# Bowdoin Science Experience:

## Computer Science

August 2014

## Plan

- Today:
    - What is Computer Science?
    - Game of life, part I
- Tomorrow:
    - Game of life, part II (Java programming)

## What is Computer Science?

- Computer Science studies computers (??)
    - This leaves aside the theoretical work in CS, which does not make use of real computers, but of formal models of computers
    - A lot of work in CS is done with pen and paper! Actually, the early work in CS took place before the development of the first computer
    - Computer Science is no more about computers than astronomy is about telescopes, biology is about microscopes, or chemistry is about test tubes.
- Computer Science studies programming (??)
    - Programming is a big part of CS.. ..but that is just part of it.
- Computer Science studies uses and applications of computers and software (??)
    - Learning to use software packages is no more a part of CS than driving education is part of automotive engineering.

## Computational Thinking

- CT involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science.
  - by Jeannette Wing, President's Professor of Computer Science and Department Head, Computer Science Department, Carnegie Mellon University

- What are some examples of computational thinking?
  - handout: Computational thinking  by Jeannette Wing,
  - www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf

## Examples of Computational Thinking

from http://www.cs.cmu.edu/~wing/

- CT is asking: How difficult is this problem and how best can I solve it?
- CT is thinking recursively.
- CT is reformulating a seemingly difficult problem into one which we know how to solve.
- CT is choosing an appropriate representation or modeling the relevant aspects of a problem to make it tractable.
- CT is using abstraction and decomposition in tackling a large complex task.
- CT is judging a system's design for its simplicity and elegance.

# Simple, daily examples

from http://www.cs.cmu.edu/~wing/

- Looking up a name in an alphabetically sorted list
  - Linear: start at the top, binary search: start in the middle
- Taking your kids to soccer, gymnastics, and swim practice
  - Traveling salesman (with more constraints)
- Cooking a gourmet meal
  - Parallel processing: You don’t want the meat to get cold while you’re cooking the vegetables.
- Putting things in your child’s knapsack for the day
  - Pre-fetching and caching
- Cleaning out your garage
  - Keeping only what you need vs. throwing out stuff when you run out of space.
- Storing away your child’s Lego pieces scattered on the floor
  - Using hashing (e.g., by shape, by color)
- Standing in line at a bank, supermarket, customs & immigration
  - Performance analysis of task scheduling

## Computational Thinking

- Jeannette Wing talking about computational thinking
    - www.youtube.com/watch?v=C2Pq4N-iE4I

## Computational thinking: Thinking like a computer scientist

- Why?
  - Universal set of skills
  - Computers are everywhere
  - IT fastest growing industry, largest number of jobs
- So what?
  - Can't one be a successful

    {chemist, biologist, physicist,  geneticist, geologist, environmentalist, banker, financial analyst,…

    without computer science?
  - Yes… if you are 70.
  - Probably, if you are 18.  But knowledge of computers will make you much more effective in any career you may choose.

# Thinking like a computer scientist

## What does it mean to do think computationally?

- Two types of knowledge:
  - declarative:  tells you how to test something
      - e.g. sqrt x is y such  that y^2 = x and y > 0
      - Does this actually tell you how to find sqrt 10?   NO.
      - If somebody tells you that sqrt 10 is 3, can you figure out whether it's true?  YES.
  - imperative:  tells you how to find something
    - e.g.
      - start with a random guess for y, y>0
      - if y^2  = x then stop, y is sqrt x
      - else set y = (y + x/y) /2
      - repeat

## Computation is about finding a set of steps to accomplish a task.

- a recipe, a set of instructions, an algorithm

## Algorithms

- Algorithm:  well-defined procedure that explains how to solve a problem
- Examples
    - a recipe for cooking
    - instructions to assemble a piece of furniture
    - instructions to program your phone
    - instructions on how to operate a gadget
- Example: shampooing your hair
    - step 1: wet hair
    - step 2: lather
    - step 3: rinse
    - step 4: repeat
- Would you manage to wash your hair with this algorithm?  What about a robot?

# Algorithms

- Example: programming a VCR
  - Step 1: If the clock and calendar are not correctly set, then go to page 9 of the instruction manual and follow the instructions before proceeding
  - Step 2: Place a blank tape into the VCR tape slot
  - Step 3: Repeat steps 4 through 7 for each program that you wish to record, up to a maximum of 10 shows
  - Step 4: Enter the channel number that you wish to record, and press the button labeled CHAN
  - Step 5: Enter the start time and press TIME-START
  - Step 6: Enter the end time and press END-TIME
  - Step 7: This completes the programming of one show. If you do not wish to program anything else press END-PROG
  - Step 8: Press the button labeled TIMER. Your VCR is ready to record.

# Algorithms

Types of operations:

- Basic operations
  - Wet hair
  - Rinse
  - Turn on VCR
- Conditional operations
  - If batter is too dry add water
- Repeat/looping operations
  - Repeat step 1 and 2 three times
  - Repeat steps 2,3,4,…10 until  batter becomes soft.

## How to come up with an algorithm?

Example:

Problem: Adding two n-digit numbers

```
    7597831 +
    1287525
---------------
    8885356
```

How would you write an algorithm to solve this problem?  Assume the basic operation is adding one-digit numbers.

## How to come up with an algorithm?

Often hard.  Comes down to problem solving. No recipe (How do people think??)

Puzzle:

- Before A, B, C and D ran a race they made the following predictions:
    - A predicted that B would win
    - B predicted that D would be last
    - C predicted that A would be third
    - D predicted that A's prediction would be correct.
- Only one of these predictions was true, and this was the prediction made by the winner.

  In what order did A, B, C, D finish the race?

# Examples of problems

Here are some problems that are studied in a CS 1 (Intro to CS) class:

- Searching
  - Given a list of student names, and a target name, find out if the name is in the list or not
  - E.g.: search name on Bowdoin website; search a phone number in the phone book
- Matching
  - Given two lists of symbols, find out whether one occurs in the other
  - E.g.: ACATTGTACATTG and CAT
- Movie search
  - Given a list of movie names, and a keyword, find ou all movies that contain the keyword

## What people say

- What it CS?  (RIT, 2 min)

https://www.youtube.com/watch?v=U0luqHYa0EI

- What is CS? (Columbia U, 2 min)

https://www.youtube.com/watch?v=iuodtnepPBQ

- Why computer science?  (2 min)

https://www.youtube.com/watch?v=1o0oA3fa2ws

- What's computer science? (from  Wellesley undergrad, 2 min)

https://www.youtube.com/watch?v=e1d83ksAIA0

- Is CS about programming only? (2 min)

https://www.youtube.com/watch?NR=1&v=yGStqRMShj4

- Pathways to CS (U. Washington, 10 min)

https://www.youtube.com/watch?v=RENVVTNsVHg

- What's an algorithm?  (D. Malan , 4.5 min)

https://www.youtube.com/watch?v=6hfOvs8pY1k

## Today's lab: The Game of Life

The  Game of Life was described by the English mathematician John Conway. The game is not really a game. There are no players, and no winning or losing. All you need to do to play the game is create an initial configuration, and let the system evolve.

# Today's lab: The Game of Life

Rules

- The universe in the Game of Life is a two-dimensional grid of cells, each of which is in one of two possible states: live or dead.

# Today's lab: The Game of Life

Rules

- Every cell interacts with its 8 neighbors, which are the cells that are horizontally, vertically or diagonally adjacent. At each step in time, the following transitions occur:

1. Any live cell with fewer than two live neighbors dies
    - as if caused by under-population
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies
    - as if by overcrowding
4. Any dead cell with exactly three live neighbors becomes a live cell
    - as if by reproduction

- Note: the number of neighbors is always based on the cells before the rule was applied.

## Today's lab: The Game of Life

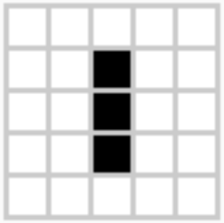
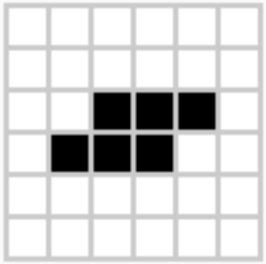
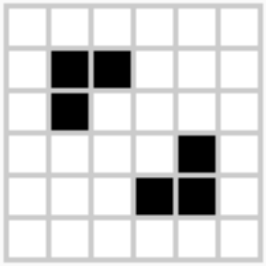
- The initial pattern constitutes the seed of the system.

- The first generation is created by applying the above rules simultaneously to every cell in the seed. Births and deaths occur simultaneously. The rules continue to be applied repeatedly to create further generations.

- A lot of interesting patterns in the Game of Life have been discovered along the years (some without the use of computers!). The simplest patterns are the ``still lives'', the ``oscilators'' (toad, blinker), the ``spaceships'' (glider), the F-pentomino, etc.

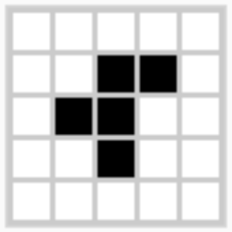
# Today's lab: The Game of Life

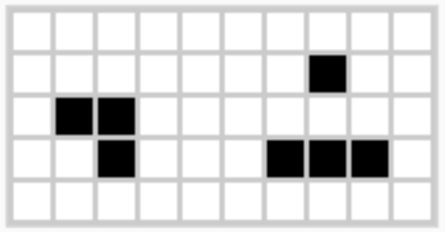
- Seed examples, with live cells shown in black and dead cells shown in white.

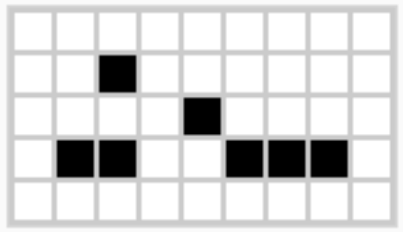
| Oscillators | |
|---|---|
| Blinker (period 2) |  |
| Toad (period 2) |  |
| Beacon (period 2) |  |

# Today's lab: The Game of Life

- Seed examples,  with live cells shown in black and dead cells shown in white.

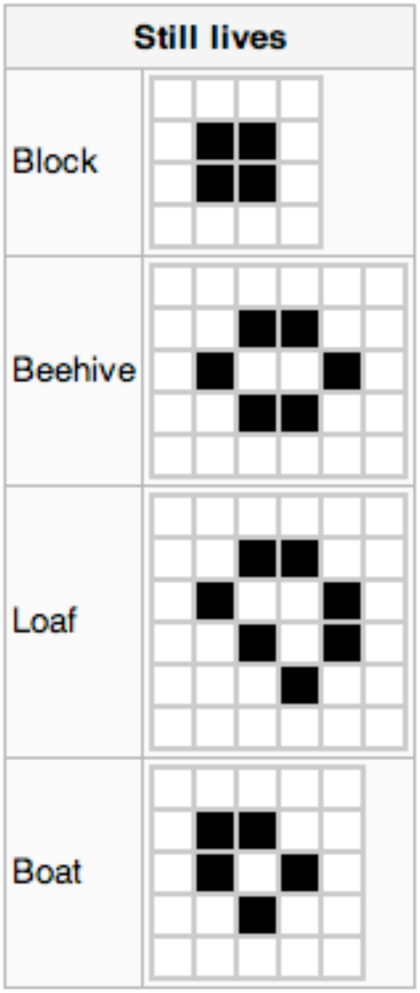

The F-pentomino


Diehard


Acorn

## Today's lab: The Game of Life

- Seed examples, with live cells shown in black and dead cells shown in white.



# The Game of Life

- What you have to do:
  - For each of the seeds in the previous slides, apply the rules and show how it evolves over a couple of generations.
  - For each generation, draw the grid representing the world.  Draw the initial grid (the seed), the grid of the  second generation (after applying the rules once), the grid of the third generation, and so on, until you see a pattern.