525C1 01A07700 37D14D00
B7125G0 024FG002 53D03C00 AD722500
BD03C00 887525C1 4F5533 5341424:
F4F3D41 4242434E 3D4A6 6465204:
6C2F4F 553D4553 414 4F3D414
425604 0031230 424 000342:
003042 4C 024E4E4F 00B1D3:
1254F1 21 09 8B33B0CC 2957EB
3ECAA CB3EBEF DF038D7F A14217
2AA4D 04143B75 4F571C83 535C0:
7DED9 B57C659E C820EE07 FA49F
96DB 7D7F743D 9A36DD29 454E0
014D 410800C8 9A54E072 5A14C

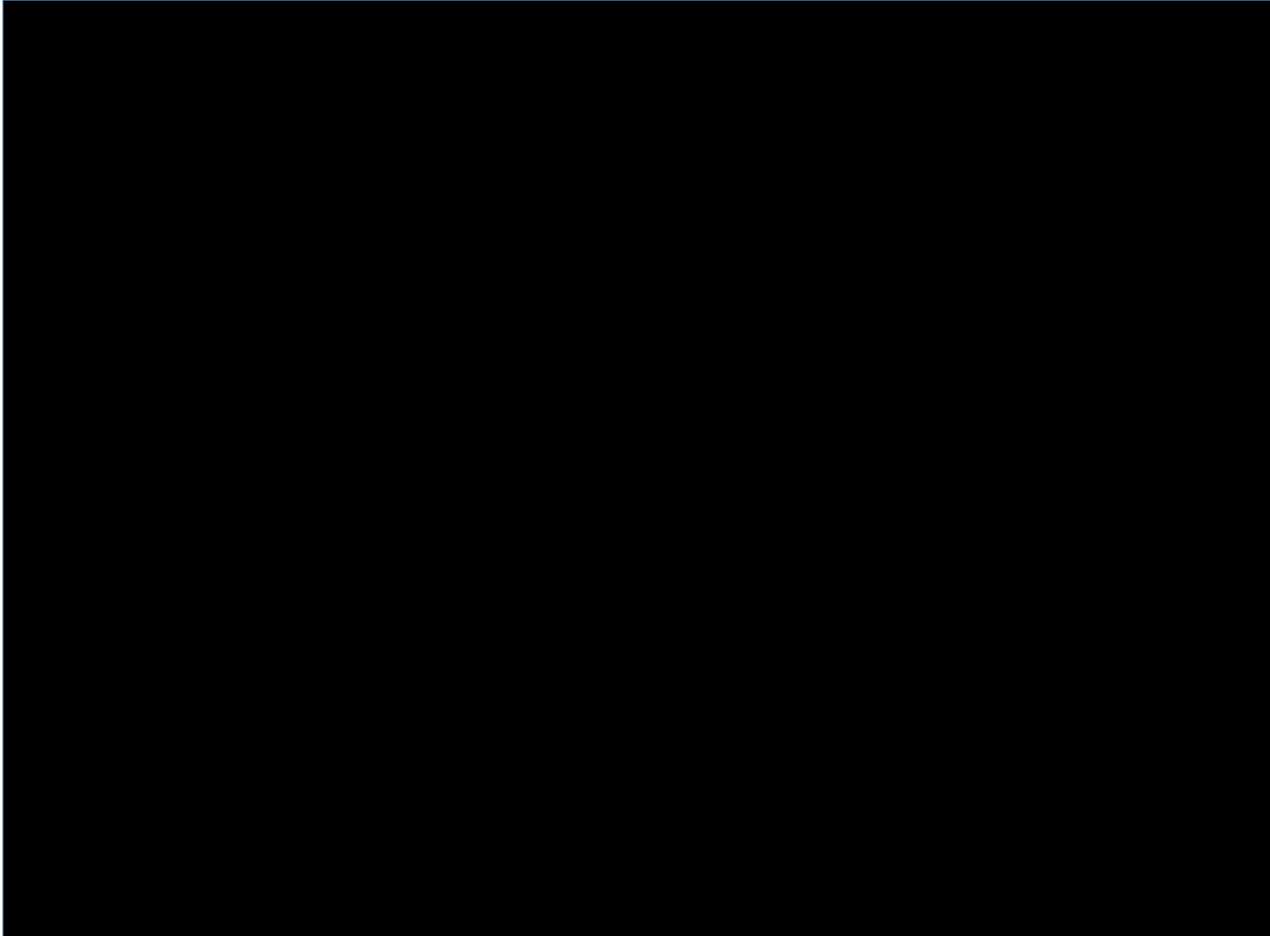
```

Example Activity: Thai 21

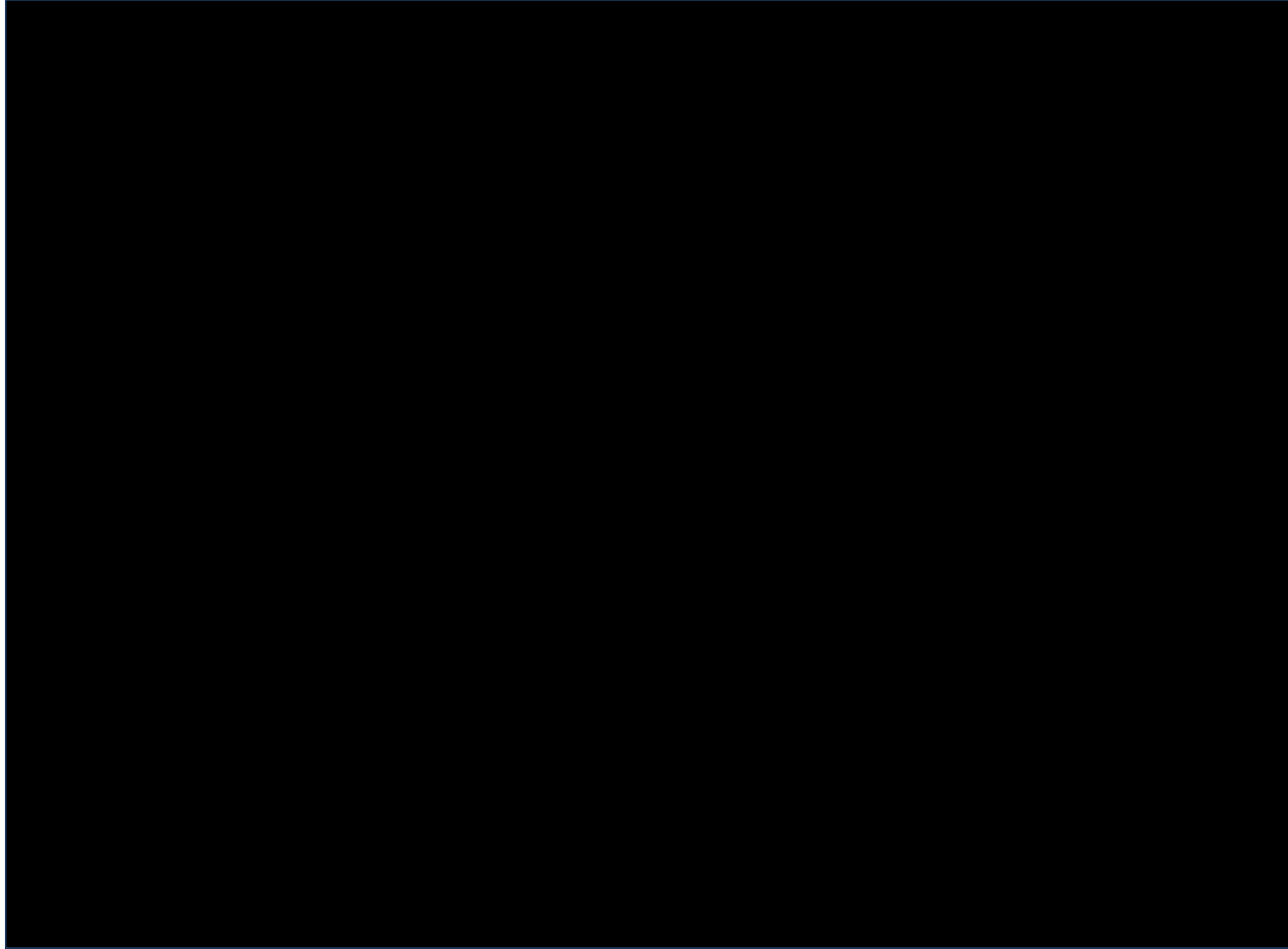


21 Flags

Each team takes 1-3 flags



Winner takes the last flag



Classroom

- Start hands on playing the game (~2 days)
- Alternate sessions:
 - Coding (screens up)
 - Debrief of coding, strategy of game, discussion (screens down)

Note:

Screens down content modified from San Diego State Discrete Math Project Collaborative

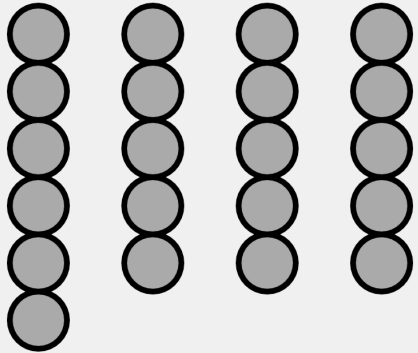


Thai 21 Coding

Part 1 – Put the rules of the game in the code

Part 2 – Keep track of turns

Part 3 – Player vs Computer (AI)



Press 1, 2 or 3

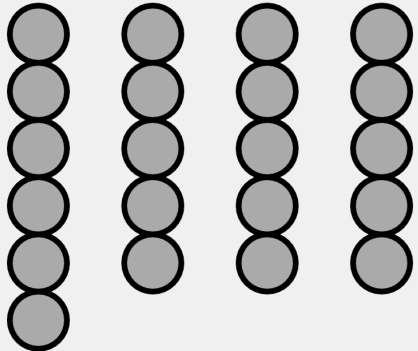
Total class time Parts 1-3:
~Three weeks

Thai 21 Coding

Part 1 – Put the rules of the game in the code

Part 2 – Keep track of turns

Part 3 – Player vs Computer (AI)



Press 1, 2 or 3

Link to activity:

<http://go.osu.edu/thai21>

Browser-based JavaScript Coding Interface



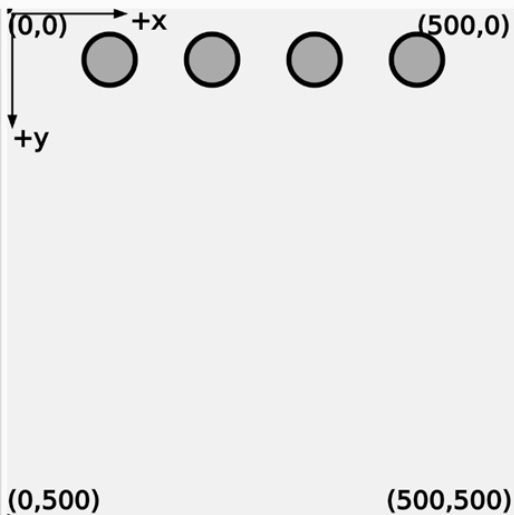
Auto-refresh

Thai 21 v0 part1 milestones by ChrisOrban

> sketch.js

```
1 showaxes = true;
2
3 function draw(){
4
5   clearscreen(240,240,240); // light gray background
6   display();
7
8   // add code below to remove flags by pressing 1,2,3
9
10  // Set the color of the flags
11  fill(170,170,170); // dark gray
12  stroke(0,0,0); // black edge
13
14  // Define the flags below
15  drawFlag(100,50); // x,y
16  drawFlag(200,50);
17  drawFlag(300,50);
18  drawFlag(400,50);
19
20
21 } // end draw() DO NOT ADD ANY CODE AFTER THIS LINE!!!
22
23
```

Preview



OBJECTIVE: Draw one flag **STATUS:** ✓

OBJECTIVE: Draw more than one flag **STATUS:** ✓

OBJECTIVE: Draw more than four flags **STATUS:** ✗

OBJECTIVE: Draw 21 flags **STATUS:** ✗

OBJECTIVE: Add a way to remove 1, 2 or 3 flags **STATUS:** ?

Compatible with:



Requires physical keyboard!





Auto-refresh

Thai 21 v0 part1 milestones by ChrisOrban



sketch.js

Preview

```
1 showaxes = false;
2
3 function draw(){
4
5   clearscreen(240,240,240); // light gray background
6   display();
7
8   // add code below to remove flags by pressing 1,2,3
9
10  // Set the color of the flags
11  fill(170,170,170); // dark gray
12  stroke(0,0,0); // black edge
13
14  // Define the flags below
15  drawFlag(100,50); // x,y
16
17 } // end draw() DO NOT ADD ANY CODE AFTER THIS LINE!!!
18
19
```



OBJECTIVE: Draw one flag **STATUS:**

OBJECTIVE: Draw more than one flag **STATUS:**

OBJECTIVE: Draw more than four flags **STATUS:**

OBJECTIVE: Draw 21 flags **STATUS:**

OBJECTIVE: Add a way to remove 1, 2 or 3 flags **STATUS:**

Here is some code you can add to plot four flags in a row:

```
y = 50;  
for (x = 100 ; x <= 400; x += 100) {  
    drawFlag(x,y);  
}
```

Copy code to clipboard



Auto-refresh

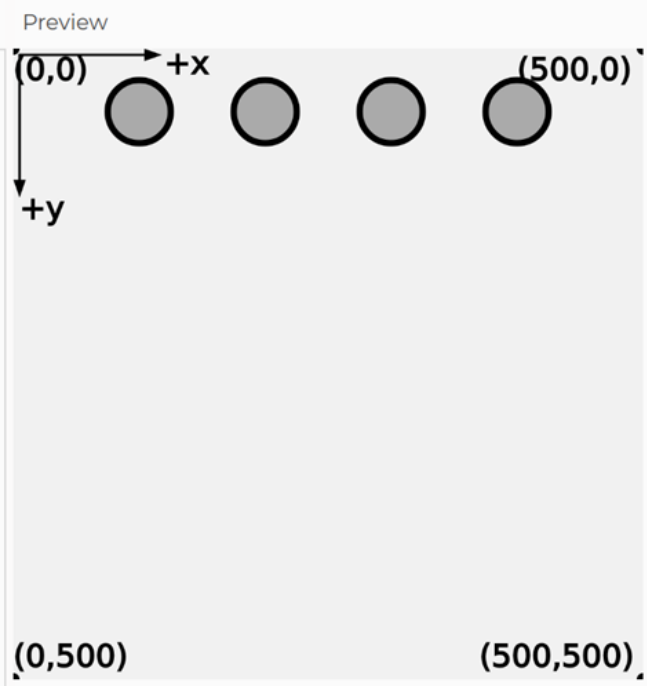
Thai 21 v0 part1 milestones by ChrisOrban

> sketch.js

```

1 showaxes = true;
2
3 function draw(){
4
5   clearscreen(240,240,240); // light gray background
6   display();
7
8   // add code below to remove flags by pressing 1,2,3
9
10  // Set the color of the flags
11  fill(170,170,170); // dark gray
12  stroke(0,0,0); // black edge
13
14  // Define the flags below
15  y = 50;
16  for (x = 100 ; x <= 400; x += 100) {
17    drawFlag(x,y);
18  }
19
20
21 } // end draw() DO NOT ADD ANY CODE AFTER THIS LINE!!!
22
23

```



- OBJECTIVE:** Draw one flag **STATUS:** ✓
- OBJECTIVE:** Draw more than one flag **STATUS:** ✓
- OBJECTIVE:** Draw more than four flags **STATUS:** ✗
- OBJECTIVE:** Draw 21 flags **STATUS:** ✗
- OBJECTIVE:** Add a way to remove 1, 2 or 3 flags **STATUS:** ?

For loop!

Fast Forward



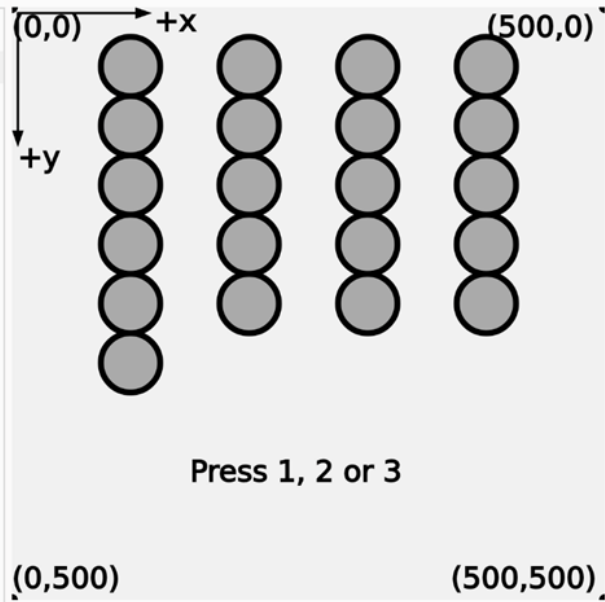
> sketch.js

```

1 showaxes = true;
2 Ntotalflags = 21;
3
4 function draw(){
5
6   clearscreen(240,240,240); // light gray background
7   display();
8
9   // add code below to remove flags by pressing 1,2,3
10  drawText('Press 1, 2 or 3',150,400);
11  if ((keyIsPressed == true) && (key == 1)) Ntotalflags -= 1;
12  if ((keyIsPressed == true) && (key == 2)) Ntotalflags -= 2;
13  if ((keyIsPressed == true) && (key == 3)) Ntotalflags -= 3;
14
15  // Set the color of the flags
16  fill(170,170,170); // dark gray
17  stroke(0,0,0); // black edge
18
19  // Define the flags below
20  Nflags_so_far = 0;
21  for (y = 50; y <= 400; y += 50) {
22    for (x = 100 ; x <= 400; x += 100) {
23      if ( Nflags_so_far < Ntotalflags ) {
24        drawFlag(x,y);
25        Nflags_so_far += 1;
26      }
27    }
28  }
29
30 } // end draw() DO NOT ADD ANY CODE AFTER THIS LINE!!!

```

Preview



- OBJECTIVE:** Draw one flag **STATUS:** ✓
- OBJECTIVE:** Draw more than one flag **STATUS:** ✓
- OBJECTIVE:** Draw more than four flags **STATUS:** ✓
- OBJECTIVE:** Draw 21 flags **STATUS:** ✓
- OBJECTIVE:** Add a way to remove 1, 2 or 3 flags **STATUS:** ?

Where is the math?

- .Coordinate system
- .Transformation of coordinates in `drawFlag(x,y);`
- .Boolean logic of if statements
 - Encoding the rules of the game
- .Visualizing the strategy with rows of 4
- .Later we will write code so a computer opponent follows the strategy
- .The strategy involves calculating the remainder of remaining flags/4

It's more than one way to solve problems.

It's collaborative

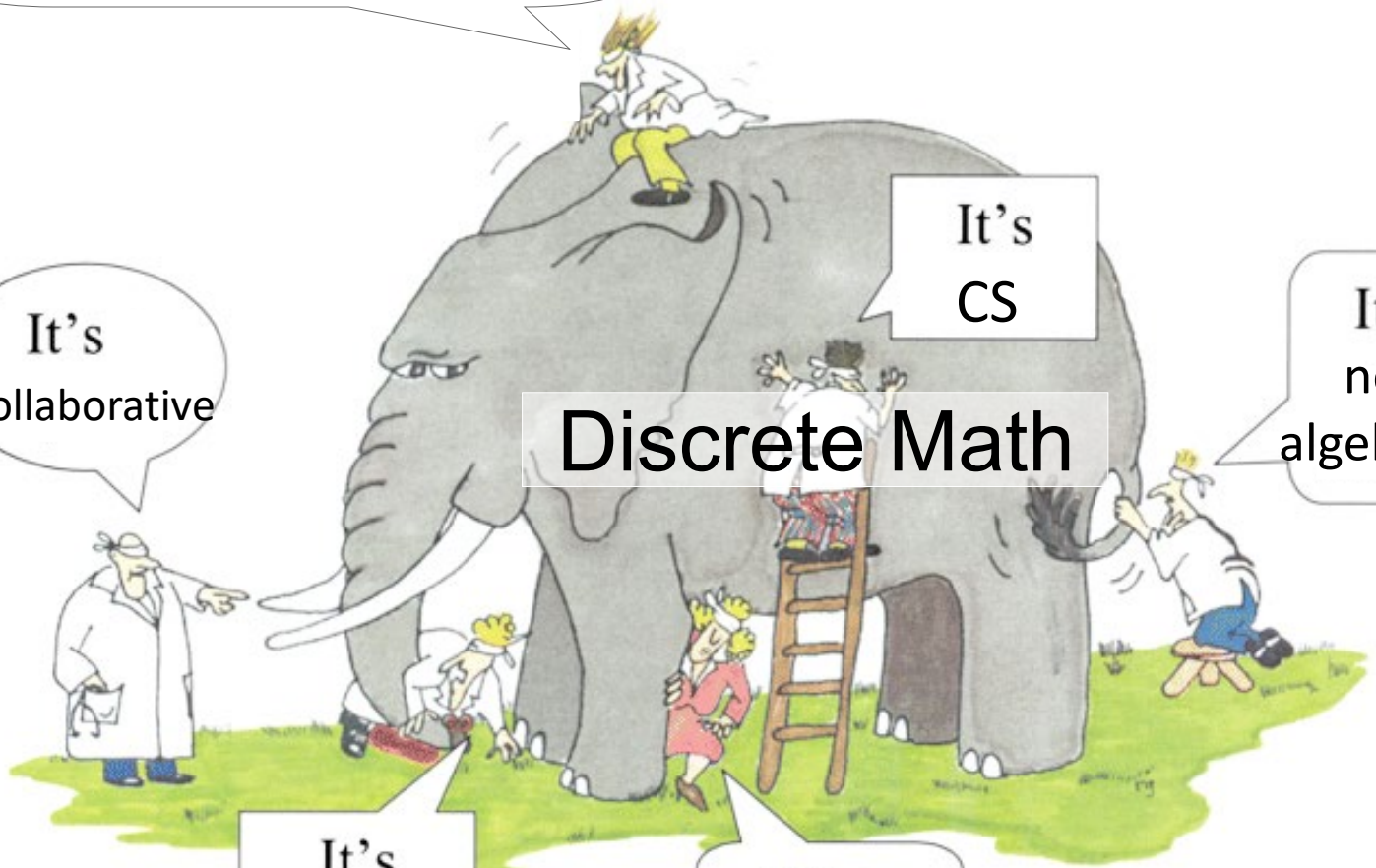
It's CS

It's not algebra 2

Discrete Math

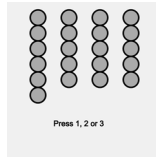
It's hands on

It's math

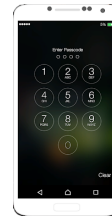


Year Long Curriculum

1. Games



2. Counting/Combinatorics



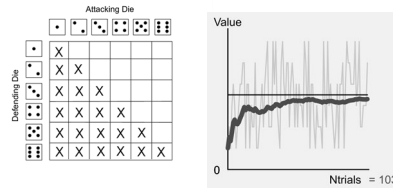
Crack the PIN
guess pin: 610

A match!

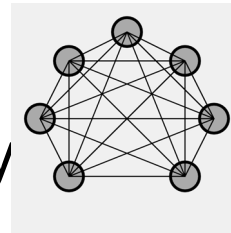
Progress bar (match may occur before scan is complete):

- With application to cybersecurity

3. Probability



4. Connectivity



- Related to Graph/Network Theory

5. Iteration and recursion

$$N! = 1 \cdot 2 \cdot 3 \cdots (N - 1) \cdot N$$

6. Cryptography

```

03003802 996CB7BA 0EG0161B G0021C06
BA7CE203 G0030200 01208600 37D14D00
1B7125G0 024FG002 53D03C00 AD722500
1BD03C00 887525C1 01A07700 37D14D00
B7125G0 024FG002 53D03C00 AD722500
BD03C00 887525C1 4F553300 53414240
F4F3D41 4242434E 3D4A6000 64652040
16C2F4F 553D4553 41A07700 4F3D414
425604 00312E30 0A242421 0003424
003042 4C000000 024E4E4F 00B1D30
1254F1 21A07700 8B33B0CC 2957BE0
3ECAA CB3E88EF DF038D7F A142170
2AA4D 04143B75 4F571C83 535C000
7DED9 B57C659E C820EE07 FA49F00
96DB 7D7F743D 9A36DD29 454E000
014D 410800C8 9A54E072 5A14C00

```

Questions?

Comments?

Feel free to email me at urban.14@osu.edu

Link to activity: <http://go.osu.edu/thai21>

Extra Slides

UNIVERSITY OF CALIFORNIA SAN DIEGO

CALIFORNIA STATE UNIVERSITY SAN MARCOS

Math is No Longer a Four-Letter Word: A Mixed Methods Study of Two Non-Traditional
Fourth-Year Mathematics Classes

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Education

in

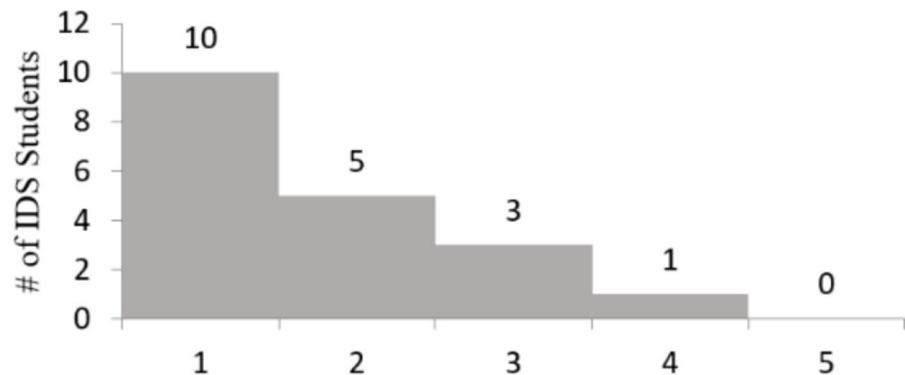
Educational Leadership

by

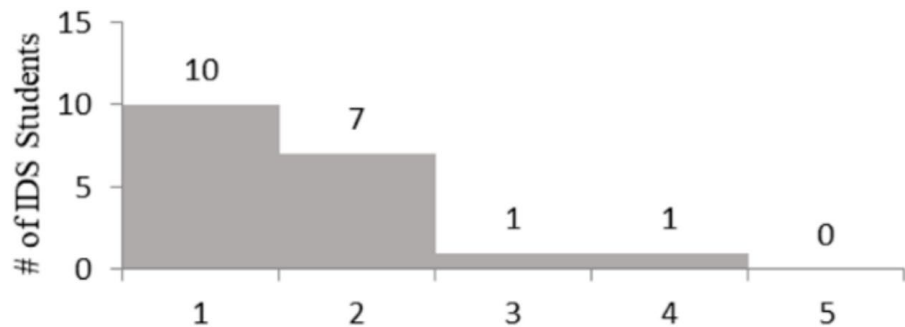
Erica Heinzman

Data Science

This class is enjoyable for me



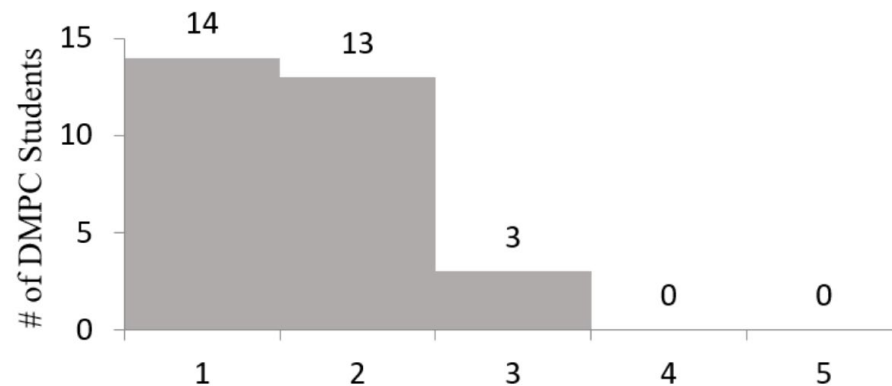
I believe there is more than one way to solve a problem



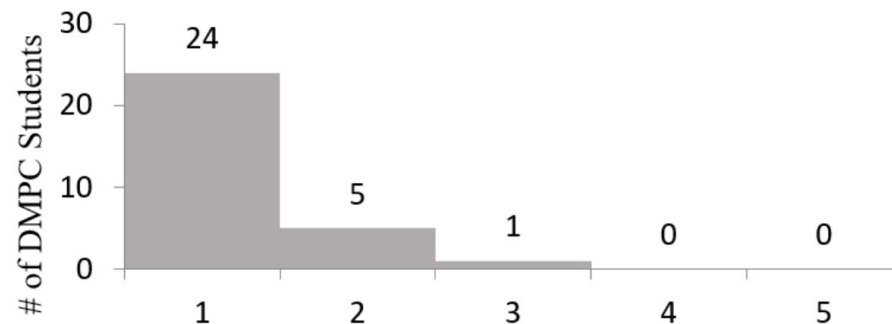
1 = "strongly agree"

Discrete Math (no CS)

This class is enjoyable for me

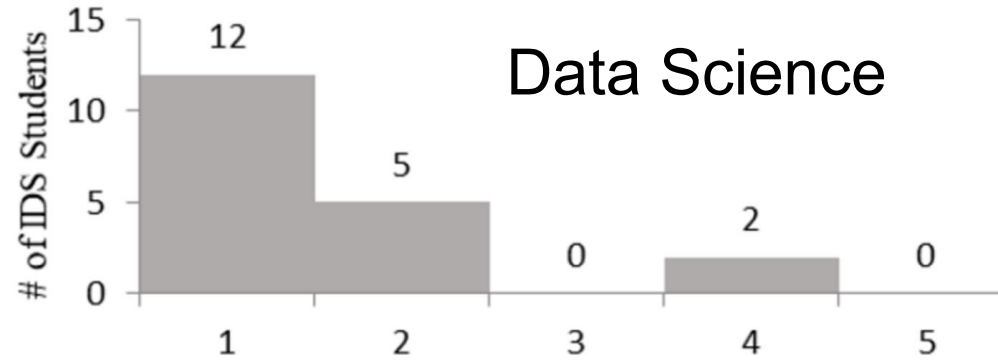


I believe there is more than one way to solve a problem

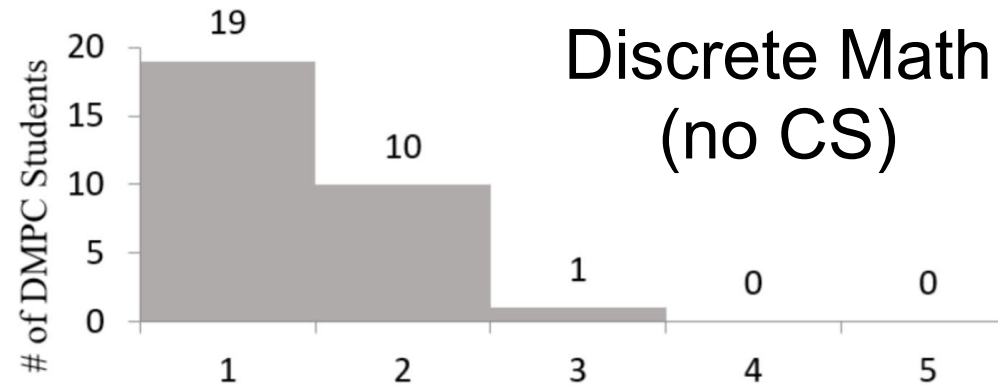


5 = "strongly disagree"

**I am confident about learning the content
in this course**



**I am confident about learning the content
in this course**



1 = "strongly agree"

5 = "strongly disagree"

Brief Summary

- .Data Science and Discrete math are both efforts to change what it feels like to be in a math class
- .This is in much the same spirit as the quantitative reasoning effort in Ohio
- .Students work with each other to solve problems