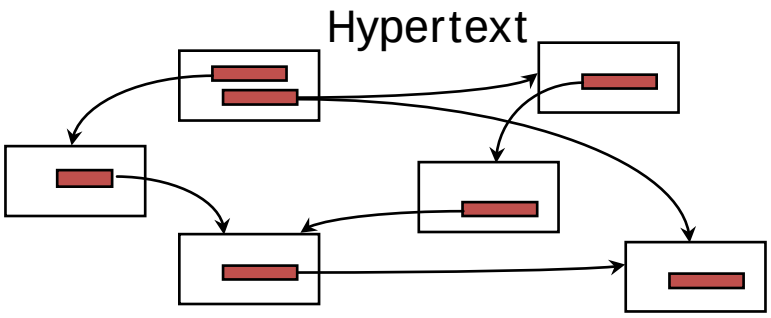
Mobile Programming and Multimedia

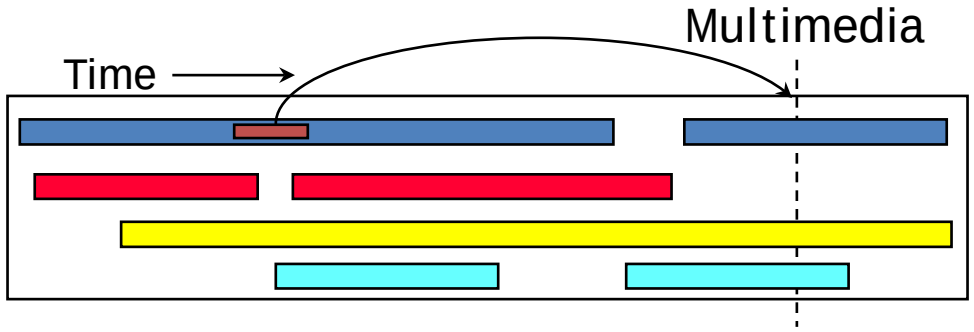
Multimedia Data

Prof. Ombretta Gaggi
University of Padua

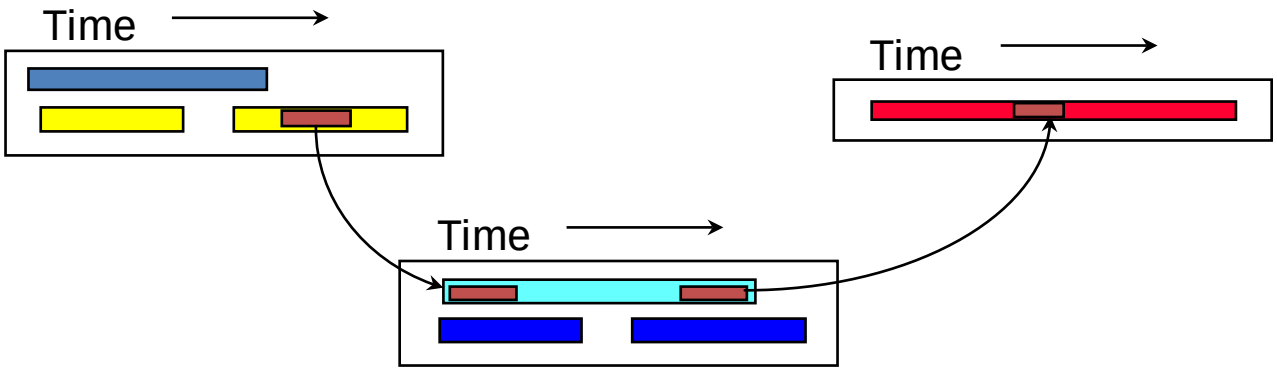
Hypertext + Multimedia = Hypermedia



- ✓ High dimension
- ✓ Isochrony of multimedia
- ✓ Synchronization between different continuous media



Hypermedia



Multimedia systems

A multimedia system is a system for storage, integration, and management of heterogeneous complex information, to represent the real world through multiple modalities, making information usage more efficient

- text, graphics
- Recorded or synthesized audio
- Animations, fixed or motion images

Media are humans' extensions: technologies and products giving our senses new possibilities to receive information

(McLuhan, 1964)

What is inteded with “media”

In the last years, the meaning of “media” has changed following the evolution of technology

- In the visual communication domain, it represents the different means of communication (print, radio, television, cinema, ...)
- Non-computer based multimedia presentations have different sources of media (ex: Artistic performances)
- With the first computer applications, the mean of communication was only one
- The term “medium” specified the domain used to represent the information: text, image, animation, sound
- The increase of different informative channels in the modern multimedia system brings back to the original terminology

Multimedia innovation

The innovations introduced by multimedia are not limited to the technological aspects of the problem

- Complex interaction between users and the informative system
  - How can we highlight a link in a video or audio file ?
- Information humanization
- Temporal dimension
- Inappropriateness of permanent copies

The production, organization, and distribution of information requires the development of new competencies

- methodology: behavior models of information access
- technology: development and support tools, standards
- dramatization: emotional interpretation of information
- psychology: efficacy, privacy, control

Open problems

- Higher expressivity means higher complexity
- The learning curve and technical management can penalize product usage
- Higher ease of use can increase products with low internal quality
- Increasing flexibility and variability of exploration can be energy-consuming and confusing
- The efficacy of the system is strictly connected to who provides the information

Multimedia for …

text, images, audio, video for…

…Web?

…Internet?

…network systems?

…distributed systems?

…all the above situations?

What are the differences?

WWW vs. Internet

The World Wide Web is a distributed application (an application container…) using Internet as the technological infrastructure

- Internet provides communication services, protocols, remote storage systems, etc.
- WWW provides the user interface, the environment for distributed applications and the applications

Multimedia systems cover both worlds

- But every world has its own usage space
- Sometimes boundaries are not clearly defined

WWW e multimedia technologies

Multimedia technologies are widely independent from the Web

- Information representation
- Information compression
- Information transmission

Web applications are not the most critical, and do not require the highest resources

- ex. video on demand
- ex. Multichannel communication
- Web evolution is imposing new goals and so new quality standards

Topics

Media classification

Media properties

Information compression

Images

- Overview, GIF, JPEG, PNG

Audio

- Overview, psychoacoustics, MP3, MIDI

Video

- Overview, MPEG, H.261, H.263, DivX, XviD

Media classification

Based on content

- text, image, audio, video
- Text is a banal case

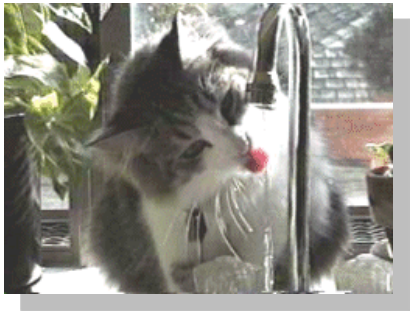
Based on media dynamic

- Diffusion and media usage
- Protocols and network technologies

Media classification: dynamic

Static media

- Information does not change over time



Dynamic media

- Information changes over time

Temporized media

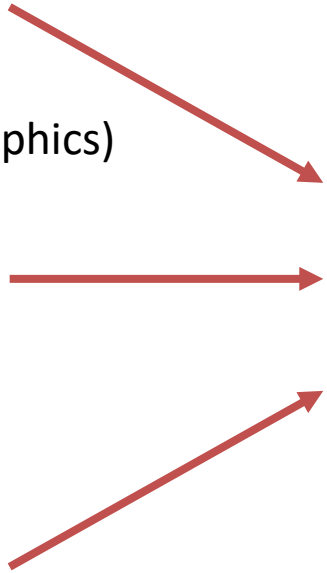
- Information changes over time subject to temporal constraints

Media classification: type

- encoding, compression, quality
- pictorial (pixel) vs. geometric (vector graphics)

- encoding, compression, quality
- voice vs. music vs. generic

- encoding, compression, quality
- animation vs. video clip



size
vs.
time
vs.
quality

Multimedia encoding goals

- ✓ Reduce size of a persistent data representation
- ✓ Optimize transmission time over reproduction time for continuous data
- ✓ Fast decoding for data encoded on conventional and not conventional architectures
- ✓ Quality control of decoded data
- ✓ Provide support to replace missing data

Static media (1)

User reads information with his/her own timing

Data transmission does not impose temporal deadlines (but with reasonable limits…)

Persistent information

Users can read the information several times

Ex. text, images

Static media (2)

Data size is not a problem

- Higher size = higher time
- compression / decompression

Information diffusion and usage takes place in separate moments

- download / display
- Save on local storage
- caching

Dynamic media (1)

User must follow information evolution in real-time

Diffusion can be delayed, but information meaning can change (ex. games)

Information is not persistent (sometimes can be reproduced several times)

Ex: animated presentations, youtube, video files

Dynamic media (2)

Information size is frequently big, and should be used within a reasonable, predefined time

Information diffusion and usage can happen at the same time

- download vs. streaming
- Proprietary solutions
- User interaction control

Temporized media (1)

User must access the information following the timing schedule of the information

Diffusion must follow media isochronous timing

Information is not persistent

The data cannot be used twice

Must be reproduced at a defined fixed frequency to be understood

Ex. audio and real-time video

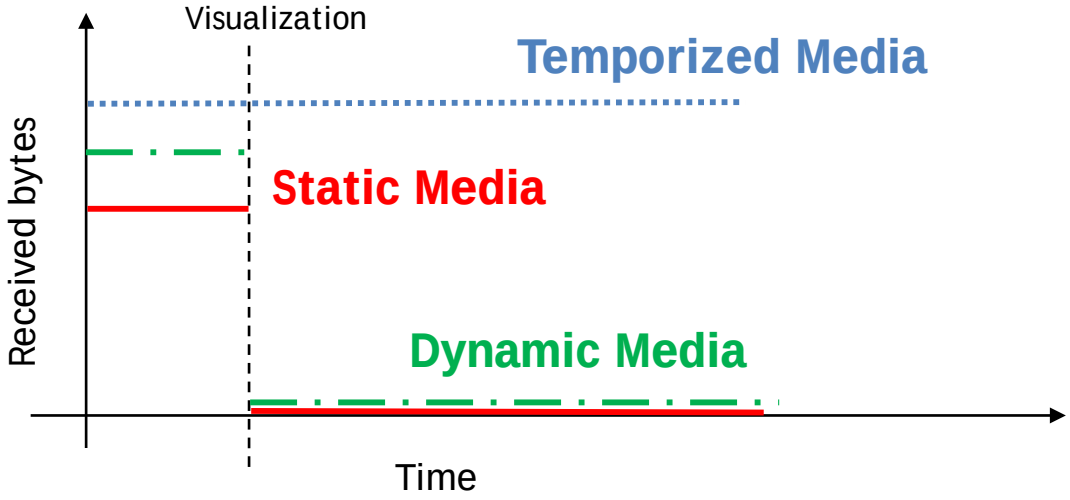
Temporized media (2)

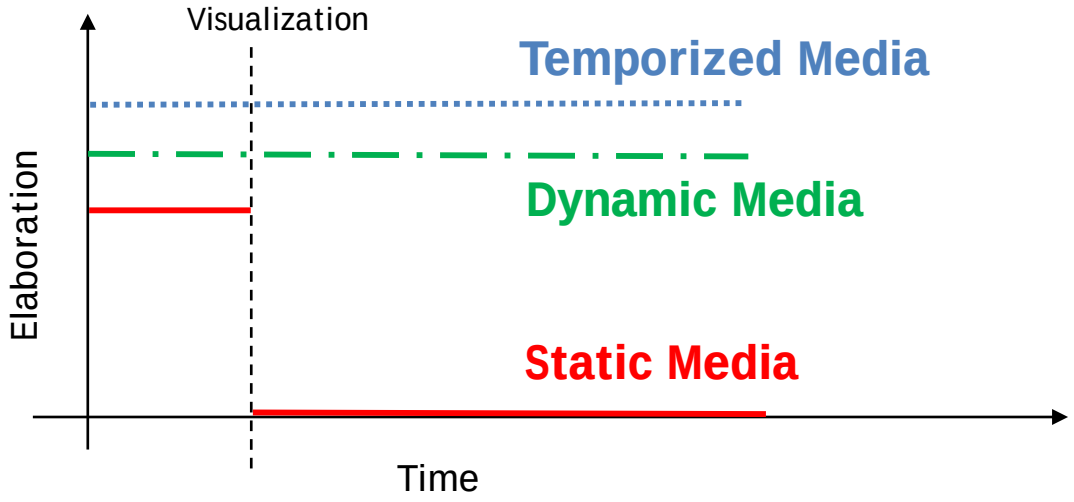
Information size is big and proportional to reproduction time

Data diffusion rhythm must be controlled

- real-time
- streaming
- Bandwidth

Static vs dynamic vs temporized media





Data compression

Multimedia data size is too big to efficiently transmit and store data without using data compression

- CD audio quality (2 channels, 16 bit at 44.1 kHz): 172 Kbyte/s
- VHS video (382*288*16 bit at 25 frames/s): 5.2 Mbyte/s
- NTSC video (640*480*24 bit at 30 frame/s): 26.4 Mbyte/s
- Full HD video (1080*1920*24 bit at 25 frame/s): 148 Mbyte/s

The main goal of data compression algorithms is to reduce storage hence reducing transmission time

- Decompression is necessary
- Data compression can be in real-time or deferred time
- Decompression needs to be in real-time
- Not necessary to save all the information contained in multimedia data

Data compression

Storage reduction can be achieved through:

- The reduction of the number of bits necessary to encode the information (*entropic* compression)
- The reduction of the information to memorize or transmit (differential compression, semantic compression)

Compression can or cannot preserve the original data

- Compression without information loss (*lossless*, reversible): based on coding redundancy of information
- Compression with information loss (*lossy*, irreversible): based on perception redundancy of the data

Compression steps

Compression happens through three phases:

- *transformation*: information data (ex: pixel of an image) are transformed in a domain that requires (can require) less bits to represent data values
- *quantization*: obtained values are (can be) grouped in a smaller number of classes and with a more uniform distribution
- *encoding*: information is encoded based on the new classes and values, together with the encoding table

Lossless compression techniques

Run-length encoding (RLE)

- Encodes sequences of the same value using a lower number of bits

Huffman coding

- Assigns a low number of bits to the more probable values using a fixed coding table

Lempel-Ziv-Welch (LZW) Compression

- Dynamically builds a coding table with a variable number of bits associated to fixed size sequences of values

Differential coding

- Each data is defined as the difference to the previous one (with linear or non-linear resolution)

Lossy compression techniques

UNIVERSITÀ DEGLI STUDI DI PADOVA

JPEG compression (for images)

- Applies to the images a transform to the frequency domain (Discrete Cosine Transform) to suppress irrelevant details, reducing the number of bits necessary to encode the image
- Possible *lossless* compression with predictive techniques

MPEG Compression (for videos)

- Encodes some of the frames as the difference to the expected values calculated with an interpolation

MP3 Compression (for audio)

- Based on psychoacoustics characteristics of human hearing, to suppress useless information

Run Length Encoding

When data contains a lot of consecutive repetitive values, there is a lot of redundancy that can be canceled

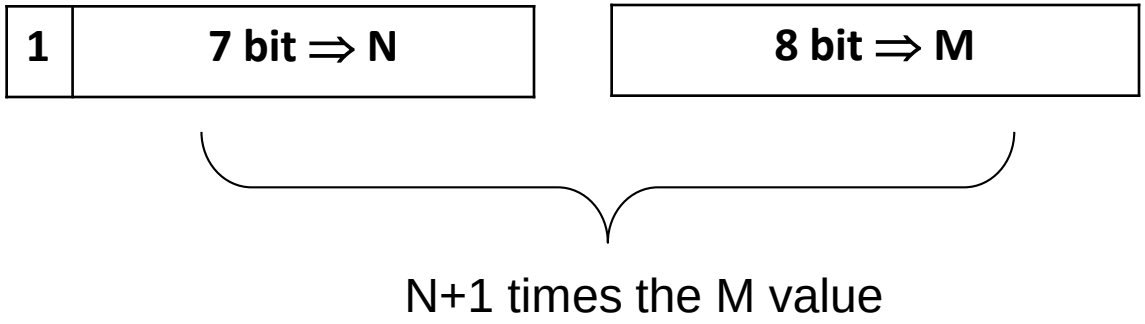
- Long sequences of the same value can be substituted with a repetition symbol, the value, and the number of repetition
- Compression is efficient for sequences of at least three equal values
- Suitable for images with little details (ex: a uniform background)
- Implemented in the bit-map BMP format

Example

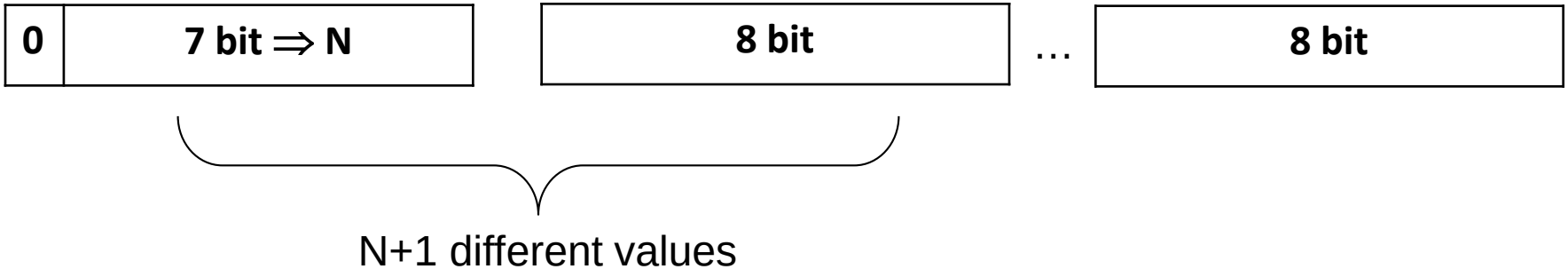
- BGFDDDDDDDDDIJUPPPPPHYTGBUYYYYINNNNNNGHHHHHHHKPPPPPPP
- BGF@9DIJU@5PHYTGBU@4YI@5NG@7HK@7P

RLE Packets

RUN LENGHT



RAW



RLE – Pros & Cons

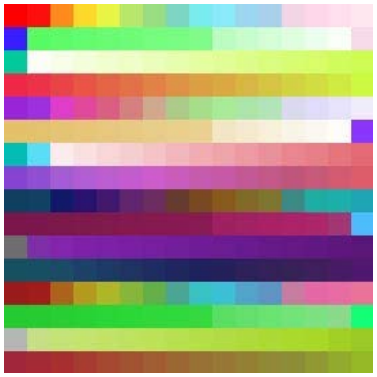
Suitable for:

- Artificial images or with few details

Not suitable for:

- text
- Images with several shades of color or high level of contrast

Enlarged image of a 16 x 16 pixel file, with 256 unique colors.
This file, saved in BMP not compressed, is 822 byte.
Saved instead in BMP format, using LRE algorithm, is 1400 byte, 1,7 times more the original size.



Entropy

The entropy of a source of information is the measure of the *quantity of information* in the bits flow that has to be compressed = number of bits (theoretical) necessary to encode

$$H(S) = \eta = \sum_i p_i \log_2 (1/p_i)$$

- p_i is the probability of the i^{th} element
- $\log_2 1/p_i$ is the minimum number of bits necessary to encode the i^{th} value so that it can be recognized between other elements

Entropy coding

Based on the probability of finding a specific value inside a sequence, it encodes a value with a different number of bits based on its frequency

Example:

- An image with uniform distribution of greyscale $p_i = 1/256$, hence 8 bits are necessary and sufficient to encode each gray tone

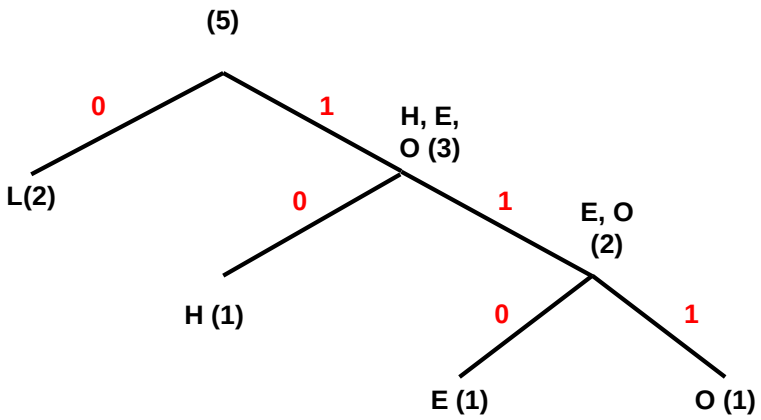
From Shannon's Theorem, for lossless compression, the lower bound of the average code length (hence the compression rate) is the entropy of the information source:

$$H(S) \leq L$$

Shannon-Fano Algorithm

It creates a binary tree with a top-down method. Each node is a symbol or a group of symbol:

- Sorts nodes based on the number of occurrences
- Recursively divides the set of nodes into two parts, each containing, approximatively, the same number of occurrences, until each part has only one symbol



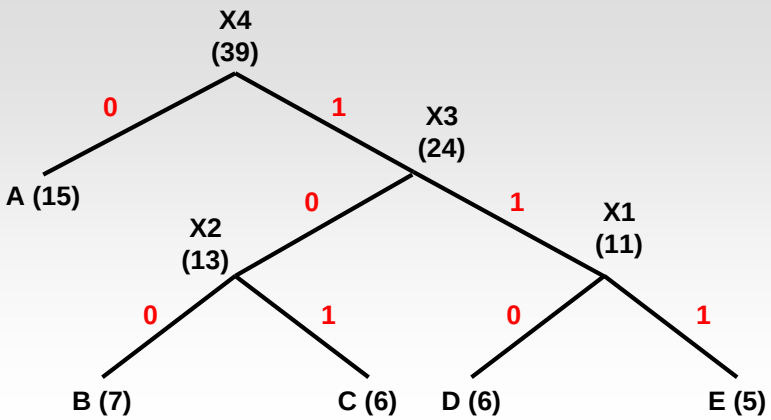
H	E	L	O
1	1	2	1
10	110	0	111

N. of Bit = 10 (theorical 9.6)

Huffmann Code

It creates a binary tree with a bottom-up method. Each node of the tree is a symbol or a group of symbols

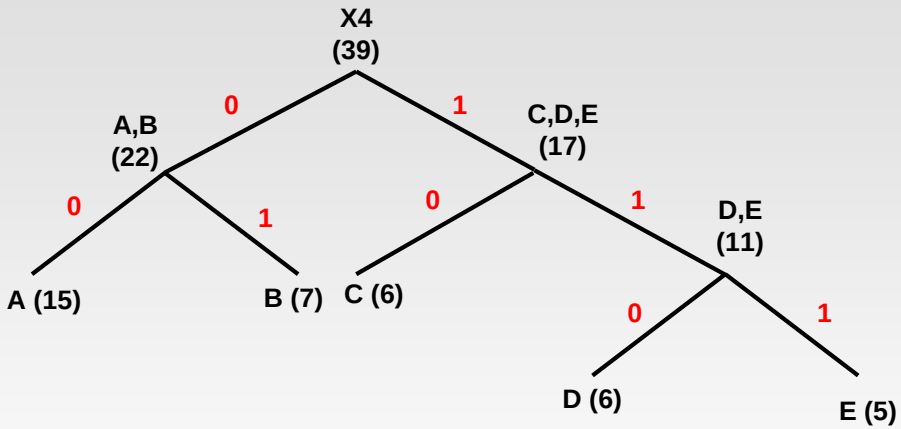
- At each step, takes the two nodes with the lowest probability, creates a subtree with probability equals to the sum of the probabilities of each node
- Labels right branches with bit 1, left branches with bit 0
- The code of each symbol is the label of the path from the root of the tree



simbolo	#	$\log_2(1/p_i)$	codice	# di bit
A	15	1.38	0	15
B	7	2.48	100	21
C	6	2.70	101	18
D	6	2.70	110	18
E	5	2.96	111	15

N. di bit = 87 (teorico 85.29)

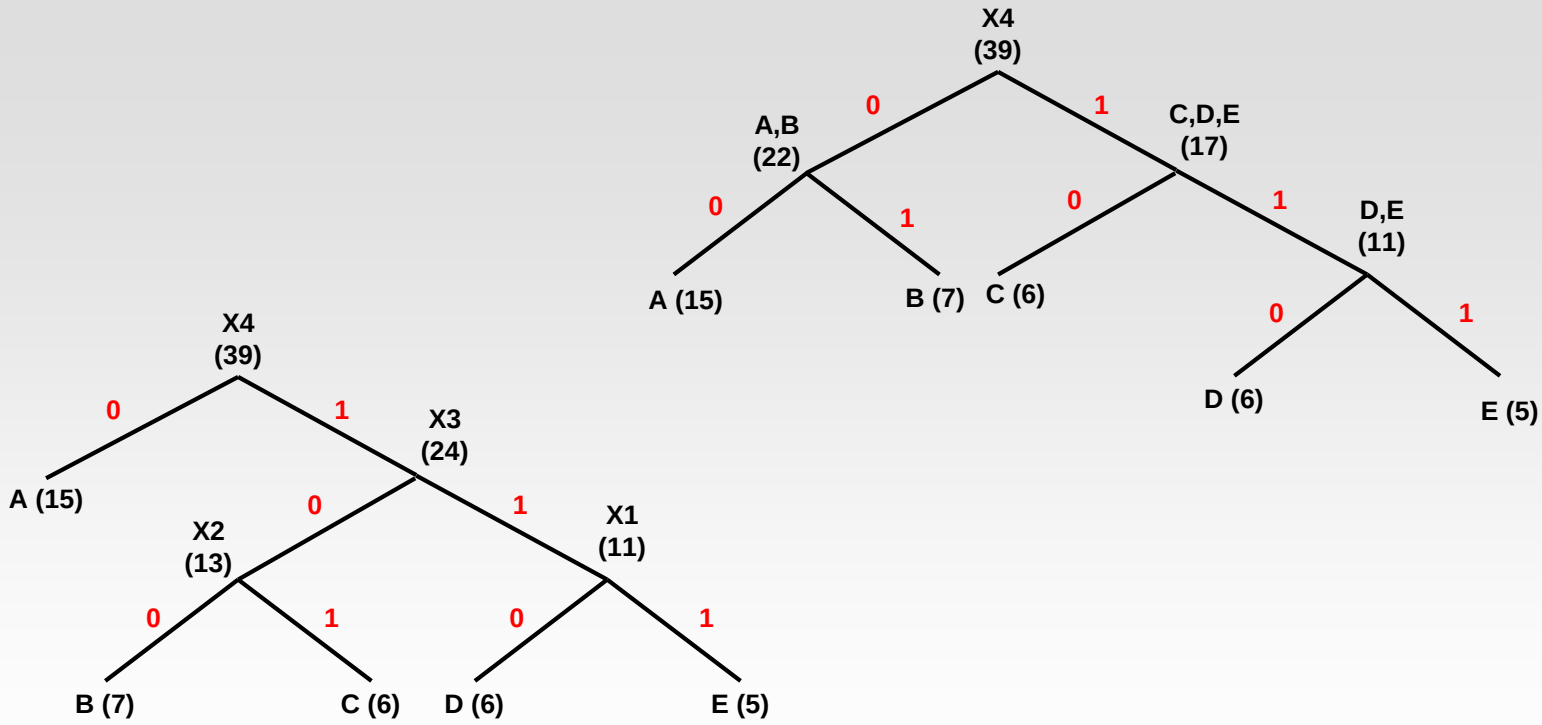
Huffman vs. Shannon-Fano



Symbol	#	$\log_2(1/p_i)$	Code	# of bit
A	15	1.38	00	30
B	7	2.48	01	14
C	6	2.70	10	12
D	6	2.70	110	18
E	5	2.96	111	15

N. of bit = 89 (87 Huffman, theorical 85.29)

Huffman vs. Shannon-Fano



N. of bit = 89 (87 Huffman, theorical 85.29)

LZW Compression

It dynamically builds the vocabulary of the symbols using fixed codes for sequences with variable length

```
w = NULL;
while (not EOF)
 { read a character k
   if wk exists in the dictionary
      w = wk;
   else
    { add wk to the dictionary;
      output the code for w;
      w = k;
    }
 } output the code for w;
```

w	k	output	code	symbol
NULL	a			
a	b	a	256	ab
b	c	b	257	bc
c	d	c	258	cd
d	a	d	259	da
a	b			
ab	c	256	260	abc
c	a	c	261	ca
a	b			
ab	c			
abc	c	260	262	abcc
c	a			
ca	b	261	263	cab
b	c			
bc	e	257	264	bce
e	a	e	265	ea
a	b			
ab	c			
abc	f	260	266	abcf
f	EOF	f		

LZW Decoding

Decompression recreates the vocabulary while expanding the text

```
read a character k;
output k;
w = k;
while ( read a symbol k )
 { entry = dictionary entry for k;
   output entry;
   add w + entry[0] to dictionary;
   w = entry;
 }
```

w	*k*	*output*	*code*	*symbol*
	a	a		
a	b	b	256	ab
b	c	c	257	bc
c	d	d	258	cd
d	256	ab	259	da
ab	c	c	260	abc
c	260	abc	261	ca
abc	261	ca	262	abcc
ca	257	bc	263	cab
bc	e	e	264	bce
e	260	abc	265	ea
abc	f	f	266	abcf
f	EOF			

Particular case

Consider the string: ababbabcabbabbax

```
w = NULL;
while (not EOF)
 { read a character k
   if wk exists in the dictionary
      w = wk;
   else
    { add wk to the dictionary;
      output the code for w;
      w = k;
    }
 } output the code for w;
```

w	k	output	code	symbol
NULL	a			
a	b	a	256	ab
b	a	b	257	ba
a	b			
ab	b	256	258	abb
b	a			
ba	b	257	259	bab
b	c	b	260	bc
c	a	c	261	ca
a	b			
ab	b			
abb	a	258	262	abba
a	b			
ab	b			
abb	a			
abba	x	262	263	abbax
x	EOF	X		

Decoding failure

Consider the string: ab 256 257 bc 258 262 x

```
read a character k;
output k;
w = k;
while ( read a symbol k )
 { entry = dictionary entry for k;
   output entry;
   add w + entry[0] to dictionary;
   w = entry;
 }
```

w	*k*	*output*	*code*	*symbol*
	a	a		
a	**b**	**b**	**256**	**ab**
b	**256**	**ab**	**257**	**ba**
ab	**257**	**ba**	**258**	**abb**
ba	**b**	**b**	**259**	**bab**
b	**c**	**c**	**260**	**bc**
c	**258**	**abb**	**261**	**ca**
abb	**262**	??	??	??

Solution

Character + String + character concatenation is the reason of the failure of the decoder (a + bb + a + …)

```
read a character k;
output k;
w = k;
while ( read a symbol k )
 { entry = dictionary entry for k;
   /*Exception Handling*/
   if (entry == null)
     entry = w + w[0];
   output entry;
   add w + entry[0] to dictionary
   w = entry;
 }
```

w	k	output	code	symbol
	a	a		
a	b	b	256	ab
b	256	ab	257	ba
ab	257	ba	258	abb
ba	b	b	259	bab
b	c	c	260	bc
c	258	abb	261	ca
abb	262	abba	262	abba
abba	x	x	263	abbax
x	EOF			