Tecnología Digital IV: Redes de Computadoras

Clase 27: Introducción a la Teoría de la Información - Parte 3

Lucio Santi & Emmanuel Iarussi

Licenciatura en Tecnología Digital
Universidad Torcuato Di Tella

26 de junio de 2025

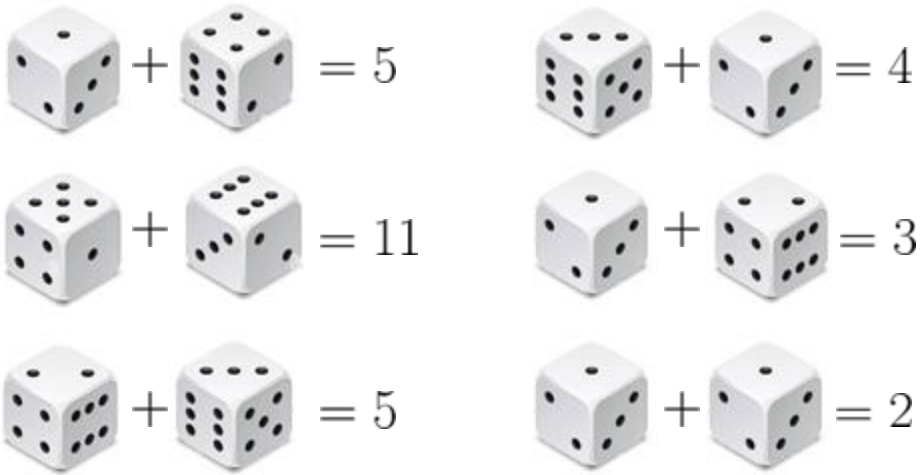
Compresión de datos

Compresión de datos

- Consideremos el caso en el que la variable no tiene una distribución uniforme. Por ejemplo, transmitir la suma de las caras de dos dados:

Compresión de datos

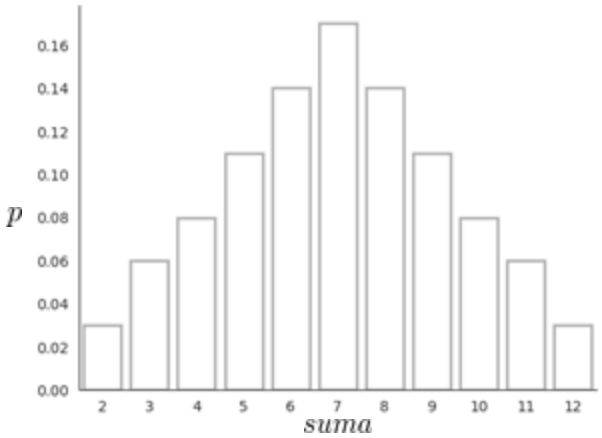
- Consideremos el caso en el que la variable no tiene una distribución uniforme. Por ejemplo, transmitir la suma de las caras de dos dados:



Compresión de datos

- Consideremos el caso en el que la variable no tiene una distribución uniforme. Por ejemplo, transmitir la suma de las caras de dos dados:

Símbolo	Suma	Dados	Frec.	p	h
s_1	2	1:1	1	0.03	5.17
s_2	3	1:2, 2:1	2	0.06	4.17
s_3	4	1:3, 3:1, 2:2	3	0.08	3.59
s_4	5	2:3, 3:2, 1:4, 4:1	4	0.11	3.17
s_5	6	2:4, 4:2, 1:5, 5:1, 3:3	5	0.14	2.85
s_6	7	3:4, 4:3, 2:5, 5:2, 1:6, 6:1	6	0.17	2.59
s_7	8	3:5, 5:3, 2:6, 6:2, 4:4	5	0.14	2.85
s_8	9	3:6, 6:3, 4:5, 5:4	4	0.11	3.17
s_9	10	4:6, 6:4, 5:5	3	0.08	3.59
s_{10}	11	5:6, 6:5	2	0.06	4.17
s_{11}	12	6:6	1	0.03	5.17



Compresión de datos

- Consideremos el caso en el que la variable no tiene una distribución uniforme. Por ejemplo, transmitir la suma de las caras de dos dados:

Símbolo	Suma	Dados	Frec.	p	h
s_1	2	1:1	1	0.03	5.17
s_2	3	1:2, 2:1	2	0.06	4.17
s_3	4	1:3, 3:1, 2:2	3	0.08	3.59
s_4	5	2:3, 3:2, 1:4, 4:1	4	0.11	3.17
s_5	6	2:4, 4:2, 1:5, 5:1, 3:3	5	0.14	2.85
s_6	7	3:4, 4:3, 2:5, 5:2, 1:6, 6:1	6	0.17	2.59
s_7	8	3:5, 5:3, 2:6, 6:2, 4:4	5	0.14	2.85
s_8	9	3:6, 6:3, 4:5, 5:4	4	0.11	3.17
s_9	10	4:6, 6:4, 5:5	3	0.08	3.59
s_{10}	11	5:6, 6:5	2	0.06	4.17
s_{11}	12	6:6	1	0.03	5.17

- Entropía:

$$H(X) = \sum_{i=1}^{m=11} p(s_i) \log_2 \frac{1}{p(s_i)}$$
$$= 3.27 \text{ bits/símbolo}$$

Compresión de datos

- Consideremos el caso en el que la variable no tiene una distribución uniforme. Por ejemplo, transmitir la suma de las caras de dos dados:

Símbolo	Suma	Dados	Frec.	p	h
s_1	2	1:1	1	0.03	5.17
s_2	3	1:2, 2:1	2	0.06	4.17
s_3	4	1:3, 3:1, 2:2	3	0.08	3.59
s_4	5	2:3, 3:2, 1:4, 4:1	4	0.11	3.17
s_5	6	2:4, 4:2, 1:5, 5:1, 3:3	5	0.14	2.85
s_6	7	3:4, 4:3, 2:5, 5:2, 1:6, 6:1	6	0.17	2.59
s_7	8	3:5, 5:3, 2:6, 6:2, 4:4	5	0.14	2.85
s_8	9	3:6, 6:3, 4:5, 5:4	4	0.11	3.17
s_9	10	4:6, 6:4, 5:5	3	0.08	3.59
s_{10}	11	5:6, 6:5	2	0.06	4.17
s_{11}	12	6:6	1	0.03	5.17

- Entropía:

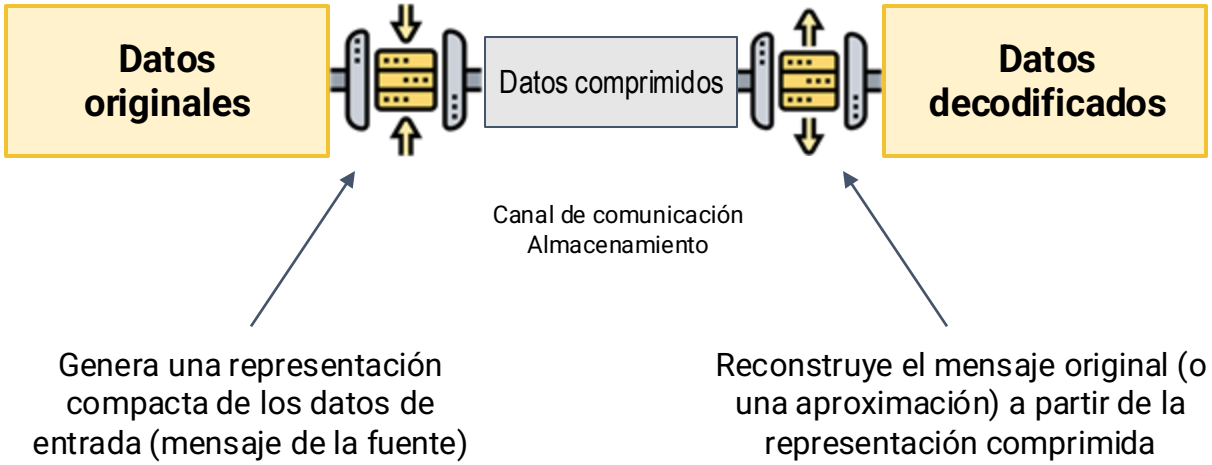
$$H(X) = \sum_{i=1}^{m=11} p(s_i) \log_2 \frac{1}{p(s_i)}$$
$$= 3.27 \text{ bits/símbolo}$$

- Eficiencia:

$$\frac{H(X)}{L(S)} = \frac{3.27}{4}$$
$$= 0.818 \text{ bits/dígito binario}$$

Compresión de datos

- Modelo general:



Métodos de compresión

- Sin pérdida o reversibles:
  - Mantienen la integridad de la información: los datos descomprimidos coinciden con los datos originales
  - Se aplica a texto, bases de datos, código fuente, imágenes, etc.
  - Tasas de compresión moderadas: 2:1 (texto), 4:1 (imágenes)
  - **La longitud media del código está acotada por la entropía**

$$L(S) \geq H(X)$$

Métodos de compresión

- Con pérdida o irreversibles:
  - No mantienen la integridad de la información: los datos descomprimidos son una aproximación de los datos originales
  - Se aplican a imágenes, video, sonido, etc.
  - Tasas de compresión altas: 30:1 (imágenes), 200:1 (video)
  - **No tienen como límite la entropía (pérdida de información)**

$$L(S) < H(X)$$

Métodos de compresión sin pérdida

Métodos de compresión sin pérdida

Métodos de compresión sin pérdida

Métodos de compresión sin pérdida

- Estadísticos
  (codifican un símbolo por vez)
- Con diccionario
  (codifican secuencias de símbolos)

Métodos de compresión sin pérdida

Métodos de compresión sin pérdida

- Estadísticos
  (codifican un símbolo por vez)
  - Estáticos
    (la distribución de probabilidades se asume universal)
  - Semiestáticos
    (la distribución de probabilidades se precomputa a partir de los datos)
  - Adaptativos/Dinámicos
    (la distribución de probabilidades se va actualizando al leer cada símbolo)
- Con diccionario
  (codifican secuencias de símbolos)
  - Estáticos
    (diccionario universal)
  - Semiestáticos
    (el diccionario se precomputa a partir de los datos)
  - Adaptativos/Dinámicos
    (el diccionario se va actualizando al leer cada símbolo)

Métodos de compresión sin pérdida

Tipo	¿Qué hacen?	Ejemplo
Estáticos	Suponen una distribución estándar.	Huffman con tabla fija (como en JPG clásico)
Semiestáticos	Calculan las probabilidades con los datos antes de codificar.	Huffman adaptado a un texto específico
Adaptativos/Dinámicos	Actualizan las probabilidades mientras leen el texto.	Arithmetic coding adaptativo

Métodos de compresión sin pérdida

- Métodos estáticos:

  Ventajas:

  - Requieren una única pasada por los datos para codificar
  - No es necesario transmitir/almacenar la distribución de probabilidades al decodificador.

Métodos de compresión sin pérdida

- Métodos estáticos:

  Ventajas:

  - Requieren una única pasada por los datos para codificar
  - No es necesario transmitir/almacenar la distribución de probabilidades al decodificador.

  Desventajas:

  - La distribución de probabilidades es fija y puede diferir de los datos a codificar

Métodos de compresión sin pérdida

- Métodos semiestáticos:

  Ventajas:

  - La distribución de probabilidades se ajusta a los datos a comprimir

Métodos de compresión sin pérdida

- Métodos semiestáticos:

  Ventajas:

  - La distribución de probabilidades se ajusta a los datos a comprimir

  Desventajas:

  - Requiere dos pasadas por los datos: construcción del código y codificación del mensaje
  - Se debe transmitir/almacenar la distribución de probabilidades al decodificador

Métodos de compresión sin pérdida

- Métodos adaptativos/dinámicos:

  Ventajas:

  - Requieren una única pasada por los datos para codificar
  - La distribución de probabilidades se va ajustando al mensaje a medida que se codifica
  - No se debe transmitir/almacenar la distribución de probabilidades al decodificador

Métodos de compresión sin pérdida

- Métodos adaptativos/dinámicos:

  Ventajas:

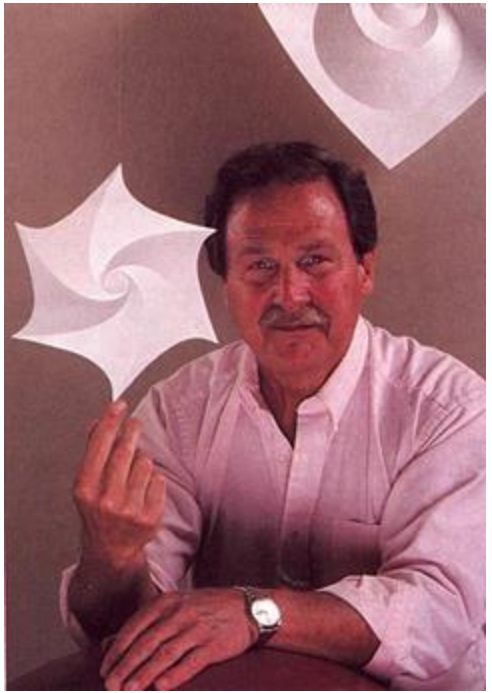
  - Requieren una única pasada por los datos para codificar
  - La distribución de probabilidades se va ajustando al mensaje a medida que se codifica
  - No se debe transmitir/almacenar la distribución de probabilidades al decodificador

  Desventajas:

  - Algoritmos complejos

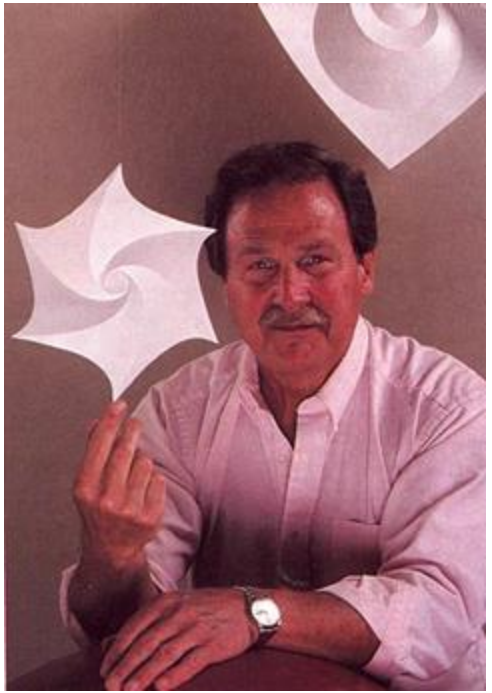
Código de Huffman (1952)

- **Idea! Asignar códigos más cortos a los símbolos más frecuentes (y viceversa)**
- Requiere conocer las probabilidades de ocurrencia
- Asume eventos independientes: los resultados de cada experimento aleatorio no están conectados entre sí de ninguna manera



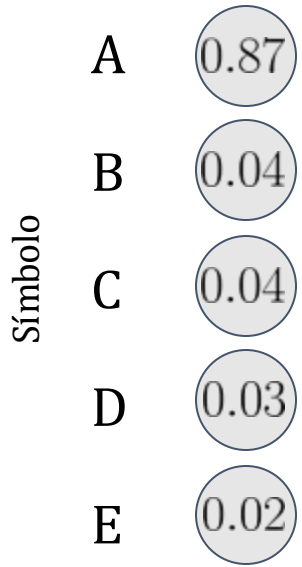
Código de Huffman (1952)

- Método de compresión sin pérdida
- Dos versiones: semiestática y adaptativa/dinámica
- Construye árboles binarios a partir de los símbolos de la fuente
- Forma parte de la especificación de estándares y formatos de compresión muy conocidos como JPEG, MP3, MPEG y GZIP



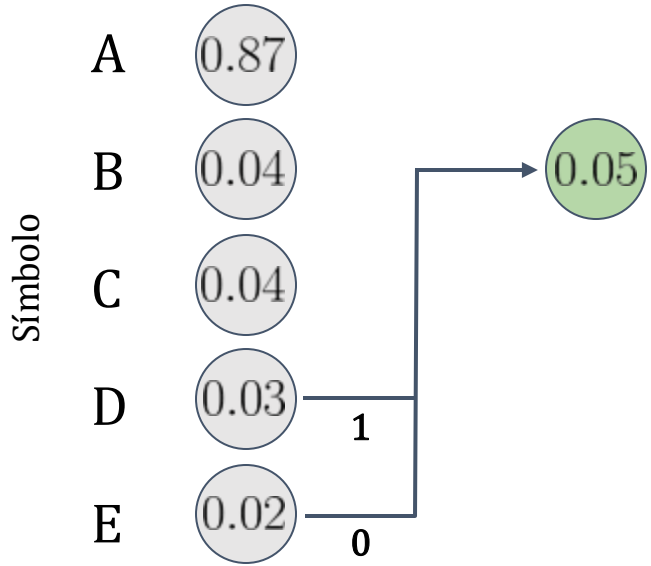
Código de Huffman: Ejemplo

- Ejemplo de construcción del código:



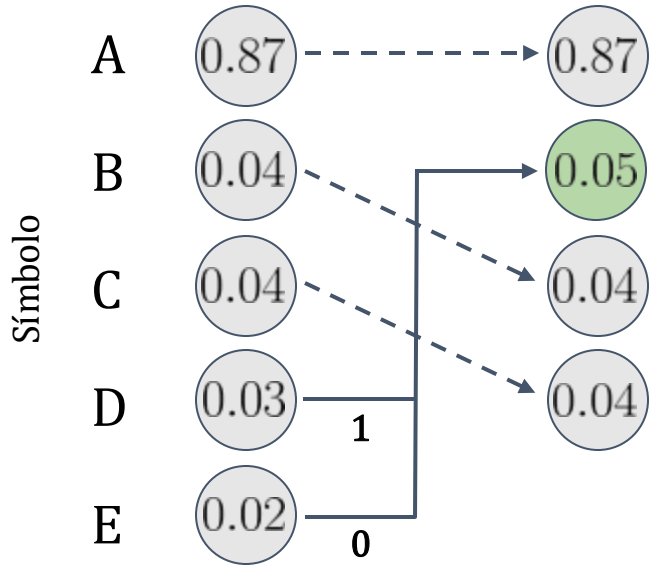
Código de Huffman: Ejemplo

- Ejemplo de construcción del código:



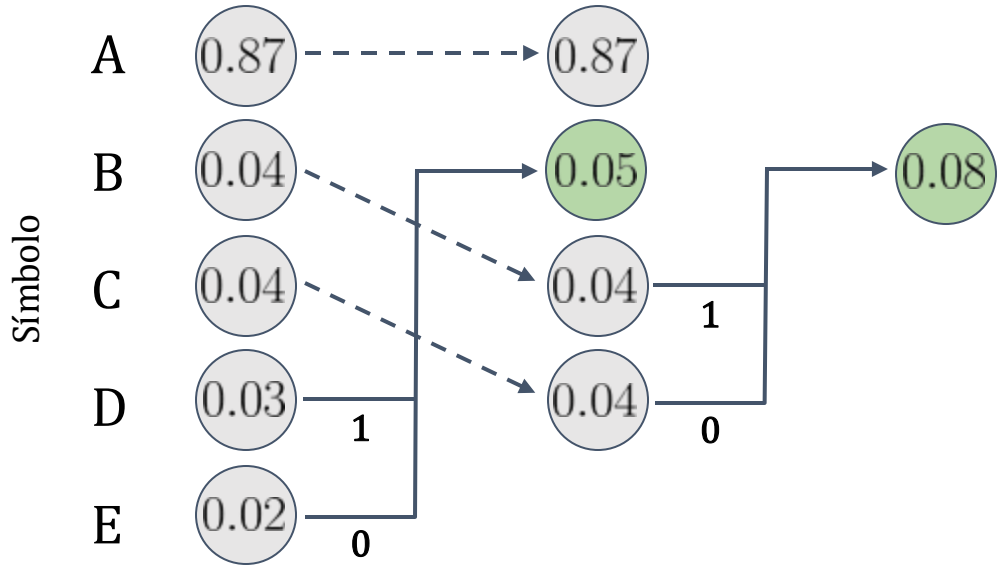
Código de Huffman: Ejemplo

- Ejemplo de construcción del código:



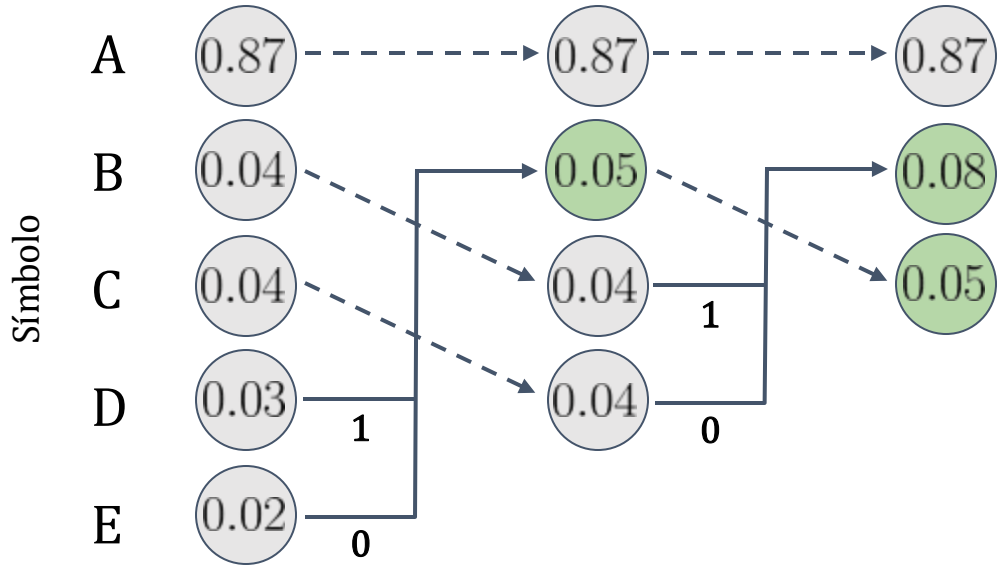
Código de Huffman: Ejemplo

- Ejemplo de construcción del código:



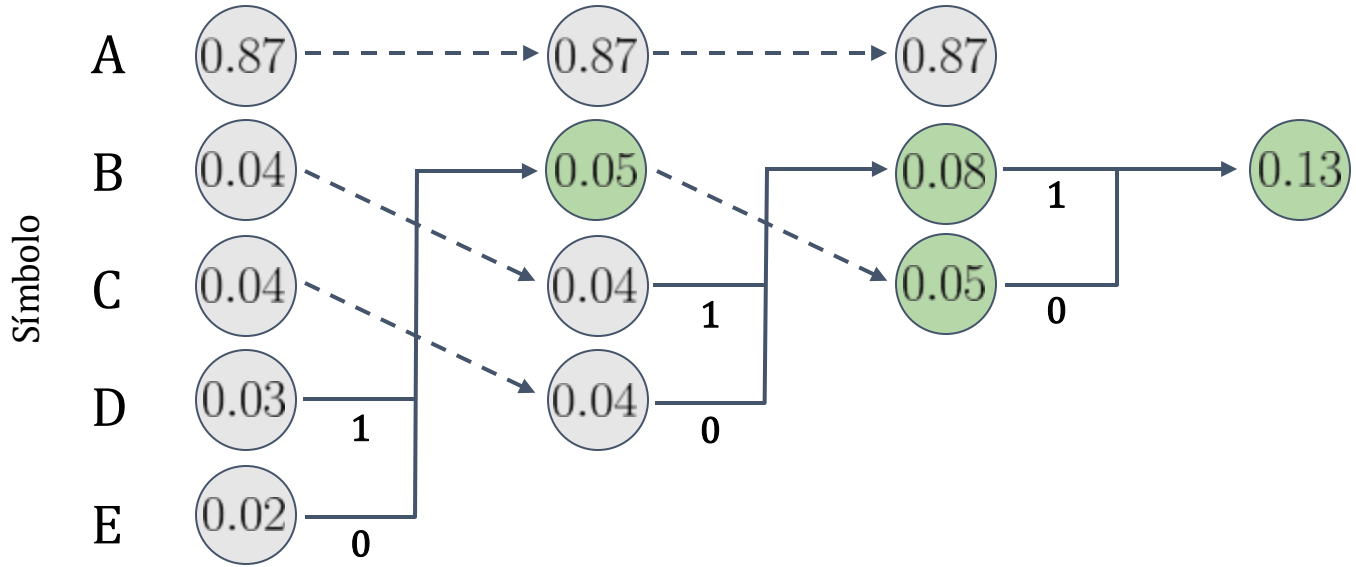
Código de Huffman: Ejemplo

- Ejemplo de construcción del código:



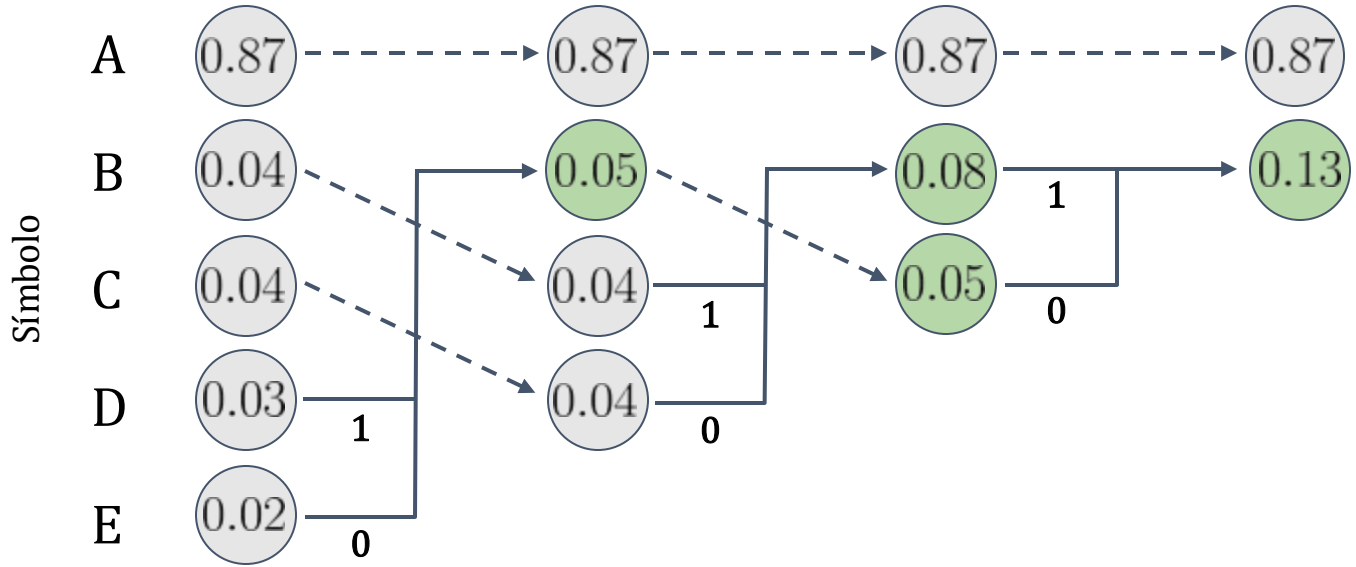
Código de Huffman: Ejemplo

- Ejemplo de construcción del código:



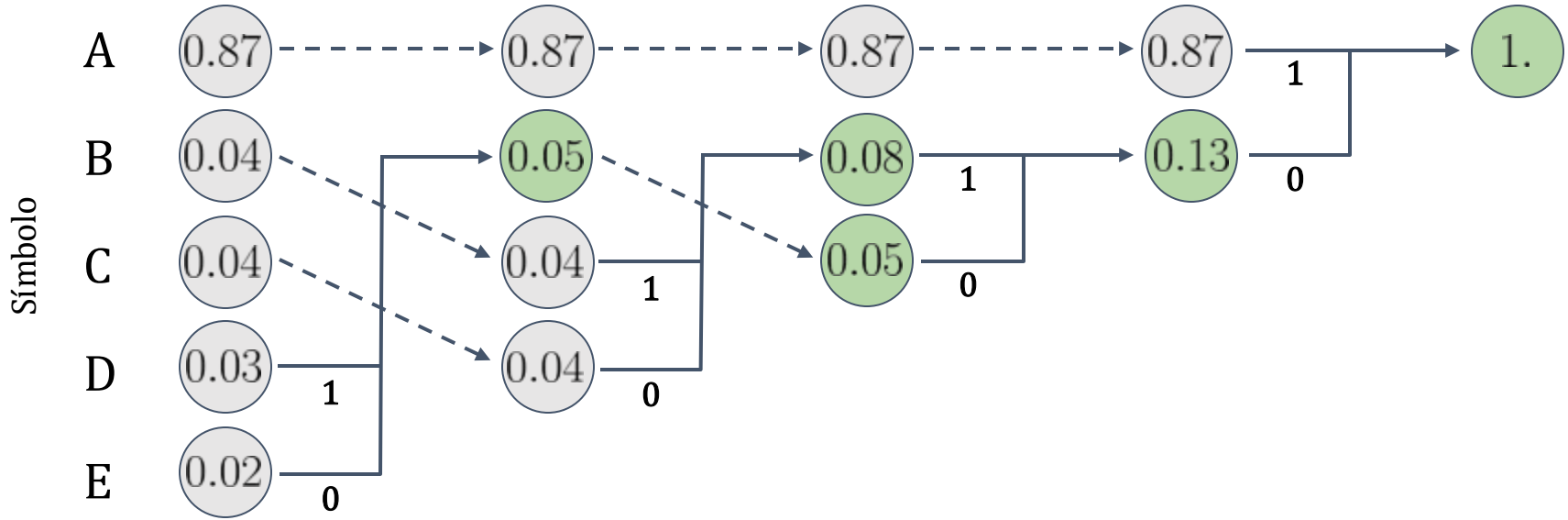
Código de Huffman: Ejemplo

- Ejemplo de construcción del código:



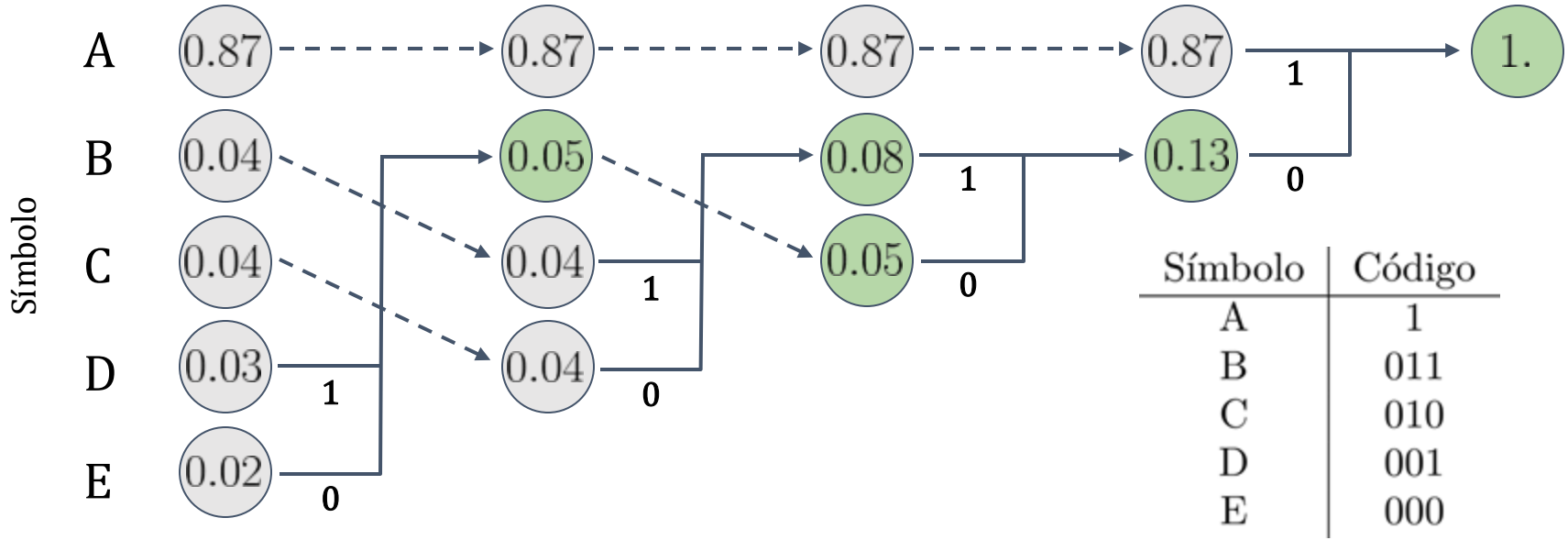
Código de Huffman: Ejemplo

- Ejemplo de construcción del código:



Código de Huffman: Ejemplo

- Ejemplo de construcción del código:



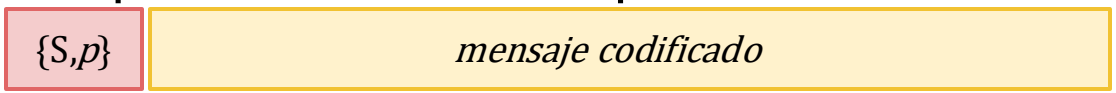
Símbolo	Código
A	1
B	011
C	010
D	001
E	000

Código de Huffman: Algoritmo

- Algoritmo de codificación (semiestático):
  - **Paso 1:** recorrer el mensaje para obtener la probabilidades de los símbolos y preparar las hojas del árbol
  - **Paso 2:** Construir iterativamente el árbol binario hasta llegar a la raíz (con probabilidad 1)
  - **Paso 3:** Codificar el mensaje usando el árbol (paso 2)

Código de Huffman: Algoritmo

- Algoritmo de codificación (semiestático):
  - **Paso 1:** recorrer el mensaje para obtener la probabilidades de los símbolos y preparar las hojas del árbol
  - **Paso 2:** Construir iterativamente el árbol binario hasta llegar a la raíz (con probabilidad 1)
  - **Paso 3:** Codificar el mensaje usando el árbol (paso 2)
- Debemos agregar un *header* con las probabilidades de cada símbolo para que el decodificador pueda reconstruir el mensaje:



Ejercicio!

- Construir la codificación de Huffman para la fuente que emite la suma de dos dados. Comparar la eficiencia respecto a una codificación directa en binario con 4 bits:

Símbolo	Suma	Dados	Frec.	p	h	Código
s_1	2	1:1	1	0.03	5.17	10000
s_2	3	1:2, 2:1	2	0.06	4.17	0110
s_3	4	1:3, 3:1, 2:2	3	0.08	3.59	1001
s_4	5	2:3, 3:2, 1:4, 4:1	4	0.11	3.17	001
s_5	6	2:4, 4:2, 1:5, 5:1, 3:3	5	0.14	2.85	101
s_6	7	3:4, 4:3, 2:5, 5:2, 1:6, 6:1	6	0.17	2.59	111
s_7	8	3:5, 5:3, 2:6, 6:2, 4:4	5	0.14	2.85	110
s_8	9	3:6, 6:3, 4:5, 5:4	4	0.11	3.17	010
s_9	10	4:6, 6:4, 5:5	3	0.08	3.59	000
s_{10}	11	5:6, 6:5	2	0.06	4.17	0111
s_{11}	12	6:6	1	0.03	5.17	10001

Ejercicio!

- Construir la codificación de Huffman para la fuente que emite la suma de dos dados. Comparar la eficiencia respecto a una codificación directa en binario con 4 bits:

Símbolo	Suma	Dados	Frec.	p	h	Código
s_1	2	1:1	1	0.03	5.17	10000
s_2	3	1:2, 2:1	2	0.06	4.17	0110
s_3	4	1:3, 3:1, 2:2	3	0.08	3.59	1001
s_4	5	2:3, 3:2, 1:4, 4:1	4	0.11	3.17	001
s_5	6	2:4, 4:2, 1:5, 5:1, 3:3	5	0.14	2.85	101
s_6	7	3:4, 4:3, 2:5, 5:2, 1:6, 6:1	6	0.17	2.59	111
s_7	8	3:5, 5:3, 2:6, 6:2, 4:4	5	0.14	2.85	110
s_8	9	3:6, 6:3, 4:5, 5:4	4	0.11	3.17	010
s_9	10	4:6, 6:4, 5:5	3	0.08	3.59	000
s_{10}	11	5:6, 6:5	2	0.06	4.17	0111
s_{11}	12	6:6	1	0.03	5.17	10001

- Eficiencia código 4 bits:

$$\frac{H(S)}{L(X)} = \frac{3.27}{4} = 0.818 \text{ bits/dígito binario}$$

- Eficiencia Huffman:

$$\frac{H(S)}{L(X)} = \frac{3.27}{3.35} = 0.976 \text{ bits/dígito binario}$$

Código de Huffman: Propiedades

- Los códigos construidos mediante el método de Huffman satisfacen:

$$H(X) \leq L(S) < H(X) + 1$$

Código de Huffman: Propiedades

- Los códigos construidos mediante el método de Huffman satisfacen:

$$H(X) \leq L(S) < H(X) + 1$$

- Cumplen con la **propiedad del prefijo**: *ningún símbolo codificado completo es prefijo de otro en el conjunto* → Resulta fácil detectar el borde entre dos símbolos codificados

- Un código que cumple la propiedad del prefijo se puede decodificar unívocamente. Ej. 10011110111

Código de Huffman Adaptativo (FGK)

- Codificación *on the fly*: no conocemos las probabilidades de los símbolos.
- Comienza con un árbol vacío
- Cada nodo del árbol almacena un valor de frecuencia de aparición (entero):
  - Las hojas del árbol contienen los símbolos observados hasta el momento y su frecuencia.
  - Los nodos internos contienen la suma de las frecuencias de sus hijos.

Código de Huffman Adaptativo (FGK)

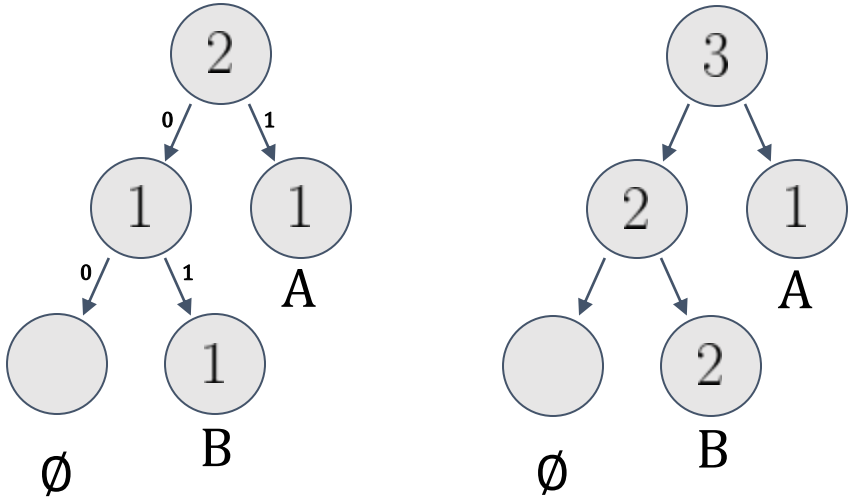
- Se utiliza un nodo especial (“nodo cero” ∅) para identificar el punto de inserción
- Para que sea un árbol de Huffman debe cumplir la propiedad de sibling:

> *“Al recorrer los nodos de abajo hacia arriba, de izquierda a derecha, las frecuencias deben aparecer en orden creciente”*

Código de Huffman Adaptativo (FGK)

- Se utiliza un nodo especial (“nodo cero” ∅) para identificar el punto de inserción
- Para que sea un árbol de Huffman debe cumplir la propiedad de sibling:

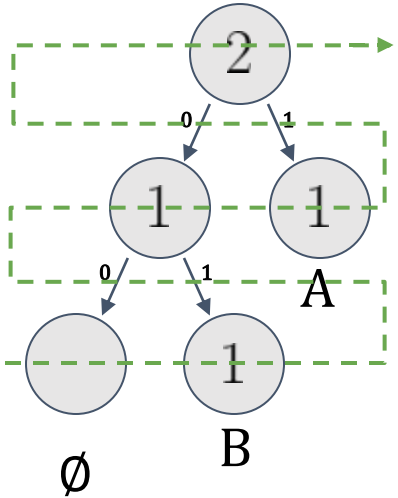
> *“Al recorrer los nodos de abajo hacia arriba, de izquierda a derecha, las frecuencias deben aparecer en orden creciente”*



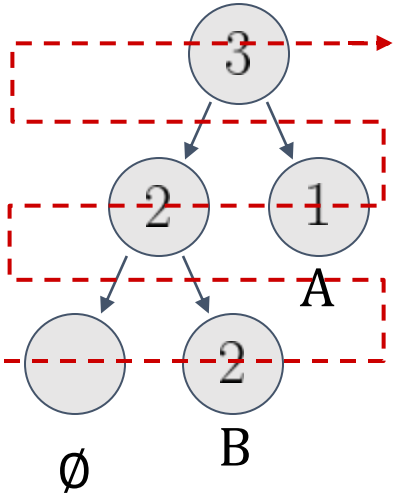
Código de Huffman Adaptativo (FGK)

- Se utiliza un nodo especial (“nodo cero” ∅) para identificar el punto de inserción
- Para que sea un árbol de Huffman debe cumplir la propiedad de sibling:

> *“Al recorrer los nodos de abajo hacia arriba, de izquierda a derecha, las frecuencias deben aparecer en orden creciente”*



verifica la propiedad



no verifica la propiedad

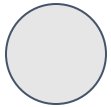
Código de Huffman Adaptativo (FGK): Algoritmo

Mientras haya símbolos:

- **Paso 1**: Leer el próximo símbolo
- **Paso 2**: Generar el código
  - Si el símbolo está en el árbol: transmitir el código asociado
  - Si el símbolo no está en el árbol: transmitir el código ∅ + símbolo literal (no hay código para la primera ocurrencia de un símbolo).
- **Paso 3**: Actualizar el árbol
  - Si el símbolo está en el árbol: incrementar la frecuencia del nodo y propagarlo hacia arriba.
  - Si el símbolo no está en el árbol: dividir el nodo ∅ en dos: el nodo ∅ a la izquierda y el nuevo símbolo con frecuencia 1 a la derecha.
- **Paso 4**: Verificar la propiedad de sibling
  - Si no se cumple: Reestructurar el árbol. Se deben intercambiar los 2 nodos en conflicto (junto con sus subárboles si los tuvieran). El nodo de mayor peso y más próximo al inicio del recorrido se intercambia con el nodo de menor peso y más cercano a la raíz.
  - Volver a verificar la propiedad y reestructurar (hasta que se cumpla)

Huffman Adaptativo (FGK): Ejemplo compresión

1) Leer símbolo 2) Generar código 3) Actualizar el árbol 4) Verificar propiedad de sibling

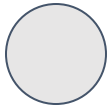


Ø

Huffman Adaptativo (FGK): Ejemplo compresión

1) Leer símbolo 2) Generar código 3) Actualizar el árbol 4) Verificar propiedad de sibling

In: A
Out:

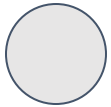


Ø

Huffman Adaptativo (FGK): Ejemplo compresión

1) Leer símbolo 2) Generar código 3) Actualizar el árbol 4) Verificar propiedad de sibling

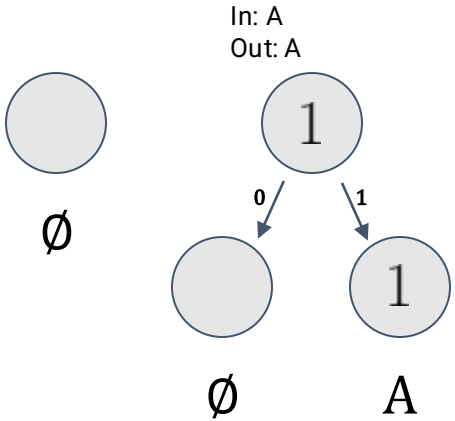
In: A
Out: A



Ø

Huffman Adaptativo (FGK): Ejemplo compresión

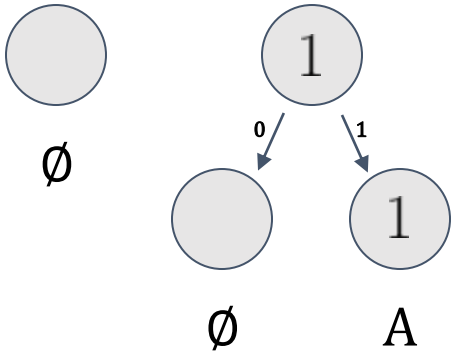
1) Leer símbolo 2) Generar código 3) Actualizar el árbol 4) Verificar propiedad de sibling



Huffman Adaptativo (FGK): Ejemplo compresión

1) Leer símbolo 2) Generar código 3) Actualizar el árbol 4) Verificar propiedad de sibling

In: A
Out: A



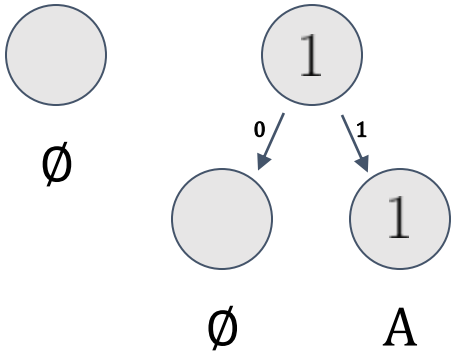
Lista sibling: Ø 1 1

Huffman Adaptativo (FGK): Ejemplo compresión

1) Leer símbolo 2) Generar código 3) Actualizar el árbol 4) Verificar propiedad de sibling

In: A
Out: A

In: B
Out:

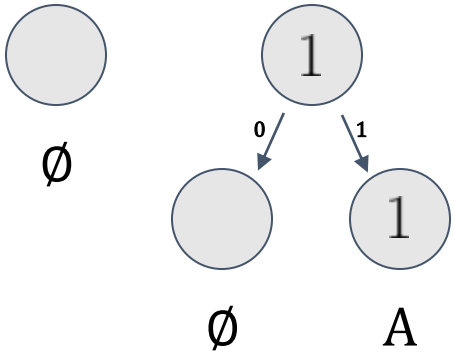


Lista sibling: Ø 1 1

Huffman Adaptativo (FGK): Ejemplo compresión

1) Leer símbolo 2) Generar código 3) Actualizar el árbol 4) Verificar propiedad de sibling

In: A
Out: A

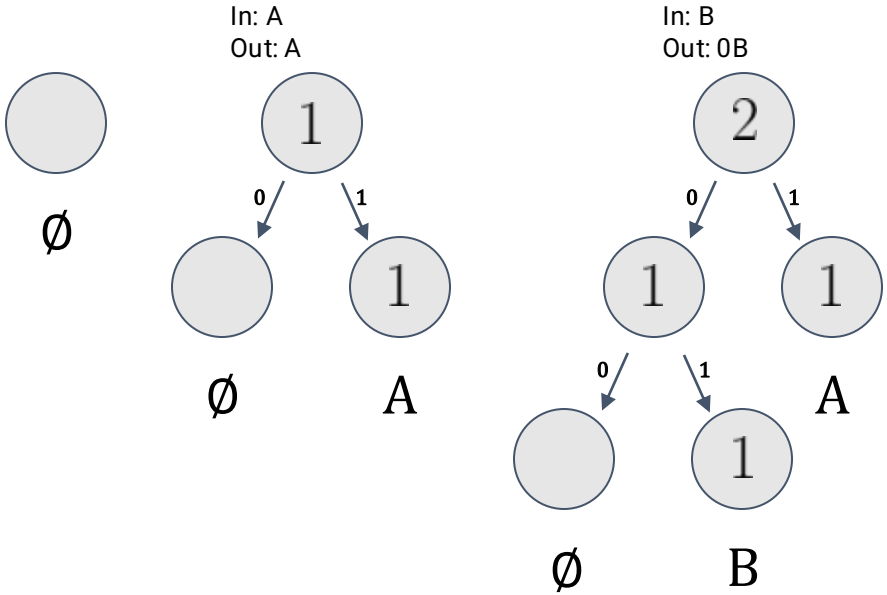


In: B
Out: 0B

Lista sibling: Ø 1 1

Huffman Adaptativo (FGK): Ejemplo compresión

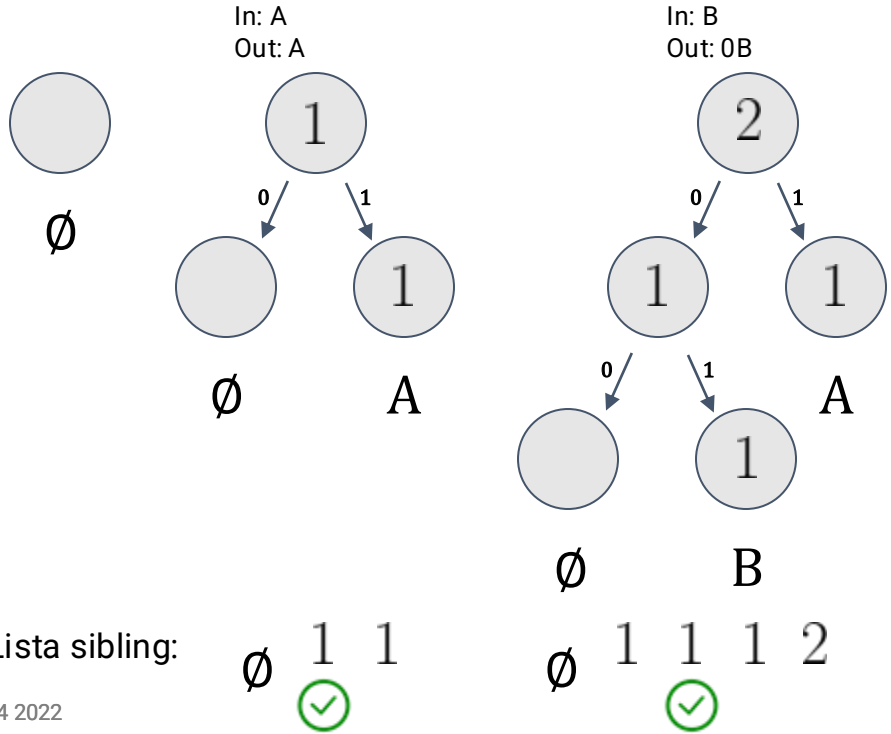
1) Leer símbolo 2) Generar código 3) Actualizar el árbol 4) Verificar propiedad de sibling



Lista sibling: Ø 1 1

Huffman Adaptativo (FGK): Ejemplo compresión

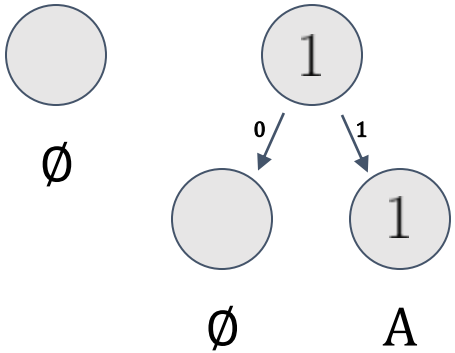
1) Leer símbolo 2) Generar código 3) Actualizar el árbol 4) Verificar propiedad de sibling



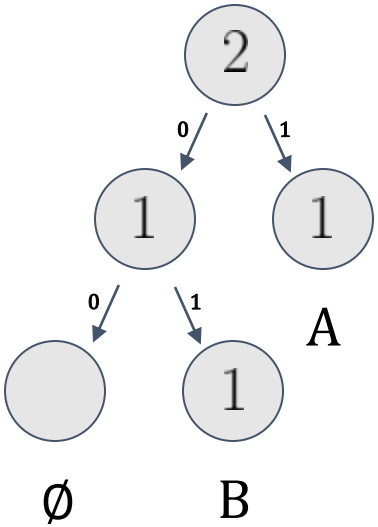
Huffman Adaptativo (FGK): Ejemplo compresión

1) Leer símbolo 2) Generar código 3) Actualizar el árbol 4) Verificar propiedad de sibling

In: A
Out: A

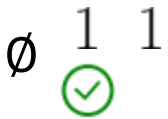


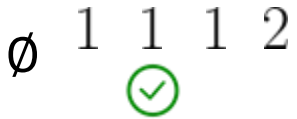
In: B
Out: 0B



In: B
Out:

Lista sibling:

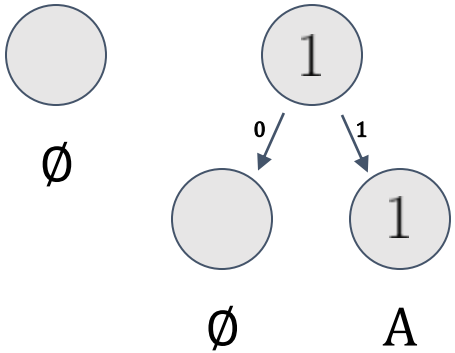




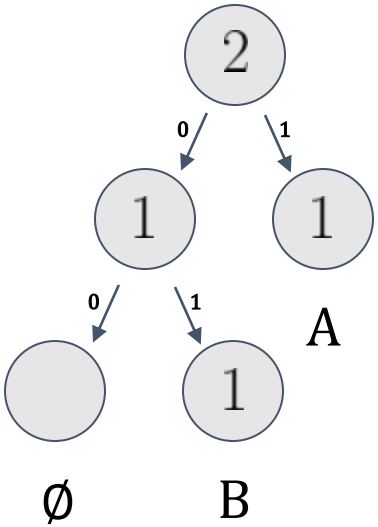
Huffman Adaptativo (FGK): Ejemplo compresión

1) Leer símbolo 2) Generar código 3) Actualizar el árbol 4) Verificar propiedad de sibling

In: A
Out: A

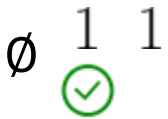


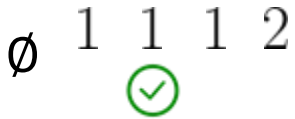
In: B
Out: 0B



In: B
Out: 01

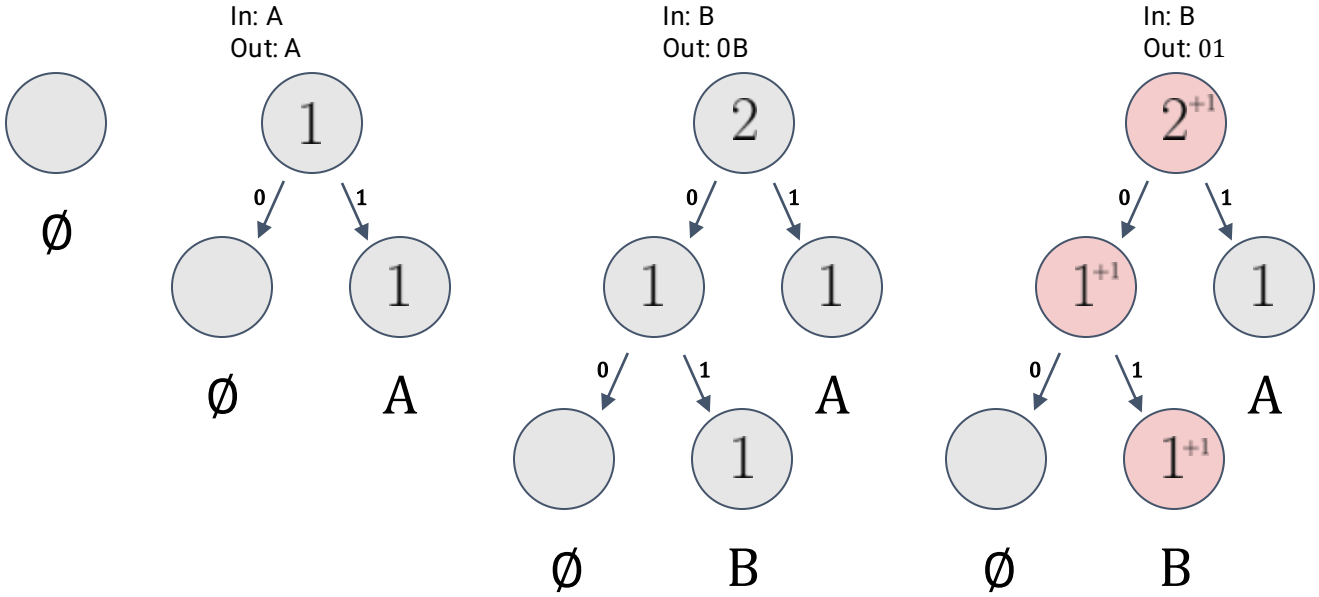
Lista sibling:





Huffman Adaptativo (FGK): Ejemplo compresión

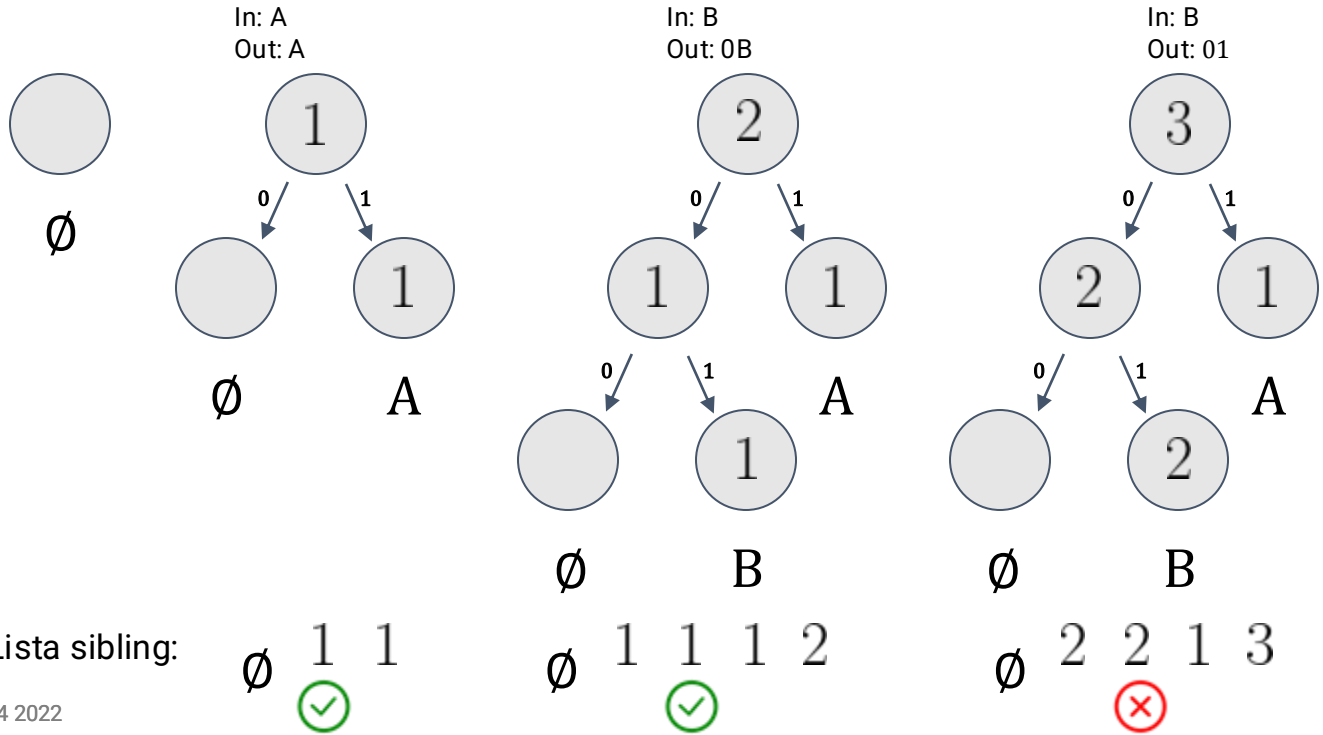
1) Leer símbolo 2) Generar código 3) Actualizar el árbol 4) Verificar propiedad de sibling



Lista sibling: Ø 1 1 Ø 1 1 1 2

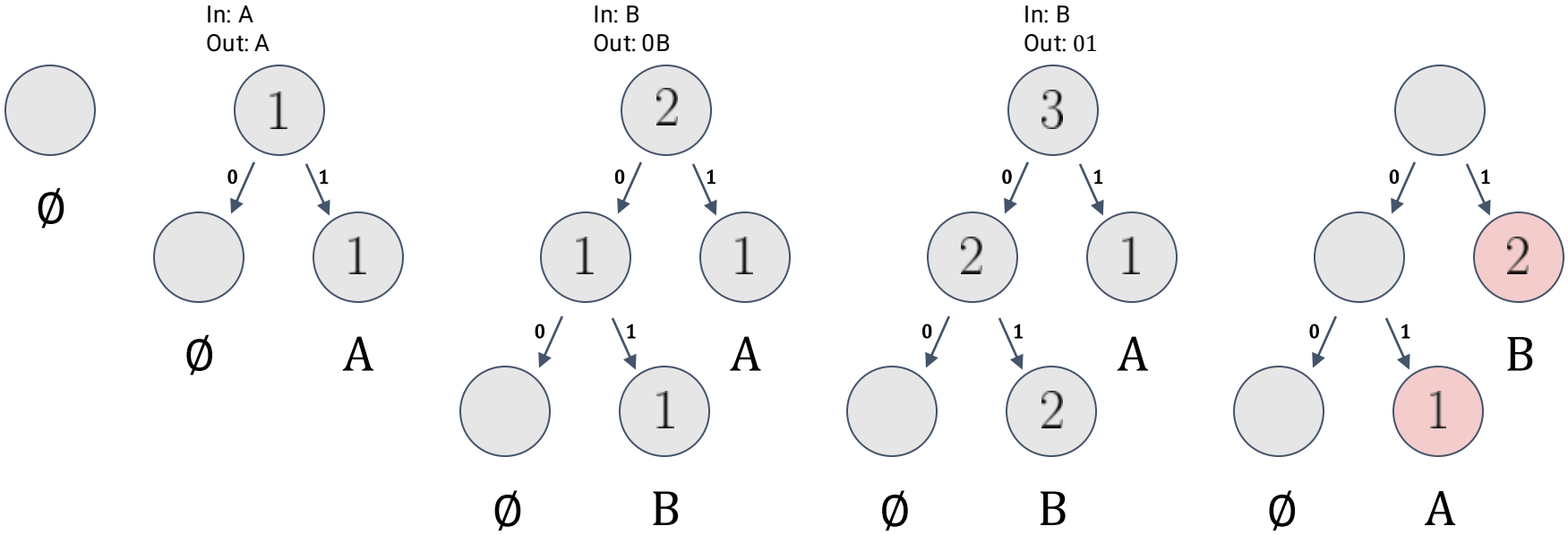
Huffman Adaptativo (FGK): Ejemplo compresión

1) Leer símbolo 2) Generar código 3) Actualizar el árbol 4) Verificar propiedad de sibling



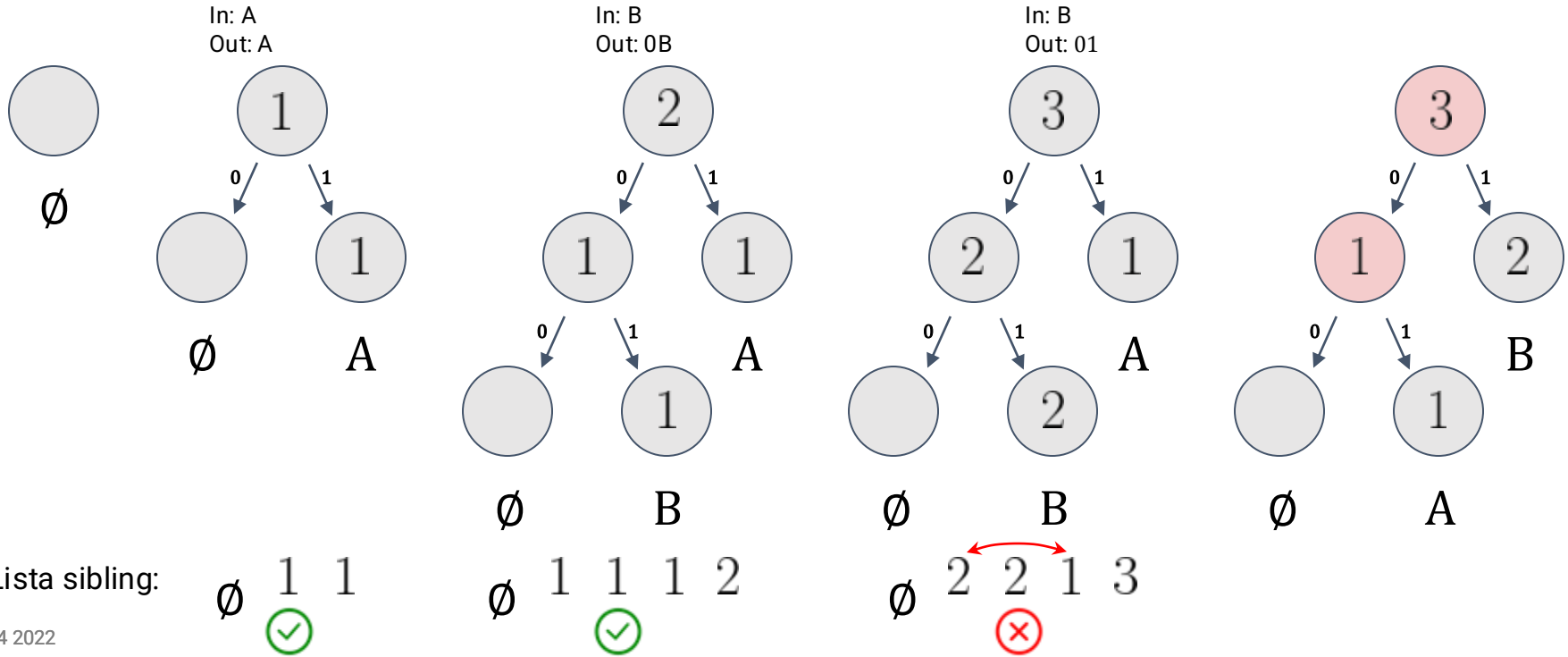
Huffman Adaptativo (FGK): Ejemplo compresión

1) Leer símbolo 2) Generar código 3) Actualizar el árbol 4) Verificar propiedad de sibling



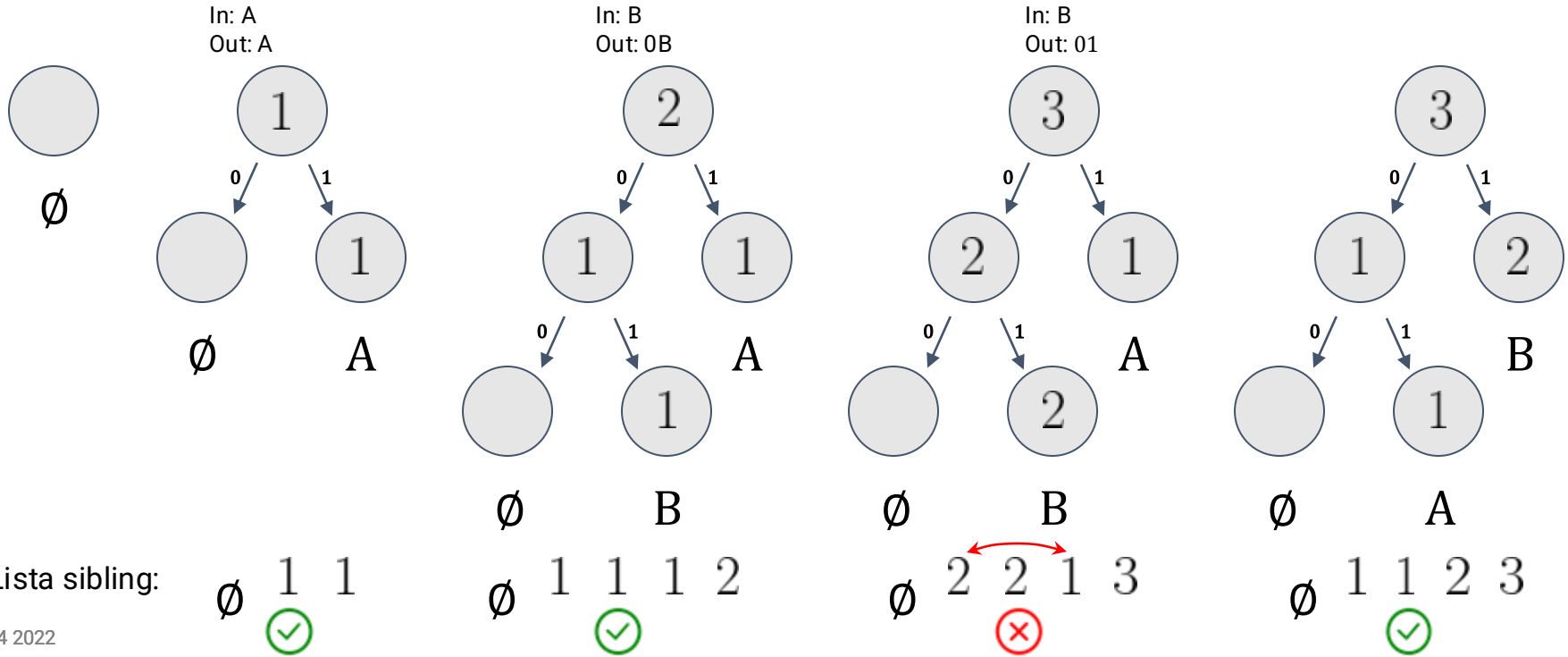
Huffman Adaptativo (FGK): Ejemplo compresión

1) Leer símbolo 2) Generar código 3) Actualizar el árbol 4) Verificar propiedad de sibling



Huffman Adaptativo (FGK): Ejemplo compresión

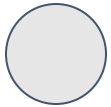
1) Leer símbolo 2) Generar código 3) Actualizar el árbol 4) Verificar propiedad de sibling



Huffman Adaptativo (FGK): Ejemplo descompresión

El descompresor es simétrico: recibe el input codificado ($\mathbf{A0B01}$) , construye el mismo árbol y decodifica con los mismos pasos que el codificador.

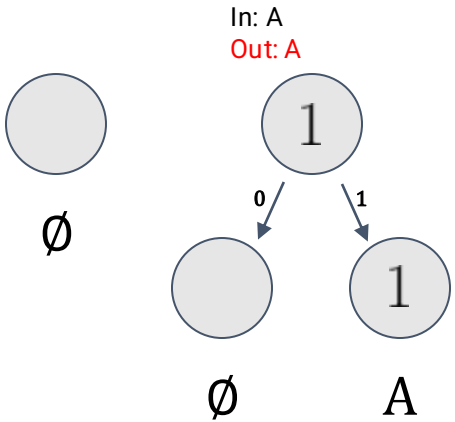
In: A
Out:



Ø

Huffman Adaptativo (FGK): Ejemplo descompresión

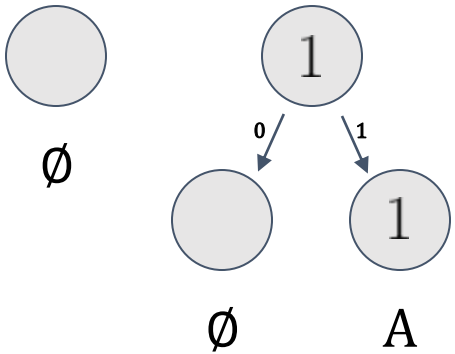
El descompresor es simétrico: recibe el input codificado (**A0B01**) , construye el mismo árbol y decodifica con los mismos pasos que el codificador.



Huffman Adaptativo (FGK): Ejemplo descompresión

El descompresor es simétrico: recibe el input codificado (`A0B01`) , construye el mismo árbol y decodifica con los mismos pasos que el codificador.

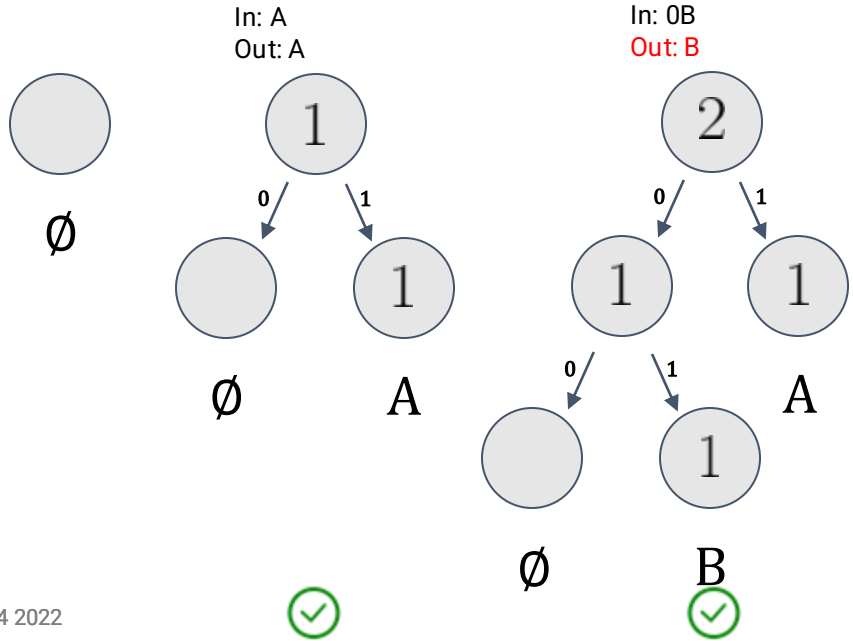
In: A
Out: A



In: 0B
Out:

Huffman Adaptativo (FGK): Ejemplo descompresión

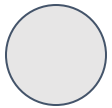
El descompresor es simétrico: recibe el input codificado (`A0B01`) , construye el mismo árbol y decodifica con los mismos pasos que el codificador.

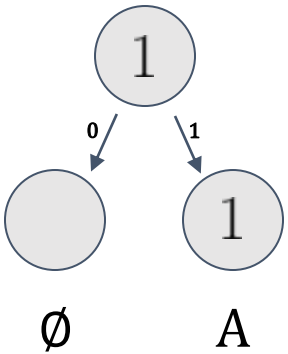


Huffman Adaptativo (FGK): Ejemplo descompresión

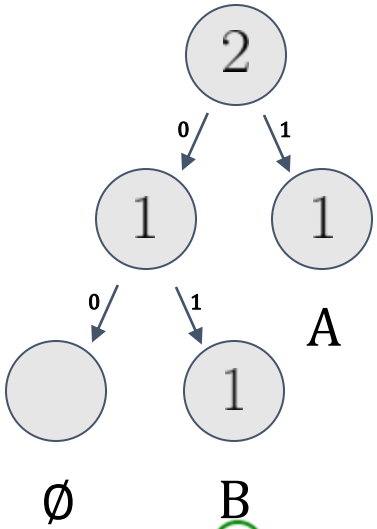
El descompresor es simétrico: recibe el input codificado (**A0B01**) , construye el mismo árbol y decodifica con los mismos pasos que el codificador.

In: A
Out: A





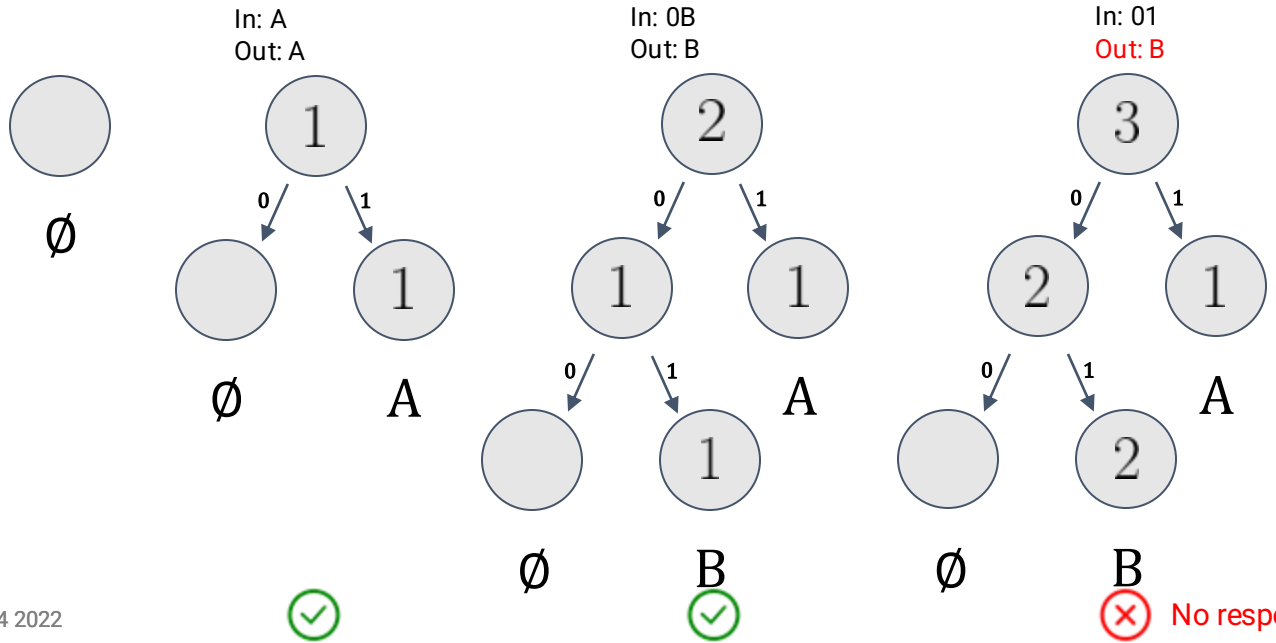
In: 0B
Out: B



In: 01
Out:

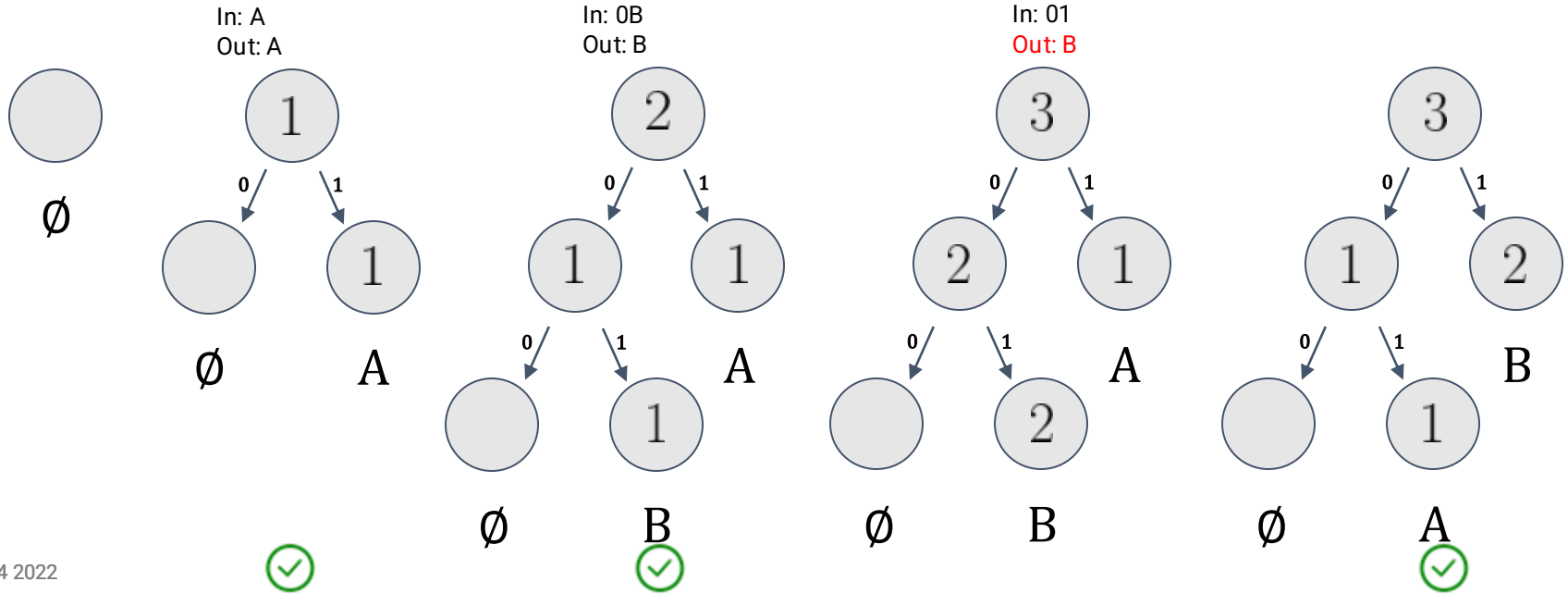
Huffman Adaptativo (FGK): Ejemplo descompresión

El descompresor es simétrico: recibe el input codificado (**A0B01**) , construye el mismo árbol y decodifica con los mismos pasos que el codificador.



Huffman Adaptativo (FGK): Ejemplo descompresión

El descompresor es simétrico: recibe el input codificado (`A0B01`) , construye el mismo árbol y decodifica con los mismos pasos que el codificador.



Métodos con diccionarios

- Se reemplazan frases (secuencias de símbolos) repetidas en el mensaje por un índice a una ocurrencia anterior.
- Construcción dinámica de un diccionario de frases.
- Los índices ocupan menos espacio que las frases.
- A mayor longitud de las frases repetidas, mayor tasa de compresión.
- Son asimétricos: el compresor es más costoso (por la búsqueda de coincidencias) que el descompresor (solo debe recuperar las frases indicadas por los elementos del diccionario)

Run Length Encoding (RLE)

- Reemplaza secuencias de símbolos consecutivos iguales por un par símbolo/repeticiones (S,R):

Run Length Encoding (RLE)

- Reemplaza secuencias de símbolos consecutivos iguales por un par símbolo/repeticiones (S,R):

  Original: AAAAAAAABBBCCCCC (16 bytes)
  Comprimido: A8B3C5 (6 bytes)

Run Length Encoding (RLE)

- Reemplaza secuencias de símbolos consecutivos iguales por un par símbolo/repeticiones *(S,R)*:

Original: AAAAAAAABBBCCCCC (16 bytes)
Comprimido: A8B3C5 (6 bytes)

Original: SALIMOS CAMPEONES DEL MUNDO (27 bytes)
Comprimido: S1A1L1I1M1O1S1 1C1A1M1P1E1O1N1E1S1 1D1E1L1 1M1U1N1D1O1 (**54 bytes!!!**)

Run Length Encoding (RLE)

- Método aplicable siempre y cuando sepamos que los datos tienen repeticiones consecutivas de símbolos (imágenes)

Run Lenght Encoding (RLE)

- Método aplicable siempre y cuando sepamos que los datos tienen repeticiones consecutivas de símbolos (imágenes)

- Forma parte de estándares de compresión (JPEG, MPEG, etc.)

RLE para imágenes binarias

- Secuencias alternadas de pixeles blancos y negros
- No se modifican los símbolos (redundante), solo la longitud
- **Convención:** La longitud inicial siempre es para el símbolo 0 (secuencia de pixeles negros)
- Si la imagen presenta un 1, se comienza con $\mathrm{R}=0$

RLE para imágenes binarias

- Comienza en 0:
  Original: 000000001111000000000
  Comprimido: 8 5 9

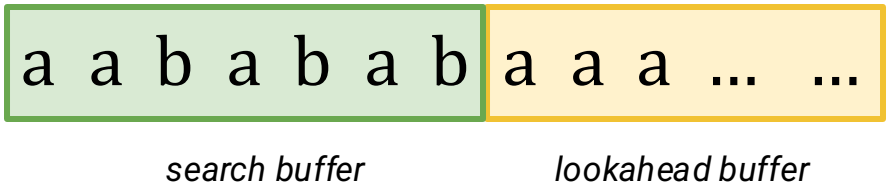
- Comienza en 1:
  Original: 11100000
  Comprimido: 0 3 5

Métodos de compresión LZ (Lempel & Ziv)

- Familia de métodos:
  - **LZ-77** → LZR, LZSS, LZH, LZB, LZFG
  - **LZ-78** → LZJ, LZW, LZMW, LZT, LZC, LZFG

Métodos de compresión LZ (Lempel & Ziv)

- Familia de métodos:
  - **LZ-77** → LZR, LZSS, LZH, LZB, LZFG
  - **LZ-78** → LZJ, LZW, LZMW, LZT, LZC, LZFG
- Utilizan una ventana de búsqueda que contiene la secuencia de símbolos a codificar (*lookahead buffer*) y la secuencia ya procesada (*search buffer*):



LZW

- Primer método universal de compresión de datos ampliamente utilizado en las computadoras
- Comprime texto en inglés a la mitad de su tamaño original (para archivos grandes)
- Es de dominio público
- Forma parte de apps/formatos de compresión:

  arj, pkzip, gif, tiff, zip, pdf

Algoritmo LZW: Compresión

Utiliza un diccionario precargado (por ejemplo, 256 ASCII)

Mientras haya símbolos de entrada:

- **Paso 1**: Buscar la frase de mayor coincidencia en el diccionario y codificar
- **Paso 2**: Agregar al diccionario una nueva entrada: frase codificada + siguiente símbolo no coincidente

Algoritmo LZW: Ejemplo compresión

Original: a a b a b a b a a a

símbolo/s	código	diccionario	
diccionario precargado		a	0
		b	1

Algoritmo LZW: Ejemplo compresión

Original: a a b a b a b a a a

símbolo/s	código	diccionario	
diccionario precargado		a	0
		b	1
a	0		

Algoritmo LZW: Ejemplo compresión

Original: a a b a b a b a a a

símbolo/s	código	diccionario	
diccionario precargado		a	0
		b	1
a	0	aa	2

Algoritmo LZW: Ejemplo compresión

Original: a a b a b a b a a a

símbolo/s	código	diccionario	
diccionario precargado		a	0
		b	1
a	0	aa	2
a	0		

Algoritmo LZW: Ejemplo compresión

Original: a a b a b a b a a a

símbolo/s	código	diccionario	
diccionario precargado		a b	0 1
a	0	aa	2
a	0	ab	3

Algoritmo LZW: Ejemplo compresión

Original: a a b a b a b a a a

símbolo/s	código	diccionario	
diccionario precargado		a	0
		b	1
a	0	aa	2
a	0	ab	3
b	1		

Algoritmo LZW: Ejemplo compresión

Original: a a b a b a b a a a

símbolo/s	código	diccionario	
diccionario precargado		a b	0 1
a	0	aa	2
a	0	ab	3
b	1	ba	4

Algoritmo LZW: Ejemplo compresión

Original: a a b a b a b a a a

símbolo/s	código	diccionario	
diccionario precargado		a	0
		b	1
a	0	aa	2
a	0	ab	3
b	1	ba	4
ab	3		

Algoritmo LZW: Ejemplo compresión

Original: a a b a b a b a a a

símbolo/s	código	diccionario	
diccionario precargado		a b	0 1
a	0	aa	2
a	0	ab	3
b	1	ba	4
ab	3	aba	5

Algoritmo LZW: Ejemplo compresión

Original: a a b a b a b a a a

símbolo/s	código	diccionario	
diccionario precargado		a b	0 1
a	0	aa	2
a	0	ab	3
b	1	ba	4
ab	3	aba	5
aba	5		

Algoritmo LZW: Ejemplo compresión

Original: a a b a b a b a a a

símbolo/s	código	diccionario	
diccionario precargado		a b	0 1
a	0	aa	2
a	0	ab	3
b	1	ba	4
ab	3	aba	5
aba	5	abaa	6

Algoritmo LZW: Ejemplo compresión

Original: a a b a b a b a a a

símbolo/s	código	diccionario	
diccionario precargado		a b	0 1
a	0	aa	2
a	0	ab	3
b	1	ba	4
ab	3	aba	5
aba	5	abaa	6
aa	2		
...	...		

Algoritmo LZW: Descompresión

Utiliza una diccionario precargado (por ejemplo, 256 ASCII)

Mientras haya símbolos de entrada

- **Paso 1**: Busca la entrada en el diccionario, decodifica la frase asociada y avanza
- **Paso 2**: Agrega al diccionario una nueva entrada: decodificación anterior + primer símbolo de la decodificación actual

  Caso especial si el código no está en el diccionario: decodificar código anterior + 1º símbolo del código anterior y agregarlo al diccionario. Usar esa nueva entrada para decodificar (se debe a que el codificador conoce el siguiente símbolo, pero el decodificador aún no!).

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

código	símbolo/s	diccionario	
diccionario precargado		0	a
		1	b

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

código	símbolo/s	diccionario
diccionario precargado		0 a 1 b
0	a	

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

código	símbolo/s	diccionario	
diccionario precargado		0	a
		1	b
0	a		

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

código	símbolo/s	diccionario	
diccionario precargado		0 1	a b
0	a	2	aa

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

código	símbolo/s	diccionario	
diccionario precargado		0	a
		1	b
0	a	2	aa
0	a		

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

código	símbolo/s	diccionario	
diccionario precargado		0	a
		1	b
0	a	2	aa
0	a		

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

código	símbolo/s	diccionario	
diccionario precargado		0 1	a b
0	a	2	aa
0	a	3	ab

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

código	símbolo/s	diccionario	
diccionario precargado		0 1	a b
0	a	2	aa
0	a	3	ab
1	b		

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

código	símbolo/s	diccionario	
diccionario precargado		0 1	a b
0	a	2	aa
0	a	3	ab
1	b		

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

código	símbolo/s	diccionario	
diccionario precargado		0 1	a b
0	a	2	aa
0	a	3	ab
1	b	4	ba

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

código	símbolo/s	diccionario	
diccionario precargado		0 1	a b
0	a	2	aa
0	a	3	ab
1	b	4	ba
3	ab		

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

Caso especial!
El 5 no está en el diccionario

código	símbolo/s	diccionario	
diccionario precargado		0	a
		1	b
0	a	2	aa
0	a	3	ab
1	b	4	ba
3	ab		

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

código	símbolo/s	diccionario	
diccionario precargado		0 1	a b
0	a	2	aa
0	a	3	ab
1	b	4	ba
3	ab	5	aba

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

código	símbolo/s	diccionario	
diccionario precargado		0 1	a b
0	a	2	aa
0	a	3	ab
1	b	4	ba
3	ab	5	aba
5	aba		

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

código	símbolo/s	diccionario	
diccionario precargado		0 1	a b
0	a	2	aa
0	a	3	ab
1	b	4	ba
3	ab	5	aba
5	aba		

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

código	símbolo/s	diccionario	
diccionario precargado		0 1	a b
0	a	2	aa
0	a	3	ab
1	b	4	ba
3	ab	5	aba
5	aba	6	abaa

Algoritmo LZW: Ejemplo descompresión

Mensaje recibido: 0 0 1 3 5 2

código	símbolo/s	diccionario	
diccionario precargado		0 1	a b
0	a	2	aa
0	a	3	ab
1	b	4	ba
3	ab	5	aba
5	aba	6	abaa
2	aa		
…	…		