Maharaja Agrasen Institute of Technology
ETCS 211
Computer Graphics & Multimedia

Data Compression

Why Data Compression?

Make optimal use of limited storage space

Save time and help to optimize resources
- If compression and decompression are done in I/O processor, less time is required to move data to or from storage subsystem, freeing I/O bus for other work
- In sending data over communication line: less time to transmit and less storage to host

Data Compression- Entropy

Entropy is the measure of information content in a message.
- Messages with higher entropy carry more information than messages with lower entropy.

How to determine the entropy
- Find the probability *p(x)* of symbol ***x*** in the message
- The entropy *H(x)* of the symbol x is:

$$H(x) = - p(x) \cdot \log_2 p(x)$$

The average entropy over the entire message is the sum of the entropy of all n symbols in the message

Data compression implies sending or storing a smaller number of bits. Although many methods are used for this purpose, in general these methods can be divided into two broad categories: lossless and lossy methods.

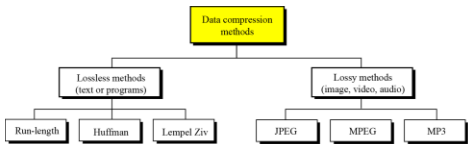

Data compression methods
Lossless methods (text or programs)
Lossy methods (image, video, audio)
Run-length
Huffman
Lempel Ziv
JPEG
MPEG
MP3

Figure 15.1 Data compression methods

LOSSLESS COMPRESSION

In lossless data compression, the integrity of the data is preserved. The original data and the data after compression and decompression are exactly the same because, in these methods, the compression and decompression algorithms are exact inverses of each other: no part of the data is lost in the process. Redundant data is removed in compression and added during decompression. Lossless compression methods are normally used when we cannot afford to lose any data.

INTRODUCTION TO DATA COMPRESSION:LOSSY AND LOSSLESS ALGORITHMS

Data compression is the process of modifying, encoding or converting the bits structure of data in such a way that it consumes less space on disk.

It enables reducing the storage size of one or more data instances or elements. Data compression is also known as source coding or bit-rate reduction.

Data compression enables sending a data object or file quickly over a network or the Internet and in optimizing physical storage resources.

Data compression has wide implementation in computing services and solutions, specifically data communications. Data compression works through several compressing techniques and software solutions that utilize data compression algorithms to reduce the data size.

A common data compression technique removes and replaces repetitive data elements and symbols to reduce the data size. Data compression for graphical data can be lossless compression or lossy compression, where the former saves all replaces but save all repetitive data and the latter deletes all repetitive data.

DATA COMPRESSION CAN BE LOSSY OR LOSSLESS

LOSSY COMPRESSION

Lossy is a data encoding and compression technique that deliberately discards some data in the compression process. The lossy compression method filters and discards needless and redundant data to reduce the amount of data being compressed and later being executed on a computer.

Lossy is derived from the word loss, which defines the primary objective of this technique.

LOSSLESSCOMPRESSION

Lossless compression involves compressing data in such a way that the original data set is fully reconstructed upon reversal of compression. This is in contrast to "lossy" compression, where some data may be lost in the reversal process.

Lossless compression is also known as lossless audio compression.

RUNLENGTH ENCODING , HUFFMAN ENCODING

HUFFMAN ENCODING

Prefix Codes, means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of code assigned to any other character. This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bitstream.

Let us understand prefix codes with a counter example. Let there be four characters a, b, c and d, and their corresponding variable length codes be 00, 01, 0 and 1. This coding leads to ambiguity because code assigned to c is the prefix of codes assigned to a and b. If the compressed bit stream is 0001, the de-compressed output may be “cccd” or “ccb” or “acd” or “ab”.

There are mainly two major parts in Huffman Coding
1) Build a Huffman Tree from input characters.
2) Traverse the Huffman Tree and assign codes to characters.

Steps to build Huffman Tree
Input is an array of unique characters along with their frequency of occurrences and output is Huffman Tree.

1. Create a leaf node for each unique character and build a min heap of all leaf nodes (Min Heap is used as a priority queue. The value of frequency field is used to compare two nodes in min heap. Initially, the least frequent character is at root)

2. Extract two nodes with the minimum frequency from the min heap.

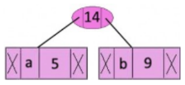
3. Create a new internal node with a frequency equal to the sum of the two nodes frequencies. Make the first extracted node as its left child and the other extracted node as its right child. Add this node to the min heap.

4. Repeat steps#2 and #3 until the heap contains only one node. The remaining node is the root node and the tree is complete.

Let us understand the algorithm with an example:

character	Frequency
a	5
b	9
c	12
d	13
e	16
f	45

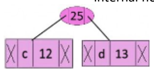
Step 1. Build a min heap that contains 6 nodes where each node represents root of a tree with single node.
Step 2 Extract two minimum frequency nodes from min heap. Add a new internal node with frequency 5 + 9 = 14.



Now min heap contains 5 nodes where 4 nodes are roots of trees with single element each, and one heap node is root of tree with 3 elements

character	Frequency
c	12
d	13
Internal Node	14
e	16
f	45

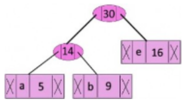
Step 3: Extract two minimum frequency nodes from heap. Add a new internal node with frequency 12 + 13 = 25



Now min heap contains 4 nodes where 2 nodes are roots of trees with single element each, and two heap nodes are root of tree with more than one nodes.

character	Frequency
Internal Node	14
e	16
Internal Node	25
f	45

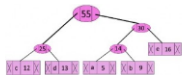
Step 4: Extract two minimum frequency nodes. Add a new internal node with frequency 14 + 16 = 30



Now min heap contains 3 nodes.

```
character                         Frequency
Internal Node                      25
Internal Node                      30
f                                  45
```

Step 5: Extract two minimum frequency nodes. Add a new internal node with frequency 25 + 30 = 55



Now min heap contains 2 nodes.

```
character                         Frequency
f                                   45
Internal Node                       55
```

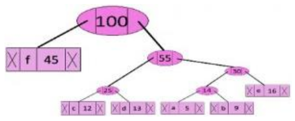
Step 6: Extract two minimum frequency nodes. Add a new internal node with frequency 45 + 55 = 100

Now min heap contains only one node.

```
Character                        Frequency
Internal Node                       100
```

Since the heap contains only one node, the algorithm stops here.

Steps to print codes from Huffman Tree:
Traverse the tree formed starting from the root.
Maintain an auxiliary array.
While moving to the left child, write 0 to the array. While moving to the right child, write 1 to the array. Print the array when a leaf node is encountered.



The codes are as follows:

character code	-word
f	0
c	100
d	101
a	1100
b	1101
e	111

RUNLENGTH ENCODING

Run-length encoding (**RLE**) is a form of lossless data compression in which **runs** of data (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**.

Given an input string, write a function that returns the Run Length Encoded string for the input string.

For example, if the input string is “wwwwaaadexxxxxx”, then the function should return “w4a3d1e1x6”.

a) Pick the first character from source string.
b) Append the picked character to the destination string.
c) Count the number of subsequent occurrences of the picked character and append the count to destination string.
d) Pick the next character and repeat steps b) c) and d) if end of string is NOT reached.

Run-length encoding (RLE) is a very simple form of data compression in which a stream of data is given as the input (i.e. "AAABBCCCC") and the output is a sequence of counts of consecutive data values in a row (i.e. "3A2B4C").

This type of data compression is lossless, meaning that when decompressed, all of the original data will be recovered when decoded. Its simplicity in both the encoding (compression) and decoding (decompression) is one of the most attractive features of the algorithm.

Here you can see a simple example of a stream ("run") of data in its original form and encoded form:

Input data:

AAAAAAFDDCCCCCCCAEEEEEEEEEEEEEEEEE

Output data:

6A1F2D7C1A17E In this example we were able to compress the data from 34 characters down to 13.

As you may have noticed, the more consecutive values in a row, the more space we save in the resulting compression. On the other hand, if you have a sequence of data that frequently changes between values (i.e. "BEFEFADED") then we won't save much space at all. In fact, we could even increase the size of our data since a single instance of a character results in 2 characters (i.e. "A" becomes "1A") in the output of the encoding.

Because of this, RLE is only good for certain types of data and applications. For example, the Pixy camera, which is a robotics camera that helps you easily track objects, uses RLE to compress labeled video data before transferring it from the embedded camera device to an external application. Each pixel is given a label of "no object", "object 1", "object 2", etc. This is the perfect encoding for this application because of its simplicity, speed, and ability to compress the low-entropy label data.

LEMPEL ZIV ENCODING, ROLE OF DICTIONARY IN ENCODING AND DECODING

The LZW algorithm is a very common compression technique. This algorithm is typically used in GIF and optionally in PDF and TIFF. Unix’s ‘compress’ command, among other uses. It is lossless, meaning no data is lost when compressing. The algorithm is simple to implement and has the potential for very high throughput in hardware implementations. It is the algorithm of the widely used Unix file compression utility compress, and is used in the GIF image format.

How does it work?
LZW compression works by reading a sequence of symbols, grouping the symbols into strings, and converting the strings into codes. Because the codes take up less space than the strings they replace, we get compression.

Characteristic features of LZW includes,
- LZW compression uses a code table, with 4096 as a common choice for the number of table entries. Codes 0-255 in the code table are always assigned to represent single bytes from the input file.
- When encoding begins the code table contains only the first 256 entries, with the remainder of the table being blanks. Compression is achieved by using codes 256 through 4095 to represent sequences of bytes.
- As the encoding continues, LZW identifies repeated sequences in the data, and adds them to the code table.
- Decoding is achieved by taking each code from the compressed file and translating it through the code table to find what character or characters it represents.

Advantages of LZW over Huffman:
LZW requires no prior information about the input data stream.
LZW can compress the input stream in one single pass.
Another advantage of LZW its simplicity, allowing fast execution.

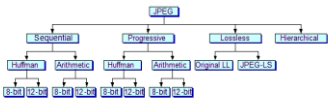
JPEG STANDARDS FOR COMPRESSING STILL IMAGES

The lossy JPEG

JPEG is an acronym for Joint Photographic Experts Group and JFIF is an acronym JPEG File Interchange Format. Normally, the files with the .JPG extension are JFIF files. If a JPEG file is loaded into a text editor such as Window's NotePad , the string JFIF is started in the first line starting at the 7th character. JPEG compression is mainly discards much of the original information so the exact original image cannot be reconstructed from a JPEG file. This is called lossy compression. While this sounds bad, a photograph actually contains considerable information that the human eye cannot detect so this can be safely discarded.

JPEG Compression Modes

The JPEG standard defined four compression modes: Hierarchical, Progressive, Sequential and lossless. Figure 0 shows the relationship of major JPEG compression modes and encoding processes.



Sequential: Sequential-mode images are encoded from top to bottom. Sequential mode supports sample data with 8 and 12 bits of precision. In the sequential JPEG, each color component is completely encoded in single scan. Within sequential mode, two alternate entropy encoding processes are defined by the JPEG standard: one uses Huffman encoding; the other uses arithmetic coding.
Progressive: In progressive JPEG images, components are encoded in multiple scans. The compressed data for each component is placed in a minimum of 2 and as many as 896 scans. The initial scans create a rough version of the image, while subsequent scans refine it.
Lossless: preserves exact, original image, small compression ration, less use
Hierarchical: JPEG is a super-progressive mode in which the image Is broken down into a number of subimages called frames. A frame is a collection of one or more scans. In hierarchical mode, the first frame creates a low-resolution version of image. The remaining frames refine the image by increasing the solution.

Baseline JPEG

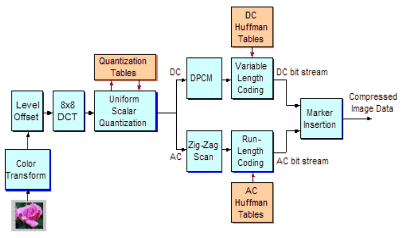
Baseline JPEG decompressor supports a minimal set of features. It must be able to decompress image using sequential DCT-based mode. For baseline compression the bit depth must be *B*=8; however, it is convenient to describe a more general situation. The image samples are assumed to be unsigned quantities in the range [0, 2^B-1]. The level offset subtract 2^(B-1) from every sample value so as to produce signed quantities in the range [-2^(B-1), 2^(B-1)-1]. The purpose of this is to ensure that all the DCT coefficients will be signed quantities with a similar dynamic range. The image is partitioned into blocks of size 8×8. Each block is then independently transformed using the 8×8 DCT. If the image dimensions are not exact multiples of 8, the blocks on the lower and right hand boundaries may be only partially occupied. These boundary blocks must be padded to the full 8×8 block size and processed in an identical fashion to every other block. The compressor is free to select the value used to pad partial boundary blocks.

Color Space

The VGA card displays colors by setting the intensity of the three colors RED, BLUE and GREEN. The three colors form the axis of a cartesian coordinate system. Any color is then a point in this color space. A grayscale image is formed by only using points in color space where RED, BLUE and GREEN intensities are all equal. The alternate set of axis so called luminance and chrominance. The luminance axis is labelled Y, the blue chrominance axis is labelled U and the red chrominance axis is labelled V. The three new axes we have created form the three components used in JPEG image files. The following formulas will convert between the two coordinate systems.

Y = 0.299 R + 0.587 G + 0.114 B U = -0.1687 R - 0.3313 G + 0.5 B + 128 V = 0.5 R - 0.4187 G - 0.0813 B + 128 R = Y + 1.402 (V-128) G = Y - 0.34414(U-128) - 0.71414(V-128) B = Y + 1.772 (U-128)

here is a reason for using the YUV color space. The human eye is more sensitive to luminance than to chrominance. Typically JPEGs throw out 3/4 of the chrominance information before any other compression takes place. This reduces the amount of information to be stored about the image by 1/2. With all three components fully stored, 4 pixels needs 3 x 4 = 12 component values. If 3/4 of two components are discarded we need 1 x 4 + 2 x 1 = 6 values.
The largest horizontal and vertical sample factors determine the height and width of the Minimum Coded Unit (MCU) respectively. For the case of figure 4 the MCU would be two 8×8 blocks high and two 8×8 blocks wide for a total of four 8×8 blocks. The blocks are stored in the file all Ys first then all Us then all Vs. For this case one MCU would contain four Y 8×8 blocks followed by one U 8×8 block and one V 8×8



MPEG STANDARD FOR COMPRESSING VIDEO

The name MPEG is an acronym for Moving Pictures Experts Group. MPEG is a method for video compression, which involves the compression of digital images and sound, as well as synchronization of the two.

There currently are several MPEG standards.

MPEG-1 is intended for intermediate data rates, on the order of 1.5 Mbit/sec.

MPEG-2 is intended for high data rates of at least 10 Mbit/sec.

MPEG-3 was intended for HDTV compression but was found to be redundant and was merged with MPEG-2.

MPEG-4 is intended for very low data rates of less than 64 Kbit/sec.

i. In principle, a motion picture is a rapid flow of a set of frames, where each frame is an image. In other words, a frame is a spatial combination of pixels, and a video is a temporal combination of frames that are sent one after another.

ii. Compressing video, then, means spatially compressing each frame and temporally compressing a set off names.

iii. Spatial Compression: The spatial compression of each frame is done with JPEG (ora modification of it). Each frame is a picture that can be independently compressed.

iv. Temporal Compression: In temporal compression, redundant frames are removed.

v. To temporally compress data, the MPEG method first divides frames into three categories:

vi. I-frames, P-frames, and B-frames. Figure1 shows a sample sequence off names.

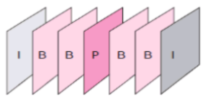
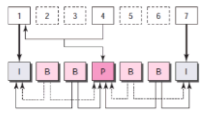


Fig1: MPEG frames

vii. Figure2 shows how I-, P-, and B-frames are constructed from a series of seven frames.



I-frames: An intracoded frame (I-frame) is an independent frame that is not related to any other frame.

They are present at regular intervals. An I-frame must appear periodically to handle some sudden change in the frame that the previous and following frames cannot show. I-frames are independent of other frames and cannot be constructed from other frames.

P-frames: A predicted frame (P-frame) is related to the preceding I-frame or P-frame. In other words, each P-frame contains only the changes from the preceding frame. P-frames can be constructed only from previous I- or P-frames. P-frames carry much less information than other frame types and carry even fewer bits after compression.

B-frames: A bidirectional frame (B-frame) is related to the preceding and following I-frame or P-frame. In other words, each B-frame is relative to the past and the future. Note that a B-frame is never related to another B-frame.

The luminance component contains the gray scale picture & the chrominance components provide the color, hue & saturation.

The MPEG decoder has three parts, audio layer, video layer, system layer.

The system layer reads and interprets the various headers in the source data and transmits this data to either audio or video layer.

The basic building block of an MPEG picture is the macro block as shown:

The macro block consist of 16×16 block of luminance gray scale samples divided into four 8×8 blocks of chrominance samples.

The MPEG compression of a macro block consist of passing each of the °6 blocks their DCT quantization and entropy encoding similar to JPEG.

A picture in MPEG is made up of slices where each slice is continuous set of macro blocks having a similar gray scale component.

The concept of slice is important when a picture contains uniform areas.

The MPEG standard defines a quantization stage having values (1, 31). Quantization for intra coding is:

QDCT =(16×DCT)+sign(DCT)×quantizer scale)2×quantizer - scale×θ

Where

DCT = Discrete cosine transform of the coefficienting encoded

Q = Quantization coefficient from quantization table

Quantization rule for encoding,

QDCT =16×DCT2×quantizer - scale×θQDCT =16×DCT2×quantizer - scale×θ

The quantized numbers Q_(DCT)are encoded using non adaptive Haffman method and the standard defines specific Haffman code tables which are calculated by collecting statistics.

VIVA VOCE

1. What is the difference between lossless and lossy compression techniques?

2. What are the advantages of huffman encoding?

3. Why are compression techniques used?

4. Define compression & decompression?

5. What is the need for compression?

6 What are the different types of compression available? List the difference between them

7. Write down the difference between jpeg2 and jpeg4

8. List the various compression techniques

9. List the benefits of DCT

10 Define quantization

11. Differentiate MPEP2 and MPEG 4

12. Write short notes on jpeg
13. Discuss the various levels of RAID technologies
14. Write short notes on mpeg
15. Write notes on file format standard
16. Explain the multimedia I/O technologies
17. With a neat diagram explain the architecture of MPEG standard
18. What is MIDI? Explain how it is used for music recording