FILE COMPRESSION

Robert P. Webber, Longwood University

If you download material from the Internet, you probably know how excruciating the wait for the download to complete can be, even on a fast connection. Computers can transmit and store enormous amounts of information, but music and video files can be so large that they tax the capabilities of even the largest and fastest systems. **File compression** is a common way to reduce the size of computer files so they can be transmitted quicker and take up less storage. You may have encountered *zip, GIF,* or *MP3* files. Those are standard compression schemes, and there are many others.

For a good overview of file compression, read http://www.howstuffworks.com/file-compression.htm. This article discusses lossless and lossy compression algorithms, and it has a nice example illustrating how the **LZ adaptive dictionary-based algorithm** can reduce the size of a passage of text. Let's look at this algorithm in more detail, beginning with a simple example. Suppose we want to compress a string of 0's and 1's, say

000111101011011111101101011011011110110101011010110110100 .

This is not as unrealistic as it might sound, because computer files, at machine level, consist of strings of 0's and 1's. This string has 57 characters.

Starting on the left, break the string into substrings, putting a space after every string we haven't seen before. The first character is 0, which we haven't seen. Put a blank after it. The second character is also 0. We've seen that, so go on to the third character, another 0. We haven't seen the string 00, so put a blank after it. The next character is 1, which is new, so insert a blank. So far, we have

0 00 1

Continuing, there are 14 substrings.

0 00 1 11 10 101 1011 111 10110 101101 10111 1011010 10110101 10110100

Number the substrings from 1 through E, using hexadecimal to save space. Now consider the last two strings. They both begin with the string 1011010, which is the third string from the end, the string numbered C in hexadecimal. Instead of writing 10110101 10110100, we could write C1 C0, with the understanding that C refers to the twelfth string. Let's organize our work into a **dictionary**.

String	Position number (hex)	Prefix	Prefix position number	Coded string
0	1	None	0	00
00	2	0	1	10
1	3	None	0	01
11	4	1	3	31
10	5	1	3	30
101	6	10	5	31
1011	7	101	6	61
111	8	11	4	41
10110	9	1011	7	70
101101	A	10110	9	91
10111	B	1011	7	71
1011010	C	101101	A	A0
10110101	D	1011010	C	C1
10110100	E	1011010	C	C0

In the coded strings, prefix position 0 denotes the empty string. The entire coded string is

0010013130316141709171A0C1C0

This string has 28 characters, less than half the length of the original string. The **compression ratio,** which is the ratio of the original length to the coded length, is

57 / 28 = 2.04

LZ compression works best when there are a lot of repeated patterns in the data. It is less effective when the data is random. Fortunately, patterns do tend to occur often in ordinary text. For large text files, reductions in size on the order of 50% are common.

To decode, we would need to know the first two columns of the table. More sophisticated versions of the LZ algorithm build a shorter dictionary which increases the compression, even when the dictionary itself is counted. The logic is pretty complicated, but the *HowStuffWorks* article indicates how it might be done.

LZ compression is a **lossless** algorithm. That is, the compressed version preserves all the information in the original, and decoding the compressed message restores the exact original. This is important in text. Dropping letters or words from a passage of text might radically change the meaning, or even make it unintelligible!

Music files typically have a lot of information that isn't necessary, however. The human ear has difficulty distinguishing between two tones that are practically the same. There are some sounds that we hear much better than others, and there are many sounds that we cannot hear at all. Compression algorithms such as MP3 use facts like these to eliminate certain parts of songs without appreciably hurting the quality of the sound for the listener. These algorithms are **lossy** in the sense that the original is lost when the compression is done. However, the human ear cannot tell the difference! Compression ratios on the order of 10:1 are common using the MP3 method. This is what allows us to save thousands of songs on a zip drive.

Video files are compressed in much the same way. The human eye cannot detect subtle differences in shade. Entire parts of a picture may look the same to the naked eye – a blue sky, for example. Instead of storing information for each distinct pixel in the picture, we could store information on only one color and the location and number of pixels to which it applies. If the scheme works well, the human viewer won't notice the change, but the file size will be greatly reduced. GIF and JPG are video compression algorithms.

For an overview of lossy algorithms, see http://computer.howstuffworks.com/mp3.htm.

Specific information on audio storage

CDs store music in an uncompressed format. When a song is digitally recorded, the audio is sampled $44{,}100$ times a second. Each sample is 16 bits (2 bytes), and separate samples are taken for the left and right stereo channels. Thus for each second of music, a CD stores

$$44{,}100\frac{samples}{\sec ond}*2\frac{bytes}{sample}*2$$
$$=176{,}400\frac{bytes}{\sec ond}=176.4K\frac{bytes}{\sec ond}$$

for both channels. A CD holds about 650 Mbytes, so it can store approximately

$$\frac{650\ Mbytes}{176.4\ Kbytes/second}=\frac{650{,}000\ Kbytes}{176.4\ Kbytes/second}=3684.8\,\text{seconds}$$

or about 61 minutes of music per CD.

Now think about how long it would take to download a song over the Internet. Suppose a song is three minutes long. We would have to transmit

$$176.4K\frac{bytes}{second}*60\frac{second}{minute}*3\,minutes=31{,}752Kbytes$$

per three minute song.

When people started to download music, the fastest download modem speed was $56Kbits$ (not bytes!) per second. No DSL or fiber optics lines were available then. Our three minute song would require

$$\frac{31,752\ Kbytes}{56Kbits\ /\ second} * \frac{8\ bits}{byte} = 4536\ \text{seconds}$$
$$= \frac{4536\ seconds}{60\ seconds\ /\ minute} = 75.6\ \text{minutes}$$

or 1 hour 15.6 minutes for a single song! Clearly, this was unacceptably slow.

Fortunately, MP3 compression was available, and it typically provided compression ratios of between 10:1 and 14:1 without noticeably affecting the audio quality. Our $32,752\ Kbyte \approx 32\ Mbyte$ file will compress down to about $3\ Mbytes$, which reduces the transmission time to a few minutes on a $56K$ modem and seconds on a DSL line. At a 12:1 compression ratio, for instance, our three minute song could be downloaded in

$$75.6\ minutes\ /\ 12 = 6.3\ \text{minutes}$$

on a $56K$ modem, or 50.8 seconds on a 5 *Gbit/second* DSL line. Data compression makes downloading music and video files practical.

EXERCISES

In exercises 1 through 3, use LZ compression to

- Obtain the substrings;
- Write out the dictionary for the string;
- Write the compressed string;
- Find the compression ratio.

1. 001000001110110001100111010010010100100010010010100111010

2. 001000001111111110010000000000011111110000000001111000111100000011

3. abbaaaabbbababbbbabbbaabbaaabbbaaabbbaabbaabbb

4. A lossy compression scheme for text that has achieved some notoriety in the media is to omit all vowels from the middle of words. The result is often understandable by humans. For instance, can you comprehend the following sentence?

It ws a drk and strmy nght. Sddnly a sht rng ot!

We informally use this scheme in text messages and vanity license plates. Use it to compress the following passages, and compute the compression ratio for each. Are the compressed versions understandable?

Ask not what your country can do for you – ask what you can do for your country.

Bill Gates once said that nobody would ever need more than 64K bytes of memory in a computer.

People often use informal compression schemes when composing text messages on a wireless phone. For example, “where are you?” might be compressed as “wr r u?” In exercises 5 through 8, express the phrase as you might write it as a text message and calculate the compression ratio.

5. are you coming to the party this weekend?

6. at the baseball game with Billy

7. see you tomorrow night at eight

8. please pick up cookies, chips, and French onion dip

A music file stores (in standard, uncompressed form) about 176.4 Kbytes of data for each second of music. Use this information in problems 9 through 18.

9. How many bits (not bytes) of data are needed for a 2 ½ minute song?

10. Suppose we transmit the 2 ½ minute song using a 56 Kbit per second modem. (That is, the modem can send 56,000 bits of data each second.) How long will it take to send the entire song?

11. How long would it take to transmit the song over a DSL line that transmits data at the rate of 2 Mbits per second?

12. Now suppose we compress our 2 ½ minute song using MP3 compression with a 10:1 compression ratio. How long would it take to transmit the compressed file using a 56 Kbit modem? a 2 Mbit DSL line? a 15 Mbit fiber optics line?

Exercises 13 through 15 use Don McLean's classic song <u>American Pie</u>, which is 8 minutes 30 seconds long.

13. Confirm that it would require approximately 90 Mbytes of memory to store this song in uncompressed format.

14. How long would it take to download the uncompressed file using
    a. an old 14.4 Kbits/second modem?
    b. a 56 Kbits/second modem?
    c. a DSL line transmitting at 5 Mbits/second?
    d. a fiber optics line transmitting at 15 Mbits/second?

15. Same as Problem 14, except the file is compressed using MP3 with a compression ratio of 12:1 before transmitting.

Exercises 16 through 18 use Queen's classic song <u>We Will Rock You</u>, which is 2 minutes 15 seconds long.

16. Confirm that it would require approximately 23.8 Mbytes of memory to store this song in uncompressed format.

17. How long would it take to download the uncompressed file using
    a. a 56 Kbits/second modem?
    b. a 3 Mbits/second cable modem line?
    c. a 20 Mbits/second fiber optics line?

18. Same as Problem 17, except the file is compressed using MP3 with a compression ratio of 14:1 before transmitting.

19. Assume that a two hour movie requires 226.8 Gbytes of storage in uncompressed format. How long would it take to download the uncompressed file using

a. a 56 Kbits/second modem?
b. a 10 Gbits/second cable modem?
c. a 18 Gbits/second fiber optics line?

20. Compression ratios as high as 50:1 are possible with MPEG4. Repeat Exercise 19 assuming the file has been compressed at 50:1 before downloading.

21. Use the logic in the *HowStuffWorks* article to determine a suitable dictionary and coded form for the string:

The rain in Spain falls mainly in the plain. Where does it rain? On the plain! On the plain! Where's the plain? In Spain! In Spain!