Lecture №1.8.
Methods and types of archiving. Main algorithms for data compression.

A file archiver is a computer program that combines a number of files together into one archive file, or a series of archive files, for easier transportation or storage. Many file archivers employ Archive formats that provide lossless data compression to reduce the size of the archive which is often useful for transferring a large number of individual files over a high latency network like the Internet.

The most basic archivers just take a list of files and concatenate their contents sequentially into the archive. In addition the archive must also contain some information about at least the names and lengths of the originals, so that proper reconstruction is possible. Most archivers also store metadata about a file that the operating system provides, such as timestamps, ownership and access control.

The process of making an archive file is called ***archiving*** or ***packing***. Reconstructing the original files from the archive is termed ***unarchiving***, ***unpacking*** or ***extracting***.

An archive file is a file that is composed of one or more files along with metadata that can include source volume and medium information, file directory structure, error detection and recovery information, file comments, and usually employs some form of lossless compression. Archive files may also be encrypted in part or as a whole. Archive files are used to collect multiple data files together into a single file for easier portability and storage.

Computer archive files are created by File archiver software, Optical disc authoring software, or Disk image software that uses an archive format determined by that software. The file extension or file header of the archive file are indicators of the file format used.

Archive files are sometimes accompanied by separate parity archive (PAR) files that allow for additional error detection and recovery, particularly in recovery of missing package files in a multi-file archive.

Archives can have extensions like ***.zip, .rar, .tar*** and etc.

Lossless data compression is a class of data compression algorithms that allows the exact original data to be reconstructed from the compressed data. The term lossless is in contrast to lossy data compression, which only allows an approximation of the original data to be reconstructed, in exchange for better compression rates.

Lossless data compression is used in many applications. For example, it is used in the popular ZIP file format and in the Unix tool gzip. It is also often used as a component within lossy data compression technologies.

Lossless compression is used when it is important that the original and the decompressed data be identical, or when no assumption can be made on whether certain deviation is uncritical. Typical examples are executable programs and source code. Some image file formats, like PNG or GIF, use only lossless compression, while others like TIFF and MNG may use either lossless or lossy methods.

Lossless compression techniques

Most lossless compression programs do two things in sequence: the first step generates a statistical model for the input data, and the second step uses this model to map input data to bit sequences in such a way that "probable" (e.g. frequently encountered) data will produce shorter output than "improbable" data.

The primary encoding algorithms used to produce bit sequences are Huffman coding and arithmetic coding. Arithmetic coding achieves compression rates close to the

best possible for a particular statistical model, which is given by the information entropy, whereas Huffman compression is simpler and faster but produces poor results for models that deal with symbol probabilities close to 1.

There are two primary ways of constructing statistical models: in a static model, the data is analyzed and a model is constructed, then this model is stored with the compressed data. This approach is simple and modular, but has the disadvantage that the model itself can be expensive to store, and also that it forces a single model to be used for all data being compressed, and so performs poorly on files containing heterogeneous data. Adaptive models dynamically update the model as the data is compressed. Both the encoder and decoder begin with a trivial model, yielding poor compression of initial data, but as they learn more about the data performance improves. Most popular types of compression used in practice now use adaptive coders.

Lossless compression methods may be categorized according to the type of data they are designed to compress. While, in principle, any general-purpose lossless compression algorithm (general-purpose meaning that they can compress any bitstring) can be used on any type of data, many are unable to achieve significant compression on data that is not of the form for which they were designed to compress. Many of the lossless compression techniques used for text also work reasonably well for indexed images.

Text

Statistical modeling algorithms for text (or text-like binary data such as executables) include:

- Context Tree Weighting method (CTW)
- Burrows-Wheeler transform (block sorting preprocessing that makes compression more efficient)
- LZ77 (used by DEFLATE)
- LZW

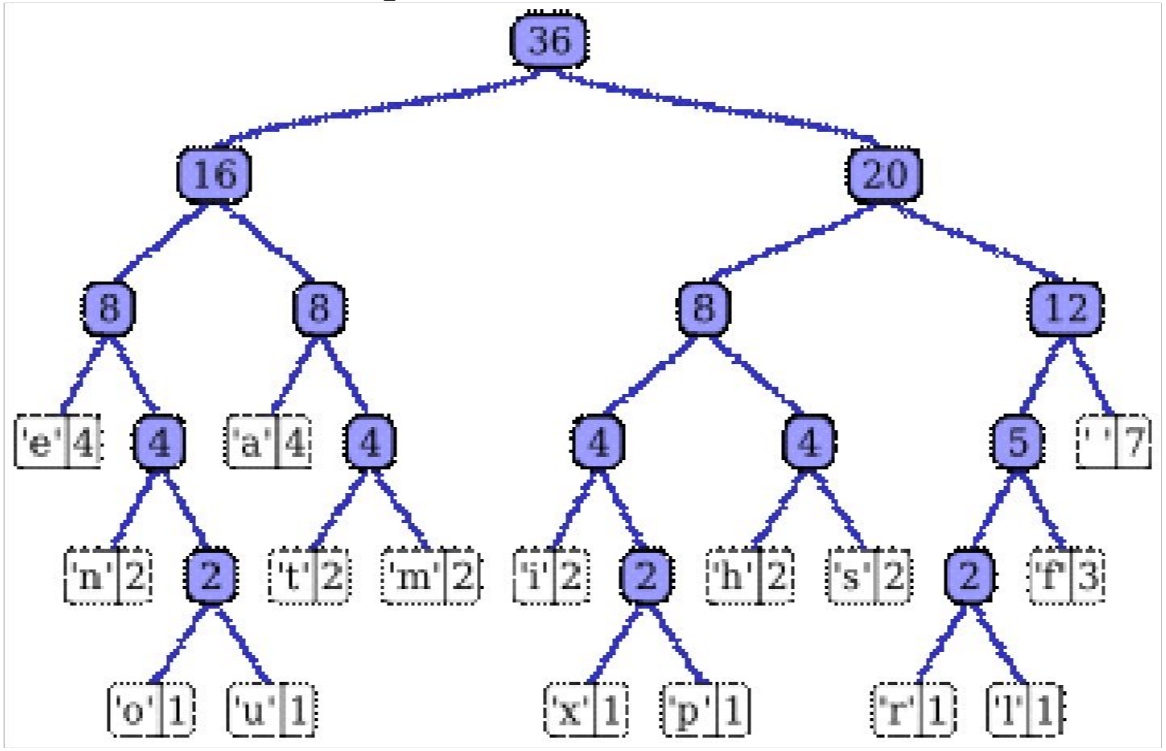
Multimedia

Techniques that take advantage of the specific characteristics of images such as the common phenomenon of contiguous 2-D areas of similar tones. Every pixel but the first is replaced by the difference to its left neighbor. This leads to small values having a much higher probability than large values. This is often also applied to sound files and can compress files which contain mostly low frequencies and low volumes. For images this step can be repeated by taking the difference to the top pixel, and then in videos the difference to the pixel in the next frame can be taken.

A hierarchical version of this technique takes neighboring pairs of data points, stores their difference and sum, and on a higher level with lower resolution continues with the sums. This is called discrete wavelet transform. JPEG2000 additionally uses data points from other pairs and multiplication factors to mix then into the difference. These factors have to be integers so that the result is an integer under all circumstances. So the values are

The adaptive encoding uses the probabilities from the previous sample in sound encoding, from the left and upper pixel in image encoding, and additionally from the previous frame in video encoding. In the wavelet transformation the probabilities are also passed through the hierarchy.

Huffman coding



Huffman tree generated from the exact frequencies of the text "this is an example of a huffman tree". The frequencies and codes of each character are below. Encoding the sentence with this code requires 135 bits, not counting space for the tree.

Char	Freq	Code	Char	Freq	Code
space	7	111	l	1	11001
a	4	010	o	1	00110
e	4	000	p	1	10011
f	3	1101	r	1	11000
h	2	1010	u	1	00111
i	2	1000	x	1	10010
m	2	0111			
n	2	0010			
s	2	1011			
t	2	0110			

In computer science and information theory, Huffman coding is an entropy encoding algorithm used for lossless data compression. The term refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol. It was developed by David A. Huffman while he was a Ph.D. student at MIT, and published in the 1952 paper "A Method for the Construction of Minimum-Redundancy Codes".

Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix code (sometimes called "prefix-free codes") (that is, the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol) that expresses the most common characters using shorter strings of bits than are used

for less common source symbols. Huffman was able to design the most efficient compression method of this type: no other mapping of individual source symbols to unique strings of bits will produce a smaller average output size when the actual symbol frequencies agree with those used to create the code. A method was later found to do this in linear time if input probabilities (also known as weights) are sorted.

For a set of symbols with a uniform probability distribution and a number of members which is a power of two, Huffman coding is equivalent to simple binary block encoding, e.g., ASCII coding. Huffman coding is such a widespread method for creating prefix codes that the term "Huffman code" is widely used as a synonym for "prefix code" even when such a code is not produced by Huffman's algorithm.

Although Huffman coding is optimal for a symbol-by-symbol coding (i.e. a stream of unrelated symbols) with a known input probability distribution, its optimality can sometimes accidentally be over-stated. For example, arithmetic coding and LZW coding often have better compression capability. Both these methods can combine an arbitrary number of symbols for more efficient coding, and generally adapt to the actual input statistics, the latter of which is useful when input probabilities are not precisely known or vary significantly within the stream. In general, improvements arise from input symbols being related (e.g., "cat" is more common than "cta").

Lempel–Ziv–Welch (LZW) is a universal lossless data compression algorithm created by Abraham Lempel, Jacob Ziv, and Terry Welch. It was published by Welch in 1984 as an improved implementation of the LZ78 algorithm published by Lempel and Ziv in 1978. The algorithm is designed to be fast to implement but is not usually optimal because it performs only limited analysis of the data.

Encoding

A dictionary is initialized to contain the single-character strings corresponding to all the possible input characters (and nothing else). The algorithm works by scanning through the input string for successively longer substrings until it finds one that is not in the dictionary. When such a string is found, it is added to the dictionary, and the index for the string less the last character (i.e., the longest substring that is in the dictionary) is retrieved from the dictionary and sent to output. The last input character is then used as the next starting point to scan for

In this way successively longer strings are registered in the dictionary and made available for subsequent encoding as single output values. The algorithm works best on data with repeated patterns, so the initial parts of a message will see little compression. As the message grows, however, the compression ratio tends asymptotically to the maximum.

Decoding

The decoding algorithm works by reading a value from the encoded input and outputting the corresponding string from the initialized dictionary. At the same time it obtains the next value from the input, and registers to the dictionary the concatenation of that string and the first character of the string obtained by decoding the next input value. The decoder then proceeds to the next input value (which was already read in as the "next value" in the previous pass) and repeats the process until there is no more input, at which point the final input value is decoded without any more additions to the dictionary.

In this way the decoder builds up a dictionary which is identical to that used by the encoder, and uses it to decode subsequent input values. Thus the full dictionary does not need be sent with the encoded data; just the initial dictionary containing the single-character strings is sufficient (and is typically defined beforehand within the encoder and decoder rather than being explicitly sent with the encoded data.)

The method became widely used in the program compress, which became a more or less standard utility in Unix systems circa 1986. (It has since disappeared from many for both legal and technical reasons, but as of 2008 at least FreeBSD includes the utility of compress and uncompress as a part of the distribution.) Several other popular compression utilities also used the method, or closely related ones.

It became very widely used after it became part of the GIF image format in 1987. It may also (optionally) be used in TIFF files.

LZW compression provided a better compression ratio, in most applications, than any well-known method available up to that time. It became the first widely used universal data compression method on computers. It would typically compress large English texts to about half of their original sizes.

Today, an implementation of the algorithm is contained within the popular Adobe Acrobat software program.

Run-length encoding (RLE) is a very simple form of data compression in which runs of data (that is, sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run. This is most useful on data that contains many such runs: for example, relatively simple graphic images such as icons, line drawings, and animations. It is not recommended for use with files that don't have many runs as it could potentially double the file size.

JBIG is an image compression standard for bi-level images, developed by the Joint Bi-level Image Experts Group. It is suitable for both lossless and lossy compression. According to a press release from the Group, in its lossless mode JBIG2 typically generates files one third to one fifth the size of Fax Group 4 and one half to one quarter the size of JBIG, the previous bi-level compression standard released by the Group. JBIG2 has been published in 2000 as the international standard ITU T.88, and in 2001 as ISO/IEC 14492.

JBIG is a method for compressing bi-level (two-color) image data. The acronym JBIG stands for Joint Bi-level Image Experts Group, a standards committee that had its origins within the International Standards Organization (ISO). The compression standard they developed bears the name of this committee.

The main features of JBIG are:

- Lossless compression of one-bit-per-pixel image data
- Ability to encode individual bitplanes of multiple-bit pixels
- Progressive or sequential encoding of image data

Lossy compression

A lossy compression method is one where compressing data and then decompressing it retrieves data that is different from the original, but is close enough to be useful in some way. Lossy compression is most commonly used to compress multimedia data (audio, video, still images), especially in applications such as streaming media and internet telephony. By contrast, lossless compression is required for text and data files, such as bank records, text articles, etc. In many cases it is advantageous to make a master lossless file which can then be used to produce compressed files for different purposes; for example a multi-megabyte file can be used at full size to produce a full-page advertisement in a glossy magazine, and a 10 kilobyte lossy copy made for a small image on a web page.

Lossy and lossless compression

It is possible to compress many types of digital data in a way which reduces the amount of information stored, and consequently the size of a computer file needed to store it or the bandwidth needed to stream it, with no loss of information. Take, for example, a picture. It is converted to a digital file by considering it to be an array of dots, and specifying the colour and brightness of each dot. If the picture contains an area of the same colour, it can be compressed without loss by saying "200 red dots" instead of "red dot, red dot, ...(197 more times)..., red dot".

The original contains a certain amount of information; there is a lower limit to the size of file that can carry all the information. As an intuitive example, most people know that a compressed ZIP file is smaller than the original file; but repeatedly compressing the file will not reduce the size to nothing.

In many cases files or data streams contain more information than is needed. For example, a picture may have more detail than the eye can distinguish when reproduced at the largest size intended; an audio file does not need a lot of fine detail during a very loud passage. Developing lossy compression techniques as closely matched to human perception as possible is a complex task. In some cases the ideal is a file which provides exactly the same perception as the original, with as much digital information as possible removed; in other cases perceptible loss of quality is considered a valid trade-off for the reduced data size.

JPEG compression

The compression method is usually lossy, meaning that some original image information is lost and cannot be restored (possibly affecting image quality.) There are variations on the standard baseline JPEG that are lossless; however, these are not widely supported.

There is also an interlaced "Progressive JPEG" format, in which data is compressed in multiple passes of progressively higher detail. This is ideal for large images that will be displayed while downloading over a slow connection, allowing a reasonable preview after receiving only a portion of the data. However, progressive JPEGs are not as widely supported, and even some software which does support them (such as some versions of Internet Explorer) only displays the image once it has been completely downloaded.

There are also many medical imaging systems that create and process 12-bit JPEG images. The 12-bit JPEG format has been part of the JPEG specification for some time, but again, this format is not as widely supported.

JPEG stands for Joint Photographic Experts Group, which is a standardization committee. It also stands for the compression algorithm that was invented by this committee.

There are two JPEG compression algorithms: the oldest one is simply referred to as ‘JPEG’ within this page. The newer JPEG 2000 algorithm is discussed on a separate page. Please note that you have to make a distinction between the JPEG compression algorithm, which is discussed on this page, and the corresponding JFIF file format, which many people refer to as JPEG files and which is discussed in the file format section.

JPEG is a lossy compression algorithm that has been conceived to reduce the file size of natural, photographic-like true-color images as much as possible without affecting the quality of the image as experienced by the human sensory engine. We perceive small changes in brightness more readily than we do small changes in color. It is this aspect of our perception that JPEG compression exploits in an effort to reduce the file size

Thank you for your attention