national center for women & INFORMATION TECHNOLOGY

COMPUTER SCIENCE-IN-A-BOX:
Unplug Your Curriculum

Computer Science-in-a-Box: Unplug Your Curriculum introduces fundamental building blocks of computer science — without using computers. Use it with students ages 9 to 14 to teach lessons about how computers work, while addressing critical mathematics and science concepts such as number systems, algorithms, and manipulating variables and logic. The National Center for Women & Information Technology (NCWIT) is pleased to offer *Computer Science-in-a-Box: Unplug Your Curriculum* in cooperation with the authors of *Computer Science Unplugged*. So unplug your computer, and get ready to explore computer science!

National Center for Women & Information Technology (NCWIT)
www.ncwit.org | info@ncwit.org | Twitter: @NCWIT | 303.735.6671

Lifetime Partner: Apple
Strategic Partners: NSF, Microsoft, Bank of America, Google, Intel, Merck, and AT&T
Investment Partners: Avaya, Pfizer, Bloomberg, Hewlett Packard Enterprise, Qualcomm, and Facebook

INTRODUCTION

Computers are everywhere, and we use them every day, but how many of us understand how they work or how they think? How can people make them operate faster and better? Computer science is a fascinating subject that explores these very questions. Every student can benefit from an introduction to the science that is possibly most central to their lives—computer science.

Computer Science-in-a-Box: Unplug Your Curriculum introduces fundamental building blocks of computer science — without using computers. This selection of activities is designed for use with students ages 9 to 14.

Use *Computer Science-in-a-Box: Unplug Your Curriculum* to teach lessons that explain how computers work, and at the same time, address critical mathematics and science concepts from number systems and algorithms to manipulating variables and logic. Presenting these activities to your students will allow them to:

- Gain fluency with decimal (base 10) and binary (base 2) numerical systems.
- Understand how 0's and 1's can be used to represent information such as digital images and numbers.
- Understand fundamental ideas of logic and apply logic to solve problems, such as sorting information into useful order quickly.
- Develop an understanding of algorithms that goes beyond basic operations of arithmetic.
- Understand that these concepts are harnessed by computer scientists to solve problems and innovate.

These learning outcomes correspond to the Mathematics Focal Points, advised by the National Council of Teachers of Mathematics (NCTM) for grades K-8:

- the use of the mathematics to solve problems
- an application of logical reasoning to justify procedures and solutions
- involvement in the design and analysis of multiple representations to learn, make connections among, and communicate about the ideas within and outside of mathematics

The *ACM (Association for Computing Machinery) K-12 Computer Science Model Curriculum*, published by The Computer Science Teachers Association, recommends using *Computer Science Unplugged* activities to address key computing concepts appropriate for Level 1, Foundations of Computer Science.

In addition to giving students experience with applied math and science in a meaningful context, *Computer Science- in-a-Box: Unplug Your Curriculum* also presents opportunities to advance communication skills, problem-solving, and creative thinking. Look for instructional objectives associated with each activity in *Computer Science-in-a-Box: Unplug Your Curriculum*.

You don't have to be a computer expert to enjoy learning these principles with your students. Answers to all problems are provided, and each activity ends with a "What's It All About?" section that can be used with students to explain the relevance of the activities. To help you get ready, the authors have created videos of each activity in action. Look for video links throughout this resource. Links to all the videos can be found at http://csunplugged.mines.edu.

ABOUT THIS RESOURCE

The National Center for Women & Information Technology (NCWIT) is pleased to offer *Computer Science-in-a-Box: Unplug Your Curriculum*. In its original version, we worked with the authors of *Computer Science Unplugged* (csunplugged.org), as well as Jane Krauss, David Burkhart, and Chris Stephenson. *Computer Science Unplugged* is available in its entirety under a Creative Commons license at www.csunplugged.org.

This revision of the NCWIT Program-in-a-Box, based on *Computer Science Unplugged* concepts, includes new activities and a more elaborate set of classroom materials to enable even novice teachers to teach these activities as part of their academic curriculum. This additional scaffolding was developed at the Colorado School of Mines through cooperation with local schools and made possible through a National Science Foundation grant (CNS# 1525652). Visit http://csunplugged.mines.edu/ for additional activities and tutorials.

NCWIT works to help teachers cultivate the next generation of innovative thinkers and increase the participation of underrepresented populations in computer science and information technology. Visit the NCWIT website to learn more: www.ncwit.org.

We welcome your feedback, input, and user experiences about this resource. Please email evaluation@ncwit.org.

CONTENTS

INTRODUCTION....2

ABOUT THIS RESOURCE....3

PART I DATA: THE RAW MATERIAL....5

1. COUNT THE DOTS: *BINARY NUMBERS* 7

2. COLOR BY NUMBERS: *IMAGE REPRESENTATION* 18

3. CARD FLIP MAGIC: *ERROR DETECTION AND PARITY* 29

PART II PUTTING COMPUTERS TO WORK....42

4. LIGHTEST AND HEAVIEST: *SORTING ALGORITHMS* 44

5. BEAT THE CLOCK: *SORTING NETWORKS* 81

6. MINIMAL SPANNING TREES: *LINKING OBJECTS IN A NETWORK*....87

7. SEARCHING: *THE BINARY SEARCH ALGORITHM* 100

8. CRYPTOGRAPHY: *ENCRYPTING AND DECRYPTING MESSAGES* 113

PART II COMPUTERS AS THINKING MACHINES 123

9. TREASURE HUNT: *FINITE-STATE AUTOMATA* 125

10. ARTIFICIAL INTELLIGENCE: *THE TURING TEST*.... 141

11. COMPUTER VISION: *EDGE DETECTION* 150

Part I
Data: The Raw Material

DATA: THE RAW MATERIAL INTRODUCTION — *WHOLE CLASS (READ ALOUD)*

How can we store information in computers?

The word "computer" comes from the Latin word *computare*, which means to calculate or add together. But, computers today are more than just giant calculators. They can be a library, help us to write, find information for us, play music, and even show movies. So, how do they store all this information? Believe it or not, the computer uses only two things: 0 and 1!

What is the difference between data and information?

Data are the raw material, the numbers that computers work with. A computer converts its data into information (words, numbers and pictures) that you and I can understand.

How can numbers, letters, words, and pictures be converted into 0's and 1's?

Let's learn how 0's and 1's operate in the binary number system, and how computers interpret these data systems to do work and represent information.

Activity 1
Count the Dots: *Binary Numbers*

Summary

Data in computers is stored and transmitted as a series of 0's and 1's. The goal of this lesson is to demonstrate how ordinary numbers can be represented with only two numbers.

Getting Ready

ACTIVITY BREAKDOWN	PAGE	ADDITIONAL FILES IN .ZIP DOWNLOAD	TIME	
Binary Numbers Demonstration	8	Large Binary Cards PDF Small Binary Cards PDF (optional)	10 min.	50 minutes TOTAL
Worksheet 1 — *Binary Numbers*	9	no additional files	5 min.	
Worksheet 2 — *What's the Number?*	12	no additional files	5 min.	
Worksheet 3 — *Check Your Understanding*	14	no additional files	10 min.	
Wrap-up Activity — *Go Fish*	16	Binary Go Fish Playing Cards PDF	15 min.	
What's It All About?	17	no additional files	5 min.	

Materials

Each teacher will need:

» one set of the Large Binary Cards

Each student will need:

» Worksheet 1 — *Binary Numbers* (especially helpful for 6th graders)
» Worksheet 3 — *Check Your Understanding*

Each pair of students swill need:

» Worksheet 2 — *What's the Number*

Each group of four students will need:

» Binary Go Fish Playing Cards (Additionally, it may be helpful to have several sets of the Small Binary Cards for struggling students to use at their desks as a visual aid.)

Lesson Preparation

Before diving into the lesson, it is recommended that you watch "CS Unplugged: Binary Go Fish," which can be found here: https://www.youtube.com/watch?v=VUguR8qoqMo&index=4&list=PL-7FSY8VA_YBo1cHdhXwrkG1EkK8nvUne (length: 1 minute and 55 seconds).

materials adapted by the Colorado School of Mines with permission from Computer Science Unplugged (http://csunplugged.org)

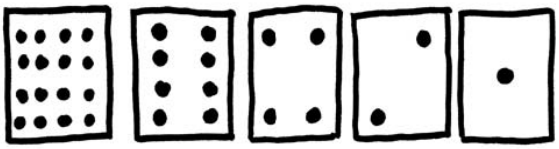
BINARY NUMBERS DEMONSTRATION GUIDE — *WHOLE CLASS*

It is helpful to demonstrate principles of binary numbers as a class activity, as in this video: https://youtu.be/b6vHZ95XDwU (length: 4 minutes and 29 seconds).

Lesson Vocabulary (You may want to write these terms on the board.)

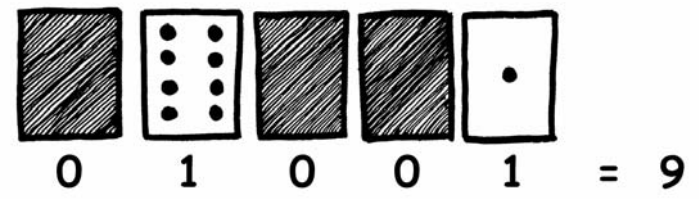
- bit
- binary

For this activity, you will need a set of five cards, as shown below, with dots on one side and nothing on the other. Choose five students to hold each of the demonstration cards at the front of the class. The cards should be held in the following order:



Start by telling the students that data is stored and transmitted as a series of 0's and 1's. Ask if any students know what each binary digit is called [**answer**: a *bit*]. Each student represents one bit. The bit may be either "on" (number visible) or "off" (number turned over).

Each card has a number of bits on it. We just need to read off the number of dots that are showing to determine what number is represented. When a binary number card is **not** showing, it is represented by a 0. When it **is** showing, it is represented by a 1. This is the binary number system.



Ask the students to make 01001. What number is this in decimal? [**answer**: 9] Try a few more until they understand the concept. Examples: What would 17 be in binary? [**answer**: 10001] 20? [**answer**: 10100] 6? [**answer**: 00110]

Now try counting from 0 onward.

The rest of the class needs to look closely at how the cards change to see if they can see a pattern in how the cards flip (each card flips half as often as the one to its right). You may like to try this with more than one group. How many dots would the next card have if we carried on to the left? [**answer**: 32] The next...? [**answer**: 64] What's the largest number that can be represented by 5 bits? [**answer**: 31]

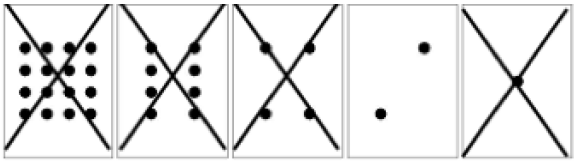
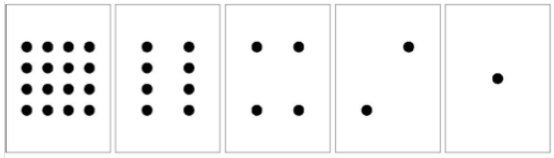
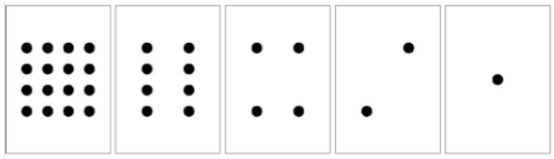
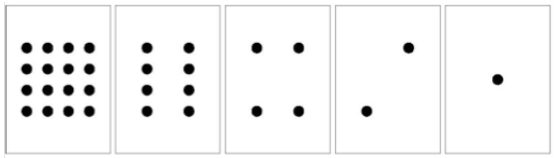
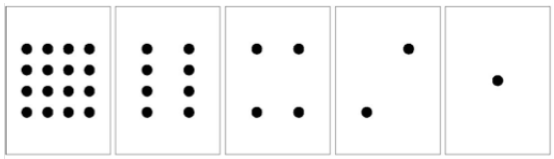
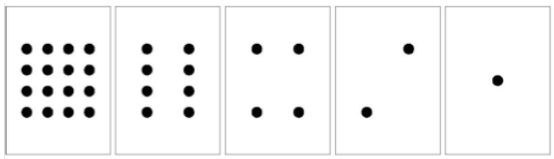
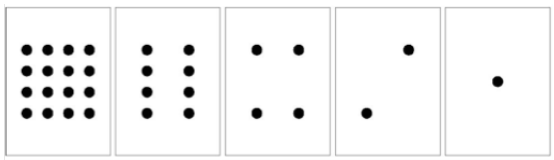
WORKSHEET 1 INTRODUCTION — *BINARY NUMBERS*

In the Binary Numbers Demonstration (page 8), students learned how to use cards representing the place value of a binary number to convert between binary and whole (decimal) numbers. This worksheet will provide students a chance to practice this conversion technique on their own.

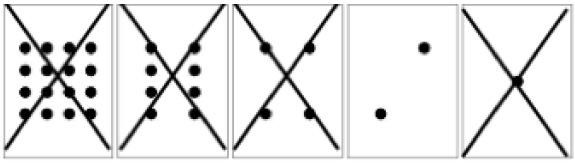
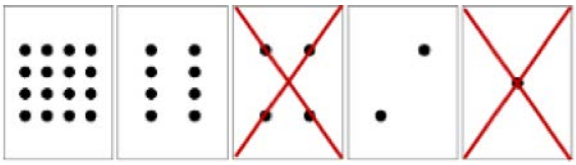
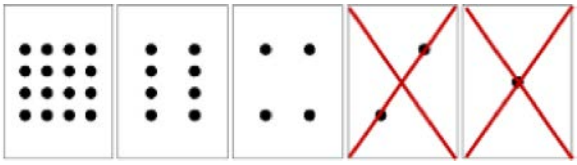
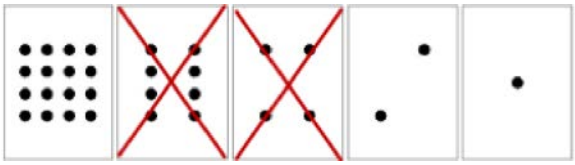
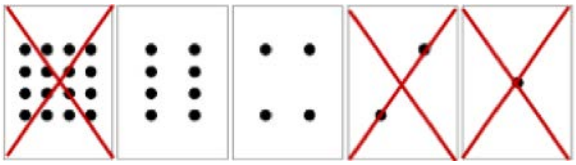
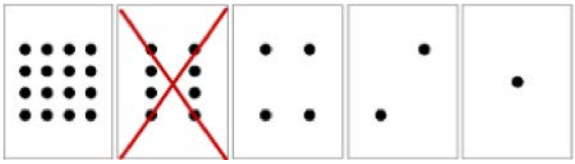
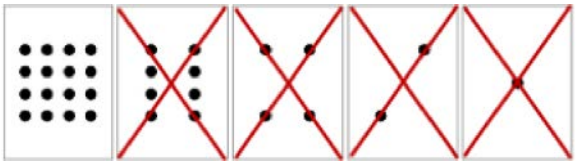
Since some students will grasp the material more quickly, you might encourage students to help each other as needed, or you might set a fixed time and then go over the solution (depends on your classroom dynamics).

WORKSHEET 1 — *BINARY NUMBERS*

Represent each normal number by crossing out the cards you do not want to use. Then, convert each set of cards into strings of 1's and 0's! The first row has been completed for you, as an example.

Number		Binary Number
2		00010
5		
3		
12		
19		
8		
15		

WORKSHEET 1 ANSWER KEY — *BINARY NUMBERS*

Number		Binary Number
2		00010
5		00101
3		00011
12		01100
19		10011
8		01000
15		01111

Quick Key

5 = 00101 **3** = -00011 **12** = 01100 **19** = 10011 **8** = 01000 **15** = 01111

WORKSHEET 2 INTRODUCTION — *WHAT'S THE NUMBER?*

As part of the Binary Numbers Demonstration (page 8), students also learned to reason about the next number in a sequence, the largest possible number, and how many cards are needed to represent various number ranges. This worksheet will reinforce those concepts, with particular focus on the relationship between the number of bits and the range of numbers that can be represented.

After giving out this bits worksheet, review the range of numbers that can be represented by 5 bits (0 to 31). Explain that in some cases the computer might store more or less than 5 bits. For example, it's common to use 8 bits together. This is known as a *byte*. For smaller numbers, we might want to use less than 5 bits. Or to save space, data might be stored as only one bit (e.g., 0 can mean false and 1 can mean true).

For this activity, students should work with a partner (or group of 3, as needed). The student should challenge their partner(s) to represent a given number using 5-bit binary (includes the leading 0's). Then, they should determine whether all 5 bits are needed, or if the number might be represented by fewer bits. Emphasize that all non-leading 0's are needed as *placeholders*, just as in the decimal system to represent a number in the hundreds we need 3 digits, to represent a number in the thousands we need 4 digits, etc.

Students who are doing well with this material can challenge each other to represent larger numbers.

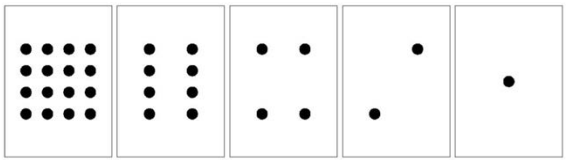
Terminology (optional)

- *binary* – a system where number representation is done through 1's and 0's
- *bit* – 1 binary number (1 or 0)
- *binary string* – a sequence of bits
- *byte* – a binary string with 8 bits
- *kilobyte* – 1,000 bytes
- *megabyte* – 1,000 kilobytes
- *gigabyte* – 1,000 megabytes

WORKSHEET 2 — *WHAT'S THE NUMBER?*

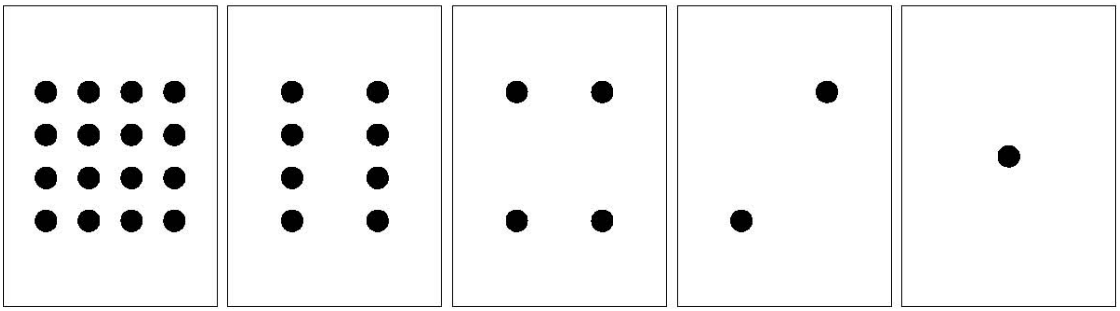
Take turns thinking of a number between 1 and 31 (or bigger numbers, if you're feeling adventurous)! Ask your partner to convert your number into binary. Then, write down the minimum number of bits needed to represent the number in binary (see example). Record your answers in the chart below.

	NUMBER	5-BIT BINARY	MINIMUM BITS
Example	3	00011	2
	22	10110	5



Scratch Work

WORKSHEET 3 — *CHECK YOUR UNDERSTANDING*



1. What is the next number in the sequence?

00001 00010 00011 00100 _____

2. What decimal number is represented by 01011?

3. How would you write the number 20 in binary?

4. What is the largest number you can represent using 5 cards (i.e., 5 bits)?

5. What is the largest number you could represent if you had only 3 cards?

6. How many cards (bits) would you need to represent the number 63?

WORKSHEET 3 ANSWER KEY — *CHECK YOUR UNDERSTANDING*

1. What is the next number in the sequence?

00001 00010 00011 00100 00101

2. What decimal number is represented by 01011?

11

3. How would you write the number 20 in binary?

10100

4. What is the largest number you can represent using 5 cards (i.e., 5 bits)?

31

5. What is the largest number you could represent if you had only 3 cards?

7

6. How many cards (bits) would you need to represent the number 63?

6

Quick Key

1. 00101 (the pattern is 1, 2, 3, 4, so the last number should be 5).
2. The number is 11.
3. 10100 is 20 in binary.
4. 31 is the largest number you can represent with 5 bits.
5. 7 is the largest number you can represent with 3 bits.
6. 6 bits would let you store values from 0 to 63.

WRAP-UP ACTIVITY — *GO FISH*

Summary

This activity will extend the concepts of how a computer works by having students practice converting between number systems in a game of Go Fish. Here, students will realize that, although the numbers are *represented* differently, they still convey the exact same meaning.

Materials

» Each group of four students will need the Binary Go Fish Playing Cards (available in .zip download).

Introduction

Before giving out the worksheet, it is beneficial to have a brief discussion to tie together what they just learned, this new activity, and real-world computer science applications.

Discussion

We just learned that data in computers is stored as a series of 0's and 1's. Our binary cards helped us to see a pattern so we could determine how many cards are needed to represent various number ranges.

What if you were creating a game with a deck of cards? A normal deck has several cards with no numbers (Ace, Jack, Queen, and King). How could we represent these face cards using only binary numbers?

If necessary, review converting between decimal and binary number systems.

This activity will extend the concepts of pattern recognition and conversion between binary and decimal by playing Go Fish.

We will be using a special deck of playing cards, in which each number is represented in binary form instead of decimal notation. Thus, the card that would normally say "8" now reads "1000." Because Go Fish is not necessarily concerned with the suit of each card (at least in our version today), no suits are printed on the playing cards.

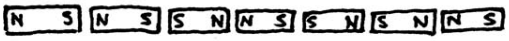
There are a few minor differences between how a normal Go Fish game would be played and how we will play today in the classroom. The rules of Go Fish (including our tweaks) are below:

1. Students should get into groups of 4 or 5. Shuffle the deck of binary playing cards.
2. Each student should be dealt 5 cards face down (each student can look at their own cards).
3. One student starts by asking another student if they have a certain card ("Sally, do you have any 5s?").
4. If the student does have any card of that rank, they must give them up to the asker. If the student does not have that rank, the asker must draw a card from the deck.
   Note: Students should *not* ask "Sally, do you have any 0101's?" as that defeats the purpose of practicing converting between the binary and decimal number systems!
5. Because we are playing binary Go Fish, you only need a pair of cards to form a set (i.e., you only need two 5s to place the set down, so two students could have a set of 5s).
6. Kids continue game play until the deck has been exhausted. If a student runs out of cards before then, they should draw another 5 cards from the deck.
7. The student with the most binary pairs wins.

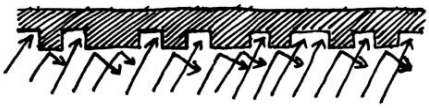
WHAT'S IT ALL ABOUT? — *WHOLE CLASS*

Computers today use the binary system to represent information. It is called binary because only two different digits are used. It is also known as base 2 (humans normally use base 10). Each 0 or 1 is called a *bit* (**bi**nary dig**it**). A bit is usually represented in a computer's main memory by a transistor that is switched on or off, or a capacitor that is charged or discharged.

When data must be transmitted over a telephone line or radio link, high- and low-pitched tones are used for the 1's and 0's. On your phones and flash drives, bits are represented by the direction of a magnetic field on a coated surface, either North-South or South-North.



CDs, DVDs, and Blu-Rays store bits optically — the part of the surface corresponding to a bit either does or does not reflect light.



One bit on its own can't represent much, so they are usually grouped together in groups of 8, which can represent numbers from 0 to 255. A group of 8 bits is called a byte.

The speed of a computer depends on the number of bits it can process at once. For example, a 64- bit computer can process 64-bit numbers in one operation, while a 32-bit computer must break 64-bit numbers down into smaller pieces, making it slower.

Ultimately, bits and bytes are all that a computer uses to store and transmit numbers, text, and all other information. In some of the later activities we will see how other kinds of information can be represented on a computer.

Computers also store letters and numbers in binary code. A computer needs 7 bits to store all of the characters. This allows for up to 128 characters. Usually the 7 bits are stored in an 8-bit byte, with 1 bit wasted. This is called ASCII (American Standard Code for Information Interchange).

Activity 2
Color by Numbers: *Image Representation*

Summary

Images are everywhere on computers. Some are obvious, like photos on web pages, but others are more subtle: A font is really a collection of images of characters, and the icons that we use to interact with the operating system on Mac and Windows machines are just images. But, how does the computer turn the 0's and 1's stored on its hard drive into colorful images on your screen? Today's lesson will answer just that.

Getting Ready

ACTIVITY BREAKDOWN	PAGE	ADDITIONAL FILES IN .ZIP DOWNLOAD	TIME	
Introduction	19	no additional files	10 min.	50 minutes TOTAL
Worksheet 1 — *Hidden Pictures*	21	no additional files	10 min.	
Worksheet 2 — *Share with A Friend*	24	no additional files	10 min.	
Wrap-up Activity — *Image Compression*	26	no additional files	10 min.	
What's It All About?	28	no additional files	10 min.	

Materials

Each student will need:

- pencil or pen
- Worksheet 1 — *Hidden Pictures*
- Worksheet 2 — *Share with A Friend*
- Wrap-up Activity — *Image Compression*

materials adapted by the Colorado School of Mines with permission from Computer Science Unplugged (http://csunplugged.org)

INTRODUCTION — *WHOLE CLASS*

Lesson Vocabulary (You may want to write these terms on the board, perhaps, as part of the discussion.)

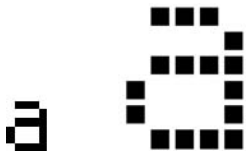
» pixel
» compression

Images are displayed on our computer screen for all sorts of reasons. Videos are just still images shown quickly after one another. You can attach and view images in your email, and you can browse cat pictures on Google's image search.

Today, we'll explore how images are represented in a computer. To keep it simple, we'll limit our discussion to black and white images. A primary focus will be answering the question, "How can we store an image using the smallest space possible on our hard drive?"

The Basics

For starters, use the included PowerPoint slides (available in the .zip download), or draw a 6x5 grid on a white board, coloring in the squares to form an "a" shape. You can tell the students this is what a letter looks like if you were to zoom in really far on the computer monitor. Ask if anyone knows what the small dots on the computer screen are called [**answer**: "picture elements" or "pixels"]. When you look at the image from far away, you can see that it is the letter "a."



We will use a fairly simple scheme to turn this image into binary numbers (remember: all of the data a computer ever uses must be stored as binary numbers). Let white pixels be represented by a 0, and black pixels be represented by a 1.

0	1	1	1	0
0	0	0	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	0	1
0	1	1	1	1

Ask students if they think this is a good idea. [**answer**: it works, but we can do better!] If we stored every pixel of every image, we would run out of room on our computers much faster than we do today. That's because our computers *compress* images to make their file size smaller, while still storing all the data to reconstruct the image. We're going to introduce a new encoding scheme that will use *fewer* numbers to represent the image.

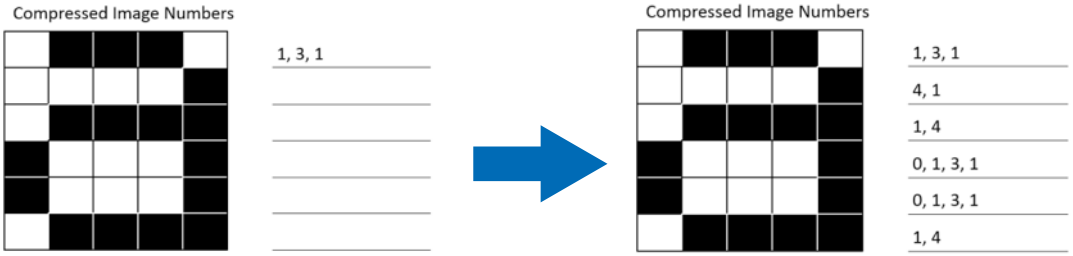
For comparison purposes, ask the students how many "bits" would be needed to represent the letter "a" using the simple method above. [**answer**: we would need 6*5 = 30 numbers]

Ask the students if they notice any patterns in the image. Point out that there are 3 black pixels together in the first row and 4 white pixels in the second. Lead the students to the idea that instead of writing down *every* bit, they can describe the pattern. Point out that there is a run of black pixels in the first row, and a run of white pixels in the second. So, we need some type of scheme that allows us to know not just how many pixels, but also what color.

Get the students to agree that we will use these rules:

1. We will clump the colored boxes together if they are adjacent in the image.
2. We will always list the number of *white* pixels in a row *first*. Then, we will alternate between black and white.

The first row of the "a" now becomes 131, because there is 1 white pixel, followed by 3 black pixels, before another white pixel at the end.



How would you encode row 4 (it starts with a black pixel)? Because we need to list the number of white pixels first, we would encode the row as 0131. Why is that? [**answer**: the computer needs a consistent method to reconstruct the image. If we left out the 0, then the colors in row 4 would be flipped because the computer would think we were starting off with 1 white pixel, then 3 black pixels, and then 1 more white pixel]. Emphasize that the sum of the compressed values should equal the total number of pixels in each row. In this example, since each row has 5 pixels, each compressed representation should add up to 5.

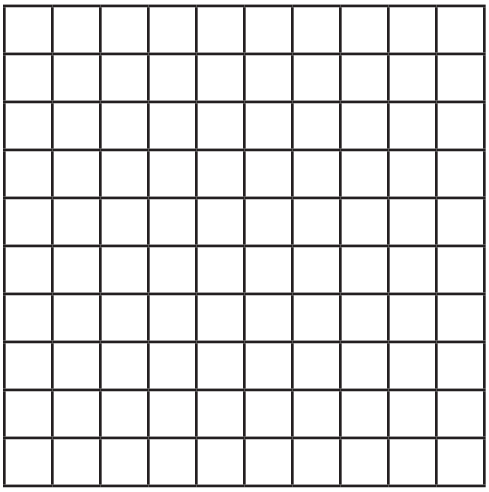
WORKSHEET 1 INTRODUCTION — *HIDDEN PICTURES*

This worksheet has students practice the exact same ideas that we just covered on the board.

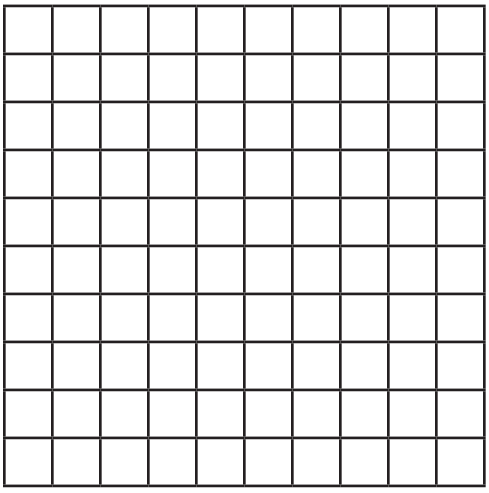
WORKSHEET 1 — *HIDDEN PICTURES*

Now that you know how numbers can represent pictures, try to draw these hidden pictures using the encoding scheme discussed.

Use "X" characters in the boxes – do not color each box!

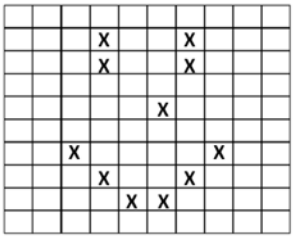


4, 2, 4
3, 4, 3
2, 6, 2
1, 8, 1
0, 10
4, 2, 4
4, 2, 4
4, 2, 4
4, 2, 4
2, 6, 2

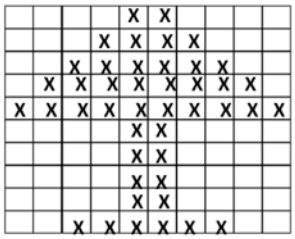


10
3, 1, 2, 1, 3
3, 1, 2, 1, 3
10
5, 1, 4
2, 1, 4, 1, 2
2, 1, 4, 1, 2
3, 1, 2, 1, 3
4, 2, 4
10

WORKSHEET 1 ANSWER KEY — *HIDDEN PICTURES*



10
3, 1, 2, 1, 3
3, 1, 2, 1, 3
10
5, 1, 4
2, 1, 4, 1, 2
2, 1, 4, 1, 2
3, 1, 2, 1, 3
4, 2, 4
10



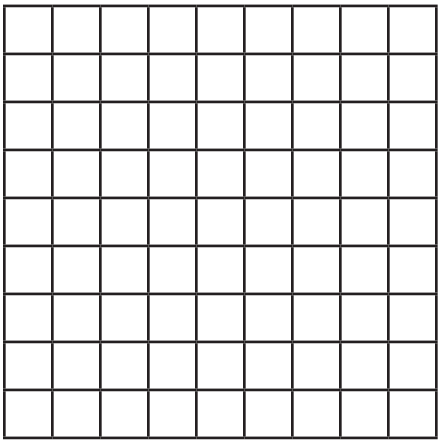
4, 2, 4
3, 4, 3
2, 6, 2
1, 8, 1
0, 10
4, 2, 4
4, 2, 4
4, 2, 4
4, 2, 4
2, 6, 2

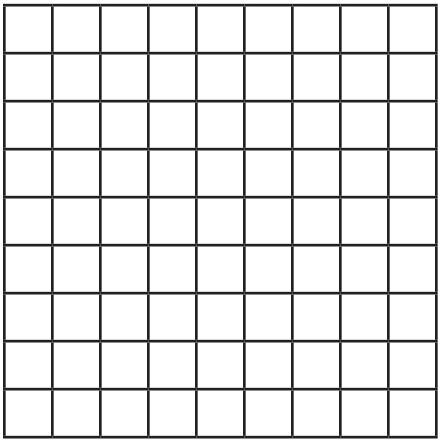
WORKSHEET 2 INTRODUCTION — *SHARE WITH A FRIEND*

Now students will have the opportunity to create their own image, encode it, and pass the encoding to a friend. Students get to practice the *encoding* process. Students know if they encoded correctly because their friend should be able to *decode* the message back into the original image using the bottom half of the worksheet.

WORKSHEET 2 — *SHARE WITH A FRIEND*

In this section, draw a picture in the 9 X 9 grid, compress the image using the method we taught you to the right of the image, and then copy the compressed image numbers on the lines on the bottom half of the picture. Rip off the bottom half of the page, trade with a neighbor, and see if they can recreate your picture just as a computer would!





WRAP-UP ACTIVITY INTRODUCTION — *IMAGE COMPRESSION*

So, why do we care about saving space? It's true that compressed images take up less space on our hard drive, but hard drives are pretty big these days. But, what about movies? Movies and film are actually made up of lots of pictures shown one after another very quickly. Let's say that our movie is made up of 120 images that make up a 'moving picture.'

If it would take 10 seconds for one image to download uncompressed, how many seconds would it take to download the entire **uncompressed** movie? Answer: 120* 10 = 1200 seconds. How many minutes is that? Answer: 1200/60 or 20 minutes.

Clearly this number is not accurate (it does not take 10 seconds/image, and there are more than 120 images in most movies) — how many students have had to wait for a video to load?

We will wrap up with one last activity.

So far, students have learned how to represent an image using two different methods: The first was to represent every pixel as a 0 or a 1, and the second was to represent groups of pixels, always starting with a group of white pixels. Let's engrain why this is so cool by demonstrating the *transfer* of an uncompressed image (the first method) and a compressed image (the second method).

Overview

Students will need to split up into groups of four (although it can be done with three). Two students will act as computers who already have a copy of the image (if doing groups of three, this will be a single student). The other two students will act as computers who are receiving a copy of the image. Note that Wrap-up Activity — *Image Compression* needs to be cut into four parts.

The students will go through two rounds of translating the image values into pictures. The key here is that during the first round, the uncompressed version is translated, and during the second round, the compressed version is translated.

Process

The student acting as the transmitting computer will verbally give the receiving computer values one by one. The receiving computer must notify the transmitting computer that they are ready for data after they have shaded in the correct number of image boxes on the worksheet. It is important to emphasize that in both rounds, students may only give ONE number at a time. For example, the uncompressed receiver may get 1 as their first number and must finish shading one box before asking for another number, while a compressed receiver may get a 3 and can immediately shade three boxes.

Display a timer that all students can see. Students should document their time immediately as they complete their image. Write down the top five times on the board. Do this for both rounds. Students should then be able to see that the compressed image times are much quicker than the uncompressed.

WRAP-UP ACTIVITY — *IMAGE COMPRESSION*

Team A

Use the compressed encoding scheme. Remember: the number of white pixels is always listed first!

-- 1,1,3,1,1

-- 0,3,1,3

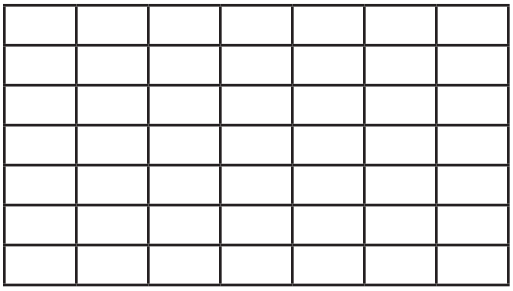
-- 0,7

-- 0,7

-- 1, 5,1

-- 2, 3, 2

-- 3, 1, 3



Team B

Use the uncompressed encoding scheme. Remember: a "0" means white, and a "1" means to color in the square (just use an "X" for speed).

-- 0, 1, 0, 0, 0, 1, 0

-- 1, 1, 1, 0, 1, 1, 1

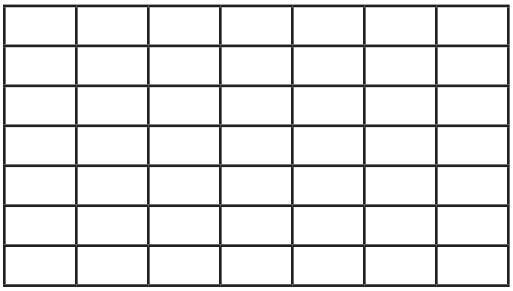
-- 1, 1, 1, 1, 1, 1, 1

-- 1, 1, 1, 1, 1, 1, 1

-- 0, 1, 1, 1, 1, 1, 0

-- 0, 0, 1, 1, 1, 0, 0

-- 0, 0, 0, 1, 0, 0, 0



WHAT'S IT ALL ABOUT? — *WHOLE CLASS*

Computers represent almost everything with some graphical interface. Anything with a screen needs to be able to interpret data and turn it into a displayable image.

Computer scientists care about compression because it allows us to do more on our computers without needing to upgrade any hardware. Imagine this: You can choose the video quality of any video on YouTube. If your internet is slow, you may want to choose a low-quality version so that it will load quicker. We don't like low-quality because it usually looks "blurry" or "pixelated."

Using techniques like we discussed today, we are able to send more information using the same number of steps (think back to the compression exercise: the student receiving the compressed image was able to fill in multiple pixels before needing to ask for more data). When computer scientists create better compression schemes, it's basically a free YouTube upgrade. We can watch better quality videos *without* having to upgrade our internet speed. Yay!

Optional Extension Discussion – Pixels per Screen

The number of pixels available is also known as the resolution of the screen. The resolution of the screen to the right is 800x1300. This means that there are 1,040,000 available on the screen.

The example to the right shows a sample of what a given image would look like on a screen. What would happen to the image of the dog if the screen resolution was doubled?

The majority of computer screens have a resolution of 1024x768 or higher.

Optional Extension Discussion – Understanding Pixels in Color

The following videos describe how the concepts in this lesson can be extended to describe how the computer deals with color images:

» https://www.youtube.com/watch?v=EXZWHumclxo&t=210s
» https://www.youtube.com/watch?v=15aqFQQVBWU&t=251s

Optional Extension Discussion – Audio compression

» http://www.bbc.co.uk/education/guides/zpfdwmn/revision/3

Activity 3
Card Flip Magic: *Error Detection and Parity*

Summary

When data is stored on a disk or transmitted from one computer to another, we usually assume that it doesn't get changed in the process. But sometimes things go wrong and the data is changed accidentally. This activity uses a magic trick to show how to detect when data has been corrupted, and how to correct it.

Getting Ready

ACTIVITY BREAKDOWN	PAGE	ADDITIONAL FILES IN .ZIP DOWNLOAD	TIME	
Magic Trick Demonstration	30	no additional files	10 min.	50-55 minutes TOTAL
ASCII Introduction	31	no additional files	5 min.	
Worksheet 1 — *Exploring ASCII*	32	no additional files	5-10 min.	
Worksheet 2 — *Using Parity for Error Checking*	35	no additional files	5 min.	
Wrap-up Activity — *Error Passing Game*	38	no additional files	15 min.	
Demonstration Explanation and What's It All About?	40	no additional files	10 min.	

Materials

Each teacher will need:

» 2 different colored sets of 36 sticky notes

Each student will need:

» Worksheet 1 — *Exploring ASCII*
» Worksheet 2 — *Using Parity for Error Checking*

Each group of 3 students will need:

» Wrap-up Activity — *Error Passing Game*

Every 2 students will need:

» 36 pennies

Lesson Preparation

Before the lesson, you might want to watch the magic trick: http://www.youtube.com/watch?v=voqghyZbZxo (length: 7 minutes and 48 seconds) and also the solution: http://www.youtube.com/watch?v=gBPZOpT4DPU (length: 4 minutes and 45 seconds).

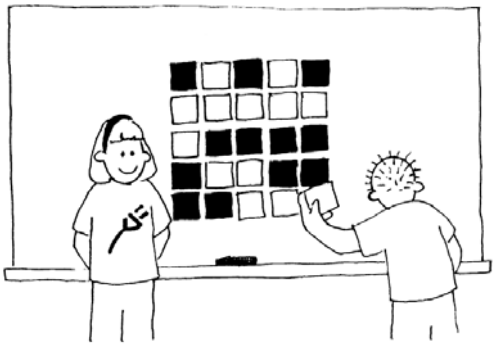
materials adapted by the Colorado School of Mines with permission from Computer Science Unplugged (http://csunplugged.org)

MAGIC TRICK DEMONSTRATION — *WHOLE CLASS*

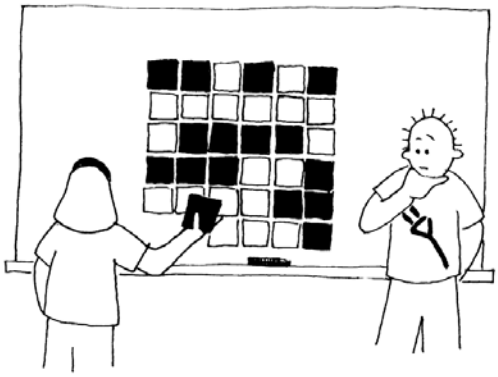
Here's your chance to be a magician!

You will need a pile of identical, two-sided cards. (To make your own, cut up a large, sturdy sheet that is colored on one side only.) For the demonstration, it is easiest to use flat magnetic cards that have a different color on each side — fridge magnets are ideal.

Choose a student to lay out the cards in a 5x5 square (or set it up yourself), with a random mixture of sides showing.



Casually add another row and column, "just to make it a bit harder."



These cards are the key to the trick. You must choose the extra cards to ensure that there is an even number of colored cards in each row and column.

Have another student volunteer flip over one card only while you cover your eyes (or leave the room if you can). The row and column containing the changed card will now have an odd number of colored cards, and this will identify the changed card.

Do not explain the magic trick! Tell the students it will be revealed soon.

ASCII INTRODUCTION — *WHOLE CLASS*

Ask the students whether computers are involved in sending information around the world. What sorts of files are sent? Have the students think about text-based information, such as email, homework assignments, letters, etc.

Ask the students whether computers are error-proof. Because computers are very fast at some tasks, and are good at things like arithmetic, it might seem like there are never errors. But, when transferring information around, errors happen all the time. Computer scientists try to think up ways to detect errors (and sometimes even correct them).

Write the word TKGER on the board. Ask the students what word they think that should be (hopefully they all guess TIGER). Tell them that before we talk more about how computers detect errors, we want to talk about how computers represent letters and numbers.

ASCII

If you did Count the Dots: Binary Numbers activity (page 7) or Color by the Numbers: Image Representation activity (page 18), students should know that the computers store all data as numbers. Specifically, all of the text files, pictures, videos, and web pages are represented on your device using the binary numbers 0 and 1. These are referred to as bits (binary digits). Many years ago, computer scientists agreed to a standard way to represent letters and numbers. This encoding scheme is called ASCII (pronounced "as key"), which stands for American Standard Code for Information Interchange. It's a system used to represent English characters. It uses 7 bits. Does anyone remember how many different binary numbers can be represented with 7 bits? [**answer**: 128]

So, ASCII includes 128 different characters, which includes numbers, lowercase letters, uppercase letters, and some special characters such as punctuation marks.

WORKSHEET 1 INTRODUCTION — *EXPLORING ASCII*

ASCII characters are stored using 8 bits (each ASCII character uses an 8-digit binary number). The left-most 7 bits in a binary number in ASCII are used to represent the character. For example, the letter "A" is number 65 in decimal, or 1000001 in binary. The right-most digit is used to represent a ***parity*** bit, which helps a computer know whether an error has occurred. The parity bit is used to make any number have an even count of 1's.

Examples (show both, to avoid the misconception that a 0 parity bit means correct, and a 1 parity bit means an error):

» For our "A" example, the binary representation has two 1's, so the eighth, right-most bit will be a 0. The ASCII value for "A" would be 1000 0010.
» For a "C" the binary value is 67, which is 1000011. This binary number has an odd number of 1 bits. So, to ensure the 8-bit code has an even number of 1's, the parity bit must be a 1 (i.e., 1000 0111).

Have students complete this worksheet, which has students decode the word "BEGIN" with 7-bit numbers (no parity bit). After adding parity, they will find that the "I" in the word BEGIN is an error (pretend the computer meant to store the word BEGAN).

WORKSHEET 1 — *EXPLORING ASCII*

ASCII stands for the American Standard Code for Information Interchange. It's a system used to represent English characters, and it was designed to encode 128 different characters. The table below maps the uppercase alphabet to 7-digit values.

A 1000 001	B 1000 010	C 1000 011	D 1000 100	E 1000 101	F 1000 110
G 1000 111	H 1001 000	I 1001 001	J 1001 010	K 1001 011	L 1001 100
M 1001 101	N 1001 110	O 1001 111	P 1010 000	Q 1010 001	R 1010 010
S 1010 011	T 1010 100	U 1010 101	V 1010 110	W 1010 111	X 1011 000
Y 1011 001	Z 1011 010				

Part I. First try translating this message from binary numbers to English letters:

1000 010 1000 101 1000 111 1001 001 1001 110

_____ _____ _____ _____ _____

Part II. Encode your own message using the binary representation of each alphabetic character that you use. Write each binary number on the lines below. Your message may be anywhere from 8 – 15 letters long. Then, trade with a friend, and see if you can decode each other's messages.

_____ _____ _____ _____ _____

_____ _____ _____ _____ _____

_____ _____ _____ _____ _____

Part III. Two friends named Alice and Bob are exchanging messages using ASCII code. It looks like Alice may have made a mistake when she converted her letters into binary. Can you find Alice's mistake? Once you have found where she went wrong, describe Alice's mistake on the lines below, and correct her binary message.

1001 000 1000 100 1001 100 1001 100 1001 111

1000 010 1001 111 1000 010

_____

_____

WORKSHEET 1 ANSWER KEY— *EXPLORING ASCII*

Part I. Students should decode the word "BEGIN" with 7-bit numbers (no parity bit).

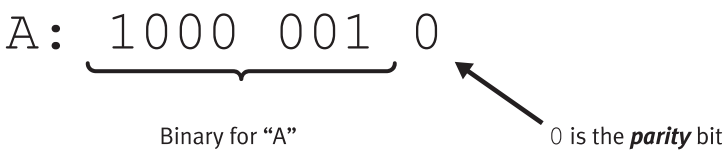
Part II. Students will create their own message using ASCII and swap with a partner. No specific answer.

Part III. The encoded message is "HDLLOBOB" but should be "HELLOBOB." The code for the second character should be 1000 101.

WORKSHEET 2 GUIDE — *USING PARITY FOR ERROR CHECKING*

Ask students how a computer might identify the error in Worksheet 1 — *Exploring ASCII* (page 33). Some students may have the idea of comparing to a dictionary. Although that's an interesting idea, the concept for this lesson is parity.

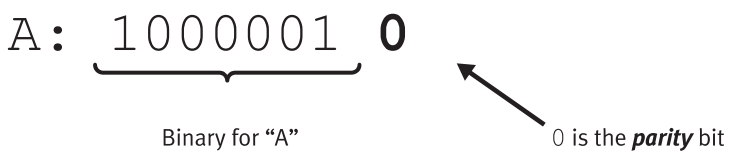
As shown in the worksheet chart, each character requires only seven binary bits (0's and 1's). To help identify errors, computer scientists came up with the idea of adding an eighth bit, known as the ***parity*** bit, to each character. The figure below shows the ASCII code for the letter A, with the parity bit added.



To practice, have students complete this worksheet. The figure above is included at the top as a reminder.

WORKSHEET 2 — *USING PARITY FOR ERROR CHECKING*

When saving data to your computer or sending data over the internet, errors can happen. The character "A" only takes 7 binary bits (0's and 1's) to represent, and the eighth bit is used as a ***parity*** bit to try and detect if an error happened while saving the letter to your computer.



Below is part of the ASCII table (the part that shows capital letters) with parity bit shown in **bold**:

A 1000 001**0**	B 1000 010**0**	C 1000 011**1**	D 1000 100**0**	E 1000 101**1**	F 1000 110**1**
G 1000 111**0**	H 1001 000**0**	I 1001 001**1**	J 1001 010**1**	K 1001 011**0**	L 1001 100**1**
M 1001 101**0**	N 1001 110**0**	O 1001 111**1**	P 1010 000**0**	Q 1010 001**1**	R 1010 010**1**
S 1010 011**0**	T 1010 100**1**	U 1010 101**0**	V 1010 110**0**		

Part I.

a) How many bits (0's and 1's) are used to represent the letter C, without a parity bit? ____

b) What parity bit is used for the letter C? (Circle a bit parity.) **1** or **0**

c) Why is 1 used as the parity bit for J, rather than 0?

Part II. Complete the table by filling in the parity bit for the letters W, X, Y, and Z. **Remember that a parity bit is 0 if there are an *even* number of 1's in the binary number, or it is 1 if there are an *odd* number of 1's in the binary number.**

W 1010 111		X 1011 000		Y 1011 001		Z 1011 010	

Part III. Below is the same message as in Worksheet 1 — *Exploring ASCII*, but this time it was sent with parity bits. Is there an error in the message? **First, underline the parity bits, then circle a binary number if you think it was sent incorrectly.**

1000 0100 1000 1011 1000 1110 1001 0010 1001 1100

WORKSHEET 2 ANSWER KEY — *USING PARITY FOR ERROR CHECKING*

Part I.

a) How many bits (0's and 1's) are used to represent the letter C, without a parity bit? **7**

b) What parity bit is used for the letter C (circle one)? (1) or **0**

c) Why is 1 used as the parity bit for J, rather than 0?

The code for J is 1001 010, which includes three 1-bits (an odd number). So we must add a 1, so that the 8-bit code for J has an even number of 1-bits.

Part II.

W 1010 111	1	X 1011 000	1	Y 1011 001	0	Z 1011 010	0

Part III.

The highlighted character is wrong, because there are an odd number of 1 bits. Note that parity can detect that the character is wrong, but by itself it cannot correct the error. Error correction is a more advanced topic that requires other techniques. One possible option is to ask the sending computer to resend (with the idea that the same error would not occur twice).

1000 0100 1000 1011 1000 1110 1001 0010 1001 1100

WRAP-UP ACTIVITY GUIDE — *ERROR PASSING GAME*

This short activity reinforces the idea that the parity bit can help computers determine whether errors occurred. Students should work in teams of 3. (If needed, a slightly larger group can be used, with more than one student as a message passer.)

This activity is similar to the classic game of telephone.

- The first student (a computer) creates a 1-character "message" along with a correct parity bit.
- The message passer may either pass along the correct code or make exactly one error. The message passer is acting like the internet, sending characters from one computer to another.
- The third student is also acting as a computer, and trying to determine whether the character was passed along correctly.

WRAP-UP ACTIVITY — *ERROR PASSING GAME*

Computer

1. Come up with a letter in your head. Using Worksheet 1 — *Exploring ASCII*, translate the letter to binary, and write the binary representation down on the line below.

   _____

2. Decide whether a 0 or 1 should be used as the parity bit. Re-write the parity bit in the blank below. **Remember that a parity bit is 0 if there are an even number of 1's in the binary number, or it is 1 if there are an odd number of 1's in the binary number.**

   _____ _____
   ↑ ↑
   Binary value for letter — Parity bit

3. Pass your sheet of paper to the person in the middle.

Message Passer

1. Double-check that the computer has selected the correct parity bit. **Remember that a parity bit is 0 if there are an *even* number of 1's in the binary number, or it is 1 if there are an *odd* number of 1's in the binary number.** If the computer has selected an incorrect parity bit, help them fix their error.
2. You get to choose whether you would like to make an error or not! If you would like to make an error, recopy the computer's binary letter value below, but change ONE of the bits. If you would not like to make an error, then copy the binary value for the letter as is.
   - You CANNOT change more than one bit.
   - You CANNOT change the parity bit.

   _____ _____
   ↑ ↑
   Binary value for letter — Parity bit

3. Pass your sheet of paper (but NOT the original paper) to the other computer to see if they can determine whether you made an error or not.

MAGIC TRICK ACTIVITY DEMONSTRATION EXPLANATION — *WHOLE CLASS*

When students have completed the worksheets, discuss the limits of error correction. Note that some of these questions may have come up during the prior worksheets.

What happens if there is more than one error?

Answer: If two errors occur, an even number of "1's" will be present, so we will not be able to detect the error!

After detecting the error, can the computer correct it?

Answer: No. The computer has no way of knowing which bit got flipped, so there is no way to get a valid character back.

How can we get a corrected character?

Answer: There are many different techniques. One that students may come up with is to store the information in several places. This idea is called redundancy, and it is used in many real-world applications. (Google stores our YouTube videos in several locations on Earth, so if one data center goes offline, we can still watch cat videos.)

Teach the trick to the students:

1. At the board, go through the magic trick once more.
2. See if any students can guess how you did it.
    1. Start with the 5 x 5 grid.
    2. Add a sixth card to each row and column; making sure the number of colored cards is always even (remember 0 is an even number).
    3. Have a student flip a card.
    4. Look for a row with an odd number of colored cards. Circle it.
    5. Look for a column with an odd number of colored cards. Circle it.
    6. The flipped card should be at the intersection.
    7. Challenge the students to try the trick. They can use colored cards or even pennies, where some are heads and some are tails. Each pair of students will need 36 pennies.
8. Now take turns to perform the trick.
9. (optional) Encourage the students to try the trick at home with their families.

WHAT'S IT ALL ABOUT? — *WHOLE CLASS*

Imagine you are depositing $10 cash into your bank account. The teller types in the amount of the deposit, and it is sent to a central computer. But, suppose some interference occurs on the line while the amount is being sent, and the code for $10 is changed to $1,000. No problem if you are the customer, but clearly a problem for the bank!

It is important to detect errors in transmitted data. So, a receiving computer needs to check that the data coming to it has not been corrupted by some sort of electrical interference on the line.

Sometimes the original data can be sent again when an error has been transmitted, but there are some situations when this is not feasible, such as if a disk or tape has been corrupted by exposure to magnetic or electrical radiation, by heat, or by physical damage. If data is received from a deep space probe, it would be very tedious to wait for retransmission if an error had occurred! (It takes just over half an hour to get a radio signal from Jupiter when it is at its closest to Earth!)

We need to be able to recognize when the data has been corrupted *(error detection)* and to be able to reconstruct the original data *(error correction)*.

The same technique as was used in the "card flip" game is used on computers. By putting the bits into imaginary rows and columns, and adding parity bits to each row and column, we can not only detect whether an error has occurred, but also *where* it has occurred. The offending bit is changed back, and so we have performed error correction.

Of course, computers often use more complex error control systems that are able to detect and correct multiple errors. The hard disk in a computer has a large amount of its space allocated to correcting errors, so that it will work reliably even if parts of the disk fail. The systems used for this are closely related to the parity scheme. And, computer scientists solve these puzzles, figuring out how to make our data reliable.

Part II
Putting Computers to Work

PUTTING COMPUTERS TO WORK *(READ ALOUD)*

Computers operate by following a list of instructions set for them. These instructions enable them to sort, find, and send information. To do these actions as quickly as possible, you need good methods for finding things in large collections of data, and for sending information through networks.

An "algorithm" is a set of instructions for completing a task. The idea of an algorithm is central to computer science. Algorithms are how we get computers to solve problems. Some algorithms are faster than others, and many of the algorithms that have been discovered have made it possible to solve problems that previously took an infeasible length of time — for example, finding millions of digits in pi, or all the pages that contain your name on the internet, or finding out the best way to pack parcels into a container, or finding out whether or not very large (100-digit) numbers are prime.

The word "algorithm" is derived from the name of Mohammed ibn Musa Al-Khowarizmi — Mohammed, son of Moses, from Khowarizm — who joined an academic center known as the House of Wisdom in Baghdad around 800 AD. His works transmitted the Hindu art of reckoning to the Arabs, and then to Europe.

Activity 4
Lightest and Heaviest: *Sorting Algorithms*

Summary

Many tasks are more easily completed if the data is sorted. Records in schools, libraries, dentists, and hospitals are often found in alphabetical order. Even your contacts on your cell phone are sorted. There are many different ways to sort data.

Getting Ready

ACTIVITY BREAKDOWN	PAGE	ADDITIONAL FILES IN .ZIP DOWNLOAD	TIME	
Sorting Cards Demonstration	46	no additional files	30 min.	50 minutes TOTAL
Worksheet 1 — *Sorting Colors*	79	no additional files	10 min.	
What's It All About?	80	no additional files	10 min.	

Materials

- Pirate Treasure Objects (in Sorting Cards Demonstration)
- Pet Rock Cards (in Sorting Cards Demonstration)
- Worksheet 1 — *Sorting Colors*
- Secret Weight Slips (in Worksheet 1 — *Sorting Colors*)
- Color Dot slips (in Worksheet 1 — *Sorting Colors*)

Lesson Preparation

Before diving into the lesson, it is recommended that you watch "Selection Sort" (which can be found here: https://youtu.be/f8hXR_Hvybo [length: 9 minutes and 6 seconds]) and "Insertion Sort" (which can be found here: http://www.youtube.com/watch?v=DFG-XuyPYUQ [length: 9 minutes and 4 seconds). The videos will give a brief introduction to two sorting algorithms: Insertion Sort and Selection Sort.

materials adapted by the Colorado School of Mines with permission from Computer Science Unplugged (http://csunplugged.org)

SORTING INTRODUCTION — *WHOLE CLASS*

Lesson Vocabulary (You may want to write these terms on the board.)

- » sorting
- » algorithm

Start off by discussing sorted data. How do you sort things? Can the students come up with an algorithm for sorting a list of numbers from least to greatest? A white board comes in handy to write down their ideas.

Let the students know that computers can only compare two numbers at a time. The algorithms that have been demonstrated show us how only comparing two things at a time can eventually lead to a sorted list!

Demonstration Prep

To see a video of this demonstration, take a look at: https://www.youtube.com/watch?v=cVMKXKoGu_Y (length: 2 minutes and 10 seconds).

You will need a balance scale (this could be created using a hanger, string, and a couple of cups) and seven weights. Label each weight with a color in the following order from lightest to heaviest:

Orange, Blue, Yellow, Red, Purple, Green, and Black.

For an engaging demo, line these up in the following order to be sorted:

Red, Blue, Orange, Yellow, Green, Purple, and then Black.

As shown in the video, you will start by finding the heaviest item (by comparing each item to the heaviest so far). Have a student volunteer help you select which items to weigh. Put the heaviest item at one end. Then, find the next heaviest. At some point, if the class is getting restless, you may want to stop and ask if they are convinced that the approach will ultimately put the items in order, and point out that since computers can only compare two items at a time, it's important to come up with a fast ***algorithm*** — *a step-by-step process to complete a task*.

SORTING CARDS DEMONSTRATION GUIDE — *WHOLE CLASS*

Prep: Print the Pirate Treasure Objects (pages 48-62) and the Pet Rock Cards (pages 63-74). Be sure to print the Sorting Cards Demonstration — *Answer Key* (page 75) also. You will hold on to it.

The purpose of this activity is to give students interactive practice with selection sort and compare that to a new type of sorting: insertion sort. See the videos listed in the *Lesson Preparation* section (page 44).

Demonstration — *Pirate Object Selection Sort*

Select student volunteers to line up in front of the class. Assign every pirate object card to a student (there are 16 cards, but you may want to use only 12, to limit the time spent on this demo and to be consistent with the Pet Rock Insertion Sort). Have these students line up in a non-sorted order. Explain that this time they will use the method previously taught (selection sort) to order themselves. To engage the class, have them guess which is the heaviest and which is the lightest. Tell them just to remember, and at the end we'll see if anyone guessed correctly.

Remind students that the goal is to first find the heaviest item and put it in place, then the next heaviest, and so forth. Choose an additional student to stand at the board, and add a tick mark every time a comparison is made. This will be used to show that the next algorithm (insertion sort) is more efficient.

Start with the first two students on the left stepping forward. Based on the answer key, tell them which is heavier. The heavier item stays in front of the list, the other student steps back, and the next student comes forward. Tell them which is heavier. Continue until all items have been compared. When the final two have been compared, have the student with the heaviest item swap places with the student at the right end of the line.

Point out that we now know the heaviest. You may want to have that student indicate s/he is in order, maybe by holding the card up in the air, or down low, or turning it over. Repeat the process. By now, the students should have the idea, so challenge them to step forward and back quickly. If possible, continue to the end, so that we have an accurate number of comparisons. This value can be calculated (as a math sidebar) as:

$12+11+10+9+8+7+6+5+4+3+2+1 = 66$

A formula for this (if students are mathematically inclined) is $n(n-1)/2 = (12*11)/2$

For fun: Did anyone guess the heaviest? The lightest?

Does anyone think the list can be sorted with less work?

Demonstration — *Pet Rock Insertion Sort*

Students will line up exactly like they did in the last activity, except this time they will be given different pet rocks rather than pirate objects. The teacher should hold the Sorting Cards Demonstration — *Answer Key* (page 75), which gives the weight of each object.

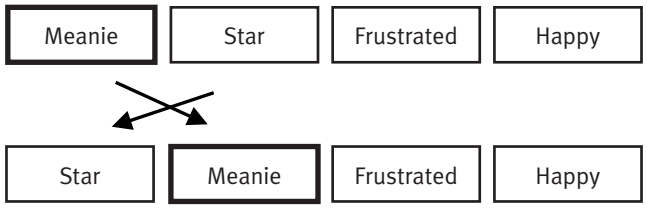
Have students line up, and hand them the cards. To ensure the first few steps are engaging and to get across how the algorithm works, give the first four students on the left the rocks:

Meanie, Star, Frustrated, and Happy

Explain that this time, they will use an insertion sort to order themselves. The idea is that we start with a sorted list of size one, and continue to add one item to the list at a time, always ensuring the list is still sorted. Again, have a volunteer student count the number of comparisons.

The student to the farthest left starts as the only element in the sorted portion of the list. Have that student step forward. Get the class to agree that if there's only one item, the list must already be sorted.

Now have the second student come forward. Tell them which is heavier (if you followed the order above, Meanie is heavier than Star). Because Star is lighter, have those two swap. Now you have a sorted list of size two. The example below shows this first step.



Now, have the third student come forward. Frustrated is heavier than Meanie, so no swap is necessary (and we have a sorted list of three).

Now, the fourth student comes forward. Happy is lighter than Frustrated, so they swap. Happy is also lighter than Meanie, so they swap. Happy is also lighter than Star, so they swap.

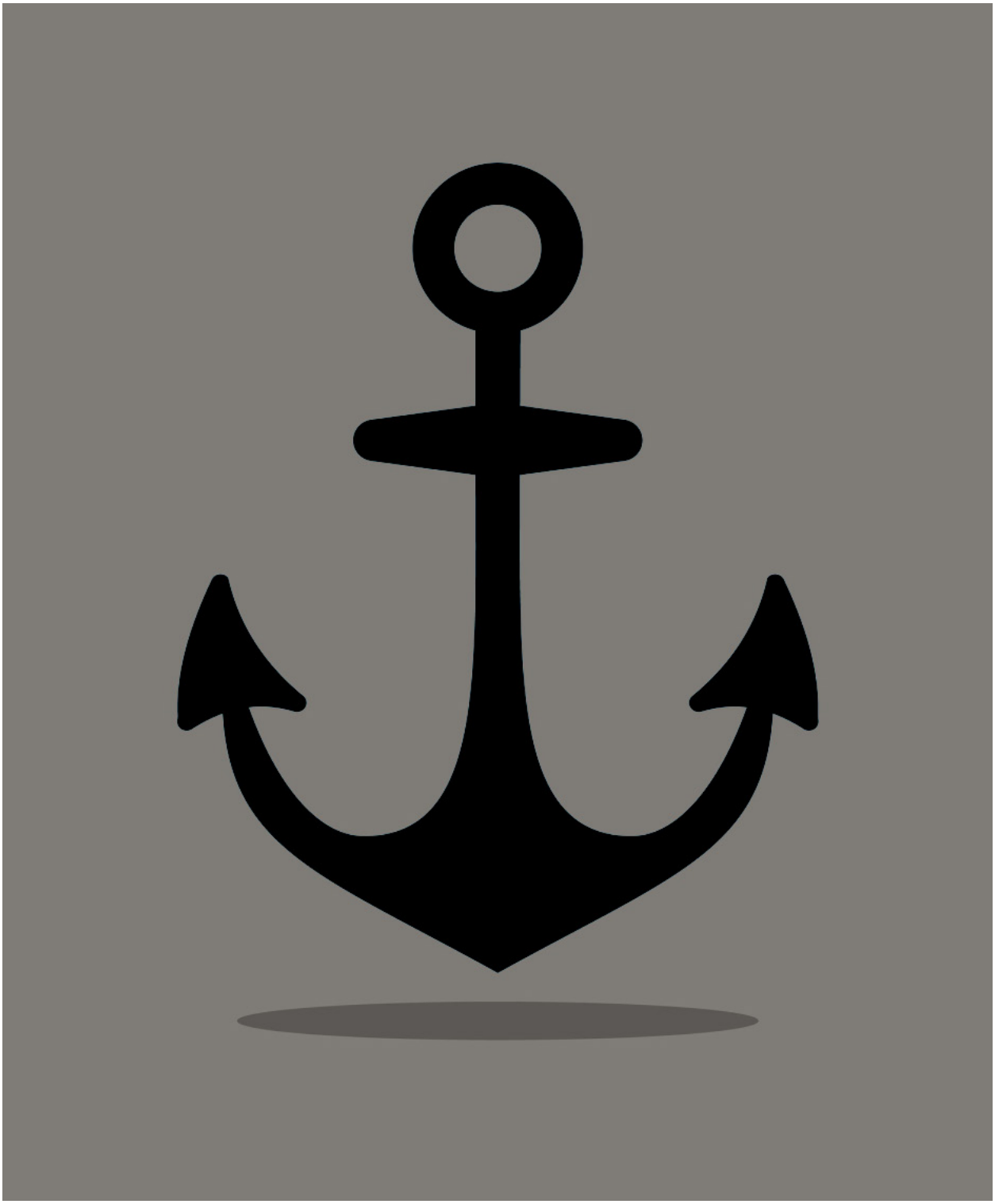
Continue this process until the entire list is in order.

Unlike selection sort, we don't have an exact formula for the number of comparisons.

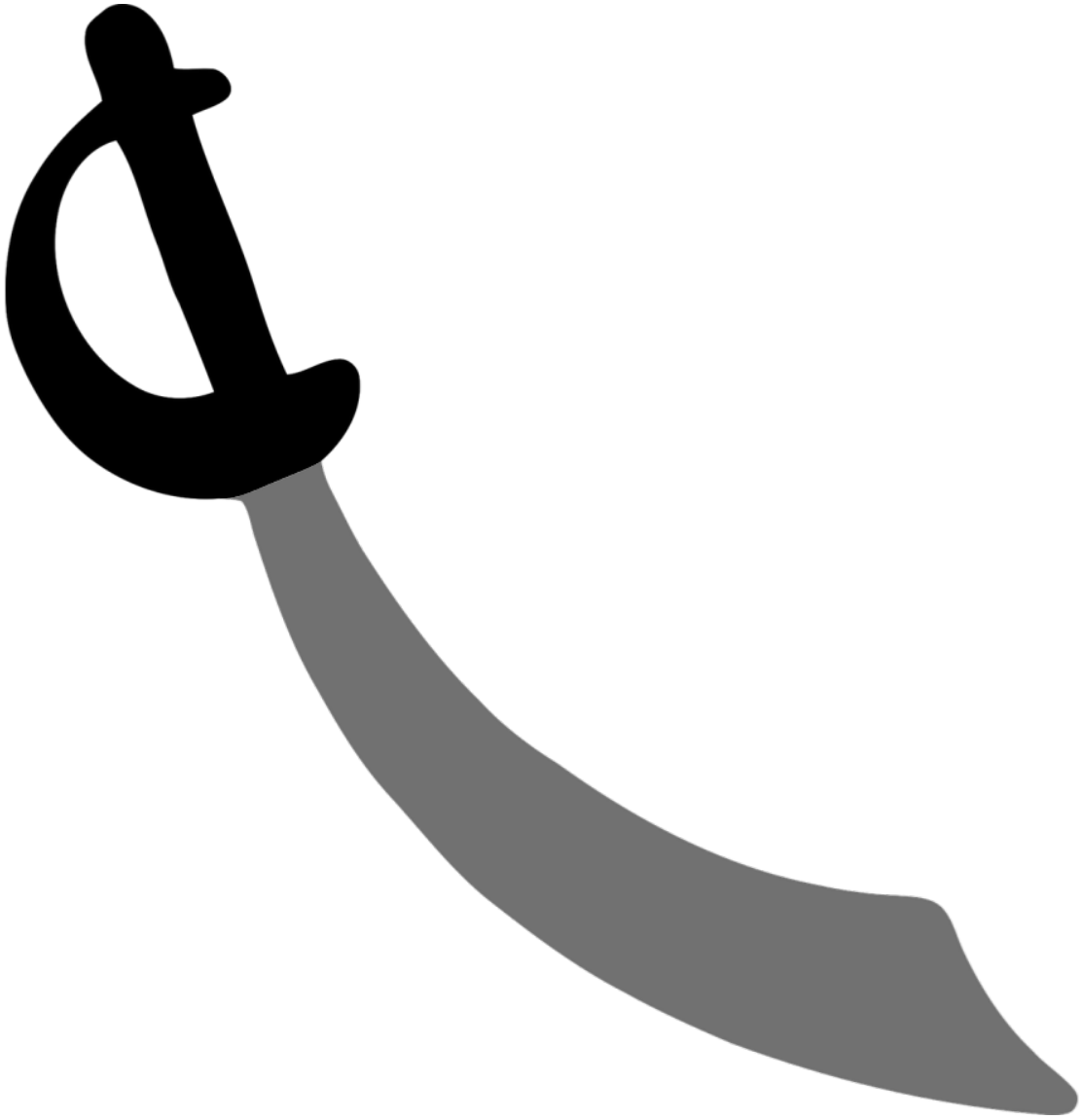
Ask the students to think about what would happen if the list were already in order. How many comparisons would be made? [**answer**: n-1, or 11 in this case]

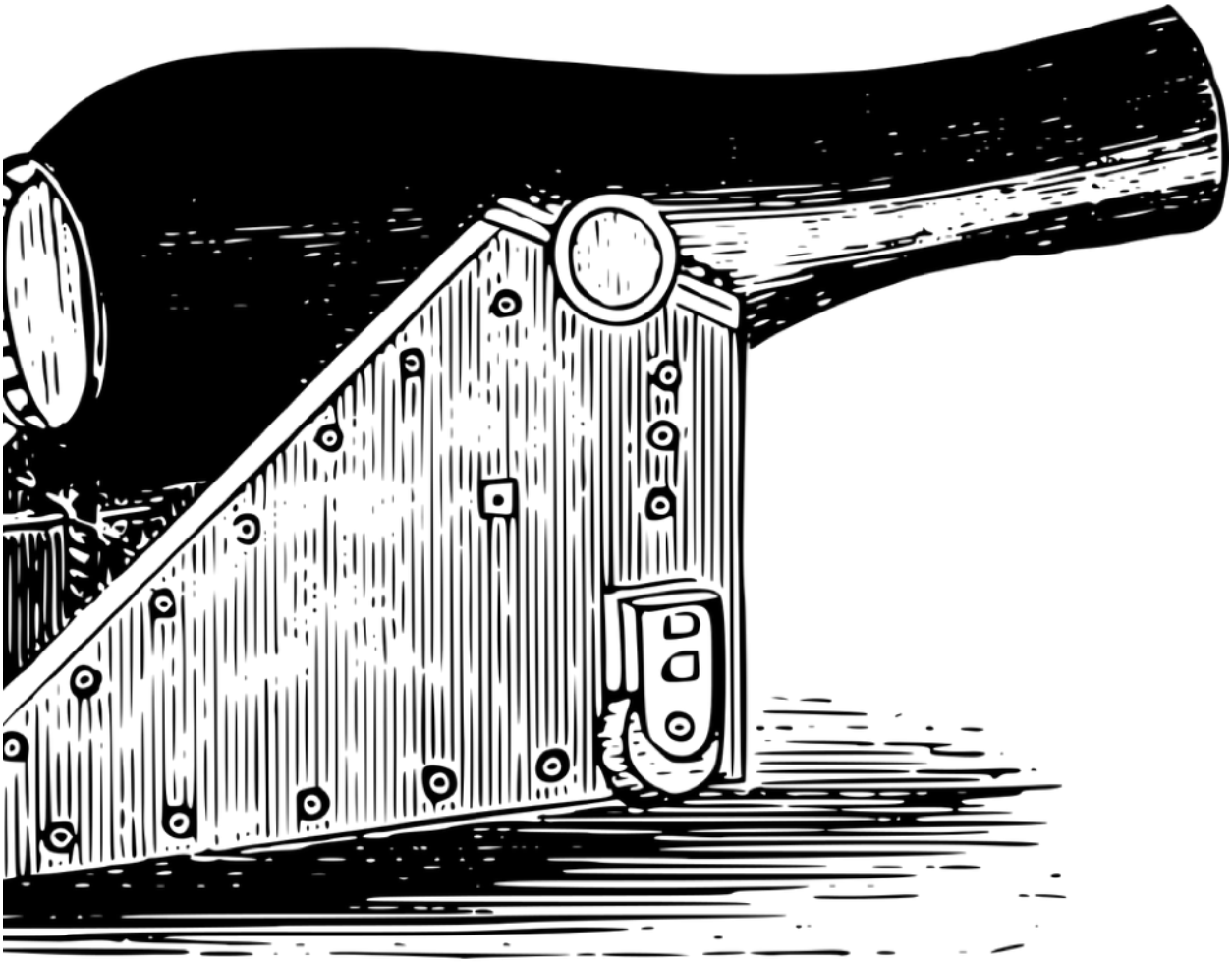
See if the students can figure out what the worst case would be. [**answer**: a list in reverse order]

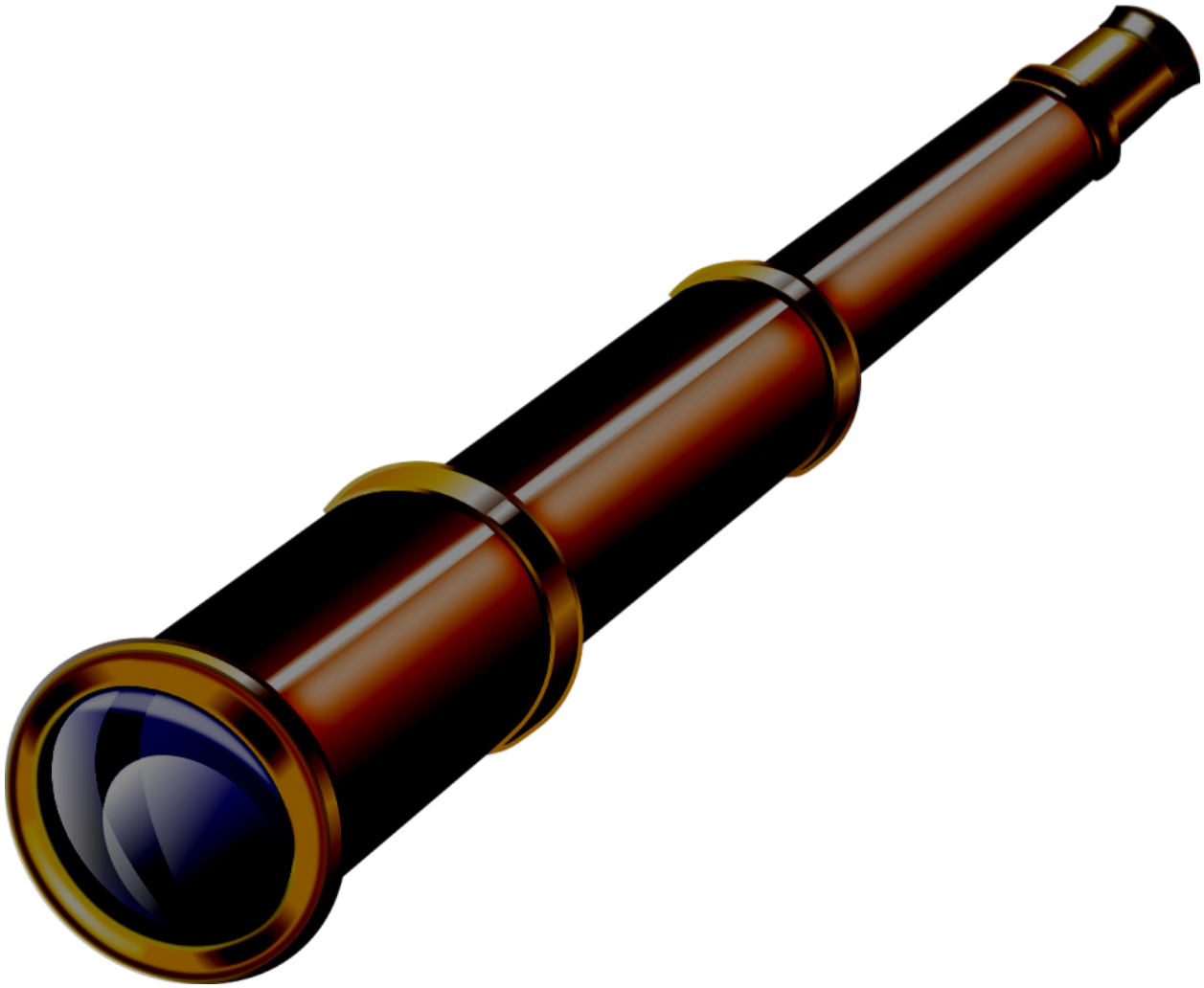
Discuss the pros and cons of each method. Emphasize that insertion sort is much quicker. Selection sort, on the other hand, is a much simpler algorithm.

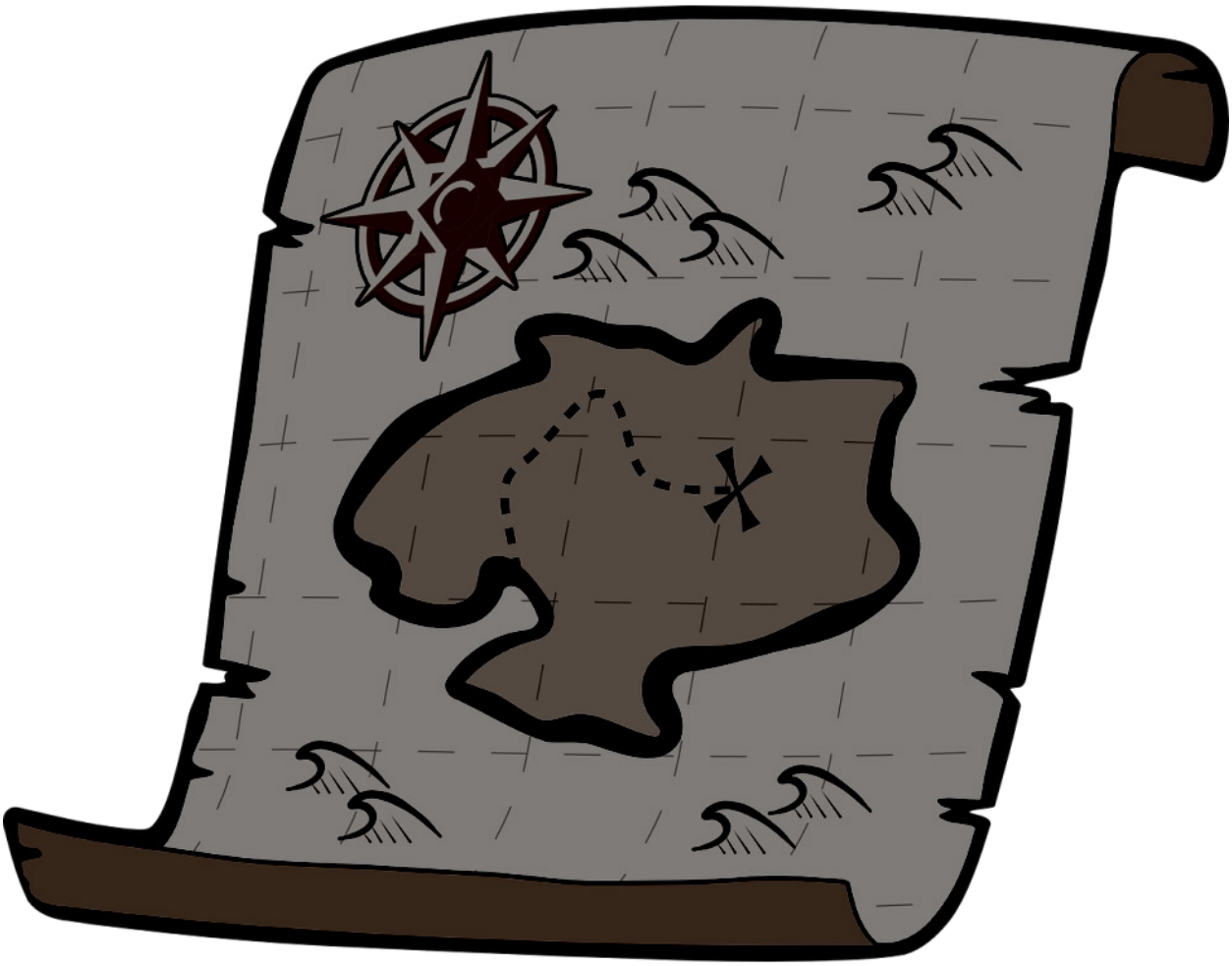


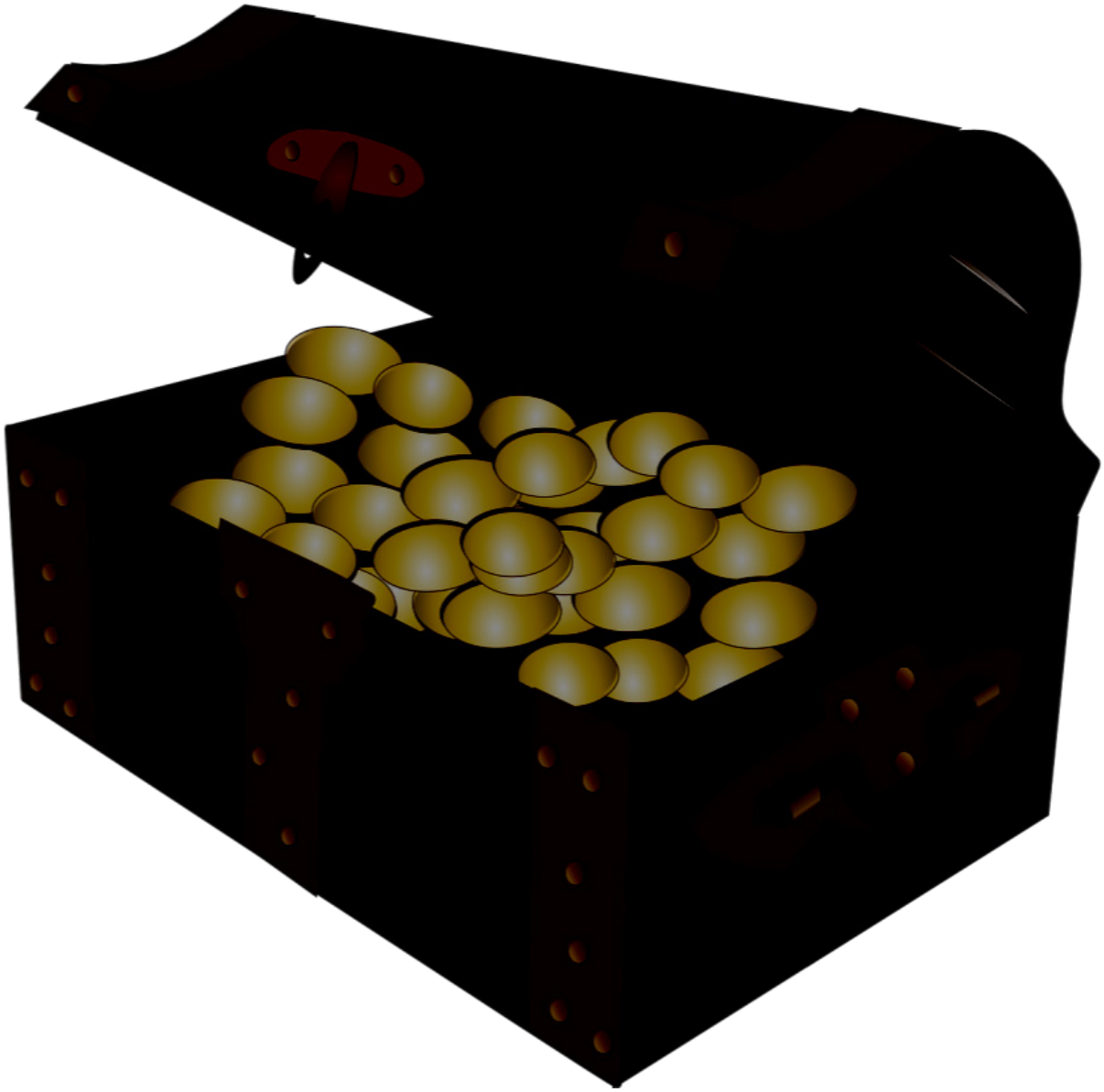

N
NE
E
SE
S
SW
W
NW
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
260
270
280
290
300
310
320
330
340
350



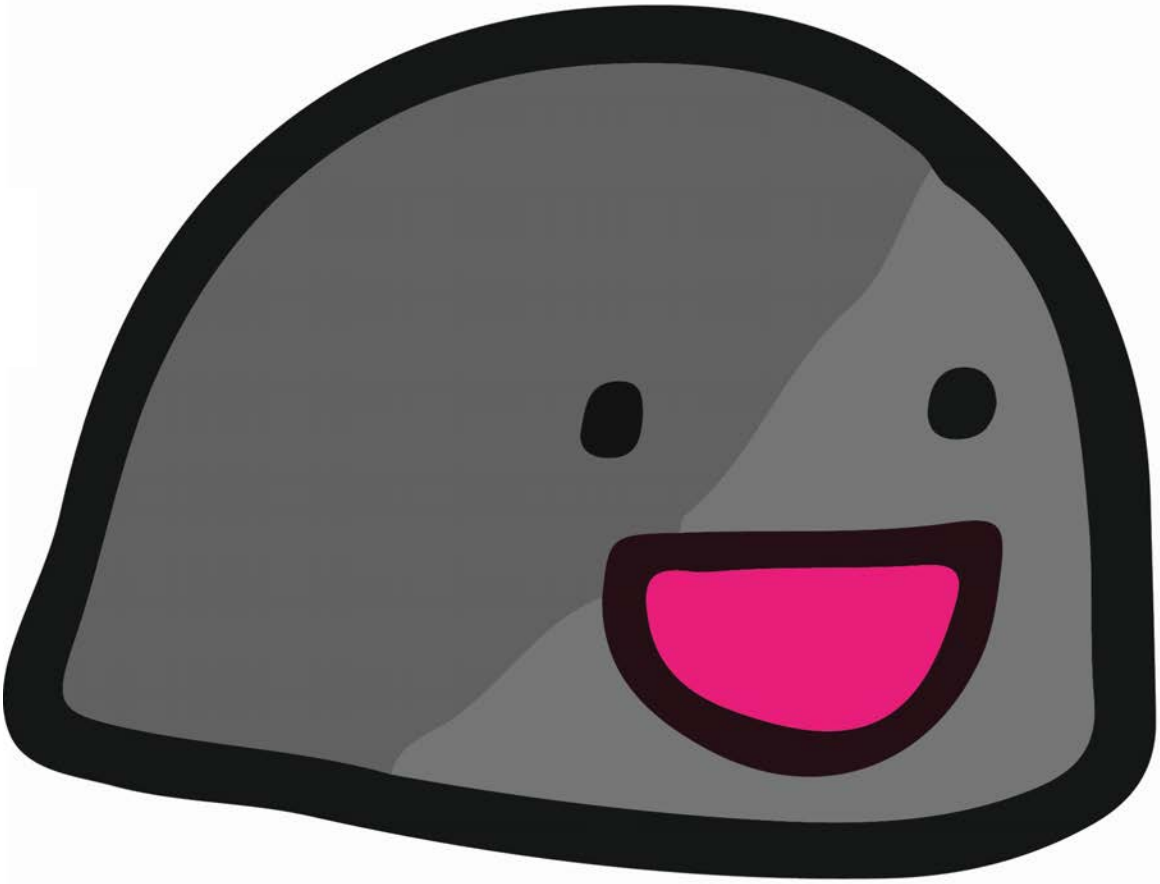




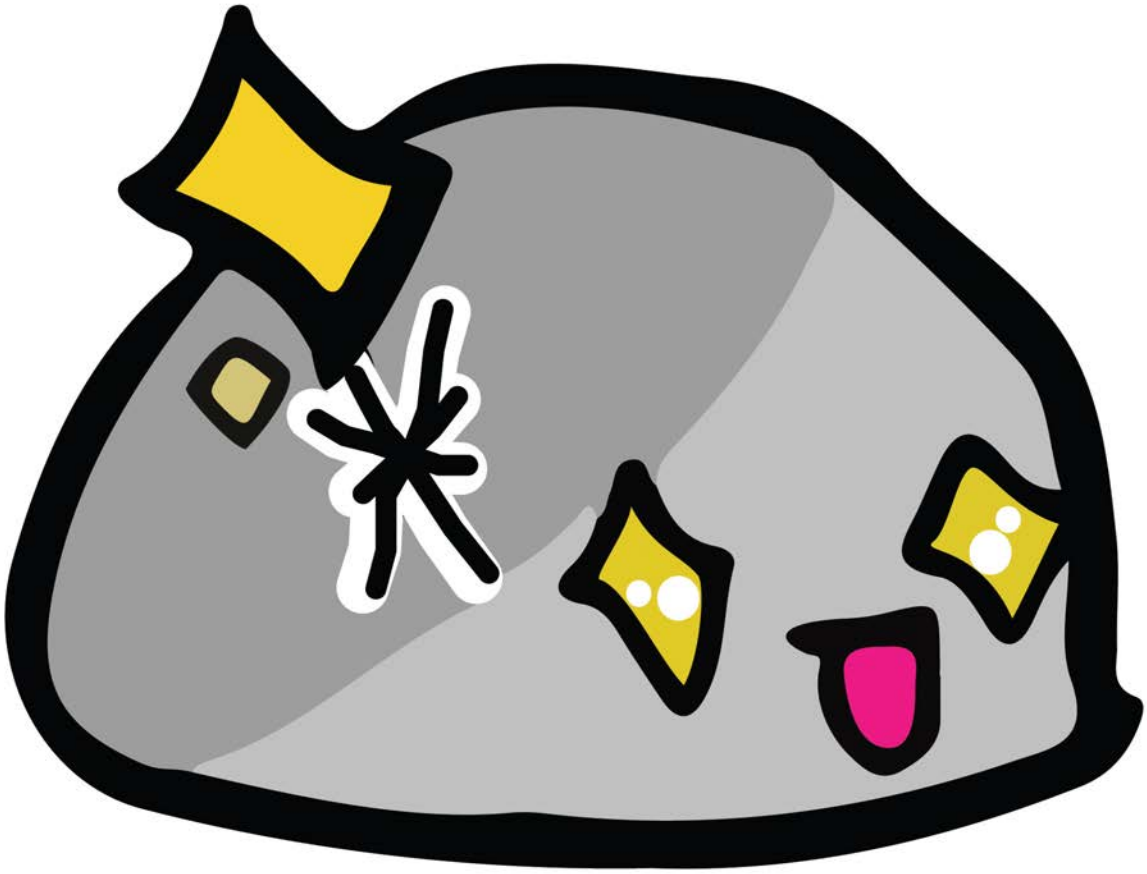




Sleepy

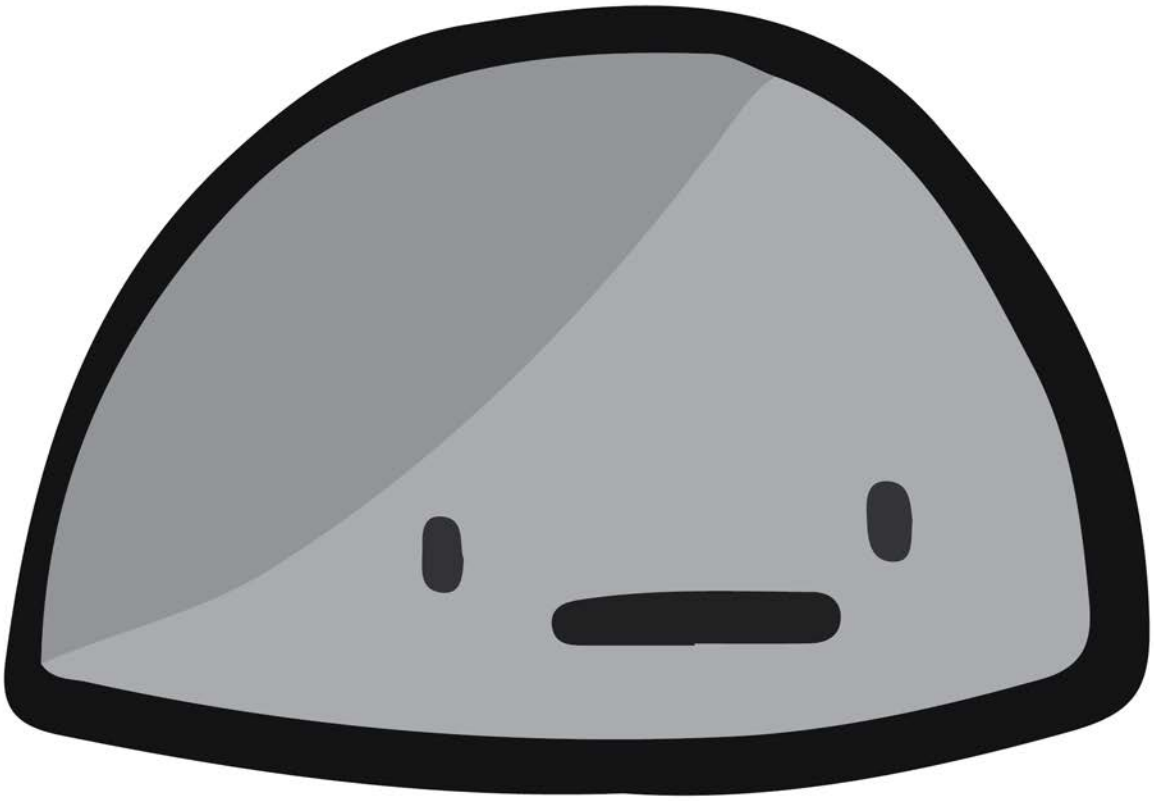


Happy

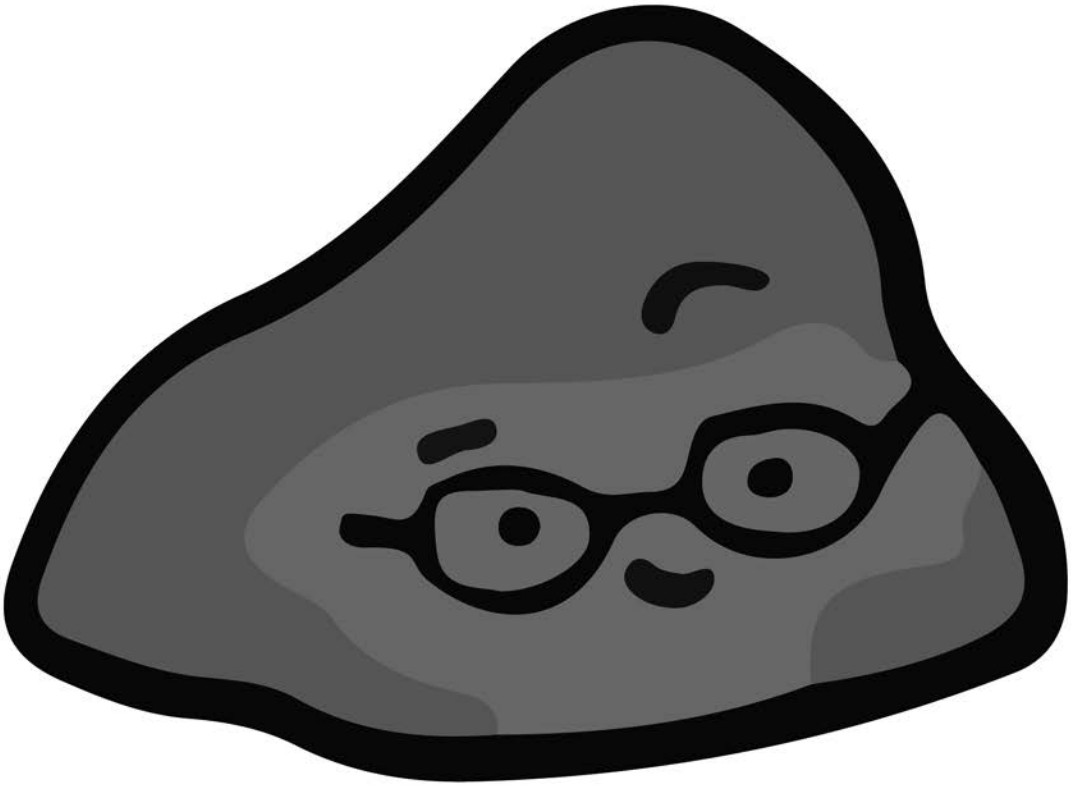


Star

Meanie



Grumpy



Nerdy

Sad

Frustrated

Cool Kid

Clever

Confused

Cutie

SORTING CARDS DEMONSTRATION — *ANSWER KEY*

Demonstration — *Pirate Object Selection Sort*

ordered from lightest to heaviest:

Demonstration — *Pet Rock Insertion Sort*

ordered from least to greatest weight:

1. Sleepy
2. Happy
3. Star
4. Meanie
5. Grumpy
6. Nerdy
7. Sad
8. Frustrated
9. Cool Kid
10. Clever
11. Confused
12. Cutie

WORKSHEET 1 GUIDE — *SORTING COLORS*

Prep: Cut "Secret Weight" slips (page 77) and "Color Dot" slips (page 78).

Split students into groups of two. Each pair will be given a set of "Color Dots." This will enable students to move the colors around. One partner will have the Sorting Colors worksheet, and the other will have the "Secret Weight" slips (only this partner is allowed to see this!).

Students will then have to sort their colors by asking questions about any two colors. Challenge students to see who can sort their colors with the fewest questions. Students are encouraged to use either insertion or selection sort.

Secret Weights – Don't show your partners!

The number represents the "weight" of each color.

(E.g., Yellow is "lighter" than Red because 3 is less than 44.)

RED = 44 GREEN = 88 YELLOW = 3 BROWN = 81 BLUE = 92 GRAY = 14

Secret Weights – Don't show your partners!

The number represents the "weight" of each color.

(E.g., Yellow is "lighter" than Red because 3 is less than 44)

RED = 70 GREEN = 22 YELLOW = 48 BROWN = 81 BLUE = 55 GRAY = 30

Secret Weights – Don't show your partners!

The number represents the "weight" of each color.

(E.g., Yellow is "lighter" than Red because 3 is less than 44.)

RED = 44 GREEN = 88 YELLOW = 3 BROWN = 81 BLUE = 92 GRAY = 14

Secret Weights – Don't show your partners!

The number represents the "weight" of each color.

(E.g., Yellow is "lighter" than Red because 3 is less than 44.)

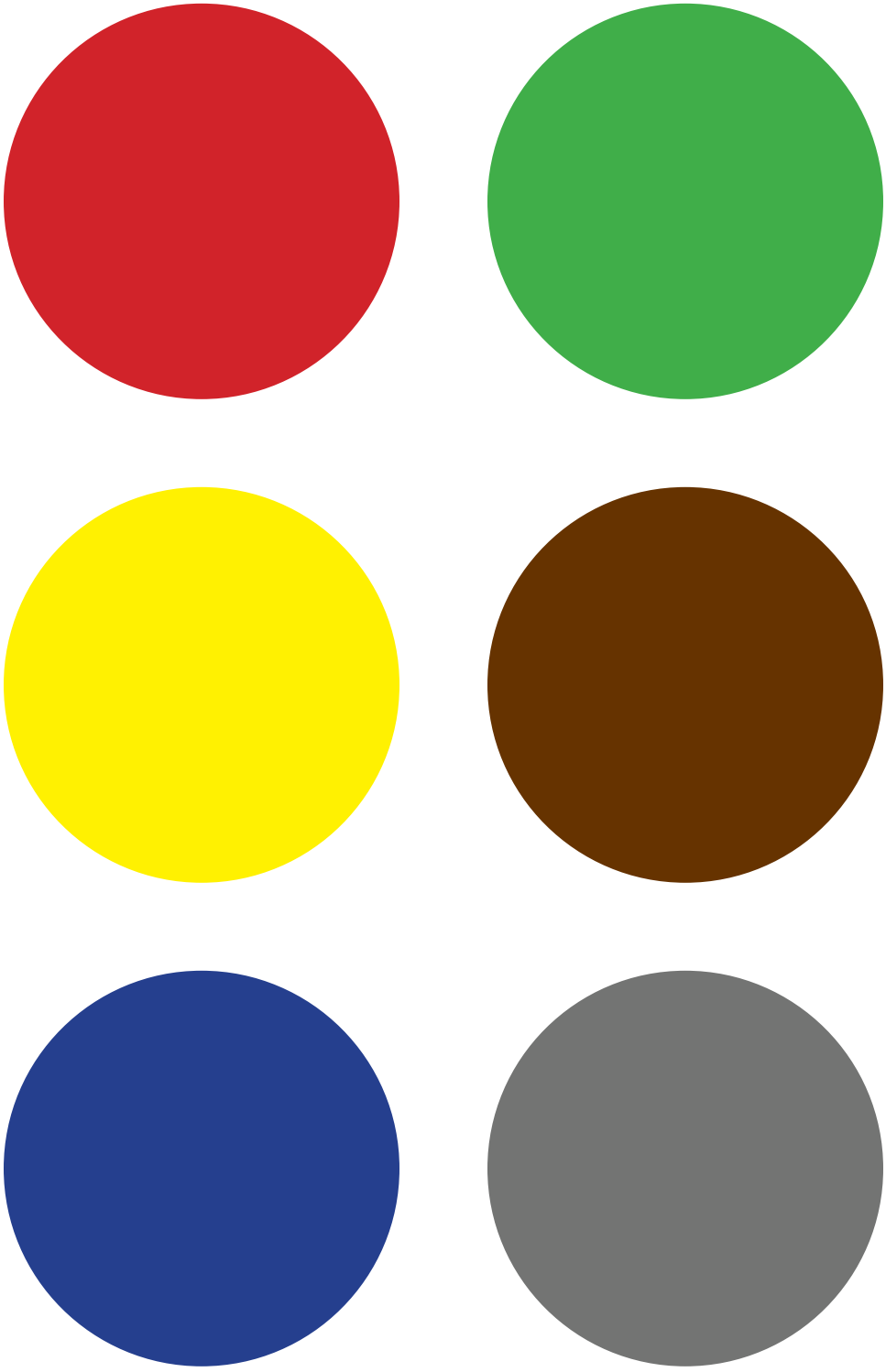
RED = 70 GREEN = 22 YELLOW = 48 BROWN = 81 BLUE = 55 GRAY = 30

Secret Weights – Don't show your partners!

The number represents the "weight" of each color.

(E.g., Yellow is "lighter" than Red because 3 is less than 44.)

RED = 70 GREEN = 22 YELLOW = 48 BROWN = 81 BLUE = 55 GRAY=30



WORKSHEET 1 — *SORTING COLORS*

You have been given 6 Color Dots, with a different color on each card. Each color has a different "weight." Follow the instructions to sort the colors from "lightest" to "heaviest" by asking **yes or no** questions about **two** colors at a time. You will end up sorting weights without actually knowing them!

1. Lay the Color Dot down in the following order: Red, Green, Yellow, Brown, Blue, and Gray. This is not the correct order. Remember, you are trying to ask questions about the colors to put them in the correct order.
2. Ask "yes" or "no" questions to try and find out where each color will go in the "lightest" to "heaviest." (*E.g., Is Red lighter than Green?*)
3. Record your questions and answers to keep track of your work.

If you need more space, use the back of this sheet of paper.

Write your question:	Circle Answer:
1. _____	Yes/No
2. _____	Yes/No
3. _____	Yes/No
4. _____	Yes/No
5. _____	Yes/No
6. _____	Yes/No
7. _____	Yes/No
8. _____	Yes/No
9. _____	Yes/No
10. _____	Yes/No
11. _____	Yes/No
12. _____	Yes/No
13. _____	Yes/No
14. _____	Yes/No

Write down the colors in their *sorted* positions here:

_____ _____ _____ _____ _____ _____

WHAT'S IT ALL ABOUT? — *WHOLE CLASS*

To conclude, discuss with the class some real-life computing examples where sorting is really important. Things like iPods are great examples, because they store so many songs. What a pain it would be if the songs weren't sorted!

Humans often try to stay organized by keeping things sorted. We look at book indexes to find certain sections, libraries by author, and so on. Computers, even though they could easily do a linear search (and much faster than a human), also need to be as efficient as possible. Imagine the Library of Congress having to look through millions of titles using a linear search, or only a handful of titles using binary search.

Because computers use such large amounts of data, sorting data using slow, inefficient methods can waste time. We don't like to wait. And, some people's entire jobs consist of coming up with better, smarter algorithms to get these jobs done faster than ever before.

Demo — Sorting Speeds

Now that students have started to grasp the idea that we can construct a pattern in order to sort a list, explain that there are many ways to methodically sort lists, but some methods are faster than others. There are a number of sites that visually demonstrate sort speed, such as:

» http://www.sorting-algorithms.com/
» http://sorting.at

Explain that computer scientists are interested in how to make these sorting methods as fast as possible.

Activity 5
Beat the Clock: *Sorting Networks*

Summary

Even though computers are fast, there is a limit to how quickly they can solve problems. One way to speed things up is to use several computers to solve different parts of a problem. In this activity, we use sorting networks that do several sorting comparisons at the same time.

Getting Ready

ACTIVITY BREAKDOWN	PAGE	ADDITIONAL FILES IN .ZIP DOWNLOAD	TIME	
Sorting Networks Demonstration	83	no additional files	45 min.	55-100 minutes TOTAL
Sorting Networks Demonstration — *Extension Activity*	84	no additional files	45 min.	
What's It All About?	86	no additional files	10 min.	

Materials

Note: This is an outdoor group activity.

The teacher will need:

» chalk
» one laminated (or sturdy) copy of Photocopy Master — *Sorting Networks* (page 82)
» a stopwatch

Each group of six students will need:

» one sets of six cards cut out from the Photocopy Master — *Sorting Networks* (page 82)

Lesson Preparation

Before diving into the lesson, it is recommended that you watch this video: https://www.youtube.com/watch?v=30WcPnvfiKE (length: 1 minute and 43 seconds).

materials adapted by the Colorado School of Mines with permission from Computer Science Unplugged (http://csunplugged.org)

PHOTOCOPY MASTER — *SORTING NETWORKS*

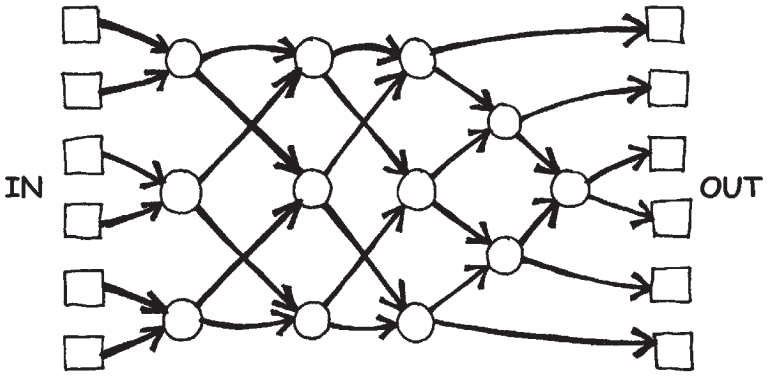
1	2
3	4
5	6

156	221
289	314
422	499

SORTING NETWORKS DEMONSTRATION GUIDE — *WHOLE CLASS*

Instructions for Teacher

Prior to the activity, use chalk to mark out this network on a court.

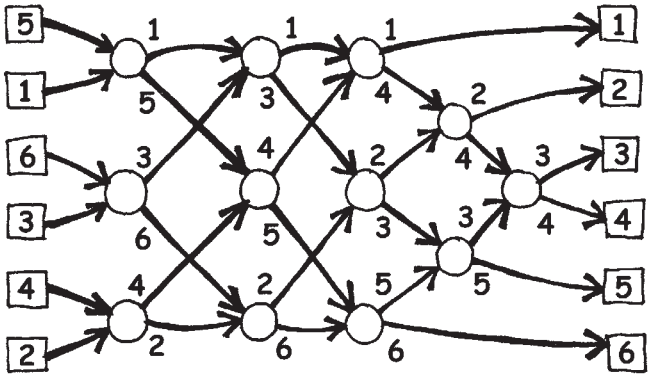


Instructions for Students

This activity will show you how computers sort random numbers into order using something called a sorting network.

1. Organize yourselves into groups of six. Only one team uses the network at a time.
2. Each team member takes a numbered card.
3. Each member stands in a square on the left hand (in) side of the court. Your numbers should be in jumbled order.
4. Move along the lines marked, and when you reach a circle you must wait for someone else to arrive.
5. When another team member arrives in your circle, compare your cards. The person with the smaller number takes the exit to their left. If you have the higher number on your card, take the right exit.
6. Are you in the right order when you get to the other end of the court?

If a team makes an error, the students must start again. Check that you have understood the operation of a node (circle) in the network, where the smaller value goes left and the other goes right. For example:



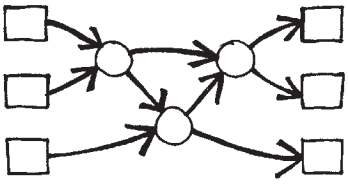
Variations

1. When the students are familiar with the activity, use a stopwatch to time how long each team takes to get through the network.
2. Use cards with larger numbers (e.g., the three-digit ones in the Photocopy Master — *Sorting Networks* on page 82).
3. Make up cards with even larger numbers that will take some effort to compare, or use words and compare them alphabetically.

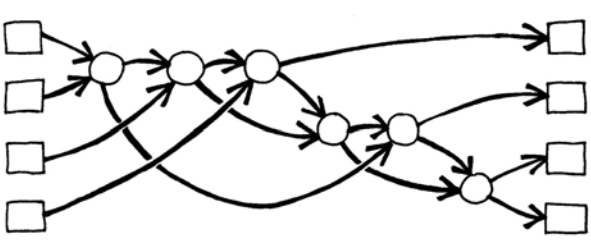
SORTING NETWORKS DEMONSTRATION — *EXTENSION ACTIVITIES*

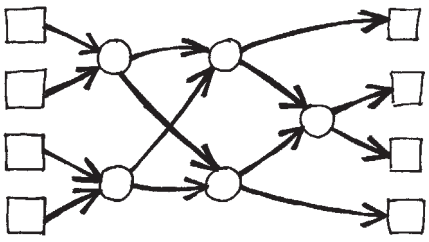
1. What happens if the smaller one goes right instead of left and vice versa? [**answer**: The numbers will be sorted in reverse order.] Does it work if the network is used backwards? [**answer**: It will not necessarily work, and the students should be able to find an example of an input that comes out in the wrong order.]

2. Try to design smaller or larger networks. For example, here is a network that sorts just three numbers. The students should try to come up with this on their own.



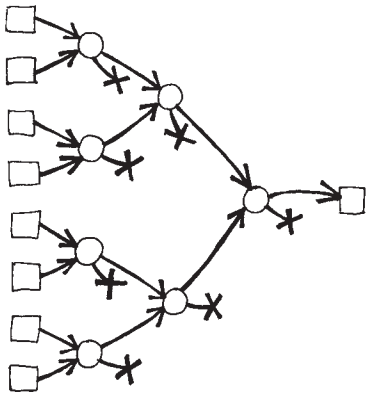
3. Below are two different networks that will sort four inputs. Which is the faster one? [**answer**: The second one is. Whereas the first requires all comparisons to be done serially, one after the other, the second has some being performed at the same time. The first network is an example of serial processing, whereas the second uses parallel processing to run faster.]





4. Try to make a larger sorting network.

5. Networks can also be used to find the minimum or maximum value of the inputs. For example, here is a network with eight inputs, and the single output will contain the minimum of the inputs. (The other values will be left at the dead ends in the network.)



6. What processes from everyday life can or can't be accelerated using parallelism? For example, cooking a meal would be a lot slower using only one cooking element, because the items would have to be cooked one after another. What jobs can be completed faster by employing more people? What jobs can't?

WHAT'S IT ALL ABOUT?

As we use computers more and more, we want them to process information as quickly as possible. One way to increase the speed of a computer is to write programs that use fewer computational steps.

Another way to solve problems faster is to have several computers work on different parts of the same task at the same time. For example, in the 6-number sorting network, although a total of 12 comparisons are used to sort the numbers, up to 3 comparisons are performed simultaneously. This means that the time required would be that needed for just 5 comparison steps. This parallel network sorts the list more than twice as quickly as a system that can only perform one comparison at a time.

Not all tasks can be completed faster by using parallel computation. As an analogy, imagine one person digging a ditch 10 meters long. If 10 people each dug one meter of the ditch, the task would be completed much faster. However, the same strategy could not be applied to a ditch 10 meters deep — the second meter is not accessible until the first meter has been dug. Computer scientists are still actively trying to find the best ways to break up problems so that computers working in parallel can solve them.

Activity 6
Minimal Spanning Trees: *Linking Objects in a Network*

Summary

Many networks link our society: telephone networks, utility supply networks, computer networks, and road networks. For a particular network there is usually some choice about where the roads, cables, or radio links can be placed. We need to find ways of efficiently linking objects in a network.

Getting Ready

ACTIVITY BREAKDOWN	PAGE	ADDITIONAL FILES IN .ZIP DOWNLOAD	TIME	
Worksheet 1 — *Muddy City*	88	no additional files	10 min.	50 minutes TOTAL
Muddy City Discussion	90	no additional files	5 min.	
Worksheet 2 — *Halloween Candy*	94	no additional files	10 min.	
Wrap-up Activity — *DIY Minimal Spanning Tree*	97	no additional files	15 min.	
What's It All About?	99	no additional files	10 min.	

Materials

Each student will need:

- Worksheet 1 — *Muddy City*
- Worksheet 2 — *Halloween Candy*
- 50 Q-tips
- 10 pennies or buttons

Lesson Preparation

Before diving into the lesson, it is recommended that you watch "CS Unplugged: DIY Minimal Spanning Tree," which can be found here: https://www.youtube.com/watch?v=pInCyryBtms&index=3&list=PL-7FSY8VA_YBo1cHdhXwrkG1EkK8nvUne (length: 49 seconds).

materials adapted by the Colorado School of Mines with permission from Computer Science Unplugged (http://csunplugged.org)

WORKSHEET 1 INTRODUCTION— *MUDDY CITY*

This activity will show how computers are used to find the best solutions for real-life problems, such as how to link power lines between houses.

Terminology

» *graph* — a network of nodes

» *cost* — the amount of resources it takes to connect two nodes together

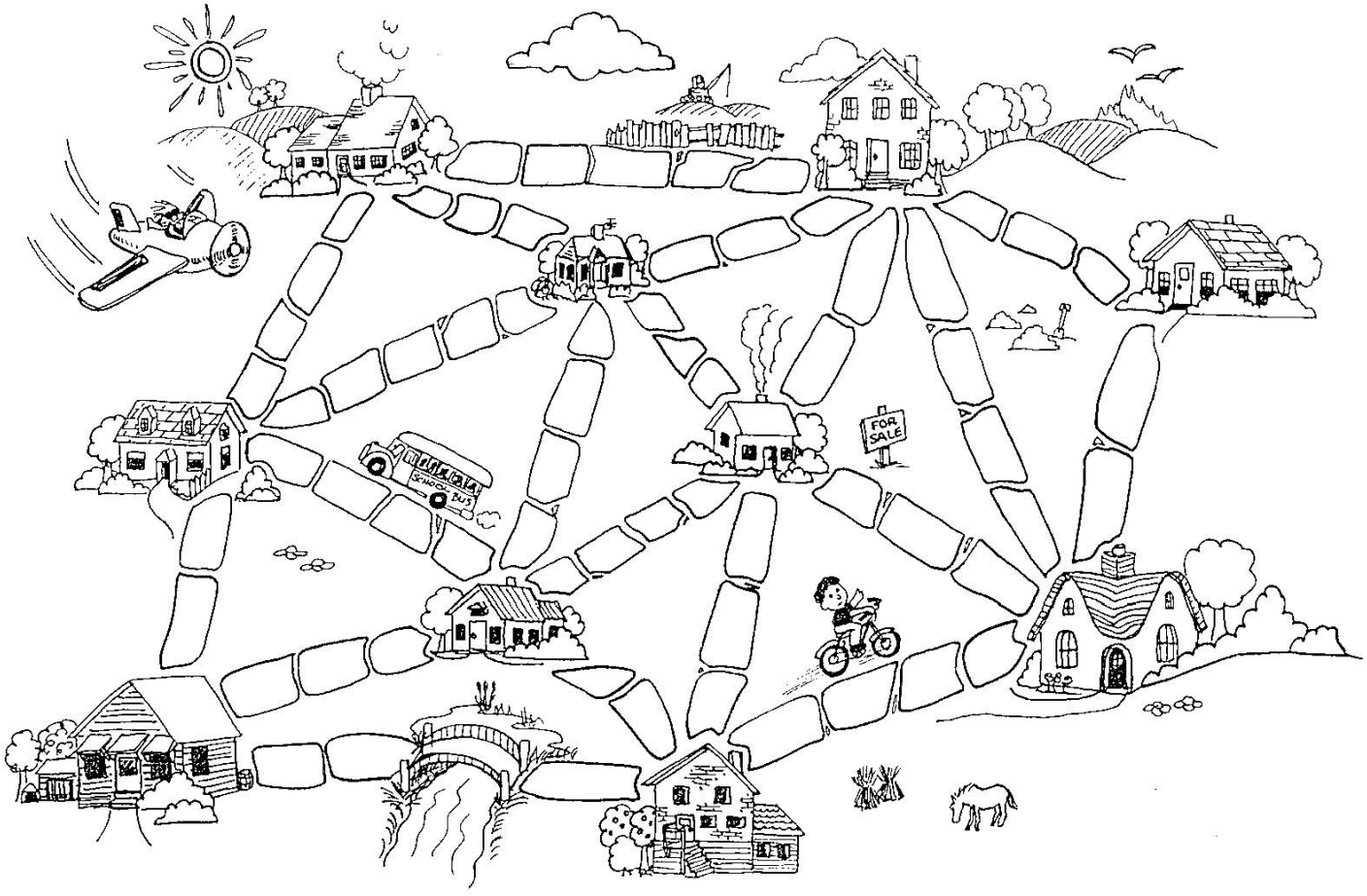
» *Minimal Spanning Tree* — a graph where every node is connected to the graph for the overall cheapest cost

WORKSHEET 1 — *MUDDY CITY*

Once upon a time, there was a city that had no roads. Getting around the city was particularly difficult after rainstorms, because the ground became very muddy — cars got stuck in the mud, and people got their boots dirty. The mayor of the city decided that some of the streets must be paved, but didn't want to spend more money than necessary because the city also wanted to build a swimming pool. The mayor therefore specified two conditions:

1. Enough streets must be paved so that it is possible for everyone to travel from their house to anyone else's house only along paved roads.
2. The paving should cost as little as possible. The cost to pave one block is $1.

The number of paving stones between each house represents the cost of paving that route. Find the best route that connects all of the houses, but uses as few counters (paving stones) as possible



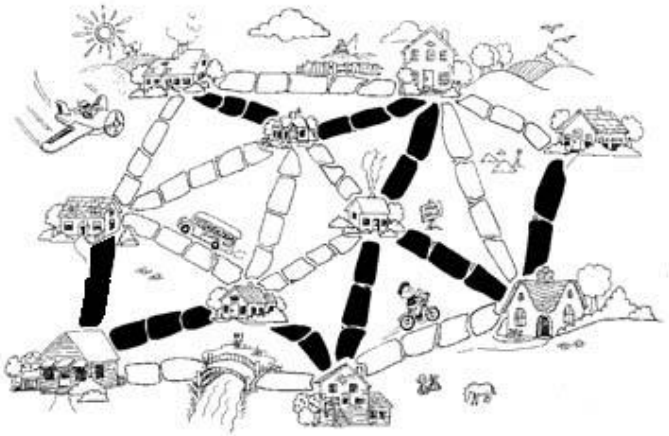
MUDDY CITY DISCUSSION — *WHOLE CLASS*

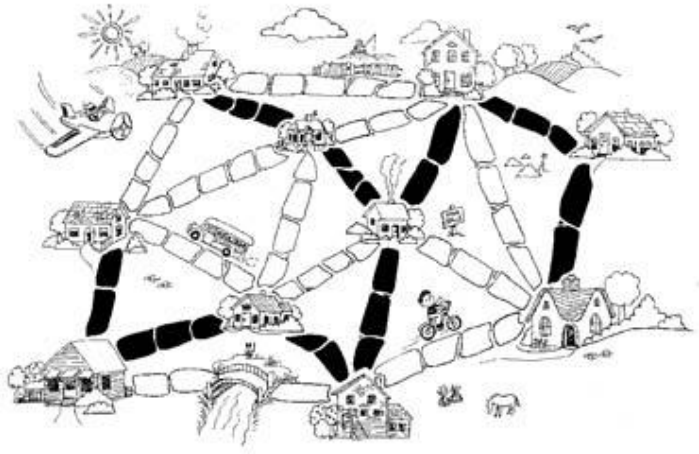
Lesson Vocabulary (You may want to write these terms on the board.)

- » graph
- » abstraction
- » algorithm
- » minimal spanning tree (optional)
- » Kruskal (optional)

Discuss the solutions the students have found. What strategies did they use?

One good strategy to find the best solution is to start with an empty map, and gradually add counters until all of the houses are linked, adding the paths in increasing order of length, but not linking houses that are already linked. Different solutions are found if you change the order in which paths of the same length are added. Two possible solutions are shown below.



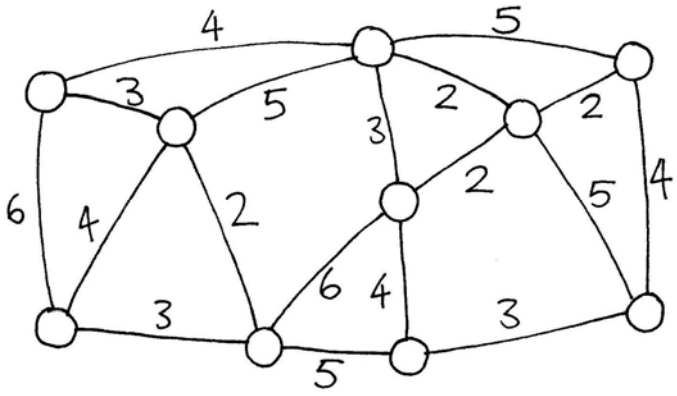


Another strategy is to start with all of the paths paved, and then remove paths you don't need. This takes much more effort, however.

Where would you find networks in real life?

a solution using graphs

Here is another way of representing the cities and roads. The houses are represented by circles, the muddy roads by lines, and the length of a road is given by the number beside the line.

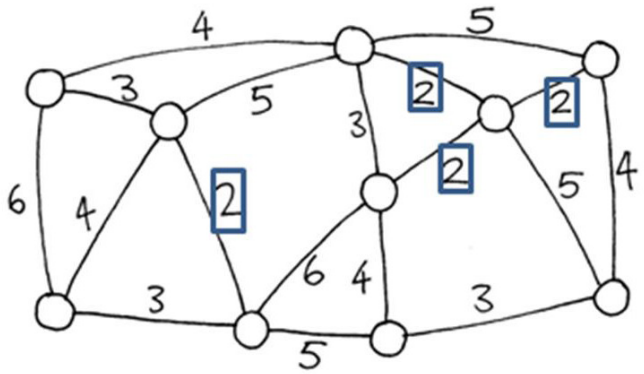


Computer scientists and mathematicians often use this sort of diagram to represent these problems. They call it a ***graph***. This may be confusing at first because "graph" is sometimes used in statistics to mean a chart displaying numerical data, such as a bar graph, but the graphs that computer scientists use are not related to these. The lengths do not have to be drawn to scale.

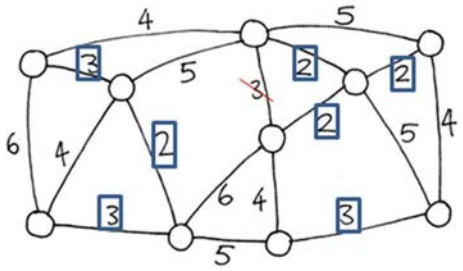
The advantage of using this representation is that it focuses on just the important details. Computer scientists call this ***abstraction***.

Using Kruskal's ***algorithm (step-by-step process)***, a minimal spanning tree can be constructed by starting with the lowest cost edge (line), and then continually adding the next lowest cost edge until all vertices (circles) have been reached.

In the above graph, first shade in all the edges with a cost of 2 (you may want to shade entire line).

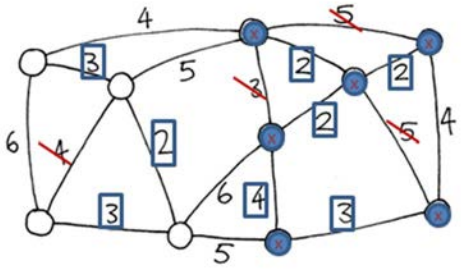


Next, shade in edges of cost 3, *unless* the two circles (houses) have already been attached to the network. **NOTES:** 1) It may seem confusing that we aren't starting in one place and continuing to draw a connected path. Because we already know the entire graph, we can make more holistic/effective decisions. 2) The path of cost 3 with the red line through it is not needed, because the two circles (houses) can already be reached via paths with a lower cost.



At this point, each house has at least one paved path leading to it. But, are they all connected? Let's ensure that we can get to every house.

- » Pick a house at random to start and fill it in (shown in the figure below as a blue circle with x).
- » Fill in every house that can be reached directly from that house. For example, if you start with the house in the top right, you would now shade the house that's 2 pavers away.
- » Continue this process until you don't have a path to follow. Note that in some cases you won't be able to get directly from one house to the next. Continuing the example, from the second house there are two more houses that can be reached on paths of size 2. These houses are both connected to the network... but not directly to each other! That's OK, our goal is to minimize overall cost of paving.
- » If two houses are connected but there's a redundant, more expensive path, cross it out. In our example, we can cross out the path of size 5 from our starting point in the top right, since we have another way to get to the house on the other end of that path. After connecting two more houses, we can cross out a second path of size 5.



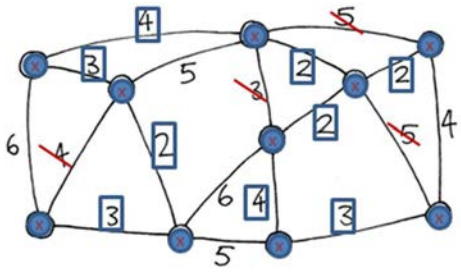
Now ask the students if all houses are connected. From the figure, we can see that it's possible to travel from any house with a filled circle to any other house that's filled ... but there's no paved path to the four remaining houses.

Make sure the students understand this point. There are six houses that are connected. The other four houses are also connected to each other. So once we have a path between those two parts of the graph, we will be done.

Ask the students which path to choose. There are four ways to connect the two subgraphs, but we want to pick the lowest cost path. So we choose to pave the path of cost 4 at the top of the graph.

Discuss: What if there were two lowest-cost paths? [**answer**: We could choose either one of them.]

A final solution is shown below. We have verified that we can get to every house. It is a minimal solution because for n houses (10 in this example), we need n-1 paths (i.e., 9). Adding extra paths would be redundant because we only need one path to each house.



WORKSHEET 2 INTRODUCTION — *HALLOWEEN CANDY*

Now that the students are familiar with the concept of minimum spanning trees, pass out the Worksheet 2 — *Halloween Candy* (page 95). Similar to Worksheet 1 — *Muddy City* (page 89), have students create an minimal spanning tree with Halloween candy.

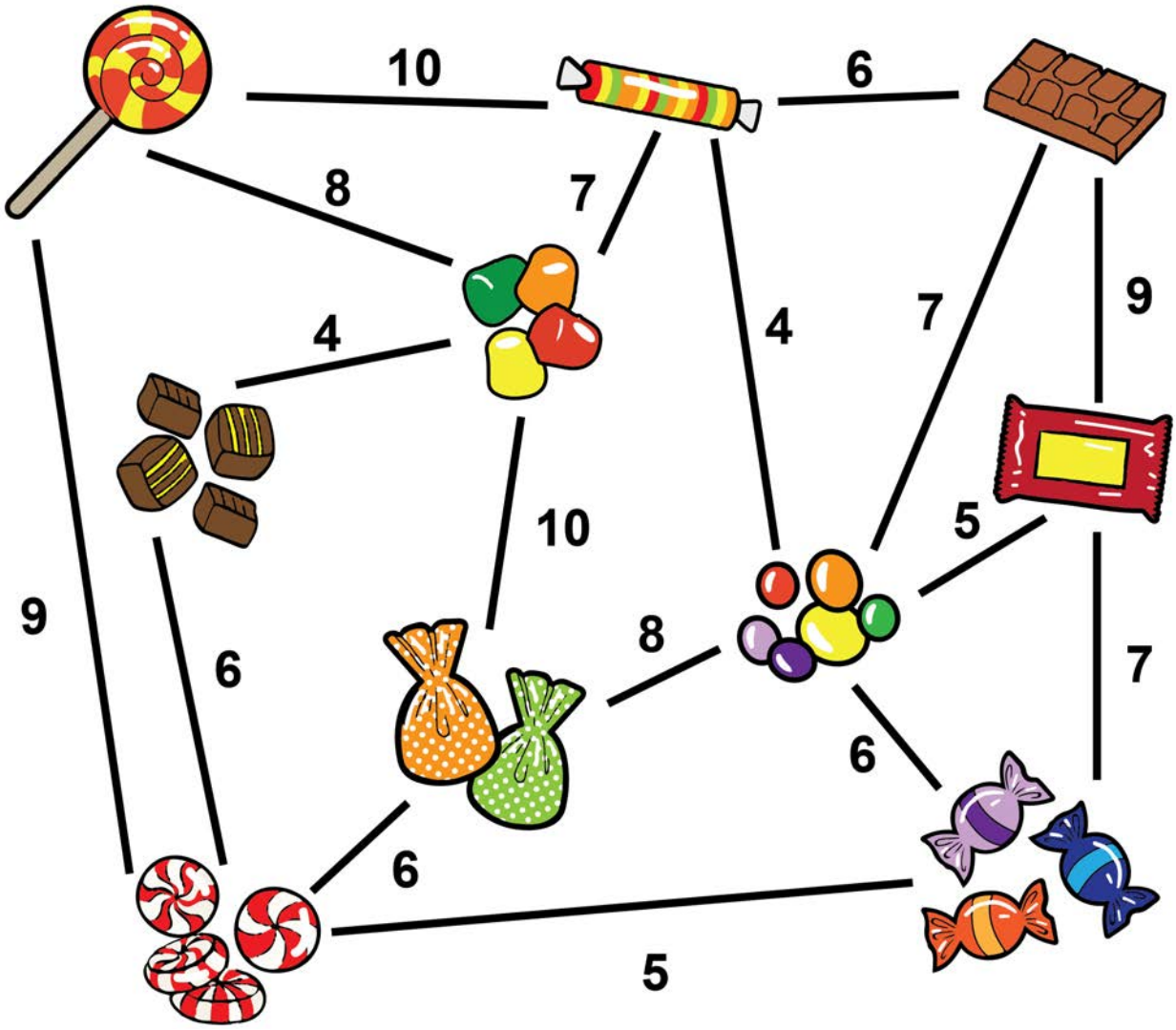
WORKSHEET 2 — *HALLOWEEN CANDY*

Halloween is coming! Your class at school is in charge of decorating the streets. You've been given money to do the decorating, and any money that you don't spend is yours to keep.

The layout below shows what type of candy each house is giving out. The numbers between the houses represent the cost of decorating that street. You use the layout to decide which streets to decorate, based on two conditions:

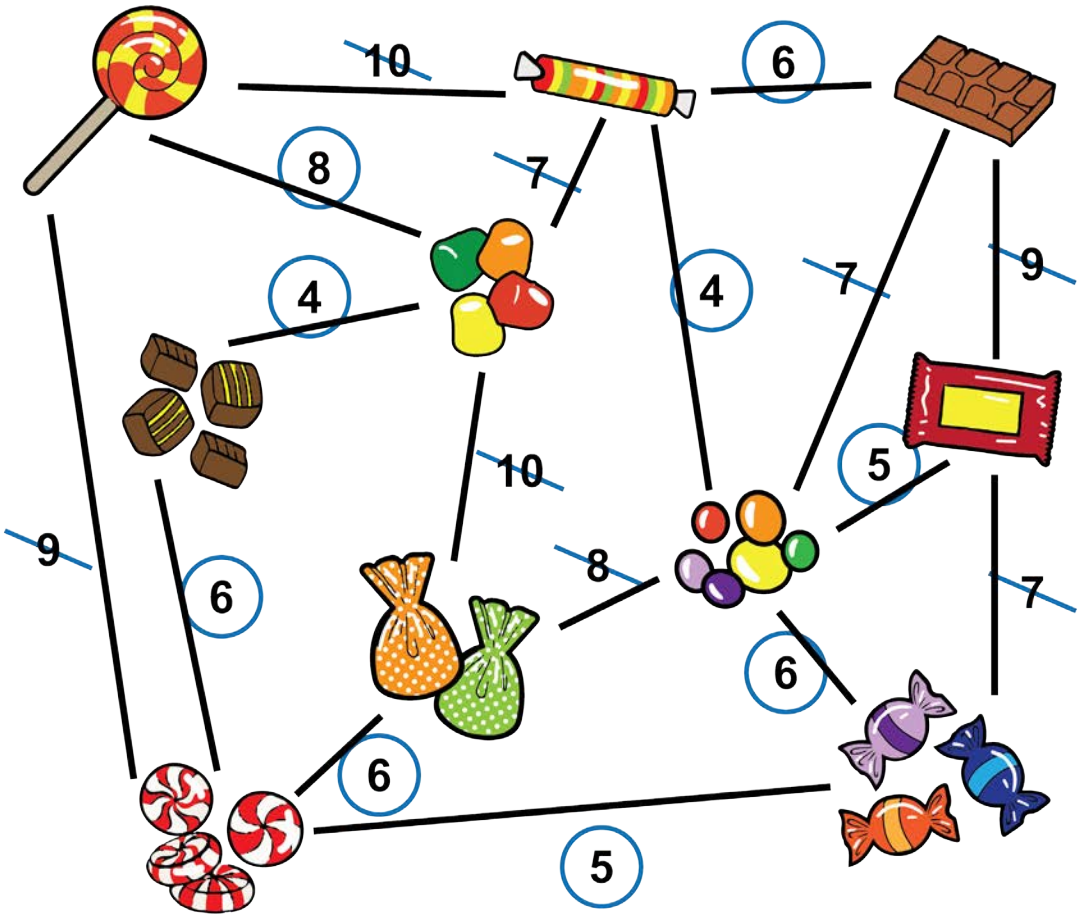
1. Enough streets must be decorated so that it is possible for everyone to travel from their house to anyone else's house only along decorated roads (so everyone can get all types of candy), and
2. The decorating should cost as little as possible.

On the layout below, circle all the streets that you plan to decorate.



WORKSHEET 2 ANSWER KEY — *HALLOWEEN CANDY*

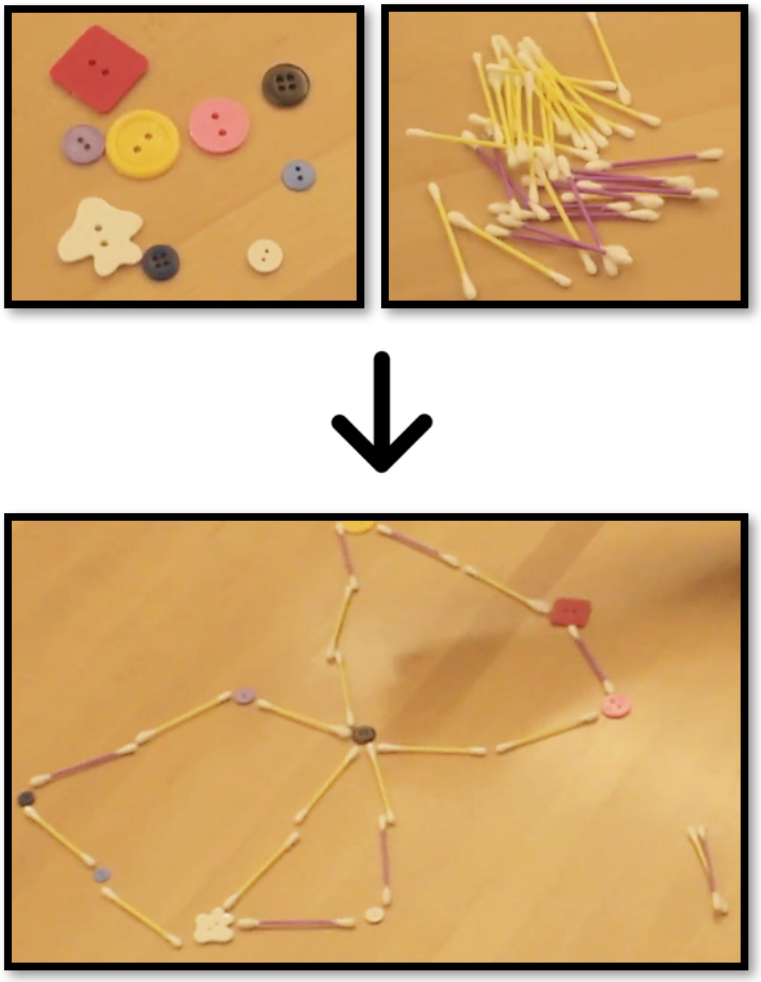
We are just asking the students to circle the streets that they would decorate. The figure below also has the extra streets crossed out. It might be good to reinforce that you are minimizing the decorating costs, *not* creating a complete circuit (that would be a different type of *algorithm*).



WRAP-UP ACTIVITY – *DIY MINIMAL SPANNING TREE*

As an exploratory activity (if time permits), split students up into groups of two. Have each pair of students construct a connected graph using buttons (or pennies) and Q-tips. Then have two groups swap and try to find the minimum spanning tree of the graph. Have each group check each other's Minimal Spanning Tree.

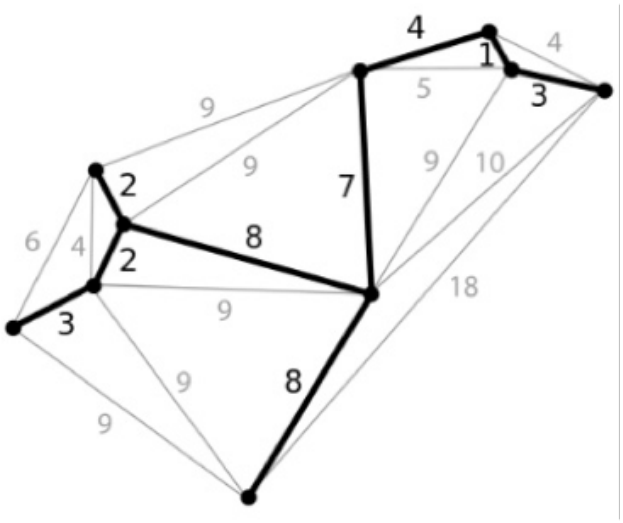
WRAP-UP ACTIVITY ANSWER KEY – *DIY MINIMAL SPANNING TREE*



WHAT'S IT ALL ABOUT? — *WHOLE CLASS*

Suppose you are designing how a utility such as electricity, gas, or water should be delivered to a new community. A network of wires or pipes is needed to connect all of the houses to the utility company. Every house needs to be connected into the network at some point, but the route taken by the utility to get to the house doesn't really matter, just so long as a route exists.

1. Power/Phone Lines: In the graph shown to the right, each node could represent a house in a neighborhood and each line represents the power line connection between each house. A city may wish to connect all houses with power lines at minimum cost. This would be a Minimal Spanning Tree example. A cable company might use the same algorithm for connecting customers to the cable network.



2. Airline: In the graph shown to the right, each node could represent a city, and each line represents the cost of fuel. The airline company might wish to connect all cities while minimizing fuel cost.

Minimal spanning trees help us solve a variety of network design problems. However, when deciding the best routes for people to travel, you do have to take into account how convenient the trip will be for the traveler as well as how much it will cost. No one wants to spend hours in an airplane taking the long way round just because it is cheaper for the airline. The muddy city algorithm may not be much use for these networks, because it simply minimizes the total length of the roads or flight paths.

In this lesson we have studied one efficient method for solving minimal spanning tree problems. This is called Kruskal's algorithm after J.B. Kruskal, who published it in 1956.

Minimal spanning trees are also useful as one of the steps for solving other problems on graphs, such as the "traveling salesperson problem" which tries to find the shortest route that visits every point in the network. For many problems on graphs, including the "traveling salesperson problem," computer scientists have yet to find fast enough methods that find the best possible solution.

There are also many other algorithms that can be applied to graphs, such as finding the shortest distance between two points, or the shortest route that visits all the points.

Activity 7
Searching: *The Binary Search Algorithm*

Summary

When searching for an item in a list, using a strategic searching method is useful. For example, when looking up a word in the dictionary, most people flip back and forth using alphabetical order to find the word, rather than starting from the first word and reading every single word consecutively until the correct word is found. How tiring that would be! When a computer is searching through a set of data, it is important to use an appropriate searching method to minimize time spent searching. This lesson introduces the binary search algorithm and explores its utility and application.

Getting Ready

ACTIVITY BREAKDOWN	PAGE	ADDITIONAL FILES IN .ZIP DOWNLOAD	TIME	
Worksheet 1 — *Raffle #1*	101	no additional files	10 min.	50 minutes TOTAL
Worksheet 2 — *Raffle #2*	101	no additional files		
Ping-Pong Ball Demonstration	104	no additional files	10 min.	
Guess My Number Demonstration	105	no additional files	10 min.	
Dragons and Cows Worksheet 3 — *Daffodai Day 1*	108	no additional files	10 min.	
Dragons and Cows Worksheet 4 — *Draconia Day 2*	108	no additional files		
Lion Hunting Demonstration	111	no additional files	5 min.	
What's It All About?	112	no additional files	5 min.	

Materials

Each teacher will need:

- 15 cups
- 15 ping-pong balls

Each pair of students will need:

- Worksheet 1 — *Raffle #1*
- Worksheet 2 — *Raffle #2*
- Dragons and Cows Worksheet 3 — *Daffodai Day 1*
- Dragons and Cows Worksheet 4 — *Draconia Day 2*

Lesson Preparation

Before diving into the lesson, it is recommended that you watch "Unplugged: Binary Search," which can be found here: https://www.youtube.com/watch?v=iDVH3oCTc2c (length: 2 minutes and 55 seconds). The video will give a brief introduction into binary search and will demonstrate how the Ping-Pong activity works.

materials adapted by the Colorado School of Mines with permission from Computer Science Unplugged (http://csunplugged.org)

WORKSHEETS 1 AND 2 GUIDE — *RAFFLE TICKETS*

Students need to work in pairs for this activity. Each pair will need a half of Worksheet 1 — *Raffle #1* (page 102) and a half of Worksheet 2— *Raffle #2* (page 103). One student will receive the half labeled "RAFFLE HOLDER," and the other student will receive the sheet labeled "RAFFLE MASTER." ***Partners cannot look at each other's papers during the activity.***

Introduce the activity. Tell students that the school has set up a raffle and that each student may have won a prize! However, the students aren't sure which ticket is the winning ticket. Each student will get the chance to be both the Raffle Holder and the Raffle Master. The goal is for each student to find their winning ticket.

Student A, holding the "RAFFLE HOLDER #1" half of the sheet, will go first. The partner (Student B) should be using the "RAFFLE MASTER #1" half of the same sheet. Student A will guess a ticket and write that guess on their page (e.g., ticket "C"). Student B will look at the raffle master list, and either say "yes" (if the number is the winner, e.g., 189) or "no" if some other number. This continues until Student A guesses the correct ticket.

The students will then use Worksheet 2 — *Raffle #2* (page 103) to swap roles, and Student B will make guesses as the Raffle Holder while Student A is the Raffle Master.

The goal of this activity is to demonstrate the variance in the number of guesses it takes to find the winning raffle ticket by randomly guessing.

After this round, discuss with the class to see how many guesses groups took. Who had the lowest number of guesses? Who had the most? You may want to write the number of guesses on the board, to give some idea of the variability of using a random approach.

WORKSHEET 1 — RAFFLE TICKET #1

RAFFLE HOLDER #1

Your school is holding a raffle to win a new 65" TV! The numbers have been called out, and you think you might have won! Unfortunately, the numbers on the tickets got worn out, so you have no way of knowing if you actually have the number. You need to talk with a Raffle Master in order to figure out if you have a winning ticket. **The number called out was 189.**

A	B	C	D	E	F
G	H	I	J	K	L
M	N	O	P	Q	R
S	T	U	V	W	Z

Instructions: Find the winning ticket! Guess a letter, and ask your partner (the Raffle Master) what number is on that ticket. Record the order of your guesses in the box below. When you have found the winning ticket, record the number of guesses. Then you will become raffle master!

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

Number of Guesses: _____

RAFFLE MASTER #1 (65" TV!)

Be sure your partner can't see these numbers.

A	B	C	D	E	F
653	98	426	799	32	974
G	**H**	**I**	**J**	**K**	**L**
1348	89	731	26	79	87
M	**N**	**O**	**P**	**Q**	**R**
467	7	294	42	9000	369
S	**T**	**U**	**V**	**W**	**X**
963	360	189	365	765	69

WORKSHEET 2 — RAFFLE TICKET #2

RAFFLE HOLDER #2

A	B	C	D	E	F
G	H	I	J	K	L
M	N	O	P	Q	R
S	T	U	V	W	Z

Your school is holding a raffle to win a new PlayStation! The numbers have been called out, and you think you might have won! Unfortunately, the numbers on the tickets got worn out, so you have no way of knowing if you actually have the number. You need to talk with a raffle master in order to figure out if you have a winning ticket. **The number called out was 303.**

Instructions: Find the winning ticket! Guess a letter and ask your partner (the raffle master) what number is on that ticket. Record the order of your guesses in the box below. When you have found the winning ticket, record the number of guesses.

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

Number of Guesses: _____

RAFFLE MASTER #2 (PLAYSTATION!)

Be sure your partner can't see these numbers.

A	B	C	D	E	F
214	420	834	22	300	99
G	**H**	**I**	**J**	**K**	**L**
1993	12	100	33	79	87
M	**N**	**O**	**P**	**Q**	**R**
1111	2	589	69	231	903
S	**T**	**U**	**V**	**W**	**X**
303	720	681	8008	765	42

PING-PONG BALL DEMONSTRATION GUIDE – *WHOLE CLASS*

Here is a video of this demo to reference: https://www.youtube.com/watch?v=iDVH3oCTc2c (length: 2 minutes and 55 seconds)

Prep: Label 15 Ping-Pong balls with various numbers. These numbers should be random and should NOT be equally incremented. Line up 15 cups with the numbered Ping-Pong balls hidden underneath them. These should be lined up from least to greatest; in other words, they should be *in order*. One way to ensure that the order can be seen by the students is to label the cups alphabetically, with the lowest number being represented by "A" and the highest number being represented by "O." Stress that the numbers are non-sequential but are in order. An example ordering can be seen below.

Have a volunteer come up and give them four pennies. Tell them that they are looking for a specific number. To find the number, they must look under a cup. Every time they look under a cup, they must hand you one penny. Challenge them to search for a specific number with only those four pennies. Though you may be tempted to pick any random position, this demonstration works best when choosing the number in the 1st, 3rd, 5th, 7th, 9th, 11th, 13th, or the 15th cup. (E.g., you might choose 24, which is cup 5, or 83, which is in cup 13.)

Because the Ping-Pong balls are sorted, with each number that is revealed, some of the Ping-Pong balls can be eliminated. If the number they look at is less than the number they are trying to find, that means that all numbers to one side of that number are also less than the number they are trying to find. After the student reveals the first Ping-Pong ball, ask them which portion of the cups should be eliminated. If the student selects an optimal cup, where half of the elements remaining can be eliminated, dramatically sweep the Ping-Pong balls and cups out of line. Otherwise, gently push the eliminated cups to the side. Keep doing this until the student finds the number. After one round, reset the cups and ask the class what the optimal position is in order to create a consistent search.

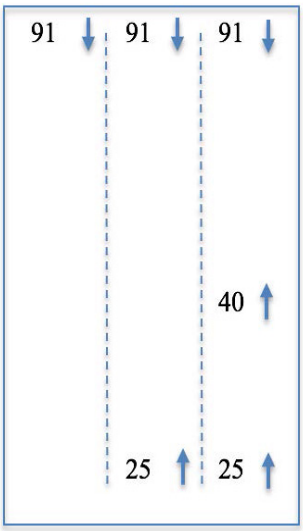
Discuss: How would they search an unordered list for a number? Let them think about this. In the worst case, the number they're looking for will be at the end of the list, or it won't exist. Then, they would have to look at every single number to be certain whether the number can be found.

The idea that a structured and methodical search on a sorted list is more effective and consistent than randomly guessing will be further explored in the next exercise.

GUESS MY NUMBER DEMONSTRATION GUIDE — *WHOLE CLASS*

Lesson Vocabulary (You may want to write these terms on the board.)

» binary search



Tell students that you will pick a number between 1 and 100. Write the numbers 1 (at the bottom) and 100 (at the top) on the board to help emphasize the location of the guesses. You want the class to guess your number. Have the students shout out numbers, and listen for a guess that is NOT optimal (e.g., 50 is optimal, so pick some number close to either end, like 91 in the figure on the right). Call on other students to make continuous guesses until the number has been correctly guessed. Document student guesses on the board vertically, indicating whether the target number is higher or lower than the guess using arrows. The figure shows what might be listed for students trying to guess the number 45 after three rounds of guessing.

IMPORTANT: It's possible that a student might randomly pick your number on the first try (or within a few guesses). In other words, a random process might on occasion out-perform the binary search. For this demo to really emphasize that binary search provides a desirable limit on the number of guesses, it's important for the random guesses not to appear superior. So, in the event that a student guesses the secret number within 10 tries, just change the number that you originally chose in your head! You don't want the demonstration to make it seem that guessing is a better method than binary search while the method is being introduced. Later, it will be good to discuss why guessing does work sometimes but is not reliable. **HINT:** As the students call out numbers, pick the direction (higher or lower) with the largest range. For example, if students guess 7, you wouldn't want to give the direction as less than 7 (there would be only 6 choices left). Instead, say that it's greater than 7 (which leaves 93 choices).

Write the total number of guesses that it took for students to determine the number on the board.

Next, say that you will play the game again but want to use fewer guesses. Lead students to select an appropriate first guess number. For example, if a student guesses the number 20 for their first guess, point out that there are now 80 numbers to guess from above 20. What is the best number to choose as a first guess to minimize the set of remaining numbers to guess from? Emphasize that splitting the list in half (the binary search algorithm) provides an efficient and reliable search method.

Next, have students guess a number between 1 and 1,000 to prove the efficiency of the binary search algorithm. During this round, the focus will be on documenting the number of values that are eliminated from the list with each round of guessing. Choose a student to take the first guess. Write the guess on the board exactly like what was done in the previous guessing game. Emphasize the number of values that have been eliminated from the set of possible guess values due to the binary search algorithm. For example, if the student guesses the number 500, and you state that the value is greater than 500, all 500 numbers below the number 500 have been eliminated. (If the student guesses some number other than 500, engage the class in a discussion of how to choose the optimal guess). Document the number of values that have been eliminated somewhere on the board. Call on a student to guess a second value. Again, emphasize and document the number of values that have been eliminated with this guess (hopefully 250). Continue having students guess values, but this time, instead of documenting each guess on the board, document the number of values that have been eliminated with each guess until the number of values remaining reaches 1. The number of guesses should be no more than 10.

Example: assume the target is 675.

First guess: 500. Eliminates 500. Number is greater.

Second guess: 750. Eliminates 250. Number is lower.

Third guess: 625. Eliminates 125. Number is higher.

Fourth guess: 688 (or 687). Eliminates 63. Number is lower.

Fifth guess: 656. Eliminates 32. Number is higher.

Sixth guess: 672. Eliminates 16. Number is higher.

Seventh guess: 680. Eliminates 8. Number is lower.

Eighth guess: 676. Eliminates 4. Number is lower.

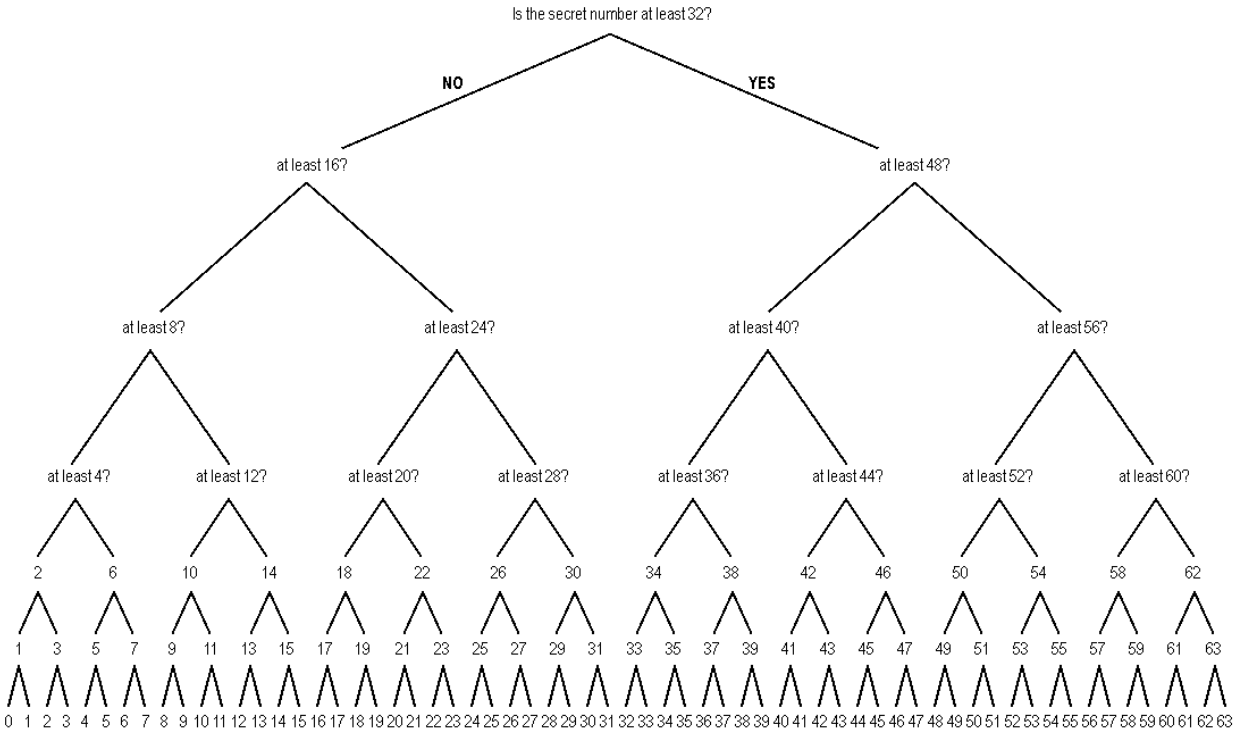
Ninth guess: 674. Eliminates 2. Number is higher.

Tenth guess: 675.

NOTE: Ten is the **maximum** number of guesses. It's possible for the number to be found in fewer than ten guesses. (E.g., if the number were 750, 625, 688, etc.)

DECISION TREES DISCUSSION — *WHOLE CLASS*

As a group discussion, show a fully-formed decision tree for a binary search on numbers from 0 to 63. This decision tree shows why some numbers will take fewer than the maximum number of guesses in a binary search.



DRAGONS AND COWS WORKSHEETS 3 AND 4 GUIDE

Students need to work in pairs for this activity. Each pair will need a set of the Dragons and Cows Records. Partners cannot look at each other's papers during the activity.

Introduce the activity. Tell students that a dragon has invaded the kingdom and is eating cows. Each student will have a chance to prevent the dragon from eating all the cows. This activity is similar to Worksheets 1 and 2 — *Raffle Ticket*, but now the cows are in order by weight, so students should try to use binary search to find the dragon as quickly as possible.

Dragons and Cows Worksheet 3 — *Daffodai Day 1*

The dragon is attacking Daffodai. The student with the top half of Worksheet 3 — *Daffodai Day 1* will go first. The goal is to find which cow the dragon is attacking by guessing a cow and asking how much it weighs. (E.g., How much does M weigh?) The student holding the bottom half of Worksheet 3 — *Daffodai Day 1* has the list of how much each cow weighs.

Dragons and Cows Worksheet 4 — *Draconia Day 2*

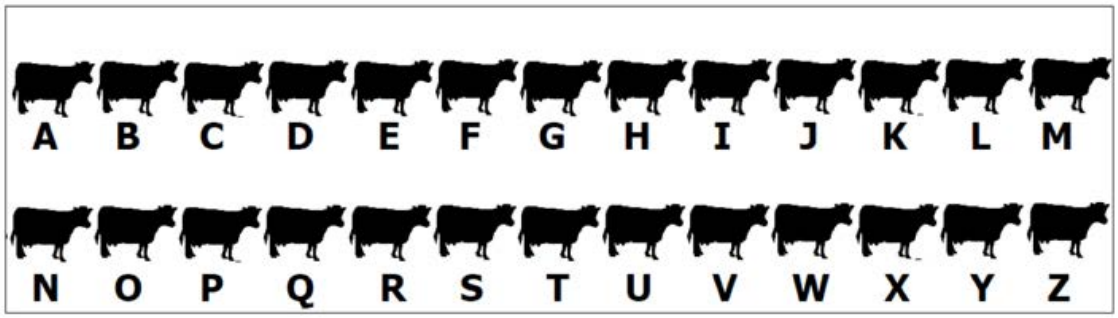
The dragon is attacking Draconia. Partners will switch roles.

If students use Binary Search, they will end up only having to guess at most five times.

DRAGONS AND COWS WORKSHEET 3 — *DAFFODAI DAY 1*

Save Daffodai's Cows: Your livestock were lazily grazing in the pasture *when one cow was suddenly attacked by a dragon*. Your magic ball says the dragon is attacking **the cow that weighs 246 pounds**, but you don't know which cow that is.

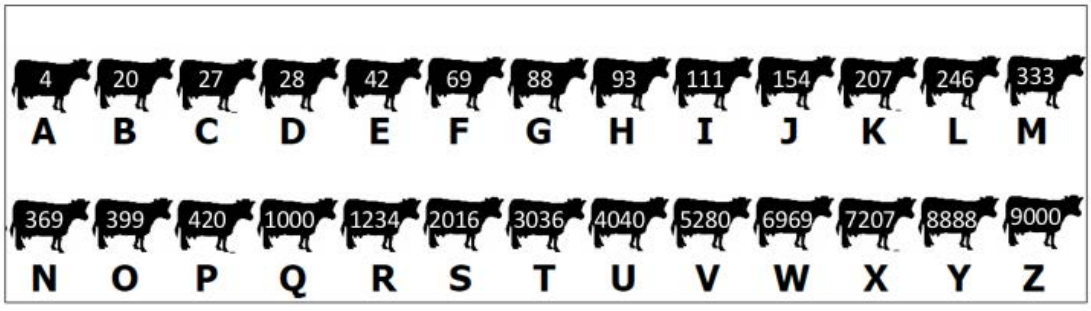
Instructions: If you can find the dragon in **5 guesses or fewer**, you can save all the cows. Your neighbor (partner) has a list of cows, *in order by weight*. Your task: Guess a letter, and ask your partner how much that cow weighs. Record your guesses below. Stop when you find the cow that is being attacked (luckily, just looking at the dragon scares it... so all you need to do is find the cow!)



Guess #	Cow	Guess #	Cow	Guess #	Cow	Guess #	Cow	Guess #	Cow
1		2		3		4		5	

DRAGONS AND COWS WORKSHEET 3 — *DAFFODAI DAY 1*

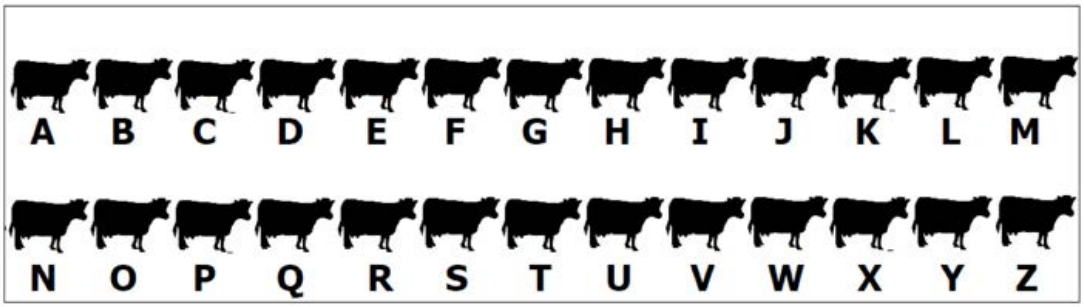
Help your neighbor in Daffodai save the cows! When asked, tell your partner how much the cow weighs.



DRAGONS AND COWS WORKSHEET 4 — *DRACONIA DAY 2*

You thought your cows were safe, but the dragons are back! *A second cow has been attacked by a dragon*. Your trusty magic ball tells you the dragon is attacking **the cow that weighs 7,621 pounds**.

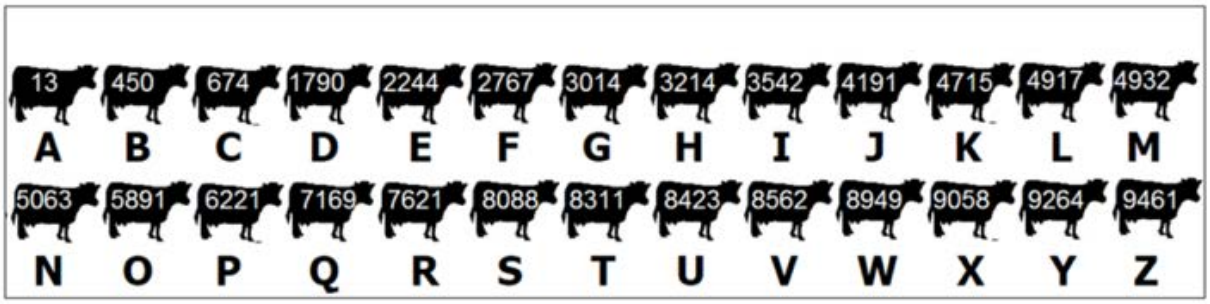
Instructions: By now you know the drill. Your partner has the list of cows *in order by weight*. Find the dragon in **5 guesses or fewer** to **save all the cows**! Record your guesses in the table below.



Guess #	Cow	Guess #	Cow	Guess #	Cow	Guess #	Cow	Guess #	Cow
1		2		3		4		5	

DRAGONS AND COWS WORKSHEET 4 — *DRACONIA DAY 2*

You have the list of cows for Draconia, and will help your partner during the second day of dragon attacks.

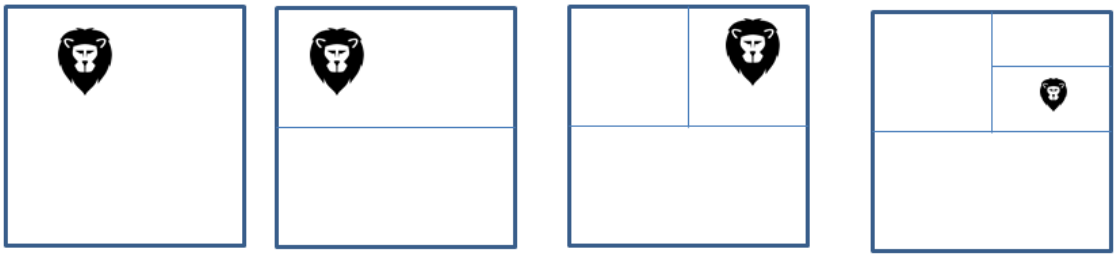


LION HUNTING DEMONSTRATION GUIDE — *WHOLE CLASS*

Draw a big square on the chalkboard/white board, and label it "the desert." Tell the students that there is a lion loose in the desert, and it's our job to build a fence around it so it doesn't hurt anyone. We don't want to catch it any other way, since it's a very cranky lion. How should we do it?

LION HUNTING ANSWER KEY — *WHOLE CLASS*

Build a fence dividing the desert into two halves. The lion is now in one of the halves. Next, build a fence across the half containing the lion, constraining the lion to one quarter of the desert. Continue building fences across the half of the remaining desert containing the lion, until the lion is cornered. (We've assumed the lion can't get out of the desert.)



WHAT'S IT ALL ABOUT? — *WHOLE CLASS*

One of the biggest search problems in the world is faced by Internet search engines, which must search billions of web pages in a fraction of a second. The data that a computer is asked to look up, such as a word, a bar code number or an author's name, is called a search key. These searches are much more complex than the binary search algorithm covered in this lesson, and computer scientists are still developing more efficient ways to address this challenge.

Computers can process information very quickly, and you might think that to find something they should just start at the beginning of their storage and keep looking until the desired information is found. This is what we did in the Raffle Tickets activity. But, this method is very slow — even for computers.

For example, suppose a supermarket has 10,000 different products on its shelves. When a bar code is scanned at a checkout, the computer must look through up to 10,000 numbers to find the product name and price. Even if it takes only one thousandth of a second to check each code, 10 seconds would be needed to go through the whole list. Imagine how long it would take to check out the groceries for a family!

A better strategy is binary search. In this method, the numbers are sorted into order. Checking the middle item of the list will identify which half the search key is in. The process is repeated until the item is found. Returning to the supermarket example, the 10,000 items can now be searched with 14 probes, which might take two hundredths of a second — hardly noticeable.

Activity 8
Cryptography: *Encrypting and Decrypting Messages*

Summary

There is a large amount of sensitive information being stored on computers and transmitted between computers today, including account passwords, trade secrets, and personal financial information. To keep this information hidden from third parties who may want access to it, cryptographic techniques must be used to encrypt it, making it difficult or impossible to actually recover the original data for anyone but the intended recipient. Because most modern cryptographic algorithms involve high-level mathematical concepts, this activity will not discuss them, but it will investigate the general ideas behind cryptography and introduce the idea of analyzing the strength of different kinds of encryption.

Getting Ready

ACTIVITY BREAKDOWN	PAGE	ADDITIONAL FILES IN .ZIP DOWNLOAD	TIME	
Introduction	114	no additional files	5 min.	50 minutes TOTAL
Worksheet 1 — *The Caesar Cipher*	116	no additional files	10 min.	
Worksheet 1 Discussion	119	Caesar Wheel PDF	5 min.	
Worksheet 2 — *Packet Villain and Surprise Party*	120	no additional files	25 min.	
What's It All About?	122	Cryptography PPT slides	5 min.	

Materials

Each student will need:

- pencil or pen
- Worksheet 1 — *The Caesar Cipher*
- Worksheet 2 — *Packet Villain and Surprise Party*
- Caesar Wheel (two on each sheet)
  - This may be used just for class discussion, or omitted completely.

materials adapted by the Colorado School of Mines with permission from Computer Science Unplugged (http://csunplugged.org)

CRYPTOGRAPHY INTRODUCTION

Cryptography is the study of encryption and decryption of messages. The principle of encoding a message is to ensure that only the intended receiver understands the message. Thus, when encoding a message, it is important to define a consistent "cipher," which is known by the recipient beforehand. A "cipher" determines how the message is encrypted.

One of the earliest known ciphers is the Caesar cipher, which Julius Caesar used to send encoded and security messages to generals in the Roman army. The Caesar cipher shifts the alphabet system by a predetermined amount so that the beginning letter of the encrypted message's alphabet is different from that of the original message's. For example, a Caesar cipher that translates the message "BAD" into "EDG" is said to have a shift of 3. This is because each letter in the original message is shifted 3 letters forward in the alphabet. This cipher is relatively easy to break, due to the limited number of combinations (25, to be exact), but it is a good representation of the basic principles of cryptography.

Cryptography is widely used in computer science. Internet traffic is often encrypted at some level and relies on consistent cipher generation and transmission in order for secure messages to be sent.

The purpose of this module is to provide students with an introduction into the world of cryptography through the Caesar cipher, and to help students understand how cryptography is used in computer science.

Terminology

» *cryptography* — the study of encryption and decryption of messages

» *encoding* — obfuscating a message

» *decoding* — figuring out the original message from the encrypted message

CRYPTOGRAPHY INTRODUCTION GUIDE — *WHOLE CLASS*

Lesson Vocabulary (You may want to write these terms on the board.)

- cryptography
- encode
- decode

Start by getting the students to think about what kinds of information might need to be kept hidden. What if all of our passwords were transmitted over the internet without any sort of encryption?

Ask for ideas about how we might protect sensitive information.

Next, explain the idea of encrypting a message (i.e., modifying it to be unrecognizable) before transmitting it to make it harder for someone who intercepts the message to actually read, unless they can figure out how to decode it. Then, transition into explaining Worksheet 1 — *The Caesar Cipher* (page 117).

The following example will give students a good idea as to how cryptography works before they start the worksheet. Tell the students you are going to encrypt a message using a cypher or "key" of size 2. Ask if they can decrypt the following message: eqorwvgt [**answer**: computer].

WORKSHEET 1 GUIDE — *THE CAESAR CIPHER*

This worksheet introduces a very simple kind of encryption that was used by Julius Caesar. The worksheet walks the students through the mechanics of the cipher, and lets them practice using it on their own. Specifically, the worksheet allows students to sharpen their skills in encoding and decoding a variety of messages using different keys.

WORKSHEET 1 — *THE CAESAR CIPHER*

Julius Caesar used a simple substitution cipher to send messages to his troops. He substituted each letter with the letter that was 3 places further along in the alphabet, so that "a" was replaced with "D," "b," with "E," and so on.

Part I. Complete the table below to show what each letter is enciphered as using this system.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
D	E	F																							

Part II. Using the Caesar Cipher, encode the name of your school. Did your partner get the same answer?

_____

Part III. Computer scientists would call 3 the "key" for this cipher. How many different keys are possible?

_____

Part IV. Decode this message, which was encoded using the Caesar cipher from the table above:

Z	K	D	W		G	R		B	R	X		J	H	W		Z	K	H	Q		B	R	X	

F	U	R	V	V		D		V	Q	R	Z	P	D	Q		Z	L	W	K		D	

							?										
Y	D	P	S	L	U	H	?		I	U	R	V	W	E	L	W	H

WORKSHEET 1 ANSWER KEY — *THE CAESAR CIPHER*

Julius Caesar used a simple substitution cipher to send messages to his troops. He substituted each letter with the letter that was 3 places further along in the alphabet, so that "a" was replaced with "D", "b" with "E" and so on.

Part I. Complete the table below to show what each letter is enciphered as using this system.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Part II. Using the Caesar Cipher, encode the name of your school. Did your partner get the same answer?

Part III. Computer scientists would call 3 the "key" for this cipher. How many different keys are possible?

25

Part IV. Decode this message, which was encoded using the Caesar cipher from the table above:

w	h	a	t		d	o		y	o	u		g	e	t		w	h	e	n		y	o	u	
Z	K	D	W		G	R		B	R	X		J	H	W		Z	K	H	Q		B	R	X	

c	r	o	s	s		a		s	n	o	w	m	a	n		w	i	t	h		a	
F	U	R	V	V		D		V	Q	R	Z	P	D	Q		Z	L	W	K		D	

v	a	m	p	i	r	e	?		f	r	o	s	t	b	i	t	e
Y	D	P	S	L	U	H	?		I	U	R	V	W	E	L	W	H

WORKSHEET 1 DISCUSSION — *WHOLE CLASS*

Discuss what happens after the letter "w" (which is encoded as a Z). Point out that the code "wraps around" to the start of the alphabet, sort of like an odometer on a car.

Discuss the number of different possible Caesar cipher keys. This is a good place to gently introduce modular arithmetic, which has many applications in computer science.

- What happens when a shift of 0 is used? What about a shift of 26?
- We can use negative numbers to represent a left shift. Can we use a positive shift to get the same effect as a shift of -1? [**answer**: 25].
- This is called modular arithmetic, where values wrap around at a certain value – the modulus. In this case, it is 26 because there are 26 letters in the alphabet, so encrypting using a given key is the same as encrypting using that key plus any multiple of 26.

(optional) You may want to show students the Caesar Wheel PDF (available in the .zip download).

How easy is it for someone who intercepts a secret message written using this cipher to work out the original message? Is there any way to make it harder to work out?

- Someone breaking the code only needs to try all 26 keys until they find one where the message makes sense.
- Changing from a simple shift to a substitution in which each letter is substituted for another changes the total keys from 26 to 26! (i.e., factorial 26), or, approximately 403 septillion (403 million billion billion).

How do we break a substitution cipher that is not just a simple shift if there are 403 septillion keys?

- Certain letters are used more than others. For example, which is used more: T or X?
- In a substitution cipher, each occurrence of a plain text letter is always mapped to the same cipher text letter.

WORKSHEET 2 GUIDE — *PACKET VILLAIN AND SURPRISE PARTY*

Divide the class into groups of 3 or 4. Explain that each group is trying to get together a surprise party without the lucky person finding out. The team will need to make up the details — who will the surprise party be for? Where? What game or activity will you do at the party? What gift will you bring? You may want to encourage students to use answers that are fairly short (e.g., 1-3 words, not entire sentences).

Packet Villain. The first part of the worksheet is just to practice encoding/decoding when the exact cipher is not specified (but limited to one of 3, to keep it fairly simple). This exercise is called Packet Villain. This small practice should not take very long, but it provides practice encoding a known message (the partner's name).

Surprise Party. Next, the teams should encode the details of the party. Each person should encode only ONE detail. After the encoding is done, have the teams swap and see if they can figure out the details. Note that this will be harder because students do not know **a)** which cipher was used, or **b)** what the answer is.

If a team is really struggling, you might ask the other team to tell which cipher they used. As teams finish the decoding, share some of the party details with the entire class.

WORKSHEET 2 — *PACKET VILLAIN AND SURPRISE PARTY*

Packet Villain

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R

Encode your name using one of the ciphers above. Ask your partner to figure out what key you used.

My name: _____ Which key/table was used? Key 5? 10? 18?

Surprise Party

You and your teammates are planning a surprise party. To ensure no one else figures out the details, you are going to encode them. Each of you will fill out **ONE** of the details below (decide in advance who is doing which part). Encode the answer using one of the ciphers above (you do not all need to use the same cipher).

When you are all done encoding, give all your papers to another team to see if they can figure out your party.

WHO (Whose surprise party is it?)

WHERE (Where are you holding the party?)

FUN (What game or activity will you do for fun?)

BRING (What gift will you bring?)

WHAT'S IT ALL ABOUT? — *WHOLE CLASS*

People rely on encryption all the time when using the internet. Some of the uses of encryption include:

- protecting credit card details or other sensitive information in online transactions
- protecting email communication from eavesdropping third parties
- verifying the authenticity of software updates to prevent installation of malicious software

Without strong cryptographic algorithms, it would be impossible to hide information from others or to verify the identity of anyone on the internet, and most of our modern internet infrastructure would not function.

A simple substitution cipher can be broken in fractions of a second by any modern computer. Modern cryptography uses the idea of computational intractability — problems that take unreasonable amounts of time to solve.

Many algorithms are based on large prime numbers:

- multiplying two large primes: 15 microseconds
- recovering the original two factors: 200,000 years

OPTIONAL: Modern decryption methods often rely on considering the context of encrypted words (i.e., what words come before and/or after) and also the frequency that different letters tend to occur in the English language. This material is covered in CryptographySlides.pdf.

Part III
Computers as Thinking Machines

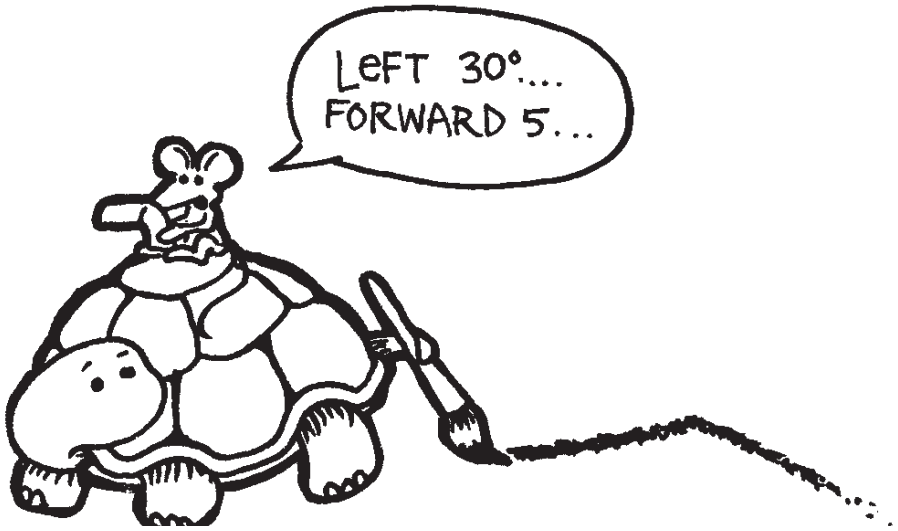
TELLING COMPUTERS WHAT TO DO *(READ ALOUD)*

Computers follow instructions — millions of instructions — every second. To tell a computer what to do, all you have to do is give it the right instructions. But that's not as easy as it sounds!

When we are given instructions we use common sense to interpret what is meant. If someone says "go through that door," they don't mean to actually smash through the door — they mean go through the door***way***, and if necessary, open the door first! Computers are different. When they are attached to mobile robots you need to be careful to take safety precautions to avoid them causing damage and danger by interpreting instructions literally — like trying to go through doors. Dealing with something that obeys instructions literally, without "thinking," takes some getting used to.

The Treasure Hunt activity in this section gives us some idea of what it is like to communicate to literal-minded machines using a fixed set of instructions.

Treasure Hunt will teach us about a "machine" that computers use to recognize words, numbers, or strings of symbols that the computer can work with. These "machines" are called finite-state automata.



Activity 9
Treasure Hunt: *Finite-State Automata*

Summary
Computer programs often need to process a sequence of symbols such as letters or words in a document, or even the text of another computer program. Computer scientists often use a finite-state automaton to do this. A finite-state automaton (FSA) follows a set of instructions to see if the computer will recognize the word or string of symbols. We will be working with something equivalent to a FSA using a wacky fruit vendor!

Getting Ready

ACTIVITY BREAKDOWN	PAGE	ADDITIONAL FILES IN .ZIP DOWNLOAD	TIME	
Fruit Vendor Exercise 1 — *Fickle Fruit*	126	Fruit Cards PDF	5 min.	50 minutes TOTAL
Introducing FSA Syntax	128	no additional files	5 min.	
Fruit Vendor Exercise 2 — *Frustrating Fruit*	130	Fruit Cards PDF	10 min.	
Fruit Vendor Exercise 3 — *Fancy Fruit*	132	Fruit Cards PDF	5 min.	
Worksheet 1 — *Robot Dog*	135	no additional files	10 min.	
Worksheet 2 — *Chores Robot*	138	no additional files	10 min.	
What's It All About?	140	no additional files	5 min.	

Materials
Each group of four students receives:

- one set of Fruit Vendor Instruction Cards (in applicable Fruit Vendor Exercises)
- Fruit Cards
  They need to be cut out, and can be laminated for future reuse. We suggest providing eight bananas and eight apples per group.
- plastic spoons and forks

Each student will need:

- pencil or pen
- Worksheet 1 — *Robot Dog*
- Worksheet 2 — *Chores Robot*

Lesson Preparation
Before diving into the lesson, it is recommended that you watch "Treasure Hunt video," which can be found here: https://www.youtube.com/watch?v=8kagtp2gWhU (length: 2 minutes and 9 seconds). Although we'll be using an alternate activity, the video provides an easy instruction to finite-state automata.

It is also recommended that you watch "CS Unplugged: FSA Fruit Vendor," which can be found here: https://www.youtube.com/watch?v=Igb4IpOXlTA&index=2&list=PL-7FSY8VA_YBo1cHdhXwrkG1EkK8nvUne (length: 3 minutes and 46 seconds).

materials adapted by the Colorado School of Mines with permission from Computer Science Unplugged (http://csunplugged.org)

FRUIT VENDORS EXERCISES GUIDE

Introduction

This activity will introduce the concept of finite-state automata using a fruit vendor who sells apples and bananas. Unlike what we would expect, the fruit vendors in these exercises do not always give you the fruit you ask for. The vendors DO follow a set of rules, however, so if students can figure out a pattern in the vendors' behaviors, they will be able to correctly order the fruit they want. This part of the lesson will include three different exercises, with whole group discussions between them.

The three exercises in this activity are organized by level of difficulty, from easiest to hardest: Fickle, Frustrating, and Fancy.

Setup

Students should be divided into groups of four. This will allow for each group to have one vendor and three buyers.

Select one student in each group to be the vendor. Different students can be the vendor for different exercises. (I.e., there can be one vendor for Fickle, a different vendor for Frustrating, and a different vendor for Fancy. Or, one student can be the vendor for all three exercises.) Solutions to all three exercises are included in this lesson plan.

The vendor should get the Fruit Vendors Instruction Cards (on pages 127, 131, or 134) for the applicable exercise, as well as 16 Fruit Cards (available in the .zip download) — eight apples and eight bananas — and a spoon. The other students *should not* be able to see the instruction card, so make sure the vendors know to keep this secret!

FRUIT VENDORS EXERCISE 1 GUIDE — *FICKLE FRUIT*

The three buyers in the fruit market will take turns trying to buy apples and bananas. While they are buying fruit, students should be trying to figure out the pattern. At this point, we have not yet introduced how to represent an FSA, so the ways students keep track of the pattern may vary.

Challenge the students to come up with a sequence of requests that gets them three apples in a row.

After the majority of the groups seem to have figured out the pattern, or the vendors are out of fruit to sell, regroup as a class, and introduce FSA.

FRUIT VENDORS EXERCISE 1 — *FICKLE FRUIT VENDOR INSTRUCTION CARD*

If you are holding:

A SPOON

If the customer asks for an **APPLE**,

give them a banana and put down the spoon.

If the customer asks for a **BANANA**,

give them an apple.

NO SPOON

If the customer asks for an **APPLE**,

give them an apple and pick up the spoon.

If the customer asks for a **BANANA**,

give them a banana.

INTRODUCING FINITE-STATE AUTOMATA SYNTAX — *WHOLE CLASS DISCUSSION*

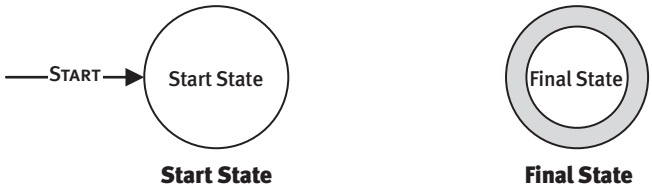
Lesson Vocabulary (You may want to write these terms on the board.)

» Finite-State Automata (FSA)
» States
» Transitions

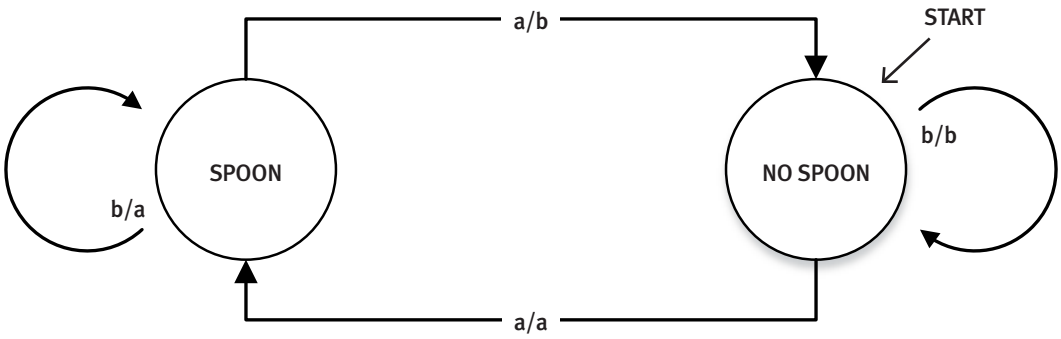
Finite-State Automata provide a visual tool for computer scientists to represent states and transitions between those states. As it turns out, FSAs are useful for modeling our fruit vendors because they follow a set pattern and set of rules.

The syntax (rules for correct format of the drawing) for an FSA is not too hard. Start with the basics, drawing them on the board:

» *States* (shown below) are represented as circles with a description of the status. The states for a light bulb might be ON or OFF. The states for a music player might be PLAYING, PAUSED, STOPPED.
  • A *start state* is a circle that has a transition arrow labeled "START" pointing to that state.
  • A *final state* is a state with two circles.
» *Transitions* are represented as arrows between circles, including an *event* (e.g., pushing a button) and a description of an *action* (e.g., music begins to play). When the event happens, the system moves to the next state. For example, pressing a Play button might cause a music player to change from STOPPED to PLAYING.



Using our newfound notation, explain to the students that we can represent the Fickle Fruit vendor using an FSA. The figure below shows the solution for the Fruit Vendors Exercise 1 — *Fickle Fruit*.



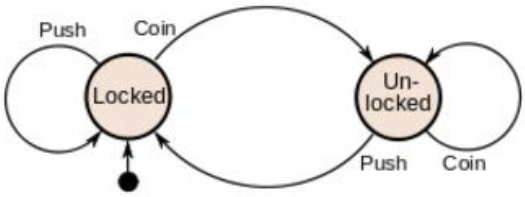
The start state for this FSA is that the vendor has no spoon. There is no final state, as play continues until students have gathered 3 apples or time expires.

To engage students in drawing this FSA:

» Ask if the vendor initially is holding a spoon. Students should say no. So draw an arrow labeled START and a circle with NO SPOON.
» Ask the students what happens if a buyer asks for a banana. Students should read the Vendor Instructions and tell you the vendor gives the buyer a banana. The vendor does not pick up a spoon. So draw the line from NO SPOON back to NO SPOON and label it as b (asked for banana) / b (gave a banana). You can spell out the words if it makes it clearer for the students.
» Now ask the students what happens if a buyer asks for an apple. Students should tell you that the buyer gets an apple, and the vendor picks up the spoon. Draw SPOON as a state, with a line from NO SPOON to SPOON labeled a/a (or apple / apple).
» Complete the FSA by asking students what happens when the vendor is holding the spoon and the buyer asks for a banana or an apple (see figure below).

To reinforce the concept, you can now relate the concept of an FSA to a real life example. Ask if students have ever gone through a turnstile. What even unlocks the turnstile? [**answer**: inserting a coin] What event allows the user to pass through? [**answer**: pushing on the bar]

Draw the diagram below on the board.

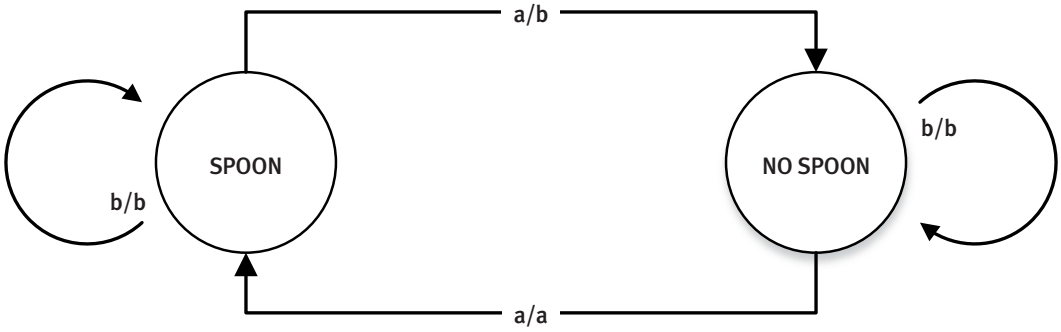


FRUIT VENDORS EXERCISE 2 GUIDE — *FRUSTRATING FRUIT*

Have students get back into their groups. Explain that they are starting a *new* exercise in which the vendor will have a different pattern for selling apples and bananas. Collect the Fruit Vendors Exercise 1 — *Fickle Fruit Vendor Instruction Card* (page 127) and distribute the Fruit Vendors Exercise 2 — *Frustrating Fruit Vendor Instruction Card* (on page 131). Have the buyers again attempt to buy apples and bananas, but this time try to write the vendor's behavior as an FSA.

Again, challenge the students to get three apples in a row. (***Note:*** *This is not possible for this exercise.*)

Regroup as a class. Discuss the exercise and attempt to draw the FSA on the board, using student input. The figure below shows the solution for the Fruit Vendors Exercise 2 — *Frustrating Fruit*.



There is only one slight difference between this FSA and the previous one: the loop transition on the "SPOON" state gives a banana instead of an apple!

FRUIT VENDORS EXERCISE 2 — *FRUSTRATING FRUIT VENDOR INSTRUCTION CARD*

If you are holding:

A SPOON

If the customer asks for an **APPLE**,

give them a banana and put down the spoon.

If the customer asks for a **BANANA**,

give them a banana.

NO SPOON

If the customer asks for an **APPLE**,

give them an apple and pick up the spoon.

If the customer asks for a **BANANA**,

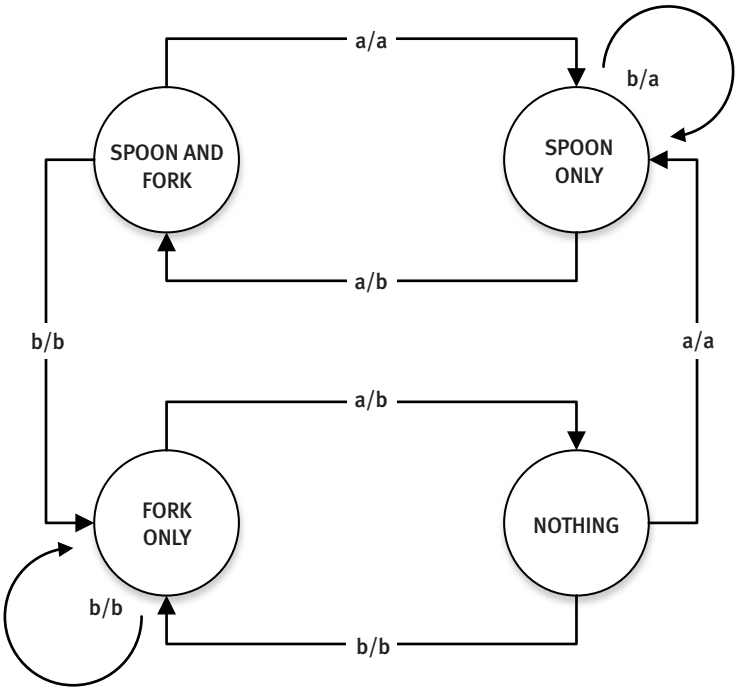
give them a banana.

FRUIT VENDORS EXERCISE 3 GUIDE — *FANCY FRUIT* (OPTIONAL EXTENSION)

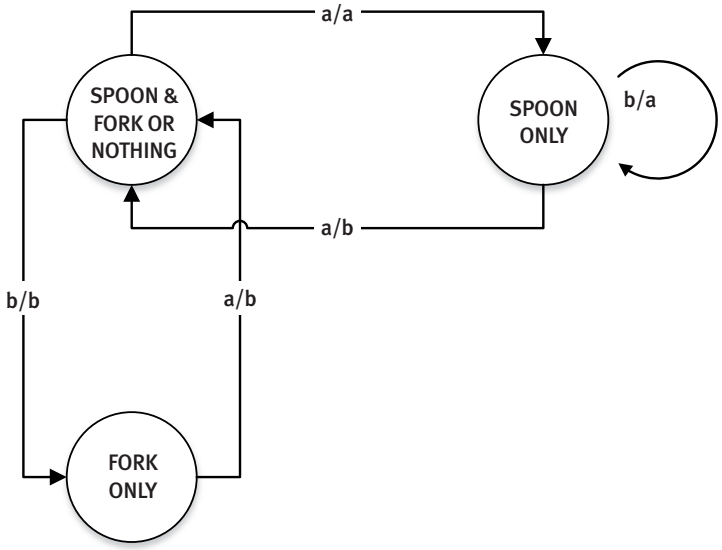
If you have enough time and the students are engaged in these exercises, you can do the Fancy Fruit activity.

Have students get back into their groups. Collect the Frustrating Fruit Vendor Instruction Card, and pass out the Fruit Vendors Exercise 3 — *Fancy Fruit Vendor Instruction Card* (on page 134), as well as a fork. Have them repeat the buying/selling process, again trying to figure out the state diagram. This activity is a challenge and therefore should be used if time permits.

The FSA for Fancy Fruit is shown in the figure below.



Optional: Notice that states "SPOON and FORK" and "NOTHING" have exactly the same transitions out of them. Namely, if you ask for a banana you are given a banana and you go to state "FORK ONLY," and if you ask for an apple you are given an apple and you go to state "SPOON ONLY." Since these two states are exactly the same, we can combine them into one called "(SPOON and FORK) or (NOTHING)." Then, the diagram looks like the figure below.



FRUIT VENDORS EXERCISE 3 — *FANCY FRUIT VENDOR INSTRUCTION CARD*

If you are holding:

SPOON AND FORK

If the customer asks for an **APPLE**,

give them an apple, and put down the fork.

If the customer asks for a **BANANA**,

give them a banana, and put down the spoon.

SPOON ONLY

If the customer asks for an **APPLE**,

give them a banana, and pick up the fork.

If the customer asks for a **BANANA**,

give them an apple.

FORK ONLY

If the customer asks for an **APPLE**,

give them a banana, and put down the fork.

If the customer asks for a **BANANA**,

give them a banana.

NOTHING

If the customer asks for an **APPLE**,

give them an apple, and pick up the spoon.

If the customer asks for a **BANANA**,

give them a banana, and pick up the fork.

WORKSHEETS 1 AND 2 INTRODUCTION — *ROBOT DOG AND CHORES ROBOT*

At this point, the students have seen finite-state automata on the board, but have not had a chance to practice on their own. The next two worksheets should help solidify their understanding.

Robot Dog Introduction

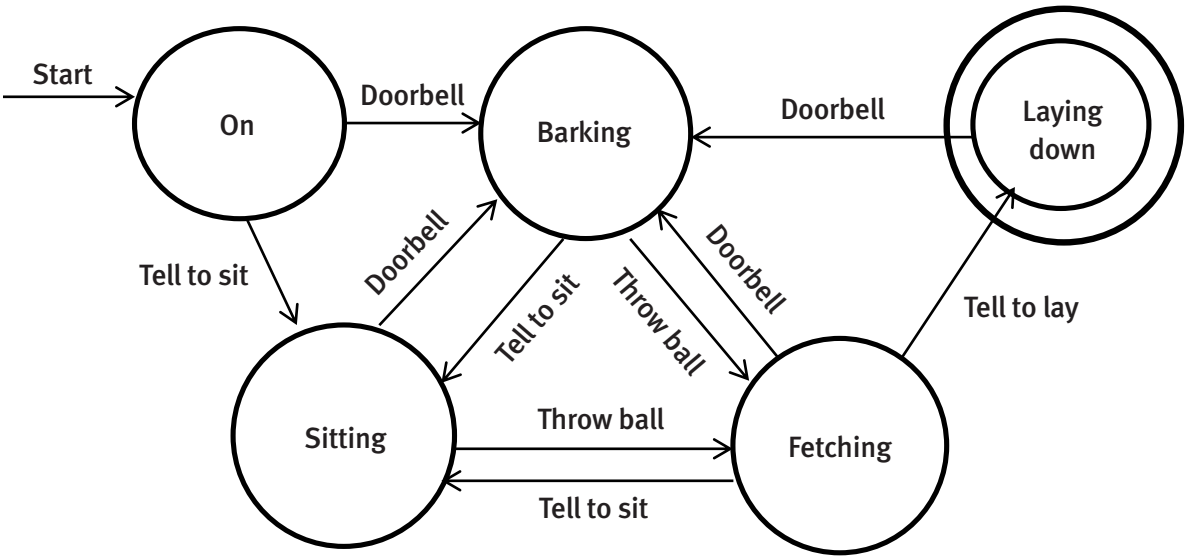
This worksheet will give students practice with recognizing the outcomes of various FSA transitional combinations.

Chores Robot Introduction

The purpose of this activity is to reinforce the understanding of the difference between states and transitions. This activity should also give students practice with constructing a finite-state automaton independently.

WORKSHEET 1 — *ROBOT DOG*

Name: _____



1. Identify the following states

   a. Start State: _____

   b. Stop State: _____

2. Identify what the dog will be doing after each set of actions, OR write **ERROR** if the set of actions is not valid.

   a. Doorbell, Tell to sit, Throw ball: _____

   b. Tell to sit, Doorbell, Tell to sit: _____

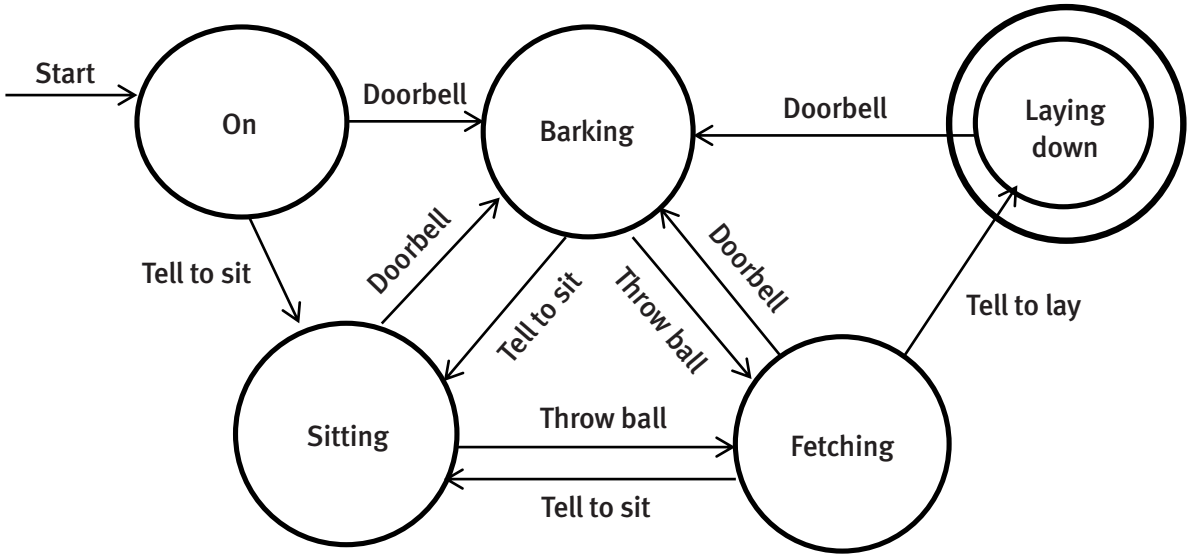
   c. Tell to sit, Throw ball, Tell to lay: _____

   d. Tell to sit, Throw ball, Throw ball: _____

   e. Doorbell, Tell to sit, Doorbell, Throw ball, Tell to sit: _____

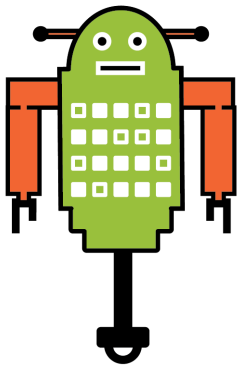
3. Circle the paths from question 2 where the dog barks.

WORKSHEET 1 ANSWER KEY — *ROBOT DOG*



1. Identify the following states.
   a. Start State: On
   b. Stop State: Laying Down
2. Identify what the dog will be doing after each set of actions OR write **ERROR** if the set of actions is not valid.
   a. Doorbell, Tell to sit, Throw ball: **Fetching**
   b. Tell to sit, Doorbell, Tell to sit: **Sitting**
   c. Tell to sit, Throw ball, Tell to Lay: **Lay Down**
   d. Tell to sit, Throw Ball, Throw Ball: **Error**
   e. Doorbell, Tell to sit, Doorbell, Throw ball, Tell to sit: **Sitting**
3. Circle the paths from question 2 where the dog barks: **a, b, e**.

WORKSHEET 2 — *CHORES ROBOT*



Name:_____

You have a robot with two buttons labeled A and B; if you hit these buttons, it will do your chores. It can only do certain chores after it has already done others. Use the instructions below to create an FSA for your robot. Remember to label all of the *transitions* *(actions)* and **states**.

1. If your robot is on **Standby**, you can *press A* and it will **make your bed.**
2. If your robot is on **Standby**, you can *press B* and it will **do the dishes.**
3. If your robot is **making your bed,** you can, *press A* to get it to **do the laundry** or *press B* to make it return to **standby.**
4. If your robot is **doing the dishes,** you can *press A* to have it **take out the trash** or *press B* to make it **do the laundry**.
5. If your robot is **doing the laundry,** *pressing A* will make it return to **standby**, and *pressing B* will have it **take out the trash.**
6. If your robot is **taking out the trash**, *pressing A* will have it **make your bed**, and *pressing B* will have it **do the laundry.**

What sequence of button pushes will get your robot to do all of your chores? Try to list two options.

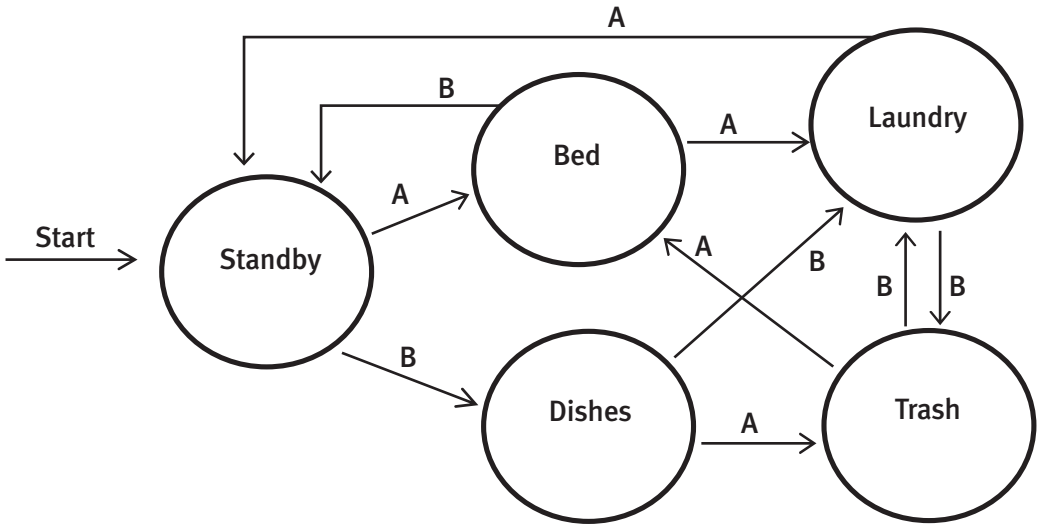
1.

2.

WORKSHEET 2 ANSWER KEY — *CHORES ROBOT*

You have a robot with 2 buttons labeled A and B, if you hit these buttons it will do your chores. It can only do certain chores after it has already done other ones. Use the instructions bellow to create a FSA for your robot, remember to label all the *transitions (actions)* and **states**.

1. If your robot is on **standby**, you can *press A* and it will **make your bed**.
2. If your robot is on **standby**, you can *press B* and it will **do the dishes**.
3. If your robot is **making your bed,** you can *press A* to get it to **do the laundry** or press B to make it return to standby.
4. If your robot is **doing the dishes,** you can *press A* to have it **take out the trash** or press B to make it **do the laundry**.
5. If your robot is **doing the laundry,** *pressing A* will make it return to **standby**, and *pressing B* will have it **take out the trash**.
6. If your robot is **taking out the trash**, *pressing A* will have it **make your bed**, *pressing B* will have it **do the laundry**.



What sequence of button pushes will get your robot to do all of your chores, try to list 2 options.

1. A A A B A
   A B B A B
2. B B B A **There are other possible solutions to this problem that are not listed.**

WHAT'S IT ALL ABOUT? — *GROUP DISCUSSION*

Finite-state automata are used in computer science to help a computer process a sequence of characters or events.

Adult Examples. A simple example is when you dial a telephone number, and you get a message that says, "Press 1 for this ... Press 2 for that ... Press 3 to talk to a human operator." Your key presses are inputs for a finite-state automaton at the other end of the line. The dialogue can be quite simple, or very complex. Sometimes you are taken around in circles because there is a peculiar loop in the finite-state automaton. If this occurs, it is an error in the design of the system — and it can be extremely frustrating for the caller!

Another example is when you get cash from a bank cash machine. The program in the machine's computer leads you through a sequence of events. Inside the program, all of the possible sequences are kept as a finite-state automaton. Every key you press takes the automaton to another state. Some of the states have instructions for the computer on them, such as "dispense $100 of cash" or "print a statement" or "eject the cash card."

Student examples. Finite-State Automata can also be applied to the concept of a vending machine. Possible transitions/actions are depositing a coin or pushing a button. States might be that more money is needed (while depositing coins) or that the item is dispensed.

A video game with a finite number of actions and levels serves as another example of Finite-State Automata. For example, in the game Mario, the character Mario can exist in multiple states. He can grow in size, shrink in size, and be able to fly or be able to throw fireballs. The various objects that Mario encounters serve as the transitions between his states. For example, if Mario eats a mushroom, he grows to his large state. If Mario gets hit by an enemy, he shrinks to his small state.

Relate to computing. Although computers are not really very good at understanding natural language, they can readily process artificial languages. One important type of artificial language is the programming language. Computers use finite-state automata to read in programs and translate them into elementary computer instructions, which can then be "executed" directly by the computer.

Activity 10
Artificial Intelligence: *The Turing Test*

Summary

Artificial Intelligence (AI) is a growing and important field in computer science. Although AI is a common buzzword, very few people stop to think about what AI actually is. This activity aims to get students thinking critically about what makes humans intelligent, and how computer scientists are designing computers to act more like we do.

Getting Ready

ACTIVITY BREAKDOWN	PAGE	ADDITIONAL FILES IN .ZIP DOWNLOAD	TIME	
Introduction	142	no additional files	10 min.	50 minutes TOTAL
Turing Test Discussion and Activity	142	Turing Test PPT slides	20 min.	
Intelligence Piece of Paper Demonstration	145	no additional files	15 min.	
What's It All About?	149	no additional files	5 min.	

Materials

Each student will need:

» one copy of the Turing Test Activity — *Responses*
» one copy of the Turing Test Activity — *Answers for Computer Actor*

Each pair of students will need:

» one laminated (or sturdy) copy of the Intelligent Piece of Paper Instruction Card (in demonstration)

materials adapted by the Colorado School of Mines with permission from Computer Science Unplugged (http://csunplugged.org)

ARTIFICIAL INTELLIGENCE INTRODUCTION — *WHOLE CLASS*

When deploying this activity, start off with a classroom discussion about artificial intelligence. Good questions to ask students include:

» What things are easy for computers to do that are hard for humans?
- math problems – adding/multiplying/dividing
- searching for an item from a list
- sorting items based on a particular attribute (numbers, letters, words, etc.)

» What tasks are easy for humans to do that are hard for computers?
- voice recognition – sarcasm
- voice recognition – the context and semantics of language
- image recognition – finding objects in images
- facial recognition – recognizing someone in a photo

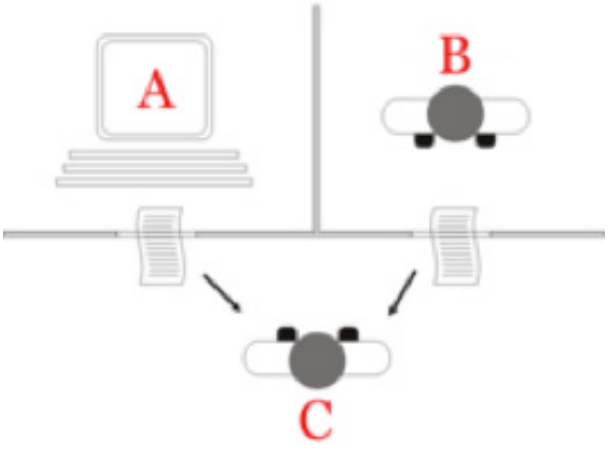
» What makes a human intelligent?

After hearing some thoughts, come up with a definition for artificial intelligence to present to the class. One that might be useful for middle school students would be:

> **Artificial Intelligence –** making a computer act more like a human so that it is able to do tasks just as well (or better) than an actual person

After we've defined AI, move on to the Turing Test. The Turing Test is named after Mathematician Alan Turing.

[from Wikipedia] The **Turing test**, developed by Alan Turing in 1950, is a test of a machine's ability to exhibit intelligent behaviour equivalent to, or indistinguishable from, that of a human. Turing proposed that a human evaluator would judge natural language conversations between a human and a machine designed to generate human-like responses. The evaluator would be aware that one of the two partners in conversation is a machine, and all participants would be separated from one another. The conversation would be limited to a text-only channel such as a computer keyboard and screen so the result would not depend on the machine's ability to render words as speech (see figure on the right). If the evaluator cannot reliably tell the machine from the human, the machine is said to have passed the test. The test does not check the ability to give correct answers to questions, only how closely answers resemble those a human would give.



TURING TEST DISCUSSION AND ACTIVITY

Use the Turing Test PowerPoint slides (available in the .zip download) to explore the idea of the Turing Test and to guide an activity. The activity will involve three students — one pretending to be a "computer" who will have a copy of questions and predefined answers (like a computer program), another as a "human" who will just answer questions as they see fit, and the go-between to ask the questions.

TURING TEST ACTIVITY — *RESPONSES*

COMPUTER ANSWER	HUMAN ANSWER

COMPUTER ANSWER	HUMAN ANSWER

COMPUTER ANSWER	HUMAN ANSWER

TURING TEST ACTIVITY — *ANSWERS FOR COMPUTER ACTOR*

QUESTION 1

1. What is the name of Bart Simpson's baby sister?
   I can't remember.
2. What do you think of Harry Potter?
   Harry Potter is an interesting wizard.
3. Are you a computer?
   Are *you* a computer?
4. What is the next number in the sequence 3, 6, 9, 12, 15?
   18.
5. What do you think of nuclear weapons?
   Nuclear weapons are very dangerous and should not be used.

QUESTION 2

1. What's the capital of Peru?
   I never really thought about it.
2. Do you like school?
   Yes, I like school.
3. Do you like dancing?
   Yes, I like dancing.
4. What day is it today?
   Give the correct day of the week.
5. What time is it?
   Give the correct time.

QUESTION 3

1. Which month is different in a leap year?
   Leap years happen every four years.
2. How many days are there in a week?
   Seven.
3. For which country is the flag a red circle on a white background?
   I don't know.
4. Do you like to read books?
   Yes, I like to read books.
5. What food do you like to eat?
   I'm not hungry, thanks.

INTELLIGENCE PIECE OF PAPER DEMONSTRATION GUIDE

This activity comes from Paul Curzon from Queen Mary University of London.

The previous discussion introduced the idea that one goal of computer scientists is to make computers seem "intelligent" – more like humans. But those activities did not explore how that might be possible. This activity encourages students to think about what makes a computer intelligent, by showing that it's the logic programmed into the computer.

The Hook: Announce that the piece of paper you are holding is more intelligent than anyone in the room (even the highly intelligent teachers there). Wax lyrical about how intelligent it is without saying why. Wave it around, keeping the written side hidden from the audience.

The Set-up: Ask the audience if they believe you, and have a show of hands, first of those who believe it is intelligent, and then those who "believe I am talking total garbage and there's no way just a piece of paper could be intelligent."

Usually most will go for the garbage option. Congratulate them on their wisdom, both for believing such a wise person as you and especially those who don't – after all, no good scientist believes claims of random people making great pronouncements, however great they are, without some evidence.

Ask them to bear with you for a while – perhaps it is intelligent, perhaps not, but ask for suggestions of what might it be about the paper that could be the basis of such an outrageous claim.

You may get suggestions such as it is something special about the ink, or that it is laminated. If things like the former, praise them for an interesting idea but ask how exactly that might make it intelligent. If the latter, explain that the plastic covering is not the special thing – it is just there to protect the paper, as it does not like you touching it all the time. Another common suggestion is that there is a computer embedded in it. This is an opportunity to bring out a musical greeting card that plays some irritating music (and is intelligent enough to know to do it on your birthday) and explain it works via an embedded chip. You may want to mention that such a chip is as complex as the embedded computer used to put Neil Armstrong on the moon). Note that your paper could be intelligent in that way, but you it isn't.

Another suggestion will be that it is what is written on the card. Ask what might be written that would make paper intelligent. Great equations? Wonderful poetry? Exciting facts? Suggest examples, and ask whether the audience thinks that would be intelligent. If not, we need to look for something more. Talk about the fact that knowledge isn't the same as intelligence – and that they surely don't just try to memorize things for exams but instead try to understand, which isn't the same. Agree that writing such things wouldn't be enough for it to be intelligent.

Point out that to convince us that it is intelligent it must be able to do something to show that intelligence. What can the paper do? Well, it has ***never lost a game of tic-tac-toe*** (and it plays regularly against humans). Remind them that the game should end in a draw if both players play perfectly. You cannot force a win. Despite that, the paper has won about half the games it has played against humans like themselves, and drawn the rest. It is a perfect intelligence. Humans aren't.

Ask them if they believe you or whether they want to see some evidence. To show the evidence you will need two volunteers.

Draw a tic-tac-toe board on the white board/flip chart. Give each volunteer a pen. Explain that to see how intelligent the paper is, you will need to play a game of tic-tac-toe. It will not be a battle between two humans, but rather between paperkind and humankind. The paper is 'peripherally challenged' – you didn't bring it a robotic arm or camera system (computer peripherals), so it needs a servant to do its bidding. You might want to note that just because someone is paralyzed doesn't mean they aren't more intelligent than you.

One person will therefore play for the piece of paper. Their job is to just do what they are told by the paper. (See the Intelligence Piece of Paper — *Instruction Card* on page 147.) They must switch off their highly intelligent brain and do exactly as they are commanded: we don't want to know how well *they* play the game, just the paper. They should just read out loud the paper's instructions (so everyone can tell it is the paper playing, not them) and do as it tells them.

The other player is there to represent the best of humanity. It may be best not to pick someone who is really keen and appearing to think they would never lose to reduce the chances of it being a drawn game (though often such volunteers still lose). Their job is to use all of their intelligence to play as well as they can. Because the paper is so intelligent, to make it more fair, say they can get help from the audience if they need it. Tell the audience to shout out if they think a mistake is being made or know the move to make.

Now get the person playing for the paper to read its instruction (about going first – comment that it is quite clever of it to want to go first). If someone complains about it being unfair it going first, point out that the game should just end in a draw. Going second isn't a reason to lose.

The paper's servant should then read out the first move and make the move – playing in a corner. Then turn it over to the human. There may be lots of shouts about different places to go. If the person is unsure, encourage them to go where they think is the best of the options shouted. Continue like this, making sure the reader does read out and follow exactly the instruction and helping them understand where they are being told to go if need be. For example, opposite corner means the diagonally opposite corner – this can later lead to a discussion of why special programming languages are needed – to be precise about what is needed.

Sometimes after the second or third move people in the audience will declare the game lost. Point out that humanity does often resign at this point. Remind them it is only a piece of paper though. It might just have been lucky this far, so it might still mess up and not see what they can see.

If the paper has forked them, then often the player will then jokingly cheat, such as drawing two os. Point out that humanity often resorts to cheating at this point, and make them play properly, reminding them it is only paper and it may still go wrong.

Either the paper will win or it will be a draw (if the human realizes they need to go on the side and not corners). If the latter, then remind them that was predicted but that even so, it was still clever of it to not let the human win. Announce that once more it has kept its unbeaten run. Say you'll accept that perhaps it's not more intelligent than the humans, but it has shown itself to be their equal. If the paper wins, announce that yet again the paper has shown itself more intelligent than humanity and that you have shown them the evidence they asked for. Either way, ask for applause both for of the volunteers as they return to their seats, and the paper for its stunning performance.

INTELLIGENCE PIECE OF PAPER — *INSTRUCTION CARD*

I am a highly intelligent piece of paper. Let's play tic-tac-toe.
I will play X, and I am going first.

Move 1

Place an X in any corner.

Wait for the other player to make their move.

Move 2

If the other player placed an O in the opposite diagonal corner,
    then place an X in a free corner.
Otherwise, place an X in the opposite diagonal corner.

Wait for the other player to make their move.

Move 3

Look at the entire board (every row, every column, and every diagonal).
If a line has two X's and a space,
    then place an X in the space to win the game.
Otherwise, if a line has two O's and a space,
    then place an X in the space to prevent the other player from winning.
Otherwise, place an X in a free corner.

Wait for the other player to make their move.

Move 4

Look at the entire board (every row, every column, and every diagonal).
If a line has two X's and a space,
    then place an X in the space to win the game.
Otherwise, if a line has two O's and a space,
    then place an X in the space to prevent the other player from winning.
Otherwise, place an X in a free corner.

Wait for the other player to make their move.

Move 5

Place an X in the last free space.

INTELLIGENCE PIECE OF PAPER DEMONSTRATION DISCUSSION — *WHOLE CLASS*

Point out that they asked for evidence and you have given it. Ask again for a show of hands as to who now believes the paper is intelligent and who believes you are talking garbage and that paper can't be intelligent. Usually everyone now is sure it isn't intelligent, despite the evidence of its abilities.

Point out that it did show intelligent behavior, so there is intelligence somewhere. Where is it? Someone will almost certainly say it is in the person who wrote the instructions. Ask whether everyone agrees that this is where the intelligence is, and get a show of hands.

Now explain that on the paper is essentially a computer program: instructions to be blindly followed. Tell them that for everything they have ever seen a computer do, it was just blindly following instructions in the same way. Point out that if they are saying the paper, just by being only rules, isn't intelligent, then they are saying no computer could ever be intelligent.

These instructions were written in a language so that a human can follow them. If they were to be written for a computer, they would be written in a programming language – a language a computer can understand and thus follow. Point out that if they think that it is the writer of the instructions who is the creative, intelligent one, then they are saying that computer programmers are intelligent and creative (which is true). It is computer programmers who have written all of the instructions that all those computers are following.

You can then let students group into pairs and play tic-tac-toe with an intelligent piece of paper until they are convinced it is correct.

WHAT'S IT ALL ABOUT? — *WHOLE CLASS*

This activity involved the class trying to judge who is human and who is a computer. Can they think of any examples where a computer might want to be the judge? A good example of a "reverse Turing Test" is CAPTCHAs – the images humans have to read to reset their passwords or perform other tasks on a computer.

Applications of artificial intelligence are all around us, and they are only going to become more common. Google's self-driving car is another great example. How do we teach a computer to drive like a human and be able to process information from all around the car quickly? These are problems that take many computer scientists to work toward a solution.

Activity 11
Computer Vision: *Edge Detection*

Summary

Computers today are being used to accomplish tasks that require using one or more of the five senses. Vision — seeing objects and identifying them — is something that humans can do relatively easily, but computers cannot. One way we help teach computers to "see" is to identify where edges are in a picture. From there, the computer can find shapes and compare them against a known database of shapes that a human has entered.

Getting Ready

ACTIVITY BREAKDOWN	PAGE	ADDITIONAL FILES IN .ZIP DOWNLOAD	TIME	
Introduction	151	Computer Vision PPT slides	10 min.	50 minutes TOTAL
Worksheet 1 — *Edge Detection (Star)*	152	no additional files	10 min.	
Worksheet 2 — *Edge Detection (Cat)*	152	Edge Detection PPT slides	10 min.	
Worksheet 3 — *Kangaroo Shape Comparison*	152	no additional files	10 min.	
What's It All About?	156	no additional files	10 min.	

Each student will need:

- pencil or pen
- Worksheet 1 — *Edge Detection (Star)*
- Worksheet 2 — *Edge Detection (Cat)*
- Worksheet 3 — *Kangaroo Shape Comparison*

materials adapted by the Colorado School of Mines with permission from Computer Science Unplugged (http://csunplugged.org)

COMPUTER VISION INTRODUCTION — *WHOLE CLASS*

Computer Vision is an exciting field of computer science that is still in its infancy. The act of seeing objects, and identifying objects, is something humans can do very well and without much thought. The same task, however, is a nontrivial problem for computers.

Have you ever wondered how your cell phone camera can detect faces in a picture automatically, and focus on them? Or how Google's self-driving car "sees" the road? Computers don't have eyes, but with clever thinking, humans have taught computers to use camera images to sense their surroundings.

One way computers see what is around them is by using edge detection to identify shapes and then objects. This is what will be explored in today's activity.

Discussion

Begin the class by getting students to think about what it means to see objects. How do you identify a spoon from a fork? When you see a person with a tennis racket, do you think they are hitting a lemon or a tennis ball? Why?

Humans are very good at identifying objects through sight. We use context — pieces of the surrounding environment — to fully understand and accurately describe what we are looking at.

Once students reach a consensus that seeing is harder than it seems, talk about what problems require vision to solve. Some examples include:

- recognizing faces in a photo
- recognizing license plates on a toll road
- counting objects

What are some activities humans do that require sight?

- driving – We have to look for other cars, people, and bicycles.
- sports – hand/eye coordination
- walking – We can see rocks, inclines, and stairs.
- reading books (the paper variety, not on your Kindle/Nook)

After discussing what you can do *with* eyesight, ask students how hard it would be to do the same activities if they were blind. Is it easier? Harder?

Use the accompanying Computer Vision PowerPoint slides (available in the .zip download) to show the class real-world examples of computer vision in action. Notes are attached to each slide explaining what the computer would see, as well as additional discussion questions.

WORKSHEETS 1, 2, AND 3 GUIDE — *EDGE DETECTION AND COMPUTER VISION*

If we are trying to program a computer to identify objects in an image, it can be helpful to find the edges in an image. From these edges, we can pick out larger features such as shapes.

Computers represent images as grids of pixels. Ask the students if they know/remember what a pixel (picture element) is. Each pixel has a single color, and we can use numbers to represent the color of a pixel. One of the simplest representations of an image is grayscale, where each pixel can only be a shade of gray. A typical range of numbers is 0 to 255, where 0 is black, 255 is white, and any number in between is a shade of gray.

Given a grayscale image, one simple way to find edges is to look at two neighboring pixels and take the difference between their values. If it's big, this means the colors are very different, so it's an edge.

Worksheet 1 — *Edge Detection (Star)*

Distribute Worksheet 1 — *Edge Detection* to the students. DO NOT TELL THEM WHAT SHAPE IS IN THE IMAGE. That is part of the fun. Give them about 5-10 minutes to complete the activity. Let them try to identify what shapes they can see, after they detect the edges. This technique is similar to what computers do. Before you detect the edges, it is very difficult to see the image. But after, you have a clear idea of the dominant shape in the image.

Worksheet 2 — *Edge Detection (Cat)*

Next, distribute Worksheet 2 — *Edge Detection*. Tell the students this one is a bit harder, and they may need to guess what the image is. Give the students 5-10 minutes to complete this activity. How many students figured out what the image is? [**answer**: the Edge Detection PPT slides (available in the .zip download) show the transformation from normal image to edge detection.]

Worksheet 3 — Kangaroo Shape Comparison

Another technique computers use to identify objects is to compare them to known shapes. Distribute Worksheet 3 (kangaroo) to the students. Give them about 5-10 minutes to complete the activity. Discuss what shapes they used to recognize the kangaroo. Did they use many or few shapes? What are some reasons that a computer might want to be detailed *(more accurately identify objects on the first try)*? What are some reasons that a computer might want to be general *(faster to calculate – answer is quick)*?

Additional Resource

The CS Field Guide has a great link for edge detection. The Field Guide can be found here: http://www.csfieldguide.org.nz/releases/1.9.9/teacher/ComputerVision.html and the link to the edge detection application can be found here: http://inspirit.github.io/jsfeat/sample_canny_edge.html (you will need a webcam).

WORKSHEET 1 — *EDGE DETECTION*

One simple way to find edges is to look at two neighboring pixels and take the difference between their values. If it's big, this means the colors are very different, so it's an edge.

Try it yourself! The grid below is filled with numbers that represent a grayscale image. See if you can detect edges the way a computer would do it. **If the values of two neighboring squares on the grid differ by more than 50, draw a thick line between them.**

21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21
21	21	21	21	21	21	21	103	103	21	21	21	21	21	21	21
21	21	21	21	21	21	21	99	103	21	21	21	21	21	21	21
21	21	21	21	21	21	21	91	103	21	21	21	21	21	21	21
21	21	21	21	21	21	21	103	99	21	21	21	21	21	21	21
21	21	21	21	21	21	91	99	103	99	21	21	21	21	21	21
21	72	93	101	103	105	99	103	91	99	104	102	101	91	103	21
21	21	99	99	91	103	99	99	103	99	99	91	99	88	21	21
21	21	21	99	99	91	91	99	103	99	91	99	103	21	21	21
21	21	21	21	99	99	99	99	99	99	99	103	21	21	21	21
21	21	21	21	21	99	103	103	91	103	103	21	21	21	21	21
21	21	21	21	21	103	91	99	99	103	91	21	21	21	21	21
21	21	21	21	103	91	99	99	103	99	103	99	21	21	21	21
21	21	21	21	91	103	99	21	21	99	91	99	21	21	21	21
21	21	21	103	103	21	21	21	21	21	21	103	99	21	21	21
21	21	21	103	3	21	21	21	21	21	21	7	103	21	21	21

WORKSHEET 2 — *EDGE DETECTION*

Given a grayscale image, one simple way to find edges is to look at two neighboring pixels and take the difference between their values. If it's big, this means the colors are very different, so it's an edge.

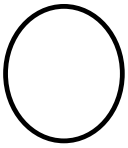
Try it yourself! The grid below is filled with numbers that represent a grayscale image. See if you can detect edges the way a computer would do it. **If the values of two neighboring squares on the grid differ by more than 40, draw a thick line between them.**

255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	80	80	255	255	255	255	255	255	255	255	255	255	80	80	255
255	90	90	80	255	255	255	255	255	255	255	255	80	85	85	255
255	90	90	80	255	255	255	255	255	255	255	80	80	85	85	255
255	90	90	90	80	255	255	255	255	255	255	80	85	85	80	255
255	90	90	90	80	255	255	255	255	255	80	80	85	85	80	255
255	90	90	90	80	80	20	80	80	20	80	80	85	85	90	255
255	255	90	90	80	80	20	80	80	20	80	80	85	85	255	255
255	255	90	90	80	80	85	80	80	85	80	80	85	90	255	255
255	80	90	20	20	20	90	85	85	90	20	20	20	80	90	255
255	80	80	62	0	62	90	85	85	90	62	0	62	85	90	255
255	90	85	62	0	62	62	85	85	80	62	0	62	85	90	255
255	90	85	85	62	85	85	85	85	80	85	62	85	85	90	255
255	255	85	85	90	90	10	10	10	10	85	85	85	85	90	255
255	255	85	85	85	85	10	40	40	10	85	85	85	85	255	255
255	255	85	85	85	85	85	10	10	85	85	85	85	85	255	255

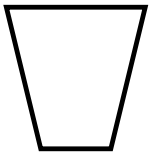
WORKSHEET 3 — *KANGAROO SHAPE COMPARISON*

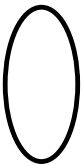
SHAPE BANK

Rectangle 

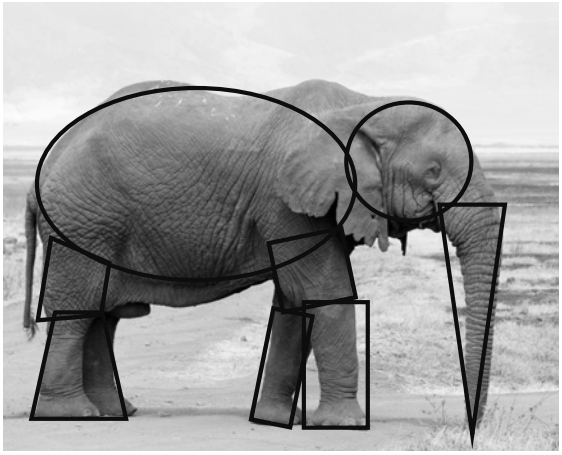
Circle 

Triangle 

Trapezoid 

Oval 

Computers look at things by breaking them down into shapes. Look at the elephant to see how a computer might break it down into basic shapes:



Now try to do the same with the image below:

WHAT'S IT ALL ABOUT? — *WHOLE CLASS*

You can regroup as a class if you have time and discuss links to other fields of computer science:

» Vision is something that is hard for computers. Other hard concepts include the other senses (sounds – talking/listening, for example). If a computer can see and hear well, does it make the computer intelligent? (See *Artificial Intelligence* section, page 141.)

» Computers don't "see" colors and shapes. They see numbers. Do you know what numbers the computer reads? (See *Binary Numbers* section, page 7.)

Computer vision is used in applications that we can see all around our daily lives. Have you ever driven on a toll road without stopping at a toll booth? Toll roads often use high-speed cameras that take a picture of your license plate, "read" the numbers and letters, and pull the address the license plate is registered with in order to process an invoice for the driver.

Social networks such as Facebook use computer vision to automatically recognize your face and the faces of your friends in photos uploaded to their website. Tagging pictures is boring, so if a computer can do it for you, you have more time to do other things.

It's important to remember that humans are good at doing hard things, and computers are good at doing easy things. Seeing objects is something that requires a great deal of contextual information that computers have trouble doing. Computer scientists are working every day to come up with better algorithms for recognizing objects in pictures, so that computers can "see" better!