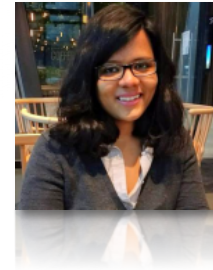
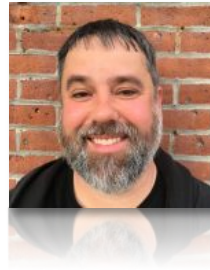
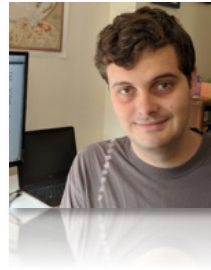
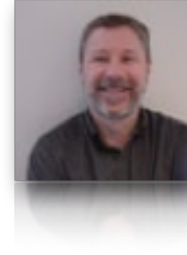
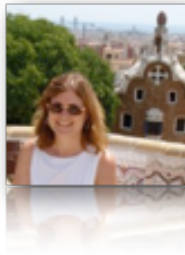


Computer Science Faculty



- Diverse interests and background!
- Short “lightning” talks today
 - Should follow-up individually for more info!

Summer Research in CS

- Spend 8-10 weeks working with individual faculty member on specific research problems
- Live in dorm with other students (not just CS students)
- Work in computer lab (mostly)
- “Flexible” schedule
- Social events (epic canoe trip)
- Chance to get to know us personally
- No classes = more time for faculty to focus on research



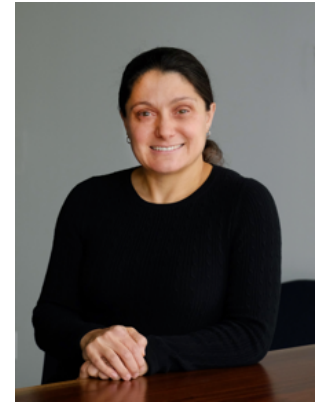
Honors Thesis in CS

- Spend senior year working on independent research project
- Advised by faculty member
- Fall is spent reading
 - Become an expert in chosen area
- WSP is spent working...hard
- Spring is spent writing
 - Gathering results
 - Writing thesis
- Present results in May in public presentation
- Lots of work, but very rewarding
- Highly recommended for those considering grad school

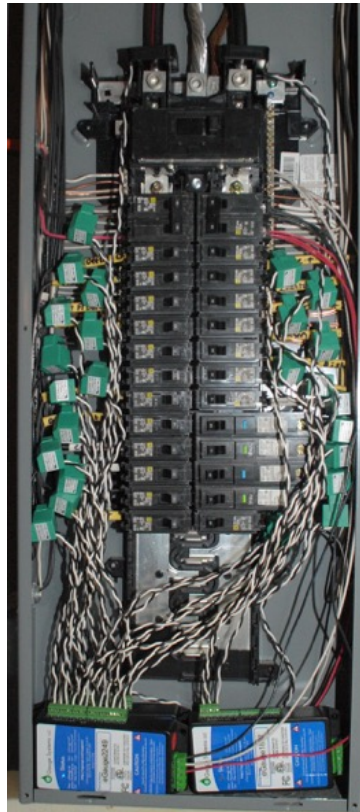


Faculty research interests

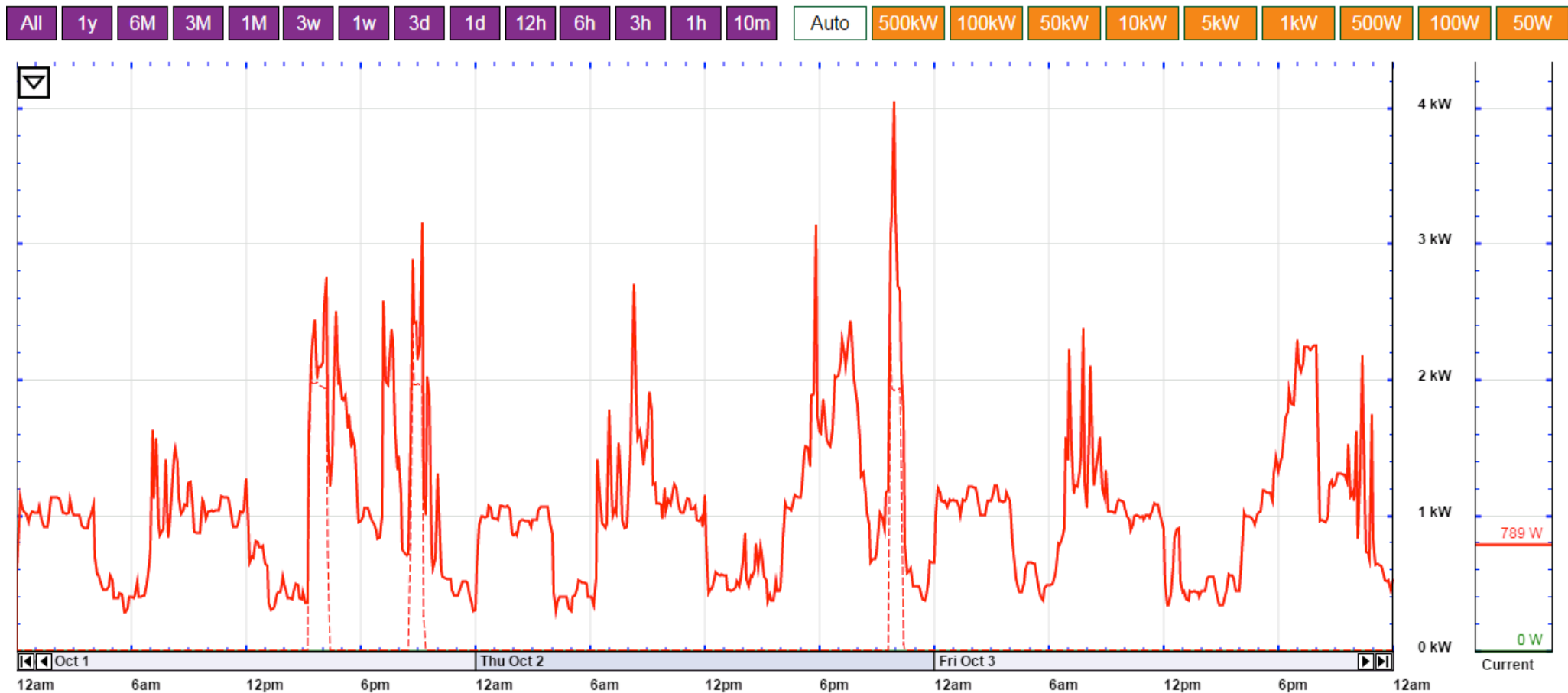
Jeannie Albrecht



- Jeannie works in Distributed System, Networks, IoT
- Lately, Smart Buildings
- Also K-8 CS ed



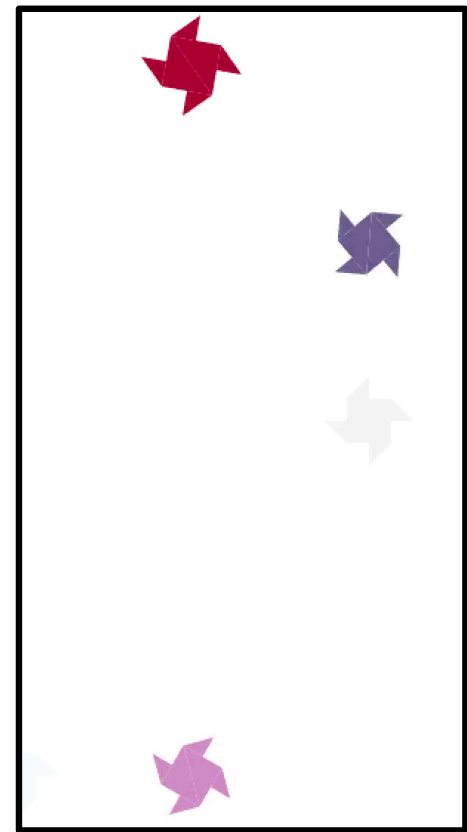
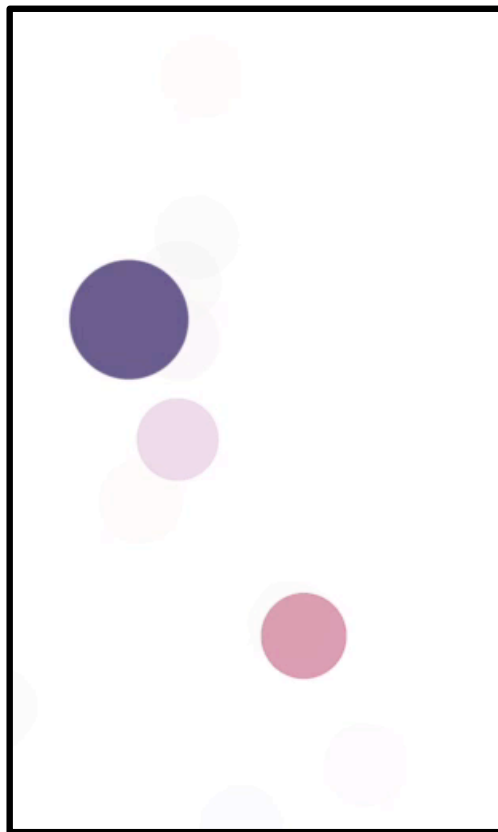
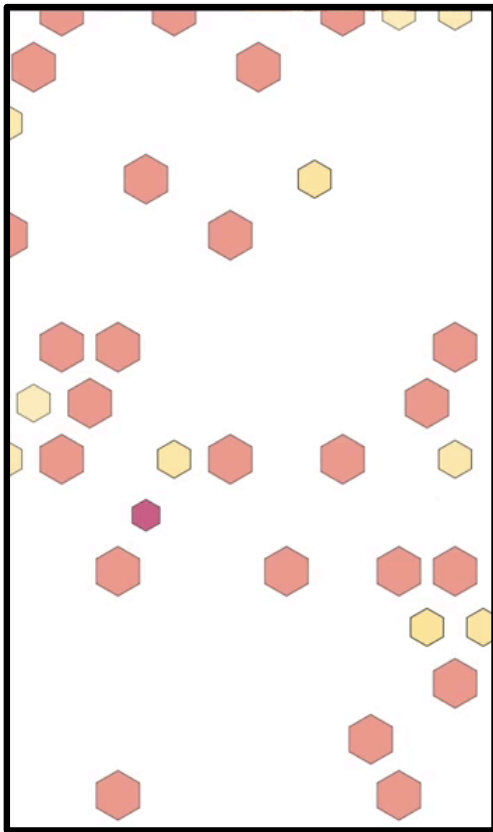
Traditional Energy Use Visualizations



- Need something that is more intuitive...

Research Goal: Make Buildings *and* Occupants Smart(er)

- Use *ambient* visualizations to help occupants better understand energy usage (in kitchen)
- Research students need to have at least taken CS 136



Kitchen Simulator



Duane Bailey



- Duane is a computer architect
- Works with hardware
- Often advises projects that use FPGAs
- FPGAs contain programmable logic blocks and interconnection circuits
- They can be programmed/ reprogrammed to the required functionality after manufacturing
- (kinda like custom built hardware)
- Need at least CSCI 237



Dan Barowy: Spreadsheet Support for End-to-End Data Analysis



data wrangling

FlashRelate

PLDI '15



data cleaning

CheckCell

OOPSLA '14



debugging

ExceLint

OOPSLA '18

Background req'd: **136!!! 334** or **376** preferred, but not required.

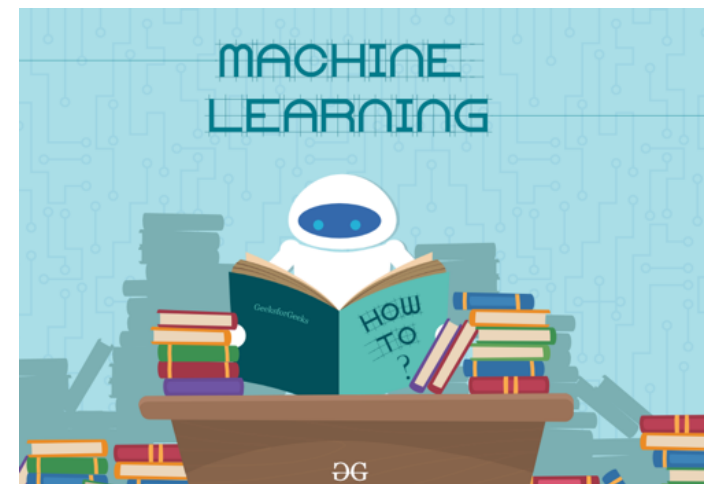
Daniel Barowy, Assistant Professor

Williams

Andrea Danyluk



- Andrea works in Machine Learning
- Projects involve “Supervised Inductive Machine Learning”
- Examples:
 - Learn a predictive model from labeled training data
 - Will this cancer patient experience a recurrence after initial treatment? Is this banknote authentic or not? Who is the author of this Amazon review?
 - Recent excitement
 - Availability of data. Increased processing power.
- Interesting research problems
 - New algorithms for new types of problems. Model interpretability for humans. Models that balance accuracy and cost. And many more.



Supervised Inductive Machine Learning: Summer Research at Williams with Andrea

Application-Driven



Model expressivity



Identify individuals



Identify keys to successful
teaching and learning

What it's like:

- Learn application domain
- Learn ML algorithms and empirical methodology
- Gather, clean, and featurize data (at least CSCI 136)
- Run experiments
- Have fun!

Steve Freund



- Steve works in Programming Languages
- Probably need to have taken at least 136 and probably 334



Google Docs

2 Days Ago

UPDATE

* Bug fixes and performance improvements



Amazon Prime Video

Jan 29, 2018

UPDATE

Bug fixes and performance improvements



Google Sheets

2 Days Ago

UPDATE

* Bug fixes and performance improvements



Netflix

3 Days Ago

[UPDATE](#)

Bug fixes



HBO GO: Stream with TV Package

3 Days Ago

[UPDATE](#)

- Bug fixes and performance improvements.
-

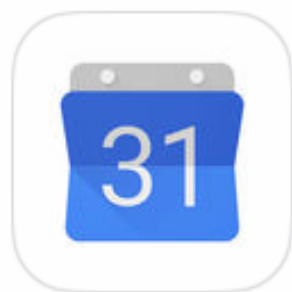


Instagram

Jan 29, 2018

[UPDATE](#)

General bug fixes and performance improvements



Google Calendar

Today

UPDATE

- Bug fixes and performance improvements.
-

Traditional Software Process



With Better Programming Tools

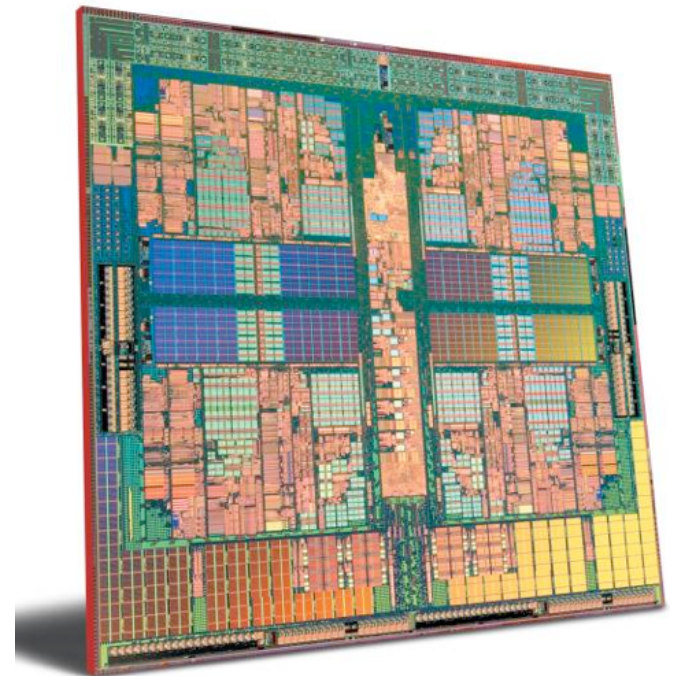


Better:

- Programming Languages
- Verification Techniques
- Debuggers
- Programming Environments
- ...

Work on Programming Languages And Analysis Tools

- language design
- theoretical foundations
- proving theorems
- systems development
- performance modeling
- experimental validation



Hi folks! My name's Iris Howley.

I'm currently at the AAAI Conference on Artificial Intelligence, Ethics and Society with a couple of my Williams research students.

There's more info on doing research with me on **my website** under "Info for Students" so check that out & contact me to talk if you're interested.

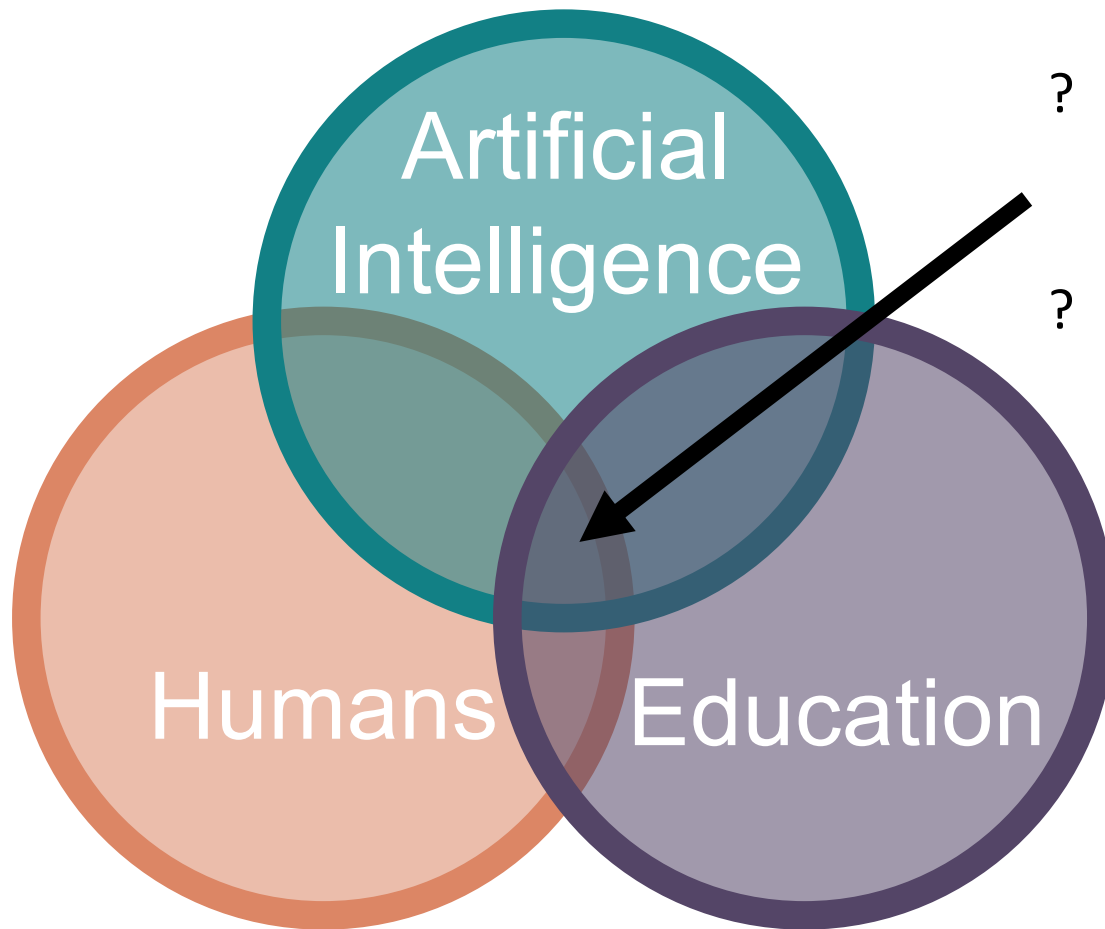
MIT Tech Review recently wrote up some related work on my topic, here:

www.bit.ly/Alexplain



Iris Howley, iris@cs.williams.edu
Assistant Professor of Computer Science

Iris works at the intersection of Human-Computer Interaction, Artificial Intelligence, and Education. Here are some questions she thinks about!

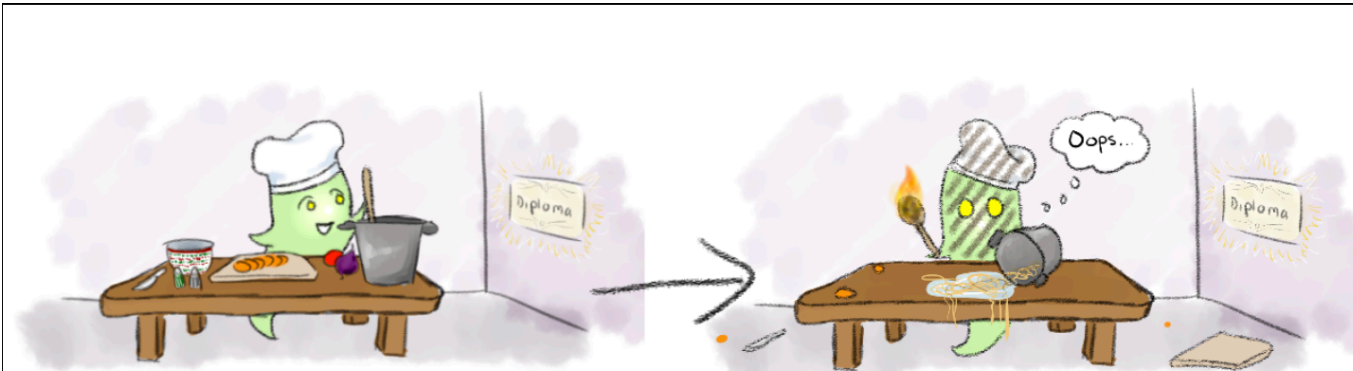


- ? How do users of ed tech systems understand the algorithms underlying their systems?
- ? Does increased algorithmic understanding lead to more informed decision-making?

Skills We Learn in this Research

- *Understand humans*: Interviews, User studies
- *Build systems to explain AI*: Javascript, web dev, computer languages
- *Evaluate*: Survey measures, behavioral measures

Iterative User-Centered Design



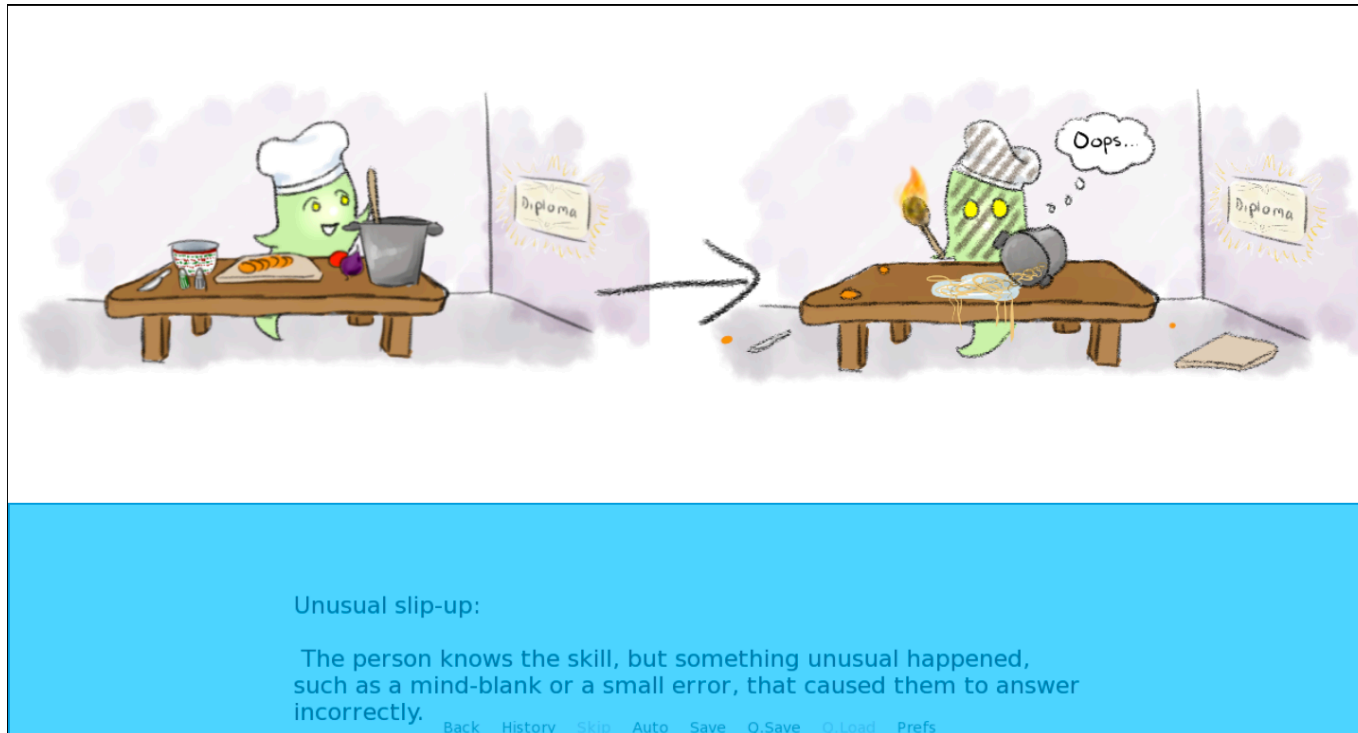
Unusual slip-up:

The person knows the skill, but something unusual happened, such as a mind-blank or a small error, that caused them to answer incorrectly.

And now for some example of past summer research student work from my group!



Iterative User-Centered Design



Iterative User-Centered Design

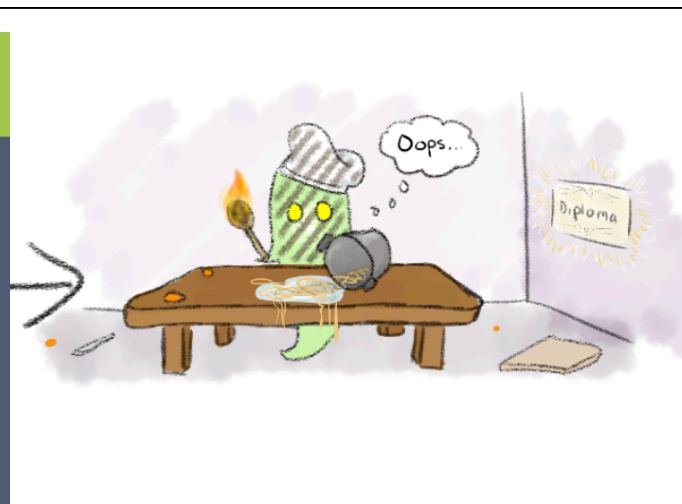
We associate the initial level of mastery with this vial to begin with.



Something unusual happened, error, that caused them to answer

Save Q.Save Q Load Prefs

initial learned slipped guessed next



They **Guess**, continuing straight. The journey will take even longer now.

In BKT, when a student answers incorrectly, this may be either a **Slip** or a **Guess**. A **Slip** predicts that the student knows the information, but made a mistake answering the question. A **Guess** predicts that the student does not know the information.

Left, left, right, left...

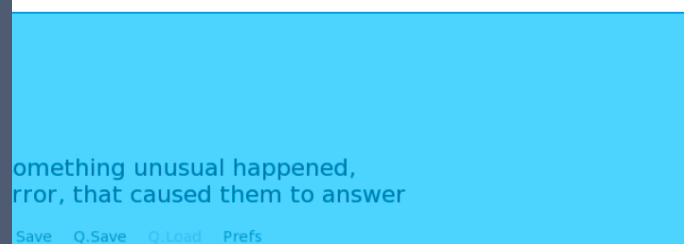
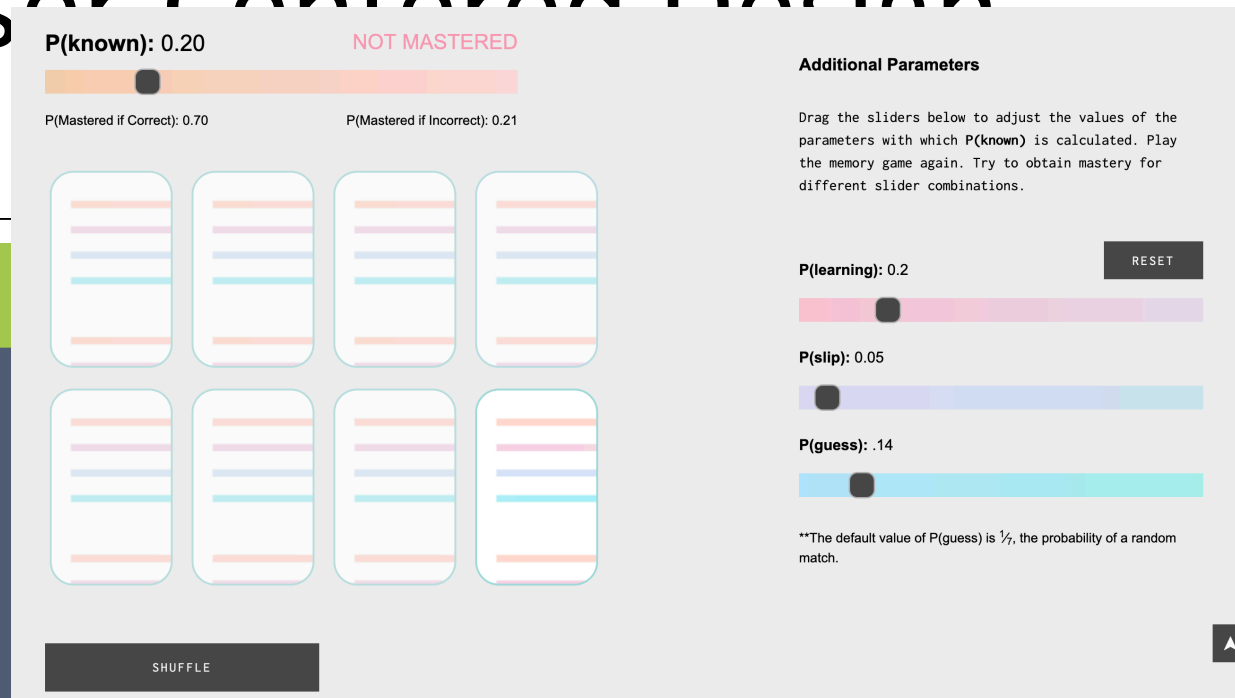
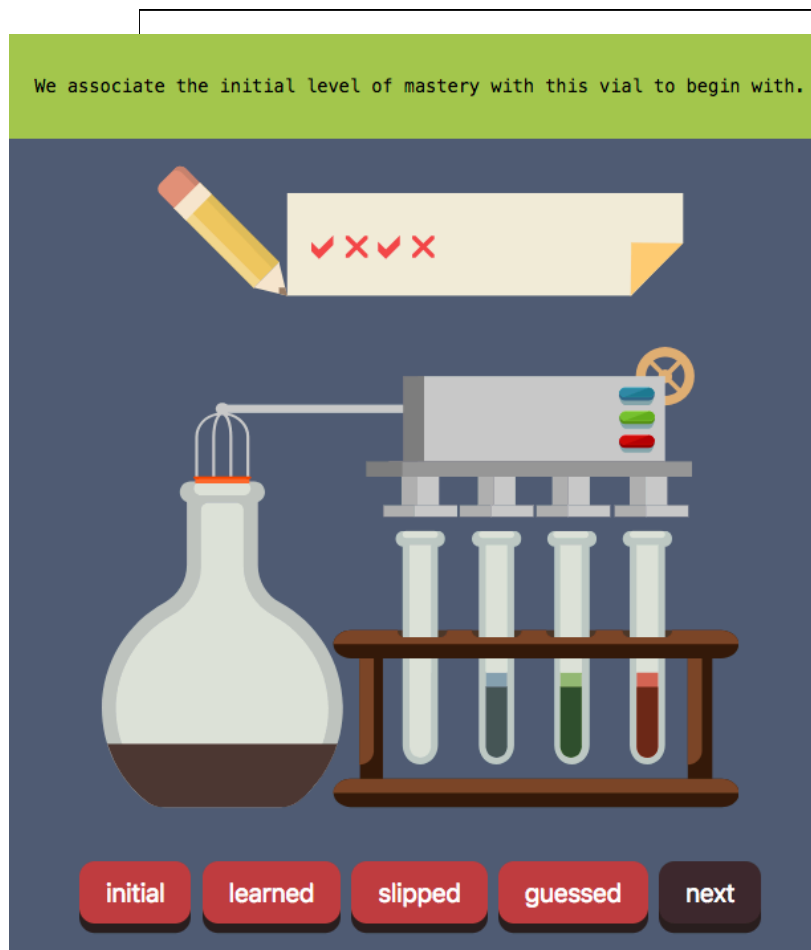
They're on their way again! The destination is approaching and it looks like the driver does know how to get there after all!

In BKT, many correct answers increase the odds that the student knows the skill, until eventually, a certain cutoff is reached, and the algorithm determines that the skill has been learned.

They've arrived!

The trip took a little longer than expected, with the accidental **Slip** and the **Guess** the driver made, but in the end, they've shown that they know how to get to their destination. Their **Prior Knowledge** of the route set them up for a good start, and every **Correct Choice** they made helped them overcome their small mistakes.

Iterative User Centered Design



Iterative User Centered Design

We associate the initial level of mastery with this vial to begin with.

BKT BALLOON SIMULATOR

Drag/click the sliders on the right to adjust the parameters and help the balloon rise. Explore on your own or use the following prompts as a guide.

1. Find two different parameter combinations that will result in mastery if the student answers correctly.
Hint: make $P(\text{learned if correct}) \geq 0.95$ and press "answer correct" to verify your results.

MORE

2. What happens to $P(\text{learned if correct})$ and $P(\text{learned if wrong})$ if $P(\text{guess})$ and/or $P(\text{slip})$ exceeds 0.5?

MORE

3. What happens to $P(\text{learned})$ if the student answers incorrectly? *Hint: compare $P(\text{learned if wrong})$ with $P(\text{known})$ (aka. your previous $P(\text{learned})$).*

MORE

4. Keep exploring! Can you find any other flaws or interesting characteristics of BKT?



$P(\text{known})$: 0.20

NOT MASTERED



$P(\text{Mastered if Correct})$: 0.70

$P(\text{Mastered if Incorrect})$: 0.21



$P(\text{known})$: 0.5

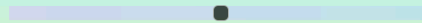
RESET



$P(\text{will learn})$: 0.5



$P(\text{slip})$: 0.5



$P(\text{guess})$: 0.5



Remember, $P(\text{learned})$ depends on whether the student answers correctly and this probability becomes the new value for $P(\text{known})$. Simulate student responses by choosing an answer button below.

$P(\text{learned if correct})$: 0.75 $P(\text{learned if wrong})$: 0.75

ANSWER CORRECT

ANSWER WRONG

Hint: hover over the parameters to see how they impact the simulator.

Additional Parameters

Drag the sliders below to adjust the values of the parameters with which $P(\text{known})$ is calculated. Play the memory game again. Try to obtain mastery for different slider combinations.

$P(\text{learning})$: 0.2

RESET



$P(\text{slip})$: 0.05



$P(\text{guess})$: .14



****The default value of $P(\text{guess})$ is $\frac{1}{7}$, the probability of a random match.**

that the student knows the skill, until eventually, a certain cutoff is reached, and the algorithm determines that the skill has been learned.

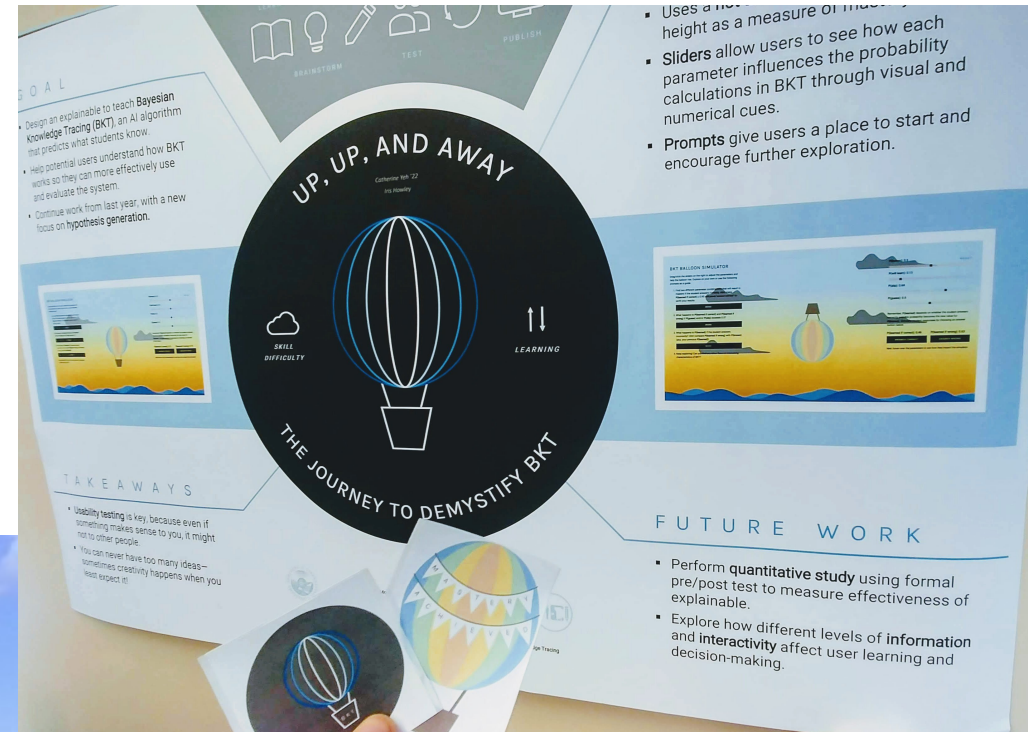
They've arrived!

The trip took a little longer than expected, with the accidental **Slip** and the **Guess** the driver made, but in the end, they've shown that they know how to get to their destination. Their **Prior Knowledge** of the route set them up for a good start, and every **Correct Choice** they made helped them overcome their small mistakes.



Fun

Stickers coordinating with our research posters



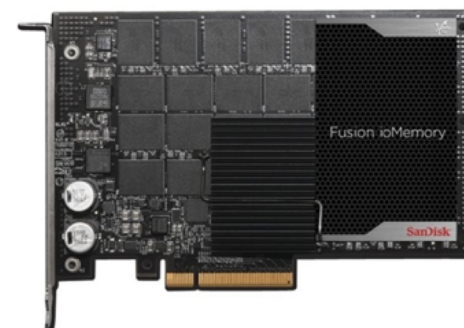
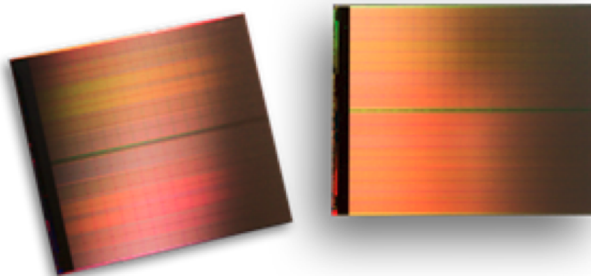
Field Farm Field Trip

Making stickers to match our research posters!

Bill Jannen

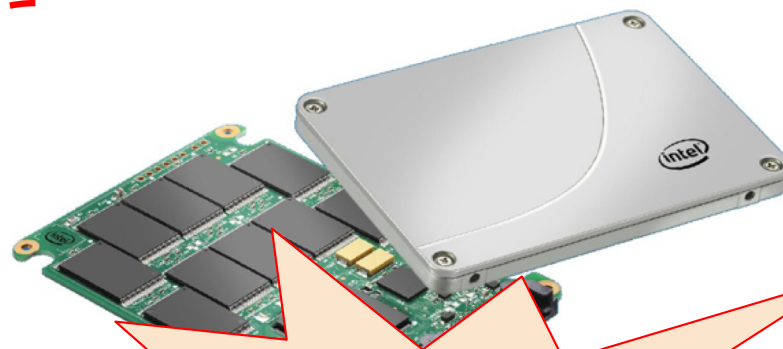


Bill works in Storage Systems

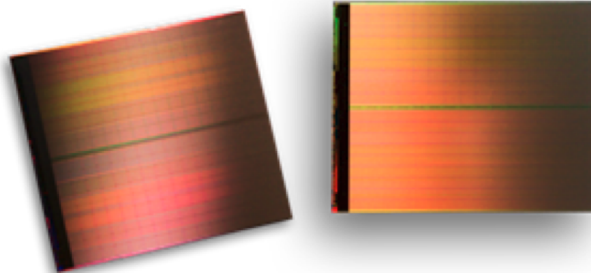
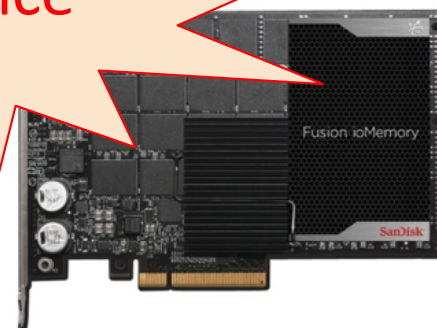


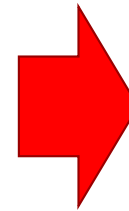
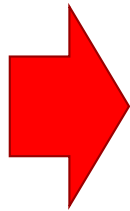
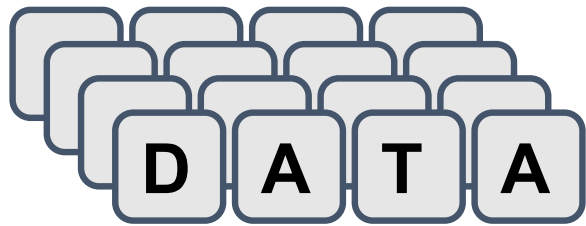


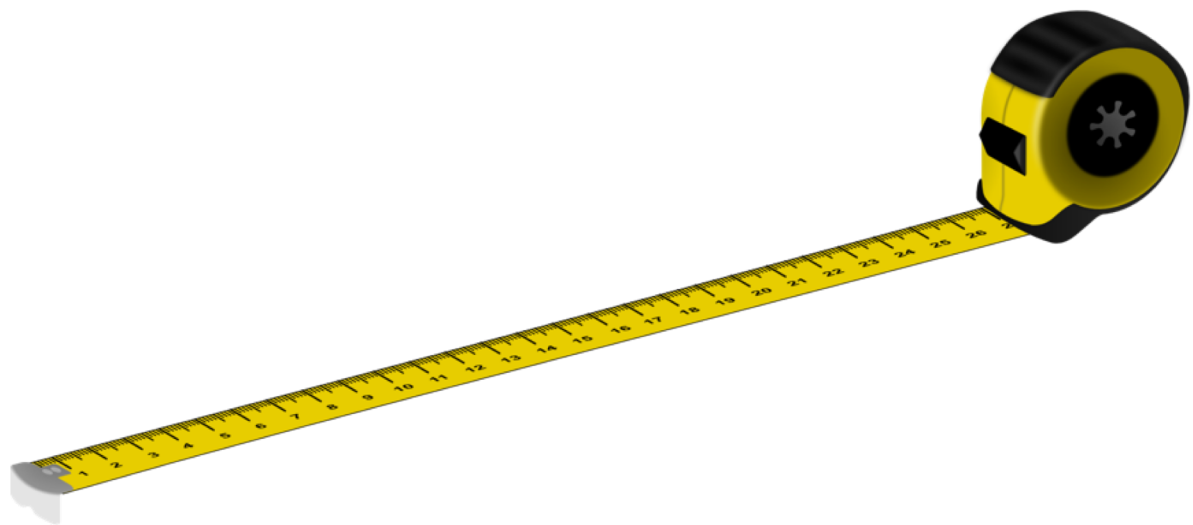
Capacity

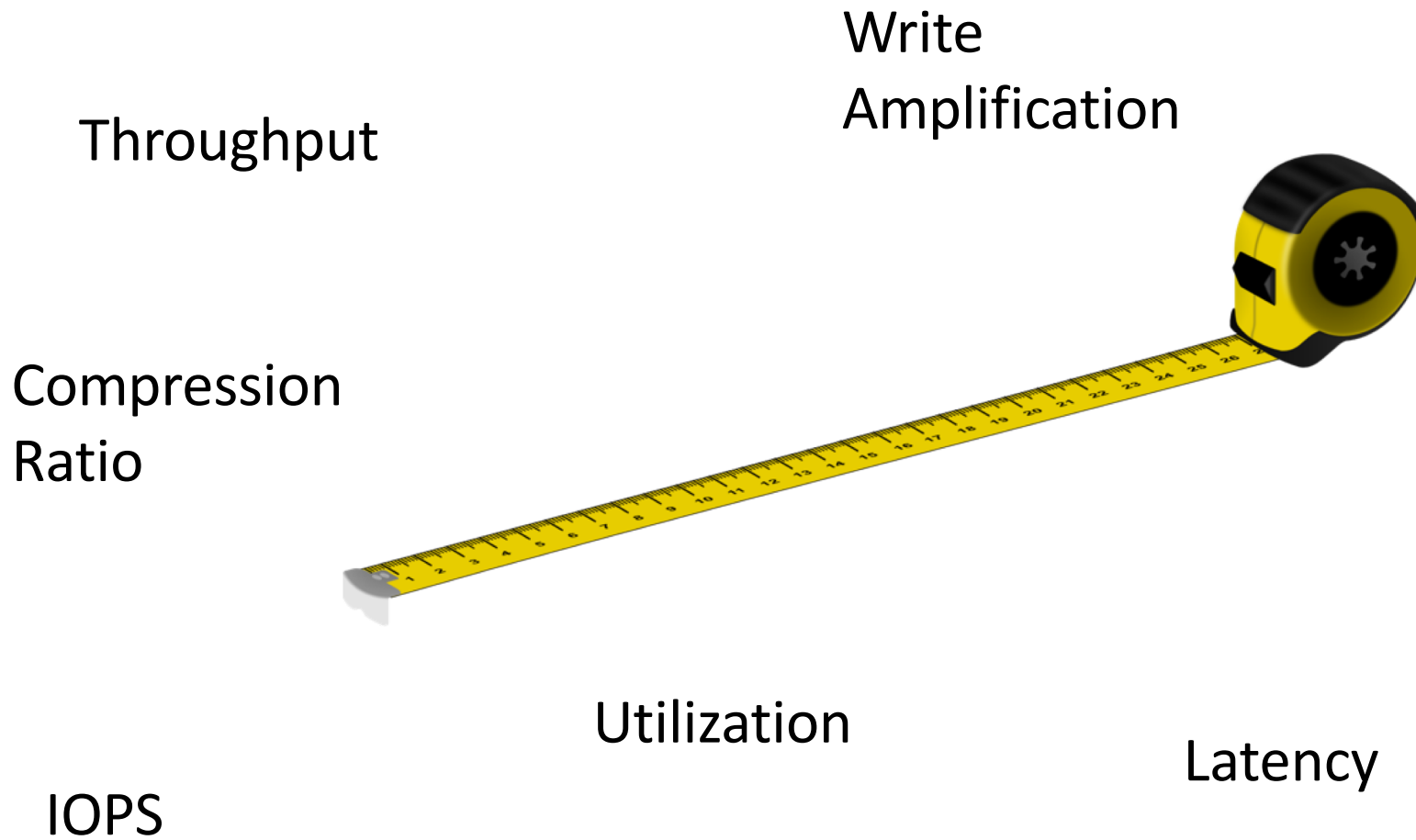


Performance









Storage Research Goals

Bill tries to build systems that help users:

- Read and write their data faster (**performance**)
- Store more data in the same physical device (**capacity**)
- Ensure that data is never lost (**safety**)

He's interested in topics like:

- New HDD and SSD technologies
- File systems
- Key-value stores
- Data deduplication
- Write optimized data structures
- Filters

His research group has a website with their project info: www.betrfs.org

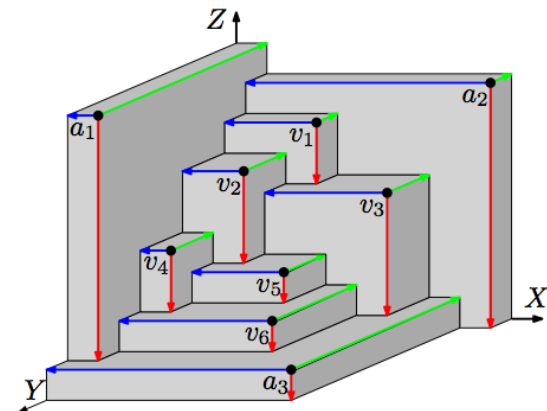
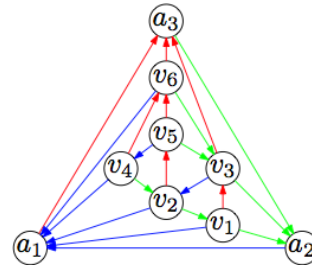
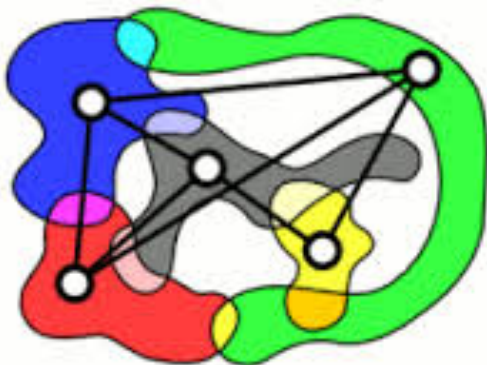
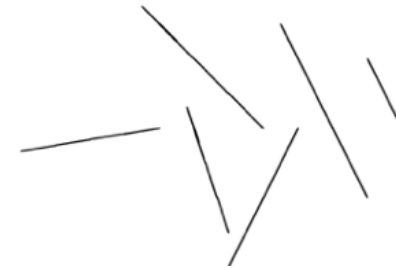
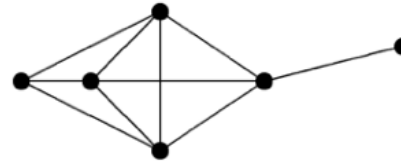
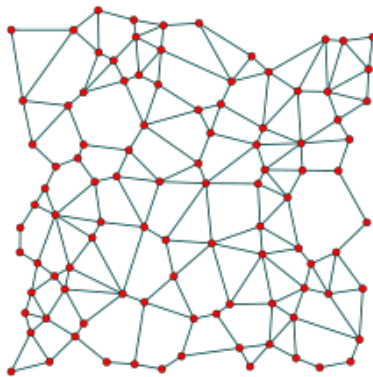
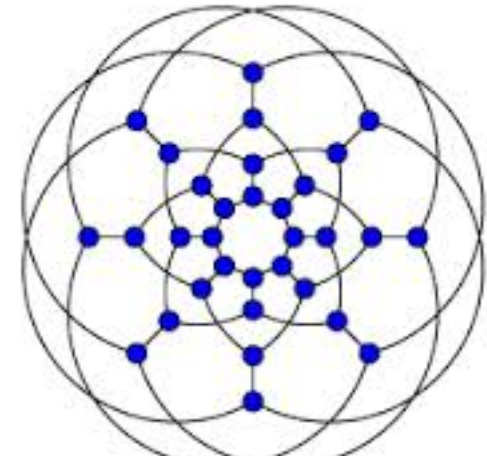
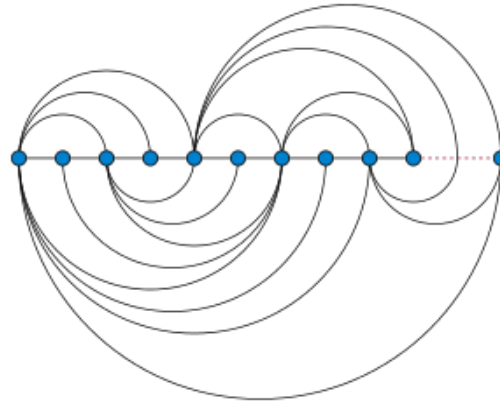
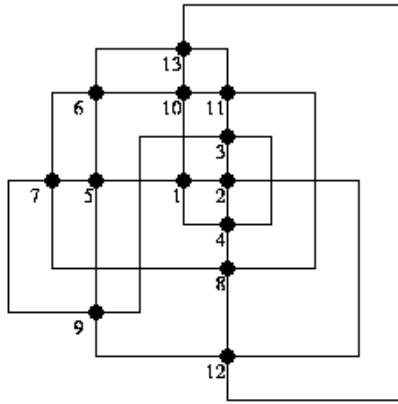
Probably need CSCI 136 and 237 (ideally)

Bill Lenhart

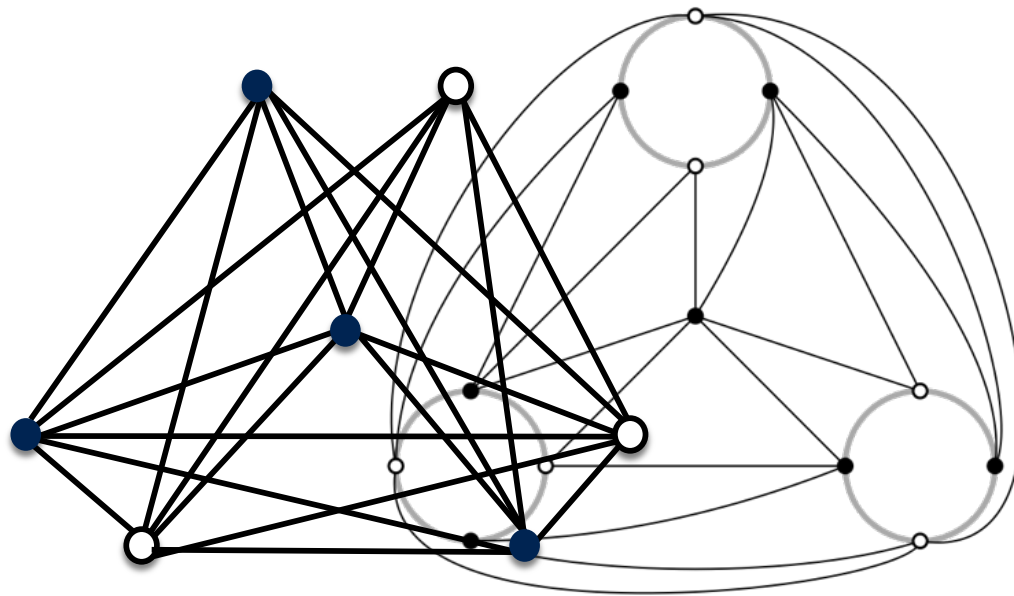


How Bill Spends His Summer Vacations

How To Draw a Graph



A (2,2)-Planar Drawing



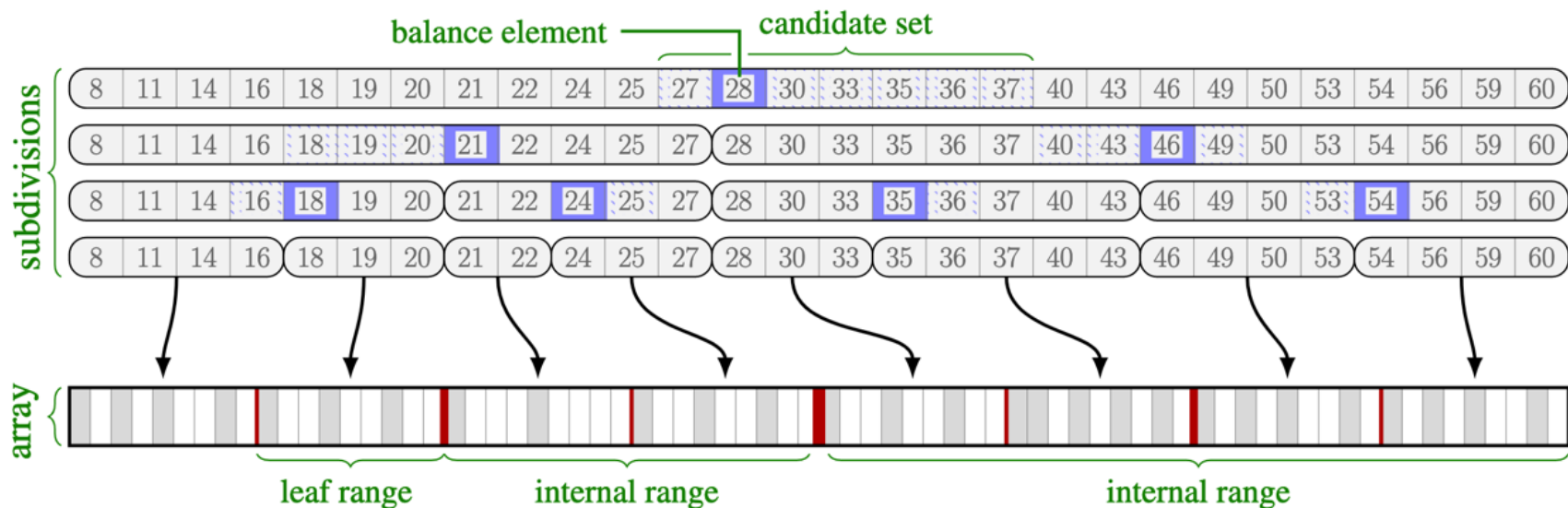
The Game

- Identify interesting/useful/important class **C** of graphs
- Specify drawing model **M**
- Ask questions
 - Can $G \in \mathbf{C}$ be drawn according to model **M**?
 - Can the drawing be computed efficiently?
 - Can the class **C_M** be easily characterized?
- Need CSCI 256 and strong background in math

Sam McCauley



- Sam works on Data Structures
- How do we store large amounts of data so that we can quickly and efficiently access it?

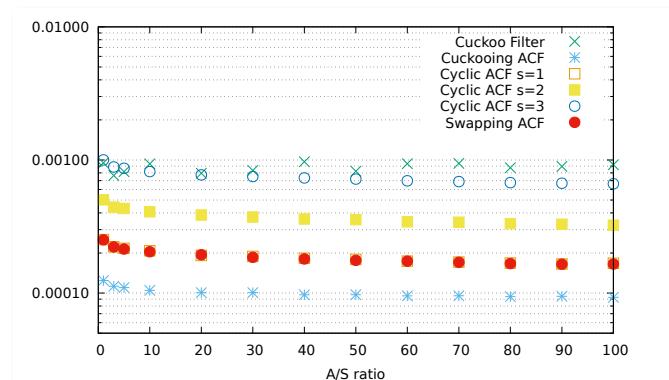


Data Structures (Sam McCauley's research)

- Main principle: use randomness (and some math) to make these data structures simpler and more efficient
- Combines coding skills with worst-case analysis

$$\mathbb{E} \left[\sum_{\ell=1}^{k_i} |\{q \in Q_0 \mid h_{\beta}(q) = h_{\beta'}(x_{i_{\ell}}) \text{ for some } \beta, \beta' \in \{0, 1\}\}| \mid q_i \text{ is costly} \right] \leq 2 + 2k_i |Q|(\epsilon/n)$$

Using probability (like this) can lead
to improved performance!



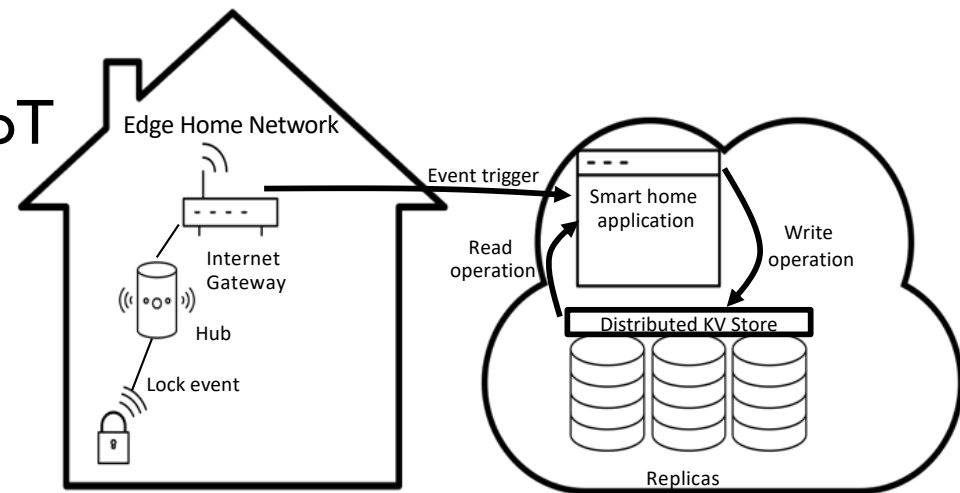
What's needed?

- Flexible—both coding and math skills are useful
- But, you'll probably want to have taken Algorithms (CSCI 256)
- Projects may entail:
 - Designing new data structures
 - Implementing data structures that work in theory (how well do they do in practice?)
 - Looking at what use cases work best

Kelly Shaw



- Kelly works on Parallel Architectures/IoT
- Interest in systems running concurrent applications
- Evaluation of existing systems
 - Analysis of application needs
 - Analysis of application/platform correctness
- System/tool creation
 - Static analysis tools examining correctness of applications
 - Amazon AWS IoT, Samsung SmartThings, Google IoT
 - Modifying existing systems to try different solutions related to concurrency
 - Mozilla WebThings – gateway for smart home devices
 - Cassandra – distributed database
- Prerequisite knowledge: CS237



Name. **Shikha Singh**

Research Area.

Algorithmic Game Theory



- Problems at the intersection of economics and computation
- Overarching goals of the field:
 - Design the rules of the game (a.k.a algorithms) so that strategic behavior by participants leads to a desirable outcome
 - Understand and analyze selfish behavior

The next time we bemoan people exploiting loopholes to subvert the intent of the rule makers, instead of asking 'What's wrong with these people?' let's instead ask, 'What's wrong with the rules?' and then adopt a scientifically principled approach to fixing them —Hartline and Kleinberg

Algorithmic Game Theory: Examples of Problems

- **Auctions.** Designing revenue maximizing auctions which incentivize bidders to bid their true value
- **Resource allocation.** Designing a fair allocation algorithm that incentivizes selfish participants to reveal true preferences
- **Selfish routing.** Analyzing global congestion in networks where each participant selfishly minimize their travel time



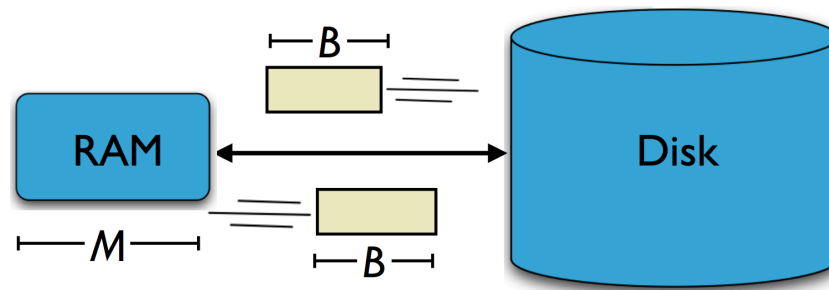
Algorithmic Game Theory: Background

- What do you need to do work on work in this area?
 - Mathematical background, such as Discrete Mathematics
 - Some familiarity with designing and analyzing algorithms
 - Programming background useful but not necessary

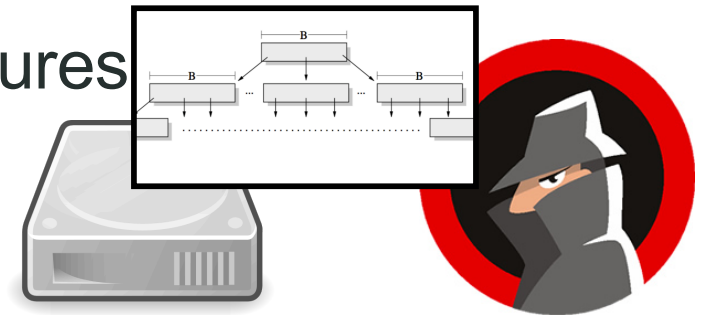


Other Research: Algorithms and Data Structures

- I/O-efficient algorithms and data structures



- Randomized and adaptive data structures



- Optimization, Approximation, online and LP duality algorithms

Aaron Williams



Retrogame Archeology / History

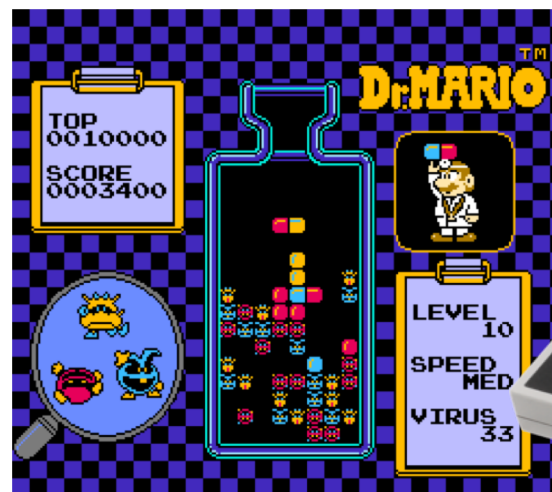
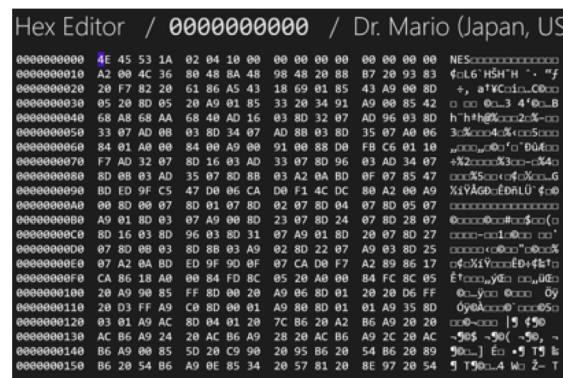
How were video games from the 1980's programmed?

We examine the machine code and figure it out!

This is a new field and there are a lot of "important" problems to be solved.

- ✦ Play study video games all summer!
- + Low-level system design and programming.
- + Learn undocumented programming tricks and optimizations from masterful coders of yore.

Requirements: Programming experience.



Dr. Mario (NES) uses a 15-bit linear feedback shift register when generating random levels.

Aaron Williams



Computational Complexity of Puzzles and Games

Which type of puzzle is *harder*: Sudoku or Rush

Hour

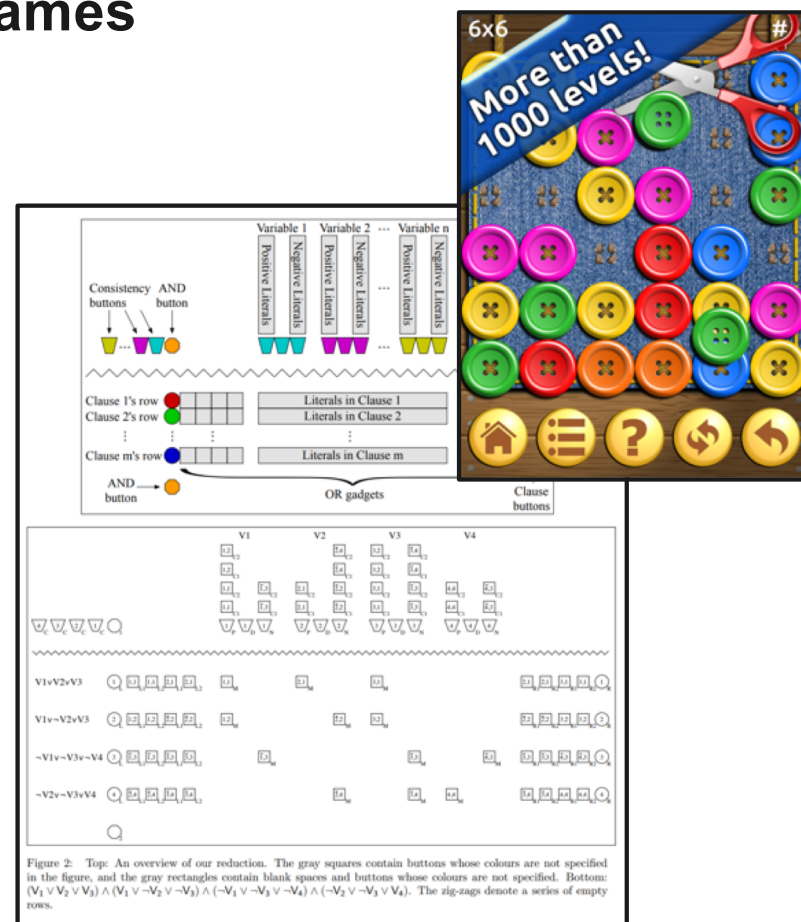
5	3		7				
6			1	9	5		
	9	8					6
8				6			3
4			8		3		1
7				2			6
	6					2	8
			4	1	9		5
			8			7	9



Apply the area of the computational complexity to well-known puzzles and games.

- + Creative thinking.
- + Experience + supervision writing a research paper.
- + Turn computational complexity into an applicable skill for use later in your career.

Requirements: CSCI 256 (required) / CSCI 361 (preferred).



The puzzle app *Buttons & Scissors* is NP-hard because it can model 3-SAT.

Aaron Williams



Efficient Generation of Combinatorial Objects

```
import itertools
for permutation in itertools.permutations([1,2,3,4,5]):
    # do something with the permutation
```

How is the above code implemented?

Why are the Python developers doing it wrong?

- + Discrete Mathematics and Algorithms.
- + Experience + supervision writing a research paper.
- + Implement practical libraries that are used in high-efficiency applications.

Check out combos.org for more details.

Requirements: CSCI 256 / MATH 200

COS ++ The Combinatorial Object Server

Home

Permutations

Generate permutations

Generate all permutations of the elements $\{1, 2, \dots, n\}$.

Size n of ground set: 4 (max. 20)

Algorithm: lexicographic order

Output format: graphics (500 objects)

Generate

Output: numbering, graphics

1: 1 2 3 4
2: 1 2 4 3
3: 1 3 2 4
4: 1 3 4 2
5: 1 4 2 3
6: 1 4 3 2
7: 2 1 3 4
8: 2 1 4 3
9: 2 3 1 4
10: 2 3 4 1
11: 2 4 1 3
12: 2 4 3 1
13: 3 1 2 4
14: 3 1 4 2
15: 3 2 1 4
16: 3 2 4 1
17: 3 4 1 2
18: 3 4 2 1
19: 4 1 2 3
20: 4 1 3 2
21: 4 2 1 3
22: 4 2 3 1
23: 4 3 1 2
24: 4 3 2 1

Total: 24

Object info

A permutation of an n -element set of objects is a linear ordering of those objects, i.e., some object is placed in the first position, another in the second position, etc. For simplicity, we think of a permutation as a bijection on the set $[n] := \{1, 2, \dots, n\}$. Such a bijection $\pi: [n] \rightarrow [n]$ can be represented in many different ways. We use one-line notation $\pi = (\pi(1), \pi(2), \dots, \pi(n))$. For example, all permutations of $[3] = \{1, 2, 3\}$ are 123, 132, 213, 231, 312, 321.

The following figure illustrates the output of all eleven algorithms running on this website for generating all permutations of $[4] = \{1, 2, 3, 4\}$ where 1=red, 2=yellow, 3=green, and 4=blue.

Lex Colex Zaks Heap SJT Ehr Der SiT SiTGC SW Corb

The first nine algorithms are part of Jörg Arndt's FXT library. The algorithm for lexicographic ordering is explained as Algorithm L in Knuth's book [Section 7.2.1.2, Knu11]. The algorithm for generating permutations by reversing prefixes was discovered by Zaks [Zak84]. The minimal-change order algorithm transposes only two elements in each step and is described in Heap's paper [Heap63]. The algorithm for generating permutations by adjacent transpositions is also known as Steinhaus-Johnson-Trotter algorithm [Ste64, Joh63, Tro62] or "plain changes". It is described as Algorithm P in Knuth's book [Knu11]. The algorithm using star transpositions always swaps the first element of the permutation with some other element. It was discovered by Gideon Ehrlich and is described as Algorithm E in Knuth's book [Knu11]. In the derangement order no two successive permutations have any element at the same position. This algorithm is described in Savage's survey [Sav97]. In a single track ordering of permutations, each column in the list of permutations is a cyclic shift of the first column. In a single track Gray code ordering, in addition to the previous property, any two consecutive permutations differ only in at most two transpositions (only a single transposition for odd n). These two algorithms are described in Arndt's documentation of the FXT library [Arn10]. The single track property of these two orderings can be seen very nicely by arranging the permutations circularly, yielding a wheel where any two circular tracks differ only by rotation (left: single track ordering, right: single track Gray code ordering).

The Combinatorial Object Server page on [permutations](https://combos.org/permutations).

Next Steps

- Come talk to us!
- Summer RA apps due Feb 17
- Thesis apps due Apr 10
- Forms can be found on dept website under research

The screenshot shows a web browser window displaying the Williams College Computer Science Research Application Forms page. The page has a purple header with the Williams logo, a search bar, and a menu icon. Below the header, the page is titled "COMPUTER SCIENCE" and "Williams » Computer Science » Computer Science Research Application Forms". The main heading is "Computer Science Research Application Forms". Under "Summer Research", it says: "We invite Williams students interested in pursuing summer research opportunities within the department to download and complete our [summer science research application](#)." It also states: "You must speak in person with the faculty who you may like to work with. Please submit your completed form to Lauren Vining, CS Administrative Assistant by February 17, 2020." Below this text is a photo of a group of students and faculty on a kayak trip. The caption reads: "SSR students and CS faculty kayak trip on the Battenkill River." The "Clare Booth Luce Summer Research" section mentions the Clare Booth Luce Scholars Program (Summer Research) and lists fields of interest: Astrophysics, Computer Science, Geosciences, Mathematics, Statistics, and Physics. On the right side, there is a sidebar menu with links: Curriculum, People, 2019-2020, Research (selected), Student Research, Research Applications, Resources, Colloquium, Class of 1960s Speakers, and News & Events. At the bottom, there is an "ANNOUNCEMENTS" section with the text: "Now accepting applications for a Tenure Track Faculty".

Williams

SEARCH MENU

COMPUTER SCIENCE
Williams » Computer Science » Computer Science Research Application Forms

Computer Science Research Application Forms

Summer Research
We invite Williams students interested in pursuing summer research opportunities within the department to download and complete our [summer science research application](#).

You must speak in person with the faculty who you may like to work with. Please submit your completed form to Lauren Vining, CS Administrative Assistant by February 17, 2020.



SSR students and CS faculty kayak trip on the Battenkill River.

Clare Booth Luce Summer Research
Clare Booth Luce Scholars Program (Summer Research)The Clare Booth Luce Scholars program was established to support women who will major in one or more of the following fields: Astrophysics, Computer Science, Geosciences, Mathematics, Statistics, and Physics.

Curriculum

People

2019-2020

Research

Student Research

Research Applications

Resources

Colloquium

Class of 1960s Speakers

News & Events

ANNOUNCEMENTS

Now accepting applications for a Tenure Track Faculty