COMPUTER SCIENCE
Unplugged

An enrichment and extension programme for primary-aged children

Created by
Tim Bell, Ian H. Witten and Mike Fellows

Adapted for classroom use by
Robyn Adams and Jane McKenzie

Illustrated by Matt Powell
Parents’ Edition by Andrea Arpaci-Dusseau
Introduction

Computers are everywhere. We all need to learn how to use them, and many of us use them every day. But how do they work? How do they think? And how can people make them go faster and better? Computer Science is a fascinating subject that explores these very questions. The easy and fun activities in this book, designed for children of a range of ages, introduce you to some of the building blocks of how computers work—without the children using a computer at all!

Many of the activities are mathematically based, e.g. exploring binary numbers, mapping and graphs, patterns and sorting problems, and cryptography. Others link in well with the technology curriculum, and the knowledge and understanding of how computers work. The children are actively involved in communication, problem solving, creativity, and thinking skills in a meaningful context.

Introduction to Parents’ Edition

The parents’ edition that you are using differs from other editions of Computer Science Unplugged in a number of ways.

Each activity is completely scripted. Even if you know nothing about computer science or computers, you will be able to guide your child through these activities. Much of the text in previous editions explaining “What it's all about” has been incorporated into the script of each activity.

The activities are structured around just you, the parent, and your child. The activities that previous required multiple children to participate have been redesigned so that all can be done with just two people.

Each activity requires a minimal amount of preparation and materials. Whenever you have the time, the activities are ready for you. In some cases, you do need to cut out cards from the handouts, but that is it. If you have a balance scale and weights, the sorting activity is a lot more fun; but this activity can be performed without special equipment.

Each activity is structured to require about 15 -20 minutes. Some of the original longer activities have been divided into smaller units.

All solutions are provided at the end of the book. Older children can feel free to read along with the parent without any worry that they will see some of the answers to the questions. Each question in the activity is clearly marked with a letter that corresponds to the answer in the solutions.

The activities are targeted to children between about ages 6 and 9, or grades 1-3. Many of the activities now include more introductory material with easier examples, in addition to the final activities. Some of the original material for “experts” has been removed.
How to use the Parent’s Edition

Each activity is preceded with a page giving an overview of the activity. This page describes the skills the activity builds upon, explains any preparation of handouts or other materials that are needed for the activity itself, and gives the list of related worksheet activities that the child can do (mostly on their own) after the activity.

Each activity itself is completely scripted. You can just read the text out loud to the child. To keep the activity interactive, the text contains many questions.

a) Each question is written in a format like this. Solutions for each question are provided at the end of the book. We recommend checking the solution to each question as you go.

The text that is in italics like this describes actions that the parent should take and is not meant to be read out loud.

Pictures and text that should be shown to the child are placed in a box like this.

Formal terminology is written like this. When handouts and worksheets are labelled a, b, and so on, the pages are related and need to be used together.
# Table of Contents

- Activity 1: Binary Numbers ................................................................. 1
- Activity 2: Representing Bits ............................................................... 7
- Activity 3: Representing Letters .......................................................... 11
- Activity 4: Representing Pictures ......................................................... 13
- Activity 5: Compressing Text ............................................................... 17
- Activity 6: Detecting Errors ................................................................. 20
- Activity 7: Detecting Harder Errors ...................................................... 24
- Activity 8: Information Theory .............................................................. 28
- Activity 9: Searching ........................................................................... 32
- Activity 10: Sorting ............................................................................ 33
- Activity 11: Faster Sorting ................................................................... 37
- Activity 12: Sorting Networks ............................................................... 41
- Activity 13: Muddy City ...................................................................... 44
- Activity 14: Routing and Deadlock ....................................................... 49
- Activity 15: Finite-State Automata ....................................................... 53
- Activity 16: FSA Representation .......................................................... 56
- Activity 17: Programming Languages .................................................. 59
Activity 1: Binary Numbers

Summary
Data in computers is stored and transmitted as a series of zeros and ones. How can we represent all numbers using just these two symbols?

Skills
✓ Counting
✓ Matching
✓ Sequencing

Preparation
✓ You will need to cut out the set of cards on Activity 1 Handout 1

Worksheet Activities
✓ Activity 1 Worksheet 1: Can be completed independently by the student
Activity 1: Binary Numbers

We’ll be using this book to learn about how computers work.

The word computer comes from the Latin computare, which means to calculate or add together, but computers today are more than just giant calculators.

a) What kinds of things do you like to use a computer for?

To do useful things for you, computers must do two types of things internally. First, computers store data. Data is the raw material (the numbers) that computers work with. A computer converts its data into information (like words, numbers and pictures) that you and I can understand.

Second, computers operate on the data by executing a list of instructions. These instructions enable them to sort, find and send information. An algorithm is a set of instructions for completing a task. Algorithms are how we get computers to solve problems.

We are going to begin by learning about data. Eventually, we will learn about binary numbers, how computers store words and pictures, how to store lots of data in a small amount of space, how to prevent errors from happening, and how to measure the amount of information we are storing.

Today we are going to learn about binary numbers. Computers store all information and numbers using only two values: zero and one. So, we are going to learn how to count to any number using only zero and one!

Let us start by laying out the five cards in the following order:

![Image of cards]

b) What do you notice about the number of dots on the cards?

c) How many dots would the next card have if we added another card on to the left?

d) How many dots would the next card have?

e) What rule are you using to create the next card?

We can use these cards to make numbers by turning some of them face down and adding up the dots that are showing. For example, we can make the number 5 by turning over the 1-dot and the 4-dot cards.
f) Can you make the number 1?
g) Can you make the number 6?
h) Can you make the number 15?
i) Can you make 21?
j) Can you make 30?

(Repeat with different numbers between 0 and 31 until student is no longer interested in practicing more.)

k) Is there more than one way to form any number? (For example, are there two ways to make the number 5?)

l) What is the biggest number you can make with these five cards?
m) What is the smallest number?

n) Is there any number you can’t make between the smallest and biggest numbers?

Now we are going to try counting from zero up to 16.

o) Can you show me how to create all the numbers between zero and 16?

Now we are going to see how binary numbers are formed; a binary number is a number represented with only zeros and ones. Sometimes binary is called “base-two” because there are only two possible values for each digit.

Each of the cards we have used represents a single ‘bit’ (‘bit’ is short for ‘binary digit’).

p) How many bits do we have with these cards?

When a card is face up, we will represent that bit position with a 1. When a card is face down, we will represent that bit position with a 0.

Let us look at how we would create the binary number 01001, which is the decimal number 9, with the cards.
q) Do you see why 01001 is equal to the number 9?

r) How would you use the cards to make the binary number 01101?

Decimal numbers are the types of numbers that humans usually use; decimal digits can have the values 0 to 9. Since each digit can have 10 different values, decimal is sometimes called base-ten.

<table>
<thead>
<tr>
<th>Binary</th>
<th>0-1</th>
<th>Base-Two</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>0-9</td>
<td>Base-Ten</td>
</tr>
</tbody>
</table>

s) What decimal number is 01101?

t) What is the binary number 00110 in decimal?
u) What is the binary number 01110 in decimal?
v) What is the binary number 10001 in decimal?

(Try a few more until they understand the concept.)

(The student could complete Worksheet 1 now. More advanced students may continue to the next questions which explore properties of binary numbers.)

We are now going to learn about two interesting properties of binary numbers. If you look at the sequence of cards carefully, you can find a very interesting relationship:

1, 2, 4, 8, 16, ...

1 + 2 = ?
1 + 2 + 4 = ?
1 + 2 + 4 + 8 = ?
1 + 2 + 4 + 16 + 16 = ?

w) Try adding the first two cards, 1+2. What does it come to?
x) Now try adding the first three cards, 1 + 2 + 4?
y) What about 1 + 2 + 4 + 8?

z) And finally, what if you add all of the cards together 1+2+4+8+16?

aa) What relationship do you notice between the sum of all of the previous cards and the next card?

Another interesting property of binary numbers is what happens when a zero is put on the right hand side of the number. If we are working in base 10 (decimal), when you put a zero on the right hand side of the number, it is multiplied by 10. For example, 9 becomes 90, 30 becomes 300.

9 → 90
30 → 300
But what happens when you put a 0 on the right of a binary number? Try this:

<table>
<thead>
<tr>
<th>Binary</th>
<th>After Adding 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>010</td>
<td>0100</td>
</tr>
<tr>
<td>101</td>
<td>1010</td>
</tr>
<tr>
<td>110</td>
<td>1100</td>
</tr>
</tbody>
</table>

bb) What number does 101 change to when you put a zero on the right hand side?
cc) What number does 010 change to?
dd) What number does 110 change to?

ee) What is the rule when you put a 0 after a binary number?

(Hint: If the student doesn’t see, then have them try more examples.)
Activity 2: Representing Bits

Summary
A bit can be represented by anything that has two different states. This lesson introduces how bits are stored within a computer and how bits can be sent over a modem or network.

Skills
✓ Counting
✓ Matching
✓ Sequencing

Preparation
✓ The cut out cards from Activity 1 Handout 1 may still be useful

Worksheet Activities
✓ Activity 1 Worksheet 2: Can be completed independently by the student
Activity 2: Representing Bits

In our last lesson, we learned about how binary numbers are formed with only two values: zeros and ones. We saw that we could make any number using only zeros and ones. Today we are going to learn about different ways that binary numbers can be stored inside of a computer.

A single bit represents just two values: zero or one. A bit can be represented with anything that has two different states, or values. For example, you could use a light switch to represent a bit if the light is on, it represents the value 1; if the light is off, it represents the value 0. Or, you could use candles: if a candle is lit it represents 1 and if it unlit it represents 0. Or, you could use water glasses: if the glass is filled, it represents 1, and if it is empty it represents 0.

a) Can you think of anything else that could be used as a bit?

We can also use our fingers to represent bits. If each finger represents one bit, you can count much higher than ten. If you use the binary system and let each finger on one hand represent one of the cards with dots, then you can count from 0 to 31. That’s 32 numbers. Don’t forget that zero is a number too!

b) Try counting in order from 0 to 15 using four fingers. If a finger is up it is a one, and if it is down it is a zero.

If you use both hands, with 10 fingers, you can actually count from 0 to 1023. That’s 1024 numbers! If you had really bendy toes, then you could count even higher, up past 1 million!
Computers today use the binary system to represent information. Bits can be stored in a computer’s memory with a transistor that is switched on or off, or a capacitor that is charged or discharged.

These are the symbols used to represent transistors and capacitors that are off or on.

![Transistor and Capacitor Symbols](image)

Bits are also stored on magnetic disks (floppy disks and hard disks) and tapes. Here, bits are represented by the direction of a magnetic field on a coated surface, either North-South or South-North.

c) If North-South represents 0 and South-North represents 1, then what binary number is this disk recording?

![Disk Recording Image](image)

Bits are also stored on Audio CDs, CD-ROMs and DVDs optically—the part of the surface corresponding to a bit either does or does not reflect light.

d) If absorbing light corresponds to a 0, and reflecting light corresponds to a 1, what binary number is this optical disk recording?

![Optical Disk Recording Image](image)

Since one bit on its own can’t represent much, bits are usually grouped together in groups of eight, which are called a byte. A byte can represent numbers from 0 to 255.

**8 bits = 1 byte**
Bits can also be sent over a network or modem. Computers and fax machines send bits over a modem to send messages to each other. The only difference is that they use beeps. A high-pitched beep is used for a one and a low-pitched beep is used for a zero. These beeps go very fast—so fast, in fact, that all we can hear is a horrible continuous screeching sound.

Let us try sending a binary number, pretending we are a modem.

e) Can you send the binary number 0111? (Start by sending the most-significant, or left-most bit.)

f) The binary number 0101?

g) The decimal number 2?

h) The decimal number 9?
Activity 3: Representing Letters

Summary
Letters (and words) are encoded as numbers inside of the computer. How can we encode letters as binary numbers and decode binary numbers into messages?

Skills
- Counting
- Matching
- Sequencing

Preparation
- None

Worksheet Activities
- Activity 3 Worksheet 1: Can be completed by the student independently
- Activity 3 Worksheets 2a and 2b: After the student has completed 2a, the parent is expected to complete 2b
Activity 3: Representing Letters

In our previous lessons, we learned about binary numbers and how a computer can store bits. Today we will learn how computers can use bits to form letters and words.

You already know how a computer can represent any number. Well, letters can be represented with numbers using a simple code. For example, the computer can represent the letter "a" with the number "1", the letter "b" with the number "2", and so on. So, a computer can also store any letter and any message.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>o</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>u</td>
<td>v</td>
<td>w</td>
<td>x</td>
<td>y</td>
<td>z</td>
<td></td>
</tr>
</tbody>
</table>

We use the term "encode" when we change a message that humans can read into a message that the computer can use (like binary digits). We use the term "decode" when we change the binary digits back into a human readable message.

a) Using this simple decimal code, how would the computer encode the letter d?

b) The letter y?

c) What letter would the number 19 represent?

d) The number 5?

e) How would the computer encode the word “hello”?

f) If you decode “1 14 14 1”, what name would you get?

g) How would the computer encode your name in decimal numbers?

This simple alphabet code can be represented using just five bits. However a computer uses a slightly more complicated code since it must know whether letters are capitals or not, and also recognise digits, punctuation and special symbols such as $ or ~. Most computers today use a representation called ASCII (American Standard Code for Information Interchange), which is based on using seven bits per character. Some non-English speaking countries have to use longer codes.

(The student is now ready to complete the two worksheets.)
Activity 4: Representing Pictures

Summary
Computers store drawings, photographs and other pictures using only numbers. The following activity demonstrates how they can do this.

Skills
✓ Counting
✓ Graphing

Preparation
✓ None

Worksheet Activities
✓ Activity 4 Worksheet 1: Can be completed by the student independently
✓ Activity 4 Worksheet 2: After the student has completed the first half, the parent is expected to complete the second half
✓ Activity 4 Worksheet 3 (Optional): After the student has completed the first half, the parent is expected to complete the second half
Activity 4: Representing Pictures

In our previous lesson, we learned how a computer can store words and messages. Today we will learn how a computer can store, send, and compress pictures.

a) Can you think of examples where a computer needs to store or send pictures?

We’ll begin by learning how a computer stores pictures. Computer screens are divided up into a grid of small dots called **pixels** (picture elements). In a black and white picture, each pixel is either black or white. The letter “a” has been magnified below to show the pixels. When a computer stores a picture, all that it needs to store is which dots are black and which are white.

The diagram below shows us how a picture of the letter “a” can be encoded in a very simple way. With this approach, every pixel is represented with a single bit: if the pixel is white, the computer stores a 0 for that bit; if the pixel is black, the computer stores a 1 for that bit.
The first line consists of one white pixel, then three black pixels, then one white pixel. Thus the first line is encoded as 0,1,1,1,0. The second line has 4 white pixels and then 1 black pixel, so this is encoded as 0,0,0,0,1.

b) How would a computer represent the third line?

c) How would a computer represent the fifth line?

d) How would a computer represent the last line?

The problem with this approach is that it uses a lot of bits to store a large picture: we need one bit for every pixel in the picture. Using lots of bits is bad because we then need to buy more memory or disk space for the computer to store the pictures; using lots of bits is also bad because if we send the picture over the web or a modem, it can take a long time.

Programmers have invented different ways to represent, or **compress**, pictures that require fewer bits. Often, pictures have large blocks of pixels that are all white (e.g., a margin) or black (e.g., a horizontal line). The idea of **run-length compression**, or run-length encoding, is to record the number of white pixels, followed by the number of black pixels.

Going back to our picture of the letter "a", with run-length compression the computer encodes the first line as simply 1, 3, 1. The second line is represented simply as 4,1.

1, 3, 1
4, 1
?, ?
0, 1, 3, 1
?, ?, ?, ?
?, ?

**e) How would a computer encode the third line?**

The fourth line is encoded with 0, 1, 3, 1. Since the first number always relates to the number of white pixels and the first pixel is black in this line, then the line begins with a zero.

**f) How would a computer encode the fifth line?**

**g) Finally, how would a computer encode the last line?**

One complication with run-length compression is that there is usually a limit to the length of a run of pixels because the length is being represented as a binary number. For example, if the computer is using three bits to represent each run, then the computer can only record runs up to length seven.
h) If you could only use numbers up to seven, how could you represent a run of ten black pixels

The pictures we’ve looked at so far have been very small, but pictures on a computer can contain many millions of pixels. So, compressing pictures on a computer is very important.

For example, a fax machine is really just a simple computer that scans a black and white page into a grid of about $1000 \times 2000$ pixels, for a total of 2,000,000 pixels. One fax machine then sends these pixels to another fax machine using a modem. The receiving fax machine then prints the pixels out on a page. Fax images are generally compressed to about a seventh of their original size. If we didn't compress images it would take seven times longer to send pictures!

(The student should now be ready to do the worksheets.)
Activity 5: Compressing Text

Summary
Since computers only have a limited amount of space to hold information, they need to represent information as efficiently as possible. This is called compression. By encoding data before it is stored, and decoding it when it is retrieved, the computer can store more data, or send it faster through the Internet.

Skills
✓ Copying written text

Preparation
✓ None

Worksheet Activities
✓ Activity 5 Worksheet 1: Can be completed by the student independently
Activity 5: Compressing Text

In our previous lessons, we learned how a computer can store and compress pictures. Now we are going to learn how computers can compress text, or words and messages.

Computers can store whole books or even libraries, and now music and movies too, if they have the storage space in them. One solution for fitting more data in a computer is to buy more storage for them; but, this can be expensive. Another problem is that sometimes we want our computers to be very small: cellphones and wristwatches are computers, and we expect them to store lots of information too.

Another solution is to compress the data so that it takes up less space. The computer compresses the information before it stores it; and then the computer decompresses the data before presenting the information to a person. This process of compressing and decompressing the data is normally done automatically by the computer. All we might notice is that the disk holds more, or that web pages display faster, but the computer is actually doing more processing.

Many methods of compression have been invented, like the run-length encoding we used for pictures in our last lesson. Today we will be looking at ‘Ziv-Lempel coding,’ or ‘LZ coding,’ which can easily halve the size of the data being compressed. It is sometimes referred to as ‘zip’ on personal computers, and is also used for ‘GIF’ images, as well as high-speed modems.

Let us look at a poem that has a lot of repeated words to see how we can compress it.

(Use Activity 5 Handout 1: Text Compression here.)

a) In the poem, can you find groups of two or more letters that are repeated, or even whole words or phrases? Don’t forget the title! (You can ignore differences in capitalization.)

b) Can you replace those repeated words with boxes as shown below? You should draw a box around the letters that are repeated; you should draw an arrow pointing backwards from the box that is later in the poem to the earlier box.

Patter pattern

Pitter pa
Let us try to figure out how much these replacements compress the poem.

c) How many letters and spaces were in the original poem?
d) How many letters and spaces are in the compressed poem?
e) How much smaller is the compressed poem than the original? (This is usually represented as a fraction, but younger students can just report the number of letters removed.)

This isn’t quite a fair comparison, because the compressed poem requires extra information to point to the removed letters. For each arrow you drew, the computer must record a pointer back to the original words plus the number of letters, or characters, in the box. For example, the computer would replace

\[
\text{Pitter patter} \rightarrow \text{Pitter pa}(7,4)
\]

The 7 means to go back seven characters and start copying (don’t forget to count the space here!); the 4 means that four letters should be copied.

f) How many arrows did you draw for the compressed poem?
g) Let us imagine that each arrow requires as much space as two characters. So how much extra space will all of the boxes need?
h) Counting the space for these boxes, how many total letters are in the compressed poem?
i) How much smaller is this version of the compressed poem than the original poem?

(The student should now be ready to do the worksheet activity.)
Activity 6: Detecting Errors

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 to correct it.

This activity is structured differently than other activities in that many of the “answers” are interwoven throughout the text.

Skills
✓ Counting
✓ Recognition of odd and even numbers

Preparation
✓ You may find it useful to peek at the parity algorithm described in the solutions ahead of time.
✓ 36 identical cards or other objects, colored on one side only. If you don’t have any suitable objects available, then the cards provided in Activity 6 Handout 1 can be cut out.

Worksheet Activities
✓ None (yet).
Activity 6: Detecting Errors

Today we are going to learn about one way that computers make sure that data isn’t changed.

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 $10 is changed to $1,000. While you might be happy, the bank is going to be quite unhappy.

So, it is important for computers to detect errors in data and correct those errors. We will learn about one of the ways this can be done with parity. We’ll begin with a magic trick.

Please lay out these 25 cards in a 5 x 5 square, with a random mixture of sides showing. *(The following text will assume that the cards have been arranged as shown below; the black cards are face down and the white cards are face up.)*

```
 a) Now I will add another row and column of cards just to make it a bit harder. The cards going across are called a row; the cards going down are called a column. *(Important: See the solutions for how these extra cards should be placed.)*
```

```
b) Now, while I close my eyes, please flip over one of the cards. Then, I’ll tell you which one you flipped. *(Close eyes while child flips a card - don’t peek!)*
```
eyes when child is done and identify the flipped card using the algorithm described in the solutions.)

Can you guess how the trick is done? (We assume that the child will not be able to guess the solution.)

Here, I will teach you how to figure out which card has been flipped.

(Switch back the flipped card so that the parity cards are all correct again.)

The cards that you put down are called data cards; they represent data that someone cares about. The cards I added earlier are called parity cards; the parity cards are added by the computer and protect the data cards to make sure that the data isn’t changed.

I set up the parity cards very carefully to ensure that each row and column has an even number of face-up cards.

See, in the first row, with the parity card included, there are 2 cards face up, which is even. In the second row, there are 6 cards face up, which is also even. In the first column, there are 2 cards face up, and in the second column, there are 4 cards face up. So, if there are no errors, then there will always be an even number of face-up cards in every row and column.

c) Imagine one of the cards is flipped after the parity cards have been set up. How would you find the row and column the flipped card is in?

d) Let’s see if you can do the trick now. Close your eyes while I flip a card. (Flip one of the cards in the original 5 x 5 array.) Can you tell which card I flipped? (Repeat with data cards multiple times while child is interested.)

In the computer, each of these cards is represented with one bit; flipping a card is the same thing as a bit changing, or having an error. Computers need to check that bits have not been flipped whenever data is sent over a network or stored on a disk or tape.

e) Imagine that a computer is receiving a message from another computer and the message is protected with parity bits. If a computer looks at the parity bits and sees that one of the data bits has been flipped, what should the computer do?

f) Let us try one that might be a little harder. Close your eyes while I flip a card. (Flip one of the parity cards after they’ve had some practice with the data cards; everything should still work fine.) Can you tell which card I flipped?

g) If a computer receiving a message sees that a parity bit is flipped, what should it do? (Hint: Is there a problem with the data that the user cares about?)

h) Let us try another one that might be a little harder. Close your eyes while I flip a card. (This time, don’t flip a card at all.) Can you tell which card I flipped?
i) Lets have you set up the parity cards now. I'll chose the data cards and you set up the parity. *(Set up a random 5 x 5 array. Check that the child computes the correct parity cards.)*

j) Lets see if we can use your parity cards to detect a card flip. *(Take turns while child is interested.)*
Activity 7: Detecting Harder Errors

Summary
This activity is a continuation of activity 6. This activity investigates what happens when instead of a single bit being flipped, multiple errors occur simultaneously. This activity can be skipped for younger children.

Skills
✓ Counting
✓ Recognition of odd and even numbers
✓ Logical reasoning

Preparation
✓ As in Activity 6, you will need 36 identical cards or other objects, colored on one side only. If you don’t have any suitable objects available, then the cards provided in Activity 6 Handout 1 can be cut out.

Worksheet Activities
✓ Activity 7 Worsheet 1: Can be completed independently by the student.
Activity 7: Detecting More Errors

In our previous activity, we saw how parity bits can be used to detect and correct one error, that is, when one of the cards was flipped. However, sometimes in a computer, more than one error can occur at the same time. Today, we will see how parity cards work when there are more than one error.

Sometimes, simply detecting that an error has occurred is enough. For example, imagine that two computers are sending data and parity bits over a network. If the receiving computer detects that data has been corrupted over the network, it can simply ask the sending computer to send the data again. However, in some cases it isn’t possible to resend the data. For example, imagine that a computer is storing data on disk or tape. If the data on a disk is corrupted from exposure to radiation or heat, then the data is lost unless the computer can correct the errors. So, both error detection and error correction are important.

Let’s see when a computer can use parity bits for error detection and correction, when different numbers of errors happen.

a) Let us set up a parity grid again. I’ll set up 5 rows and columns of data cards and you can set up the parity cards.

Now, let’s see what happens when two cards are flipped. Let’s see if we can detect that there was an error and if we can correct the error.

b) Close your eyes while I flip two cards. (Flip two cards that are in different rows and columns from each other.) Can you figure out which two cards I flipped?

In this case, you were able to detect some errors occurred, but you aren’t able to correct them!

c) Lets try a different pair of two cards. (This time, flip two cards that are in the same row.) Can you figure out which two cards I flipped? Can you figure out something about where those two errors occurred?

In this case, you were able to detect that two errors occurred, but you aren’t able to correct the two errors. Using this scheme on a computer, when two or three errors occur, the computer can always detect that some error happened, but it can’t always correct them.

d) If a computer receives a message that seems to have two errors in it, what do you think it should do?

e) Now it is your turn to flip two cards when I’m not looking and I’ll see if I can figure out what you did. (Turn your back while the student flips the cards. Follow the procedure to isolate where the errors could have occurred.)
f) Finally, let us see what happens with four flips. Can you figure out if I flipped any cards? *(Flip four cards all in the same row.)*

g) I’m going to do one last set of four card flips. Can you figure out if I flipped any cards? *(Flip two cards in one column and then the two cards in the same rows of another column.)*

So, if four errors occur, a computer may not be able to even detect that an error occurred at all!

The following table summarizes what we have found. If one error occurs, it can always be detected and corrected. *(Point to row 1 of table.)* If two or three errors occur, the computer can always detect that an error occurred, but it might not be able to fix the errors. *(Point to row 2 of table.)* And, if four errors occur, the computer might not even be able to detect that an error happened! *(Point to row 3 of table.)*

<table>
<thead>
<tr>
<th>Number of Errors</th>
<th>Always Detect?</th>
<th>Always Correct?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2 or 3</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

A checking technique is also used with book codes. Published books have a ten-digit code, called ISBN for International Standard Book Number, which is usually found on the back cover. The tenth digit is a check digit, just like the parity bits in the exercise. This means that if you order a book using its ISBN, the publisher can check that you haven’t made a mistake. Some common errors are:

- a digit has its value changed;
- two adjacent digits are swapped with each other;
- a digit is inserted in the number; and
- a digit is removed from the number

The publisher simply looks at the checksum to make sure that you didn’t make any of these errors. That way you don’t end up buying the wrong book!

Here’s how to work out the checksum:

Multiply the first digit by ten, the second by nine, the third by eight, and so on, down to the ninth digit multiplied by two. Now add all of these values together. You next divide your answer by eleven and remember the remainder; subtract the remainder

---

*Computer Science Unplugged: Parents' Edition ©2007  Version 1.0*
from 11 and this should be the last digit of the ISBN. It is possible to come up with a checksum of the value of 10; when this happens, the character X is used.

Let us look at an example:

\[
\text{ISBN 0-13-911991-4 :} \\
(0 \times 10) + (1 \times 9) + (3 \times 8) \\
+ (9 \times 7) + (1 \times 6) + (1 \times 5) \\
+ (9 \times 4) + (9 \times 3) + (1 \times 2) \\
= 172 \\
172 \div 11 = 15 \text{ with a remainder 7} \\
11 - 7 = 4
\]

h) Look back. Is 4 the last digit of the ISBN? If the last digit wasn't a four, then we would know that a mistake had been made.

i) Can you calculate the checksum for 1-00-235045?

Another example of the use of a check digit is the bar codes on grocery items. This uses a different formula. If a bar code is misread the final digit should be different from its calculated value. When this happens the scanner beeps and the checkout operator re-scans the code.

\[\text{\footnotesize A barcode (UPC) from a box of Weet-Bix™}\]
Activity 8: Information Theory

Summary
How much information is there in a message. This activity introduces a way of measuring information content.

Skills
✓ Comparing numbers and working with ranges of numbers
✓ Deduction
✓ Asking questions

Preparation
✓ Obtain two small objects (e.g., pennies) to mark the min and max guesses on Handout 1
✓ Activity 8 Handout 1: Used to track guesses between 1 and 100

Worksheet Activities
✓ Worksheet Activity (Optional): Can be completed by student independently
Activity 8: Information Theory

Today we will be learning about information. Knowing how much information is in a message is important because that determines how much space is needed to store that message. The less information in a message, the more that message can be compressed.

a) What do you think information is? Can you think of any examples?

b) How do you think we could measure how much information there is in a book?

Computer scientists measure the amount of information in a message by measuring how surprising a message is.

Telling you something that you know already—for example, when a friend who always walks to school says “I walked to school today”—doesn’t give you any information, because it isn’t surprising. If your friend said instead, “I got a ride to school in a helicopter today,” that would be surprising, and would therefore tell us a lot of information.

c) How would you order these books from the most information to the least?

- 1000 pages of blank paper
- 1000 pages of the words “blah, blah, blah” repeated over and over
- 1000 pages of a telephone book
- 500 pages of a telephone book

d) How do you think we could measure how surprising a message is?

One way to measure surprise is to see how hard it is to guess the information. If your friend says, “Guess how I got to school today,” and they had walked, you would probably guess right first time. It might take a few more guesses before you got to a helicopter, and even more if they had travelled by spaceship.

To see how much information is in a message, we are going to measure the number of guesses it takes to find a secret. We’ll play a game like 20 questions. You can ask any question to try to guess what object I am thinking of, and I can only answer ‘yes’ or ‘no’. Try to guess what I am thinking about with as few questions as possible.

e) (Play 20 questions about three times, with progressively harder objects to guess. Record the number of tries it takes for the student to guess the secret object. The rarer objects should take more questions to guess correctly. A few suggestions for secret objects are in the solutions, if needed.)

Let’s now see how much information is in different numbers by playing the same guessing game with numbers.
This time, I’m thinking of a number between 1 and 100. Can you guess it? (Repeat a few times, recording how many guesses were required. Unless the student is performing a “binary search”, he or she will probably need more than 7 guesses sometimes. If they are already performing a binary search, it still should be useful to go through the binary search algorithm to formalize this process.)

The best strategy for guessing numbers is to use what is called a **binary search**. With a binary search, you halve the range of possible answers each time. By dividing the search space into two with each guess, anyone can guess any secret number between 1 and 100 in only 7 guesses. To help remember the minimum and maximum numbers that have been guessed, it can help to mark them. (Use the number sheet with small objects such as coins marking the lowest and highest possible values for the secret number.)

Let’s see how a binary search works when the secret number is 42. At the beginning, the secret number could be any number between 1 and 100, so we put the markers at the ends. (Put object markers at extreme ends of number list.)

1. We want to divide the range of numbers into two halves. What number is at about the half-way point? (50.) So, our first question is “Is the secret number less than 50?”. The answer is “Yes”, so we move the MAX marker to 50. (Place the MAX marker appropriately.)

2. We want to continue to divide the range of possible numbers into two halves. What number is about half-way between 0 and 50? (25.) So, our second question is “Is it less than 25?” The answer is “No”, so we move the MIN marker to 25.

3. Now we need a number halfway between 25 and 50. What number should we choose? (37.) So, our third question is “Is it less than 37? No, so we move the MIN marker to 37.

4. What is the number halfway between 37 and 50? (43). Is it less than 43? Yes, so we move the MAX marker to 43.

5. What is the number halfway between 43 and 37? (40.) Is it less than 40? No, so we move the MIN marker to 40.

6. What is the number halfway between 40 and 43? (41.) Is it less than 41? No, so we move the MIN marker to 41.

7. Since there is only one number left, the secret number must be 42! Yes!

Now, let’s have you think of a secret number and I’ll guess it in no more than 7 guesses. (Use the algorithm described above. Repeat a few times, depending upon interest.)

h) Now, I’ll think of a secret number again, and see if you can guess it in 7 guesses.
Interestingly if the range is increased from 100 to 1000 it doesn’t take 10 times the effort—just three more questions are needed. Every time the range doubles you just need one more question to find the answer.

i) Do you want to play the guessing game with numbers up to 1000? *(If the student has interest, take turns guessing.)*

*(The student should now be ready to do the worksheet.)*
Activity 9: Searching

Summary
Computers are often required to find information in large collections of data. They need to develop quick and efficient ways of doing this. This activity demonstrates three different search methods: linear searching, binary searching and hashing.

Skills
✓ Logical reasoning
✓ Greater than, less than, equal to

Preparation
✓ The battleship cards need to be cut out from the 3 pairs of handouts. One person should have the A handouts and the other person the B handouts; it is best if each person cuts out their own battleships so the other person doesn’t see the ship numbers ahead of time. Each pair of handouts will be used for a different game.
✓ Handout 1: The tops of the cards should have question marks.
✓ Handout 2: The tops of the cards should have the numbers 1-25.
✓ Handout 3: The tops of the cards should have numbers from 0 to 9.

Worksheet Activities
✓ None
Activity 9: Searching

In our previous activities, we’ve been learning mostly about data. We are now going to focus on algorithms. Remember, algorithms are the instructions that tell computers what to do.

Some algorithms are faster than others. It is important for people to discover fast algorithms so that computers can solve problems quickly. If an algorithm is too slow, people might not be able to wait long enough for the answer the computer is computing. Some examples of algorithms that need to be fast are finding millions of digits in pi, or all pages that contain your name on the World-Wide Web, 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.

Today, we are going to explore different algorithms for searching for a particular number in a large group. Searching is an important problem because computers store a lot of information, and they need to be able to sift through it quickly. 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.

We will learn about searching by playing games similar to Battleship. We will each pick a secret battleship, and the other person has to find it in the group of all the battleships. 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. Our search key will be the number of the secret battleship.

(Start with the battleships from Handouts 1a and 1b, labelled Secret Ships; one player uses handout 1a and the other player 1b. These cards have question marks written on the back.)

Pick a battleship as your secret ship and tell me its number for the search key. Then, hide your secret battleship face down among all of the others and mix them up. I’ll do the same thing.

Now, we’ll take turns guessing where the secret ship is located. We’ll flip over one of the battleships on each guess and we’ll see how many guesses each of us take to find the secret ship. You can go first.

(Take turns playing the Linear Searching Game. Make sure to keep each battleship turned face up after it has been guessed.)

a) How many shots did it take to locate your partner’s secret ship? This is your score for the game.

(Repeat the game while student is interested, recording the score of each game.)

b) What is the best score that we received? What is the best possible score?
c) What is the worst score that we received? What is the worst possible score?

d) What would happen if the secret ship wasn’t there? How many guesses would it take to find that out?

e) About how many guesses do you think it will usually take to find the secret ship with the Linear Searching Game?

f) If there were 100 battleships, how many guesses do you think it would usually take?

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 called **Linear Searching**, and is what we did in this game of Battleship. But this algorithm 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, ten seconds would be needed to go through the whole list. Imagine how long it would take to check out the groceries for a family!

We will now try a faster searching algorithm called **Binary Searching**. This is the same algorithm that we used in the last lesson activity to play 20 questions.

*(Get out the cards from handouts 2a and 2b handouts cards that have the sorted order printed on the other side.)*

This time when we play Battleships, we will sort the battleships from lowest to highest, and put them in a line. When the battleships are sorted and you pick a battleship in the middle, you can always figure out on which half the secret ship must be. If the secret ship has a higher number than the middle ship, then you know the secret ship is somewhere in the higher half; if the secret ship has a lower number, then the secret ship is somewhere in the lower half.

Let’s try. Pick a secret ship and tell me its number (but don’t show me what the back looks like). Then, put the secret ship face down with all of the other ships in sorted order. On the back of these cards, the sorted order is written for us.

Now, we’ll take turns guessing where the secret ship is located.

g) How many shots did it take to locate your partner’s ship? This is your score for the game.

*(Repeat the game while student is interested, recording the score of each game.)*

h) What is the best score that we received? What is the best (minimum) possible score?
i) What is the worst score that we received? What is the worst (maximum) possible score?

j) What would happen if the secret ship wasn’t there? How many guesses would it take to find that out?

k) About how many guesses do you think it will usually take to find the secret ship?

The binary search game should take fewer guesses than the linear search game. 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 fourteen probes, which might take two hundredths of a second—hardly noticeable.

However, there is an even faster searching algorithm that can sometimes be used. This time, instead of first sorting the battleships, we will hash them into different buckets. Hashing is a way to manipulate search keys and put related search keys in the same place. Then, when you want to find those search keys, you know exactly where to look.

A simple hash function is to add together the digits of the ship’s number and then take the last digit of the sum. The last digit of the sum is the bucket the ship is in. For example, to locate a ship numbered 2345, add the digits 2+3+4+5, giving 14. The last digit of the sum is 4, so that ship must be in bucket 4.

\[
\begin{align*}
\text{Ship 2345} & \rightarrow 2 + 3 + 4 + 5 = 14 \\
& \rightarrow \text{Bucket 4}
\end{align*}
\]

This last time when we play Battleships, we will hash the battleships into buckets numbered from 0 to 9. On the backs of all of these cards, the bucket number is printed. In this game, you can find out which bucket (0 to 9) the secret ship is in with hashing. Once you know the bucket, you only need to look at ships which are in the desired bucket.

Let’s try. Pick a secret ship and tell me the number. Then, put the secret ship face down with all of the ships. Now, we’ll take turns guessing where the secret ship is located.

l) The first step is to calculate the bucket for the secret ship. \((Figure \ out \ the \ bucket \ of \ the \ student’s \ secret \ ship \ out \ loud.\) In which bucket is my secret ship? \((Have \ the \ student \ figure \ out \ in \ which \ bucket \ your \ secret \ ship \ is \ located.\)

m) How many shots did it take to locate your partner’s ship? This is your score for the game.
(Repeat the game while student is interested, recording the score of each game.)

n) What is the best score that we received? What is the best (minimum) possible score?

o) What is the worst score that we received? What is the worst (maximum) possible score?

p) What would happen if the secret ship wasn’t there? How many guesses would it take to find that out?

q) About how many guesses do you think it will usually take to find the secret ship?

r) If you want to hide your secret ship for as long as possible, which bucket should you pick?

s) If you want to find your opponents secret ship as quickly as possible, in which bucket do you hope it is in?

Hashing is often the best searching algorithm, but its speed depends upon how many items are in each bucket. The fewer the items in each bucket, the faster the search.

To make searching fast, the computer uses a lot of buckets and chooses a hash function that assigns about the same number of items in each bucket. In the best case, there is only one item in each bucket – then, once the computer knows the right bucket, it doesn’t have to search at all!

t) What are the advantages of each of the three different ways of searching? (Hint: Which is the fastest? Which require that the items are organized in some way?)
Activity 10: Sorting

Summary
Computers are often used to put lists into some sort of order. This activity introduces three basic sorting algorithms: selection sort, insertion sort, and bubble sort. Students learn to count the number of comparison operations a sorting algorithm is expected to require.

Skills
✓ Ordering
✓ Comparing

Preparation
✓ This activity is the most fun if you have a balance scale and a set of 8 weights. The 8 weights should all look identical; that is, the student shouldn’t be able to tell which weighs more just by looking at them.
✓ If you do not have a balance scale, then you can cut out the cards on the handout; instead of using the scale, the student will pass the cards to the adult who will tell them which card is higher and which is lower. The student should not be able to see the numbers on the cards.

Worksheet Activities
✓ None
Activity 10: Sorting

Computers are often used to put lists into some type of order, for example names into alphabetical order, appointments or e-mail by date, or items in numerical order. Sorting lists helps us find things quickly; for example, in the Searching Activity, we saw that we could perform a fast binary search if the list is already sorted. Sorting lists also makes extreme values easy to see; for example, if you sort the grades for a class test into numeric order, the lowest and highest marks become obvious.

a) Brainstorm all the places where putting things into order is important. What would happen if these things were not in order?

Computers can usually only compare two values at once. For example, if the computer has the task of finding the smallest number in the list “10 50 2 30 8” it isn’t able to examine the whole list at once and just “see” that 2 is the smallest number. The computer has to compare just two numbers at a time. The problem is, when we try to pretend they are doing the same work and comparisons as a computer, it is very difficult for us to avoid comparing more than 2 numbers at once.

(If you are using a balance scale, then read the following.)

To make sure you only compare two numbers at a time, you will be using a balance scale. This way, you can carefully count the number of comparisons that you do. To compare two different weights, you must always use the scale and you can only compare two weights at the same time.

Let’s begin by practicing with the scale. Compare just two weights. Which is lighter? (Make sure the child knows how to use the scale correctly.)

(If you do not have a balance scale, then read the following.)

To make sure you only compare two numbers at a time, you will not be allowed to look at any of the numbers on these cards. When you want to compare two cards, you must give them both to me face down, and I will tell you which is smaller.

Knowing the number of comparisons a sorting algorithm must perform is important for knowing how fast that sort will run on a computer.

Sorting takes as its input a list of unsorted keys and produces as output a list of sorted keys. If you are using the balance scale, then the keys are the weight of each object. If you are using the cards, then the keys are the numbers on the back.

b) Find the lightest weight (or the smallest number) among the eight objects. What is the easiest way of doing this? Count the number of comparisons that you make as you go.

c) How many comparisons did you make?
d) Choose 3 weights (or cards) at random and sort them into order from lightest to heaviest (or lowest to highest). How did you do this?

e) What is the minimum number of comparisons you can make? Why?

If you use the wrong method for sorting, it can take a long time to sort a large list into order, even on a fast computer. Fortunately, several fast methods are known for sorting. In this activity you will discover different methods for sorting, and see how a clever method can perform the task much more quickly than a simple one.

One method a computer might use is called selection sort. This is how selection sort works. First find the lightest weight in the set and put this first in the sorted list. Next, find the lightest of the weights that are left, and put it second in the sorted list. Repeat this until all the weights have been removed.

![Selection Sort Diagram]

f) Mix up 8 of the weights so that they are in a random order. Then, use a selection sort to sort the 8 objects. Count how many comparisons you made.

Insertion sort works by removing each object from an unsorted group and inserting it into its correct position in a growing list (see picture below). With each insertion the group of unsorted objects shrinks and the sorted list grows, until eventually the whole list is sorted. Card players often use this method to sort a hand into order.

Specifically, follow these steps to implement an insertion sort. First, pick one object to form your (very short) list of sorted objects on the left. Next, pick a second object from the unsorted group; use the scale to determine if this second object is less than or greater than the first object. Then, pick a third object; use the scale to determine how much the third object weighs relative to the first two objects. Continue to pick objects from the unsorted group and insert them into the sorted list.

g) Mix up 8 of the weights so that they are in a random order. Then, use an insertion sort to sort the 8 objects. Count how many comparisons you made.

![Insertion Sort Diagram]

Bubble sort involves going through the list again and again, swapping any objects side-by-side that are in the wrong order. The list is sorted when no swaps occur during
a pass through the list. This method is not very efficient, but some people find it easier to understand than the others.

h) Mix up 8 of the weights so that they are in a random order. Then, use an bubble sort to sort the 8 objects. Count how many comparisons you made.

i) Which of the sorting algorithms required the fewest number of comparisons?

j) Expert: In the worst case, for all of these sorting algorithms, how many comparisons do you think are needed if you have 20 objects instead of 8?
Activity 11: Faster Sorting

Summary

This activity continues Activity 10 on Sorting.

If you use the wrong algorithm for sorting, it can take a long time to sort a large list into order, even on a fast computer. Fortunately, several fast methods are known for sorting. In this activity children will explore two faster sorting algorithms: Quicksort, Mergesort. The children will see that a clever method can perform the task much more quickly than a simple one.

Skills

✓ Ordering
✓ Comparing

Preparation

✓ The same weights or cards are required as for Activity 10.
✓ This activity is the most fun if you have a balance scale and a set of 8 weights. The 8 weights should all look identical; that is, the student shouldn’t be able to tell which weighs more just by looking at them.
✓ If you do not have a balance scale, then you can cut out the cards on the handout; instead of using the scale, the student will pass the cards to the adult who will tell them which card is higher and which is lower. The student should not be able to see the numbers on the cards.

Worksheet Activities

✓ None.
Activity 11: Faster Sorting

Computers spend a lot of their time sorting things into order, so computer scientists have to find fast and efficient ways of doing this. Some of the slower methods such as insertion sort, selection sort and bubble sort can be useful in special situations, but the fast ones such as quicksort are usually used.

Quicksort is a lot faster than selection sort, particularly for larger lists. In fact, it is one of the best methods known. This is how quicksort works.

First, choose one of the objects at random, and place it on one . Next, compare each of the remaining objects with the chosen object. Put the objects that are lighter on the left, the chosen object in the middle, and the heavier objects on the right. (By chance you may end up with many more objects on one side than on the other.)

Choose one of the groups and repeat this procedure. Do the same for the other group. Remember to keep the one you know in the centre.

Keep repeating this procedure on the remaining groups until no group has more than one object in it. Once all the groups have been divided down to single objects, the objects will be in order from lightest to heaviest.

a) Mix up 8 of the weights so that they are in a random order. Then, use quicksort to sort the 8 objects. Count how many comparisons you made.

b) Whether or quicksort is quick or not depends upon how lucky you are when you pick the “pivot” object that you divide the other objects around. What is a good pivot object? What does a good pivot object do?

Quicksort uses a concept called recursion in which the computer keeps dividing the list into smaller parts and performing the same kind of sort on each of the parts. This particular approach is called divide and conquer. The list is divided repeatedly until it
is small enough to conquer. For quicksort, the lists are divided until they contain only one item. It is trivial to sort one item into order! Although this seems very involved, in practice it is dramatically faster than other methods.

**Mergesort** is another method that uses ‘divide and conquer’ to sort a list of items. First, the list is divided at random into two lists of equal size (or nearly equal if there are an odd number of items). Each of the two half-size lists is sorted, and the two lists are merged together. Merging two sorted lists is easy—you repeatedly remove the smaller of the two items at the front of the two lists. In the figure below, the 40 and 60-gram weights are at the front of the lists, so the next item to add is the 40-gram weight.

How do you sort the smaller lists? Simple—just use mergesort! Eventually, all the lists will be cut down into individual items, so you don’t need to worry about knowing when to stop.

Here is another way of looking at mergesort. Begin by putting each weight into its own list of size one. Then, pick pairs of weights and “merge” them into a sorted list of size two. For your next step, again pick pairs of sorted lists (this time each sorted list will have two elements) and merge them into lists of four objects. Finally, for your last step, merge the two lists of four objects into a single sorted list of eight objects.

c) Mix up 8 of the weights so that they are in a random order. Then, use mergesort to sort the 8 objects. Count how many comparisons you made.

d) Which algorithm was the fastest of all of those you’ve seen: selection sort, insertion sort, bubble sort, quicksort, and mergesort?
Activity 12: 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 which do several sorting comparisons at the same time.

Skills
- Ordering
- Comparing
- Developing algorithms

Materials
- Handout 1: The sorting network
- Handout 2: Cut out 6 cards (smaller numbers for younger children, larger numbers for older children).
Activity 12: Sorting Networks

We want computers process information as quickly as possible. One way to increase the speed of a computer is to develop algorithms that use fewer computational steps. For example, we saw that Quick Sort generally takes fewer steps than Bubble Sort or Insertion Sort.

Another way to solve problems faster is to have several computers (or parts of the computer) work on different parts of the same task at the same time. For example, to sort faster, one computer could compare two numbers while another computer compares two different numbers. This is an example of parallel computation.

Today, we will see how a sorting network can be used to sort a list of numbers more quickly. The sorting network we will use is shown on Handout 1.

Here, each node, or circle, in the network represents one of three different computers, A, B, and C. (Point to the different nodes representing computers.)

Each edge, or arrow, shows how the numbers are sent between computers. (Point to the edges.)

At each time step, each computer compares two numbers and sends the higher number to one computer and the lower number to a different computer. We’ll say that the higher number is sent along the upper outgoing edge and the lower number is sent along the lower outgoing edge. The first column of nodes represents the work that must be done in the first time step. (Point to the three nodes in the first column.)

For example, in time step 1, computer B sends its higher number to computer A and its lower number to computer B. (Point to this on the sorting network.)

a) In Step 1, where does computer A send its higher number?
b) In Step 1, where does computer A send its lower number?
c) In Step 1, where does computer C send its higher number?
d) In Step 1, where does computer C send its lower number?

In Step 1, computers A, B, and C are all performing these comparisons at the same time, or in parallel. After each computer receives their next two numbers, the comparisons for Step 2 can begin. The nodes in the second column correspond to the work that must be done in the second time step. (Point to the three nodes in the second column.)

e) In Step 2, where will computer A send its higher number?
f) In Step 2, where will computer A send its lower number?
(You may ask about computers B and C as well for step 2, if needed.)

g) Where are the nodes corresponding to the third time step?

h) Where are the nodes corresponding to the fourth time step?

i) And, finally, which nodes are involved in the fifth time step?

j) Let us now see if this sorting network will really sort random numbers. Start with the cards 5, 1, 6, 3, 4, 2 assigned to the boxes starting from the top. In the first time step, what will the compute nodes do?

k) In the second time step, what will the compute nodes do?

l) What will the compute nodes do in the third time step?

m) What do you notice happens after three time steps?

n) What will the compute nodes do in the fourth time step?

o) And, finally, what happens in the fifth time step?

p) Try sorting a different combination of the numbers as well. (You may want to use the larger numbers here.)

This six-number sorting network can faster than running the quicksort algorithm because the sort completes in just 5 comparison steps. Even though the sorting network requires a total of 12 comparisons, up to 3 of them are being performed simultaneously. With the quicksort algorithm, none of the comparisons were being performed simultaneously. This parallel network sorts the list more than twice as quickly as a system that can only perform one comparison at a time.

Now, let us consider some variations of this sorting network.

q) What happens if the smaller one goes up instead of down, and vice versa?

r) Expert: Can you design your own sorting network that will sort three numbers? (Remember, each compute node can compare only two numbers.)
s) Below are two different networks that sort four inputs. Which is the faster? Why?

Networks can also be used to perform other computations across a set of inputs.

t) Can you figure out what function this network computes? In other words, which of the input numbers will this network always give as output?. The X’s all
represent dead ends in which the number isn’t sent, or propagated, any further.

Some processes from everyday life can be accelerated using parallelism, and some can’t.

u) Can digging a 10-foot long ditch be accelerated with more people and shovels?

v) Can digging a 10-foot deep ditch be accelerated?

w) What aspects of cooking a meal can and can’t be accelerated using parallelism?

x) What aspects of washing clothes can and can’t be parallelized?

In summary, some tasks in life and on computers can be parallelized to go faster, and some can’t. Computer Scientists are still actively trying to find the best ways to break problems up so that they can be solved by computers working in parallel.
Activity 13: Muddy City

Summary
Our society is linked by many networks: 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.

Skills
✓ Problem solving

Preparation
✓ Counters or squares of cardboard (approximately 40 per child)

Worksheet Activities
✓ Worksheet 1: The muddy city problem
Activity 13: The Muddy City

This activity will show you how computers are used to find the best solutions for real-life problems such as connecting houses with the minimal number of roads.

Imagine there is a city with no paved roads. Getting around the city is difficult after rainstorms because the ground becomes very muddy—cars get stuck in the mud and people get their boots dirty. The mayor of the city decrees that some of the streets must be paved, but doesn’t want to spend more money than necessary because the city also wanted to build a swimming pool. The mayor therefore specifies 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, and
2. The paving should cost as little as possible. The number of paving stones between each house represents the cost of paving that route.

a) Consider this very small town with only three houses and three roads. Which roads should be paved to connect all houses, using the fewest stones?

b) Given these paved roads, what path would you take to walk from house A to house C?
c) Let us examine a slightly larger town with five houses. Which roads should you pave in this town to minimize the number of stones needed?

![Diagram of a town with five houses and paved roads]


d) How would you walk from house A to house D with your paved roads?

e) Are there other choices for paving roads that are just as good? Why?

f) If you were walking from house B to house E, which version of paved roads would be better for you?

The task of designing a network with a minimal total length is called the **minimal spanning tree** problem. You have been solving minimal spanning trees. Minimal spanning trees are useful for other problems as well.

g) Can you think of any other problems where a minimal spanning tree could be used? (Hint: Besides roads, what else needs to go to every house?)

Another way of representing houses and roads is with a diagram, called a **graph**. In these graphs, the houses are represented with circles, the muddy roads with lines between the circles, and the length of each road with a number beside the line. The following is an example graph of a rather large town.

h) How many houses are in this town?

![Diagram of a large town with graph representation]
Here are four versions of a graph representing the first town with 3 houses. Although the four graphs might look a little different to you, all four graphs are actually the same, in that they all represent the same town.

i) How can all four graphs represent the same town?

j) Can you draw the graph that corresponds to the second town with 5 houses?

There are also many other algorithms besides minimal spanning trees that can be applied to graphs, such as finding the shortest distance between two points, or the shortest route that visits all the points. This problem is often referred to as the travelling salesperson problem. For many problems on graphs, including the “travelling salesperson problem”, computer scientists are yet to find fast enough methods that find the best possible solution.
Activity 14: Routing and Deadlock

Summary
When you have a lot of people using one resource (such as cars using roads, or messages getting through the Internet), there is the possibility of “deadlock”. A way of working co-operatively is needed to avoid this happening.

Curriculum Links
✓ Mathematics: Developing logic and reasoning

Skills
✓ Logical reasoning

Preparation
✓ Cut out the cards representing messages in Handout 3.

Worksheet Activities
✓ None.
Activity 14: Routing and Deadlock

Computers that are connected to one another with a network usually want to send messages to one another. The problem of sending a message to the correct destination computer is called routing.

We are going to see how computers can send messages in a routing network. In this picture, each circle represents a computer; each line represents a path that the computers can send a message down; each message is represented by a piece of paper that we move. Your goal is to route, or send, each message to the computer with the same name following only the marked paths. At the end, all of the computers will be holding their correct messages. While you are routing the message to the correct destination, each computer has room for only two messages at a time. Only one message at a time can be in flight between computers.

a) First try to route with just two computers. (Use the worksheet with two computers and three messages.) Two messages that are destined for A start in computer B and one message destined for B starts in computer A. How many moves does it take to correctly route the three messages?

b) Does the problem work if there are two messages destined for both A and B? Why or why not?

c) Now try three computers, A, B, and C, with the messages randomly assigned at the beginning. (Use the worksheet with three computers and seven messages.) C should have only one message destined for it. Can you route the messages to the correct destination?

d) Now try four computers, A, B, C, and D, with the messages randomly assigned at the beginning. (Use the worksheet with four computers and nine messages.) Now D should have only one message destined for it. Can you route the messages to the correct destination?

(Repeat with more computers and messages if the student is interested.)

When you were routing messages, you had control over all of the messages in the network and you could control what each computer did at each step. This is different than computers on a real network because there, each computer acts independently with no one telling all of the computers what they should do at each step.

Sometimes computers use greedy algorithms, even though they are all working toward a common goal. A greedy algorithm is an algorithm in which every computer tries to do the best that it can for itself at each moment in time. A greedy routing algorithm would state that once a computer receives one of its destination messages, the computer holds onto it and won’t let it go.

e) Do you think a greedy algorithm works for this routing game?
A greedy algorithm could lead to the following situation with four computers. Computers A and C each have their correct messages, while computers B and D do not.

f) If computers A and C are greedy and refuse to let go of their messages, can the correct messages get to B and D?

In this routing game, a greedy algorithm can lead to **deadlock**. When there is deadlock, progress cannot be made because somebody is holding on to a resource that someone else needs and they won't let go.

g) If only computer C is greedy can the correct messages get to B and D?

Deadlock happens in situations other than computers sending messages over a network.

h) Can you think of any other examples where deadlock could happen in the real world? (Hints: With people or with cars.)

Routing and deadlock are problems in many networks, such as road systems, telephone and computer systems. Engineers spend a lot of time figuring out how to solve these problems—and how to design networks that make the problems easier to solve.
Activity 15: 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 recognise the word or string of symbols. We will be working with something equivalent to a FSA—treasure maps!

Skills
✓ Simple map reading
✓ Recognising patterns
✓ Logic
✓ Following instructions

Preparation
✓ Cut out the island cards from the Handouts. The Island Cards should be folded along the dotted line and placed so that the student cannot see where path A or B from that island takes one.

Worksheet Activities
✓ None.
Activity 15: Finite-State Automata

Computer programs often need to process a sequence of symbols such as letters or words in a document. 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 recognise the word or string of symbols. We will be working with something equivalent to a FSA—treasure maps!

Imagine that you live in a world with three islands: Pirates’ Island, Shipwreck Bay, and Dead Man’s Island. Your goal is to create a map showing how to get from one island to the next.

(Begin with the first map plus the six Island Cards cut out. The Island Cards should be folded along the dotted line and placed so that the student cannot see where path A or B from that island takes one.)

Friendly pirate ships sail between the islands, offering rides to travellers. Each island has two departing ships, A and B, which you can choose to travel on. At each island you arrive at you may ask for either ship A or B (not both). There are two cards for each Island, one showing where ship A will take you and one showing where ship B will take you. Flipping over the card for that ship will tell you the island your ship will take you to next.

a) Can you create a map of the paths between the three islands? If you reach a dead end, you must start over again at Pirates’ Island.

Now let us try a larger map with seven islands. This time, you are trying to find a path to Treasure Island.

(Use the second worksheet map plus the 12 Island Cards cut out. The Island Cards should be folded along the dotted line and placed so that the student cannot see where path A or B from that island takes one.)

b) Can you find a path to Treasure Island? Mark the paths you discover on your map as you go.

c) If you didn’t create a complete map to get to Treasure Island, complete the map showing all possible paths between islands.

d) Now that you have the complete map, what is the quickest route to Treasure Island?

e) Can you find a slower route to Treasure Island, such as one that goes to all of the islands?

f) Can you create a route to Treasure Island that involves loops (that is, that visits some islands more than one time)?
You can also create your own treasure hunt across the islands with your own paths.

g) Using Handout 3, draw your own map to Treasure Island. How difficult can you make it to find the treasure? Keep your map secret from me.

h) Then, using your map, fill in the blank island cards to show where ship A and ship B will take a traveller. Set up of the filled-in island cards so that I can’t see where each ship goes until I turn them over.

Now it is my turn to try to find Treasure Island. *(Follow paths while filling in the other blank map to find Treasure Island.)*

i) What is the most efficient sequence of routes to reach Treasure Island?
Activity 16: FSA Representation

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 recognise the word or string of symbols. We will be working with something equivalent to a FSA—treasure maps!

Skills
✓ Simple map reading
✓ Recognising patterns
✓ Logic
✓ Following instructions

Preparation
✓ None.

Worksheet Activities
✓ None.

ARR! THIS CAN'T BE A TREASURE MAP! WHERE'S THE 'X'?
Activity 16: FSA Representation

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

A simple example is when you dial up 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 phone. The dialogue can be quite simple, or very complex. Sometimes you are taken round 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 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, like "dispense $100 of cash" or "print a statement" or "eject the cash card".

Computer scientists draw maps of finite state machines using circles to represent each state and arrows to show how to move from one state to the next. For example, our first Pirate Map with Dead Man’s Island would be represented as a FSA like this:
Similarly, the map to Treasure Island would be represented with an FSA like this:

In this example, the final island with the treasure is shown with a double circle. In FSA terminology, this is the \textbf{accept state}.

a) Using the FSA, what is the most efficient way to get to Treasure Island?

b) Are there paths with loops in them that get to Treasure Island? Can you give an example?

c) Does the path BBBABBABAB lead to Treasure Island?

d) Does the path BAABAAABABABAB?

e) Does the path BBBAAAB?
Here are three more examples of FSAs.

In map (a), which of the following strings will be accepted?

f) AB

h) ABBBA

i) AAABABA

j) AAABA

k) Can you describe what is the same about all of the strings that are accepted by map a? (Hint: Count the number of As). Why is this?

In map (b) which of the following strings of letters will be accepted?

l) AB

m) BA

n) ABAB

p) ABABA

q) ABBA

r) Can you describe what is the same about all of the strings accepted by map b?

In map (c) which of the following strings of letters will be accepted?

s) B

t) BBB

u) BBA

v) ABBA
w) AAA

x) Can you describe what is the same about all of the strings that are accepted by map c?

Some computer programs deal with English sentences using maps. They can both generate sentences themselves, and process sentences that the user types in.

Here is a way of constructing sentences by choosing random paths through the map and noting the words that are encountered.

y) Can you use this map to create a sentence?

In the 1960s a computer scientist wrote a famous program called “Eliza” (after Eliza Dolittle) that had conversations with people. The program pretended to be a psychotherapist, and came out with leading questions like “Tell me about your family” and “Do go on.” Although it didn’t “understand” anything, it was sufficiently plausible—and its human users were sufficiently gullible—that some people really did think they were talking to a human psychotherapist.

z) Try creating a FSA to produce sentences of your own. Can someone else use your FSA to produce sentences?

Although computers are not really very good at understanding natural language, they can readily process programming languages. Computers use finite-state automata to read in programs and translate them into the form of elementary computer instructions, which can then be “executed” directly by the computer.
Activity 17: Programming Languages

Summary
Computers are usually programmed using a "language," which is a limited vocabulary of instructions that can be obeyed. One of the most frustrating things about programming is that computers always obey the instructions to the letter, even if they produce a crazy result. This activity gives children some experience with this aspect of programming.

Skills
✓ Giving and following instructions.

Preparation
✓ The handout contains pictures that the adult should not see ahead of time. The adult must be careful not to look at this handout!
✓ Need pencil, paper, and maybe a ruler.

Worksheet Activities
✓ None.
Activity 17: Programming Languages

Computers operate by following a list of instructions, called a **program**, that has been written to carry out a particular task. Programs are written in languages that have been specially designed, with a limited set of instructions, to tell computers what to do.

One of the most frustrating things about programming is that computers always obey the instructions exactly, even if they produce a crazy result. This activity will give you some experience with this aspect of programming.

a) Do you think it would be a good if people followed instructions exactly?

b) Can you think of any examples where it would be bad if people followed instructions exactly?

Let us see if I can give you precise enough directions that you can draw the picture I want.

1. Draw a dot in the center of your page.
2. Starting at the top left-hand corner of the page draw a straight line through the dot finishing at the bottom right hand corner.
3. Starting at the bottom left-hand corner of the page draw a line through the dot, finishing at the top right hand corner.
4. Write your name in the triangle in the center of the left-hand side of the page.

c) What does your picture look like?

Now it is your turn to pick a picture and describe it precisely to me. In our first version, you can watch what I’m doing and correct me if I am doing anything wrong. Let us see how quickly and accurately we can do this.

*(Have the child choose an image from the next page and describe it to you in steps. You should not look at the picture!)*

d) Does the picture I drew look exactly the same as the one you were describing?

Let us try another picture, but this time you aren’t allowed to look at what I’m doing and correct me, but I’m allowed to ask questions. You need to turn your back to me while you describe the picture this time.

*(Have the child choose another image from the next page and describe it to you in steps. You should not look at the picture! The child should have their back to you while they describe the picture.)*
e) How does the picture I drew compare to the one you were describing this time? Why is it harder to get me to draw the correct picture when you can’t see what I am drawing?

Let us try one last picture. This time you aren’t allowed to look at what I’m doing and I’m not allowed to ask questions.

This form of communication is most like the one that computer programmers experience when writing programs. They give a set of instructions to the computer, and don’t find out the effect of the instructions until afterwards.

(Have the child choose a third image from the next page and describe it to you in steps. You should not look at the picture! The child should have their back to you while they describe the picture and you cannot ask questions.)

f) How does the last picture I drew compare to the one you were describing this time? Why is it harder to get me to draw the correct picture when you can’t see what I am drawing and I can’t ask questions?

It is important that programs are well written. A small error can cause a lot of problems.

g) What do you think would happen if there is an error in the program of a computer in a space shuttle launch, a nuclear power plant, or the signals on a train track?

Errors are commonly called “bugs” in honor (so it is said) of a moth that was once removed (“debugged”) from an early electronic calculating machine in the 1940s.

![Bug](image)

The more complex the program, the more errors there are likely to be. Software needs to be tested carefully to find as many bugs as possible before it is used, especially in important systems that interact with people.

Computer scientists still do not know how to prove that a computer program does not have any bugs and will behave as expected.