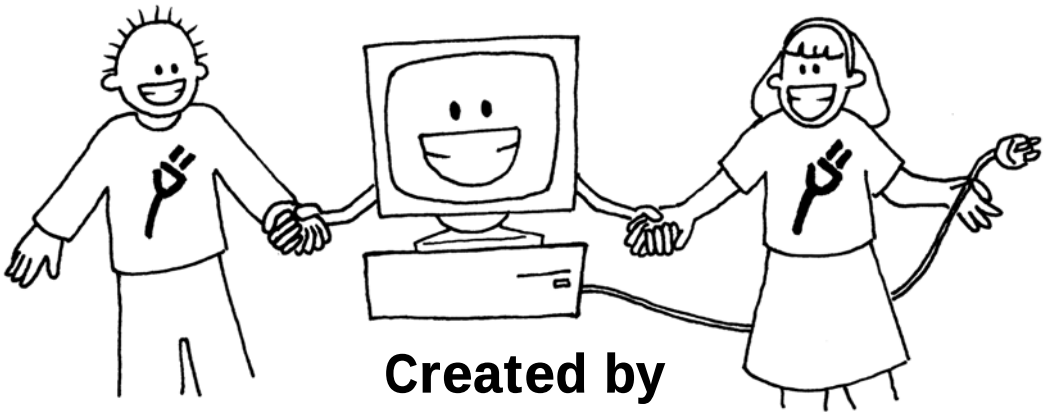
COMPUTER SCIENCE Unplugged

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

December 2006

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!

This book can be effectively used in enrichment and extension programmes, or even in the regular classroom. You don't have to be a computer expert to enjoy learning these principles with your children. The book contains a range of activities, with background information explained simply. Answers to all problems are provided, and each activity ends with a 'what's it all about?' section that explains the relevance of the activities.

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.

This book was written by three Computer Science lecturers and two school teachers, and is based on our experience in classrooms. We have found that many important concepts can be taught without using a computer—in fact, sometimes the computer is just a distraction from learning. So unplug your computer, and get ready to learn what Computer Science is really about!

Acknowledgements

Many children and teachers have helped us to refine our ideas. The children and teachers at South Park School (Victoria, BC), Shirley Primary School, Ilam Primary School and Westburn Primary School (Christchurch, New Zealand) were guinea pigs for many activities. We are particularly grateful to Linda Picciotto, Karen Able, Bryon Porteous, Paul Cathro, Tracy Harrold, Simone Tanoa, Lorraine Woodfield, and Lynn Atkinson for welcoming us into their classrooms and making helpful suggestions for refinements to the activities. Gwenda Bensemann has trialed several of the activities for us and suggested modifications. Richard Lynders and Sumant Murugesh have helped with classroom trials. Parts of the cryptography activities were developed by Ken Noblitz. Some of the activities were run under the umbrella of the Victoria "Mathmania" group, with help from Kathy Beveridge. Earlier versions of the illustrations were done by Malcolm Robinson and Gail Williams, and we have also benefited from advice from Hans Knutson. Matt Powell has also provided valuable assistance during the development of the "Unplugged" project. We are grateful to the Brian Mason Scientific and Technical Trust for generous sponsorship in the early stages of the development of this book.

Special thanks go to Paul and Ruth Ellen Howard, who tested many of the activities and provided a number of helpful suggestions. Peter Henderson, Bruce McKenzie, Joan Mitchell, Nancy Walker-Mitchell, Gwen Stark, Tony Smith, Tim A. H. Bell[1], Mike Hallett, and Harold Thimbleby also provided numerous helpful comments.

We owe a huge debt to our families: Bruce, Fran, Grant, Judith, and Pam for their support, and Andrew, Anna, Hannah, Max, Michael, and Nikki who inspired much of this work,[2] and were often the first children to test an activity.

We are particularly grateful to Google Inc. for sponsoring the Unplugged project, and enabling us to make this edition available for free download.

We welcome comments and suggestions about the activities. The authors can be contacted via `www.unplugged.canterbury.ac.nz`.

[1] No relation to the first author.
[2] In fact, the text compression activity was invented by Michael.

Contents

Part I

Data: the raw material—
Representing information

Data: The Raw Material

How can we store information in computers?

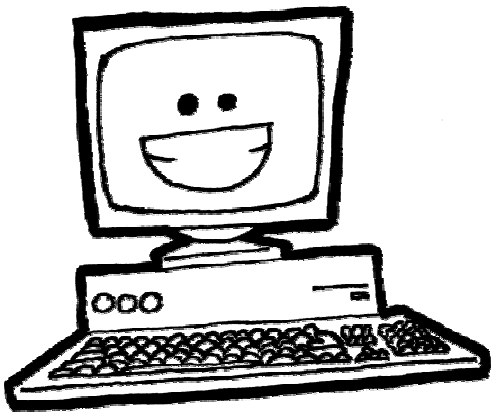
The word computer comes from the Latin *computare*, which means to calculate or add together, but computers today are more than just giant calculators. They can be a library, help us to write, find information for us, play music and even show movies. So how do they store all this information? Believe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

What is the difference between data and information?

Data is 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 zeros and ones?

In this section we will learn about binary numbers, how computers draw pictures, how fax machines work, what is the most efficient way to store lots of data, how we can prevent errors from happening and how we measure the amount of information we are trying to store.



Activity 1

Count the Dots—*Binary Numbers*

Summary

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

Curriculum Links

- ✓ Mathematics: Number Level 2 and up. Exploring numbers in other bases. Representing numbers in base two.
- ✓ Mathematics: Algebra Level 2 and up. Continue a sequential pattern, and describe a rule for this pattern. Patterns and relationships in powers of two.

Skills

- ✓ Counting
- ✓ Matching
- ✓ Sequencing

Ages

- ✓ 7 and up

Materials

- ✓ You will need to make a set of five binary cards (see page 6) for the demonstration. A4 cards with smiley face sticker dots work well.

Each child will need:

- ✓ A set of five cards.
  Copy Photocopy Master: Binary numbers (page 6) onto card and cut out.
- ✓ Worksheet Activity: Binary numbers (page 5)

There are optional extension activities, for which each child will need:

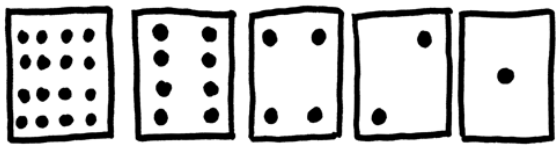
- ✓ Worksheet Activity: Working with binary (page 7)
- ✓ Worksheet Activity: Sending secret messages (page 8)
- ✓ Worksheet Activity: Fax machines and modems (page 9)
- ✓ Worksheet Activity: Counting higher than 31 (page 10)
- ✓ Worksheet Activity: More on binary numbers (page 11)

Binary Numbers

Introduction

Before giving out the worksheet on page 5, it can be helpful to demonstrate the principles to the whole group.

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 children to hold the demonstration cards at the front of the class. The cards should be in the following order:



Discussion

What do you notice about the number of dots on the cards? (Each card has twice as many as the card to its right.)

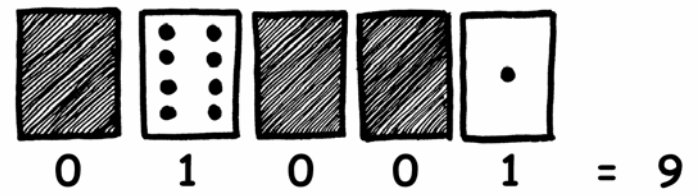
How many dots would the next card have if we carried on to the left? (32) The next…?

We can use these cards to make numbers by turning some of them face down and adding up the dots that are showing. Ask the children to make 6 (4-dot and 2-dot cards), then 15 (8-, 4-, 2- and 1-dot cards), then 21 (16, 4 and 1)…

Now try counting from zero onwards.

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.

When a binary number card is **not** showing, it is represented by a zero. When it **is** showing, it is represented by a one. This is the binary number system.



Ask the children to make 01001. What number is this in decimal? (9) What would 17 be in binary? (10001)

Try a few more until they understand the concept.

There are five optional follow-up extension activities, to be used for reinforcement. The children should do as many of them as they can.

Worksheet Activity: Binary Numbers

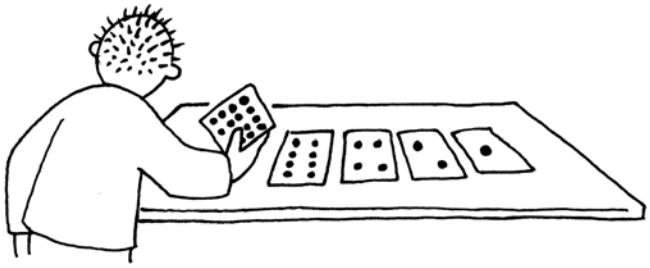
Learning how to count

So, you thought you knew how to count? Well, here is a new way to do it!

Did you know that computers use only zero and one? Everything that you see or hear on the computer—words, pictures, numbers, movies and even sound is stored using just those two numbers! These activities will teach you how to send secret messages to your friends using exactly the same method as a computer.

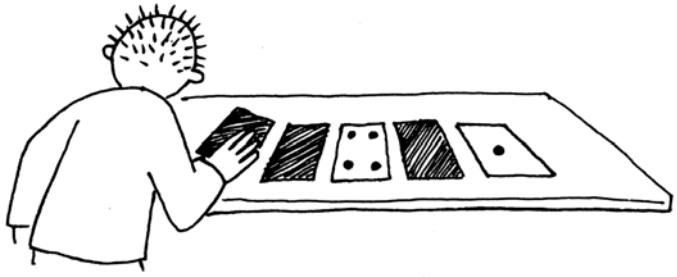
Instructions

Cut out the cards on your sheet and lay them out with the 16-dot card on the left as shown here:



Make sure the cards are placed in exactly the same order.

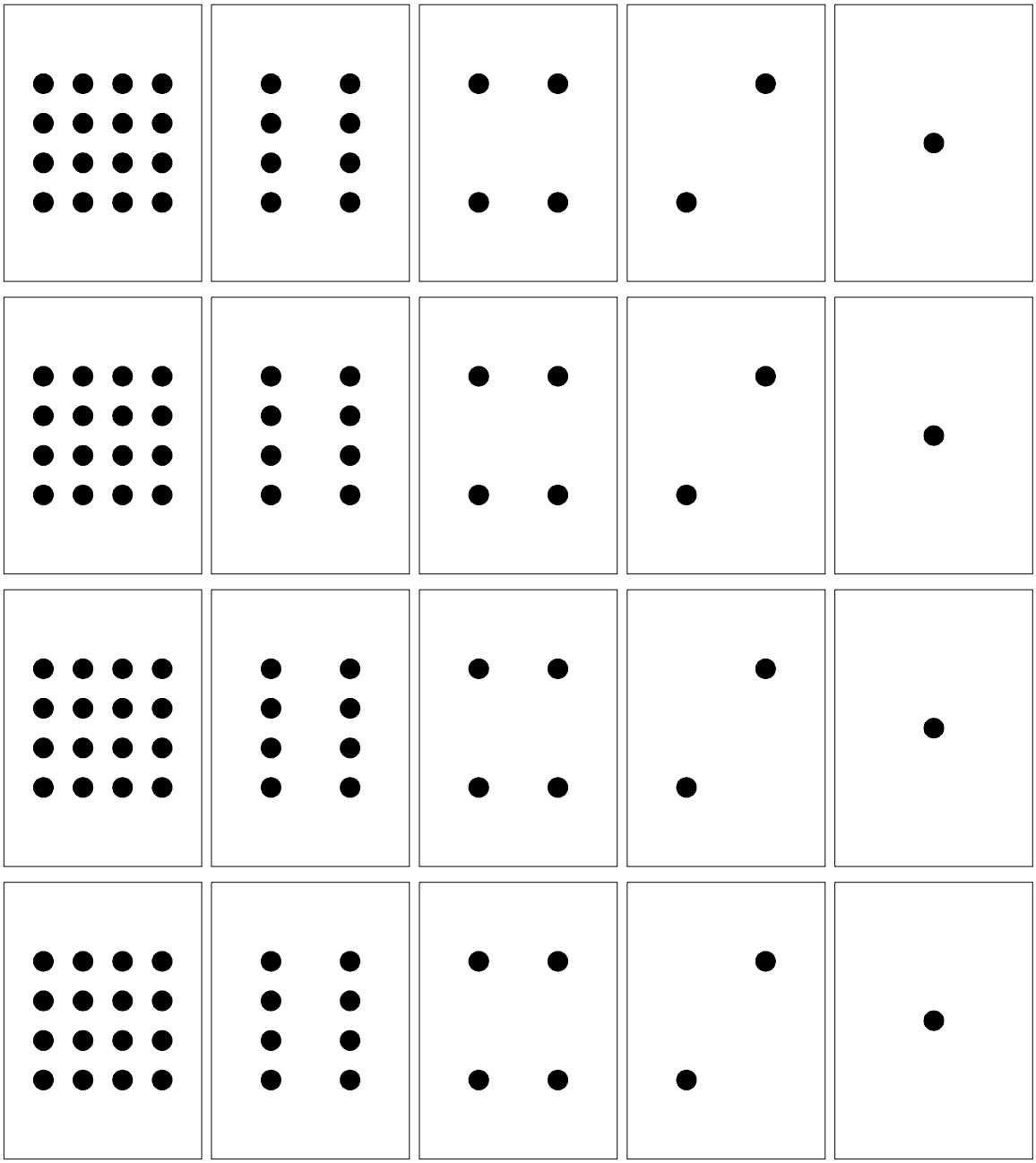
Now flip the cards so exactly 5 dots show—keep your cards in the same order!



Find out how to get 3, 12, 19. Is there more than one way to get any number? What is the biggest number you can make? What is the smallest? Is there any number you can't make between the smallest and biggest numbers?

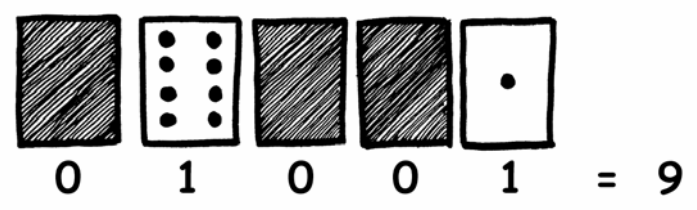
Extra for Experts: Try making the numbers 1, 2, 3, 4 in order. Can you work out a logical and reliable method of flipping the cards to increase any number by one?

Photocopy Master: Binary Numbers



Worksheet Activity: Working With Binary

The binary system uses **zero** and **one** to represent whether a card is face up or not. **0** shows that a card is hidden, and **1** means that you can see the dots. For example:



Can you work out what **10101** is? What about **11111**?

What day of the month were you born? Write it in binary. Find out what your friend's birthdays are in binary.

Try to work out these coded numbers:

☒☑☒☒☑ =
(☑=1, ☒=0)

👍👎👍👎 =
(👍=1, 👎=0)

⇧⇩⇧ =
(⇧=1, ⇩=0)

✚✚✖✚ =
(✚=1, ✖=0)

○○○○○ =
(⊙=1, ○=0)

↻↻↻↻↻ =
(↻=1, ↺=0)

📬📪 =
(📬=1, 📪=0)

▲▼▲▼▼ =
(▲=1, ▼=0)

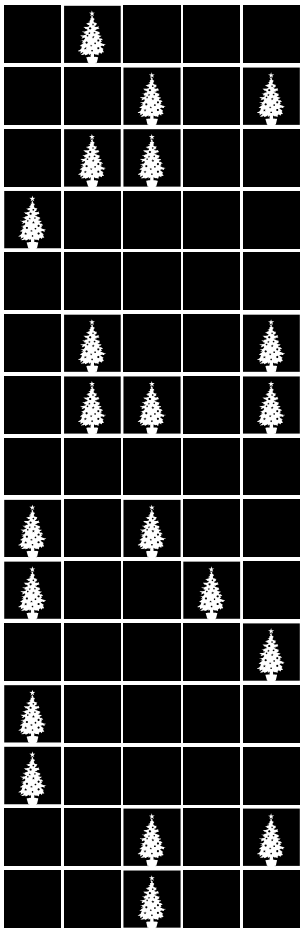
☹ =
(☺=1, ☹=0)

♠♠♠♠♠ =
(♠=1, ♣=0)

Extra for Experts: Using a set of rods of length 1, 2, 4, 8 and 16 units show how you can make any length up to 31 units. Or you could surprise an adult and show them how they only need a balance scale and a few weights to be able to weigh those heavy things like suitcases or boxes!

Worksheet Activity: Sending Secret Messages

Tom is trapped on the top floor of a department store. It's just before Christmas and he wants to get home with his presents. What can he do? He has tried calling, even yelling, but there is no one around. Across the street he can see some computer person still working away late into the night. How could he attract her attention? Tom looks around to see what he could use. Then he has a brilliant idea—he can use the Christmas tree lights to send her a message! He finds all the lights and plugs them in so he can turn them on and off. He uses a simple binary code, which he knows the woman across the street is sure to understand. Can you work it out?



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

Worksheet Activity: E-mail and Modems

Computers connected to the internet through a modem also use the binary system to send messages. 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 tones go very fast—so fast, in fact, that all we can hear is a horrible continuous screeching sound. If you have never heard it, listen to a modem connecting to the Internet, or try calling a fax machine—fax machines also use modems to send information.

Using the same code that Tom used in the department store, try sending an e-mail message to your friend. Make it easy for yourself and your friend though—you don't have to be as fast as a real modem!

Worksheet Activity: Counting higher than 31

Look at the binary cards again. If you were going to make the next card in the sequence, how many dots would it have? What about the next card after that? What is the rule that you are following to make your new cards? As you can see, only a few cards are needed to count up to very big numbers.

If you look at the sequence carefully, you can find a very interesting relationship:

1, 2, 4, 8, 16…

Try adding: 1 + 2 + 4 = ? What does it come to?

Now try 1 + 2 + 4 + 8 = ?

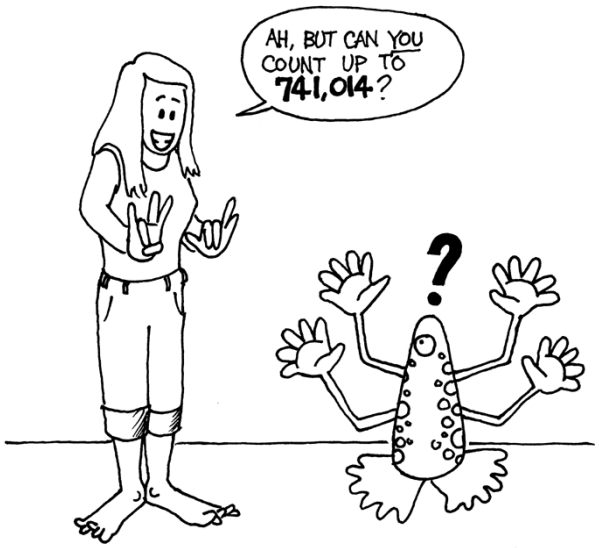
What happens if you add all the numbers up from the beginning?

Have you ever heard of "let your fingers do the walking"? Well now you can let your fingers do the counting, but you can get much higher than ten—no, you don't have to be an alien! If you use the binary system and let each finger on one hand represent one of the cards with dots you can count from 0–31. That's 32 numbers. (Don't forget that zero is a number too!)

Try counting in order using your fingers. If a finger is up it is a one, and if it is down it is a zero.

You can actually get from 0–1023 if you use both hands! That's 1024 numbers!

If you had really bendy toes (now you would have to be an alien) you could get even higher. If one hand can be used to count 32 numbers, and two hands can count to 32 × 32 = 1024 numbers, what is the biggest number Miss Flexi-Toes can reach?



Worksheet Activity: More on Binary Numbers

1. 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.

   But what happens when you put a 0 on the right of a binary number? Try this:

   1001 $\rightarrow$ **10010**

   (9) (?)

   Make up some others to test your hypothesis. What is the rule? Why do you think this happens?

2. Each of the cards we have used so far represents a 'bit' on the computer ('bit' is short for '**b**inary dig**it**'). So our alphabet code we have used so far can be represented using just five cards, or 'bits'. However a computer has to know whether letters are capitals or not, and also recognise digits, punctuation and special symbols such as **$** or **~**.

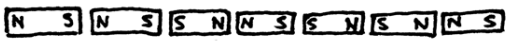
   Go and look at a keyboard and work out how many characters a computer has to represent. So how many bits does a computer need to store all the characters?

   Most computers today use a representation called ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange), which is based on using this number of bits per character, but some non-English speaking countries have to use longer codes.

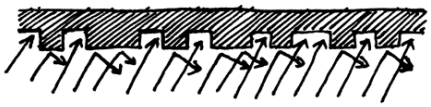
What's it all about?

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 two (humans normally use base 10). Each zero or one is called a *bit* (**b**inary 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 ones and zeros. On magnetic disks (floppy disks and hard disks) and tapes, bits are represented by the direction of a magnetic field on a coated surface, either North-South or South-North.



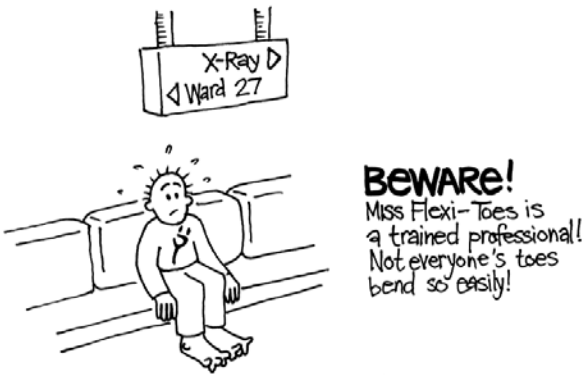
Audio CDs, CD-ROMs and DVDs store bits optically—the part of the surface corresponding to a bit either does or does not reflect light.



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

The speed of a computer depends on the number of bits it can process at once. For example, a 32-bit computer can process 32-bit numbers in one operation, while a 16-bit computer must break 32-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.



Solutions and hints

Binary Numbers (page 5)

3 requires cards 2 and 1
12 requires cards 8 and 4
19 requires cards 16, 2 and 1

There is only one way to make any number.

The biggest number you can make is 31. The smallest is 0. You can make every number in between, and each has a unique representation.

Experts: To increase any number by one, flip all the cards from right to left until you turn one face up.

Working with binary (page 7)

10101 = 21, 11111 = 31

Sending Secret Messages (page 8)

Coded message: HELP IM TRAPPED

Counting higher than 31 (page 10)

If you add the numbers up from the beginning the sum will always be one less than the next number in the sequence.

Miss Flexi-toes can count 1024 × 1024 = 1,048,576 numbers—from 0 to 1,048,575!

More on Binary Numbers (page 11)

When you put a zero on the right hand side of a binary number the number doubles.

All of the places containing a one are now worth twice their previous value, and so the total number doubles. (In base 10 adding a zero to the right multiplies it by 10.)

A computer needs 7 bits to store all the characters. This allows for up to 128 characters. Usually the 7 bits are stored in an 8-bit byte, with one bit wasted.

Activity 2

Colour by Numbers—*Image Representation*

Summary

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

Curriculum Links

- ✓ Mathematics: Geometry Level 2 and up. Exploring Shape and Space.

Skills

- ✓ Counting
- ✓ Graphing

Ages

- ✓ 7 and up

Materials

- ✓ OHP transparency made from OHP Master: Colour by numbers (page 16)

Each child will need:

- ✓ Worksheet Activity: Kid Fax (page 17)
- ✓ Worksheet Activity: Make your own picture (page 18)

Colour by Numbers

Introduction

Discussion Questions

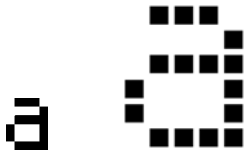
1. What do facsimile (fax) machines do?

2. In what situations would computers need to store pictures? (A drawing program, a game with graphics, or a multi-media system.)

3. How can computers store pictures when they can only use numbers?

(You may like to arrange for the children to send and/or receive faxes as a preparation for this activity)

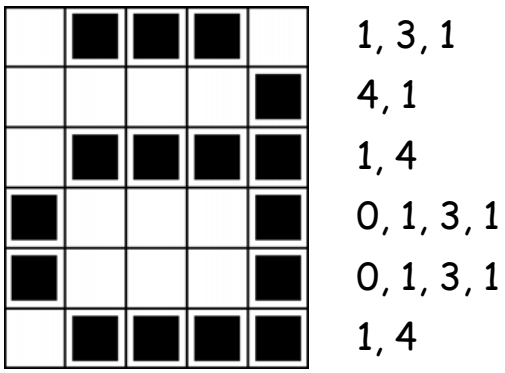
Demonstration using OHP transparency



Computer screens are divided up into a grid of small dots called *pixels* (**pic**ture **el**ements).

In a black and white picture, each pixel is either black or white.

The letter “a” has been magnified above 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.

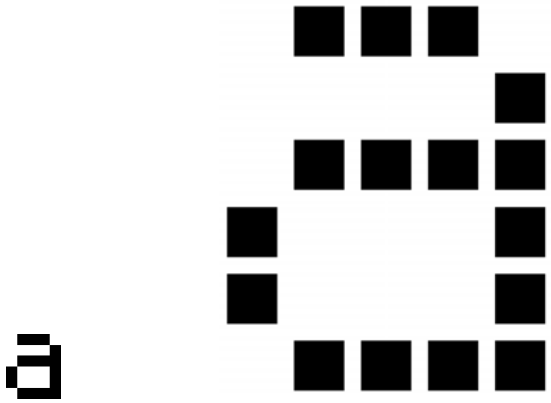


The picture above shows us how a picture can be represented by numbers. The first line consists of one white pixel, then three black, then one white. Thus the first line is represented as 1, 3, 1.

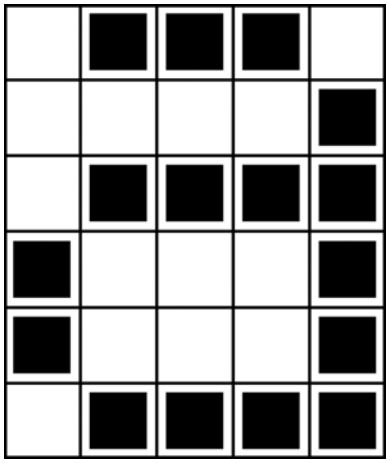
The first number always relates to the number of white pixels. If the first pixel is black the line will begin with a zero.

The worksheet on page 17 gives some pictures that the children can decode using the method just demonstrated.

OHP Master: Colour by numbers

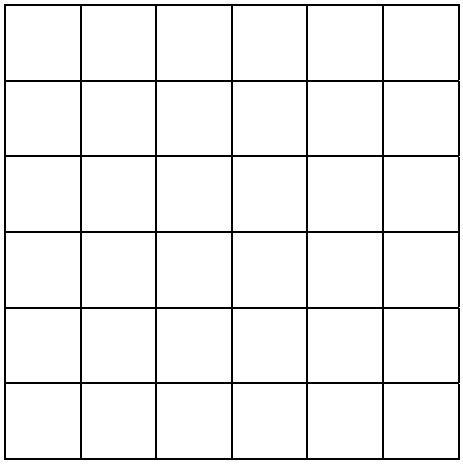


▲ A letter “a” from a computer screen and a magnified view showing the pixels that make up the image



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

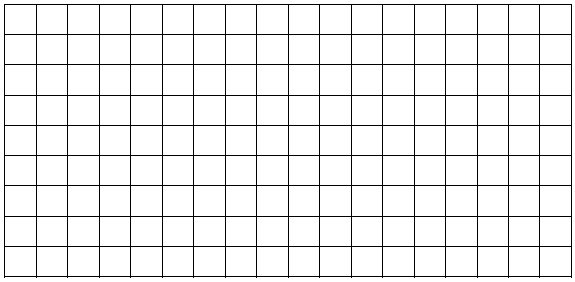
▲ The same image coded using numbers



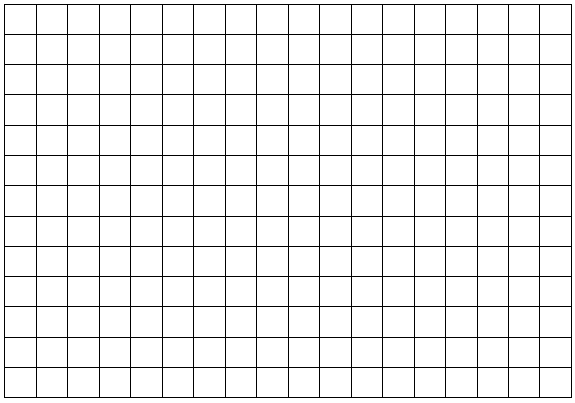
▲ Blank grid (for teaching purposes)

Worksheet Activity: Kid Fax

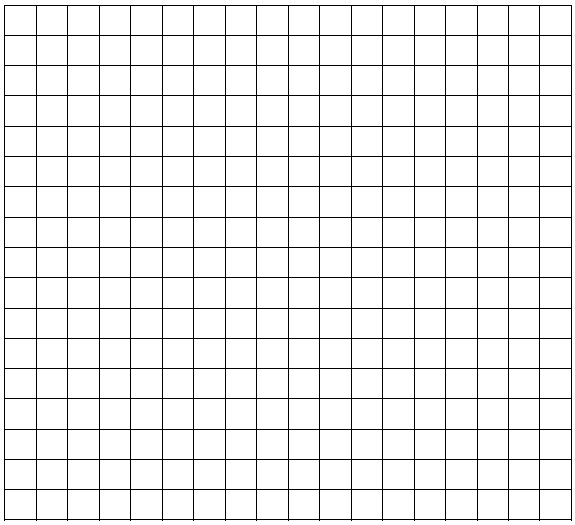
The first picture is the easiest and the last one is the most complex. It is easy to make mistakes and therefore a good idea to use a pencil to colour with and have a rubber handy!



4, 11
4, 9, 2, 1
4, 9, 2, 1
4, 11
4, 9
4, 9
5, 7
0, 17
1, 15



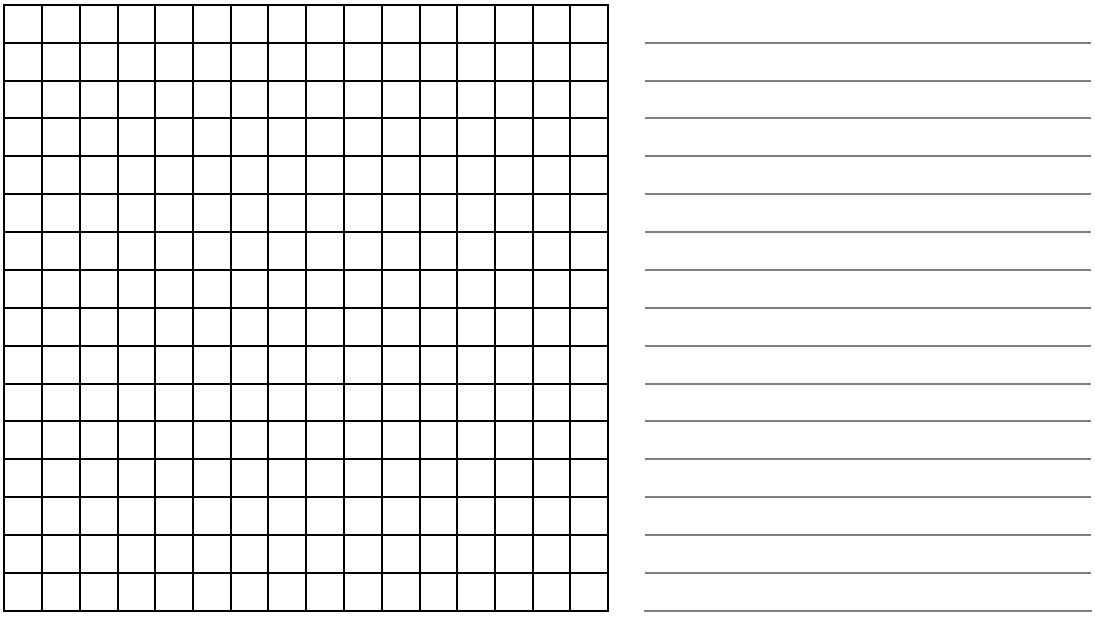
6, 5, 2, 3
4, 2, 5, 2, 3, 1
3, 1, 9, 1, 2, 1
3, 1, 9, 1, 1, 1
2, 1, 11, 1
2, 1, 10, 2
2, 1, 9, 1, 1, 1
2, 1, 8, 1, 2, 1
2, 1, 7, 1, 3, 1
1, 1, 1, 1, 4, 2, 3, 1
0, 1, 2, 1, 2, 2, 5, 1
0, 1, 3, 2, 5, 2
1, 3, 2, 5

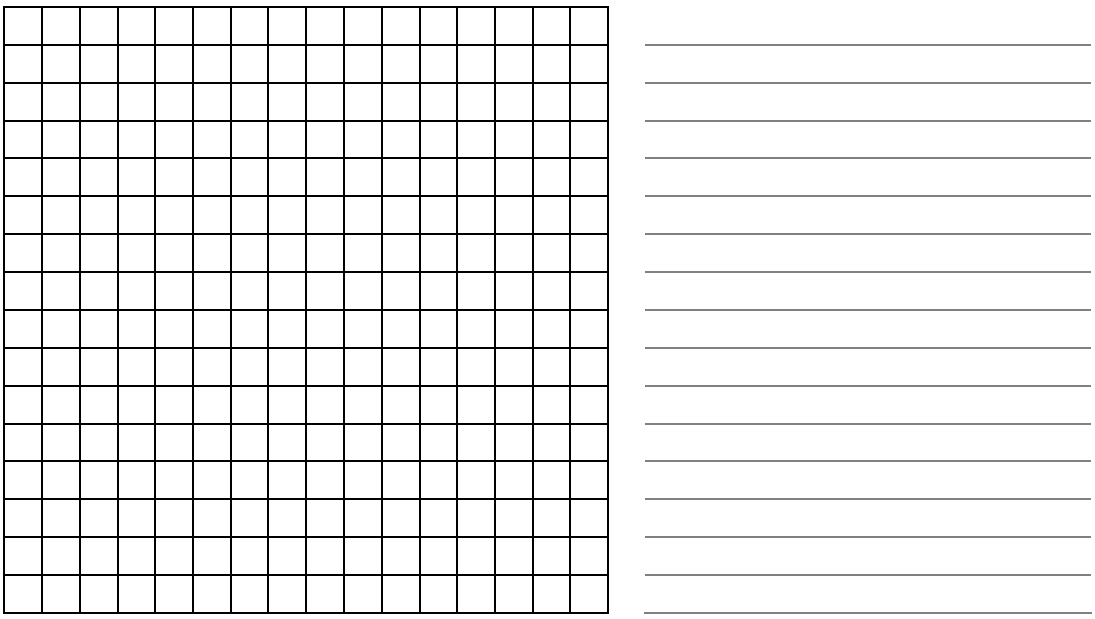


6, 2, 2, 2
5, 1, 2, 2, 2, 1
6, 6
4, 2, 6, 2
3, 1, 10, 1
2, 1, 12, 1
2, 1, 3, 1, 4, 1, 3, 1
1, 2, 12, 2
0, 1, 16, 1
0, 1, 6, 1, 2, 1, 6, 1
0, 1, 7, 2, 7, 1
1, 1, 14, 1
2, 1, 12, 1
2, 1, 5, 2, 5, 1
3, 1, 10, 1
4, 2, 6, 2
6, 6

Worksheet Activity: Make Your Own Picture

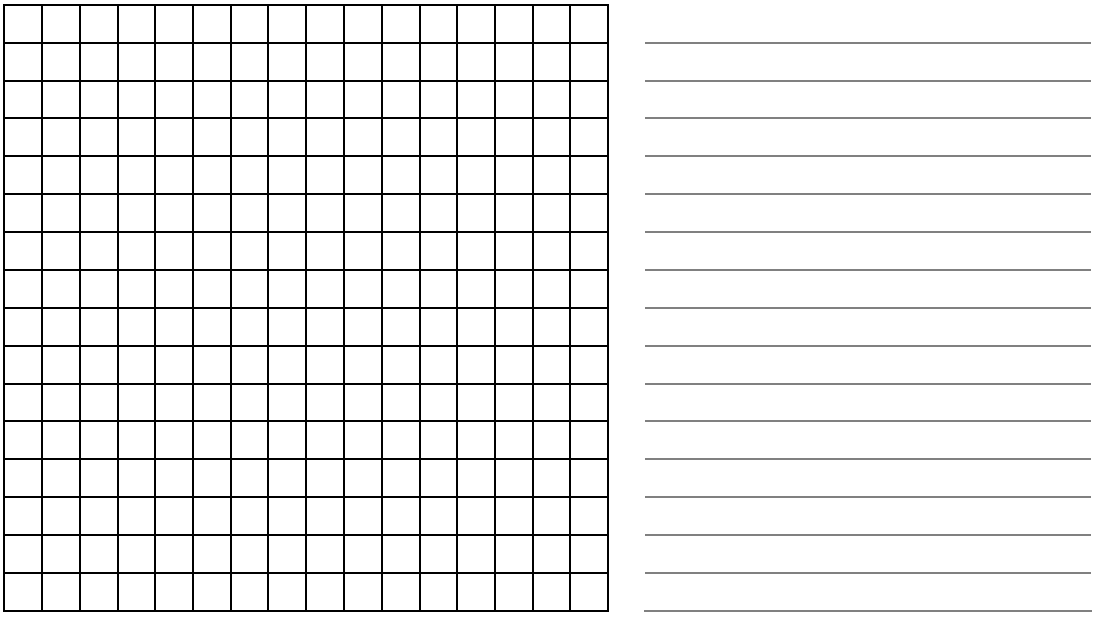
Now that you know how numbers can represent pictures, why not try making your own coded picture for a friend? Draw your picture on the top grid, and when you've finished, write the code numbers beside the bottom grid. Cut along the dotted line and give the bottom grid to a friend to colour in. (Note: you don't have to use the whole grid if you don't want to—just leave some blank lines at the bottom if your picture doesn't take up the whole grid.)

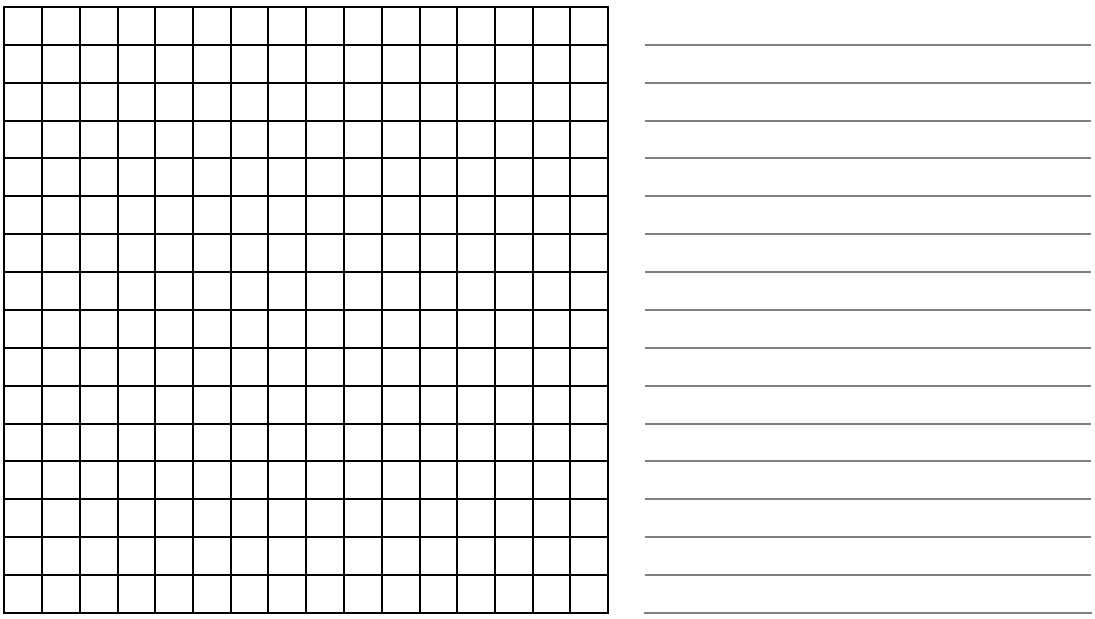




Worksheet Activity: Make Your Own Picture

Extra for Experts: If you want to produce coloured images you can use a number to represent the colour (e.g. 0 is black, 1 is red, 2 is green etc.) Two numbers are now used to represent a run of pixels: the first gives the length of the run as before, and the second specifies the colour. Try making a coloured picture for a friend. Don't forget to let your friend know which number stands for which colour!





Variations and Extensions

1. Try drawing with a sheet of tracing paper on top of the grid, so that the final image can be viewed without the grid. The image will be clearer.

2. Instead of colouring the grid the children could use squares of sticky paper, or place objects, on a larger grid.

Discussion Point

There is usually a limit to the length of a run of pixels because the length is being represented as a binary number. How would you represent a run of twelve black pixels if you could only use numbers up to seven? (A good way is to code a run of seven black pixels, followed by a run of zero white, then a run of five black.)

What's it all about?

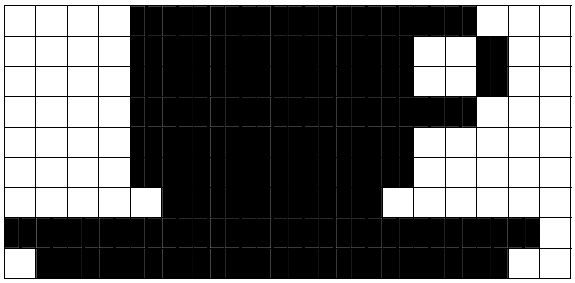
A fax machine is really just a simple computer that scans a black and white page into about 1000 × 2000 pixels, which are sent using a modem to another fax machine, which prints the pixels out on a page. Often fax images have large blocks of white (e.g. margins) or black pixels (e.g. a horizontal line). Colour pictures also have a lot of repetition in them. To save on the amount of storage space needed to keep such images programmers can use a variety of compression techniques. The method used in this activity is called 'run-length coding', and is an effective way to compress images. If we didn't compress images it would take much longer to transmit pictures and require much more storage space. This would make it infeasible to send faxes or put photos on a web page. For example, fax images are generally compressed to about a seventh of their original size. Without compression they would take seven times as long to transmit!

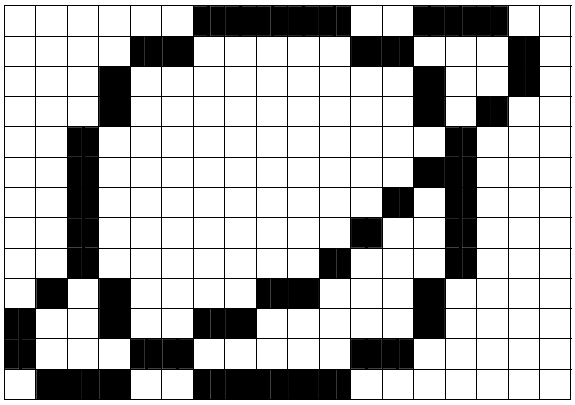
Photographs and pictures are often compressed to a tenth or even a hundredth of their original size (using a different technique). This allows many more images to be stored on a disk, and it means that viewing them over the web will take a fraction of the time.

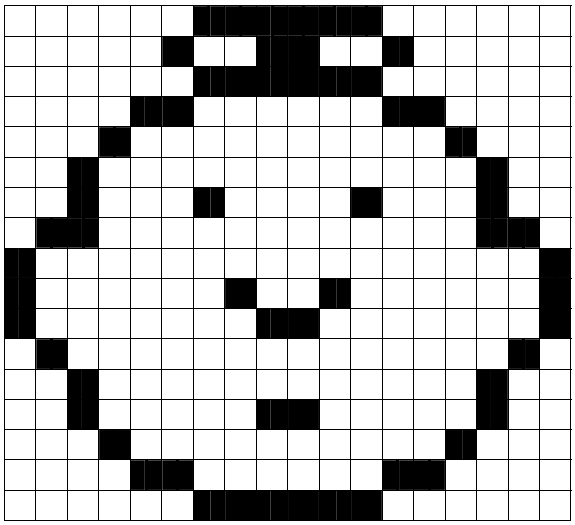
A programmer can choose which compression technique best suits the images he or she is transmitting.

Solutions and hints

Answers to Kid Fax Worksheet







Activity 3

You Can Say That Again! —*Text Compression*

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 coding 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.

Curriculum Links

- ✓ English: Recognising patterns in words and text.
- ✓ Technology: Technological knowledge and understanding. How computers work.

Skills

- ✓ Copying written text

Ages

- ✓ 9 and up

Materials

- ✓ OHP transparency made from OHP Master: You can say that again! (page 25)

Each child will need:

- ✓ Worksheet Activity: You can say that again! (page 26)
- ✓ Worksheet Activity: Extras for experts (page 27)
- ✓ Worksheet Activity: Short and sweet (page 28)
- ✓ Worksheet Activity: Extras for *real* experts (page 29)

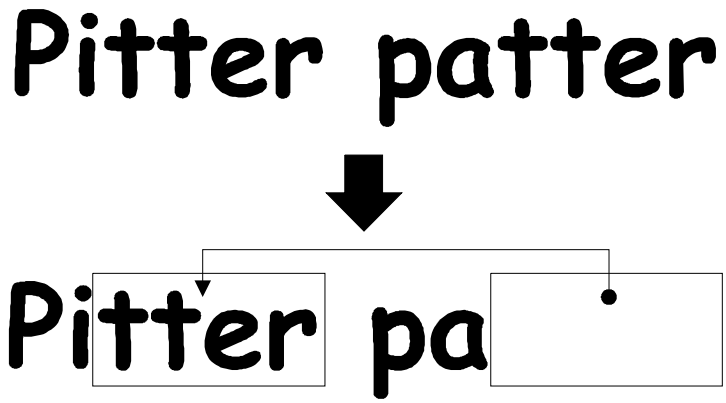
You can say that again!

Introduction

Computers have to store and transmit a lot of data. So that they don't have to use up too much storage space, or take too long to send information through a modem connection, they compress the text a bit like this.

Demonstration and Discussion

Show "The Rain" OHP (page 25). Look for the patterns of letters in this poem. Can you find groups of 2 or more letters that are repeated, or even whole words or phrases? (Replace these with boxes as shown in the diagram below.)



The Rain

Pitter patter

Pitter patter

Listen to the rain

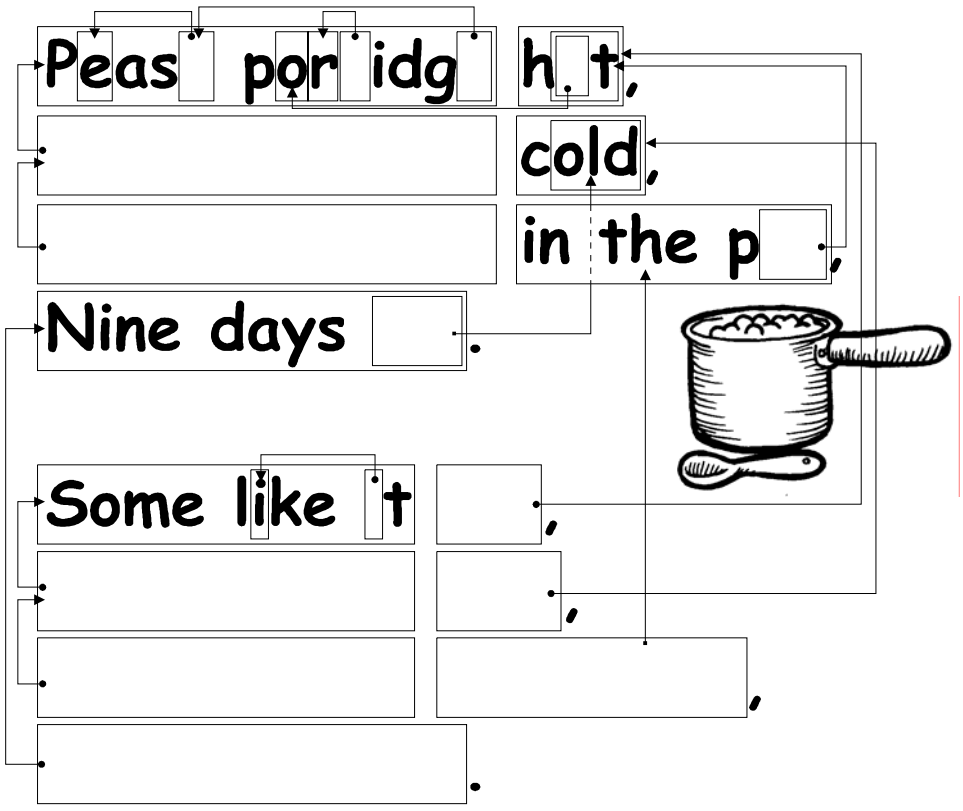
Pitter patter

Pitter patter

On the window pane

Worksheet Activity: You can say that again!

Many of the words and letters are missing in this poem. Can you fill in the missing letters and words to complete it correctly? You will find these in the box that the arrow is pointing to.



Now choose a simple poem or nursery rhyme and design your own puzzle. Make sure your arrows always point to an earlier part of the text. Your poem should be able to be decoded from left to right and from top to bottom in the same way we read.

Challenge: See how few of the original words you need to keep!

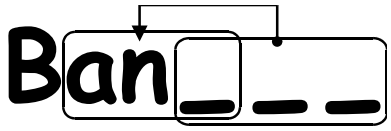
Here are some suggestions: Three Blind Mice, Mary Mary Quite Contrary, Hickory Dickory Dock—or try some Dr Seuss books!

Hint: Try to avoid overcrowding of arrows. Leave a lot of space around letters and words as you write them so that you have room for the boxes within boxes and the arrows pointing to them.

It is easier to design the puzzle if you write out the poem first and then decide where the boxes need to be.

Worksheet Activity: Extra for Experts

How would you solve this puzzle?



Sometimes missing text points to part of itself. In this case it can be decoded correctly if the letters are copied from left to right. Then each letter is available to be copied before it is needed. This is useful in computers if there is a long run of a particular character or pattern.

Try drawing some of your own.

On computers the boxes and arrows are represented by numbers. For example,

Banana

can be written as **Ban(2,3)**. "**2**" means count back two characters to find the starting point for copying,

Ban---

and "**3**" means copy three consecutive characters:

Bana--

Banan-

Banana

As two numbers are used to code these words, usually only groups of two or more letters are worth compressing, otherwise there is no saving of space. In fact the size of the file could go up if two numbers are used to code one letter.

Make up some words of your own written in the way a computer would if they were compressed. Can your friends decode them?

Worksheet Activity: Short and Sweet

How many words do you really need here?

Pretend you are a computer trying to fit as much into your disk as possible. Cross out all the groups of two or more letters that have already occurred. These are no longer needed as they could be replaced by a pointer. Your goal is to get as many letters crossed out as possible.

I know an old lady who swallowed a bird
How absurd! She swallowed a bird!
She swallowed the bird to catch the spider
That wriggled and jiggled
and tickled inside her
She swallowed the spider to catch the fly
I don't know why she swallowed a fly
Perhaps she'll die…

Worksheet Activity: Extra for *Real* Experts

Ready for some *really* tough compression?

The following story was run through a computer program, which found that there are at least 1,633 letters that can be crossed out. How many can you find? Remember, only groups of two or more repeated characters can be eliminated. Good luck!

Once upon a time, long, long ago, three little pigs set out to make their fortunes. The first little pig wasn't very clever, and decided to build his house out of straw, because it was cheap. The second little pig wasn't very clever either, and decided to build his house out of sticks, for the "natural" look that was so very much in fashion, even in those days. The third little pig was much smarter than his two brothers, and bought a load of bricks in a nearby town, with which to construct a sturdy but comfortable country home.

Not long after his housewarming party, the first little pig was curled up in a chair reading a book, when there came a knock at the door. It was the big bad wolf, naturally.

"Little pig, little pig, let me come in!" cried the wolf.

"Not by the hair on my chinny-chin-chin!" squealed the first little pig.

"Then I'll huff, and I'll puff, and I'll blow your house down!" roared the wolf, and he *did* huff, and he *did* puff, and the house soon collapsed. The first little pig ran as fast as he could to the house of sticks, and was soon safe inside. But it wasn't long before the wolf came calling again.

"Little pig, little pig, let me come in!" cried the wolf.

"Not by the hair on my chinny-chin-chin!" squealed the second little pig.

"Then I'll huff, and I'll puff, and I'll blow your house down!" roared the wolf, and he *did* huff, and he *did* puff, and the house was soon so much firewood. The two terrified little pigs ran all the way to their brother's brick house, but the wolf was hot on their heels, and soon he was on the doorstep.

"Little pig, little pig, let me come in!" cried the wolf.

"Not by the hair on my chinny-chin-chin!" squealed the third little pig.

"Then I'll huff, and I'll puff, and I'll blow your house down!" roared the wolf, and he huffed, and he puffed, and he huffed some more, but of course, the house was built of brick, and the wolf was soon out of breath. Then he had an idea. The chimney! He clambered up a handy oak tree onto the roof, only to find that there *was* no chimney, because the third little pig, being conscious of the environment, had installed electric heating. In his frustration, the wolf slipped and fell off the roof, breaking his left leg, and severely injuring his pride. As he limped away, the pigs laughed, and remarked how much more sensible it was to live in the city, where the only wolves were in the zoo. And so that is what they did, and of course they all lived happily ever after.

What's it all about?

The storage capacity of computers is growing at an unbelievable rate—in the last 25 years, the amount of storage provided on a typical computer has grown about a millionfold—but we still find more to put into our computers. Computers can store whole books or even libraries, and now music and movies too, if only they have the room. Large files are also a problem on the Internet, because they take a long time to download. We also try to make computers smaller—even a cellphone or wristwatch can be expected to store lots of information!

There is a solution to this problem, however. Instead of buying more storage space, or a faster modem, we can *compress* the data so that it takes up less space. 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. The method used in this activity, with the principle of pointing to earlier occurrences of chunks of text, is often referred to as 'Ziv-Lempel coding,' or 'LZ coding', invented by two Israeli professors in the 1970s. It can be used for any language and 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. In the case of modems, it reduces the amount of data that needs to be transmitted over the phone line, so it goes much faster.

Some other methods are based on the idea that letters that are used more often should have shorter codes than the others. Morse code used this idea.

Solutions and hints

You can say that again! (page 26)

Pease porridge hot,
Pease porridge cold,
Pease porridge in the pot,
Nine days old.

Some like it hot,
Some like it cold,
Some like it in the pot,
Nine days old.

Activity 4

Card Flip Magic—*Error Detection & Correction*

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.

Curriculum Links

- ✓ Mathematics: Number Level 3 and up. Exploring computation and estimation.
- ✓ Algebra Level 3 and up. Exploring patterns and relationships.

Skills

- ✓ Counting
- ✓ Recognition of odd and even numbers

Ages

- ✓ 9 years and up

Materials

- ✓ A set of 36 "fridge magnet" cards, coloured on one side only
- ✓ A metal board (a whiteboard works well) for the demonstration.

Each pair of children will need:

- ✓ 36 identical cards, coloured on one side only.

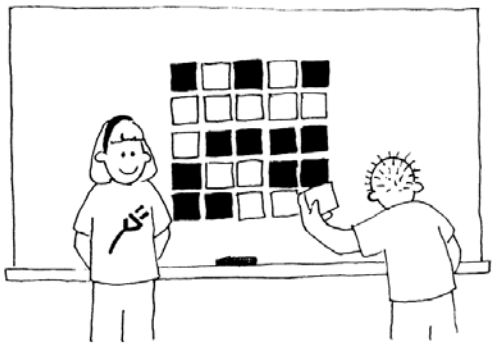
The "Magic Trick"

Demonstration

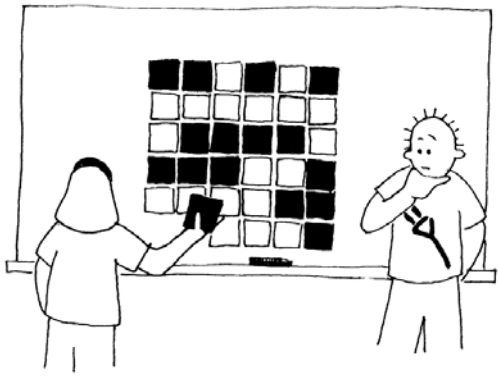
Here's your chance to be a magician!

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

1. Choose a child to lay out the cards in a 5 × 5 square, 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 coloured cards in each row and column.

2. Get a child to flip over one card only while you cover your eyes. The row and column containing the changed card will now have an odd number of coloured cards, and this will identify the changed card.

Can the children guess how the trick is done?

Teach the trick to the children:

1. Working in pairs, the children lay out their cards 5 × 5.

2. How many coloured cards are there in each row and column? Is it an odd or even number? Remember, 0 is an even number.

3. Now add a sixth card to each row, making sure the number of coloured cards is always even. This extra card is called a "parity" card.

4. Add a sixth row of cards along the bottom, to make the number of cards in each column an even number.

5. Now flip a card. What do you notice about the row and column? (They will have an odd number of coloured cards.) Parity cards are used to show you when a mistake has been made.

6. Now take turns to perform the 'trick'.

Extension Activities:

1. Try using other objects. Anything that has two 'states' is suitable. For example, you could use playing cards, coins (heads or tails) or cards with 0 or 1 printed on them (to relate to the binary system).

2. What happens if two, or more, cards are flipped? (It is not always possible to know exactly which two cards were flipped, although it is possible to tell that something has been changed. You can usually narrow it down to one of two pairs of cards. With 4 flips it is possible that all the parity bits will be correct afterwards, and so the error could go undetected.)

3. Another interesting exercise is to consider the lower right-hand card. If you choose it to be the correct one for the column above, then will it be correct for the row to its left? (The answer is yes, always.)

4. In this card exercise we have used even parity—using an even number of coloured cards. Can we do it with odd parity? (This is possible, but the lower right-hand card only works out the same for its row and column if the numbers of rows and columns are both even or both odd. For example, a 5 × 9 layout will work fine, or a 4 × 6, but a 3 × 4 layout won't.)

A Real-Life Example for Experts!

This same checking technique is used with book codes. Published books have a ten-digit code 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 (International Standard Book Number), the publisher can check that you haven't made a mistake. They simply look at the checksum. That way you don't end up waiting for 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. Each of these values is then added together.

For example, the ISBN 0-13-911991-4 gives a value

$$\begin{aligned} & (0 \times 10) + (1 \times 9) + (3 \times 8) + (9 \times 7) + (1 \times 6) \\ + \; & (1 \times 5) + (9 \times 4) + (9 \times 3) + (1 \times 2) \\ = \; & 172 \end{aligned}$$

Then divide your answer by eleven. What is the remainder?

$$172 \div 11 = 15 \text{ remainder } 7$$

If the remainder is zero, then the checksum is zero, otherwise subtract the remainder from 11 to get the checksum.

$$11 - 7 = 4$$

Look back. Is this the last digit of the ISBN? Yes!

If the last digit of the ISBN wasn't a four, then we would know that a mistake had been made.

It is possible to come up with a checksum of the value of 10, which would require more than one digit. When this happens, the character X is used.

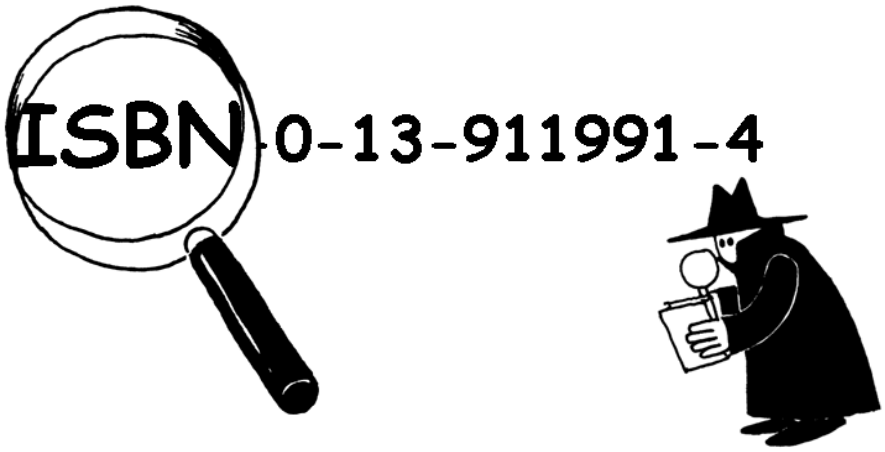
▲ A barcode (UPC) from a box of Weet-Bix™

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.

Check that book!

Detective Blockbuster

Book Tracking Service, Inc.



We find and check ISBN checksums for a small fee.

Join our agency—look in your classroom or library for real ISBN codes.

Are their checksums correct?

Sometimes errors are made.

Some of the 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

Can you find a book with the letter X for a checksum of 10? It shouldn't be too hard to find—one in every 11 should have it.

What sort of errors might occur that wouldn't be detected? Can you change a digit and still get the correct checksum? What if two digits are swapped (a common typing error)?

What's it all about?

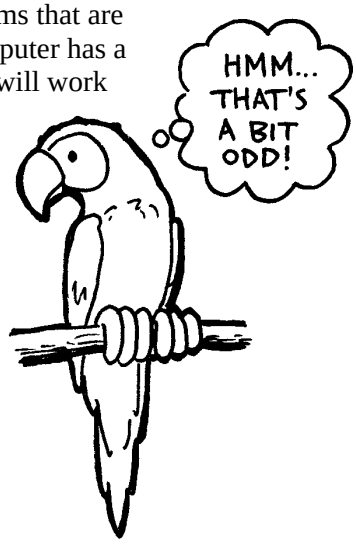
Imagine you are depositing $10 cash into your bank account. The teller types in the amount of the deposit, and it is sent to a central computer. But suppose some interference occurs on the line while the amount is being sent, and the code for $10 is changed to $1,000. No problem if you are the customer, 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, for example 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 if an error has occurred, but *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 to finish, a joke that is better appreciated after doing this activity:

Q: What do you call this: "Pieces of nine, pieces of nine"?

A: A parroty error.

Solutions and hints

Errors that would not be detected are those where one digit increases and another decreases. Then the sum might still be the same.

Activity 5

Twenty Guesses—*Information Theory*

Summary

How much information is there in a 1000-page book? Is there more information in a 1000-page telephone book, or in a ream of 1000 sheets of blank paper, or in Tolkien's *Lord of the Rings*? If we can measure this, we can estimate how much space is needed to store the information. For example, can you still read the following sentence?

Ths sntnce hs th vwls mssng.

You probably can, because there is not much 'information' in the vowels. This activity introduces a way of measuring information content.

Curriculum links

- ✓ Mathematics: Number Level 3 and up. Exploring number: Greater than, less than, ranges.
- ✓ Algebra Level 3 and up. Patterns and sequences
- ✓ English

Skills

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

Ages

- ✓ 10 and up

Materials

- ✓ No materials are required for the first activity

There is an extension activity, for which each child will need:

- ✓ Worksheet Activity: Decision trees (page 40)

Twenty Guesses

Discussion

1. Discuss with the children what they think information is.

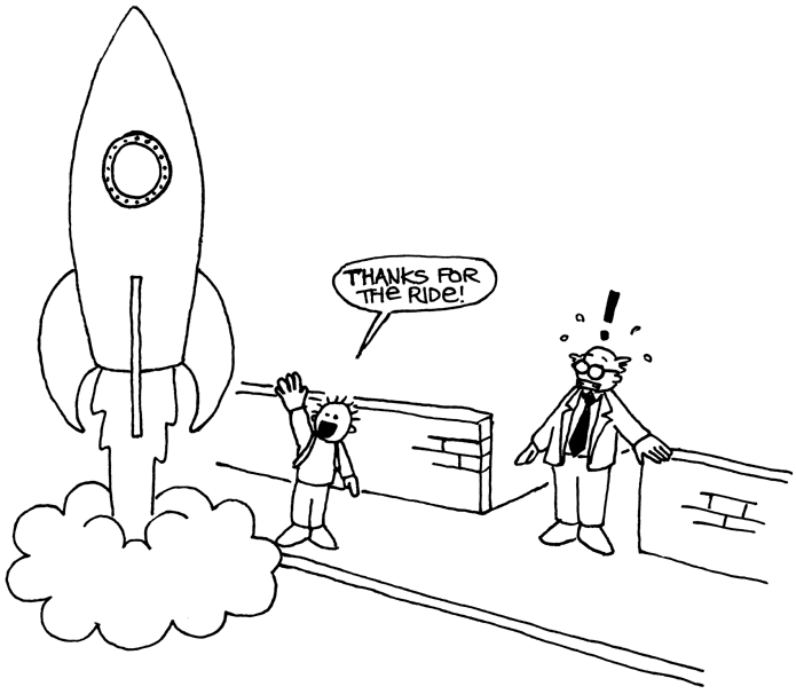
2. How could we measure how much information there would be in a book? Is the number of pages or number of words important? Can one book have more information than another? What if it is a very boring book, or a particularly interesting one? Would 400 pages of a book containing the phrase "blah, blah, blah" have more or less information than, say, the telephone directory?

   Explain that computer scientists measure information by how surprising a message (or book!) 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.

   How can the surprise value of a message be measured?

   One way 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.

   The amount of information that messages contain is measured by how easy or hard they are to guess. The following game gives us some idea of this.



Twenty Questions Activity

This is an adapted game of 20 questions. Children may ask questions of a chosen child, who may only answer yes or no until the answer has been guessed. Any question may be asked, provided that the answer is strictly 'yes' or 'no'.

Suggestions:

I am thinking of:

- ✓ a number between 1 and 100
- ✓ a number between 1 and 1000
- ✓ a number between 1 and 1,000,000.
- ✓ any whole number
- ✓ a sequence of 6 numbers in a pattern (appropriate to the group). Guess in order from first to last. (e.g. 2, 4, 6, 8, 10)

Count the number of questions that were asked. This is a measure of the value of the "information".

Follow-up Discussion

What strategies did you use? Which were the best ones?

Point out that it takes just 7 guesses to find a number between 1 and 100 if you halve the range each time. For example:

Is it less than 50?	Yes.
Is it less than 25?	No.
Is it less than 37?	No.
Is it less than 43?	Yes.
Is it less than 40?	No.
Is it less than 41?	No.
It must be 42!	Yes!

Interestingly if the range is increased 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.

A good follow up would be to let the children play Mastermind.

Extension: How much information is there in a message?

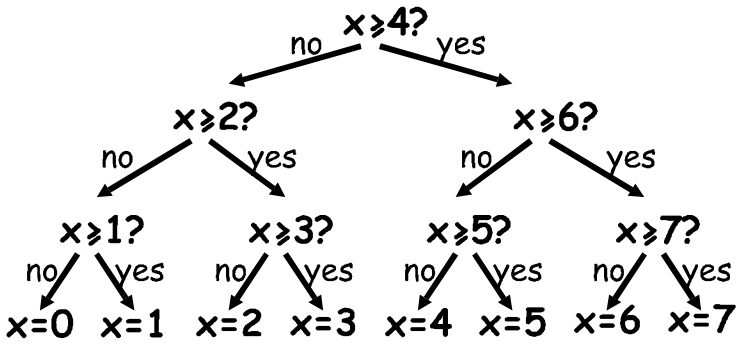
Computer scientists don't just use guessing with numbers—they can also guess which letter is more likely to be next in a word or sentence.

Try the guessing game with a short sentence of 4–6 words. The letters must be guessed in the correct order, from first to last. Get someone to write down the letters as they are found and keep a record of how many guesses it takes to find each letter. Any questions with a yes/no answer can be used. Examples would be, "It it a *t*?" "Is it a vowel?" "Does it come before *m* in the alphabet?" A space between words also counts as a "letter" and must be guessed. Take turns and see if you can discover which parts of messages are easiest to find out.

Worksheet Activity: Decision Trees

If you already know the strategy for asking the questions, you can transmit a message without having to ask anything.

Here is a chart called a 'decision tree' for guessing a number between 0 and 7:



What are the yes/no decisions needed to 'guess' the number 5?

How many yes/no decisions do you need to make to work out any number?

Now look at something very fascinating. Underneath the numbers 0, 1, 2, 3... in the final row of the tree write the number in binary (see Activity 1).

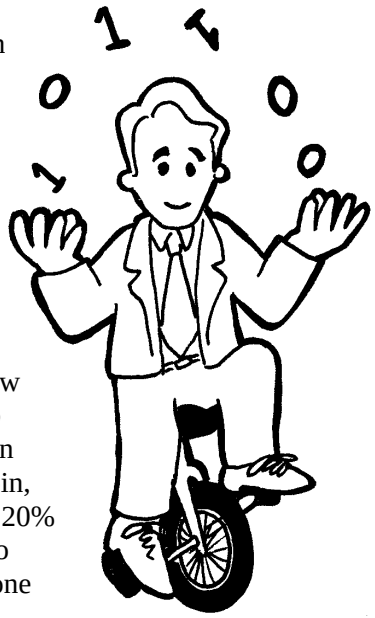
Look closely at the tree. If no=0 and yes=1, what do you see?

In the number guessing game we try to choose questions so that the sequence of answers works out to represent the number in exactly this way.

Design your own decision tree for guessing numbers between 0 and 15.

Extra for experts: What kind of tree would you use to guess someone's age? What about a tree to guess which letter is next in a sentence?

What's it all about?



A celebrated American mathematician (and juggler, and unicyclist) called Claude Shannon did a lot of experiments with this game. He measured the amount of information in bits—each yes/no answer is equivalent to a 1/0 bit. He found that the amount of "information" contained in a message depends on what you already know. Sometimes we can ask a question that eliminates the need to ask a lot of other questions. In this case the information content of the message is low. For example, the information in a single toss of a coin is normally one bit: heads or tails. But if the coin happens to be a biased one that turns up heads nine times out of ten, then the information is no longer one bit—believe it or not, it's less. How can you find out what a coin toss was with less than one yes/no question? Simple—just use questions like "are the next *two* coin tosses both heads?" For a sequence of tosses with the biased coin, the answer to this will be "yes" about 80%, of the time. On the 20% of occasions where the answer is "no," you will have to ask two further questions. But on average you will be asking less than one question per coin toss!

Shannon called the information content of a message "entropy". Entropy depends not only on the *number* of possible outcomes—in the case of a coin toss, two—but also on the *probability* of it happening. Improbable events, or surprising information, need a lot more questions to guess the message because they tell us more information we didn't already know—just like the situation of taking a helicopter to school.

The entropy of a message is very important to computer scientists. You cannot compress a message to occupy less space than its entropy, and the best compression systems are equivalent to a guessing game. Since a computer program is making the 'guesses', the list of questions can be reproduced later, so as long as the answers (bits) are stored, we can reconstruct the information! The best compression systems can reduce text files to about a quarter of their original size—a big saving on storage space!

The guessing method can also be used to build a computer interface that predicts what the user is going to type next! This can be very useful for physically disabled people who find it difficult to type. The computer suggests what it thinks they are likely to type next, and they just indicate what they want. A good system needs an average of only two yes/no answers per character, and can be of great assistance to someone who has difficulty making the fine movements needed to control a mouse or keyboard. This sort of system is also used in a different form to 'type' text on some cellphones.

Solutions and hints

The answer to a single yes/no question corresponds to exactly one bit of information—whether it is a simple question like “Is it more than 50?” or a more complex one like “Is it between 20 and 60?”

In the number-guessing game, if the questions are chosen in a certain way, the sequence of answers is just the binary representation of the number. Three is 011 in binary and is represented by the answers “No, yes, yes” in the decision tree, which is the same if we write no for 0 and yes for 1.

A tree you would use for someone’s age might be biased towards smaller numbers.

The decision about the letters in a sentence might depend upon what the previous letter was.

Part II

Putting Computers to Work—*Algorithms*

Putting Computers to Work

Computers operate by following a list of instructions set for them. These instructions enable them to sort, find and send information. To do these things 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 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.

The word “algorithm” is derived from the name of Mohammed ibn Musa Al-Khowarizmi—Mohammed, son of Moses, from Khowarizm—who joined an academic centre known as the House of Wisdom in Baghdad around 800AD. His works transmitted the Hindu art of reckoning to the Arabs, and thence to Europe. When they were translated into Latin in 1120AD, the first words were “Dixit Algorismi”—“thus said Algorismi”.

Activity 6

Battleships—*Searching Algorithms*

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.

Curriculum Links

- ✓ Mathematics: Number Level 3 and up. Exploring numbers: Greater than, less than and equal to
- ✓ Geometry Level 3 and up. Exploring shape and space: Co-ordinates

Skills

- ✓ Logical reasoning

Ages

- ✓ 9 years and up

Materials

Each child will need:

- ✓ Copy of battleships games
  - 1A, 1B for game 1
  - 2A, 2B for game 2
  - 3A, 3B for game 3

- ✓ You may also need a few copies of the supplementary game sheets, 1A', 1B', 2A', 2B', 3A', 3B'.

Battleships

Introductory Activity

1. Choose about 15 children to line up at the front of the classroom. Give each child a card with a number on it (in random order). Keep the numbers hidden from the rest of the class.

2. Give another child a container with four or five sweets in it. Their job is to find a given number. They can "pay" to look at a particular card. If they find the correct number before using all their sweets, they get to keep the rest.

3. Repeat if you wish to.

4. Now shuffle the cards and give them out again. This time, have the children sort themselves into ascending order. The searching process is repeated.

   If the numbers are sorted, a sensible strategy is to use just one "payment" to eliminate half the children by having the middle child reveal their card. By repeating this process they should be able to find the number using only three sweets. The increased efficiency will be obvious.

Activity

The children can get a feel for how a computer searches by playing the battleship game. As they play the game, get them to think about the strategies they are using to locate the ships.

Battleships—A *Linear* Searching Game

Read the following instructions to the children

1. Organise yourselves into pairs. One of you has sheet 1A, the other sheet 1B. Don't show your sheet to your partner!

2. Both of you circle one battleship on the top line of your game sheet and tell your partner its number.

3. Now take turns to guess where your partner's ship is. (You say the letter name of a ship and your partner tells you the number of the ship at that letter.)

4. How many shots does it take to locate your partner's ship? This is your score for the game.

   (Sheets 1A' and 1B' are extras provided for children who would like to play more games or who "inadvertently" see their partner's sheet. Sheets 2A', 2B' and 3A', 3B' are for the later games.)

Follow Up Discussion

1. What were the scores?

2. What would be the minimum and maximum scores possible? (They are 1 and 26 respectively, assuming that the children don't shoot at the same ship twice. This method is called 'linear search', because it involves going through all the positions, one by one.)

Battleships—A *Binary* Searching Game

Instructions

The instructions for this version of the game are the same as for the previous game but the numbers on the ships are now in ascending order. Explain this to the children before they start.

1. Organise yourselves into pairs. One of you has sheet 2A, the other sheet 2B. **Don't** show your sheet to your partner!

2. Both of you circle one battleship on the top line of your game sheet and tell your partner its number.

3. Now take turns to guess where your partner's ship is. (You say the letter name of a ship and your partner tells you the number of the ship at that letter.)

4. How many shots does it take to locate your partner's ship? This is your score for the game.

Follow Up Discussion

1. What were the scores?

2. What strategy did the low scorers use?

3. Which ship should you choose first? (The one in the middle tells you which half of the line the chosen ship must be in.) Which location would you choose next? (Again the best strategy is always to choose the middle ship of the section that must have the selected ship.)

4. If this strategy is applied how many shots will it take to find a ship? (Five at most).

   This method is called 'binary search', because it divides the problem into two parts.

Battleships—A Search Game using *Hashing*

Instructions

1. Each take a sheet as in the previous games and tell your partner the number of your chosen ship.

2. In this game you can find out which column (0 to 9) the ship is in. You simply add together the digits of the ship's number. The last digit of the sum is the column 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 column 4. Once you know the column you need to guess which of the ships in that column is the desired one. This technique is called 'hashing', because the digits are being squashed up ("hashed") together.

3. Now play the game using this new searching strategy. You may like to play more than one game using the same sheet—just choose from different columns.

   (Note that, unlike the other games, the spare sheets 3A' and 3B' must be used as a pair, because the pattern of ships in columns must correspond.)

Follow Up Discussion

1. Collect and discuss scores as before.

2. Which ships are very quick to find? (The ones that are alone in their column.) Which ships may be harder to find? (The ones whose columns contain lots of other ships.)

3. Which of the three searching processes is fastest? Why?

   What are the advantages of each of the three different ways of searching? (The second strategy is faster than the first, but the first one doesn't require the ships to be sorted into order. The third strategy is usually faster than the other two, but it is possible, by chance, for it to be very slow. In the worst case, if all the ships end up in the same column, it is just as slow as the first strategy.)

Extension Activities

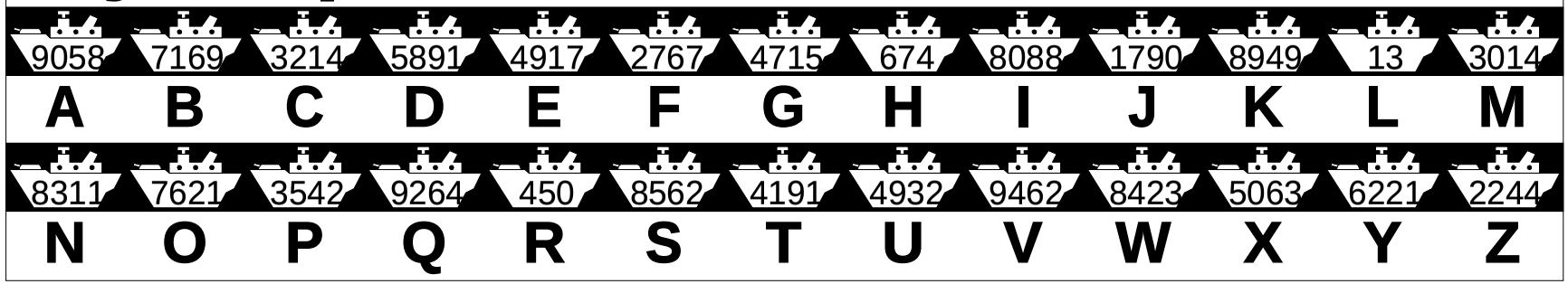
1. Have the children make up their own games using the three formats. For the second game they must put the numbers in ascending order. Ask how they might make the Hashing Game very hard. (The hardest game is when all the ships are in the same column.) How could you make it as easy as possible? (You should try to get the same number of ships into each column.)

2. What would happen if the ship being sought wasn't there? (In the Linear Search game it would take 26 shots to show this. In the Binary Search game you would need five shots to prove this. When using the Hash System it would depend on how many ships appeared in the relevant column.)

3. Using the Binary Search strategy how many shots would be required if there were a hundred locations (about six shots), a thousand locations (about nine), or a million (about nineteen)? (Notice that the number of shots increases very slowly compared to the number of ships. One extra shot is required each time the size doubles, so it is proportional to the logarithm of the number of ships.)

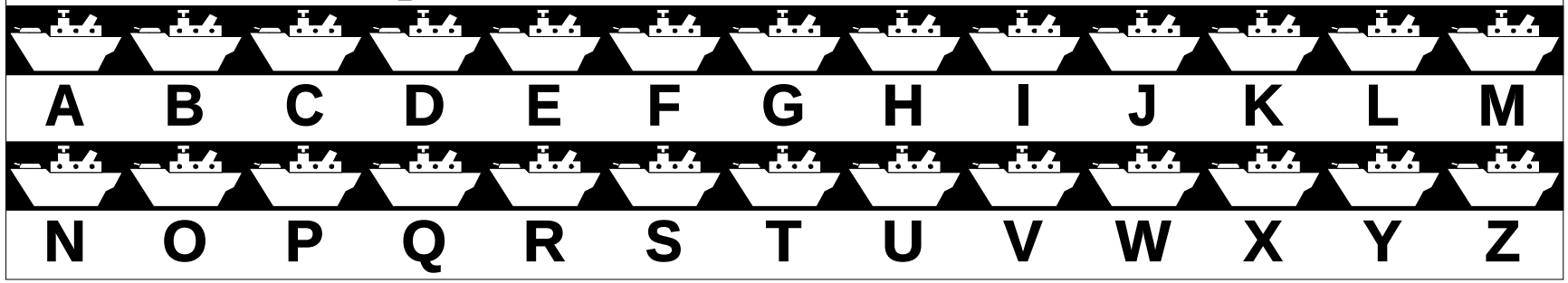
My Ships

Number of Shots Used:



Your Ships

Number of Shots Used:



1A

1B

My Ships

Number of Shots Used:

1630	9263	4127	405	4429	7113	3176	4015	7976	88	3465	1571	8625
A	B	C	D	E	F	G	H	I	J	K	L	M
2587	7187	5258	8020	1919	141	4414	3056	9118	717	7021	3076	3336
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

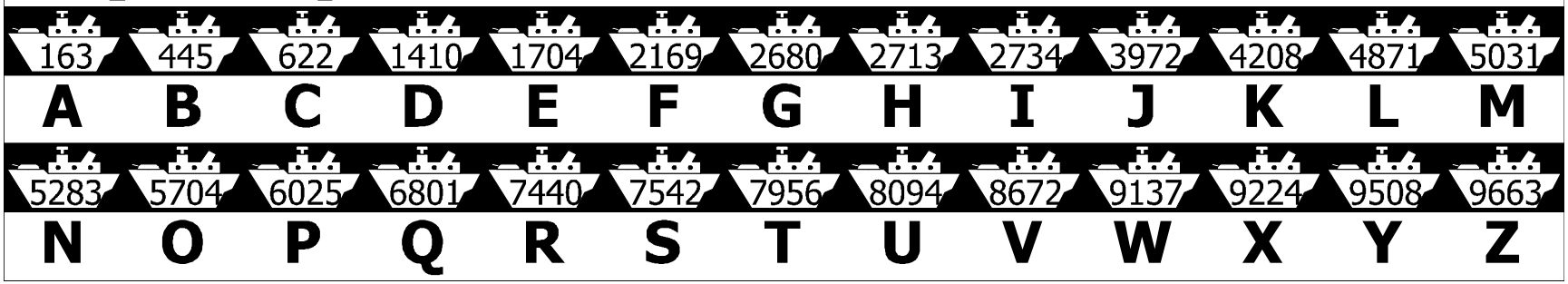
Your Ships

Number of Shots Used:

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

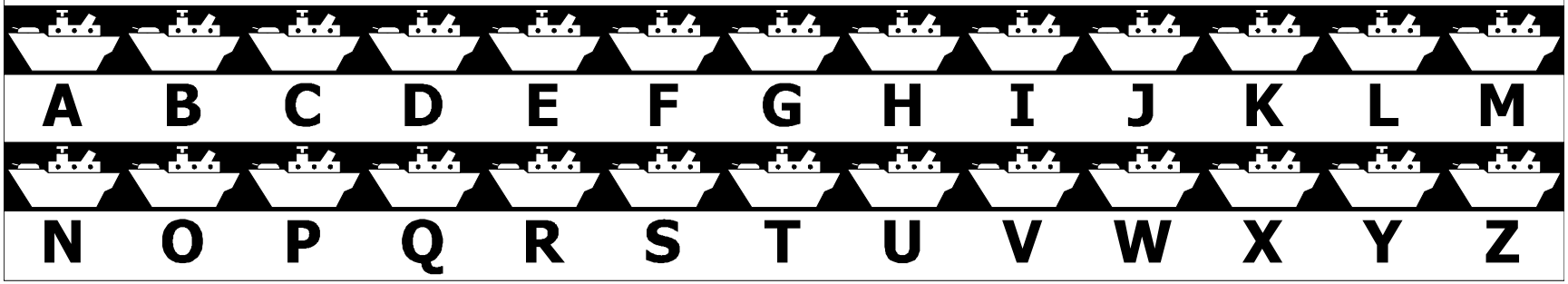
My Ships

Number of Shots Used:



Your Ships

Number of Shots Used:



2A

2B

My Ships

Number of Shots Used:

33	183	730	911	1927	1943	2200	2215	3451	3519	4055	5548	5655
A	B	C	D	E	F	G	H	I	J	K	L	M
5785	5897	5905	6118	6296	6625	6771	6831	7151	7806	8077	9024	9328
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Your Ships

Number of Shots Used:

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

My Ships

Number of Shots Used:

0	1	2	3	4	5	6	7	8	9
A 9047	C 3080		E 5125	H 8051	L 7116	O 6000	R 9891		W 1062
B 1829	D 9994		F 1480	I 1481	M 8944	P 7432	S 1989	V 4392	X 2106
			G 8212	J 4712	N 4128	Q 4110	T 2050		Y 5842
				K 6422			U 8199		Z 7057

Your Ships

Number of Shots Used:

0	1	2	3	4	5	6	7	8	9
A	E	H	K	L		O	R	V	Y
B	F	I		M		P	S	W	Z
C	G	J		N		Q	T	X	
D							U		

3A

My Ships

Number of Shots Used:

0	1	2	3	4	5	6	7	8	9
A 9308	E 6519	H 1524	K 4135	L 9050		O 4200	R 3121	V 2385	Y 1990
B 1478	F 2469	I 8112		M 1265		P 7153	S 9503	W 5832	Z 2502
C 8417	G 5105	J 2000		N 5711		Q 6028	T 1114	X 1917	
D 9434							U 7019		

Your Ships

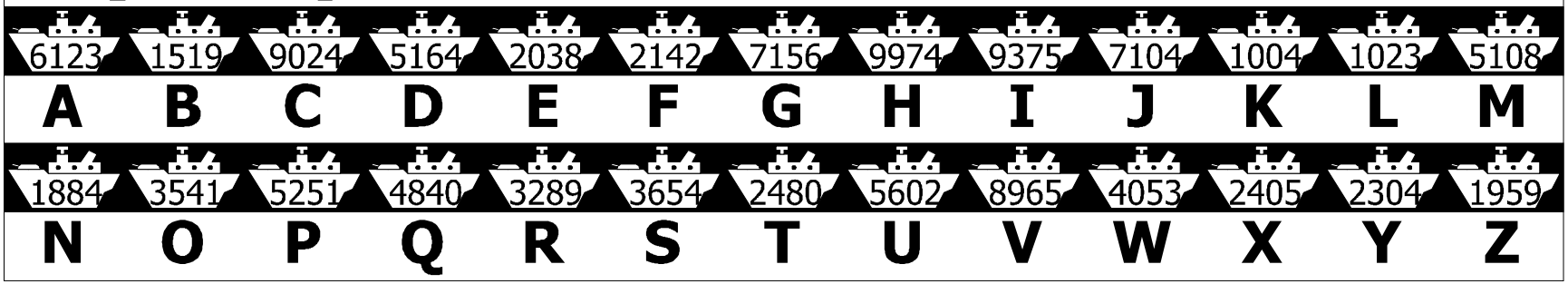
Number of Shots Used:

0	1	2	3	4	5	6	7	8	9
A	C		E	H	L	O	R	V	W
B	D		F	I	M	P	S		X
			G	J	N	Q	T		Y
				K			U		Z

3B

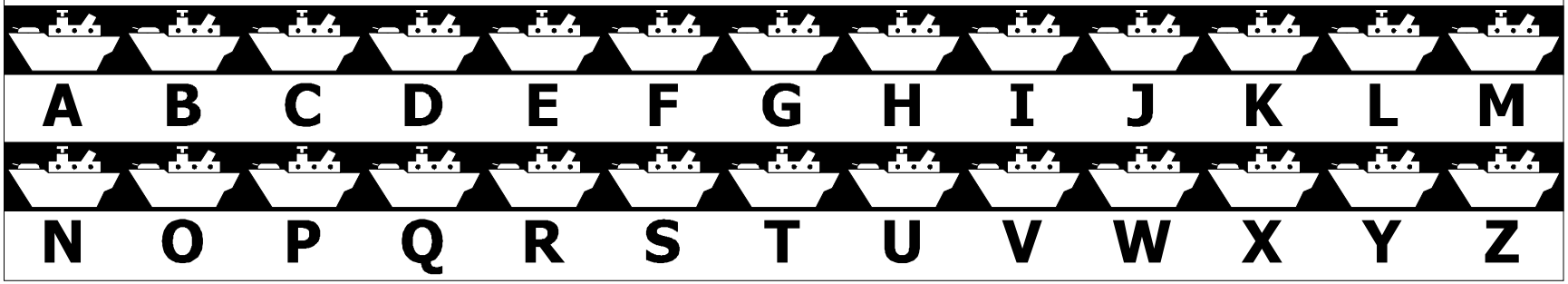
My Ships

Number of Shots Used:



Your Ships

Number of Shots Used:



1A′

1B′

My Ships

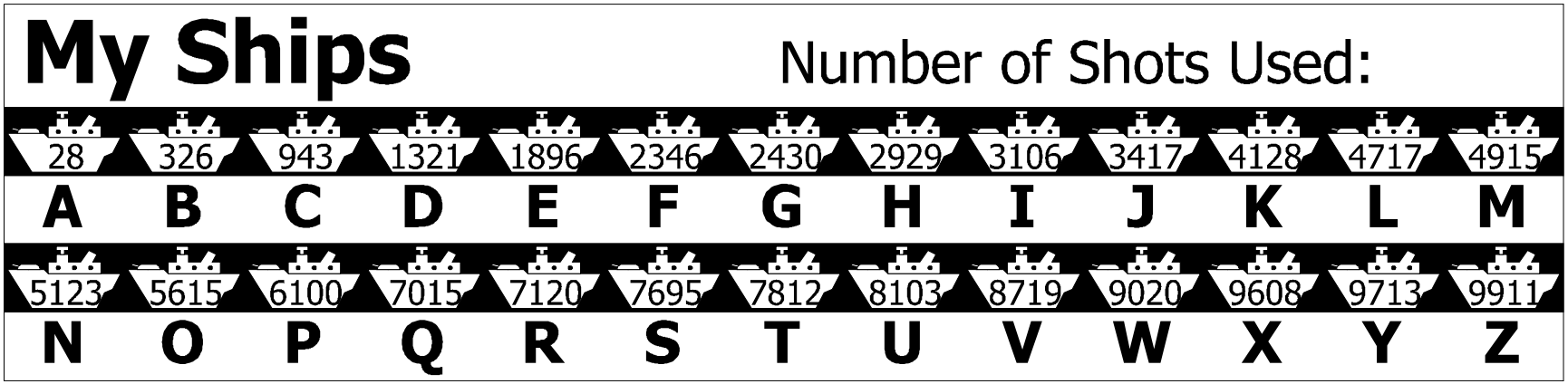
Number of Shots Used:

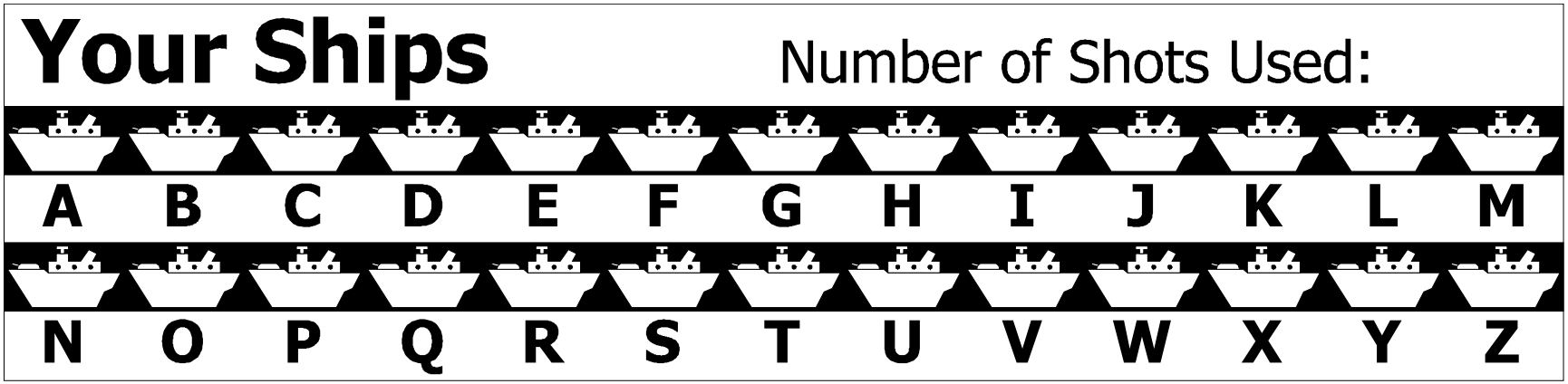
2387	9003	3951	5695	1284	4761	7118	1196	1741	3791	3405	3132	6682
A	B	C	D	E	F	G	H	I	J	K	L	M
9493	9864	7359	1250	7036	2916	7562	9299	8910	6713	5173	8617	4222
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Your Ships

Number of Shots Used:

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





2A′

2B′

My Ships

Number of Shots Used:

56	194	306	1024	1510	1807	2500	2812	3011	3902	4178	5902	5915
A	B	C	D	E	F	G	H	I	J	K	L	M
6102	6526	6818	7020	7155	7913	8016	8230	8599	8902	9090	9526	9812
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Your Ships

Number of Shots Used:

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

My Ships

Number of Shots Used:

0	1	2	3	4	5	6	7	8	9
A 1982	C 6113		E 9121	H 5009	L 1248	O 2004	R 9369		W 9172
B 7841	D 1055		F 1011	I 2651	M 1716	P 5173	S 1321	V 3285	X 2052
			G 2984	J 1751	N 2148	Q 2806	T 3004		Y 6012
				K 4848			U 7190		Z 7525

Your Ships

Number of Shots Used:

0	1	2	3	4	5	6	7	8	9
A	E	H	K	L		O	R	V	Y
B	F	I		M		P	S	W	Z
C	G	J		N		Q	T	X	
D							U		

3A′

My Ships

Number of Shots Used:

0	1	2	3	4	5	6	7	8	9
A 8615	E 1361	H 7726	K 3000	L 1814		O 9656	R 6993	V 8208	Y 2917
B 7003	F 7644	I 9003		M 2002		P 4002	S 3121	W 9423	Z 4122
C 1991	G 5600	J 5557		N 8844		Q 1221	T 4300	X 4176	
D 6211							U 1907		

Your Ships

Number of Shots Used:

0	1	2	3	4	5	6	7	8	9
A	C		E	H	L	O	R	V	W
B	D		F	I	M	P	S		X
			G	J	N	Q	T		Y
				K			U		Z

3B′

What's it all about?

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. 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*.

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 Linear Searching Game. 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, 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!

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 fourteen probes, which might take two hundredths of a second—hardly noticeable.

A third strategy for finding data is called *hashing*. Here the search key is manipulated to indicate exactly where to find the information. For example, if the search key is a telephone number, you could add up all the digits in the number and take the remainder when divided by 11. In this respect, a hash key is a little like the check digits discussed in Activity 4—a small piece of data whose value depends on the other data being processed. Usually the computer will find what it is looking for straight away. There is a small chance that several keys end up in the same location in which case the computer will need to search through them until it finds the one it is seeking.

Computer programmers usually use some version of the hashing strategy for searching, unless it is important to keep the data in order, or unless an occasional slow response is unacceptable.

Activity 7

Lightest and Heaviest—*Sorting Algorithms*

Summary

Computers are often used to put lists into some sort 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, and also makes extreme values easy to see. If you sort the marks for a class test into numeric order, the lowest and highest marks become obvious.

If you use the wrong method, 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 discover different methods for sorting, and see how a clever method can perform the task much more quickly than a simple one.

Curriculum links

- ✓ Mathematics: Measurement Level 2 and up. Carrying out practical weighing tasks.

Skills

- ✓ Using balance scales
- ✓ Ordering
- ✓ Comparing

Ages

- ✓ 8 and up

Materials

Each group of children will need:

- ✓ Sets of 8 containers of the same size but different weights (e.g. milk cartons or film canisters filled with sand)
- ✓ Balance scales
- ✓ Worksheet Activity: Sorting weights (page 66)
- ✓ Worksheet Activity: Divide and conquer (page 67)

Lightest and Heaviest

Discussion

Computers often have to sort lists of things into order. Brainstorm all the places where putting things into order is important. What would happen if these things were not in order?

Computers usually only compare two values at once. The activity on the next page uses this restriction to give children an idea of what this is like.

Activity

1. Divide the children into groups.

2. Each group will need a copy of the activity sheet on page 66, and its own weights and scales.

3. Have the children do the activity, then discuss the result.

Worksheet Activity: Sorting Weights

Aim: To find the best method of sorting a group of unknown weights into order.

You will need: Sand or water, 8 identical containers, a set of balance scales

What to do:

1. Fill each container with a different amount of sand or water. Seal tightly.

2. Mix them up so that you no longer know the order of the weights.

3. Find the lightest weight. What is the easiest way of doing this?

   Note: You are only allowed to use the scales to find out how heavy each container is. Only two weights can be compared at a time.

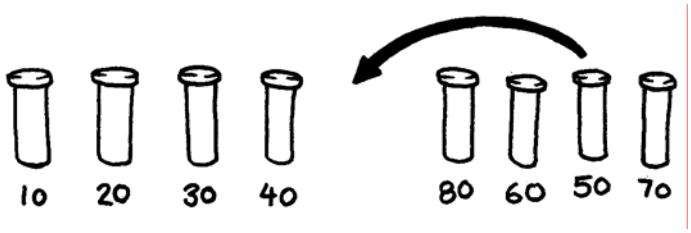
4. Choose 3 weights at random and sort them into order from lightest to heaviest using only the scales. How did you do this? What is the minimum number of comparisons you can make? Why?

5. Now sort all of the objects into order from lightest to heaviest.

When you think you have finished, check your ordering by re-weighing each pair of objects standing together.

Selection Sort

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 it to one side. Next, find the lightest of the weights that are left, and remove it. Repeat this until all the weights have been removed.



Count how many comparisons you made.

Extra for Experts: Show how you can calculate mathematically how many comparisons you need to make to sort 8 objects into order. What about 9 objects? 20?

Worksheet Activity: Divide and Conquer

Quicksort

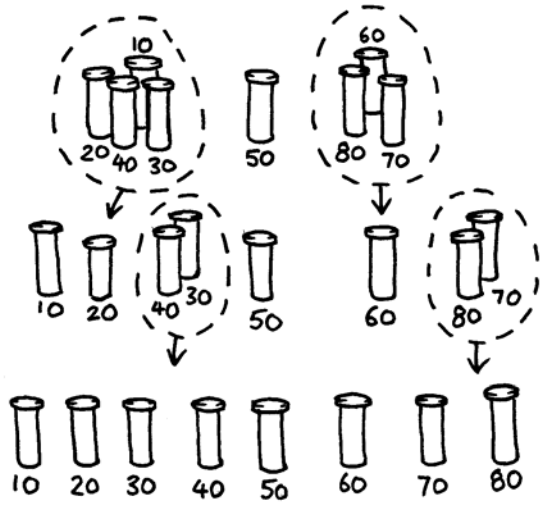
Quicksort is a lot faster than selection sort, particularly for larger lists. In fact, it is one of the best methods known. This is how quicksort works.

Choose one of the objects at random, and place it on one side of the balance scales.

Now compare each of the remaining objects with it. Put those that are lighter on the left, the chosen object in the middle, and the heavier ones 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.



How many comparisons did this process take?

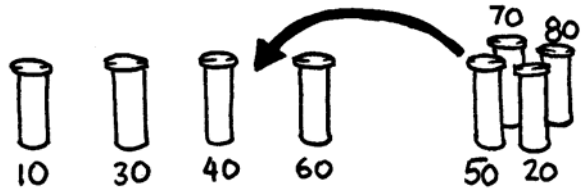
You should find that quicksort is a more efficient method than selection sort unless you happen to have chosen the lightest or heaviest weight to begin with. If you were lucky enough to have chosen the middle weight, you should have taken only 14 comparisons, compared with the 28 for selection sort. At any rate the quicksort method will never be any worse than selection sort and may be much better!

Extra for Experts: If quicksort accidentally always chose the lightest object, how many comparisons would it use?

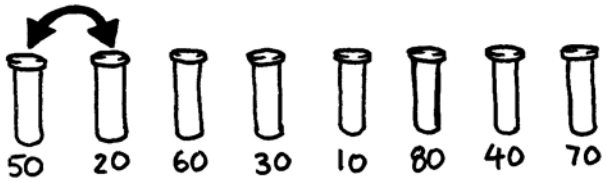
Variations and extensions

Many different methods for sorting have been invented. You could try sorting your weights using these:

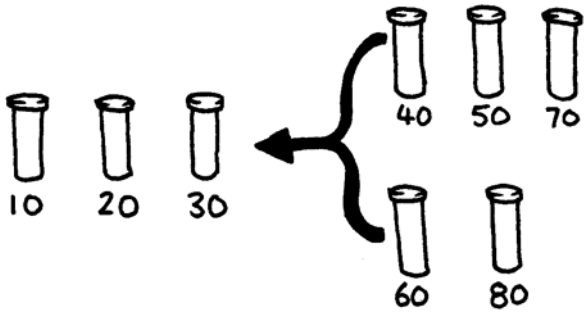
Insertion sort works by removing each object from an unsorted group and inserting it into its correct position in a growing list (see picture below). With each insertion the group of unsorted objects shrinks and the sorted list grows, until eventually the whole list is sorted. Card players often use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



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.



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.



What's it all about?

Information is much easier to find in a sorted list. Telephone directories, dictionaries and book indexes all use alphabetical order, and life would be far more difficult if they didn't. If a list of numbers (such as a list of expenses) is sorted into order, the extreme cases are easy to see because they are at the beginning and end of the list. Duplicates are also easy to find, because they end up together.

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 uses a concept called recursion. This means you keep dividing a list into smaller parts, and then 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.

Solutions and hints

4. The best way to find the lightest weight is to go through each object in turn, keeping track of the lightest one so far. That is, compare two objects, and keep the lighter one. Now compare that with another, keeping the lighter from the comparison. Repeat until all the objects have been used.

5. Compare the weights on the balance scales. This can easily be done with three comparisons, and sometimes just two will suffice—if the children realize that the comparison operator is transitive (that is, if A is lighter than B and B is lighter than C, then A must be lighter than C).

Experts:

Here is a short cut for adding up the number of comparisons that selection sort makes.

To find the minimum of two objects you need one comparison, three needs two, four needs three, and so on. To sort eight objects using selection sort takes 7 comparisons to find the first one, six to find the next, five to find the next and so on. That gives us:

$7 + 6 + 5 + 4 + 3 + 2 + 1 = 28$ comparisons.

n objects will take $1 + 2 + 3 + 4 + \ldots + n - 1$ comparisons to sort.

Adding up these numbers is easy if we regroup them.

For example, to add up the numbers $1 + 2 + 3 + \ldots + 20$, regroup them as

$(1 + 20) + (2 + 19) + (3 + 18) + (4 + 17) + (5 + 16) +$

$(6 + 15) + (7 + 14) + (8 + 13) + (9 + 12) + (10 + 11)$

$= 21 \times 10$

$= 210$

In general, the sum $1 + 2 + 3 + 4 \ldots + n - 1 = n(n - 1)/2$.

Activity 8

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 which do several sorting comparisons at the same time.

Curriculum Links

- ✓ Mathematics: Number level 2 and up. Exploring number: Greater than, less than

Skills

- ✓ Comparing
- ✓ Ordering
- ✓ Developing algorithms
- ✓ Co-operative problem solving

Ages

- ✓ 7 years and up

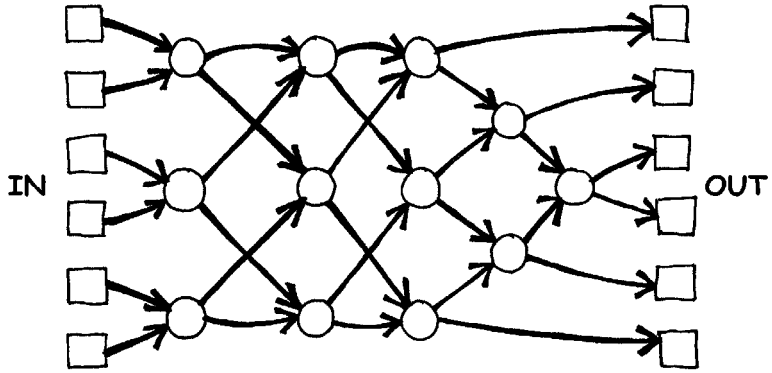
Materials

This is an outdoor group activity.

- ✓ Chalk
- ✓ Two sets of six cards.
  Copy Photocopy Master: Sorting networks (page 73) onto card and cut out
- ✓ Stopwatch

Sorting Networks

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



Instructions for Children

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

1. Organise yourselves into groups of six. Only one team uses the network at a time.

2. Each team member takes a numbered card.

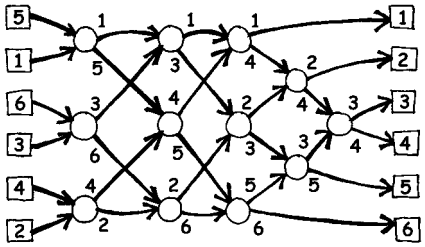
3. Each member stands in a square on the left hand (IN) side of the court. Your numbers should be in jumbled order.

4. You 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 children 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:



Photocopy Master: Sorting networks

1	2
3	4
5	6

156	221
289	314
422	499

Variations

1. When the children 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).

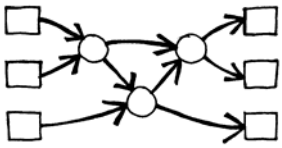
3. Make up cards with even larger numbers that will take some effort to compare, or use words and compare them alphabetically.

Extension Activities

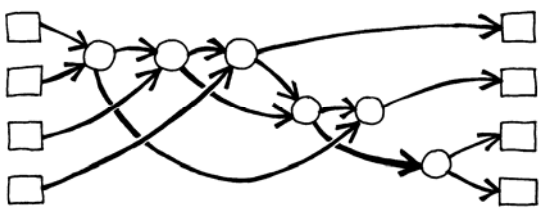
1. What happens if the smaller one goes right instead of left and vice versa? (The numbers will be sorted in reverse order.)

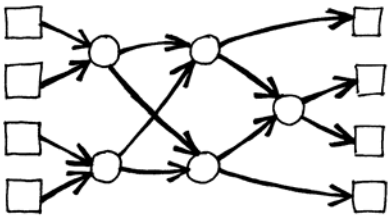
   Does it work if the network is used backwards? (It will not necessarily work, and the children should be able to find an example of an input that comes out in the wrong order.)

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



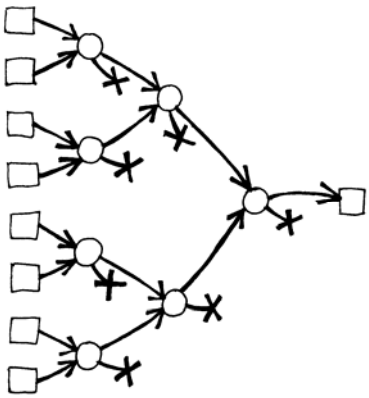
3. Below are two different networks that will sort four inputs. Which is the faster? (The second one is. Whereas the first requires all comparisons to be done serially, one after the other, the second has some being performed at the same time. The first network is an example of serial processing, whereas 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 (as shown in Activities 6 and 7).

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 six-number sorting network, although a total of 12 comparisons are used to sort the numbers, up to three comparisons are performed simultaneously. This means that the time required will 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 ten metres long. If ten people each dug one metre of the ditch the task would be completed much faster. However, the same strategy could not be applied to a ditch ten metres deep—the second metre is not accessible until the first metre has been dug. 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 9

The Muddy City—*Minimal Spanning Trees*

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.

Curriculum Links

- ✓ Mathematics: Geometry Level 2/3 and up. Exploring shape and space: Finding the shortest paths around a map

Ages

- ✓ 9 and up

Skills

- ✓ Problem solving

Materials

Each child will need:

- ✓ Workshop Activity: The muddy city problem (page 78)
- ✓ Counters or squares of cardboard (approximately 40 per child)

The Muddy City

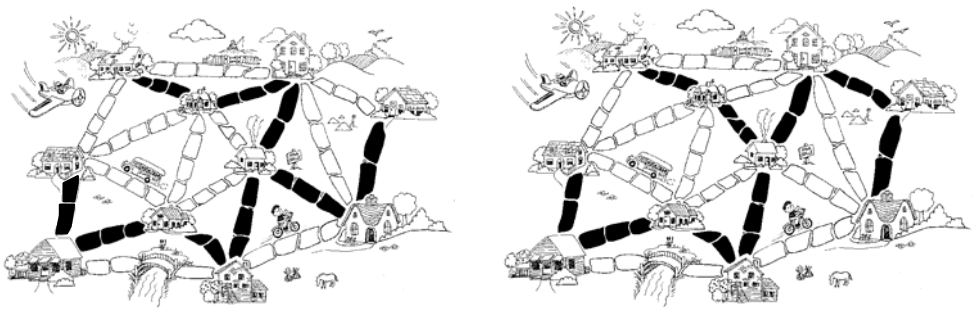
Introduction

This activity will show you how computers are used to find the best solutions for real-life problems such as how to link power lines between houses. Have the children use the worksheet on page 78, which explains the ‘Muddy City’ problem.

Follow-up discussion

Share the solutions the children 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?

Computer scientists call the representations of these networks “graphs”. Real networks can be represented by a graph to solve problems such as designing the best network of roads between local cities, or aeroplane flights around the country.

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.

Worksheet Activity: The Muddy City Problem

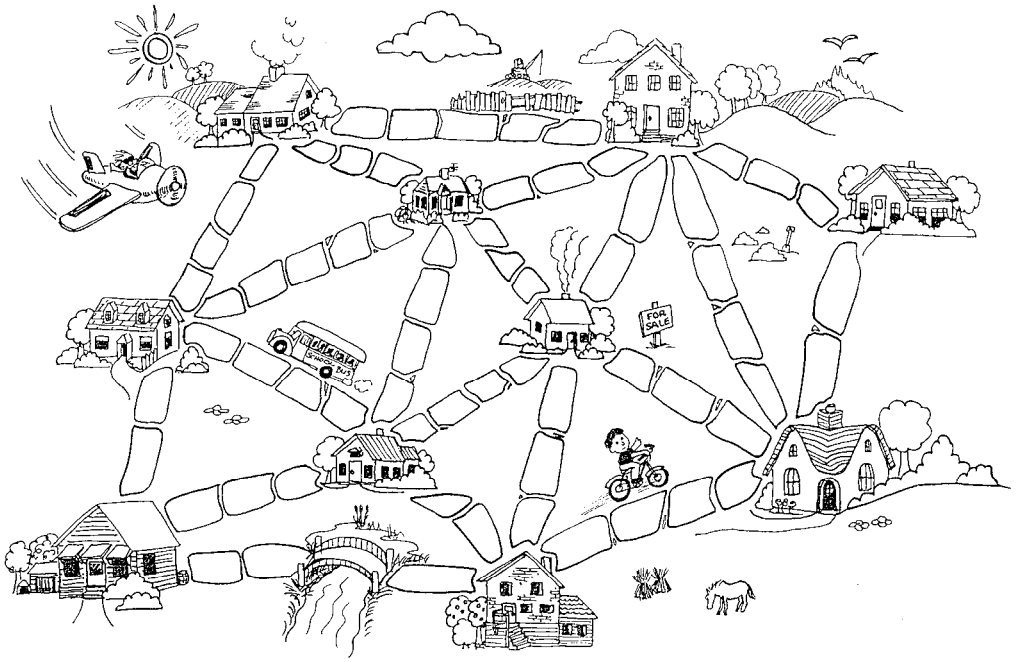
Once upon a time there was a city that had no roads. Getting around the city was particularly difficult after rainstorms because the ground became very muddy—cars got stuck in the mud and people got their boots dirty. The mayor of the city decided that some of the streets must be paved, but didn't want 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, and

2. The paving should cost as little as possible.

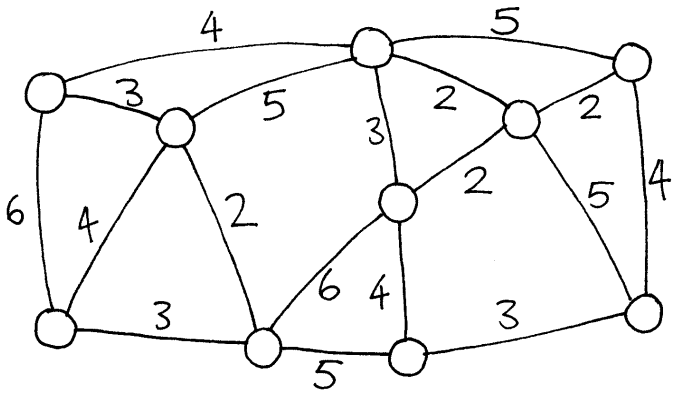
Here is the layout of the city. The number of paving stones between each house represents the cost of paving that route. Find the best route that connects all the houses, but uses as few counters (paving stones) as possible.

What strategies did you use to solve the problem?



Variations and extensions

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.

Make up some of your own muddy city problems and try them out on your friends.

Can you find out a rule to describe how many roads or connections are needed for a best solution? Does it depend on how many houses there are in the city?

What's it all about?

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 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.

The task of designing a network with a minimal total length is called the *minimal spanning tree* problem.

Minimal spanning trees aren't only useful in gas and power networks; they also help us solve problems in computer networks, telephone networks, oil pipelines, and airline routes. 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 traveller as well as how much it will cost. No-one wants to spend hours in an aeroplane taking the long way round to a new country just because it is cheaper. 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.

Minimal spanning trees are also useful as one of the steps for solving other problems on graphs, such as the "travelling salesperson problem" which tries to find the shortest route that visits every point in the network.

There are efficient algorithms (methods) for solving minimal spanning tree problems. A simple method that gives an optimal solution is to start with no connections, and add them in increasing order of size, only adding connections that join up part of the network that wasn't previously connected. This is called Kruskal's algorithm after J.B. Kruskal, who published it in 1956.

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.

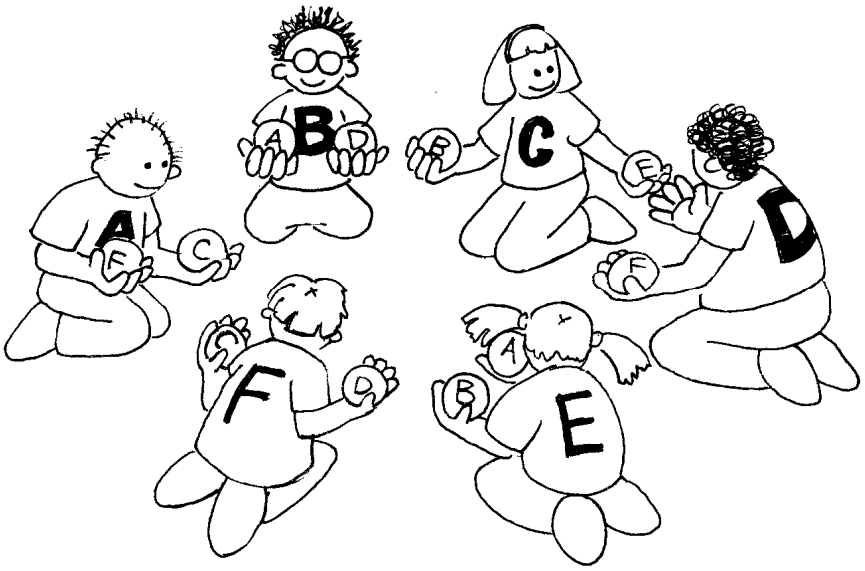
Solutions and hints

Variations and extensions (page 79)

How many roads or connections are needed if there are n houses in the city? It turns out that an optimal solution will always have exactly n–1 connections in it, as this is always sufficient to link up the n houses, and adding one more would create unnecessary alternative routes between houses.

Activity 10

The Orange Game—*Routing and Deadlock in Networks*



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

- ✓ Co-operative problem solving
- ✓ Logical reasoning

Ages

- ✓ 9 years and up

Materials

Each child will need:

- ✓ Two oranges or tennis balls
- ✓ Name tag or sticker

The Orange Game

Introduction

This is a co-operative problem solving game. The aim is for each person to end up holding the oranges labelled with their own letter.

1. Groups of five or more children sit in a circle.

2. The children are labelled with a letter of the alphabet (using name tags or stickers). There are two oranges with each child's letter on them, except for one child, who only has one corresponding orange to ensure that there is always an empty hand.

3. Distribute the oranges randomly to the children in the circle. Each child has two oranges, except for one child who has only one. (No child should have an orange with their letter on it.)

4. The children pass the oranges around until each child gets the oranges labelled with their letter of the alphabet. You must follow two rules:

   a) Only one orange may be held in a hand.

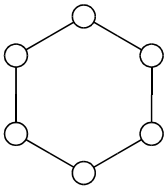
   b) An orange can only be passed to an empty hand of an immediate neighbour in the circle. (A child can pass either of their two oranges to their neighbour.)

Children will quickly find that if they are "greedy" (hold onto their own oranges as soon as they get them) then the group might not be able to attain its goal. It may be necessary to emphasize that individuals don't "win" the game, but that the puzzle is solved when everyone has their oranges.

Follow up Discussion

What strategies did the children use to solve the problem?

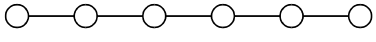
Where in real life have you experienced deadlock? (Some examples might be a traffic jam, getting players around bases in baseball, or trying to get a lot of people through a doorway at once.)

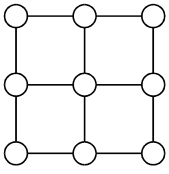


Extension Activities

Try the activity with a smaller or larger circle.

- Have the children come up with new rules.
- Carry out the activity without any talking.
- Try different configurations such as sitting in a line, or having more than two neighbours for some children. Some suggestions are shown here.





What's it all about?

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.

Routing, congestion and deadlock can present frustrating problems in many different networks. Just think of your favourite rush-hour traffic! It has happened several times in New York City that the traffic in the streets has become so congested that it deadlocks: no-one can move their car! Sometimes when the computers are "down" in businesses (such as banks) the problem is caused by a communication network deadlock. Designing networks so that routing is easy and efficient and congestion is minimized is a difficult problem faced by many kinds of engineers.

Sometimes more than one person wants the same data at the same time. If a piece of data (such as a customer's bank balance) is being updated, it is important to "lock" it during the update. If it is not locked, someone else could update it at the same time and the balance might be recorded incorrectly. However, if this locking is interfered with by the locking of another item, deadlock may occur.

One of the most exciting developments in computer design is the advent of parallel computing, where hundreds or thousands of PC-like processors are combined (in a network) to form a single powerful computer. Many problems like the Orange Game must be played on these networks continuously (but much faster!) in order for these parallel computers to work.

Part III

Telling Computers What To Do—
Representing Procedures

Telling Computers What To Do

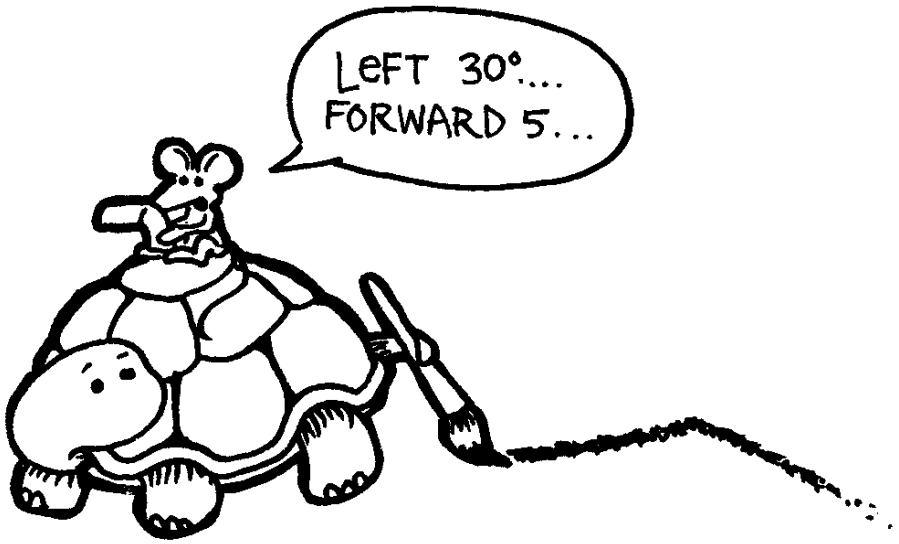
Computers follow instructions—millions of instructions every second. To tell a computer what to do, all you have to do is give it the right instructions. But that's not as easy as it sounds!

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*, if necessary opening the door first! Computers are different. Indeed, 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 exactly, without "thinking," takes some getting used to.

The two activities in this section give us some idea of what it is like to communicate to literal-minded machines using a fixed set of instructions.

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

The second activity introduces us to how we can communicate with computers. A good programmer has to learn how to tell the computer what to do using a fixed set of instructions that are interpreted literally. The list of instructions is the program. There are lots of different programming languages a programmer can choose to write these instructions in, but we will be using a simple language that can be used without a computer.



Activity 11

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

Curriculum Links

- ✓ Mathematics: Developing logic and reasoning—using words and symbols to describe and continue patterns
- ✓ Social Studies
- ✓ English

Skills

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

Ages

- ✓ 9 and up

Materials

You will need:

- ✓ One set of island cards (the instructions must be kept hidden from those trying to draw the map!)
  Copy Photocopy Master: Island cards (page 92 onwards) and cut out.
  Fold along the dotted line and glue, so that the front of the card has the name of the island, and the back has the instructions.

Each child will need:

- ✓ Worksheet Activity: Find your way to the riches on Treasure Island (page 91)
- ✓ Pen or pencil

There are optional extension activities, for which each child will need:

- ✓ Worksheet Activity: Treasure islands (page 97)
- ✓ Worksheet Activity: The mysterious coin game (page 98)

Treasure Island

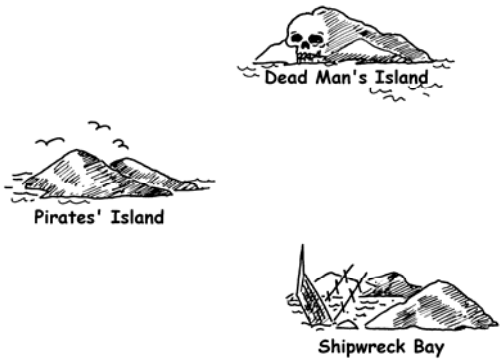
Introduction

Your goal is to find Treasure Island. Friendly pirate ships sail along a fixed set of routes between the islands in this part of the world, offering rides to travellers. Each island has two departing ships, A and B, which you can choose to travel on. You need to find the best route to Treasure Island. At each island you arrive at you may ask for either ship A or B (not both). The person at the island will tell you where your ship will take you to next, but the pirates don't have a map of all the islands available. Use your map to keep track of where you are going and which ship you have travelled on.

Demonstration

(**Note:** This is a different map from the actual activity.)

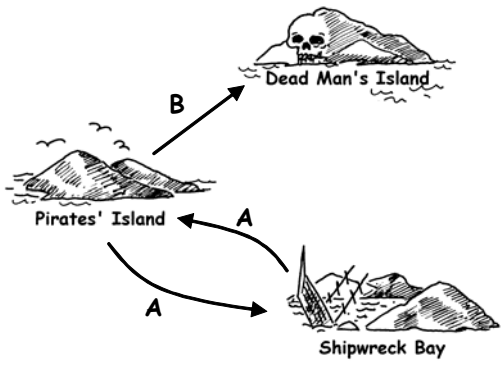
Using an OHP or board, draw a diagram of three islands as shown here:



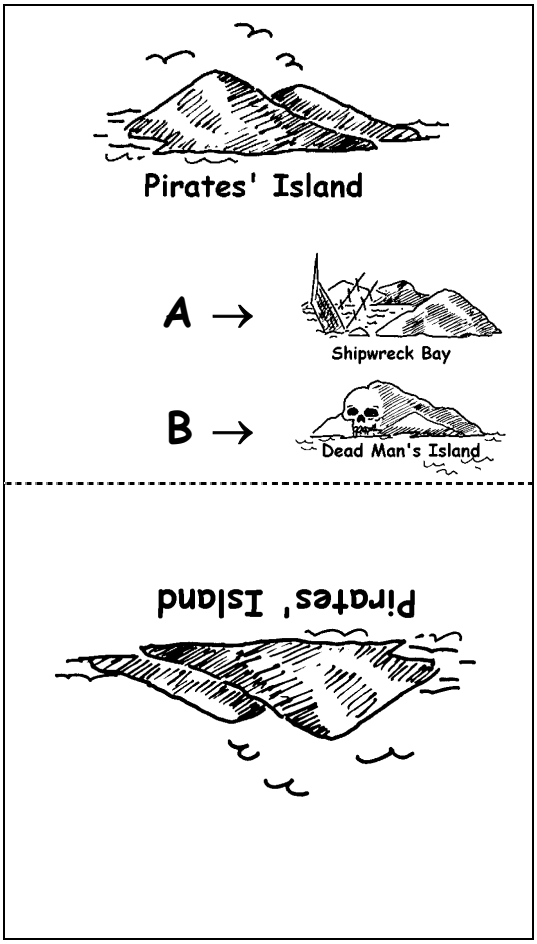
Copy the three cards on the next two pages, and have one child hold each card. Note that the routes on these cards are different from those in the main activity.

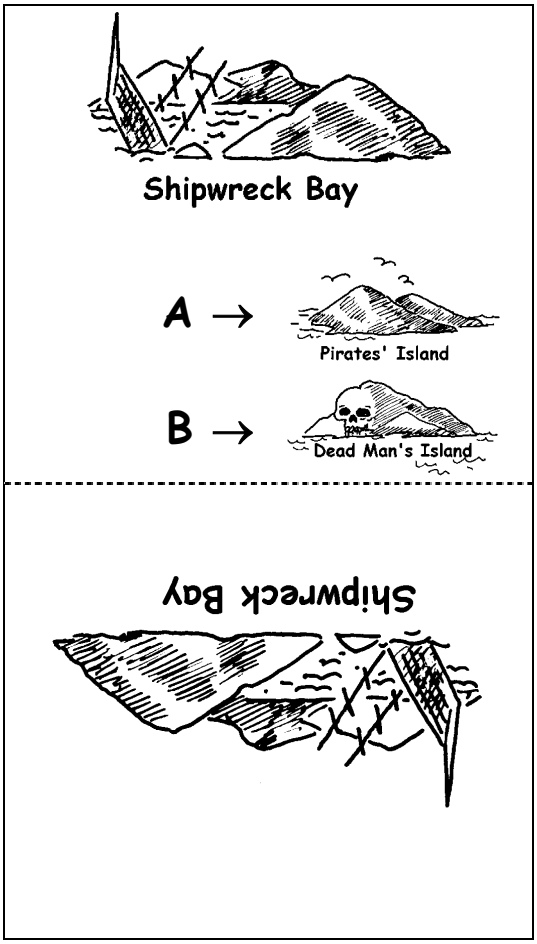
Starting at Pirates' Island ask for ship A. The child should direct you to Shipwreck Bay. Mark the route in on the map. At Shipwreck Bay ask for ship A again. You will be directed back to Pirates' island. Mark this on the map. This time ask for ship B. Mark this on the map. This route goes to Dead Man's Island, at which stage you will be stuck!

Your final map should look like this:



Cards for demonstration activity





Cards for demonstration activity

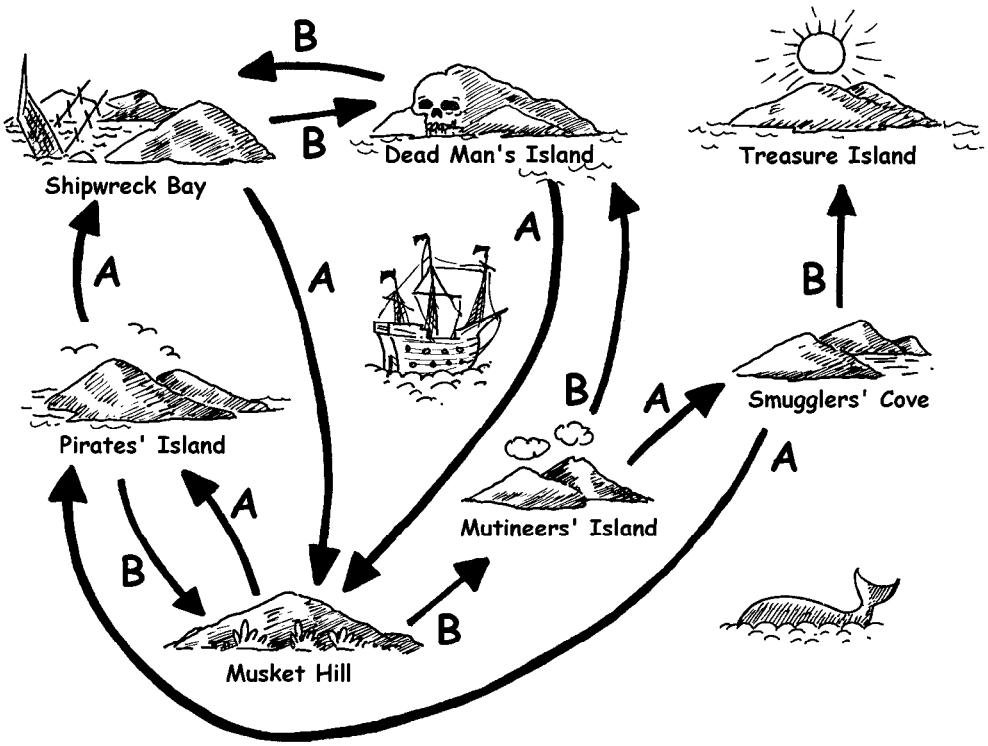
No ships sail from Dead Man's Island!

Activity

Choose 7 children to be "islands". The children will hold cards identifying their island, with the secret instructions on the back. Position them randomly around the room or playground. The rest of the children are given the blank map and have to navigate a route from Pirates' Island to Treasure Island, marking it carefully on their maps. (It is a good idea to send the children off one at a time so they cannot hear the routes in advance.)

Fast Finishers: Try to find more than one route.

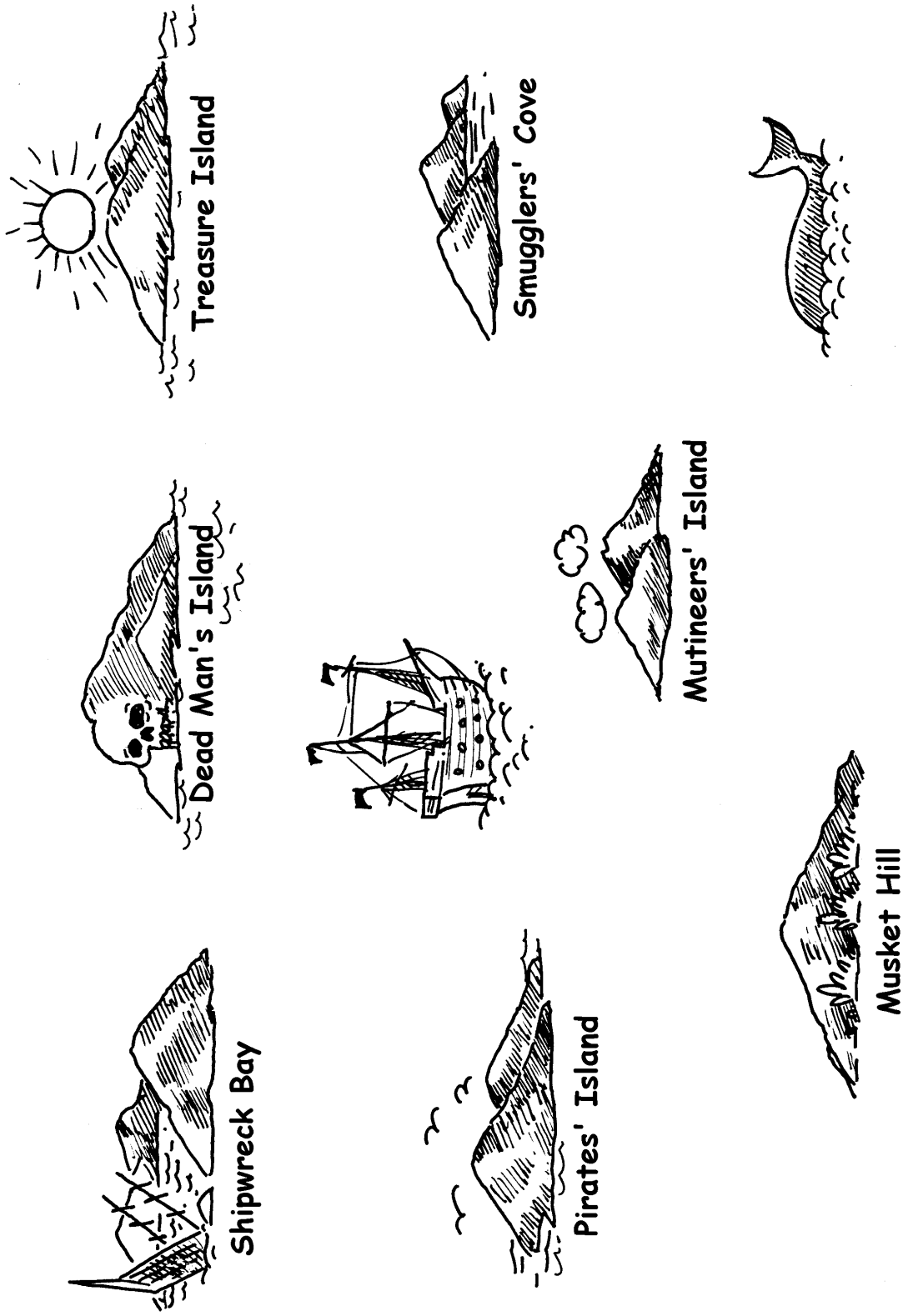
The complete map looks like this:



Follow-up discussion

What is the quickest route? What would be a very slow route? Some routes may involve loops. Can you find an example of this? (For example, BBBABAB and BBBABBABAB both get to Treasure Island.)

Worksheet Activity:
Find your way to the riches on Treasure Island



Photocopy Master: Island cards (1/4)

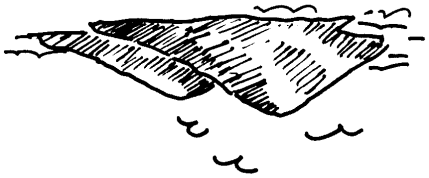
Pirates' Island

A →

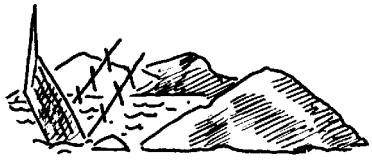
Shipwreck Bay

B →

Musket Hill



Pirates' Island



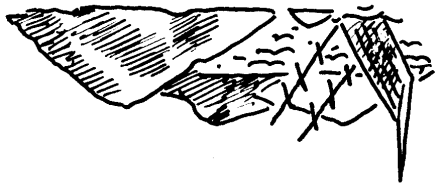
Shipwreck Bay

A →

Musket Hill

B →

Dead Man's Island



Shipwreck Bay

Photocopy Master: Island cards (2/4)

Musket Hill

A → 

Pirates' Island

B → 

Mutineers' Island

Musket Hill

Dead Man's Island

A → 

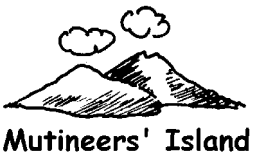
Musket Hill

B → 

Shipwreck Bay

Dead Man's Island

Photocopy Master: Island cards (3/4)



Mutineers' Island

A → 
Smugglers' Cove

B → 
Dead Man's Island

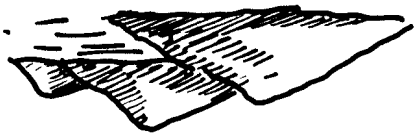
Mutineers' Island

Smugglers' Cove

A → 
Pirates' Island

B → 
Treasure Island

Smugglers' Cove

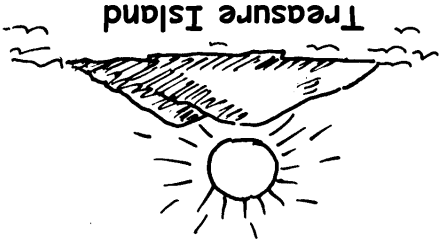


Photocopy Master: Island cards (4/4)

Treasure Island

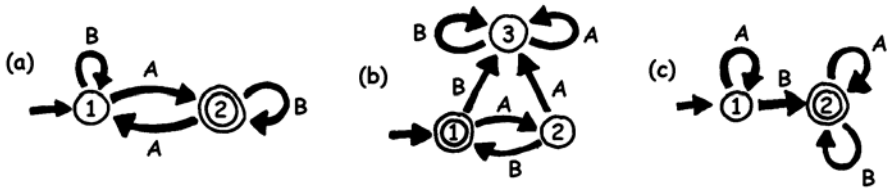
Congratulations!

Treasure Island



Finite-State Automata

Another way of drawing a map is like this:



The islands are shown as numbered circles, and the final island (with the treasure) has a double circle. What routes can we travel around to get to the final island?

Note: Map (a) will finish at the double circle (island 2) only if the sequence has an odd number of As (for example, AB, BABAA, or AAABABA).

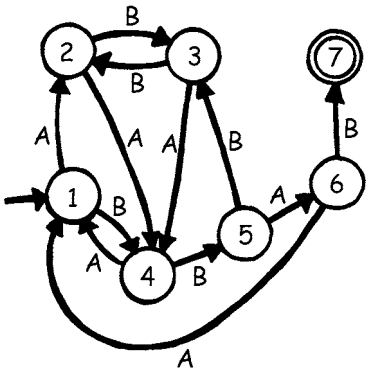
Map (b) only gets to the double circle with an alternating sequence of As and Bs (AB, ABAB, ABABAB, ...).

Map (c) requires that the sequence contains at least one B (the only sequences *not* suitable are A, AA, AAA, AAAA, ...).

Worksheet Activity: Treasure Islands

Can you hide your buried treasure well? How hard can you make it to find the treasure? It's time to make your own map!

1. Here is a more complicated version of the same idea of representing a map. This map is the same as for the previous exercise. Computer Scientists use this quick and easy way of designing routes for their patterns.

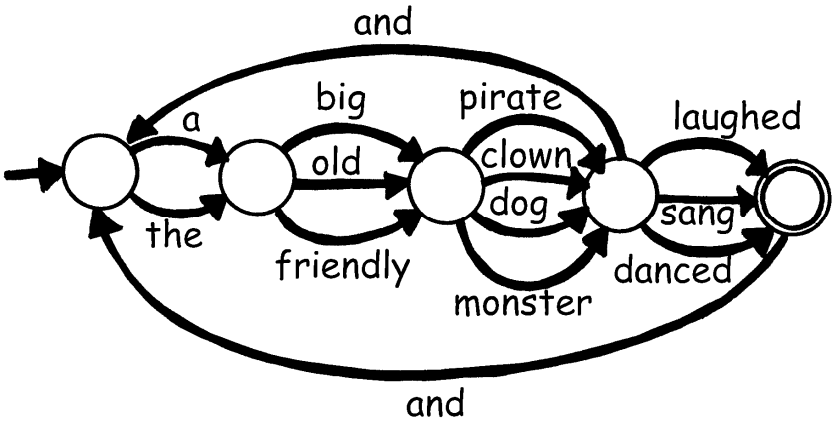


Draw your own basic plan like this so you can clearly see the routes your Pirate ships will travel and then make up your own blank maps and island cards. What is the most efficient sequence of routes to reach your Treasure Island?

2. How well can your friends follow your map? Give them a sequence of As and Bs, and see if they can reach the correct island.

You can make up a variety of games and puzzles based on this idea of finite-state automata.

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



Now try the same idea for yourself. Perhaps you could even make up a funny story!

Worksheet Activity: The Mysterious Coin Game

Some friends downloaded a game from the Internet in which a robot flipped a coin and they had to guess whether it would turn up heads or tails. At first the game looked very easy. At least they would have a 50/50 chance of winning—or so they thought! After a while though they started to get suspicious. There seemed to be a pattern in the coin tosses. Was the game rigged? Surely not! They decided to investigate. Joe wrote down the results of their next attempts at the game and this is what they found: (h = heads, t = tails)

```
h h t h h t h h h t t h h h h t t h t t t h h h h h t h h h
t t t h h h t t t h h h h h h t t h t t t t t h t t h t t t
h h h t t h h h t h h h h h h h h h t t h h h t t t t h h h
h h t t t t t t t
```

Can you find a predictable pattern?

There is a very simple ‘map’ that will describe the sequence of coin tosses. See if you can figure it out. (**Hint:** it has just 4 ‘islands’)

What's it all about?

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".

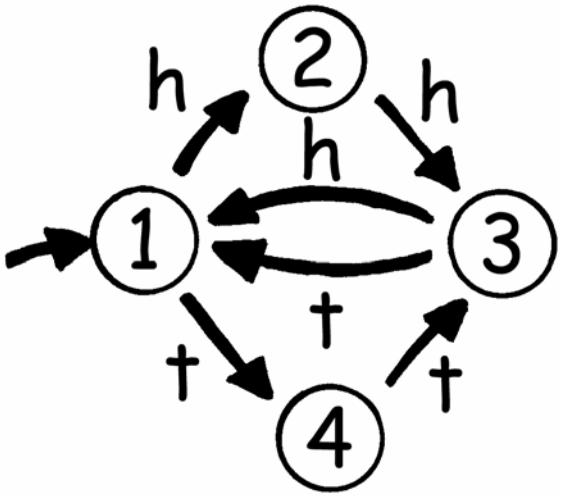
Some computer programs really do deal with English sentences using maps like the one on page 97. They can both generate sentences themselves, and process sentences that the user types in. In the 1960s a computer scientist wrote a famous program called "Eliza" (after Eliza Dolittle) that had conversations 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.

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 the form of elementary computer instructions, which can then be "executed" directly by the computer.

Solutions and hints

The Mysterious Coin Game (page 98)

The mysterious coin game uses the following map to toss coins:



If you follow it, you will see that the first two coin tosses of each three have the same outcome.

Activity 12

Marching Orders—*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.

Curriculum Links

- ✓ English: Interpersonal Listening Level 3

Skills

- ✓ Giving and following instructions.

Ages

- ✓ 7 years and up

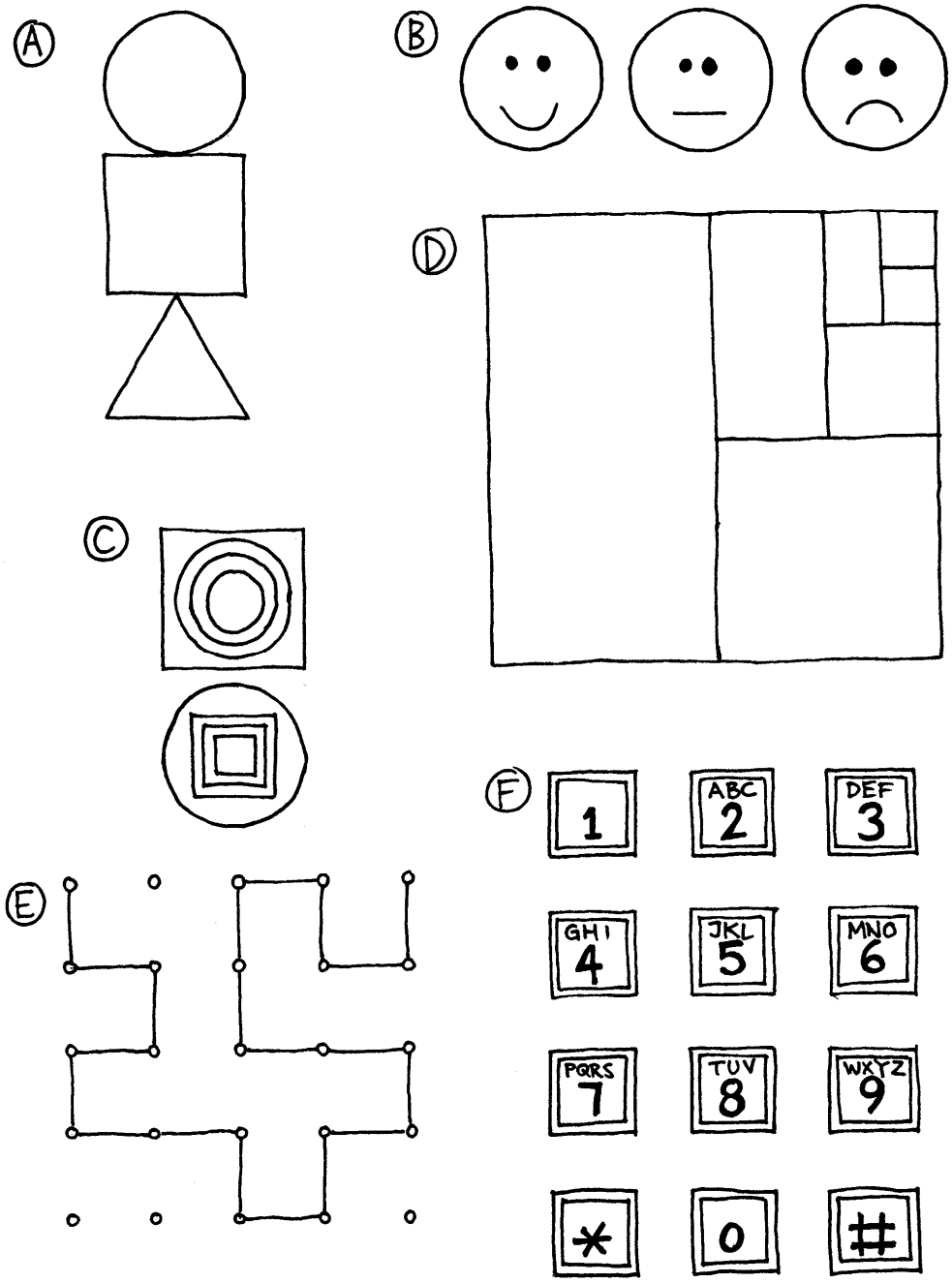
Materials

You will need:

- ✓ Cards with pictures such as the ones shown on the next page.

Each child will need:

- ✓ Pencil, paper and ruler



Marching Orders

Introduction

Discuss whether it would be good if people followed instructions exactly. For example, what would happen if you pointed to a closed door and said, "Go through that door"?

Computers work by following lists of instructions, and they do exactly what the instructions say—even if they don't make sense!

Demonstration Example

See if the children can draw the picture from these instructions.

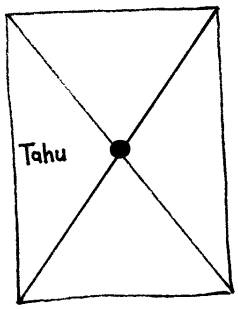
1. Draw a dot in the centre of your page.

2. Starting at the top left-hand corner of the page rule a straight line through the dot finishing at the bottom right hand corner.

3. Starting at the bottom left-hand corner of the page rule a line through the dot, finishing at the top right hand corner.

4. Write your name in the triangle in the centre of the left-hand side of the page.

The result should look something like this:



Activities

Choose a child and give them an image (such as the examples on page 102). The child describes the picture for the class to reproduce. The children can ask questions to clarify the instructions. The object is to see how quickly and accurately the exercise can be completed.

Repeat the exercise, but this time the children are not allowed to ask questions. It is best to use a simpler image for this exercise, as the children can get lost very quickly.

Now try the exercise with the instructing child hidden behind a screen, without allowing any questions, so that the only communication is in the form of instructions.

Point out that 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 children draw a picture and write down their own instructions. Try them out in pairs or as a whole class.

Variations

1. Write instructions to construct a paper dart.

2. Write instructions on how to get to a mystery location around the school using such instructions as "Go forward *x* metres", "turn left" (90 degrees), and "turn right" (90 degrees).

   Children should test and refine their instructions until they have the desired effect.

3. Blind Game. Blindfold a child and have the other children direct them around the room.

What's it all about?

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. Some languages are more suitable for some purposes than others.

Regardless of what language they use, programmers must become adept at specifying *exactly* what they want the computer to do. Unlike human beings, a computer will carry out instructions to the letter even if they are patently ridiculous.

It is important that programs are well written. A small error can cause a lot of problems. Imagine the consequences of 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 honour (so it is said) of a moth that was once removed ("debugged") from an electrical relay in an early 1940s electronic calculating machine.

The more complex the program, the more errors there are likely to be. This became a major issue when the USA was working on the Strategic Defence Initiative ("Star Wars") program, a computer controlled system that was intended to form an impenetrable defence against nuclear attack. Some computer scientists claimed that it could never work because of the complexity and inherent unreliability of the software required. Software needs to be tested carefully to find as many bugs as possible, and it wouldn't be feasible to test this system since one would have to fire missiles at the United States to be sure that it worked!