

## Estructuras de Datos

### Clase 19 – Procesamiento de Texto (Codificación de Huffman)



Dr. Sergio A. Gómez  
<http://cs.uns.edu.ar/~sag>



Departamento de Ciencias e Ingeniería de la Computación  
Universidad Nacional del Sur  
Bahía Blanca, Argentina

## Compresión de datos

- La **compresión de datos** consiste en la reducción del volumen de información tratable (procesar, transmitir o grabar).
- En principio, con la compresión se pretende transportar la misma información, pero empleando menor cantidad de espacio.

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:  
"Estructuras de Datos. Notas de Clase". Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019.

## Tipos de compresión

- **Compresión sin pérdida de información:**
  - los datos antes y después de comprimirlos son exactos en la compresión sin pérdida.
  - En el caso de la compresión sin pérdida una mayor compresión solo implica más tiempo de proceso.
  - Se utiliza principalmente en la compresión de texto.
- **Compresión con pérdida de información:**
  - Puede eliminar datos para reducir aún más el tamaño, con lo que se suele reducir la calidad.
  - Hay que tener en cuenta que una vez realizada la compresión, no se puede obtener la señal original, aunque sí una aproximación cuya semejanza con la original dependerá del tipo de compresión.
  - Se utiliza principalmente en la compresión de imágenes, videos y sonidos. Ej: Formatos jpg, mpeg, mp3

## Compresión de texto (sin pérdida)

- **Problema:** Dado un string  $X$  definido sobre un alfabeto  $\Sigma$  (Ascii o Unicode), deseamos codificar  $X$  en forma eficiente en forma binaria (es decir, obtener un string  $Y$  formado sólo por 0 y 1).
- **Ventajas:**
  - Ahorro de ancho de banda en las comunicaciones
  - Ahorro de espacio en disco en archivos almacenados

## Codificación binaria

- Si el alfabeto  $\Sigma$  tiene  $n$  símbolos se necesitan  $\lceil \log_2(n) \rceil$  bits para codificar cada símbolo
- Ejemplo: Como el alfabeto Ascii tiene 256 símbolos, necesito 8 bits.
- Ejemplo: Si  $\Sigma = \{ a, b, c, d, e, f, g, h \}$ , entonces  $n = 8$   
Como  $\log_2(8) = 3$ , entonces una codificación posible es:  

a = 000	b = 001	c = 010	d = 011
e = 100	f = 101	g = 110	h = 111

La cadena "aaabbccdefad" se codifica sin compresión como  
(los puntos son para leerla más fácil y no se almacenan):  
000.000.000.001.001.010.010.011.100.101.000.011

## Codificación de Huffman: Preliminares

- Objetivo: En lugar de usar un número fijo de bits para codificar cada carácter de un string  $X$ , los caracteres que más se usan se codifican con menos bits y los que menos se usan se codifican con más bits.
- Por cada carácter es necesario conocer su frecuencia de aparición.
- Es decir, por cada carácter  $c$  tendremos  $f(c) =$  cantidad de apariciones de  $c$  en la cadena  $X$  a comprimir
- La concatenación de los bits de la compresión de los caracteres de la cadena forman el string comprimido.
- Código prefijo: Para evitar ambigüedades, no va a haber ningún código que sea prefijo de otro.

## Codificación de Huffman: Idea

- Construye un árbol binario  $T$  que representa la codificación para una cadena  $X$
- Cada nodo, excepto la raíz, representa un bit del código.
- El hijo izquierdo representa un 0 y el derecho un 1
- Cada hoja representa un carácter  $c$
- La codificación de un carácter  $c$  se define como la secuencia de bits determinada por el camino de la raíz a la hoja que contiene a  $c$  en el árbol  $T$ .
- Cada hoja tiene una frecuencia  $f(v)$  correspondiente a la frecuencia del carácter  $c$  almacenado en  $v$
- Cada nodo interno tiene una  $f(v)$  que corresponde a la suma de las frecuencias de los caracteres en dicho subárbol.
- Ejemplo: Figura 12.11 de GT

Estructuras de datos - Dr. Sergio A. Gómez

7

## Ejemplo

Supongamos queremos comprimir la cadena  $X =$   
 $aaaaabbbaaaaacd$  (que tiene longitud 17)

El alfabeto tiene 4 símbolos, luego necesito 2 bits  
 para representar cada uno:

$a=00$ ,  $b=01$ ,  $c=10$ , y  $d=11$  .

La cadena sin comprimir se representa como:

00.00.00.00.00.00.01.01.01.00.00.00.00... etc

Necesito  $17 \cdot 2 = 34$  bits para representarla.

El cálculo de las frecuencias de apariciones da:

$f(a) = 11$ ,  $f(b)=3$ ,  $f(c)=2$  y  $f(d)=1$ .

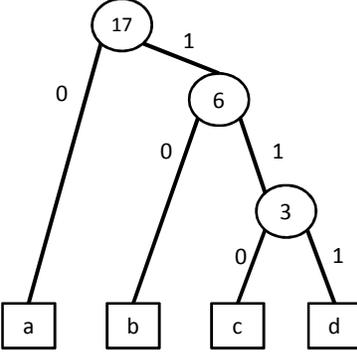
Estructuras de datos - Dr. Sergio A. Gómez

8

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:  
 "Estructuras de Datos. Notas de Clase". Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019.

### Ejemplo (cont)

El algoritmo genera el siguiente código:



a = 0  
b = 10  
c = 110  
d = 111

Por lo tanto, la cadena X =  
aaaaaabbbaaaaccd se  
comprime como:  
00000010101000000110110111,  
la cual mide 26 bits.  
Así tengo una tasa de compresión  
del:

34 bits \_\_\_\_\_ 100%  
26 bits \_\_\_\_\_ x =  
100%\*26bits/34bits = 76%

Estructuras de datos - Dr. Sergio A. Gómez 9

## Algoritmo de Huffman

**Algoritmo** Huffman( X )

**Entrada:** Un string X de longitud n sobre alfabeto de tamaño d

**Salida:** árbol para codificar X

Computar la frecuencia f(c) para cada carácter c de X

Crear una cola con prioridad C

**Para** cada carácter c en X **hacer**

Crear un árbol binario hoja T con rótulo c

Insertar T en Q con prioridad f(c)

**Mientras** Q.size() > 1 **hacer**

$f_1 \leftarrow Q.min().key(); \quad T_1 \leftarrow Q.removeMin()$

$f_2 \leftarrow Q.min().key(); \quad T_2 \leftarrow Q.removeMin()$

Crear un nuevo árbol binario T con hijo izquierdo  $T_1$  e hijo derecho  $T_2$

Insertar T en Q con prioridad  $f_1+f_2$

**Retornar** el árbol Q.removeMin()

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:  
"Estructuras de Datos. Notas de Clase". Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019.

## Aplicando Huffman al idioma castellano

- En GT, Huffman se estudia orientado a comprimir una cadena de texto en particular y generar una compresión óptima para dicha cadena.
- Para aplicarlo a documentos arbitrarios sin necesidad de armar cada vez el árbol de compresión se puede:
  - Determinar la frecuencia de aparición de cada letra/símbolo en el idioma castellano
  - Construir el árbol de compresión de Huffman
  - Utilizar siempre el mismo árbol para comprimir/descomprimir
  - Algunas veces el resultado será malo y otras bueno pero considerando todos los archivos comprimidos en conjunto el resultado será bueno.

Estructuras de datos - Dr. Sergio A. Gómez

11

## Bibliografía

- Capítulo 12, Sección 3 de M. Goodrich & R. Tamassia, Data Structures and Algorithms in Java. Fourth Edition, John Wiley & Sons, 2006.

Estructuras de datos - Dr. Sergio A. Gómez

12

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente: "Estructuras de Datos. Notas de Clase". Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019.