Activity 3

You can say that again!—*Text compression*

Age group	Early elementary and up.
Abilities assumed	Copying written text.
Time	10 minutes or more.
Size of group	From individuals to the whole class.

Focus

Writing.

Copying.

Repetition in written text.

Summary

Despite the massive capacity of modern computer storage devices, there is still a need for systems to store data as efficiently as possible. By coding data before it is stored, and decoding it when it is retrieved, the storage capacity of a computer can be increased at the expense of slightly slower access time. This activity shows how text can be coded to reduce the space needed to store it.

Technical terms

Text compression; Ziv-Lempel coding.

Materials

Each child will need:

a copy of the blackline master on page 32.

You will also need:

a copy of other poems or text for the children to encode.

What to do

1. Give each child a copy of the blackline master on page 32, and have them "decode" the message by filling in empty boxes with the contents of the box that their arrow points to. For example, the first empty box should contain the letter *e*. The first empty box on the second line refers to the phrase *Pease porridge* on the first line. The completed worksheet is shown in Figure 3.1.

2. Now have the children write their own representation of some text using boxes and arrows. The goal is to have as few of the original characters left as possible. The arrows should always point to an earlier part of the text to ensure that it can be decoded if the boxes are filled in from top to bottom, left to right. The children can either choose their own text, make some up, or encode some that has been provided. Many nursery rhymes and poems for children can be coded particularly effectively because they tend to have a lot of repetition, at least in the rhyming parts[1]. Books such as Dr. Seuss's "Green eggs and ham" also provide a good source of compressible text.

Variations and extensions

The arrow-and-box code requires that arrows always point to an earlier piece of text, although it is only the first letter pointed to that needs to be earlier. For example, Figure 3.2 shows a coded version of the word "Banana." Even though the missing text points to part of itself, it can be decoded correctly provided the letters are copied from left to right—each letter becomes available for copying before it is needed. This kind of self-referential representation is useful if there is a long run of a particular character or pattern.

On computers the boxes and arrows are represented by numbers. For example, the code in Figure 3.2 might be represented as "Ban(2,3)", where the 2 means count back two characters

[1]A good rhyme is "A dillar, a dollar / a ten o'clock scholar / what makes you come so soon / you used to come at ten o'clock / and now you come at noon."

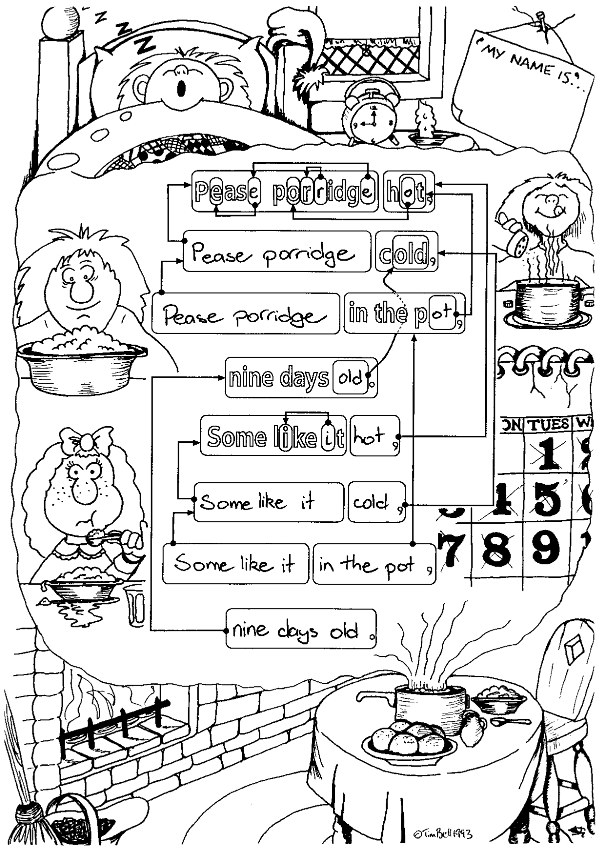


Figure 3.1: The completed worksheet

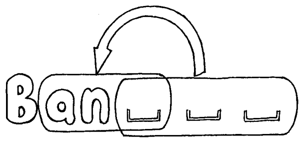


Figure 3.2: A self-referential code

to find the starting point for copying, and the 3 means copy three consecutive characters. On a computer the pair of numbers typically take up a similar amount of space to two letters, so matches of only one character are not normally encoded. Children could try encoding and decoding texts using the number representation.

To get a feel for how this technique would work on real text, children could be given a photocopied page containing a quantity of text, and starting about half-way down the page, cross out any text that could be replaced by a pointer to an earlier occurrence. The earlier occurrences should only be groups of two or more letters, since there is no saving if a single letter is substituted with two numbers. The goal is to get as many letters crossed out as possible.

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 instead of satisfying demand, the growth is fueling it. The ability of computers to store whole books suggests the possibility of storing libraries of books on computer; being able to show a high quality image on a computer suggests displaying movies; and the availability of high capacity CD-ROM simplifies the distribution of data and programs that had previously been limited by the size of a floppy disk. Anyone who owns a computer will have experienced the variation of Parkinson's law that states that the data always expands to fill the storage available for it.

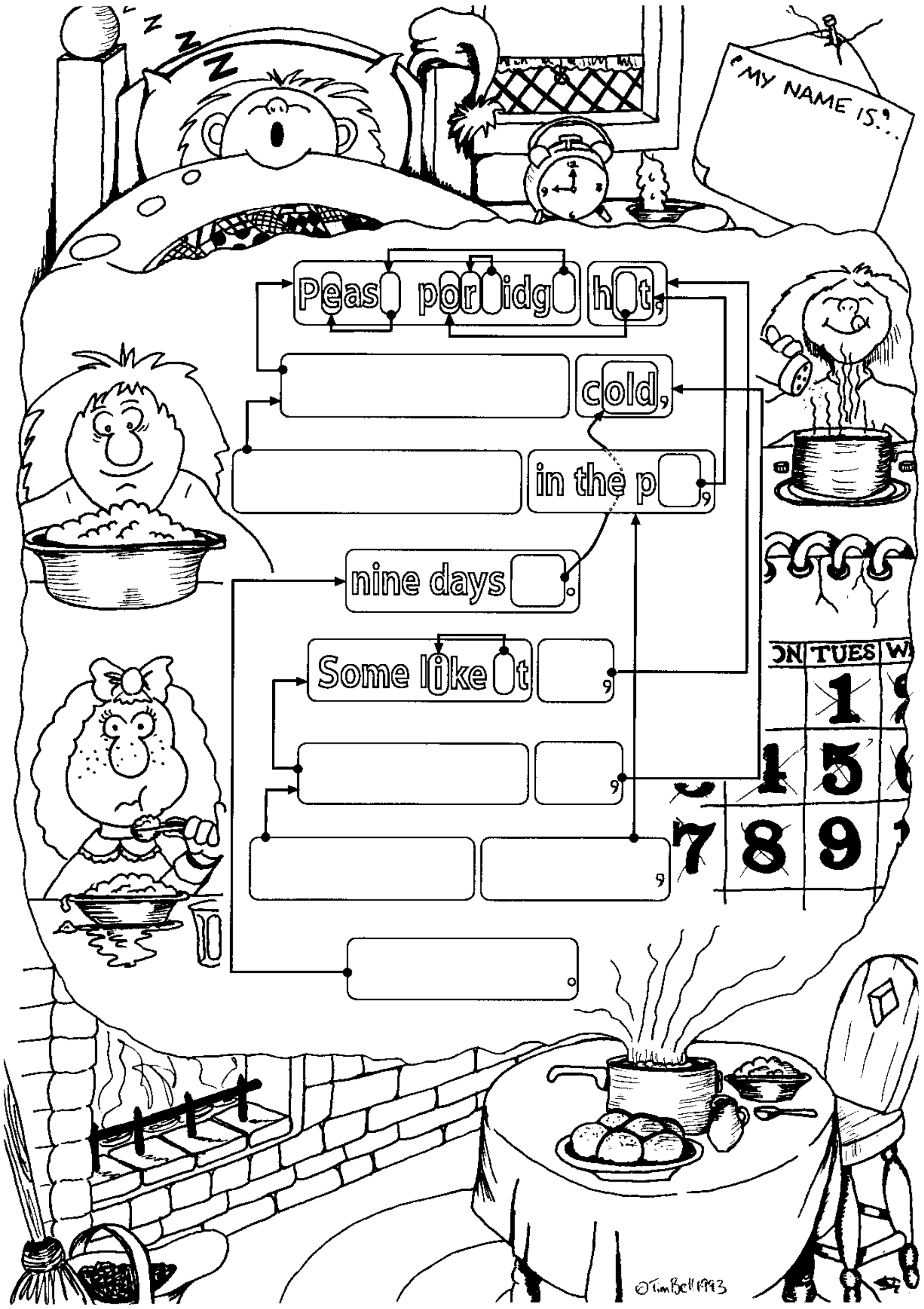
An alternative to buying more storage space is to *compress* the data on hand. This activity illustrates the compression process: the representation of the data is changed so that it takes up less space. The process of compressing and decompressing the data is normally done automatically by the computer, and the user need not even be aware that it is being done except that they might notice that the disk holds more, and that it takes a little more time to get files from the disk. Essentially some computing time is being traded for storage space. Sometimes the system can even be faster using compression because there is less to read from the disk, and this more than compensates for the small amount of time needed to decompress the material.

Many methods of compression have been invented. The principle of pointing to earlier occurrences of chunks of text is a popular technique, often referred to as "Ziv-Lempel coding," or LZ coding, after two Israeli professors who published several important papers in the 1970s and 1980s about this kind of compression. It is very effective because it adapts to whatever sort of data is being coded—it is just as suitable for Spanish, or even Japanese which uses a completely different alphabet, as it is for English. It even adapts to the subject of the text, since any word that is used more than once can be coded with a pointer. LZ coding will typically halve the size of the data being compressed. It is used in popular archiving programs with names like *Zip* and *ARC*, and in "disk doubling" systems. It is also used in high-speed modems, which are devices that allow you to interact with computers over ordinary telephone lines. Here it reduces the amount of data that needs to be transmitted over the phone line, noticeably increasing the apparent speed of transmission.

Another class of compression methods is based on the idea that frequently occuring letters should have shorter codes than rare ones (Morse code uses this idea.) Some of the best methods (which tend to be slower) use the idea that you can have a good guess at what the next letter will be if you know the last few letters. We will encounter this principle in Activity 5.

Further reading

A comprehensive introduction to compression methods can be found in the books *Text compression* by Bell, Cleary and Witten, and *Managing Gigabytes: Compressing and indexing documents and images* by Witten, Moffat and Bell—although these are aimed at university-level computer science students rather than lay people. If you are interested in computer programming, you will enjoy Mark Nelson's *The data compression book*, which is a practically-oriented guide to the subject. Dewdney's *Turing Omnibus* discusses a technique called "Huffman coding" in a section about text compression.



Instructions: *Fill in each blank box by following the arrow attached to the box and copying the letters in the box that it points to.*