Facultad de Ciencias Exactas
UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES

TEORÍA DE LA INFORMACIÓN

Tratamiento probabilístico de la Información

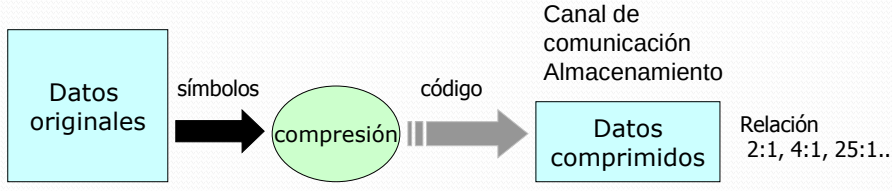
Compresión de datos

INGENIERÍA DE SISTEMAS

Cursada 2017

COMPRESIÓN DE DATOS

Objetivo: Reducir la cantidad de datos para representar la información, eliminando (o reduciendo) la redundancia.



La compresión permite:

- Aumentar la capacidad de almacenamiento de los dispositivos
- Transmitir en menos tiempo información por el canal

COMPRESIÓN Y REDUNDANCIA

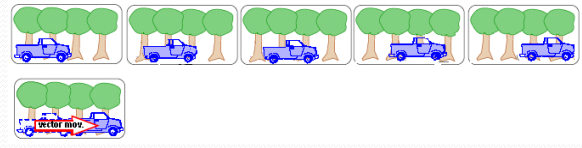
Algunos ejemplos

Texto AAAAAAAAAAABBBBBBBBCCCCC
(A,11)(B,8)(C,5)

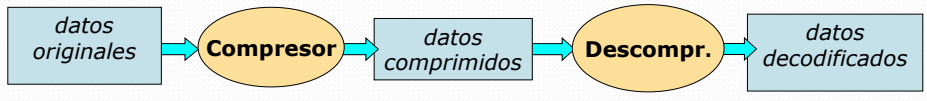
Imágenes


Original (1.1 Mb) **(38Kb)** **30:1**

Video



COMPRESIÓN / DESCOMPRESIÓN



canal de comunicación almacenamiento

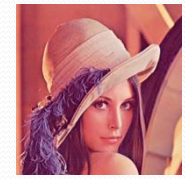
genera una representación compacta de los datos de entrada (mensaje de la fuente)

reconstruye el mensaje original (o una aproximación) a partir de la representación comprimida

MÉTODOS DE COMPRESIÓN

Sin Pérdida o reversibles (loosless compression)

- Mantienen la integridad de la información (los datos descomprimidos *coinciden* con los datos originales)
- Se aplica a texto, bases de datos, código fuente, imágenes críticas, …
- Tasas de compresión moderadas: *2:1* (texto) *4:1* (imágenes)
- La longitud media del código está acotada por la entropía $(H \leq \bar{L})$



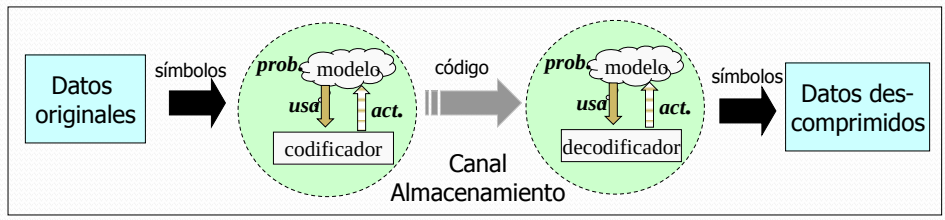
Original

Con Pérdida o irreversibles (lossy compression)

- No mantienen la integridad de la información (los datos descomprimidos son *aproximación* a los originales)
- Se plantean métodos aplicables a imágenes, video, sonido
- Tasas de compresión altas: *30:1* (imágenes) *200:1* (video)
- No tienen como límite la entropía (pérdida de información) $(\bar{L} < H)$

Descomprimida (128:1)

COMPRESIÓN : MODELO/CODIFICADOR



- **Modelo:** Se basa en las *probabilidades* de los símbolos de la fuente (sin memoria, con memoria, extensión a orden n)
- **Codificador:** Codifica los datos de entrada usando el modelo
- El codificador y el decodificador se basan en el mismo modelo
- El modelo puede ser:
  - **Estático** → es fijo, es conocido por codificador y decodificador (universal)
  - **Semi-estático** → es fijo, pero se construye a partir de los datos a comprimir
  - **Dinámico o adaptativo** → el modelo se actualiza durante el proceso

MODELOS: CARACTERISTICAS

Métodos de compresión sin pérdida

$(H \leq \bar{L})$

Estadísticos
(codifican un símbolo por vez)

- **Estáticos**
  (no cambia modelo es universal)
- **Semiestáticos**
  (arma el modelo con los datos)
- **Adaptativos**
  (actualiza el modelo al leer cada dato)

Basados en diccionario
(codifican secuencias de símbolos)

- **Estáticos**
  (no cambia diccionario)
- **Semiestáticos**
  (arma diccionario con datos)
- **Adaptativos**
  (actualiza el diccionario al leer cada dato)

Ej. Tabla

pi	si	ci
1/2	A	0
1/4	B	10
1/4	C	11

Ej. Diccionario

ci	frases
0	ABAB
1	ABCD
2	CBA
..	..
30	A

MODELOS: CARACTERÍSTICAS

- **Estático**
  (+) Requiere una única pasada por los datos para codificar
  (-) La distribución de probabilidades (fija) puede diferir de los datos a codificar
  (+) No se debe transmitir/almacenar la distrib. de prob. al decodificador
- **Semi-estático**
  (-) Requiere dos pasadas por los datos: una arma la tabla, y otra codifica
  (+) La distribución de probabilidades se ajusta a los datos a comprimir
  (-) Se debe transmitir/almacenar la distrib. de prob. al decodificador
- **Dinámico**
  (+) Requiere una única pasada por los datos para codificar
  (+) La distribución de probabilidades se va ajustando a los datos al procesar
  (+) No se debe transmitir/almacenar la distrib. de prob. al decodificador
  (-) Algoritmos más complejos, de mayor costo computacional

MÉTODOS DE COMPRESIÓN

SIN PÉRDIDA	CON PÉRDIDA
• Métodos estadísticos (codifican un símbolo por vez): ○ Shannon, Fano ○ Huffman estático y semiestático ○ Huffman dinámico (FGK) ○ Codificación aritmética ... • Basados en diccionario o sustitucionales: ○ Algoritmos LZ (codifican referencias a cadenas de símbolos previos): LZ77, LZ78 y sus variantes (pkzip, arj, winrar, compress,pdf, png) • Run-Length (codifican secuencias de símbolos coincidentes)	• JPEG • Run-Lenght c/pérdida • Compresión Fractal • VQ (Cuantificación vectorial) • MPEG • ...

MEDIDAS DE PERFORMANCE

Compiten entre sí. Depende del interés del usuario.

Tasa de compresión → Relación del tamaño de los datos originales (t_O) vs. tamaño de los datos comprimidos (t_C)

Se expresa generalmente como ***N : 1*** donde $N = t_O / t_C$

Ejemplo archivo original: 100 Mb - archivo comprimido: 20 Mb
→ compresión 5:1 (reducción 80%)

Tiempo de compresión/descompresión

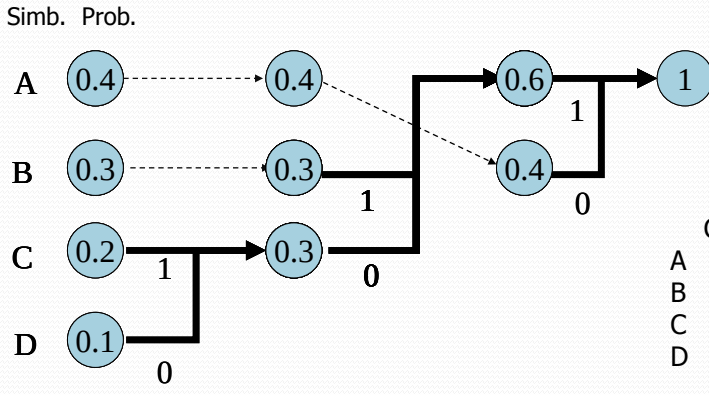
Los algoritmos pueden ser simétricos o asimétricos (dependiendo si involucran igual costo computacional o no)

Calidad→ error o pérdida de calidad entre datos originales y reconstruidos (para los métodos irreversibles)

Ej: error medio (absoluto o cuadrático) y otras medidas derivadas

ALGORITMO de HUFFMAN

David Huffman (1950)



Ejemplo In: AABADDC Out: 00110100100101

IMPORTANTE: Para que la compresión sea efectiva se debe hacer un procesamiento de los sucesivos códigos a nivel bit

ALGORITMO de HUFFMAN

- Genera un código óptimo (mínima longitud media $<L>$) para el conjunto de símbolos de una fuente S
- Para fuentes especiales, con probabilidades $\sim 1/2^a$, $<L> \sim H(S)$
- No suele alcanzar el límite $H(S)$ porque no considera ´bits fraccionales´ (necesita al menos un bit para codificar un símbolo)

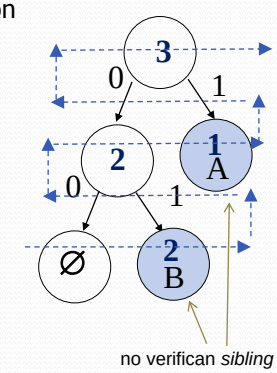
  Ejemplo: S= {s1, s2} con prob. {0.99, 0.01} → $H(S) \approx 0{,}081$
  pero el código asignará 1 bit a cada símbolo
- Si $H(S)$ es grande, la diferencia fraccional $<L> - H(S)$ es despreciable, pero si $H(S)$ es pequeña la diferencia puede ser considerable
- Para la extensión de orden m se cumple que: $H(S) \leq <L_m>/m < H(S) + 1/m$
  → La performance del código mejora a expensas de un aumento exponencial del tamaño de la tabla de códigos
- Forma parte de la especificación de formatos y estándares de compresión (JPEG, MP3, MPEG, GZIP, …)

HUFFMAN DINÁMICO – ALGORITMO FGK

Faller(1973), Gallager(1978), Knuth(1985)

- El árbol de código inicialmente está vacío y se construye dinámicamente a medida que se procesa el mensaje
- A cada nodo del árbol se le asocia un símbolo y un peso (frecuencia)
- Nodo especial –nodo ∅ - asociado al punto de inserción
- Para que sea un árbol de Huffman debe cumplir siempre la propiedad de *sibling*:

  si se recorren los nodos de abajo hacia arriba y de izquierda a derecha, los pesos de los nodos deben aparecen en orden no decreciente



HUFFMAN DINÁMICO – ALGORITMO FGK

Faller(1973), Gallager(1978), Knuth(1985)

Algoritmo:

1) Leer el próximo símbolo del mensaje

2) Generar el código:

- si el símbolo está en el árbol → emitir el código binario asociado a este símbolo
- si no está → emitir el código del nodo ∅ seguido del símbolo literal nuevo (para el primer símbolo del mensaje no hay código para el nodo ∅)

3) Actualizar el árbol:

- Si el símbolo está en el árbol → incrementar la frecuencia del nodo y de los nodos en los niveles superiores acordemente
- Si no está → dividir el nodo ∅ en dos: a la izquierda se deja el nodo ∅ y a la derecha se inserta el nuevo símbolo con frecuencia 1

4) Verificar la propiedad de *sibling* :

- Si no se cumple → reestructurar el árbol, intercambiando los 2 nodos en conflicto (con sus subárboles si los tuvieran):
  - el nodo de peso mayor y más próximo al inicio del recorrido
  - con el nodo de peso menor y más cercano a la raíz

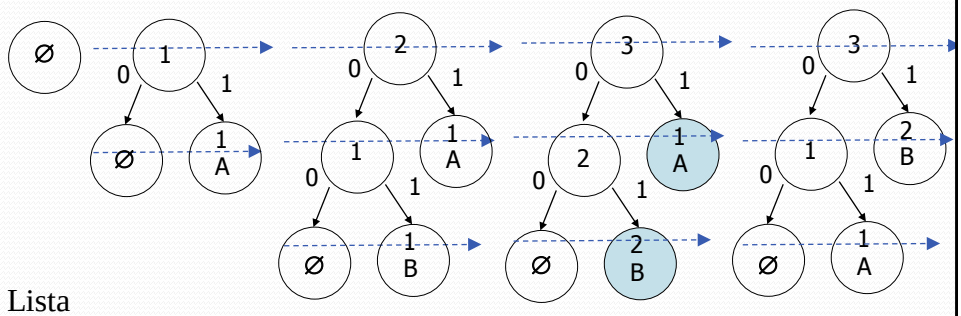
5) Si no terminó el mensaje volver al paso 1)

HUFFMAN DINÁMICO – ALGORITMO FGK

In: A
Out: **A**

In: B
Out: **0 B**

In: B
Out: **01**



Lista

Ø 1 1 | Ø 1 1 1 2 | Ø 2 2 1 3 | Ø 1 1 2 3

No verifica la propiedad de *sibling*:
Intercambiar el de peso mayor y más cercano al inicio del recorrido con el de peso menor más cercano a la raíz

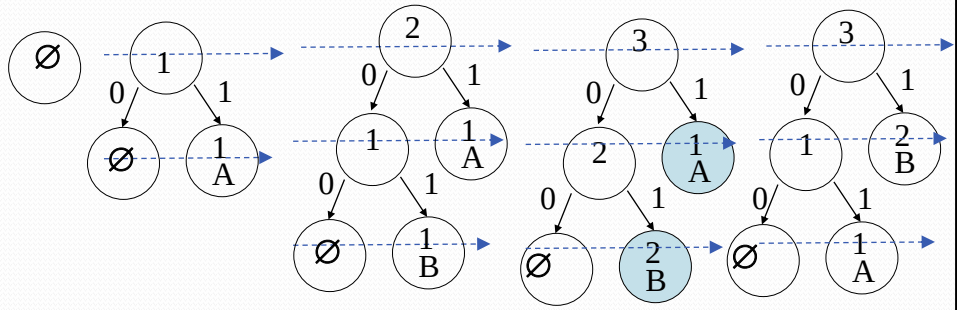
HUFFMAN DINÁMICO – ALGORITMO FGK

El decodificador recibe la salida del codificador, construye el árbol y decodifica siguiendo los mismos pasos que realizó el codificador

In: A
Out: **A**

In: 0 B
Out: **B**

In: 01
Out: **B**



Ø 1 1 | Ø 1 1 1 2 | Ø 2 2 1 3 | Ø 1 1 2 3

No verifica la propiedad de *sibling* (intercambiar nodos)