BENC 3443
Multimedia Technology & Application

Chapter 4: Multimedia Data (Part 2)

Outline

- Compression Classification
- History of Compression
- Lossless Compression
  - Run-Length Encoding, Entropy Coding, Huffman Coding, Arithmetic Coding
- Lossy Compression
  - JPEG, MPEG, MP3
- Examples of Compression Algorithms & Codecs

Compression Classification

- **Spatial Compression**
  - Finds similarities in an image and compresses those similarities in a smaller form
  - Intra-frame
- **Temporal Compression**
  - Finds similarities across images and compresses those similarities in a smaller form
  - Inter-frame
- **Quality of Compression**
  - Lossless
  - Lossy

Types of Compression

- digital media files are usually very large, and they need to be made smaller
- won't have the storage capacity
- won't be able to communicate them across networks without overly taxing the patience of the recipients
- don't want to sacrifice the quality

Is it possible to reduce the size of digital media files with no significant loss of quality?

Types of Compression

Lossless compression

-no information is lost between the compression and decompression steps

-reduces the file size to fewer bits. Then decompression restores the data values to exactly what they were before the compression.

Lossy compression

-sacrifice some information

-information lost is generally not important to human perception

History of Data Compression

- The late 40's were the early years of Information Theory, the idea of developing efficient new coding methods was just starting to be fleshed out. Ideas of entropy, information content and redundancy were explored.
- One popular notion held that if the probability of symbols in a message were known, there ought to be a way to code the symbols so that the message will take up less space.
- The first well-known method for compressing digital signals is now known as Shannon- Fano coding. Shannon and Fano [~1948] simultaneously developed this algorithm which assigns binary codewords to unique symbols that appear within a given data file.
- While Shannon-Fano coding was a great leap forward, it had the unfortunate luck to be quickly superseded by an even more efficient coding system : Huffman Coding.
- The Huffman coding [1952] shares most characteristics of Shannon-Fano coding.

History of Data Compression

- Huffman coding could perform effective data compression by reducing the amount of redundancy in the coding of symbols.
- It has been proven to be the most efficient fixed-length coding method available.
- In the last fifteen years, Huffman coding has been replaced by arithmetic coding.
- Arithmetic coding bypasses the idea of replacing an input symbol with a specific code.
- It replaces a stream of input symbols with a single floating-point output number.
- More bits are needed in the output number for longer, complex messages.

UTeM
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Terminology

- **Compressor–Software (or hardware)** device that compresses data
- **Decompressor–Software (or hardware)** device that decompresses data
- **Codec–Software (or hardware) device that** compresses and decompresses data
- **Algorithm–The logic that governs the** compression/decompression process

Compression Ratio

Compression is normally **measured** with the compression ratio :

$$\boldsymbol{compression\,ratio} = \frac{\textbf{original size}}{\textbf{Compressed size}} : \mathbf{1}$$

For indefinite file size such as steaming audio or video:

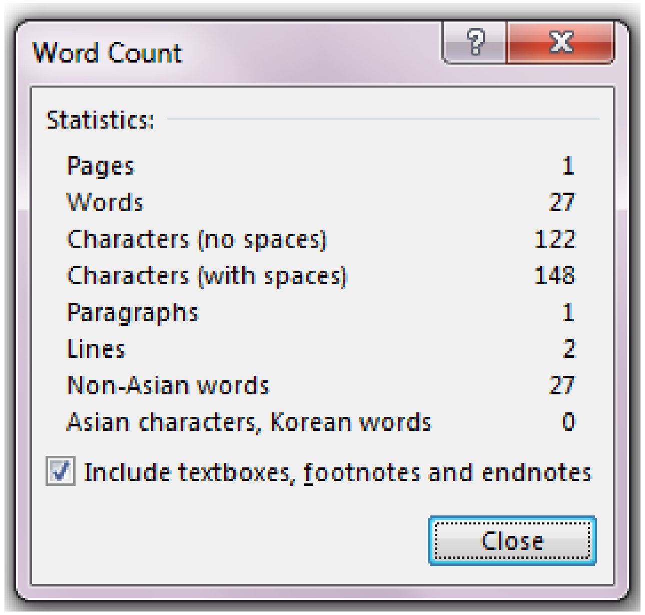
$$\boldsymbol{compression\,ratio} = \frac{\textbf{Uncompressed data rate}}{\textbf{Compressed data rate}} : \mathbf{1}$$

$$\textbf{space savings} = \mathbf{1} - \boldsymbol{compression\,ratio}$$

Space saving can also be presented in the form of percentage (%).

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

"One swallow does not make a summer, neither does one fine day; similarly one day or brief time of happiness does not make a person entirely happy."


Word Count

Statistics:

Pages	1
Words	27
Characters (no spaces)	122
Characters (with spaces)	148
Paragraphs	1
Lines	2
Non-Asian words	27
Asian characters, Korean words	0

[✓] Include textboxes, footnotes and endnotes

Let us say that each character (including space) takes up 1 bit, therefore the size of the sentence in the quote is 148 bits. Ignoring capital letters, there are words that are repeated which we can review in the table that is shown below:

Index Number	Repeated Word	Number of repetitive times
1	one	3
2	does	3
3	not	2
4	make	2
5	day	2

- By substituting the index number in the place of the word, this is the result:

"1 swallow 2 3 4 a summer, neither 2 1 fine 5; similarly 1 5 or brief time of happiness 2 3 4 a person entirely happy."

- From a size of 146 bits to a 118 bit size, which converts to only a 19% space saving. This example does not significantly compress the data, but you get the idea. More complex compression algorithms seek out patterns that include eliminating punctuation and spaces and therefore compressing files a lot more.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Lossless Compression Algorithms:

- Repetitive Sequence Suppression*
- Run-length Encoding*
- Pattern Substitution
- Entropy Encoding
  - The Shannon-Fano Algorithm*
  - Huffman Coding*

Repetitive Sequence Suppression

- If a sequence a series on n successive tokens appears
- Replace series with a token and a count number of

occurrences.

- Usually need to have a special flag to denote when the

repeated token appears

- Example

89400000000000000000000000000000000

- we can replace with 894f32, where f is the flag for zero.

Run-length encoding (RLE)

- a technique used to **reduce the size** of a **repeating string of characters**.
- This repeating string is called a ***run***, typically RLE encodes a **run of symbols** into **two bytes**, a **count** and a **symbol**.
- RLE can compress any type of data regardless of its information content, but the **content of data** to be compressed **affects the compression ratio**.
- RLE cannot achieve high compression ratios; but it is easy to implement and is quick to execute.
- Run-length encoding is supported by most bitmap file formats such as TIFF, BMP and PCX.

Example 1

Consider a **character run of 15 'A' characters** which normally would require 15 bytes to store :

AAAAAAAAAAAAAAA

With RLE, this would **only** require **two bytes to store**, the count (15) is stored as the first byte and the symbol (A) as the second byte.

15A

Example 2

Consider another example with **16 characters string** of :

000ppppppXXXXaaa

- This string of characters **can be compressed** to form

3(0),6(p),4(X),3(a)

- **16 byte string** would **only require 8 bytes** of data **to represent** the string. In this case, RLE yields a **compression ratio of 2:1**.

Example 3

In run-length encoding, **repetitive source** such as **a string of numbers** can be represented in a compressed form, for example,

1,4,5,1,4,5,1,4,5

can be **compressed** to form

3(1,4,5)

Thus, giving a compression ratio of = 9 : 4 which is almost 2 : 1.

Example 4

Another simple example is when we have a **source** of **incremental patterns** which can be compressed by **differencing**.

1,2,3,5,6,7,9

- taking the **difference between two adjacent values** (eg. 2-1=1, 3-2=1, 5-3=2... etc) we will obtain

1,1,2,1,1,2

- This result **could be further compressed** by representing it as repeated strings, i.e;

2(1,1,2)

Long runs are rare in certain types of data. For instance, in ASCII text files, long runs seldom occur. To **encode a run** in RLE, it is **required** that there is **a minimum of two characters** worth of information, **otherwise** a run of **single character** takes **more space**.

XttmprsQssqznO

we obtain

1(X),2(t),1(m),1(p),1(r),1(s),1(Q),2(s),1(q), 1(z),1(n),1(O)

Exercise

Do a run-length encoding of the following sequence grayscale values. Explain your encoding strategy, and compute the compression rate.

240 240 240 240 240 240 240 238 238 238 238 238
230 230 230 230 229 228 228 227 227 227 227 227
227 227 227 227 227 227 227 227 227 227 227 227
227 227 227 227

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:

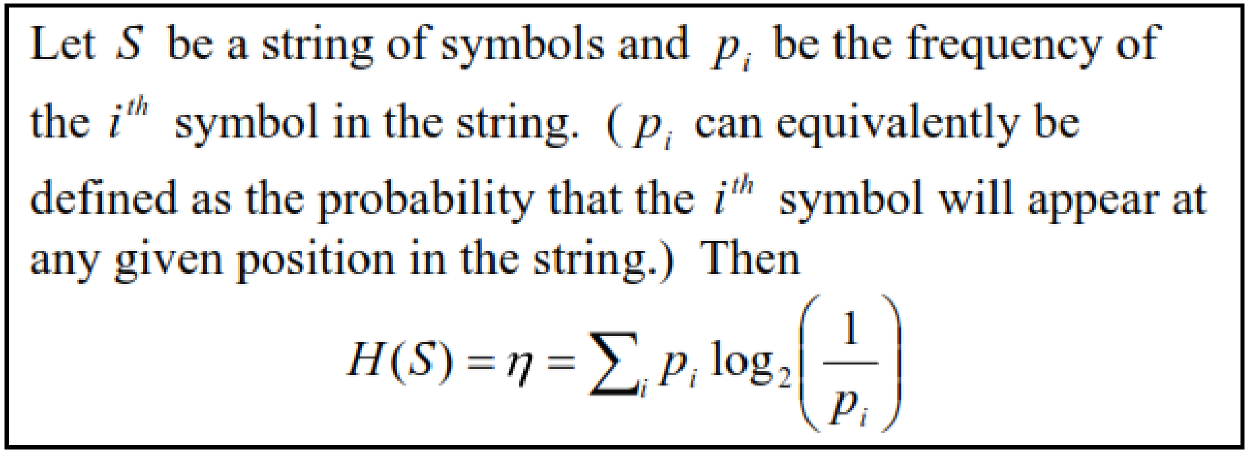
$$H(x) = - p(x) \bullet \log_2 p(x)$$

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

Entropy Encoding

Entropy encoding works by means of variable-length codes, using fewer bits to encode symbols that occur more frequently, while using more bits for symbols that occur infrequently.

Shannon defines the ***entropy of an information source S*** as follows:



Let S be a string of symbols and p_i be the frequency of the i^{th} symbol in the string. (p_i can equivalently be defined as the probability that the i^{th} symbol will appear at any given position in the string.) Then

$$H(S) = \eta = \sum_i p_i \log_2\left(\frac{1}{p_i}\right)$$

To determine an optimum value for the average number of bits needed to represent each symbol-instance in a string of symbols

Think about an image file that has exactly 256 pixels in it, and each pixel is a different color. Then the frequency of each color is 1/256. Thus, Shannon's equation reduces to:

$$\sum_{0}^{255} \frac{1}{256}\left(\log_2\left(\frac{1}{\frac{1}{256}}\right)\right) = \sum_{0}^{255} \frac{1}{256}(\log_2(256)) = \sum_{0}^{255} \frac{1}{256}(8) = 8$$

the average number of bits needed to encode each color is 8

Example 5

color	frequency	optimum number of bits to encode this color	relative frequency of the color in the file	product of columns 3 and 4
black	100	1.356	0.391	0.530
white	100	1.356	0.391	0.530
yellow	20	3.678	0.078	0.287
orange	5	5.678	0.020	0.111
red	5	5.678	0.020	0.111
purple	3	6.415	0.012	0.075
blue	20	3.678	0.078	0.287
green	3	6.415	0.912	0.075

$$\log_2\left(\frac{256}{100}\right)$$

What is the (minimum) average number of bits used to encode each color in this file?

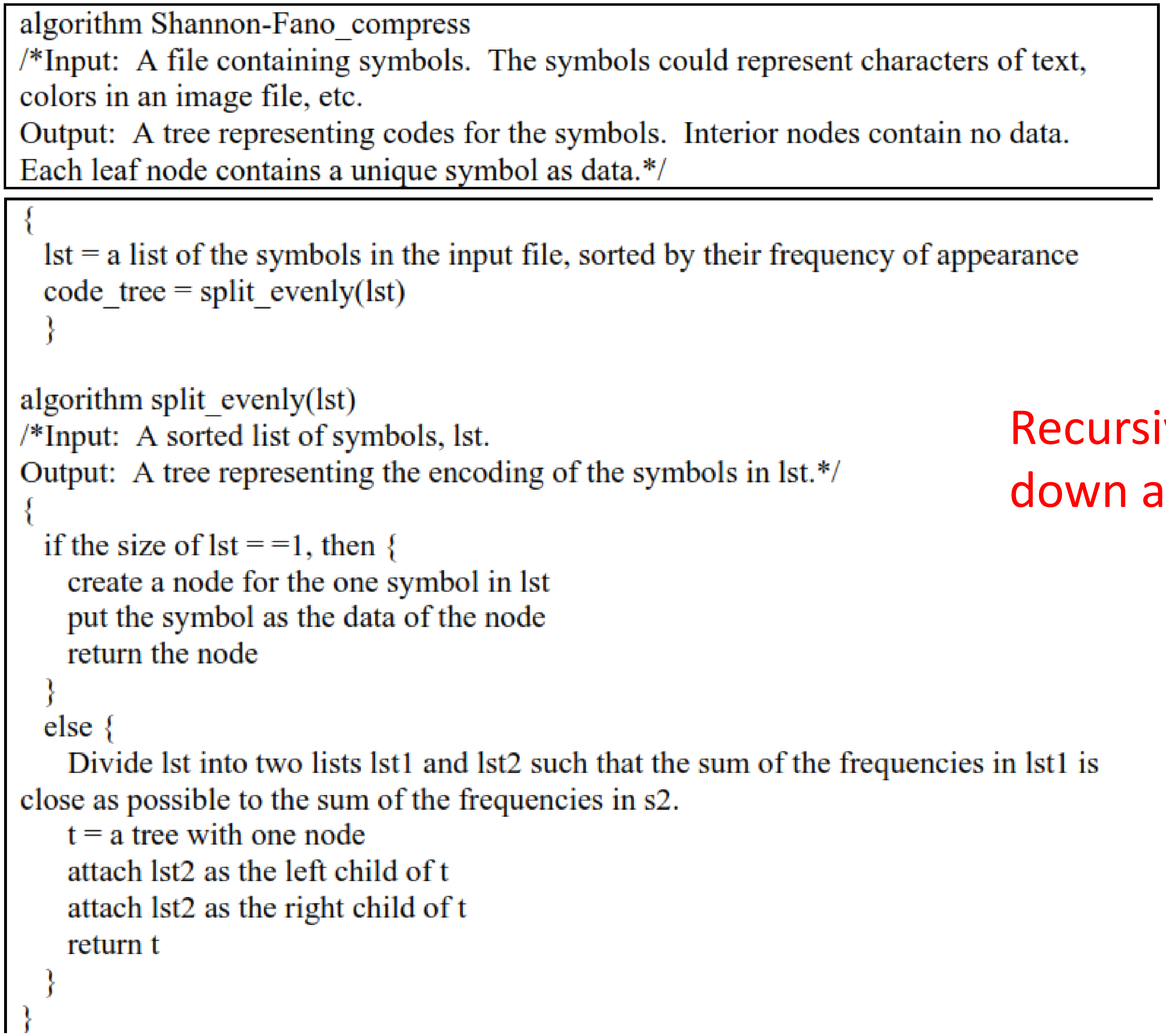
color	frequency	optimum number of bits to encode this color	relative frequency of the color in the file	product of columns 3 and 4
black	100	1.356	0.391	0.530
white	100	1.356	0.391	0.530
yellow	20	3.678	0.078	0.287
orange	5	5.678	0.020	0.111
red	5	5.678	0.020	0.111
purple	3	6.415	0.012	0.075
blue	20	3.678	0.078	0.287
green	3	6.415	0.912	0.075

$$\frac{100}{256}\log_2\left(\frac{256}{100}\right)+\frac{100}{256}\log_2\left(\frac{256}{100}\right)+\frac{20}{256}\log_2\left(\frac{256}{20}\right)+\frac{5}{256}\log_2\left(\frac{256}{5}\right)+\frac{5}{256}\log_2\left(\frac{256}{5}\right)+$$

$$\frac{3}{256}\log_2\left(\frac{256}{3}\right)+\frac{20}{256}\log_2\left(\frac{256}{20}\right)+\frac{3}{256}\log_2\left(\frac{256}{3}\right)\approx$$

$$0.530+0.530+0.287+0.111+0.111+0.075+0.287+0.075\approx 2.006$$

Shannon-Fano algorithm

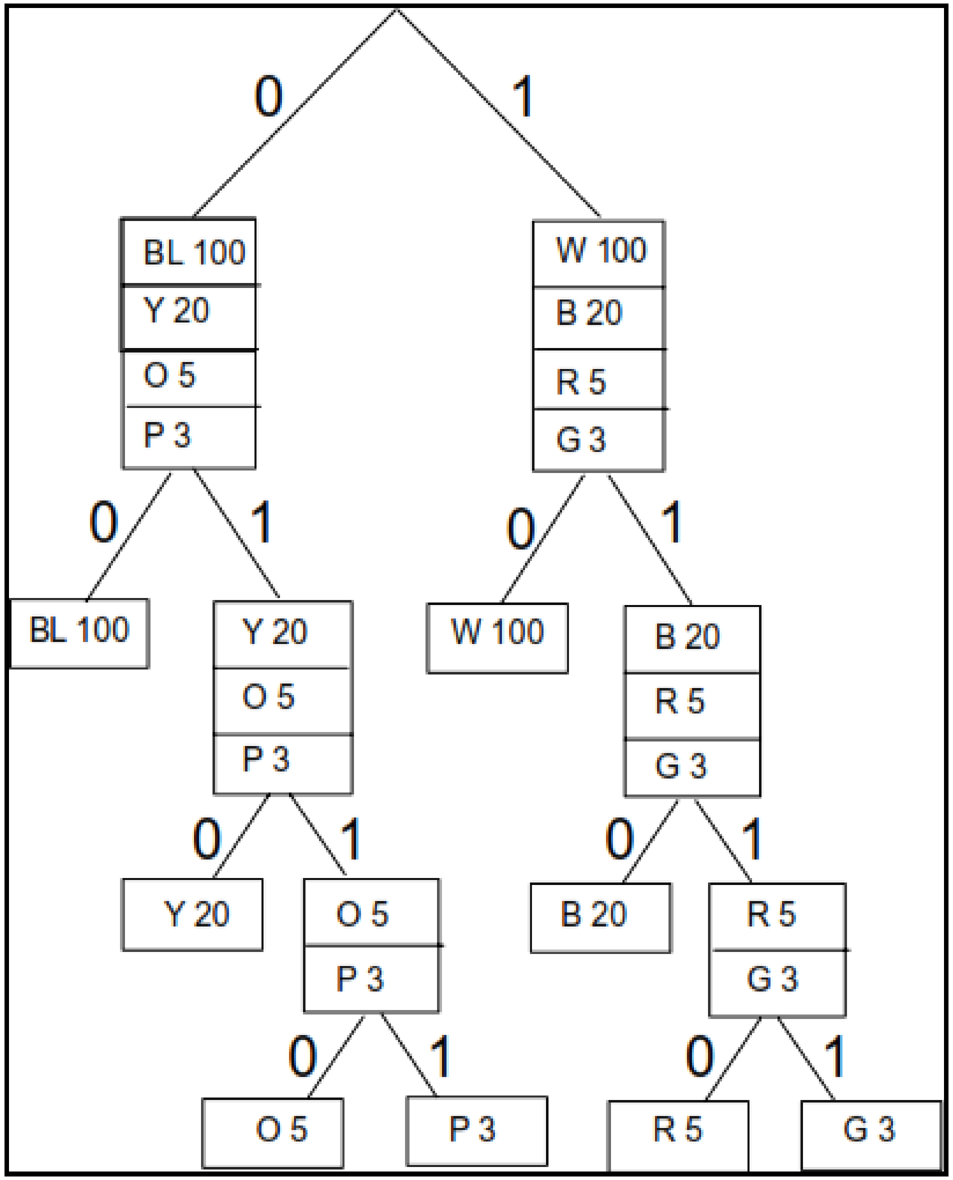


```
algorithm Shannon-Fano_compress
/*Input:  A file containing symbols.  The symbols could represent characters of text,
colors in an image file, etc.
Output:  A tree representing codes for the symbols.  Interior nodes contain no data.
Each leaf node contains a unique symbol as data.*/
{
  lst = a list of the symbols in the input file, sorted by their frequency of appearance
  code_tree = split_evenly(lst)
  }

algorithm split_evenly(lst)
/*Input:  A sorted list of symbols, lst.
Output:  A tree representing the encoding of the symbols in lst.*/
{
  if the size of lst = =1, then {
     create a node for the one symbol in lst
     put the symbol as the data of the node
     return the node
  }
  else {
     Divide lst into two lists lst1 and lst2 such that the sum of the frequencies in lst1 is
close as possible to the sum of the frequencies in s2.
     t = a tree with one node
     attach lst2 as the left child of t
     attach lst2 as the right child of t
     return t
  }
}
```

Recursive top-down approach

Example 6: Compression using the Shannon-Fano algorithm



color	frequency	code
black	100	00
white	100	10
yellow	20	010
orange	5	0110
red	5	1110
purple	3	0111
blue	20	110
green	3	1111

(100*2)+(100*2)+(20*3)+(5*4)+(5*4)+(3*4)+(20*3)+(3*4) = 584 bits

The average =

$$\frac{584}{256} = 2.28 \text{ bits per symbol-instance}$$

Compression rate: $\frac{8}{2.28}$

about 3.5:1

Huffman coding

- Huffman compression **reduces** the **average code length** used **to represent** the **symbols of an alphabet**.
- JPEGs do use Huffman as part of their compression process.
- Symbols of the source alphabet which **occur frequently** are assigned with **short length codes**.
- Huffman compression is performed by constructing a **binary tree** using a simple example set. (bottom-up)
- This is done by **arranging the symbols** of the alphabets in **descending** order of probability.

- Then repeatedly adding **two lowest probabilities** and resorting. This process goes on until the sum of probabilities of the **last two symbols is 1**.
- Once this process is complete, a Huffman binary tree can be generated.
- If we do not obtain a probability of 1 in the last two symbols, most likely there is a mistake in the process.
- This probability of 1 which forms the last symbol is the **root** of the binary tree.

How to Generate Huffman Tree

(1)**Initialization:** put all symbols on the list sorted according to their frequency counts.

(2)**Repeat** until the list has only one symbol left.

(a) From the list, pick two symbols with the lowest frequency counts. Form a Huffman subtree that has these two symbols as child nodes and create a parent node for them.

(b) Assign the sum of children's frequency counts to the parent and insert it into list, such that the order is maintained.

(c) Delete the children from the list.

(3)**Assign a codeword** for each leaf based on the path from the root.

Example 7

Given a set of **symbols** with a list of relative **probabilities** of occurrence within a message.

m0	m1	m2	m3	m4
0.10	0.36	0.15	0.2	0.19

(1) List symbols in the order of **decreasing probability**.

m1	m3	m4	m2	m0
0.36	0.20	0.19	0.15	0.10

(2) Get **two** symbols with **lowest probability**. Give the combined symbol a **new name**.

m2	m0
0.15	0.10

Combines to form

A
0.25

UNIVERSITI TEKNIKAL

(3) The **new list** obtained is shown below. Repeating the previous step will give us a new symbol for the **next two lowest probabilities**.

m1	A	m3	m4
0.36	0.25	0.20	0.19

m3	m4
0.20	0.19

Combines to form

B
0.39

(4) A **new list** is obtained. Repeating the previous step will give us a **new symbol** for the following **two lowest probabilities**.

B	m1	A
0.39	0.36	0.25

m1	A
0.36	0.25

Combines to form

C
0.61

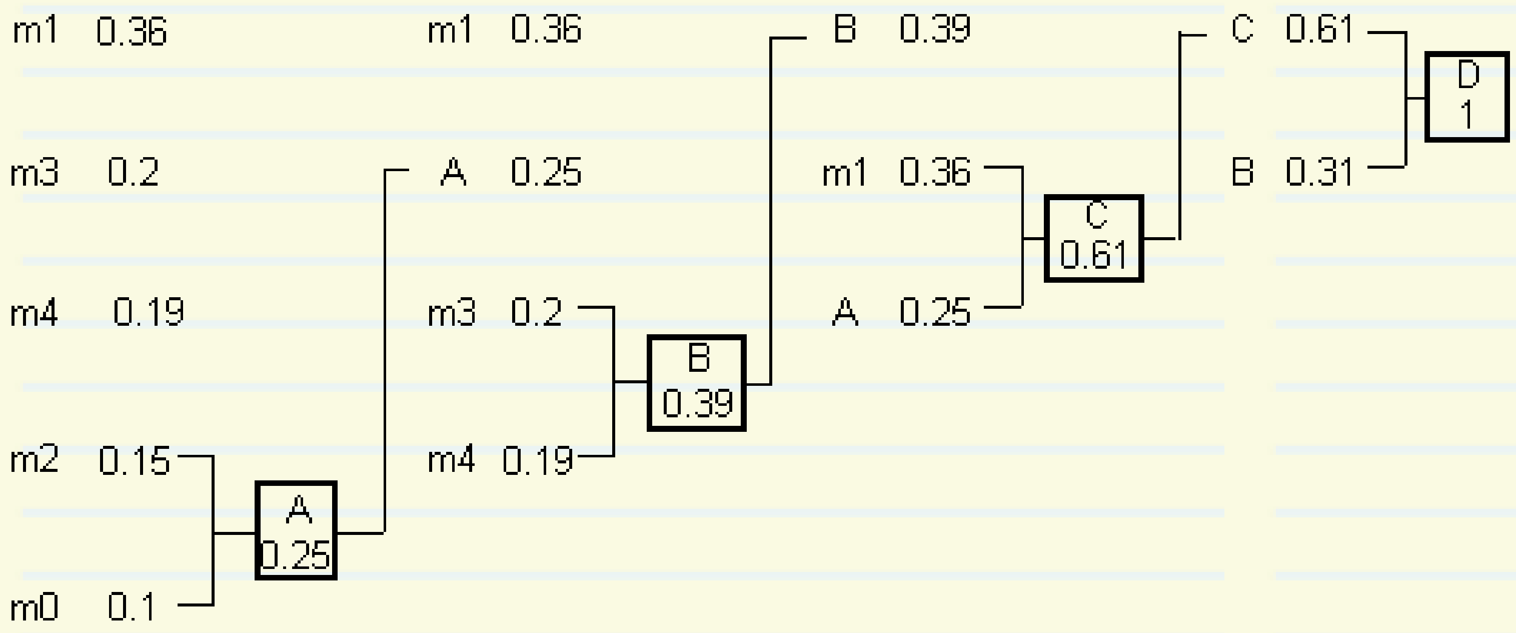
(5) Finally there is only one pair left and we simply combine them and name them as a **new symbol.**

B	C
0.39	0.61

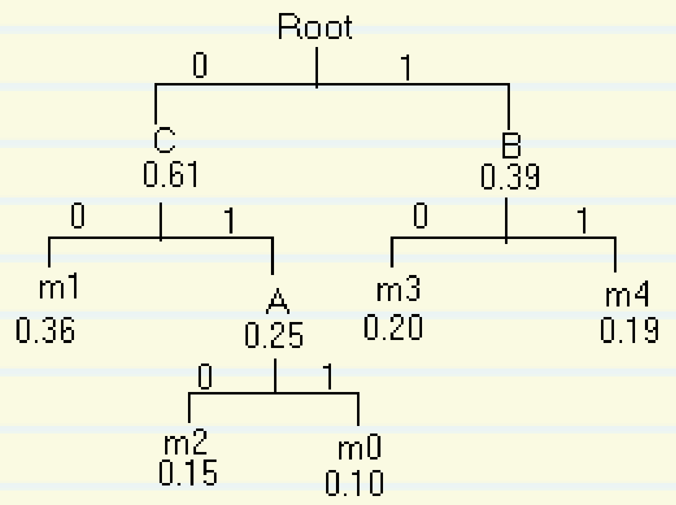
Combines to form

D
1.0

(6) Having finished these steps we have :



(7) Now, a **Huffman tree** can be constructed, **0**'s and **1**'s are assigned to the branches.



(8)The **resultant codewords** are formed by tracing the tree path from the root node to the codeword leaf.

Symbols	Probabilities	Codewords
m0	0.10	011
m1	0.36	00
m2	0.15	010
m3	0.20	10
m4	0.19	11

Example 8

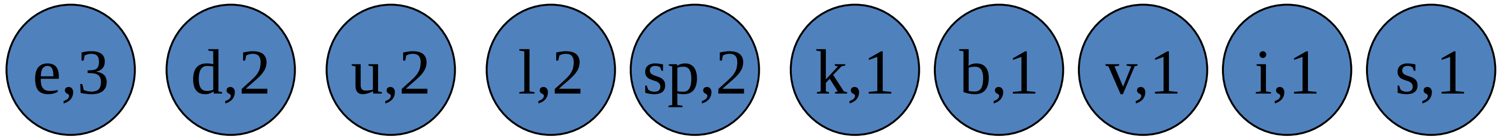
Lets take the string:

"duke blue devils"

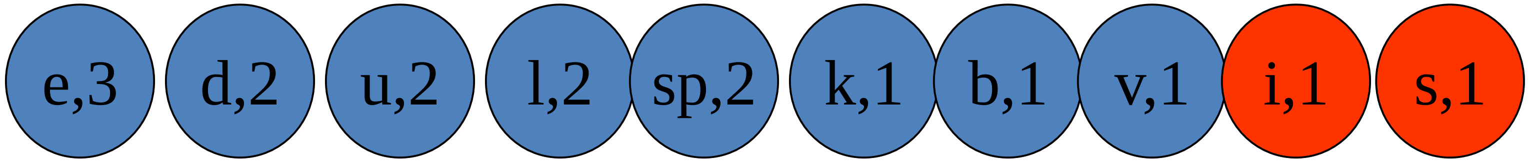
- We first do a frequency count of the characters:

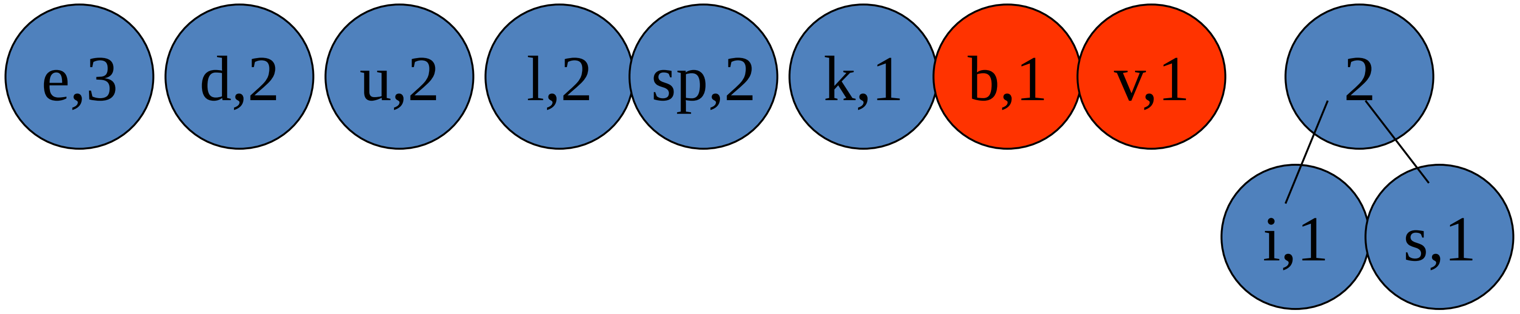
  e:3, d:2, u:2, l:2, space:2, k:1, b:1, v:1, i:1, s:1

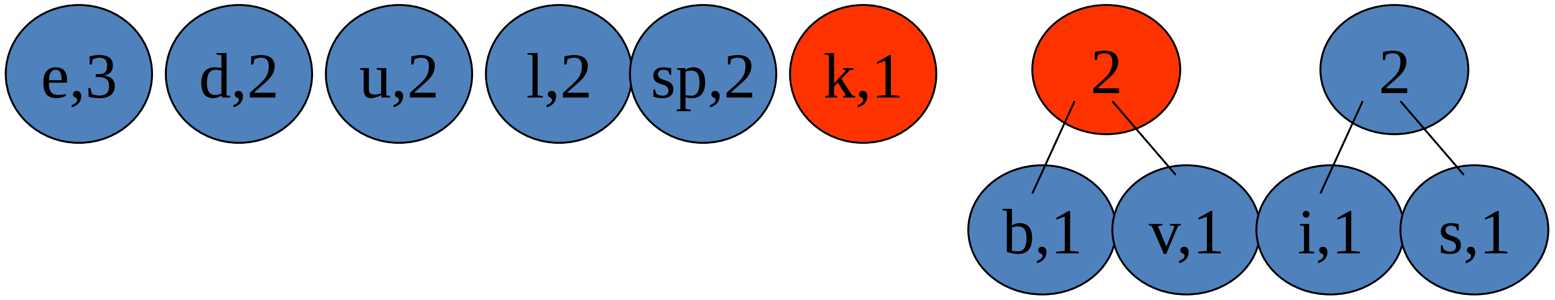
- Next we build up a Huffman Tree
  - We start with nodes for each character

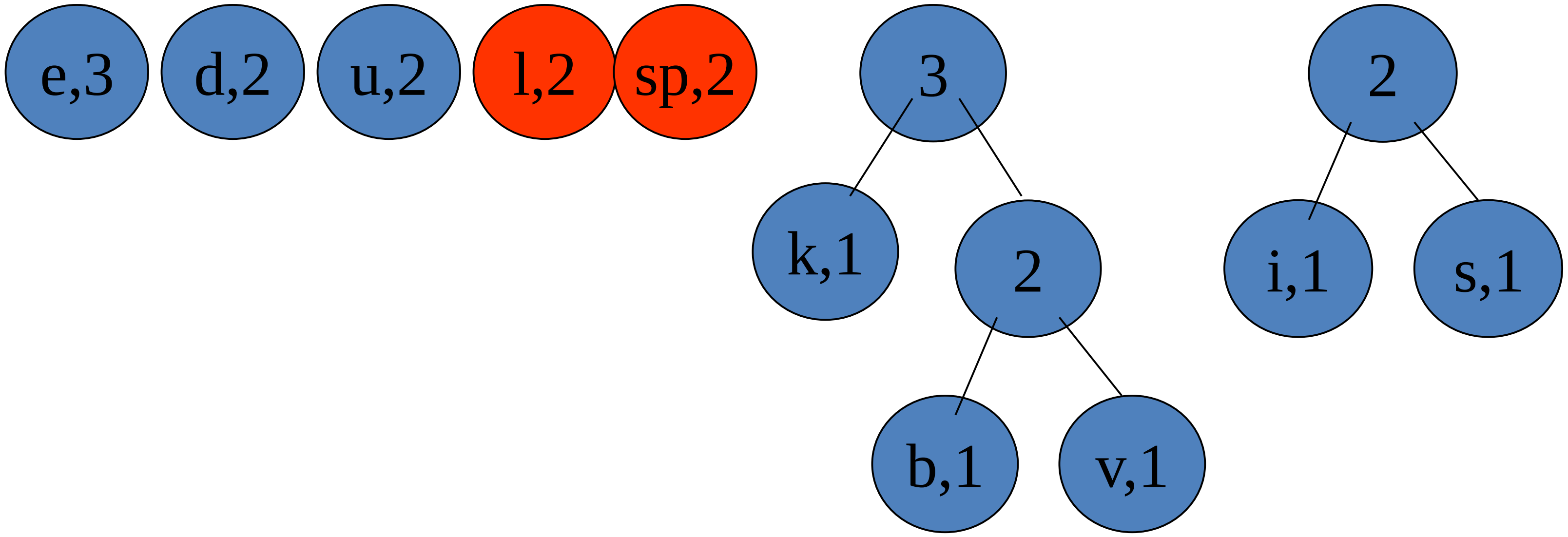


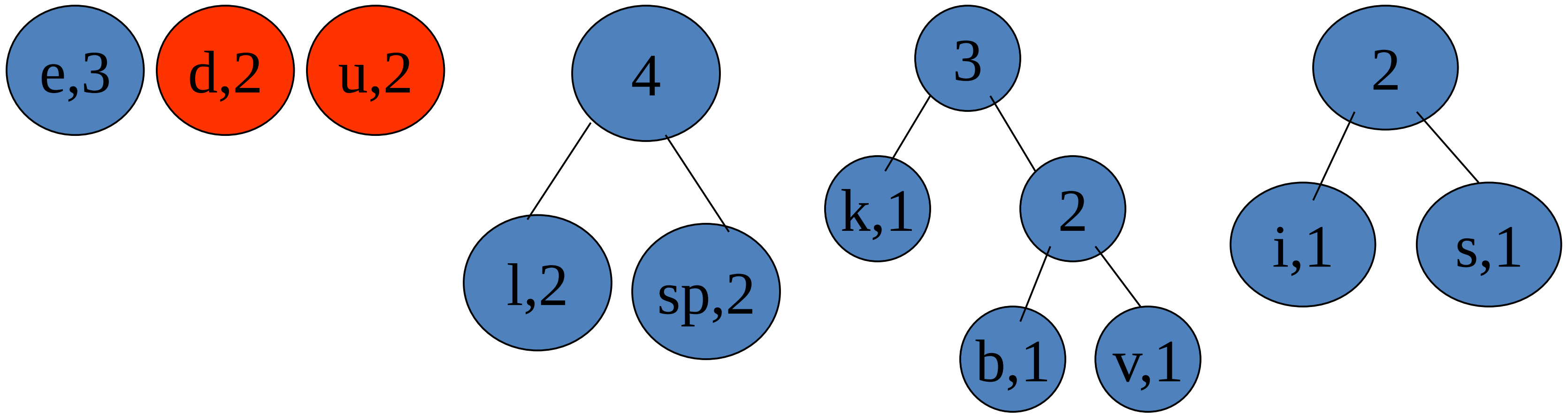
- We then pick the nodes with the smallest frequency and combine them together to form a new node
- The two selected nodes are removed from the set, but replace by the combined node
- This continues until we have only 1 node left in the set

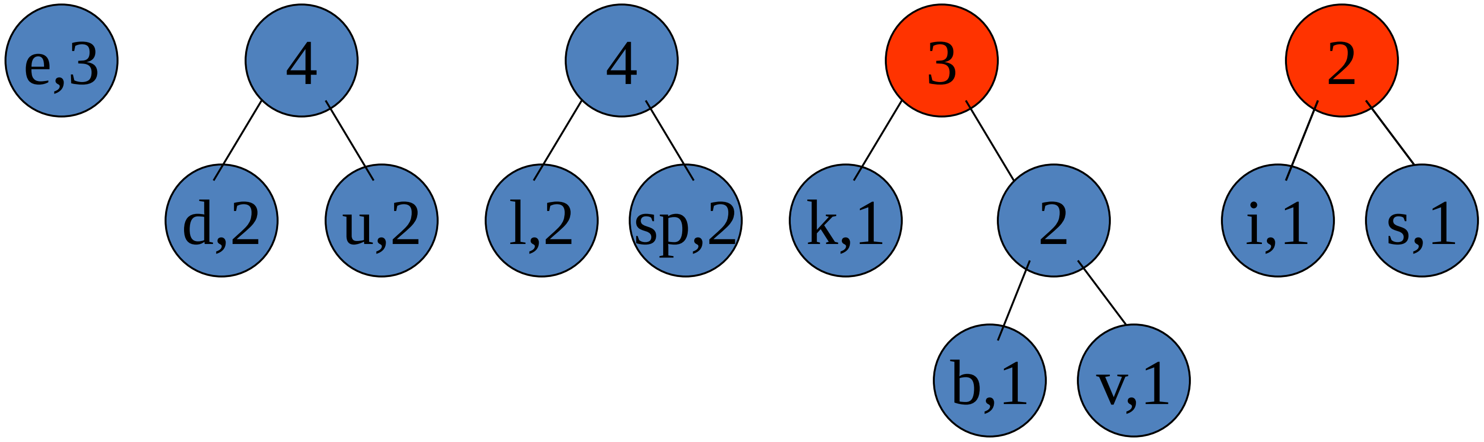

e,3
d,2
u,2
l,2
sp,2
k,1
b,1
v,1
i,1
s,1

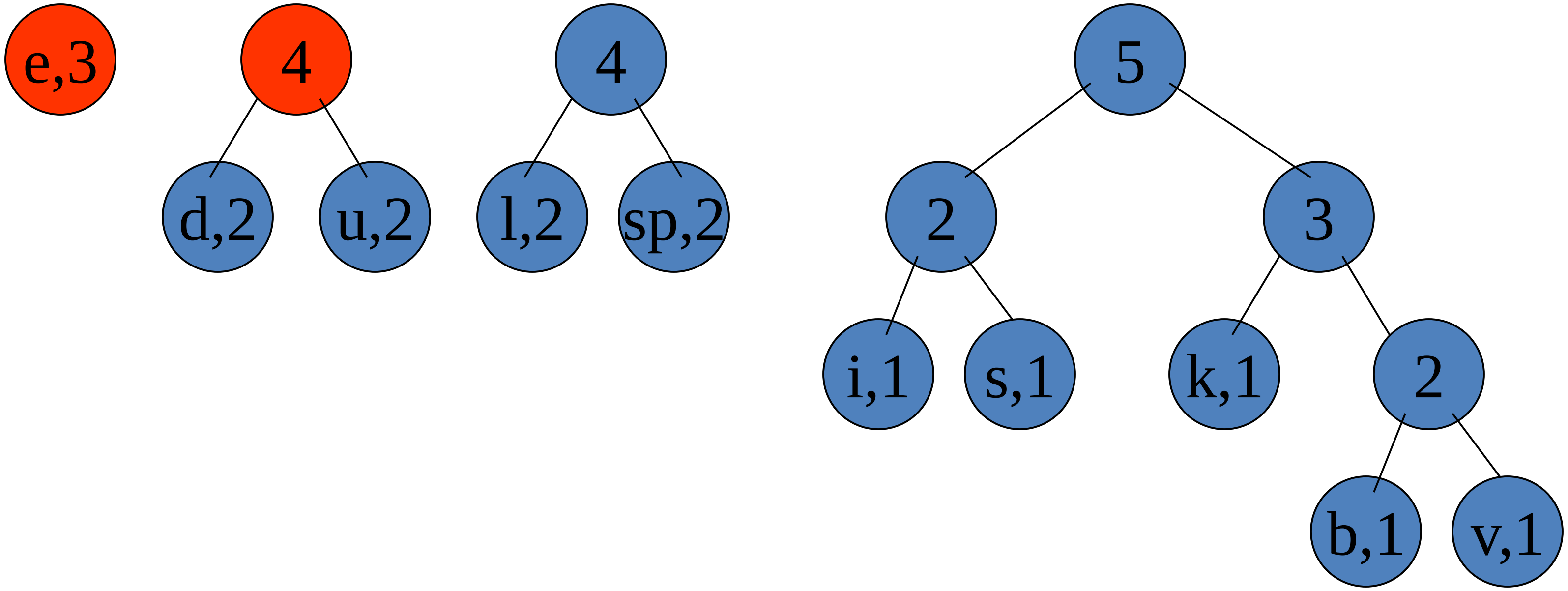

e,3
d,2
u,2
l,2
sp,2
k,1
b,1
v,1
2
i,1
s,1

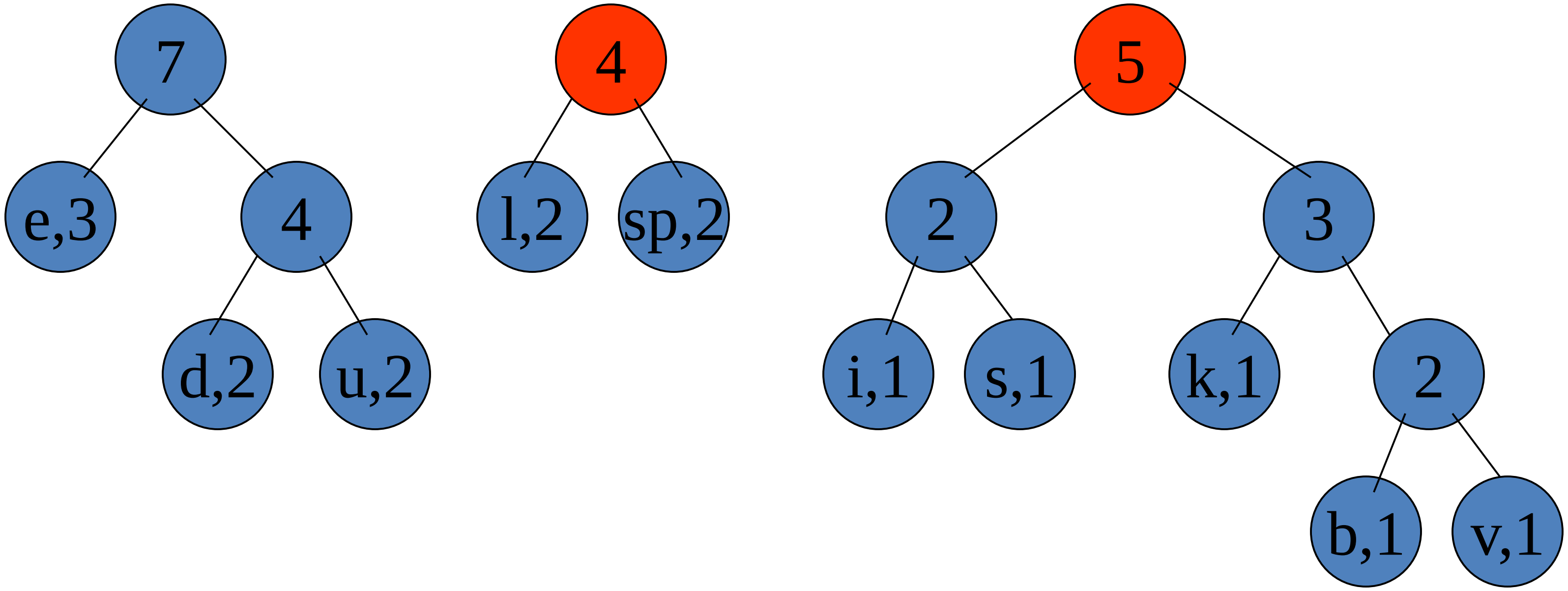

e,3
d,2
u,2
l,2
sp,2
k,1
2
2
b,1
v,1
i,1
s,1

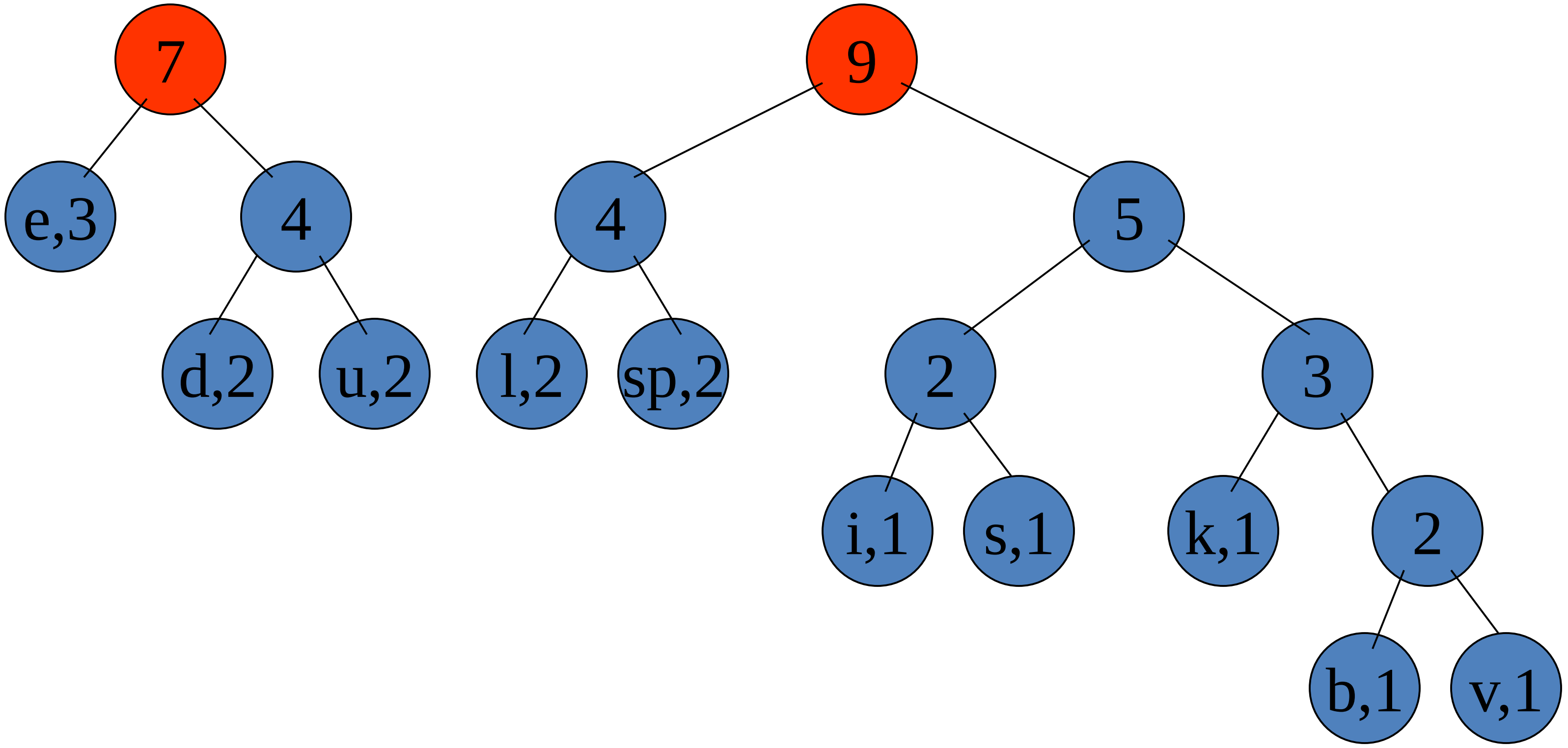

e,3
d,2
u,2
l,2
sp,2
3
k,1
2
b,1
v,1
2
i,1
s,1

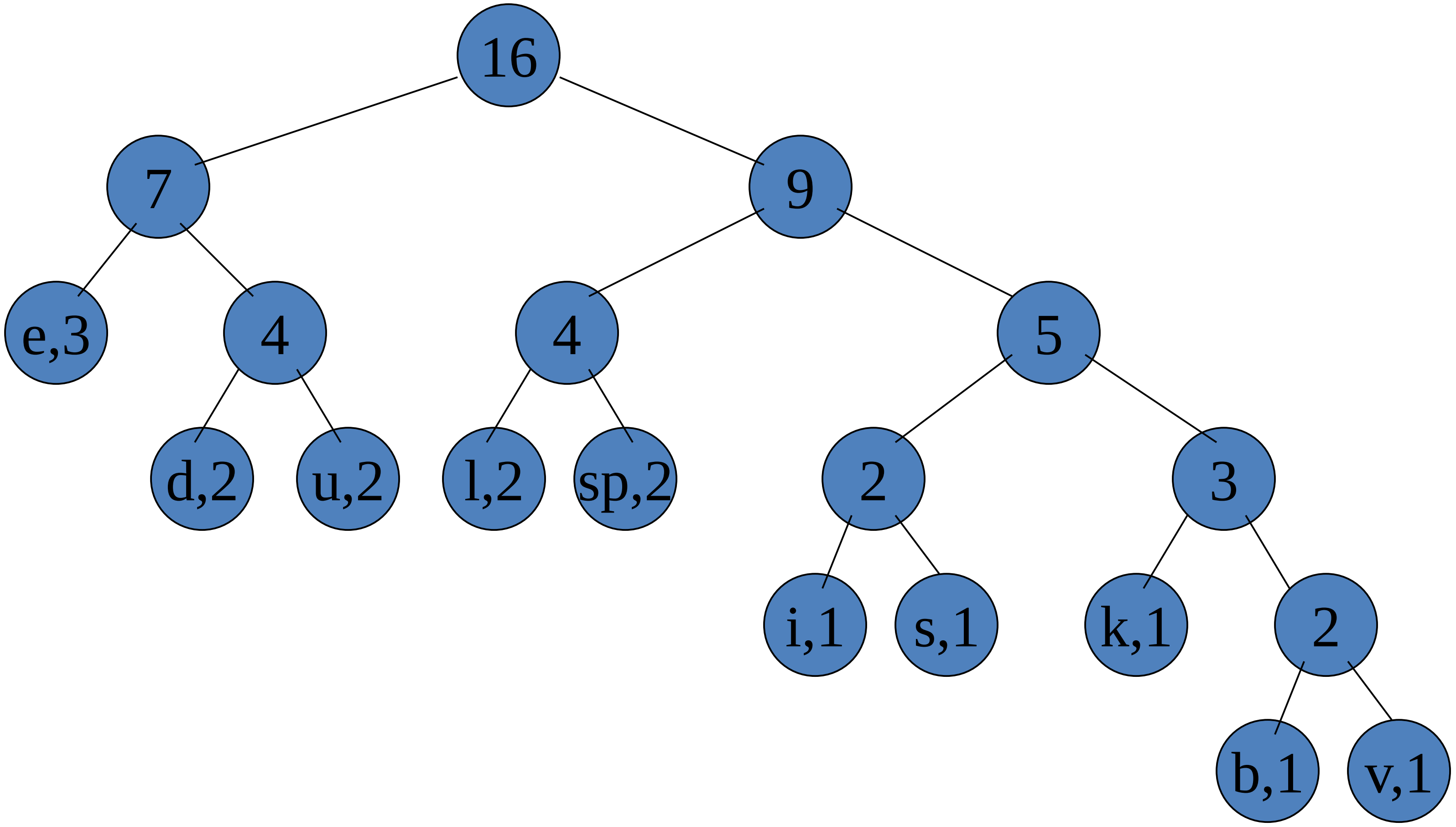

e,3
d,2
u,2
4
l,2
sp,2
3
k,1
2
b,1
v,1
2
i,1
s,1


e,3
4
d,2
u,2
4
l,2
sp,2
3
k,1
2
b,1
v,1
2
i,1
s,1


e,3
4
d,2
u,2
4
l,2
sp,2
5
2
3
i,1
s,1
k,1
2
b,1
v,1

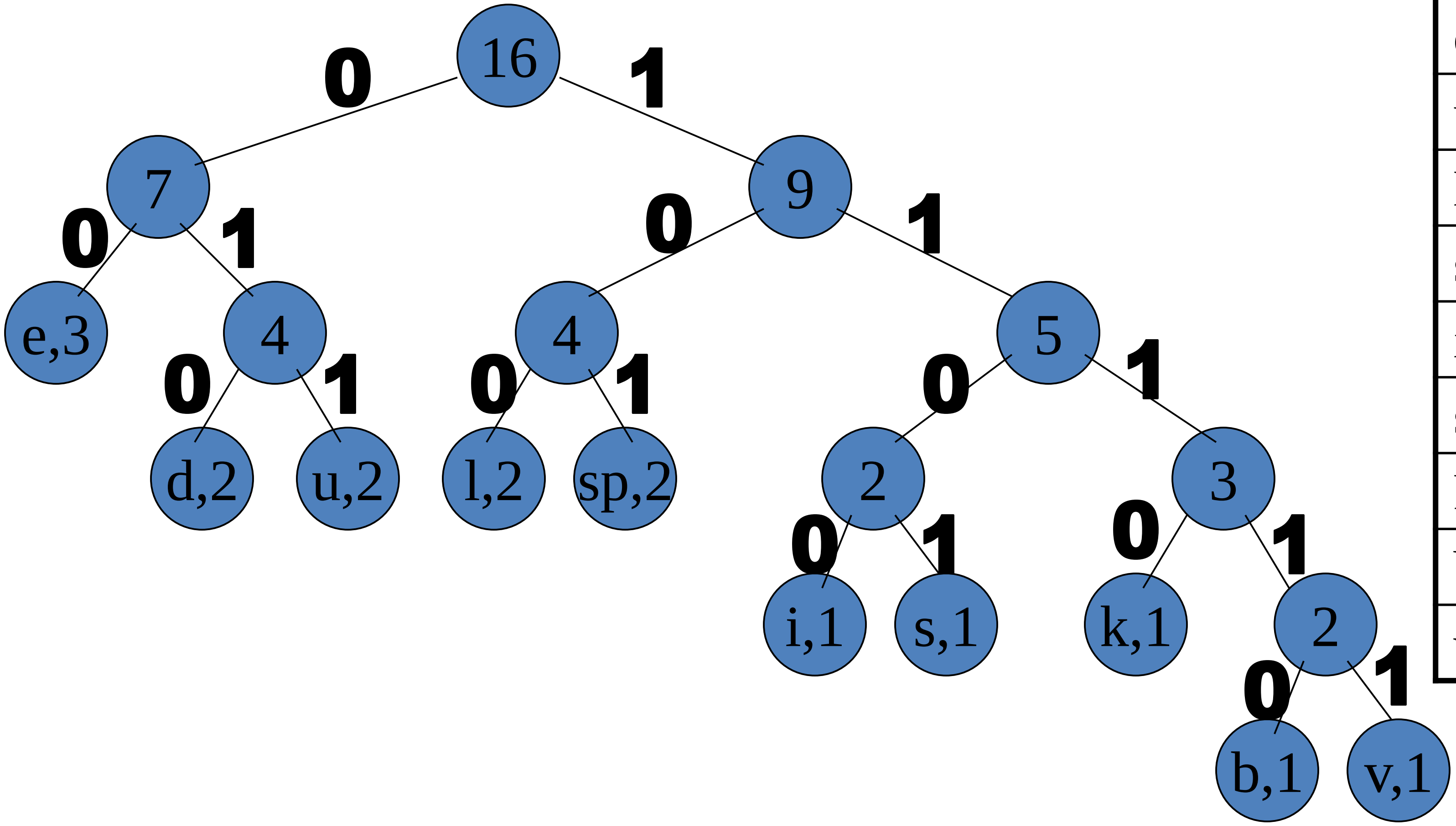

7
e,3
4
d,2
u,2
4
l,2
sp,2
5
2
3
i,1
s,1
k,1
2
b,1
v,1


7
e,3
4
d,2
u,2
9
4
l,2
sp,2
5
2
i,1
s,1
3
k,1
2
b,1
v,1


16
7
9
e,3
4
4
5
d,2
u,2
l,2
sp,2
2
3
i,1
s,1
k,1
2
b,1
v,1

- Now we assign codes to the tree by placing a 0 on every left branch and a 1 on every right branch
- A traversal of the tree from root to leaf give the Huffman code for that particular leaf character
- Note that no code is the prefix of another code

Huffman Coding



e	00
d	010
u	011
l	100
sp	101
i	1100
s	1101
k	1110
b	11110
v	11111

- These codes are then used to encode the string
- Thus, “duke blue devils” turns into:

010 011 1110 00 101 11110 100 011 00 101 010 00 11111 1100 100 1101

- When grouped into 8-bit bytes:

01001111 10001011 11101000 11001010 10001111 11100100 1101xxxx

- Thus it takes 7 bytes of space compared to

16 characters * 1 byte/char = 16 bytes uncompressed

- Uncompressing works by reading in the file bit by bit
  - Start at the root of the tree
  - If a 0 is read, head left
  - If a 1 is read, head right
  - When a leaf is reached decode that character and start over again at the root of the tree
- Thus, we need to save Huffman table information as a header in the compressed file
  - Doesn’t add a significant amount of size to the file for large files (which are the ones you want to compress anyway)
  - Or we could use a fixed universal set of codes/frequencies

Exercise

Given the symbols with the characters in the word HELLO, perform:

1. Shannon-Fano Algorithm
2. Huffman coding.

Lossy Compression

- Lossless compression algorithms do not deliver *compression ratios* that are high enough. Hence, most multimedia compression algorithms are *lossy*.

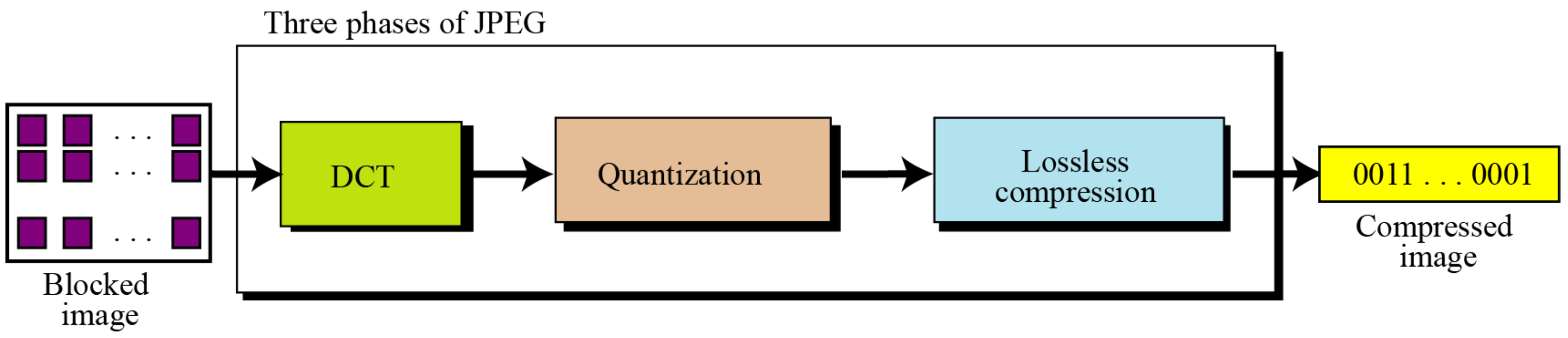
- What is *lossy compression*?
  – The compressed data is not the same as the original data, but a close approximation of it.
  – Yields a much higher compression ratio than that of lossless compression.

Lossy Compression

- Used for compressing images and video files (our eyes cannot distinguish subtle changes, so lossy data is acceptable).
- These methods are cheaper, less time and space.
- Several methods:
    - JPEG: compress pictures and graphics
    - MPEG: compress video
    - MP3: compress audio

JPEG Encoding

- Used to compress pictures and graphics.
- In JPEG, a grayscale picture is divided into 8x8 pixel blocks to decrease the number of calculations.
- Basic idea:
  - Change the picture into a linear (vector) sets of numbers that reveals the redundancies.
  - The redundancies is then removed by one of lossless compression methods.

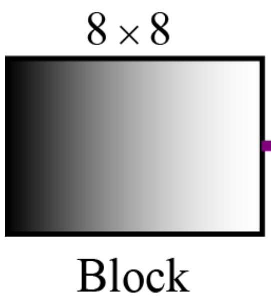


UNIVERSITI TEKNIKAL MALAYSIA MELAKA

JPEG Encoding- DCT

- DCT: Discrete Concise Transform
- DCT transforms the 64 values in 8x8 pixel block in a way that the relative relationships between pixels are kept but the redundancies are revealed.
- Example:

A gradient grayscale



20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90

P (x, y)

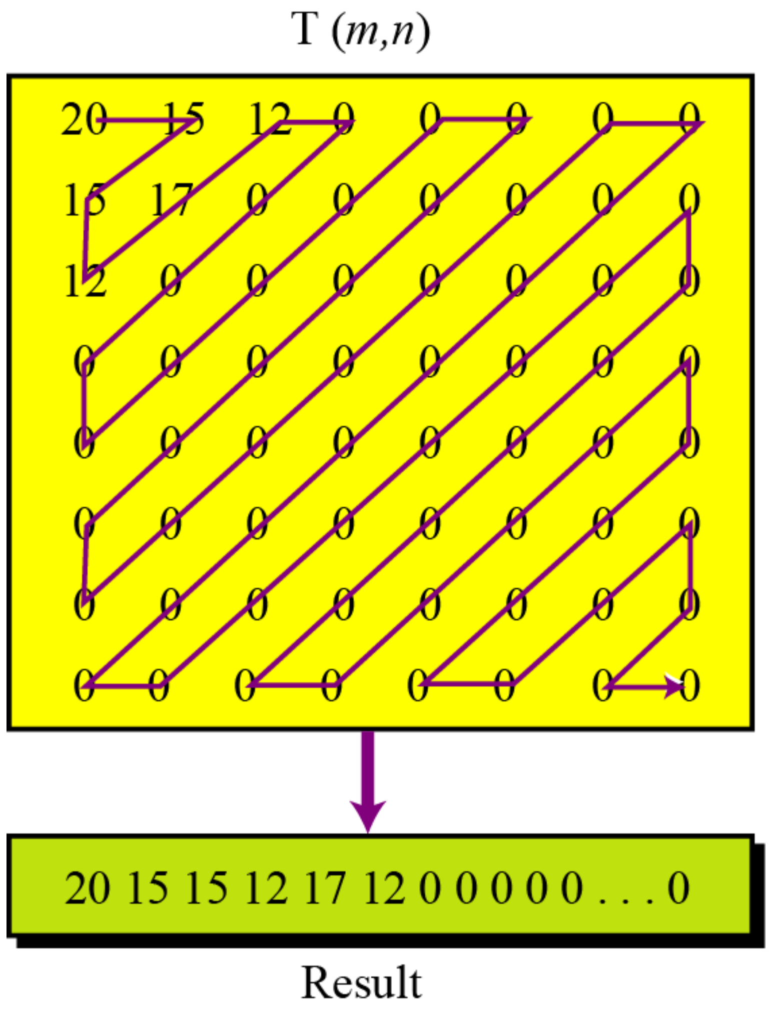
400	−146	0	−31	−1	3	−1	−8
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

T (m, n)

Quantization & Compression

- **Quantization:**
  - After T table is created, the values are quantized to reduce the number of bits needed for encoding.
  - Quantization divides the number of bits by a constant, then drops the fraction. This is done to optimize the number of bits and the number of 0s for each particular application.
  - Main source of the “loss” in lossy compression
- **Compression:**
  - Quantized values are read from the table and redundant 0s are removed.
  - To cluster the 0s together, the table is read diagonally in an zigzag fashion. The reason is if the table doesn’t have fine changes, the bottom right corner of the table is all 0s.
  - JPEG usually uses lossless run-length encoding at the compression phase.

JPEG Encoding



MPEG Encoding

- Used to compress video.
- Basic idea:
  - Each video is a rapid sequence of a set of frames. Each frame is a spatial combination of pixels, or a picture.
  - Compressing video =

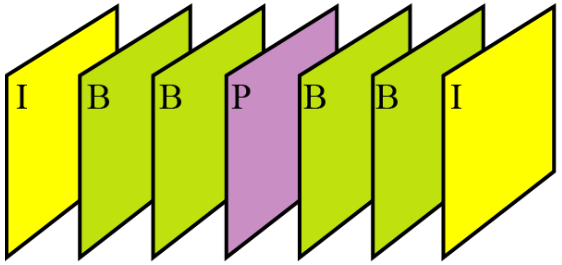
    spatially compressing each frame

    +

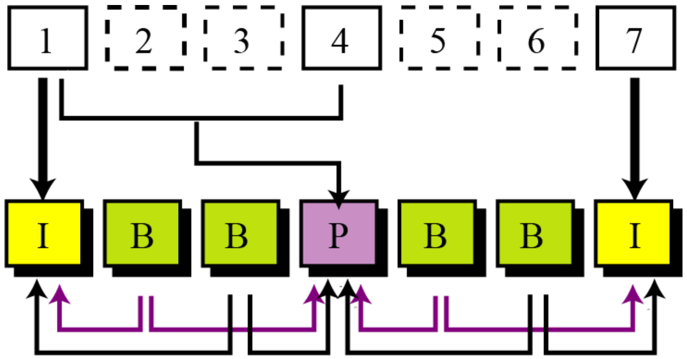
    temporally compressing a set of frames.

MPEG Encoding

- Spatial Compression
  - Each frame is spatially compressed by JPEG.
- Temporal Compression
  - Redundant frames are removed.
  - For example, in a static scene in which someone is talking, most frames are the same except for the segment around the speaker's lips, which changes from one frame to the next



a. Frames



b. Frame construction

Audio Compression

- Used for speech or music
  - Speech: compress a 64 kHz digitized signal
  - Music: compress a 1.411 MHz signal
- Two categories of techniques:
  - Predictive encoding
  - Perceptual encoding

Audio Encoding

- Predictive Encoding
  - Only the differences between samples are encoded, not the whole sample values.
  - Several standards: GSM (13 kbps), G.729 (8 kbps), and G.723.3 (6.4 or 5.3 kbps)

- Perceptual Encoding: **MP3**
  - CD-quality audio needs at least 1.411 Mbps and cannot be sent over the Internet without compression.
  - MP3 (MPEG audio layer 3) uses perceptual encoding technique to compress audio.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Examples of Compression Algorithms & Codecs

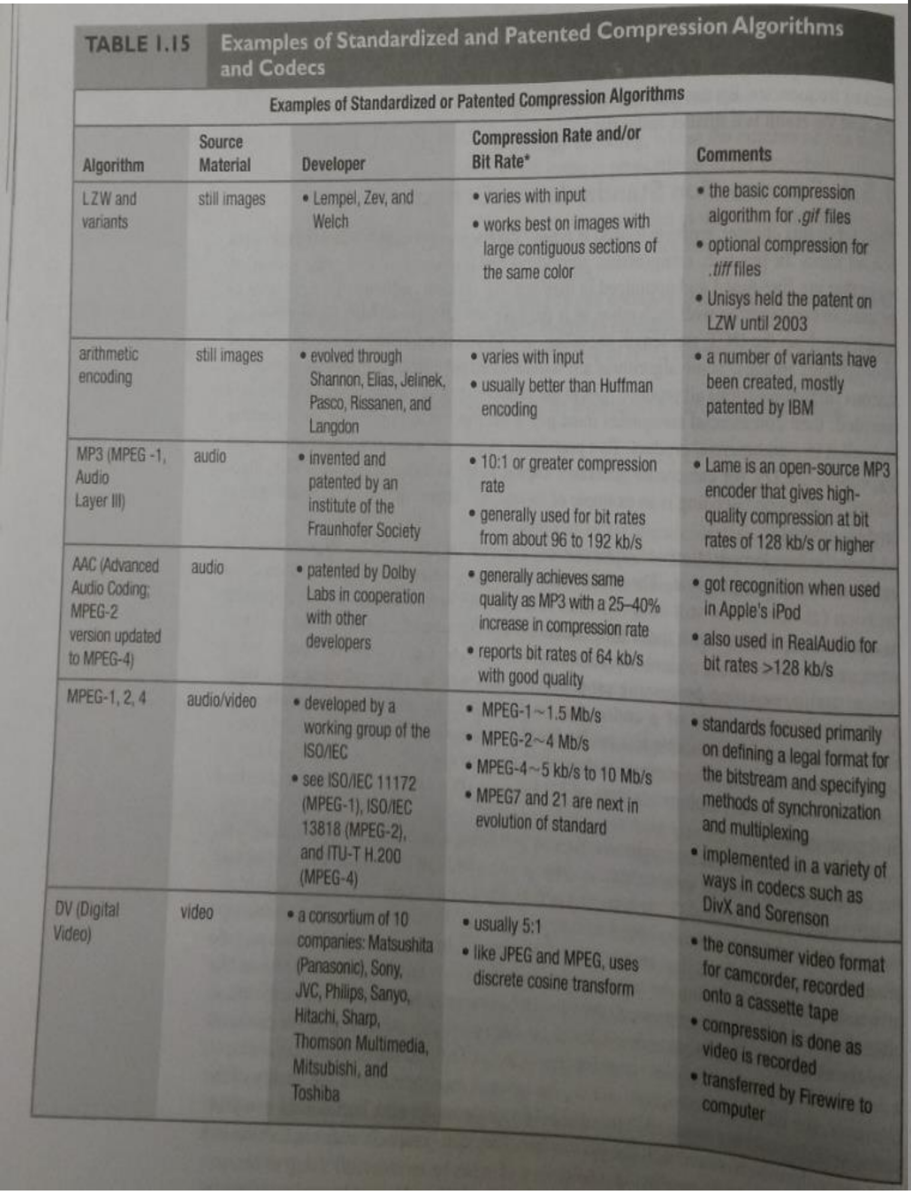


TABLE 1.15 Examples of Standardized and Patented Compression Algorithms and Codecs

Examples of Standardized or Patented Compression Algorithms

Algorithm	Source Material	Developer	Compression Rate and/or Bit Rate*	Comments
LZW and variants	still images	• Lempel, Zev, and Welch	• varies with input • works best on images with large contiguous sections of the same color	• the basic compression algorithm for *.gif* files • optional compression for *.tiff* files • Unisys held the patent on LZW until 2003
arithmetic encoding	still images	• evolved through Shannon, Elias, Jelinek, Pasco, Rissanen, and Langdon	• varies with input • usually better than Huffman encoding	• a number of variants have been created, mostly patented by IBM
MP3 (MPEG -1, Audio Layer III)	audio	• invented and patented by an institute of the Fraunhofer Society	• 10:1 or greater compression rate • generally used for bit rates from about 96 to 192 kb/s	• Lame is an open-source MP3 encoder that gives high-quality compression at bit rates of 128 kb/s or higher
AAC (Advanced Audio Coding; MPEG-2 version updated to MPEG-4)	audio	• patented by Dolby Labs in cooperation with other developers	• generally achieves same quality as MP3 with a 25–40% increase in compression rate • reports bit rates of 64 kb/s with good quality	• got recognition when used in Apple's iPod • also used in RealAudio for bit rates >128 kb/s
MPEG-1, 2, 4	audio/video	• developed by a working group of the ISO/IEC • see ISO/IEC 11172 (MPEG-1), ISO/IEC 13818 (MPEG-2), and ITU-T H.200 (MPEG-4)	• MPEG-1 ~1.5 Mb/s • MPEG-2 ~4 Mb/s • MPEG-4 ~5 kb/s to 10 Mb/s • MPEG7 and 21 are next in evolution of standard	• standards focused primarily on defining a legal format for the bitstream and specifying methods of synchronization and multiplexing • implemented in a variety of ways in codecs such as DivX and Sorenson
DV (Digital Video)	video	• a consortium of 10 companies: Matsushita (Panasonic), Sony, JVC, Philips, Sanyo, Hitachi, Sharp, Thomson Multimedia, Mitsubishi, and Toshiba	• usually 5:1 • like JPEG and MPEG, uses discrete cosine transform	• the consumer video format for camcorder, recorded onto a cassette tape • compression is done as video is recorded • transferred by Firewire to computer

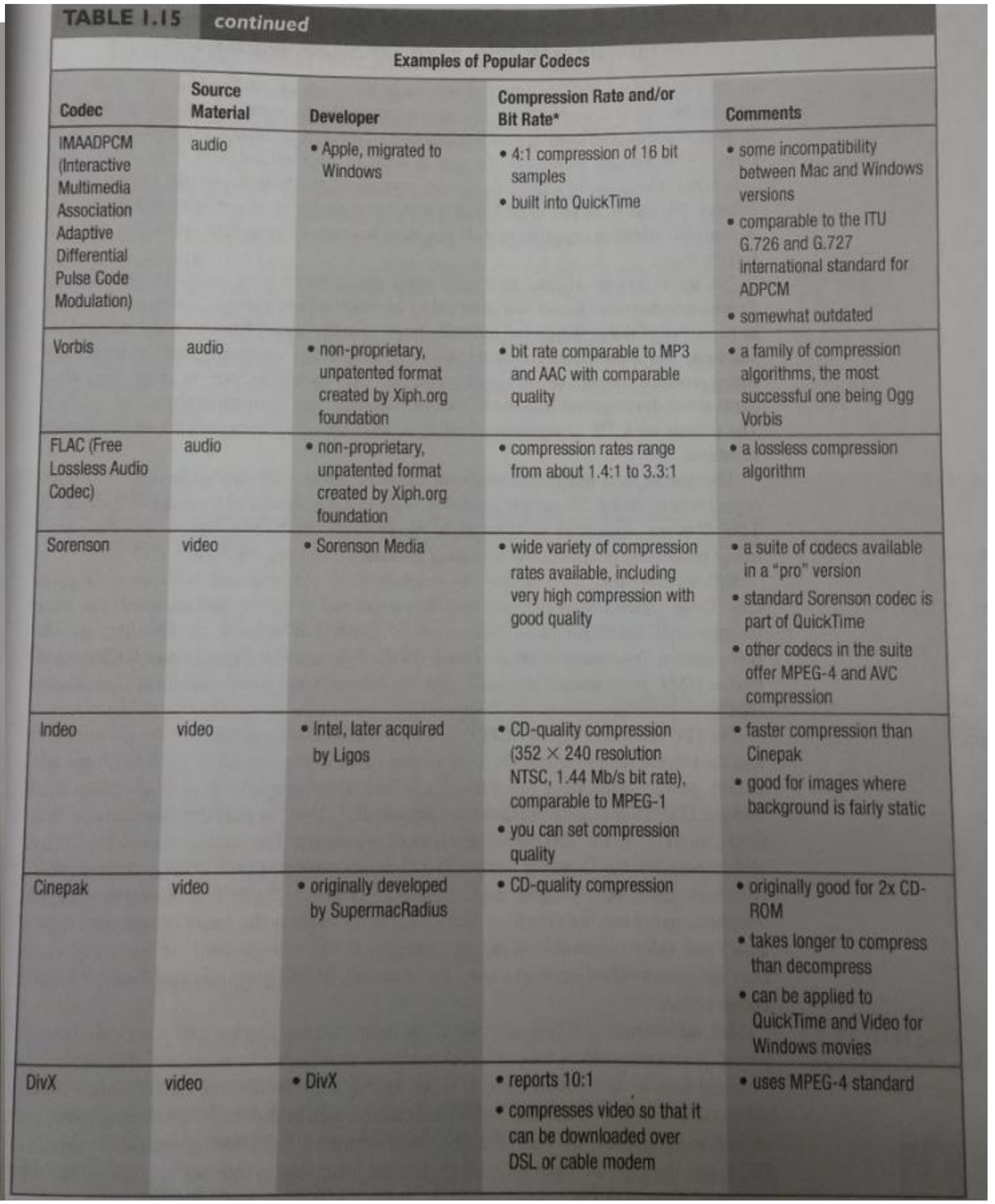


TABLE 1.15 continued

Examples of Popular Codecs

Codec	Source Material	Developer	Compression Rate and/or Bit Rate*	Comments
IMAADPCM (Interactive Multimedia Association Adaptive Differential Pulse Code Modulation)	audio	• Apple, migrated to Windows	• 4:1 compression of 16 bit samples • built into QuickTime	• some incompatibility between Mac and Windows versions • comparable to the ITU G.726 and G.727 international standard for ADPCM • somewhat outdated
Vorbis	audio	• non-proprietary, unpatented format created by Xiph.org foundation	• bit rate comparable to MP3 and AAC with comparable quality	• a family of compression algorithms, the most successful one being Ogg Vorbis
FLAC (Free Lossless Audio Codec)	audio	• non-proprietary, unpatented format created by Xiph.org foundation	• compression rates range from about 1.4:1 to 3.3:1	• a lossless compression algorithm
Sorenson	video	• Sorenson Media	• wide variety of compression rates available, including very high compression with good quality	• a suite of codecs available in a "pro" version • standard Sorenson codec is part of QuickTime • other codecs in the suite offer MPEG-4 and AVC compression
Indeo	video	• Intel, later acquired by Ligos	• CD-quality compression (352 × 240 resolution NTSC, 1.44 Mb/s bit rate), comparable to MPEG-1 • you can set compression quality	• faster compression than Cinepak • good for images where background is fairly static
Cinepak	video	• originally developed by SupermacRadius	• CD-quality compression	• originally good for 2x CD-ROM • takes longer to compress than decompress • can be applied to QuickTime and Video for Windows movies
DivX	video	• DivX	• reports 10:1 • compresses video so that it can be downloaded over DSL or cable modem	• uses MPEG-4 standard