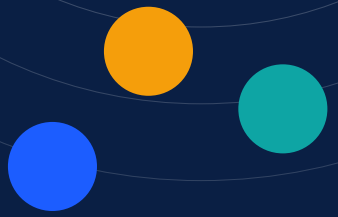
1 0 1 1 0 1 0 0 1

TEORÍA DE LA INFORMACIÓN

# Información

*De Hartley a Shannon: medir, codificar y proteger*

Georgy Nuzhdin · U-TAD · EPM



$$i = -\log_2 p$$

01

PARTE I

# Información según Hartley

DEFINICIÓN INTUITIVA

# ¿Qué es la información?

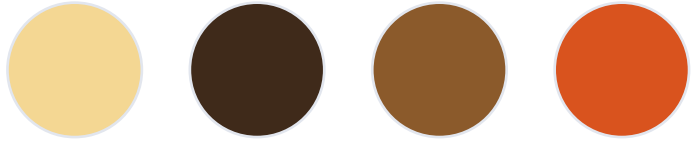
Imagina todos los mundos posibles. Cada dato que recibes reduce ese conjunto: te dice en cuál de ellos estás. La información mide cuánto se reduce.

**Fórmula de Hartley**

$$i = -\log_2 ( \text{opciones que se dan} / \text{total de opciones} )$$

*Se mide en bits. Un bit = elegir entre dos opciones equiprobables.*

**Ejemplo: el color del pelo**



Rubio Moreno Castaño Pelirrojo

- Si hay 4 colores equiprobables, saber que alguien tiene el pelo moreno nos aporta:
- $i = -\log_2\left(\frac{1}{4}\right) = 2$ bits

- Si los pelirrojos son ~2% de la población, saber que alguien es pelirrojo, nos aporta:
- $i = -\log_2(0,02) \approx 5.6$ bits

# Espacio muestral e información

**Definición**

Sea A un conjunto finito (espacio muestral). Para identificar un elemento de A necesitamos exactamente

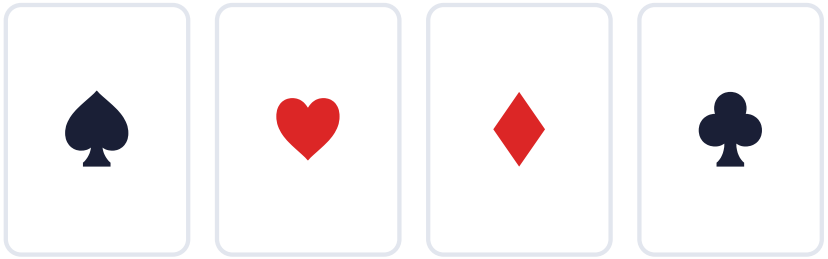
$$\log_2 |A| \quad \text{bits}$$

Si después nos informan que $x \in B \subset A$, hemos recibido

$$\log_2 |A| - \log_2 |B| = \log_2 ( |A|/|B| ) \quad \text{bits}$$

*Como $p(B) = |B|/|A|$, reencontramos $i = -\log_2 p$.*

**Ejemplo · Baraja francesa**



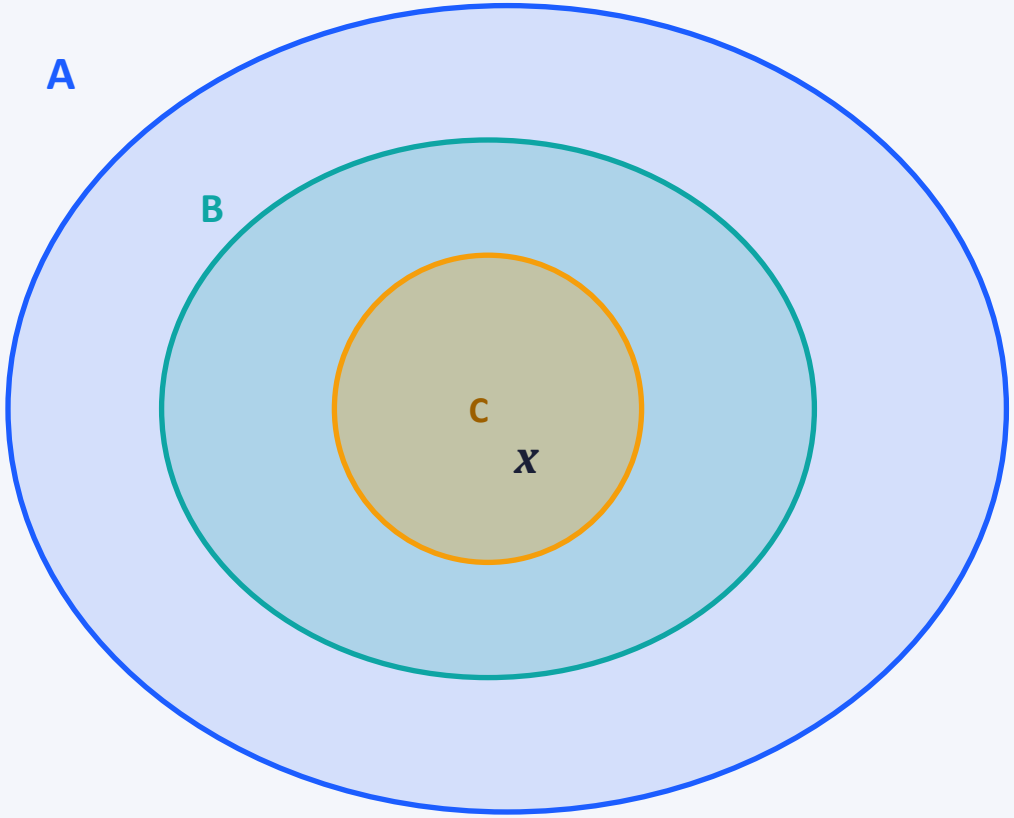
- Para determiner una carta al azar de 52 necesitamos
- $-\log_2 \mathbf{1/52} = \log_2 \mathbf{52} \approx \mathbf{5.7}$ bits

- Saber el palo nos aporta:
- $\log_2(\mathbf{52/13}) = \log_2 \mathbf{4} = \mathbf{2}$ bits

- **¿Cuánto aporta saber el valor de la carta pero no el palo?**
- $\log^2 \left(\frac{\mathbf{52}}{\mathbf{4}}\right) \approx \mathbf{5.7 - 2 = 3.7}$ bits

PROPIEDAD CLAVE

# La información es aditiva




**Mensajes encadenados**

Sabíamos que $x \in A$.

Nos dicen **x ∈ B**: $\log_2(|A|/|B|)$ bits.

Luego **x ∈ C**: $\log_2(|B|/|C|)$ bits.

**Total:**

$$\log_2(|A|/|B|) + \log_2(|B|/|C|) = \log_2(|A|/|C|)$$

EJEMPLO RESUELTO

# Sumando información

EJEMPLO

ENUNCIADO

Saco una carta al azar de una baraja francesa de 52 cartas. **Primero** me dicen que es ***roja***, y **luego** que es de ***corazones***. ¿Cuántos bits aporta cada mensaje, y cuántos los dos juntos?

EJEMPLO RESUELTO

# Sumando información

EJEMPLO

ENUNCIADO

Saco una carta al azar de una baraja francesa de 52 cartas. **Primero** me dicen que es ***roja***, y **luego** que es de ***corazones***. ¿Cuántos bits aporta cada mensaje, y cuántos los dos juntos?

SOLUCIÓN

**Paso 1: roja**

$|A| = 52 \rightarrow |B| = 26$

$$\log_2 (52 / 26) = 1 \text{ bit}$$

**Paso 2: corazones**

$|B| = 26 \rightarrow |C| = 13$

$$\log_2 (26 / 13) = 1 \text{ bit}$$

**Total acumulado**

$|A| = 52 \rightarrow |C| = 13$

$$\log_2 (52 / 13) = 2 \text{ bits}$$

**Comprobación de aditividad:** $1 + 1 = 2$ bits ✓

# El juego de las preguntas

RETO 1 DIFICULTAD 2

ENUNCIADO

Hay un número desconocido entre 1 y 1000. Se permite hacer cualquier pregunta con respuestas SÍ/NO.

**¿Cuántas preguntas son necesarias y suficientes para adivinar el número?**

# El juego de las preguntas

RETO 1 DIICULTAD 2

ENUNCIADO

Hay un número desconocido entre 1 y 1000. Se permite hacer cualquier pregunta con respuestas SÍ/NO. ¿Cuántas preguntas son necesarias y suficientes para adivinar el número?

SOLUCIÓN

Respuesta: **10 preguntas.**

**Suficiencia.** El número se escribe con 10 dígitos binarios; basta preguntar por cada bit.

**Necesidad (vía información).** Cualquier pregunta SÍ/NO aporta como mucho 1 bit: si A es el conjunto actual y $B \subset A$ es el subconjunto preguntado, alguno de B o su complemento tiene cardinal $\geq |A|/2$.

*Por aditividad, 9 preguntas dan como mucho 9 bits, y se necesitan $\log_2 1000 > 9$ bits.*

EJEMPLO RESUELTO

# Una pesada vale más que una pregunta

EJEMPLO

ENUNCIADO

Tenemos 9 monedas idénticas; exactamente una de ellas es falsa y más ligera. Disponemos de una balanza de platillos. ¿Cuántas pesadas bastan para encontrarla?

EJEMPLO RESUELTO

# Una pesada vale más que una pregunta

EJEMPLO

ENUNCIADO

Tenemos 9 monedas idénticas; exactamente una de ellas es falsa y más ligera. Disponemos de una balanza de platillos. ¿Cuánta información hay en este conjunto?

EJEMPLO RESUELTO

# Una pesada vale más que una pregunta

EJEMPLO

ENUNCIADO

Tenemos 9 monedas idénticas; exactamente una es falsa y más ligera. Disponemos de una balanza de platillos. ¿Cuántas pesadas bastan para encontrarla?

SOLUCIÓN

**Bastan 2 pesadas.**

Cada pesada tiene **3 resultados posibles** (izquierda más ligera, equilibrio, derecha más ligera).

Con 2 pesadas: $3 \times 3 = 9$ ***resultados*** distinguibles, justo lo que necesitamos.

¿Por qué no funciona $\log_2 9 > 3$ ?

Porque en cada paso dividimos en 3, no en 2.

Estrategia

**Pesada 1: 3 vs 3 (sobran 3)**


**Pesada 2: 1 vs 1 dentro del grupo identificado**

→ *si una baja, esa es la falsa; si están en equilibrio, lo es la tercera.*

*Si solo tuviéramos preguntas SÍ/NO haríamos falta* $\lceil \log_2 9 \rceil = 4$ *preguntas.*

EJEMPLO RESUELTO

# Una pesada vale más que una pregunta

EJEMPLO

ENUNCIADO

Tenemos 9 monedas idénticas; cualquiera de ellas es falsa y más ligera. Disponemos de una balanza de platillos. ¿Cuánta información hay en este conjunto?

Problema de la farmacia

# Las diez letras

RETO 2 DIFICTAD 3

ENUNCIADO

Los dígitos del 0 al 9 están cifrados con las letras A, B, C, D, E, F, G, H, I, J en algún orden.

En una pregunta se puede averiguar la representación cifrada de la suma de varias letras distintas.

*Por ejemplo, si se pregunta 'A + B = ?' y A=9, B=1, C=0, la respuesta sería 'A + B = BC'.*

**¿Cómo se puede determinar en cinco preguntas qué letras corresponden a qué dígitos?**

# Las diez letras

RETO 2 DIFICULTAD 3

## ENUNCIADO

Los dígitos del 0 al 9 están cifrados con las letras A, …, J. En una pregunta se averigua la representación cifrada de la suma de varias letras distintas. Determinar la correspondencia en 5 preguntas.

## SOLUCIÓN

**Idea clave:** $0 + 1 + 2 + \ldots + 9 = 45$.

**Pregunta 1:** suma de las 10 letras → resultado cifrado de 45. Identificamos las letras de los dígitos 4 y 5.
**Pregunta 2:** suma de las 8 letras restantes → cifrado de 36 = 9 · 4. Identificamos 3 y 6.
**Pregunta 3:** suma de las 6 restantes → 27. Identificamos 2 y 7.
**Pregunta 4:** suma de las 4 restantes → 18. Identificamos 1 y 8.
**Pregunta 5:** suma de las 2 letras restantes → 9. Distinguimos 9 de 0.

*Cada pregunta 'poda' una pareja simétrica respecto a 4.5, separando las dos posibilidades complementarias.*

02

PARTE II

# Entropía de Shannon

# Entropía de Shannon

### Definición

Sea X una variable aleatoria discreta sobre $\{x_1, \ldots, x_n\}$ con probabilidades $p_i$. Su entropía (en bits) es:

$$H(X) = -\Sigma_i\, p_i \cdot \log_2 p_i$$

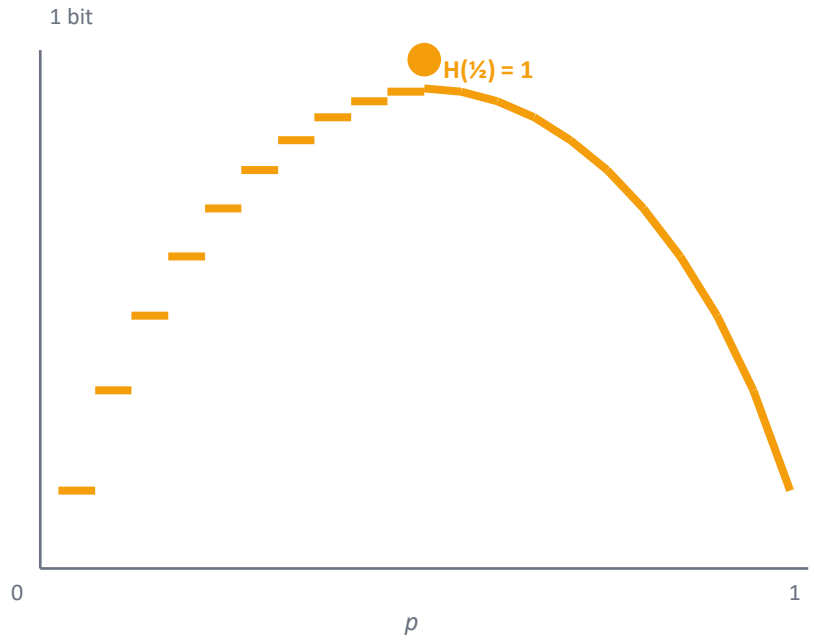
### Propiedades

- $H(X) \geq 0$ y $H(X) = 0$ si y sólo si algún $p_i = 1$ (sin incertidumbre).
- $H(X) \leq \log_2 n$, con igualdad si y sólo si X es uniforme.
- X, Y independientes $\Rightarrow$ $H(X, Y) = H(X) + H(Y)$.

### Ejemplo · moneda

Variable binaria con P(X=1) = p:

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p)$$



EJEMPLO RESUELTO

# La moneda trucada

EJEMPLO

ENUNCIADO

Calcula la entropía de una moneda trucada con **P(cara) = 0.25** y **P(cruz) = 0.75**. ¿Cuántos bits de información aporta cada lanzamiento, en media?

EJEMPLO RESUELTO

# La moneda trucada

EJEMPLO

ENUNCIADO

Calcula la entropía de una moneda trucada con **P(cara) = 0.25** y **P(cruz) = 0.75**. ¿Cuántos bits de información aporta cada lanzamiento, en media?

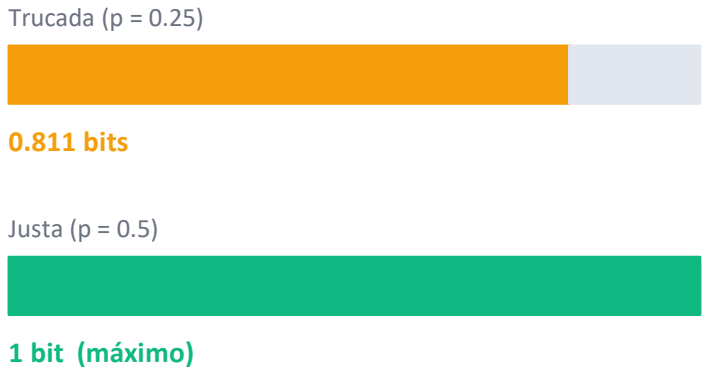
SOLUCIÓN

$$H = -p \log_2 p - (1-p) \log_2 (1-p)$$

$$= -0.25 \cdot \log_2(0.25) - 0.75 \cdot \log_2(0.75)$$
$$= -0.25 \cdot (-2) - 0.75 \cdot (-0.415)$$
$$= 0.5 + 0.311$$

$$H \approx 0.811 \text{ bits}$$

Comparación




*La moneda sesgada es más predecible: menos sorpresa, menos bits.*

# Codificar 4 letras con bits

Queremos transmitir mensajes con cuatro letras { a, b, c, d } usando solo 0 y 1.  La opción más obvia: dar a cada letra una pareja de bits.

### Codificación uniforme

| Letra | Código |
|---|---|
| a | 00 |
| b | 01 |
| c | 10 |
| d | 11 |

*2 bits por letra,  decodificación trivial.*

*Ahora, imagínate que en nuestro código las letras aparecen con distinta frecuencia, pongamos,*

- *A aparece con la probabilidad 0,4*
- *C aparece con la probabilidad 0,3*
- *B aparece con la probabilidad 0,2*
- *D aparece con la probabilidad 0,1*

*En nuestro código habrá demasiados 0 y pocos 1…*

# Codificar 4 letras con bits

Queremos transmitir mensajes con cuatro letras { a, b, c, d } usando solo 0 y 1. La opción más obvia: dar a cada letra una pareja de bits.

### Codificación uniforme

| Letra | Código |
|---|---|
| a | 00 |
| b | 01 |
| c | 10 |
| d | 11 |

*2 bits por letra, decodificación trivial.*

### Pregunta a la clase

?

- *A aparece con la probabilidad 0,4*
- *C aparece con la probabilidad 0,3*
- *B aparece con la probabilidad 0,2*
- *D aparece con la probabilidad 0,1*

**¿Se os ocurre otra manera de codificar { a, b, c, d } con 0 y 1?**

*¿Podríamos usar menos de 2 bits para alguna letra? ¿Qué problemas aparecerían?*

# Código interpretable de longitud variable

Reto 1

## El problema: códigos ambiguos

Si elegimos **a = 0, b = 1, c = 01, d = 101**, parece más corto. Pero al recibir **1 0 1**, ¿qué leemos? Hay tres lecturas posibles:

· todo de una vez: **d** · partido en tres: **b a b** · partido en dos: **b c**

## ¿Podéis proponer otra solución?

| Letra | P | Código |
|---|---|---|
| a | 0,4 | |
| c | 0,3 | |
| d | 0,2 | |
| b | 0,1 | |

*Ningún código es prefijo de otro: decodificación unívoca.*

# Código interpretable de longitud variable

**El problema: códigos ambiguos**

Si elegimos **a = 0, b = 1, c = 01, d = 101**, parece más corto. Pero al recibir **1 0 1**, ¿qué leemos? Hay tres lecturas posibles:

· todo de una vez: **d** · partido en tres: **b a b** · partido en dos: **b c**

**La solución: Shannon–Fano**

| Letra | P | Código |
|---|---|---|
| a | 0,4 | 0 |
| c | 0,3 | 10 |
| d | 0,2 | 110 |
| b | 0,1 | 111 |

*Ningún código es prefijo de otro: decodificación unívoca.*

**Longitud media**

$$L = 0.4\times1 + 0.3\times2 + 0.2\times3 + 0.1\times3$$

$$L = 1.9 \text{ bits} < 2 \text{ bits (uniforme)}$$

*Teorema (Shannon): ningún código puede bajar de la entropía $H \approx 1.85$ bits.*

# El algoritmo de Huffman

Construir el código prefijo de longitud media mínima para una distribución dada.  Reglas: fusionar los dos símbolos menos probables y repetir.

### El algoritmo

1. Cada símbolo es un nodo suelto con su probabilidad.
2. Fusiona los dos nodos de menor probabilidad en un padre cuya probabilidad es la suma.
3. Repite hasta tener un único nodo (la raíz).
4. Asigna 0 a cada arista izquierda y 1 a cada derecha. El código de un símbolo es la cadena de bits del camino raíz → hoja.

### Ejemplo: 4 letras

a:0.4  c:0.3  d:0.2  b:0.1

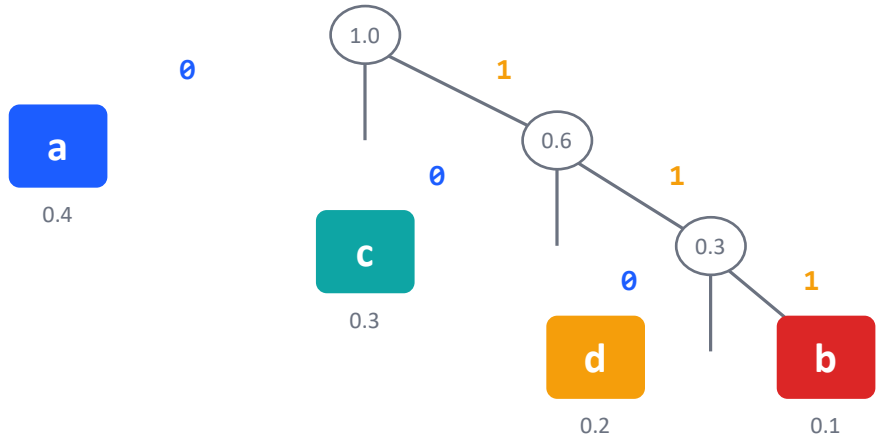
**Códigos:**

a → 0

c → 10

d → 110

b → 111




$L = 1.9$ bits $\approx H \approx 1.85$ bits

RETO PRÁCTICO

# Construye el código de Huffman

RETO 2

ENUNCIADO

Una fuente emite seis símbolos con las siguientes probabilidades:

| Símbolo | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| P | 0.33 | 0.04 | 0.20 | 0.11 | 0.27 | 0.05 |

**Construye el código de Huffman para esta fuente.**

**(a)** Dibuja el árbol de fusiones, paso a paso.
**(b)** Da la tabla final de códigos para A, B, C, D, E, F.
**(c)** Calcula la longitud media L y compárala con la entropía H ≈ 2.3 bits.

# Huffman para 6 letras

RETO 1

**ENUNCIADO:** P(A)=0.33, P(B)=0.04, P(C)=0.20, P(D)=0.11, P(E)=0.27, P(F)=0.05.

## Pasos de fusión

**Paso 1:** fusiono F+B = **BF (0.09)**
A:0.33 BF:0.09 C:0.20 D:0.11 E:0.27

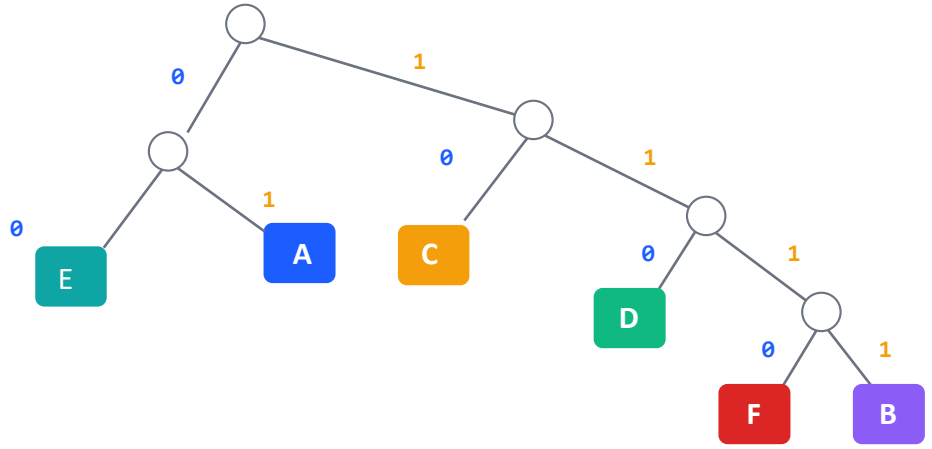
**Paso 2:** fusiono D+BF = **DBF (0.20)**
A:0.33 E:0.27 DBF:0.20 C:0.20

**Paso 3:** fusiono C+DBF = **CDBF (0.40)**
CDEF:0.40 A:0.33 E:0.27

**Paso 4:** fusiono A+E = **AE (0.60)**

**Paso 5:** fusiono AE+CDBF = **raíz (1.0)**

## Árbol final & códigos




**Códigos:** **A=01 B=1111 C=10 D=110 E=00 F=1110** · L = (0.33+0.27)×2 + 0.11×3 + (0.09+0.11)×4 ***= 2.33 bits*** *(H ≈ 2.3)*

# Dos urnas, dos incertidumbres

RETO 3

ENUNCIADO

Tenemos dos urnas con 20 bolas cada una.

**Urna 1:** 10 blancas, 5 negras y 5 rojas.
**Urna 2:** 8 blancas, 6 negras y 6 rojas.

**Se extrae una bola de cada urna. ¿Cuál de los dos resultados es más incierto?**

# Dos urnas, dos incertidumbres

RETO 3 DIFICULTAD 2

## ENUNCIADO

Urna 1: 10 blancas, 5 negras, 5 rojas (de 20). Urna 2: 8 blancas, 6 negras, 6 rojas (de 20). ¿Cuál extracción es más incierta?

## SOLUCIÓN

Calculamos la entropía de cada experimento.

**Urna 1 (½, ¼, ¼):** $H_1 = ½ \times 1 + ¼ \times 2 + ¼ \times 2 = 1.5$ ***bits.***

**Urna 2 (2/5, 3/10, 3/10):** $H_2 = (2/5) \times 1.32 + 2 \times (3/10) \times 1.74 \approx 1.57$ ***bits.***

**Conclusión:** la urna 2 es más incierta. Se acerca más a la equidistribución, que maximiza la entropía (H_max = $\log_2 3 \approx 1.585$ bits).

# Cadenas con sesgo

RETO 4 DIFICLTAD 3

ENUNCIADO

Tenemos cadenas de n bits sobre las que sabemos que los unos aparecen con probabilidad 1/3.

**De media, ¿cuántas preguntas SÍ/NO hacen falta para determinar la cadena de 60 bits?**

# Cadenas con sesgo

RETO 4 DIFICULTAD 3

## ENUNCIADO

Cadenas de n bits con P(1) = 1/3. ¿Cuántas preguntas SÍ/NO de media hacen falta para determinar la cadena?

## SOLUCIÓN

El número de cadenas típicas (con n/3 unos) es ***C(n, n/3).***

*Por Stirling:* $\log_2 C(n, n/3) \approx \log_2 \left[ \frac{\left(\frac{n}{e}\right)^n}{\left(\frac{2n}{3e}\right)^{\frac{2n}{3}} \left(\frac{n}{3e}\right)^{\frac{n}{3}}} \right]$

$= n \times \left[ \left(\frac{1}{3}\right) \log_2 3 + \left(\frac{2}{3}\right) \log_2 \left(\frac{3}{2}\right) \right] \approx 55$ para $n = 60$

Y ese resultado coincide con la entropía de Shannon de la fuente:

***H = −(1/3) log₂(1/3) − (2/3) log₂(2/3) ≈ 0.918 bits/símbolo.***

*Total: en media ≈ 0.918 × n preguntas, frente a las n del caso uniforme.*

EJERCICIO PRÁCTICO

# ¿Y si recodificamos los bits?

EJEMPLO

ENUNCIADO

Las cadenas tienen P(1) = 1/3, P(0) = 2/3. Su entropía es $H \approx 0.918$ bits/símbolo. ¿Podemos transmitir estas cadenas usando menos de 1 bit por símbolo?

EJERCICIO PRÁCTICO

# ¿Y si recodificamos los bits?

EJEMPLO

ENUNCIADO

Las cadenas tienen P(1) = 1/3, P(0) = 2/3. Su entropía es $H \approx 0.918$ bits/símbolo. ¿Podemos transmitir estas cadenas usando menos de 1 bit por símbolo?

## Idea: agrupar los bits en parejas

| Pareja | P | Código |
|---|---|---|
| 00 | 4/9 | 0 |
| 01 | 2/9 | 10 |
| 10 | 2/9 | 110 |
| 11 | 1/9 | 111 |

**Bits/pareja:** $(4/9)\cdot 1 + (2/9)\cdot 2 + (2/9)\cdot 3 + (1/9)\cdot 3 \approx 1.89$

*Bits/símbolo: $1.89 / 2 \approx 0.94$*

## Codificar y decodificar

Mensaje original (10 bits):

**00 00 01 10 00**

Codificado (8 bits):

**0 · 0 · 10 · 110 · 0**

→ *¡2 bits ahorrados sin perder información!*

*Agrupando de a más bits podemos acercarnos arbitrariamente a 0.918.*

**Ejemplo 2.** *Supongamos que necesitamos conocer un ordenamiento desconocido* $A$ *del conjunto de cinco elementos* $\{1, 2, 3, 4, 5\}$.*Supongamos que nos informaron que* $1 > 2$ *o* $3 > 4$. *¿Cuántos bits hay en esta información?*

**(a)** Dos personas realizan un truco de cartas. La primera toma cinco cartas de una baraja de 52 barajada por un espectador, las mira y a continuación las coloca en fila de izquierda a derecha: una de ellas boca abajo y las demás boca arriba. La segunda adivina la carta tapada. Demostrar que pueden ponerse de acuerdo de antemano para que la segunda siempre adivine la carta.

**(b)** El segundo truco difiere del primero en que la primera persona coloca cuatro cartas boca arriba (de izquierda a derecha) y *no coloca* la quinta. ¿Pueden ponerse de acuerdo para que la segunda siempre adivine la carta no colocada?

Aquí la segunda persona ve cuatro cartas en orden y debe adivinar la quinta, que no está sobre la mesa. El número de permutaciones de 4 cartas es $4! = 24 < 48$, insuficiente en general. La clave es el **principio del palomar**: entre 5 cartas y solo 4 palos, al menos dos cartas tienen el mismo palo.

Ambos numera previamente los valores del $1$ al $13$ ($\mathrm{A} = 1, 2 = 2, \ldots, \mathrm{K} = 13$), y las $3! = 6$ permutaciones de $\{a, b, c\}$ del $1$ al $6$.

Al recibir las cinco cartas, la primera persona:

a. Elige dos cartas $P$ y $Q$ del mismo palo. Coloca mentalmente las 13 cartas de ese palo en un *círculo* en orden horario: $\mathrm{A}, 2, 3, \ldots, \mathrm{K}, \mathrm{A}, \ldots$

b. Cuenta $p$ pasos en sentido horario de $P$ a $Q$, y $q = 13 - p$ pasos de $Q$ a $P$. Como $p + q = 13$, exactamente uno de los dos (llamémoslo $p$) satisface $p \leq 6$.

c. Coloca $P$ en primer lugar (boca arriba) y *oculta* $Q$ (no la pone sobre la mesa).

d. Etiqueta las otras tres cartas $a < b < c$ en orden ascendente de sus números de mazo. Coloca a continuación la *permutación número* $p$ de $(a, b, c)$.

La segunda persona ve cuatro cartas, lee la permutación de las tres últimas (número $p$), coloca mentalmente la primera carta en el círculo de su palo, cuenta $p$ pasos en sentido horario y obtiene $Q$.

**¿Por qué funciona?** Las $3! = 6$ permutaciones de $(a, b, c)$ codifican exactamente los valores $p \in \{1, 2, 3, 4, 5, 6\}$, y la restricción $p \leq 6$ siempre se cumple.

**Ejemplo.** Las mismas cinco cartas: ♠$5$, ♡$3$, ♢$7$, ♣$2$, ♣$9$.

Dos tréboles: ♣$2$ y ♣$9$.

La primera cuenta $p = 6$ pasos horarios de ♣$9$ a ♣$2$ (pasan ♣10, J, Q, K, A, 2). Las otras tres cartas son $a =$ ♠$5$ (n°\,5), $b =$ ♡$3$ (n°\,16), $c =$ ♢$7$ (n°\,33). La permutación número $6$ de $(a, b, c)$ en orden lexicográfico es $(c, b, a)$. Disposición final:

APLICACIÓN

# Compresión de archivos

Lo que hemos hecho con bits sesgados es exactamente la idea detrás de cualquier compresor moderno: exprimir la redundancia hasta acercarse a la entropía.

FRECUENCIA

## Símbolos frecuentes

En español la 'e' aparece muchísimo más que la 'x'. Códigos cortos para letras frecuentes.

*Huffman, Shannon-Fano*

REPETICIÓN

## Patrones repetidos

Si una secuencia ya apareció, basta con apuntar dónde estaba. No hace falta reescribirla.

*LZ77, LZ78 (zip, gzip)*

CONTEXTO

## Modelos de contexto

Tras 'qu' casi siempre viene una vocal. Ajustamos las probabilidades a lo que ya hemos visto.

*PPM, codificación aritmética*

**Cota inferior de Shannon:** *ningún algoritmo sin pérdida puede comprimir por debajo de la entropía H del mensaje.*

APLICACIÓN

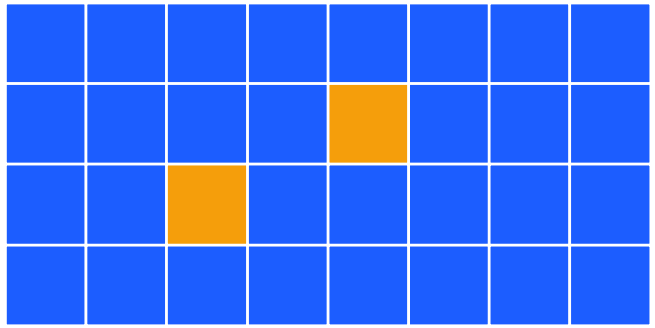
# Compresión de imágenes y vídeo

SIN PÉRDIDA

## PNG, GIF, FLAC, ZIP

Codifican exactamente los píxeles, byte a byte. Limitados por la entropía del archivo.

*Una imagen con grandes zonas de un color tiene baja entropía:*



→ *Se puede codificar: «32 píxeles azules excepto en las posiciones 5 y 18 (amarillas)»*

CON PÉRDIDA

## JPEG, MP3, H.264, MP4

Tiran a la basura información que el ojo o el oído no van a notar. Ya no hay cota de Shannon: el límite lo pone la calidad.

Qué se descarta:

- Detalles finos en zonas con mucho color (JPEG)
- Frecuencias altas inaudibles (MP3)
- Diferencias entre fotogramas casi iguales (vídeo)

*JPEG comprime una foto típica entre 10× y 20× con calidad casi imperceptible.*

PREGUNTA RÁPIDA

# ¿Cuánto pesa 1 minuto de audio?

Un minuto de música en estéreo, calidad CD.  ¿Cuántos bits de información tiene?

- Menos de 10Kbit
- Entre 10Kbit y 100Kbit
- Entre 100Kbit y 1Mbit
- Entre 1Mbit y 10Mbits
- Entre 10 Mbits y 100Mbits

PREGUNTA RÁPIDA

# ¿Cuánto pesa 1 minuto de audio?

Un minuto de música en estéreo, calidad CD. ¿Cuántos bits de información tiene?

**Lo que vamos a contar**

| | |
|---|---|
| Mediciones por segundo | $44\,100$ / s |
| Bits por medición | $16$ bits |
| Canales (estéreo) | $2$ |
| Duración | $60$ segundos |

**Cálculo**

$$\text{Total} = 44\,100 \times 16 \times 2 \times 60$$
$$= 84\,672\,000 \text{ bits}$$
$$\approx 10.6 \text{ millones de bytes}$$

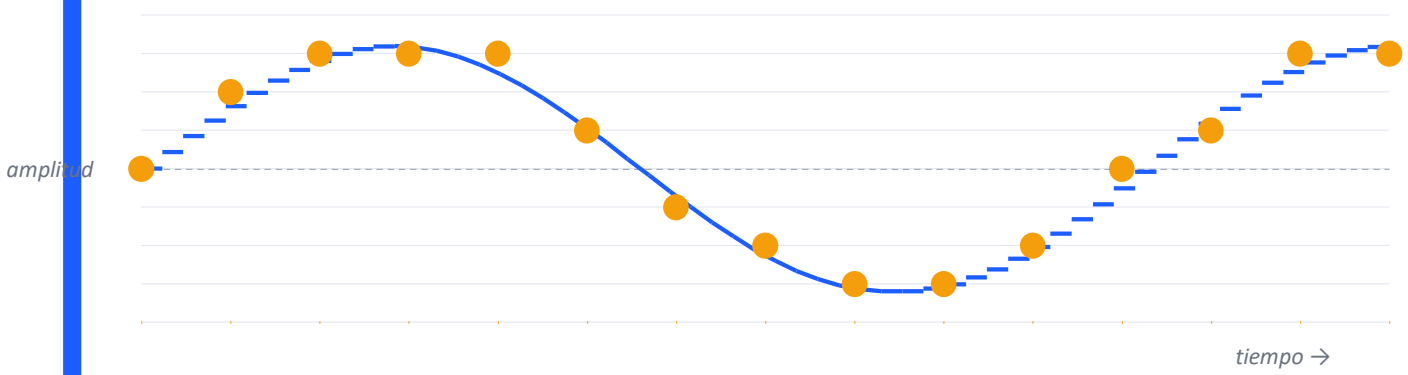
**≈ 10.6 MB**

EXPLICACIÓN

# Cómo se digitaliza un sonido

Un sonido es una onda continua. Para guardarla en bits hay que muestrearla muchas veces por segundo y, para cada muestra, anotar la altura de la onda con un número entero.

## Muestreo y profundidad de bit




• 44 100 muestras por segundo · • 16 bits = 65 536 niveles posibles

## El resultado

**WAV / CD**

***1 411 kbps***

*(estéreo, 16-bit, 44.1 kHz)*

| | |
|---|---|
| 1 minuto | ***10.6 MB*** |
| 1 hora | ***635 MB*** |
| 1 CD (74 min) | ***≈ 780 MB*** |

COMPRESIÓN

# MP3: tirar lo que el oído no oye

El MP3 no comprime con teoría de Shannon, sino con un modelo del oído humano. Descarta sonidos que físicamente no llegamos a percibir.

ENMASCARAMIENTO

### Sonido fuerte tapa al débil

Un tono fuerte cercano en frecuencia oculta a uno más débil: no hace falta guardarlo.

FRECUENCIAS LÍMITE

### El oído no llega tan alto

Por encima de ~16 kHz casi nadie oye nada. Esos componentes se descartan o se cuantizan en grueso.

ENMASCARAMIENTO TEMPORAL

### Tras un golpe fuerte, sordera breve

Después de un sonido intenso, el oído queda «atontado» unos milisegundos. Esos instantes se simplifican.

1 minuto de música: el ahorro real

# JPEG: tirar lo que el ojo no ve

### Foto del móvil, sin comprimir

12 megapíxeles = 4000 × 3000 píxeles


→ *3 bytes por píxel (24 bits color)*

$$12\,000\,000 \times 3 = 36\,000\,000 \text{ bytes}$$

**≈ 36 MB**

*Cada foto, sin comprimir, ocupa lo que un álbum entero de música.*

### Después de pasarla por JPEG

**Tres trucos clave:**

- El ojo distingue mejor el brillo que el color

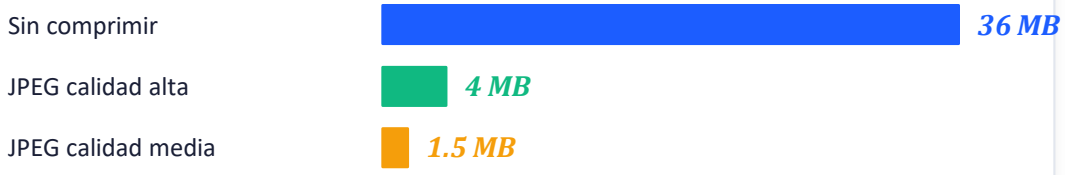
→ *guardar el color con resolución reducida*

- Bloques de 8×8 píxeles → análisis de frecuencias (DCT)

→ *los detalles muy finos se cuantizan en grueso*

- Compresión sin pérdida final (Huffman)

→ *aprovecha la entropía del resultado*



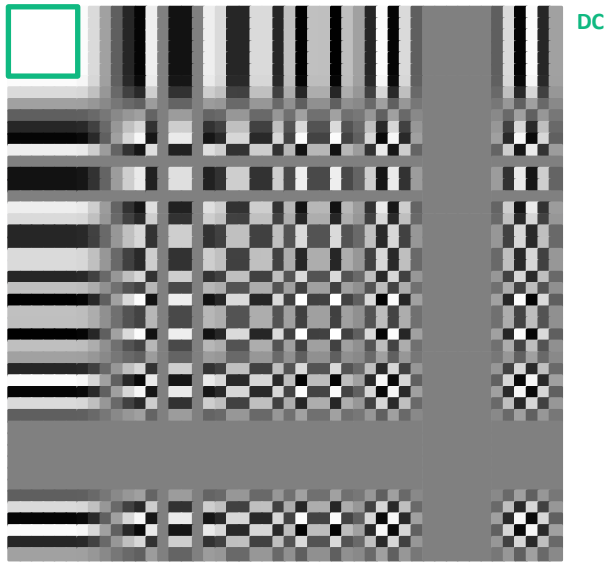

*≈ 10× a 25× más pequeño, con calidad casi imperceptible.*

# DCT: cambiar de representación

Una imagen tiene mucha redundancia entre píxeles vecinos. La DCT transforma cada bloque de 8×8 píxeles en una suma de 64 patrones-base de distintas frecuencias.

## Los 64 patrones-base

*u → frecuencia horizontal*




## ¿Qué representa cada patrón?

**Esquina superior izquierda (DC):** valor uniforme — el «brillo medio» del bloque.

**Filas/columnas iniciales:** patrones suaves — gradientes y transiciones lentas.

**Esquina inferior derecha (AC altas):** rayas finas — detalles minúsculos píxel a píxel.

*Cada bloque 8×8 de la imagen* ***= suma ponderada de los 64 patrones.***

*La DCT calcula los 64 pesos. Es reversible: no comprime nada por sí sola.*

# DCT: cambiar de representación

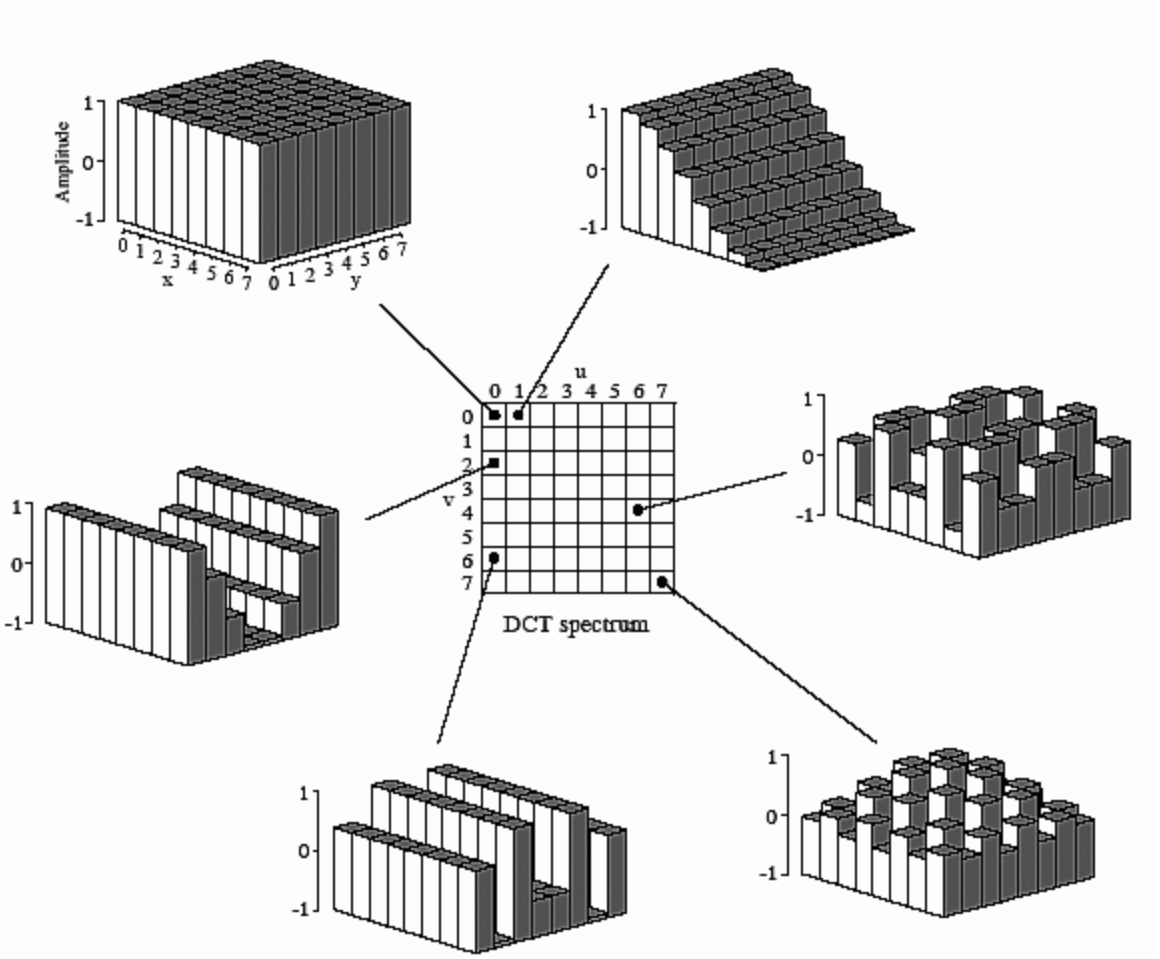



FIGURE 27-10
The DCT basis functions. The DCT spectrum consists of an 8×8 array, with each element in the array being an amplitude of one of the 64 basis functions. Six of these basis functions are shown here, referenced to where the corresponding amplitude resides.

## ¿Qué representa cada patrón?

**Esquina superior izquierda (DC):** valor uniforme — el «brillo medio» del bloque.

**Filas/columnas iniciales:** patrones suaves — gradientes y transiciones lentas.

**Esquina inferior derecha (AC altas):** rayas finas — detalles minúsculos píxel a píxel.

*Cada bloque 8×8 de la imagen* ***= suma ponderada de los 64 patrones.***

*La DCT calcula los 64 pesos. Es reversible: no comprime nada por sí sola.*

JPEG EN ACCIÓN

# DCT + cuantización + Huffman

JPEG no es una técnica única, sino una cadena. Cada paso prepara el siguiente.

PASO 1

## DCT

*Reordenar*

Transforma el bloque 8×8 de píxeles en 64 coeficientes de frecuencia. La energía se concentra arriba-izquierda.

*sin pérdida*

PASO 2

## Cuantización

*Tirar lo invisible*

Divide cada coeficiente por una tabla. Los detalles finos quedan en cero. Aquí ocurre la pérdida.

CON PÉRDIDA

PASO 3

## Zig-zag + RLE

*Agrupar ceros*

Recorrido en zigzag para juntar los ceros al final. Luego se codifican como «N ceros + valor».

*sin pérdida*

PASO 4

## Huffman

*Comprimir lo que queda*

Código prefijo óptimo: los pares más frecuentes obtienen códigos cortos.

*sin pérdida*

La conclusión

Solo Huffman (sin DCT) sobre los píxeles RGB: **≈ 30% de ahorro.**
Solo DCT (sin cuantización ni Huffman): **no comprime — es solo un cambio de representación.**
DCT + cuantización + Huffman juntos: **10:1 a 25:1, con calidad casi imperceptible.**

03

PARTE III

# Distancia de Hamming y corrección de errores

# Quiero transmitir dos letras, A y B

**Problema**

Pero en la transmisión un bit puede llegar con error

**¿Me basta un bit?**

Si A=0, B=1 y me llega 1 puede ser
- A (ha llegado con error)
- B (ha llegado sin errores)

Necesito más bits para codificarlas

**¿Me bastan dos bits?**

Si A=00, B=11 y me llega 10 puede ser
- A (ha llegado con error en el 1 bit)
- B (ha llegado  con error en el 2 bit)

Necesito más bits para codificarlas

# Distancia de Hamming

### Definición

Para x, y ∈ {0,1}ⁿ:

$$d(x,y) = \#\{\, i : x_i \neq y_i \,\}$$

### Es una métrica:

- $d(x,y) \geq 0,\ d(x,y) = 0 \Leftrightarrow x = y.$
- $d(x,y) = d(y,x).$
- $d(x,z) \leq d(x,y) + d(y,z).$

### Ejemplo:

x = (1, 0, 1, 1, 0)  y = (1, 1, 1, 0, 0)  → d(x, y) = 2.

### Bits con error

Transmitido

1 1 0 0 1 0 0 0 1 0 1 0 0 0 1 0

Recibido

0 1 0 0 1 0 0 0 1 0 1 0 1 0 1 1

**3 posiciones difieren → d(tx, rx) = 3**

*La distancia de Hamming cuantifica cuántos errores se han producido.*

# Códigos correctores de errores

Un código $C \subset A^n$ (palabras de longitud n sobre el alfabeto A) es un conjunto de palabras válidas. Su distancia mínima d(C) determina cuántos errores podemos detectar y corregir.

**Detecta**

$$d - 1$$

errores

**Corrige**

$$\lfloor (d - 1) / 2 \rfloor$$

errores

**Bola de Hamming**

$$B(x, e)$$

esferas disjuntas si $d \geq 2e + 1$

**Código de repetición C = {000, 111}**

A=000, B=111

d(000, 111) = 3 → detecta 2 errores y corrige 1.

*Si recibimos 001 → más cercano a 000 → corregimos.*

**Código con paridad**

Las palabras suman 0 (mod 2). d(C) = 2.
*Detecta 1 error pero no lo corrige unívocamente.*

# Ahora queremos codificar 4 letras, no 2. 3 bits no bastarán. ¿Y cuatro bits?

RETO 5 DIFICULTAD 2

ENUNCIADO

**Queremos codificar 4 letras, {A,B,C,D} con 4 bits con una distancia de Hamming entre cualesquiera dos de ellas mayor o igual a 3. ¿Podemos conseguirlo?**

*NO.*

*La segunda debe ser distinta de la primera en 3 bits, pongamos, los 3 primeros. Entonces la tercera palabra coincidira con la primera o la segunda al menos en 2 bits (Palomar)*

# Ahora queremos codificar 4 letras, no 2. 3 bits no bastarán. 4 tampoco

RETO 5 DIFICULTAD 2

ENUNCIADO

**Construye 4 cadenas de 5 bits con una distancia de Hamming entre cualesquiera dos de ellas mayor o igual a 3.**

*Equivale a encontrar un código binario $C \subset \{0,1\}^5$ con $|C| = 4$ y $d(C) \geq 3$.*

*Este código corregiría 1 error y detectaría hasta 2.*

# Cuatro cadenas separadas

RETO 5

ENUNCIADO

Construye 4 cadenas de 5 bits con distancia de Hamming entre cualesquiera dos ≥ 3.

SOLUCIÓN

Una elección posible:

**C = { 00000, 00111, 11001, 11110 }**

*Comprobaciones (todas las parejas):*

```
d(00000, 00111) = 3        d(00000, 11001) = 3        d(00000, 11110) = 4
d(00111, 11001) = 4        d(00111, 11110) = 3        d(11001, 11110) = 3
```

**Distancia mínima d(C) = 3 ✓**

# Cinco cadenas separadas

RETO 5’

ENUNCIADO

**Construye 5 cadenas de 5 bits con una distancia de Hamming entre cualesquiera dos de ellas mayor o igual a 3.**

¿Es posible?

¡No!

# Tres sabios

RETO 5’’

ENUNCIADO

A tres sabios se les vendan los ojos, se les colocan sombreros negros o blancos y se les quita la venda.

Cada uno ve los sombreros de todos los demás, pero no el suyo.

Tras ponerse de acuerdo previamente sobre una estrategia, todos deben actuar simultáneamente: o bien dicen el color de su propio sombrero, o bien guardan silencio.

Los sabios ganan si al menos un participante acierta el color de su sombrero y nadie se equivoca. Construye una estrategia cuya probabilidad de éxito sea 3/4, suponiendo que los colores son independientes y equiprobables.

# Tres sabios

RETO 5’’

ENUNCIADO

A tres sabios se les vendan los ojos, se les colocan sombreros negros o blancos y se les quita la venda.

Cada uno ve los sombreros de todos los demás, pero no el suyo.

Tras ponerse de acuerdo previamente sobre una estrategia, todos deben actuar simultáneamente: o bien dicen el color de su propio sombrero, o bien guardan silencio.

SOLUCIÓN

Primero definimos dos palabras del código A=000, B=111.
La estrategia de cada jugador es la siguiente. Observa los bits de los demás y considera las dos configuraciones completas posibles: una suponiendo que su propio bit es 0 y otra suponiendo que es 1. Si exactamente una de esas dos configuraciones pertenece al código, el jugador anuncia el color correspondiente a la otra configuración, es decir, aquella para la que la configuración completa no es una palabra de código. Si ninguna de las dos configuraciones pertenece al código, permanece en silencio.

# El oráculo mentiroso (1 mentira)

RETO 6 DIFICULTAD 4

ENUNCIADO

Un oráculo ha pensado un número del 1 al 8. Puedes hacerle preguntas SÍ/NO, pero el oráculo puede mentir hasta una vez (eligiendo cuándo después de oír la pregunta).

Tienes que formular de antemano una serie de preguntas que te permitan averiguar el número.

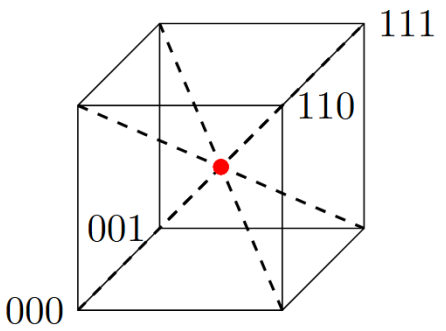
**¿Cuántas preguntas necesitas? Construye una estrategia.**

# El oráculo mentiroso (1 mentira)

RETO 6

## ENUNCIADO

Oráculo piensa x ∈ {1,…,8} y miente como mucho 1 vez. Diseñar k preguntas SÍ/NO no adaptativas que determinen x.




## SOLUCIÓN

**Cota inferior:** Cada número genera k+1 secuencias compatibles (0 mentiras + k posiciones de mentira). Por tanto:

$2^k \geq 8\times (k+1)$.

`k = 5: 32 < 48` `k = 6: 64 > 56.` ✓

**Construcción:** asignar a cada número del 1 al 8 una palabra de 6 bits, con distancia mínima 3 entre cualesquiera dos. La pregunta i es: ¿el número está marcado con 1 en la posición i?

*Como mucho una respuesta es errónea ⇒ palabra recibida a distancia ≤ 1 de la verdadera ⇒ decodificación unívoca (las bolas de radio 1 son disjuntas, por d ≥ 3).*

**Bastan 6 preguntas, el doble de lo que necesitamos si no hay mentiras. ¿Serás capaz de construir estas 8 cadenas? Piensa en un hipercubo en 6D.**

1. 000000 2. 111000 3. 001101 4. 110101 5. 010011 6. 101011 7. 011110 8. 100110

04

PARTE IV

# Pesadas y búsqueda óptima

MÁS DE DOS RESULTADOS

# Cuando una pregunta da $\log_2 3$ bits

Una balanza de platillos compara dos grupos disjuntos. Resultados posibles:

| | |
|---|---|
| $<$ | izquierda más ligera |
| $=$ | equilibrio |
| $>$ | derecha más ligera |

**Información que aporta cada pesada**

En el peor caso, cada pesada aporta como mucho $\log_2 3 \approx 1.585$ bits.

Por tanto, para distinguir entre N resultados se necesitan al menos $\lceil \log_3 N \rceil$ pesadas.

*Una buena estrategia divide el conjunto de hipótesis posibles en tres partes lo más equilibradas posible (en información: ~$\log_2 3$ bits por pesada, sin importar el resultado).*

# Cota inferior y diseño de pesadas

## 1. Cota inferior

Se cuentan los resultados posibles N (qué moneda es falsa, si es más ligera o más pesada, etc.).

Necesitan $k \geq \lceil \log_3 N \rceil$ pesadas.

*Ej: 12 monedas con falsa más ligera o pesada → N = 24, y como $3^3 = 27 \geq 24$, basta con 3 pesadas.*

## 2. Construcción de la estrategia

Cada pesada debe partir el conjunto de resultados aún posibles en tres partes con cardinales aproximadamente iguales.

Si el cardinal del conjunto sospechoso después de una pesada supera 3^(k−1), el algoritmo restante es imposible.

*Esta tensión entre los tres resultados de la balanza permite descartar variantes y guiar el diseño paso a paso.*

# 81 monedas, una más ligera

RETO 7

ENUNCIADO

Dadas 81 monedas idénticas en apariencia, exactamente una es falsa y más ligera. Las verdaderas pesan todas lo mismo.

**Disponemos de una balanza de platillos. ¿Cuántas pesadas son necesarias y suficientes para encontrar la moneda falsa?**

# 81 monedas, una más ligera

RETO 7 DIFICULTAD 2

## ENUNCIADO

81 monedas idénticas, una es falsa y más ligera. ¿Cuántas pesadas hacen falta?

## SOLUCIÓN

Respuesta: **4 pesadas.**

**Cota inferior.** Hay $N = 81 = 3^4$ resultados; cada pesada da como mucho $\log_2 3$ bits, y necesitamos $\log_2 81 = 4 \times \log_2 3$ bits.

**Cota superior (estrategia).** Dividir en 3 grupos de 27 y comparar dos de ellos.

- Si **equilibrio**: la falsa está en el tercer grupo.
- Si **desequilibrio**: la falsa está en el grupo más ligero.

*Repetir recursivamente. Total: 4 pesadas.* ✓

# 12 monedas, peso desconocido

RETO 8 DIFICULTAD 4

ENUNCIADO

Dadas 12 monedas, una es falsa pero NO sabemos si es más ligera o más pesada que las verdaderas (las verdaderas pesan todas lo mismo).

**Demuestra que en 3 pesadas en una balanza de platillos se puede encontrar la moneda falsa y averiguar si es más ligera o más pesada.**

# 12 monedas, peso desconocido

RETO 8 DIFICULTAD 4

## ENUNCIADO

12 monedas; la falsa puede ser más ligera o más pesada. En 3 pesadas, hallarla y decir su tipo.

## SOLUCIÓN

*Resultados posibles $N = 12 \times 2 = 24$ ; $3^3 = 27 \geq 24$, así que cabe en 3 pesadas (apenas).*

**Pesada 1:** comparar dos grupos de 4 monedas. (Si fueran $k \neq 4$ monedas, alguno de los conjuntos resultantes superaría $3^2 = 9$.)

**Pesada 2 — caso equilibrio:** la falsa está entre las 4 no pesadas. Comparamos 3 de ellas con 3 verdaderas.

- Equilibrio: la falsa es la cuarta; basta una pesada con una verdadera para saber el tipo.
- Desequilibrio: la falsa está entre esas 3, y sabemos su tipo; comparamos dos de ellas.

**Pesada 2 — caso desequilibrio:** tenemos 4 ‘pesadas sospechosas’ y 4 ‘ligeras sospechosas’. Reorganizamos: en cada platillo, 2 pesadas + 1 ligera.

*Pesada 3 distingue los casos restantes (a lo más, 3 candidatos por resultado). ✓*

05

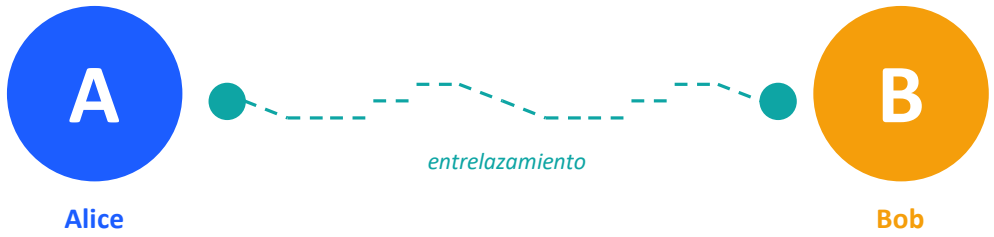
PARTE V

# Información y física

# ¿Puede la información ir más rápido que la luz?

Dos partículas entrelazadas comparten un estado cuántico. Al medir una, la otra «sabe» instantáneamente el resultado, por lejos que estén. ¿Es eso transmitir información a velocidad infinita?

## La situación




Alice y Bob comparten dos partículas entrelazadas. Si Alice mide la suya y obtiene «↑», la de Bob será «↓» con certeza, aunque esté a años-luz de distancia.

## La respuesta: no

**Teorema de no-comunicación.**

El resultado de Alice es **aleatorio** (50% «↑», 50% «↓»). No puede ***elegir*** qué obtiene, así que no puede codificar un mensaje en su medida.

Bob, mirando solo sus medidas, ve también **ruido aleatorio**. Para descubrir la correlación necesitan comparar resultados...

***...y eso requiere un canal clásico, limitado por c.***

# La misma fórmula en dos sitios

La entropía termodinámica de Boltzmann y la entropía de Shannon no son analogías: son la misma cosa.

BOLTZMANN (1872)

## Termodinámica

$$S = k_B \ln \Omega$$

$\Omega$ = número de microestados compatibles con un macroestado.

*Cuántas formas distintas hay de organizar las moléculas de un gas dejando intacta su temperatura, presión y volumen.*

SHANNON (1948)

## Información

$$H = -\Sigma\, p_i \log_2 p_i$$

$H$ = bits necesarios, en media, para describir un mensaje aleatorio.

*Cuántas preguntas SÍ/NO hacen falta, en media, para identificar el resultado de una variable aleatoria.*

LA EQUIVALENCIA

## Misma fórmula

$$S = k_B \ln 2 \cdot H$$

*Si todos los microestados son equiprobables ($p_i = 1/\Omega$):*

$$H = \log_2 \Omega$$
$$S = k_B \ln \Omega = k_B \ln 2 \times \log_2 \Omega$$

***Boltzmann mide la entropía en J/K, Shannon en bits. Solo cambian las unidades.***

# Borrar un bit cuesta energía

Si información y entropía son lo mismo, manipular bits debe tener un coste físico real. Y lo tiene.

PRINCIPIO DE LANDAUER (1961)

## Borrar un bit de información

*requiere disipar al menos*

$$E \geq k_B \, T \, \ln 2$$

*A temperatura ambiente (T = 300 K):*

$$E \approx 2.9 \times 10^{-21} \text{ julios por bit}$$

Diminuto a nivel individual, pero los servidores actuales ya disipan miles de millones de veces este límite por cada operación.

LA ENTROPÍA DEL UNIVERSO

## Información cósmica

**La segunda ley de la termodinámica:**
*la entropía del universo solo puede crecer.*

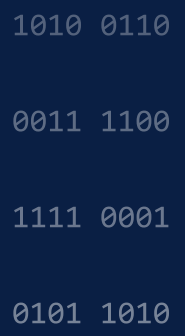
**Traducción informacional:** el número de bits necesarios para describir el universo crece sin parar.

**Agujeros negros:** Bekenstein y Hawking demostraron que su entropía es proporcional al área del horizonte. *Cada bit ocupa un trocito de superficie de tamaño ≈ longitud de Planck al cuadrado.*

**Principio holográfico:** *toda la información de una región 3D del universo cabe en su frontera 2D.*

RECAPITULANDO

# Información




## 01

### Cuantificar

$i = -\log_2 p$ mide en bits cuánto reduce un mensaje las posibilidades.

## 02

### Codificar

La entropía H es el límite teórico para la compresión sin pérdida.

## 03

### Proteger

La distancia de Hamming nos permite detectar y corregir errores en la transmisión.

*¿Preguntas?*