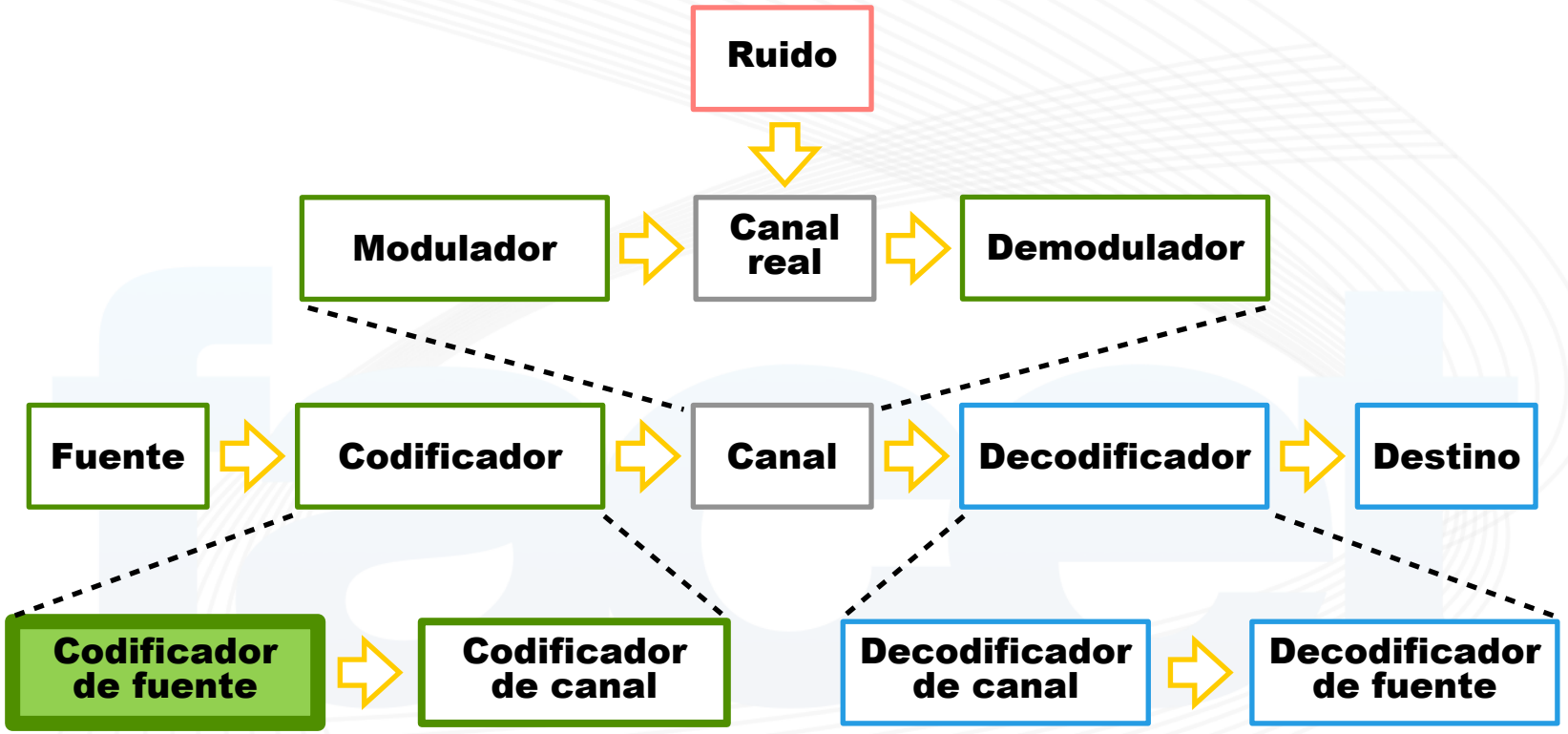
Codificaciones de fuente

Transmisión de Datos

Ing. Luis Di Pinto (ldipinto@herrera.unt.edu.ar)

http://www.microprocesadores.unt.edu.ar/transmision/

Repaso Esquema general



- Nos enfocaremos en el codificador de fuente.

Codificación de fuente

- La codificación de fuente extrae la información de la fuente y la representa con un alfabeto binario.
- Se realiza antes que la información sea transmitida (o almacenada).
  - Y es previa a las codificaciones de línea y de canal.
- El objetivo amplio es eliminar ineficiencias o redundancias en la información provista por la fuente para así optimizar el uso del canal.

Codificación de fuente

- En otras palabras, busca **comprimir el mensaje**.
  - Transmitir la misma cantidad de información, pero utilizando menor cantidad de bits.
  - Es por ello que la codificación de fuente se la conoce como **compresión de datos**.
  - Técnicas muy utilizadas, principalmente para la transmisión de imágenes, de audio y de video.

Temas que veremos

- Introducción a la compresión de datos.
- Introducción a la teoría de la información.
  - Entropía.
- Codificaciones sin pérdida.
  - Códigos de Huffman.
  - Códigos de directorios (LZ).
- Codificaciones con pérdida.
  - Transformada Discreta del Coseno.

Tipos de información

- La información de la fuente se divide en:
  - **Básica**: es la información relevante y que debe ser transmitida para poder reconstruir el mensaje.
  - **Redundante**: corresponde a información repetitiva o predecible.
  - **Irrelevante**: información que los humanos no podemos apreciar, y que se podría eliminar casi sin alterar el mensaje.

Tipos de compresión

- Existen dos formas de compresión:
  - **Sin pérdidas reales (*lossless*):** se transmite toda la información básica e irrelevante del mensaje, eliminando sólo la redundante.
  - **Subjetivamente con pérdidas (*lossy*):** se elimina una parte de la información básica, por lo que el mensaje se reconstruye con errores perceptibles pero tolerables.

Codificadores

- Típicamente, los dispositivos o programa encargados de la codificación de fuente, o compresión de información, son llamados codificadores o compresores.
- A su vez, los que realizan el proceso inverso son llamados decodificadores o descompresores.
- Lo más usual, es que ambos sean partes independientes del mismo dispositivo, que se encarga tanto de la codificación como de la decodificación.
  - Y reciben el nombre de **codecs**.
- Existen varios esquemas, con diferentes parámetros y compromisos:
  - El grado o factor de compresión.
  - La capacidad de cálculo necesaria para la codificación/decodificación.
  - La cantidad de distorsión introducida (en el caso de compresiones con pérdidas)

Compresiones sin pérdida (*lossless*)

- No pierden información.
- Son procesos normalmente reversibles.
- Eliminan la información redundante **estadísticamente**.
  - Por ejemplo, en vez de transmitir "1111111111111111" se podría transmitir "16 '1's".
- Los primeros métodos son basados en el concepto de **Entropía**.
  - Es difícil obtener factores de compresión mayores a 2:1.

Compresiones sin pérdida (*lossless*)

- Los algoritmos de compresión sin pérdida más populares son los de **Lempel-Ziv** (LZ):
  - Hay muchas variantes, pero todos son basados en **diccionarios (tablas)**, que usualmente son generados dinámicamente.
  - Por ejemplo, el algoritmo LZW es usado en GIF, mientras que el algoritmo DEFLATE es usado en PNG y en ZIP.
  - Se pueden obtener factores de compresión del orden de 5:1.
- Los más modernos, usan modelos probabilísticos y aritméticos.
  - Permiten alcanzar mayores niveles de compresión.
  - Por ejemplo, se usa en video (MPEG-4 o H.264) o audio (FLAC).

Compresiones con pérdida (*lossy*)

- Muchos se basan en investigaciones sobre la manera en la que los humanos percibimos la información.
- Debajo hay dos imágenes de piedras en un estanque, una comprimida y la otra no.
- ***¿Pueden distinguir cuál es cuál?***

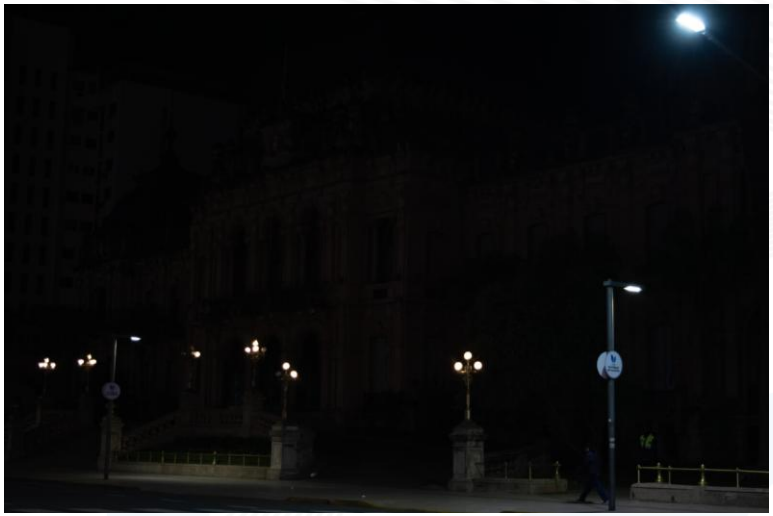


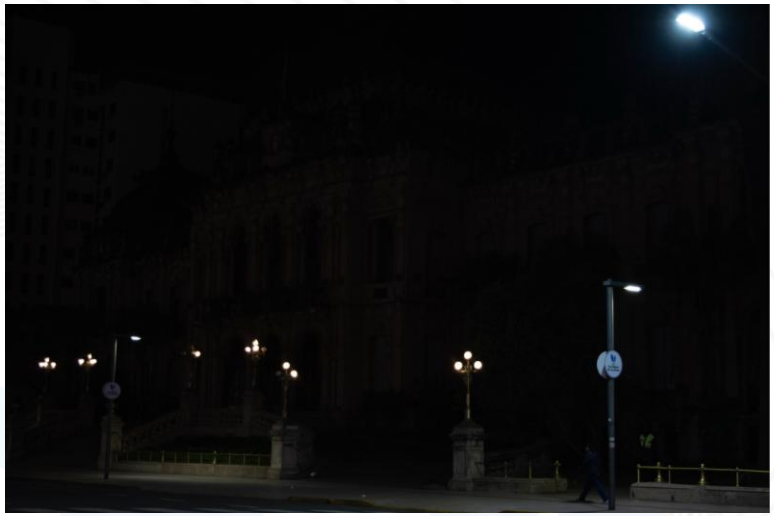


Compresiones con pérdida (*lossy*)

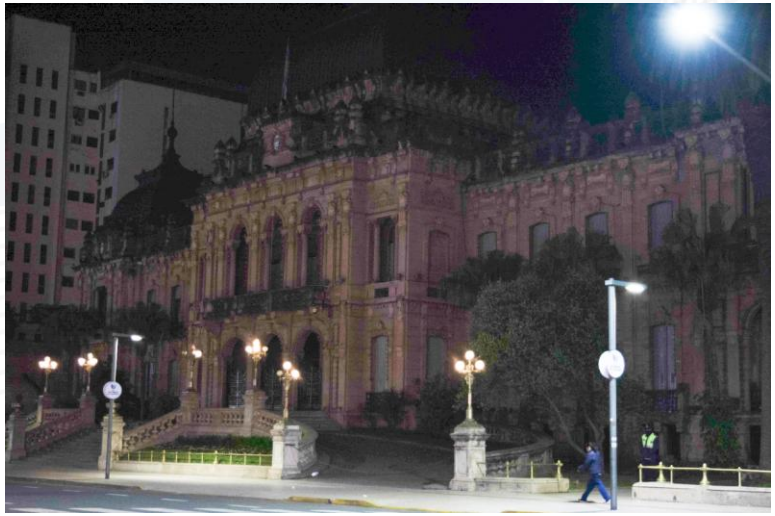
- Toleran cierta pérdida de información.
  - Por lo tanto, están en constante compromiso.
  - Mayor compresión → Menor tamaño → Mayor pérdida de información.
- Son procesos normalmente irreversibles.
- La mayoría usa métodos de compresión basados en la **Transformada Discreta del Coseno (DCT)**.
  - Algoritmo creado por Nasir Ahmed.
  - JPEG, MPEG, MP3, cámaras digitales, streaming, y muchos etc.
  - Pueden obtener factores de compresión del orden 50:1.

Ejemplos de Compresión lossy

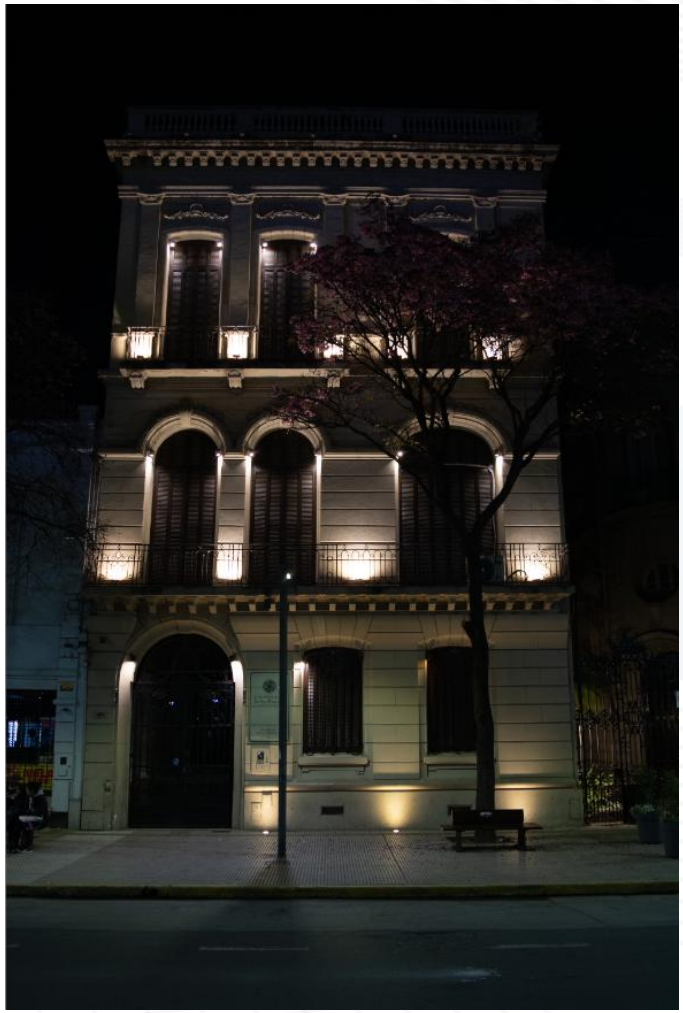


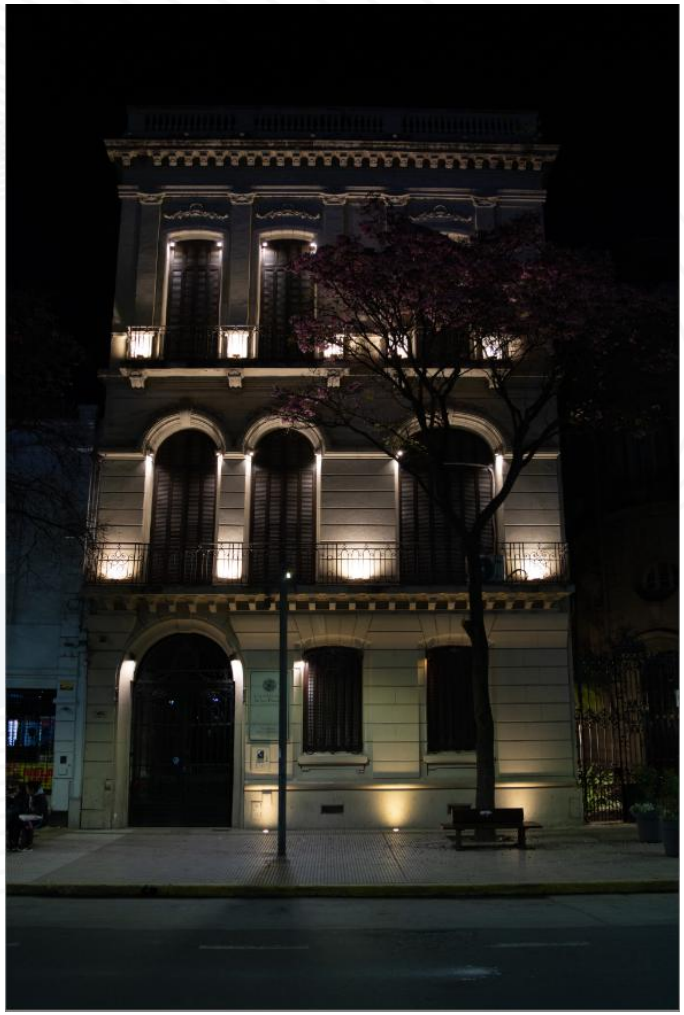




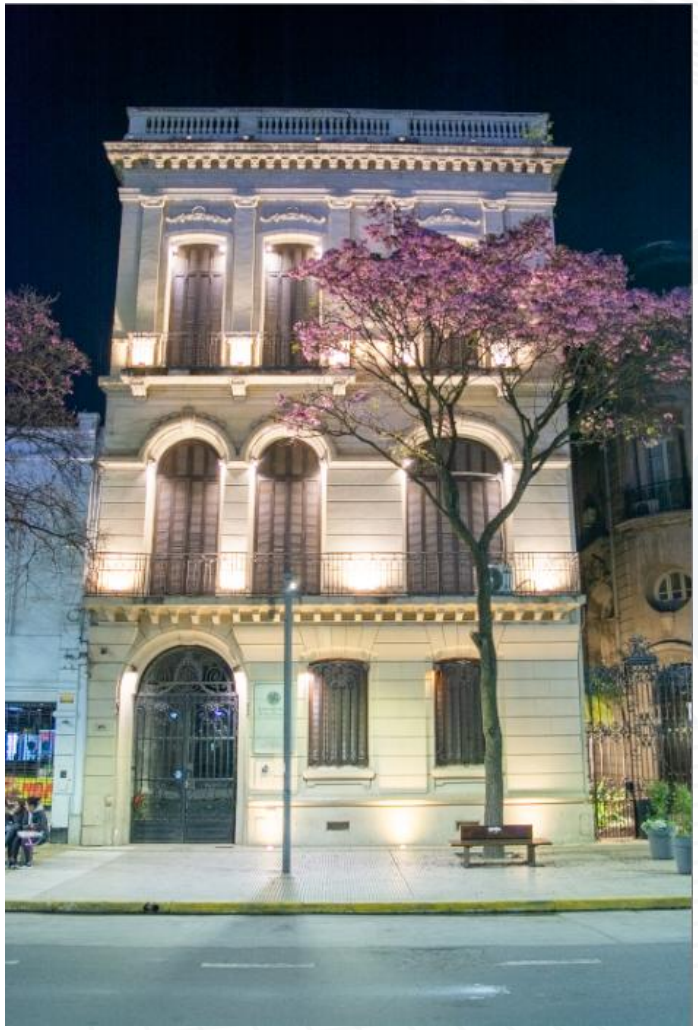


Ejemplos de Compresión lossy

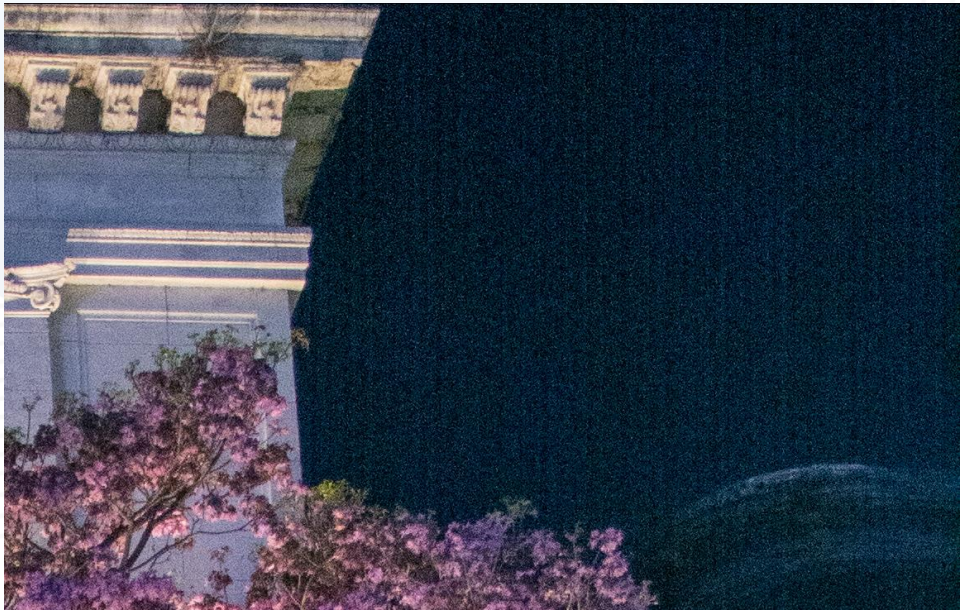




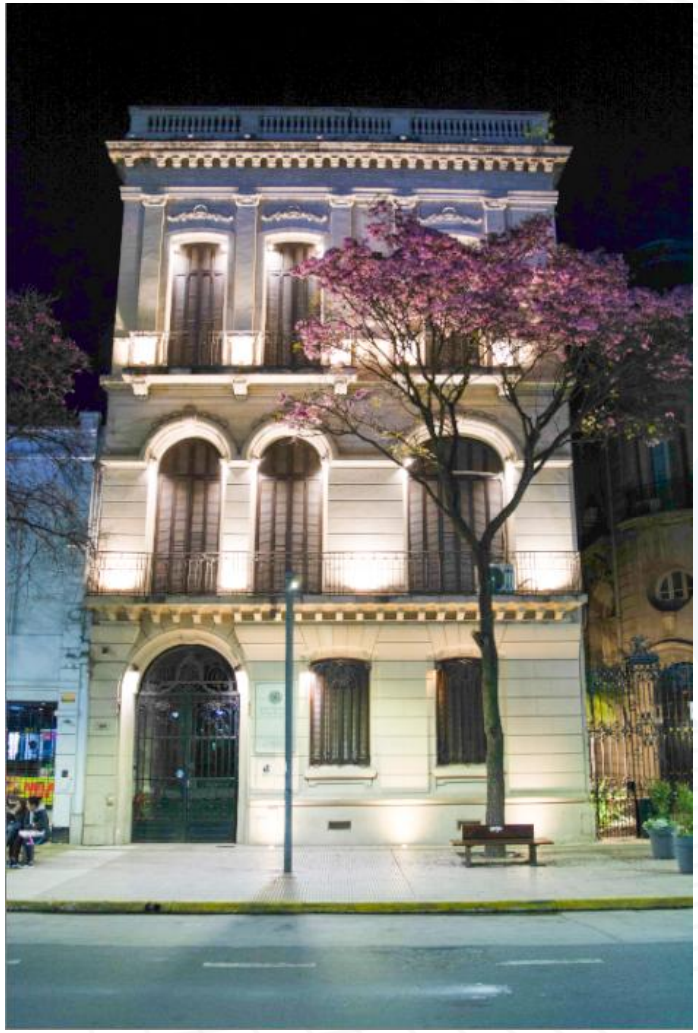
RAW editado



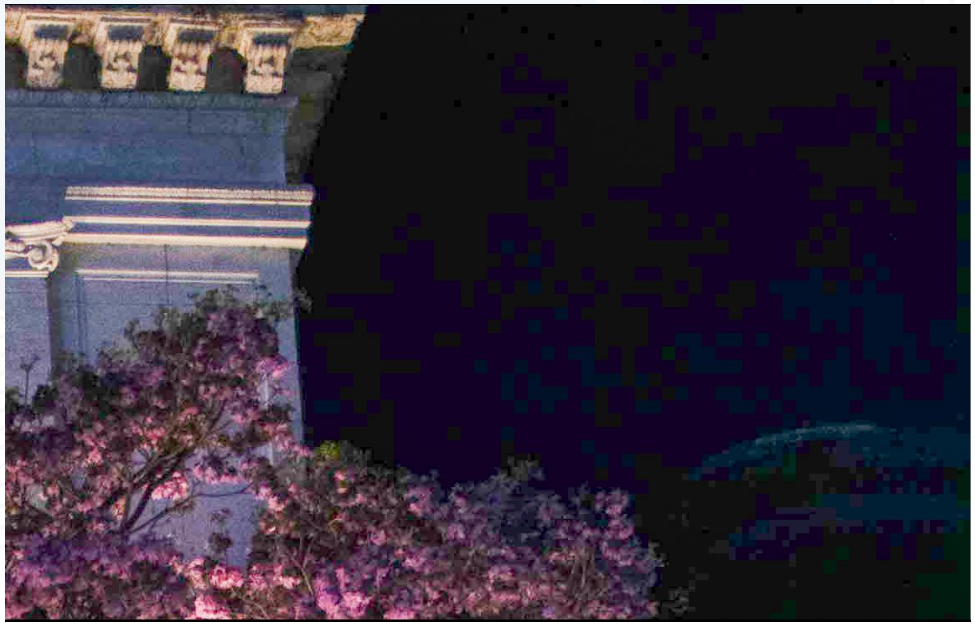




JPG editado







Medida de la información

- La idea principal de la teoría de la información es que el "valor informativo" de un mensaje depende de cuán **sorprendente** sea.
- Shannon, al sentar las bases de la teoría de la información, estableció algunos axiomas:
  - Un evento cuya probabilidad de ocurrencia es 1, no transmite ninguna información.
  - Cuanto menor sea la probabilidad de un evento, mayor será la cantidad de información que transmita.
- Se puede encontrar que existe una relación entre la probabilidad de un mensaje y la información que transmite, que se expresa matemáticamente como:

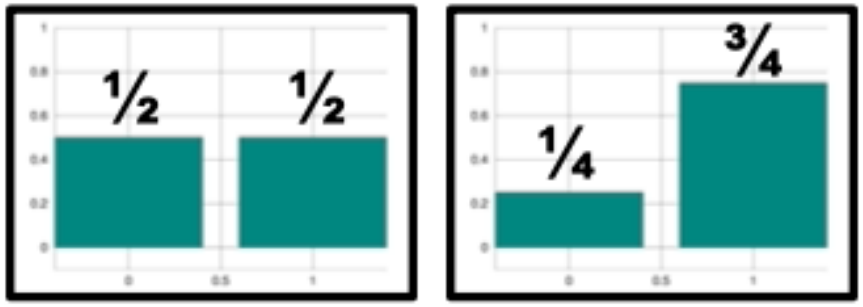
$$Info = -log_2(Prob(X))$$

Fuentes discretas

- Una fuente discreta utiliza un alfabeto para codificar la información.
- Un alfabeto está compuesto por un conjunto definido de símbolos.
- Un mensaje es una secuencia ordenada de símbolos que pertenecen al alfabeto.
- Se puede calcular una probabilidad de aparición de cada símbolo en un mensaje.

Fuentes discretas

- En un mensaje binario codificado en bits el alfabeto está compuesto por solo dos símbolos: 0 y 1.
- Si en los mensajes se usan con la misma frecuencia ambos símbolos tendrán probabilidad 1/2.
- Pero podría ocurrir que alguno de los dos símbolos se utilice más frecuentemente que el otro.



Fuentes con y sin memoria

- Una fuente no tiene memoria cuando la probabilidad de cada letra que forma el mensaje es independiente de las letras anteriores.

01001010100101010101011101100
11010111101010100101010100110

wqmfotisuxelrpyredkjweo
iuqwzmdapquresnqbsmqs

00000000011111111111111111111
00000111111111111111110000000

Hola como estas?
Yo por suerte estoy bien

Fuentes estacionarias

- Una fuente es estacionaria cuando sus propiedades estadísticas no cambian en el tiempo.

```
0100101010011100011010111101010100101010011001010110101010110101
1010100101010011001010110101111001101010110001101011110101010101
1010110101110010101001100101100011010111101010010101001100
```

```
0110100010001110010101010100010001010010101000001000001001010001
0001000010100000000100000000010000000001001000000000000010000000
0000000000010000000000001000000000000000000000000001000000000
```

Entropía de la fuente

- En función de la probabilidad se puede calcular la información transmitida por cada símbolo.
  - Como vimos anteriormente.
- A partir de la información de cada símbolo, se puede calcular también la información de un mensaje, considerando la probabilidad como la frecuencia de aparición del símbolo en el mensaje.

Entropía de la fuente

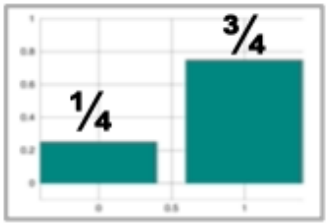
- La entropía de un símbolo es el promedio de la información que transporta cada símbolo.

$$H(X) = \sum p(x) \quad \cdot I(X)$$

- La entropía de un mensaje es, para una fuente discreta, estacionaria y sin memoria, igual a la longitud del mensaje por la entropía de cada letra.

Entropía de la fuente





Información de cada simbolo

$I(X)_{X=0} = -log_2(1/2) = 1$

$I(X)_{X=1} = -log_2(1/2) = 1$

$I(X)_{X=0} = -log_2(1/4) = 2$

$I(X)_{X=1} = -log_2(3/4) = 0{,}415$

Información promedio de un simbolo

$$H(X) = \sum_{i=0}^{1} p(x)_{x=i} \cdot I(X)_{X=i}$$

$$H(X) = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1 = 1$$

$$H(X) = \sum_{i=0}^{1} p(x)_{x=i} \cdot I(X)_{X=i}$$

$$H(X) = \frac{1}{4} \cdot 2 + \frac{3}{4} \cdot 0{,}415 = 0{,}811$$

Información del mensaje

$H(M) = H(\{X_1, X_2, \ldots, X_L\})$

$H(M) = H(X_1) + \ldots + H(X_L)$

$H(M) = L \cdot H(X) = L$

$H(M) = H(\{X_1, X_2, \ldots, X_L\})$

$H(M) = H(X_1) + \ldots + H(X_L)$

$H(M) = L \cdot H(X) = 0{,}811 \cdot L$

Entropía de la fuente

- La entropía define de esta manera la capacidad mínima de un canal requerida para transmitir un mensaje sin errores.
- Se defina la tasa de entropía de una fuente como el promedio de bits necesarios para codificar sus símbolos.
- Impone un límite a los factores de compresión:
  - Cualquier método de compresión de datos sin pérdidas debe tener una longitud de código esperada mayor o igual que la entropía de la fuente.
  - Es el teorema de Shannon sobre la codificación de fuente.

Codificadores sin pérdida

- La mayoría de los codificadores sin pérdida realizan dos tareas de manera secuencial.
- Primero, generan un modelo estadístico de los datos de entrada.
- Luego, usa ese modelo para realizar un mapeo (codificación) de los datos de entrada en una secuencia de símbolos.
- El mapeo se hace de manera que los datos más probables (más frecuentes) producen una secuencia de símbolos más corta que los datos menos frecuentes.
  - O sea, un código de longitud variable.

Códigos de largo variable

- Son útiles cuando las letras de la fuente no son equiprobables.
- Se asigna el código más corto a la letra más frecuente y el más largo a la menos frecuente.
  - Un ejemplo es el código Morse.
- Tomemos por ejemplo una fuente que genera mensajes con un alfabeto de cuatro letras con las siguientes probabilidades.
  - A={00} con p_1=1/2
  - B={01} con p_2=1/4
  - C={10} con p_3=1/8
  - D={11} con p_4=1/8

Códigos de largo variable

- Analizando las probabilidades de cada letra se podrían generar la siguientes opciones:

Letra	Probabilidad	Original	Código 1	Código 2	Código 3
A	1/2	00	1	0	0
B	1/4	01	00	10	01
C	1/8	10	01	110	011
D	1/8	11	10	111	111

Códigos de largo variable

- Si se calcula el largo promedio de un mensaje de L letras para cada código

- $L_0 = \frac{1}{2} \cdot 2bits + \frac{1}{4} \cdot 2bits + \frac{1}{8} \cdot 2bits + \frac{1}{8} \cdot 2bits = 2bits$
- $L_{C1} = \frac{1}{2} \cdot 1bit + \frac{1}{4} \cdot 2bits + \frac{1}{8} \cdot 2bits + \frac{1}{8} \cdot 2bits = 1{,}5bits$
- $L_{C2} = \frac{1}{2} \cdot 1bit + \frac{1}{4} \cdot 2bits + \frac{1}{8} \cdot 3bits + \frac{1}{8} \cdot 3bits = 1{,}75bits$
- $L_{C3} = \frac{1}{2} \cdot 1bit + \frac{1}{4} \cdot 2bits + \frac{1}{8} \cdot 3bits + \frac{1}{8} \cdot 3bits = 1{,}75bits$

Decodificación única e instantánea

- ***¿Cómo se interpreta la secuencia S={001001}?***

Letra	Original	Código 1	Código 2	Código 3
A	00	1	0	0
B	01	00	10	01
C	10	01	110	011
D	11	10	111	111

- C_O: Si, Si ⇒ M = {A,C,B}
- C_1: No, No ⇒ M = {B,A,B,A} o M = {B,D,C}
- C_2: Si, Si ⇒ M = {A,A,B,A} La letra termina en 0 o con tres bits
- C_3: Si, No ⇒ M = {A,B,A,B}

Códigos de largo variable

- Resumen de las propiedades de cada código:

Propiedad	Original	Código 1	Código 2	Código 3
Largo promedio	2 bits	1,5 bits	1,75 bits	1,75 bits
Decodificación única	Si	No	Si	Si
Decodificación Instantánea	Si	No	Si	No

- Sólo nos interesan los códigos con decodificación única e instantánea, y además lo más cortos posibles.
- Para garantizar la decodificación única e instantánea **ningún código debe ser prefijo de otro código**.

Códigos de largo variable

- Para una fuente discreta sin memoria es posible construir un código que cumpla con:

$$H(x) \leq L \leq H(x) + 1$$

- En el ejemplo anterior:

$$H(x) = -\left(\frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) + \frac{1}{4} \cdot log_2\left(\frac{1}{4}\right) + \frac{1}{8} \cdot log_2\left(\frac{1}{8}\right) + \frac{1}{2} \cdot log_2\left(\frac{1}{2}\right)\right)$$

$$H(x) = -\left(\frac{1}{2} \cdot -1 + \frac{1}{4} \cdot -2 + \frac{1}{8} \cdot -3 + \frac{1}{8} \cdot -3\right) = \left(\frac{1}{2} + \frac{2}{4} + \frac{2 \cdot 3}{8}\right) = 1{,}75 bits$$

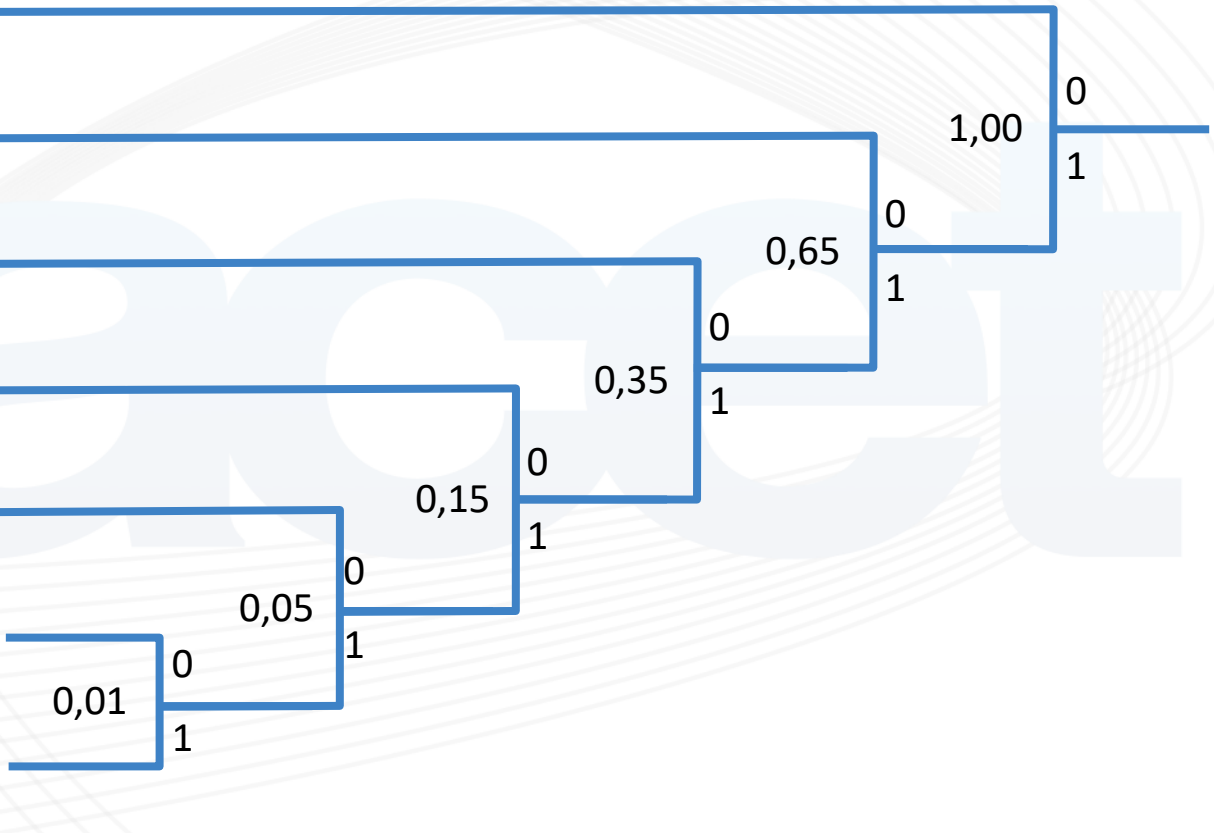
- Se definen:
  - Eficiencia $\eta = \frac{H(x)}{L}$, y en el ejemplo anterior vale 1.
  - Redundancia $\gamma = 1 - \eta$, y en el ejemplo anterior vale 0.
  - Si no hay redundancia, el código no se puede comprimir.

Códigos de Huffman

- Son un tipo de codificación de longitud variable que generan un código óptimo.
- Se construye una tabla con la frecuencia de ocurrencia (o la probabilidad estimada) de cada letra del alfabeto.
- Luego, el código se construye como un árbol binario sobre la tabla anterior ordenada.
  - Tomando siempre en cada paso las dos letras con menor probabilidad.

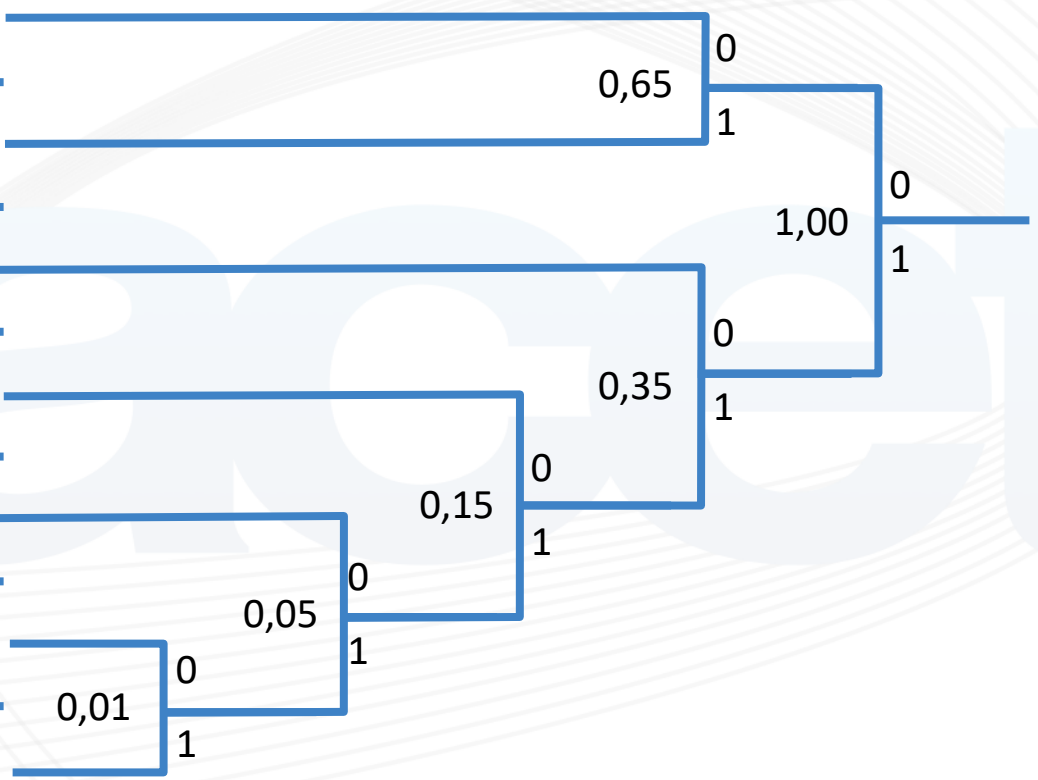
Código de Huffman

Letra	Probabilidad
x_1	0,35
x_2	0,30
x_3	0,20
x_4	0,10
x_5	0,04
x_6	0,005
x_7	0,005



Código de Huffman

Letra	Probabilidad
x_1	0,35
x_2	0,30
x_3	0,20
x_4	0,10
x_5	0,04
x_6	0,005
x_7	0,005



Código de Huffman

- La codificación no es única, se pueden obtener más de un código con las mismas propiedades.
- La tabla que resulta con el mapeo de cada símbolo en cada codificación binaria, se denomina **diccionario**.

Letra	Probabilidad	Información	Código 1	Código 2
x_1	0,35	1,5146	00	0
x_2	0,30	1,7370	01	10
x_3	0,20	2,3219	10	110
x_4	0,10	3,3219	110	1110
x_5	0,04	4,6439	1110	11110
x_6	0,005	7,6439	11110	111110
x_7	0,005	7,6439	11111	111111
	H(x) = 2,11		L = 2,21	L=2,21

Problemas del código de Huffman

- Para hallar el código es necesario conocer las probabilidades de cada símbolo.
- Esto implica disponer de todo el mensaje para calcular las probabilidades.
- La alternativa es perder eficiencia usando aproximaciones de las probabilidades.
- El resultado es que no resulta práctico para la mayoría de los casos.

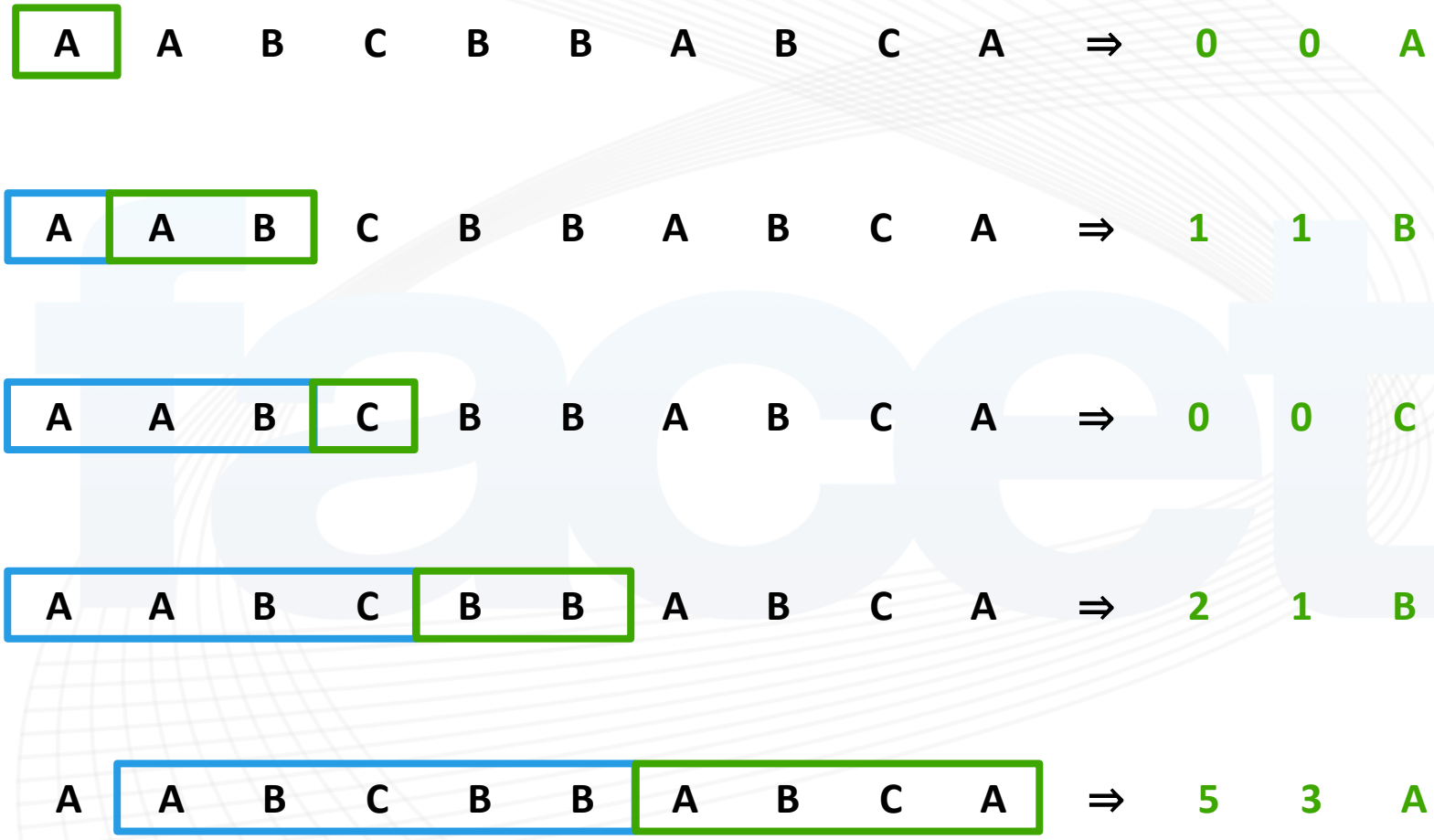
Algoritmo LZ77 (o LZ1)

- Es un algoritmo de compresión sin pérdidas que sirve para cualquier tipo de fuente.
- Funciona reemplazando las repeticiones por referencias a una sola copia de esos datos.
  - Se busca siempre **la cadena repetida más larga**.
  - Cada referencia es codificada utilizando un par *distancia-longitud*.
  - “Cada uno de los siguientes [*longitud*] caracteres son iguales a los caracteres encontrados exactamente [*distancia*] posiciones atrás”.

Algoritmo LZ77 (o LZ1)

- La codificación se produce generando una terna (d, l, c):
  - d y l son los valores de referencia ya mencionados.
  - c es el siguiente caracter que no coincide con la referencia.
- Para detectar las repeticiones utiliza una ventana deslizante, que es equivalente al uso de un diccionario.
  - El contenido del diccionario va cambiando con el tiempo.
  - La ventana mantiene los últimos N bytes procesados.
    - Tamaños normales para la ventana son 2, 4 y 32 KB.

Algoritmo LZ77 (o LZ1)



Variantes de LZ77

- La versión original genera siempre tres valores para todos los datos, y codifica los literales con distancia y longitud igual a cero.
  - Esto da pie a muchas variantes que modifican la manera de generar los pares (d,l).
- LZSS utiliza una bandera de un bit para distinguir literales de pares posición-longitud.
  - Evita así poner ceros.
- El algoritmo DEFLATE combina LZSS con un código Huffman adaptable.
  - Se genera un alfabeto con los literales, las longitudes y un caracter especial para indicar el fin del bloque.

Algoritmo LZ78 (o LZ2)

- LZ77 solo puede referenciar el contenido de la ventana deslizable.
- Aumentar la ventana implica más cantidad de bits en los pares posición-longitud
- Tambien aumenta el tiempo que toma buscar las repeticiones en la ventana.
- El algoritmo LZ78 resuelve estos problemas utilizando un diccionario.
  - Cada entrada del diccionario es una dupla (*índice*, *token*).
    - El *índice* es una referencia a una entrada ya perteneciente al diccionario.
    - El *token* es la siguiente secuencia de datos que no es parte del diccionario.

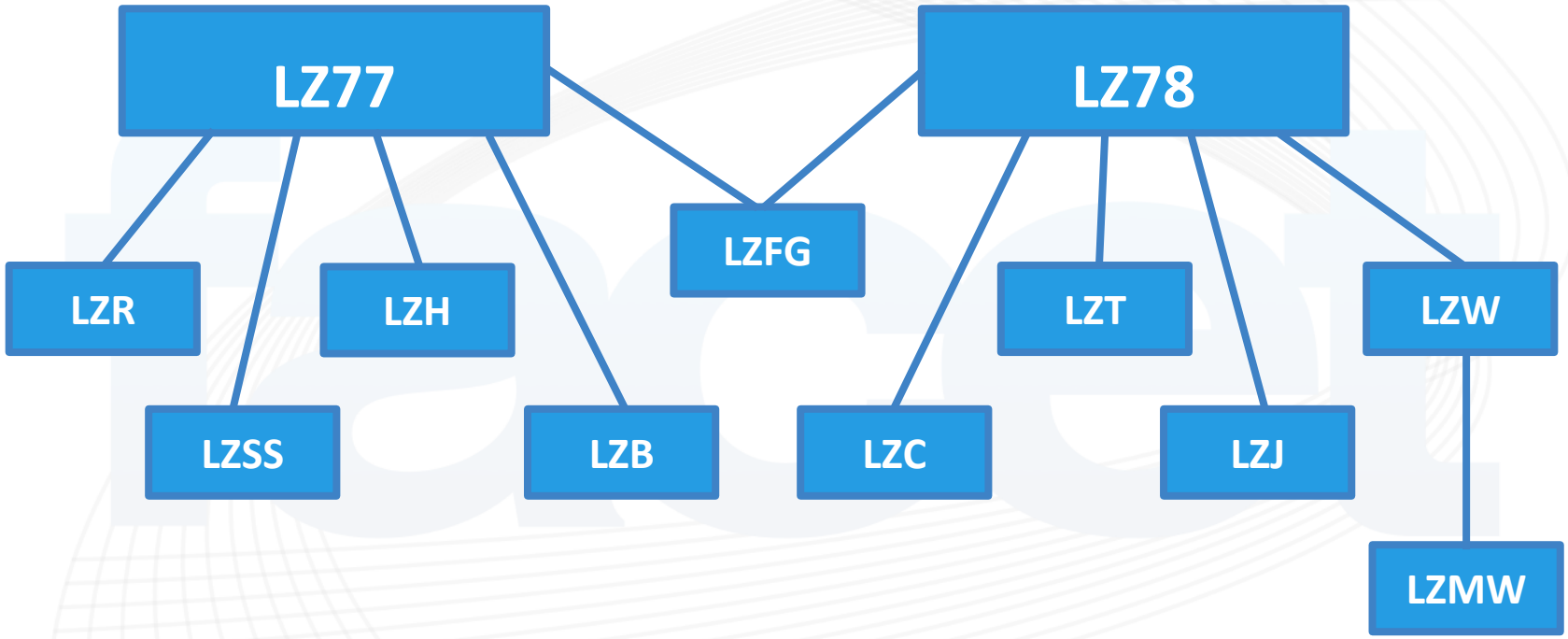
Algoritmo LZ78 (o LZ2)

A	A	B	C	B	B	A	B	C	A	⇒	1	0	A
A	**A**	B	C	B	B	A	B	C	A	⇒	2	1	A
A	A	**B**	C	B	B	A	B	C	A	⇒	3	0	B
A	A	B	**C**	B	B	A	B	C	A	⇒	4	0	C
A	A	B	C	**B**	**B**	A	B	C	A	⇒	5	3	B
A	A	B	C	B	B	**A**	**B**	C	A	⇒	6	1	B
A	A	B	C	B	B	A	B	**C**	**A**	⇒	7	4	A

Variantes de LZ78

- Tanto LZ78 como LZ77 pueden producir salidas más grandes que la entrada original, sobre todo en cadenas cortas.
- Ambos funcionan muy bien cuando el diccionario (o la ventana) son grandes.
- LZW es una variante de LZ78 que usa un diccionario inicializado previamente con todos los símbolos posibles.
  - De esta manera, se codifica solamente el índice.

Familia de algoritmos LZ



Codificadores con pérdida

- Hay dos esquemas básicos:
  - Compresión por transformación: se dividen los datos de entrada en segmentos pequeños, que son convertidos a un nuevo espacio de representación, cuantificados y codificados.
  - Compresión por predicción: se usa parte de los datos de entrada para predecir los siguientes, y se codifica el error de la predicción.
  - Muchas veces se usan ambos.

Transformada Discreta del Coseno (DCT)

- La idea básica es expresar una secuencia finita de datos como una suma de cosenos de distintas frecuencias y amplitudes.
- Es similar a la Transformada Discreta de Fourier (DFT), pero sólo con valores reales.
- Comprime los datos en bloques.
  - Por ejemplo, en imágenes, bloques de 8x8 píxeles.
  - La transformada se aplica a cada fila y a cada columna del bloque.
  - Muchas veces, la información tiende a estar concentrada en pocos componentes de baja frecuencia.
  - El resultado es un matriz de 8x8 con coeficientes, en el que el elemento (0,0) posee el valor de la componente continua y el resto los coeficientes de distintas frecuencias.
  - Finalmente, estos coeficientes son cuantificados y codificados con alguna de las técnicas vistas anteriormente.
- Hay muchas variantes, inclusive una que comprime sin pérdidas.

DCT modificada (MDCT)

- Es una variante de la DCT, pero en la que los bloques se superponen.
  - La segunda mitad de un bloque se superpone con la primera mitad del bloque siguiente.
  - Evita problemas que pueden ocurrir al definir de manera fija los bloques, por los límites entre ellos.
  - Por ello, es la variante más usada en datos que son continuos, como los de audio.

Resumen

- Codificación de fuente = Compresión de datos
  - Transmitir la misma cantidad de información, con la menor cantidad de bits posible.
- Tipos de compresión: Sin pérdidas (lossless) y con pérdidas (lossy).
- Teoría de la Información
  - $I(X) = -\log_2(p(X))$
  - Entropía de la fuente: $H(X) = \sum p(X) * I(X)$
    - Establece un límite mínimo para la capacidad del canal.
    - Establece un mínimo para la longitud de código esperada.
- Compresión sin pérdidas
  - Primero hacen un muestreo estadístico de los datos de entrada.
  - Luego usan ese modelo para codificar los símbolos, usualmente con longitud variable (Huffman).
  - Códigos Lempel-Ziv, basados en directorios.
- Compresión con pérdidas, basada en la Transformada Discreta del Coseno (DCT).

Agradecimientos

- Las diapositivas de este tema fueron basadas en las realizadas por los Ingenieross Esteban Volentini y Nicolás Majorel Padilla.