Greedy: Códigos de Huffman

Teoría de Algoritmos I (75.29 / 95.06)

Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Compresión de datos

Reducción del volumen de datos

para representar una determinada información empleando una menor cantidad de espacio

Llamaremos fuente a todo aquello que emita mensajes

Existe un conjunto finito de mensajes posibles

Los mensajes se representaran mediante códigos

El objetivo es elegir códigos de tal forma de reducir al mínimo el tamaño de lo emitido por la fuente

Códigos

Convención que establece como representar cada mensaje utilizando una combinación de símbolos.

Ejemplos: ASCII, Morse, ...

Pueden tener longitud fija o variable

No deben ser ambiguos!

Códigos decodificables

Para cualquier sucesión de códigos, solo existe un único conjunto de mensajes validos.

Los códigos de longitud fija son siempre decodificables

Código 1: ”010” puede ser: “AD” o “B”
Código 2: es decodificable (probar todas las combinaciones!)

Mensaje	Código 1	Código 2
A	0	10
B	010	00
C	01	11
D	10	110

Códigos prefijos

Son siempre decodificables

Un código es prefijo si no existe ningún código que tenga un prefijo igual a otro código completo

Mensaje	Código 2	Código 3
A	10	0
B	00	10
C	11	110
D	110	111

Códigos de Huffman

Presentado en 1952 por David Huffman

- En paper “A method for the construction of minimum redundancy codes”
- http://compression.ru/download/articles/huff/huffman_1952_minimum-redundancy-codes.pdf

Son códigos de longitud variable y prefijos

La longitud de cada código está basada en su frecuencia de cada mensaje en el emisor (fuente)

- Se basa en la teoría de información de Claude Shannon
- Son óptimos: no existen otros códigos prefijos para la misma fuente que la codifique en menor longitud

Códigos de Huffman (cont.)

Sea Alfabeto A = (a_1, a_2, …, a_n)

Llamaremos W=(w_1,w_2,…, w_n) al peso (frecuencia) de cada a_i

Construiremos C(W) = (c_1,c_2,..c_n) códigos prefijos y binarios

Llamamos:

$$\text{Longitud}(C(W)) = \sum_{i=1}^{n} w_i * \text{size}(C_i)$$

De tal forma que:

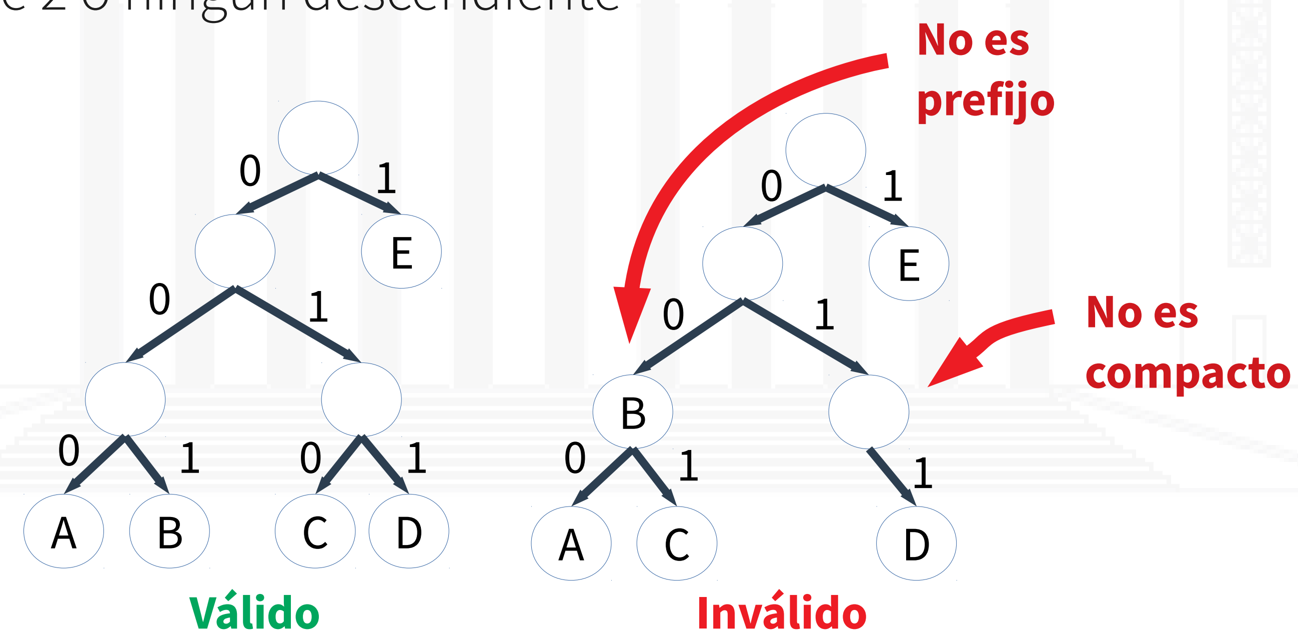
$\text{Longitud}(C(W)) \leqslant \text{Longitud}(T(W))$ para cualquier otro código prefijo T(W)

Código prefijo: Árbol binario

Podemos representar un código prefijo mediante un árbol binario

- Las hojas son los códigos
- Cada nodo tiene 2 o ningún descendiente

A=000
B=001
C=010
D=011
E=1



Algoritmo Greedy

Genera un árbol de "huffman"

Para armar el optimal prefix code

Inicialmente

cada código ci es un nodo hoja con peso wi

Mientras quede más de un nodo sin padre

Toma los dos nodos x, y de menor peso sin padre.

Crear un nuevo nodo z con wz = wx + wy

Definir a z como padre de x e y

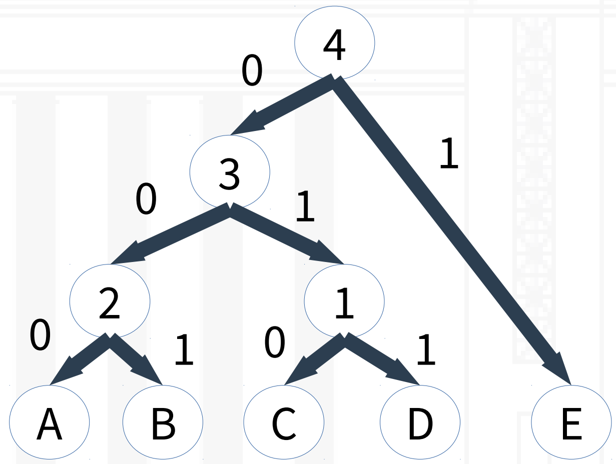
El ultimo nodo sin padre sera la raíz del árbol

Fuente: ABEEECAEEEDBEEEE

A → wa = 2 B → wb = 2

C → wc = 1 D → wd = 1

E-→ we = 10



w1 = wc + wd = 2

w2 = wa + wb = 4

w3 = w1 + w2 = 6

w4 = w3 + we = 16

A=000

B=001

C=010

D=011

E=1

Algoritmo Greedy

$$\text{Longitud}(C(W)) = \sum_{i=1}^{n} w_i * \text{size}(C_i)$$

i	Cod	w	size
A	000	2	3
B	001	2	3
C	010	1	3
D	011	1	3
E	1	10	1

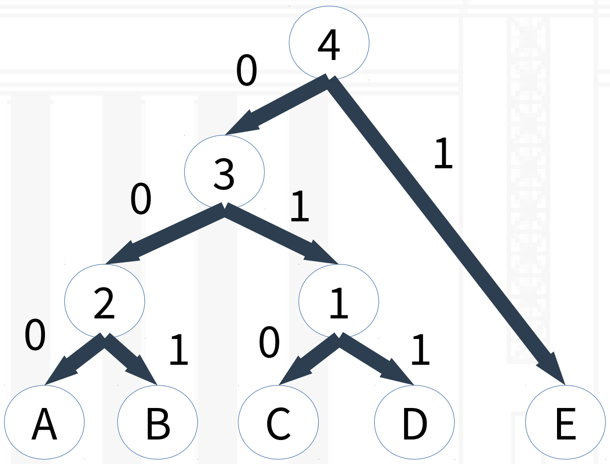
2*3 + 2*3 + 1*3 + 1*3 + 10*1 = 28

Fuente: ABEEECAEEEDBEEEE

A → wa = 2 B → wb = 2

C → wc = 1 D → wd = 1

E--> we = 10



w1 = wc + wd = 2 w3 = w1 + w2 = 6

w2 = wa + wb = 4 w4 = w3 + we = 16

A=000 C=010

B=001 D=011 E=1

Implementación

Utilizaremos un Heap de mínimos

El nodo del árbol será el elemento

La frecuencia será la clave

En cada iteración

Se obtienen los 2 nodos de menor peso

Se ingresará un nuevo creado

El ultimo elemento en le heap

es la raíz del árbol

La complejidad algorítmica es

O(nlogn)

```
Desde i=1 a n
    Crear nodo z
    z.char = a[i]
    z.w = w[i]

    Heap.add(z,z.w)

Desde i=1 a n-1
    x = Heap.get()
    y = Heap.get()

    Crear nodo z
    z.left = x
    z.right = y
    z.w = x.w + y.w

    Heap.add(z,z.w)

Retornar Heap.get()
```

Optimalidad

Para probar que nuestro resultado es optimo debemos realizar 2 demostraciones

Prueba de selección greedy

Lograr mostrar que elegir en los dos mensajes de menor peso nos acerca a la solución optima global

Prueba de los subproblemas

Lograr demostrar que el subproblema derivado de nuestra elección se puede solucionar mediante la misma selección greedy

Selección greedy

Sea

$A=(a_1,a_2,\ldots,a_n)$ alfabeto con pesos $W=(w_1,w_2,\ldots,w_n)$

Sean a y b $\in$ A los 2 mensajes con menor frecuencia de la colección

Existe un código prefijo óptimo tal que

$size(C_a) = Size(C_b)$ y solo difieren en su ultimo bit

Ademas $size(C_a) = size(C_a) \geq size(C_i)$ con $i \in A - \{a,b\}$

Selección greedy - demostración

Sea

$x,y \in A$ siblings en hojas de máxima profundidad en árbol T

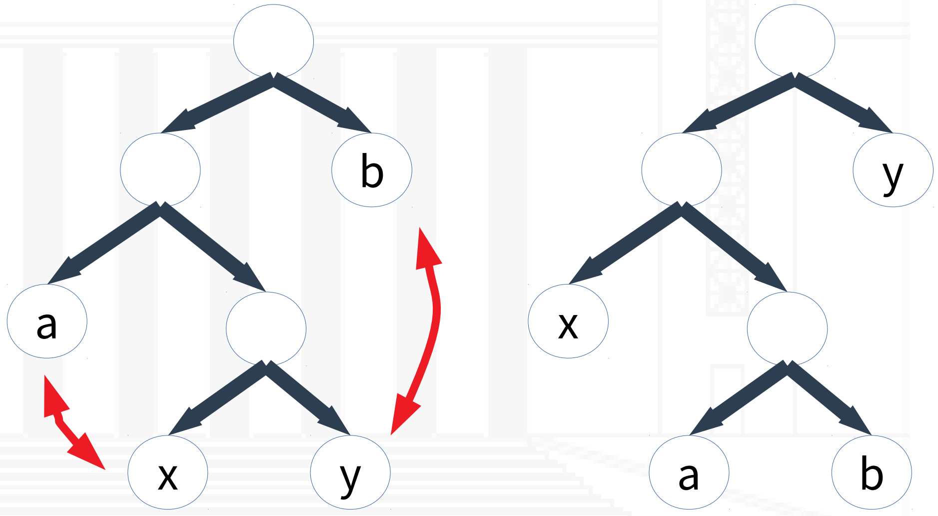
$A,b \in A$ tal que $w_x \geq w_y \geq w_b \geq w_a$

Intercambiamos

a con x

b con y

Llamaremos T' al nuevo árbol



Arbol T

Arbol T'

Selección greedy – demostración (cont.)

Tenemos que

$$\mathrm{L}(C(W))=\sum_{i=1}^{n} w_i * \mathrm{size}(C_i)$$

$$\mathrm{L}(T(W))-\mathrm{L}(T'(W))=$$

$$=w_a*\mathrm{size}(a)+w_b*\mathrm{size}(b)+w_x*\mathrm{size}(x)+w_y*\mathrm{size}(y)$$

$$-w_{a'}*\mathrm{size}(a')+w_{b'}*\mathrm{size}(b')+w_{x'}*\mathrm{size}(x')+w_{y'}*\mathrm{size}(y')$$

$\mathrm{size}(a)=\mathrm{size}(x')$ $\quad$ $\mathrm{size}(a')=\mathrm{size}(x)$

$\mathrm{size}(b)=\mathrm{size}(y')$ $\quad$ $\mathrm{size}(b')=\mathrm{size}(y)$

$\Rightarrow$ $\mathrm{L}(T(W))-\mathrm{L}(T'(W))\geqslant 0$

Disminuye o se mantiene el tamaño de la fuente comprimida

Prueba de los subproblemas

Sea

$A=(a_1,a_2,...,a_n)$ alfabeto con pesos $W=(w_1,w_2,...,w_n)$

Sean a y b $\in$ A los 2 mensajes con menor frecuencia de la colección

$A' = A - \{a,b\} + \{z\}$ tal que $w_z = w_a + w_b$ y el resto de los pesos iguales

Sea T' arbol representando un código prefijo óptimo para C'

Sea T arbol representando un código prefijo óptimo para C

Entonces

T se puede obtener de T' reemplazando el nodo hoja z por un nodo con hijos a y b

Prueba de los subproblemas - demostración

Por un lado podemos ver que

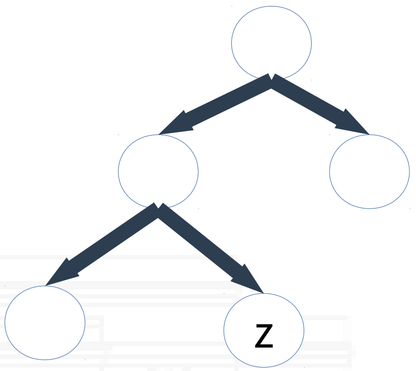
Todos los pesos y longitudes son iguales excepto a,b y z

$$\text{size}(a)=\text{size}(b)=\text{size}(z)+1$$

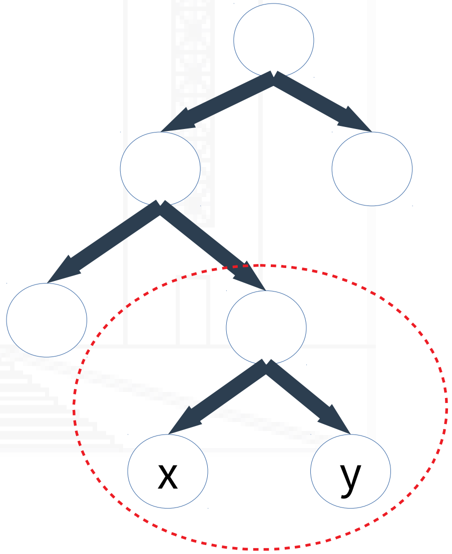
$$w_a*\text{size}(a)+w_b*\text{size}(b)=(w_a+w_b)*(\text{size}(z)+1)$$

$$=(w_z)*\text{size}(z)+(w_a+w_b)$$

$$L(T')=L(T)-(w_a+w_b)$$



Arbol T’



Arbol T

Prueba de los subproblemas - demostración (cont.)

Supongamos que T no representa un código optimo prefijo de A

Entonces existe un T'' optimo tal que L(T'') < L(T)

T'' tiene que tener a y b como siblings (por demostración anterior)

Sea T''' el árbol T''con el padre de a y b reemplazado por la hoja z

Entonces

$$L(T''') = L(T'') - wa - wb$$
$$< L(T) - wa - wb = L(T')$$

Es una contradicción

Si T' es un código optimo, entonces debe tener la misma longitud de T'''

.UBA fiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Abril de 2020