Estructuras de Datos Compactas y Algoritmos Avanzados

Curso obligatorio, 6 ECTS, 2º cuatrimestre

Curso 2010/2011

Profesores:

Antonio Fariña Martínez	2 ECTS	20 h
Juan Ramón López Rodríguez	0,75 ECTS	7 h
José Ramón Paramá Gabía	0,25 ECTS	3 h
Nieves Rodríguez Brisaboa	0,75 ECTS	8 h
Miguel Rodríguez Penabad	0,75 ECTS	7 h
Ángeles Saavedra Places	1,5 ECTS	15 h

Bibliografía básica:

- R. Baeza-Yates y B. Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley Longman, 1999.
- G. Navarro y M Raffinot. Flexible Pattern Matching in Strings. Cambridge University Press.
- T. C. Bell, J. G. Clearly y I. H. Witten. Text Compression. Prentice Hall, 1990.
- I.H. Witten, A. Moffat, T.C. Bell. Managing Gigabytes. Compressing and Indexing Documents and Images (2nd ed.), Morgan Kaufmann Pub, 1999

- **Descripción:**
  - Descriptores: análisis de algoritmos y complejidad, estructuras de datos y algoritmos para gestión eficiente en memoria y/o disco, aplicación a transmisión de datos o acceso Web, técnicas de compresión de textos e imágenes estáticas y dinámicas, búsqueda de patrones (en texto, ADN, proteínas, etc), índices y autoíndices, métodos de indexación en espacios métricos.

Programa. Aspectos a revisar:

- Introducción á compresión
- Compresión dinámica
- Compresión dinámica orientada a palabras

- Indexación/autoindexación
- BWT e aplicacións
- Arrays de Sufixos
- CSA e psi.

Avaliación

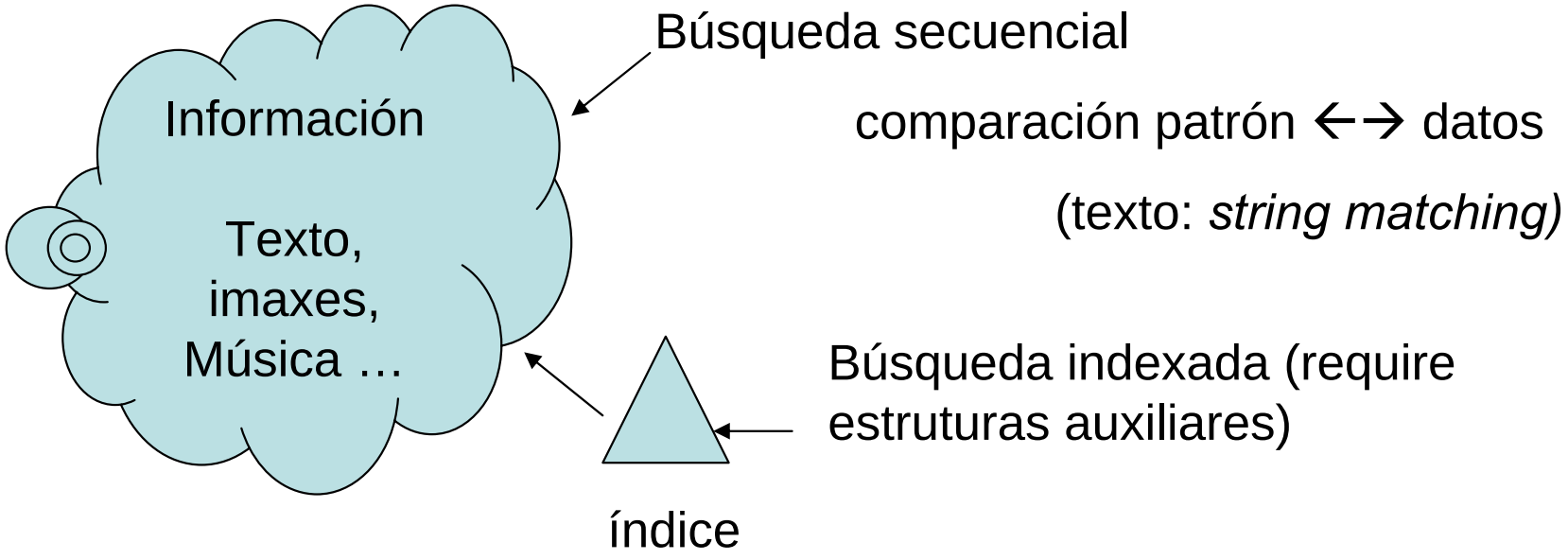
- Exercicios feitos na clase
  - Participación.
  - “Deberes”
- Traballo/s final/is: boletín.
- Revisión artigos/exposición oral.
- Proba escrita

Sesión 1: Compresión

- Introducción
- Conceptos
- Clasificación de técnicas
- Compresión estatística: Huffman
- String matching
  - Horspool, Set-Horspool (tries)

Onde estamos

- Nos últimos anos …
  - Temos “demasiada” información
    - Multimedia (imaxes, vídeo, son,…)
    - Texto: BD textuais (xurisprudencia, datos corporativos,…), secuencias biolóxicas



¿Comprimir ou non comprimir?… esa é a cuestión

Introdución

- Texto nos últimos anos …
  - Incremento número colección texto e o seu tamaño
    - **Web**, BD textuais (xurisprudencia, datos corporativos,…)
- Compresión reduce:
  - Tamaño necesario para almacenamento
  - Tempo de acceso a disco
  - Tempo de transmisión
- Aplicacións
  - Recuperación (retrieval) eficiente. Buscas son máis rápidas que en texto non comprimido
  - Trasmisión de datos (de xeito comprimido).
  - …

Introdución

- Importancia da compresión en BD Textuais
  - Reduce o seu tamaño: 30% ratio compresión
  - Reduce tempo de acceso a disco e de transmisión
  - É posible **buscar** o texto comprimido sen descomprimir
    - Buscas ata 8 veces máis rápido.
  - Descompresión necesaria só para **presentar** resultados
  - A Compresión pode **integrarse** nos Sistemas de Recuperación de texto, mellorando o seu rendemento.
  - ...
- Propiedades desexables básicas
  - Búsquedas directas e descompresión aleatoria

Introducción

- Teño un texto… ¿Como podo comprimilo?
  - Seleccionar **símbolos** do texto (vocabulario) e substituilos por uns **códigos** (creados sobre un alfabeto de saída)

Compresión = Modelado + codificación

  - Debe haber un xeito de coñecer a correspondencia entre os símbolos detectados no texto, e os códigos que se lles asociaron para permitir o proceso de **descompresión**.

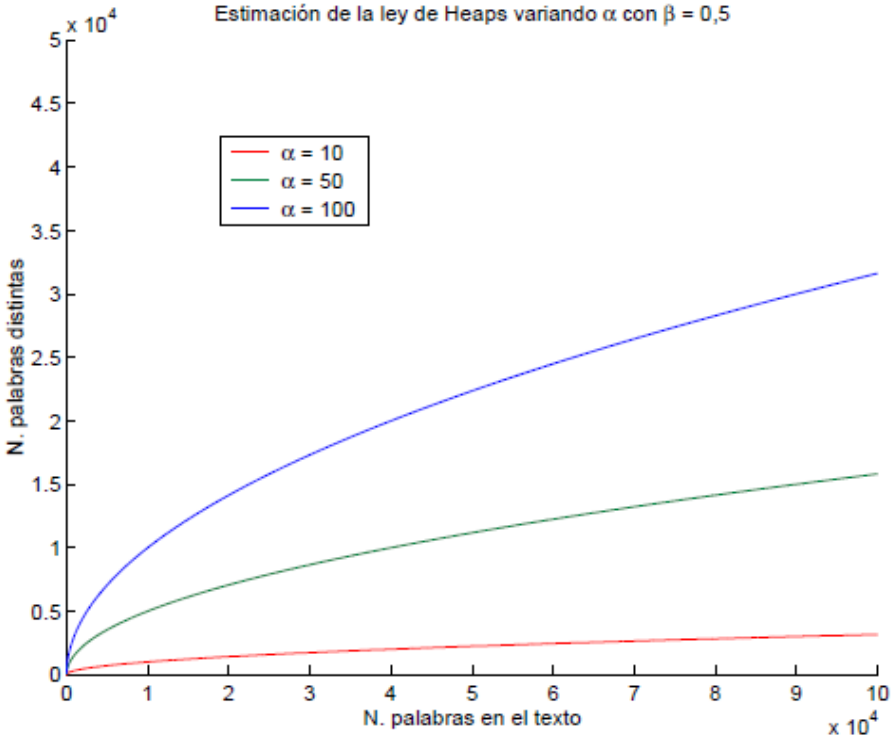
Introducción. Alfabeto de entrada

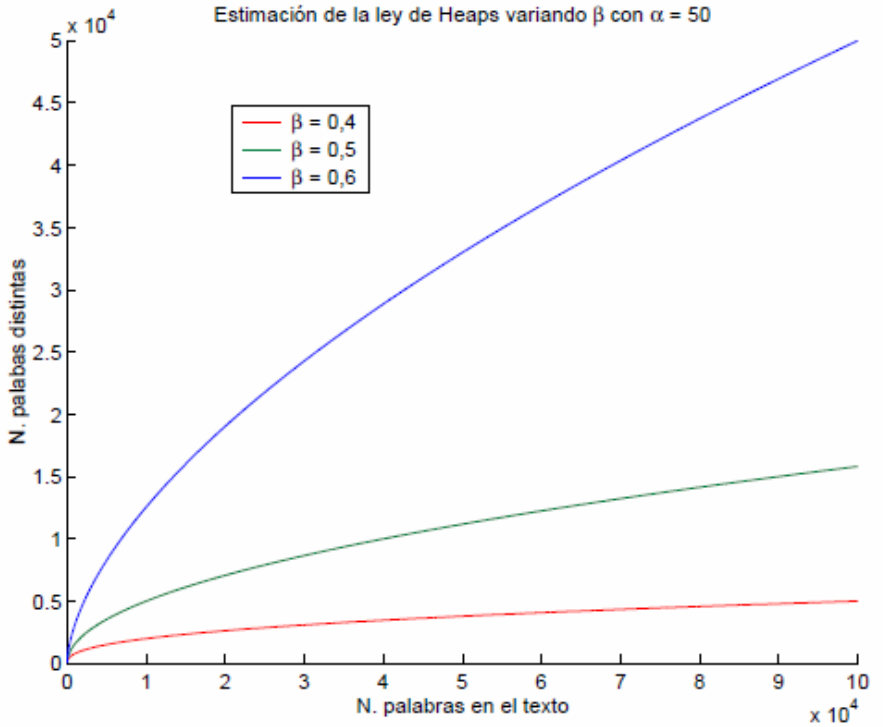
- Técnicas baseadas en caracteres Vs baseadas en palabras.
  - O alfabeto o vocabulario de entrada é distinto:
    - Con caracteres:
      - Vocabulario: ata 256 caracteres distintos.
      - O texto ten millóns de símbolos (|T|)
    - Con palabras
      - Vocabulario: Miles de palabras distintas.
      - Ao usar palabras: o número de símbolos a tratar é menor. (aprox |T| / 4).
  - ¿Pode ser bo utilizar palabras en lugar de caracteres?
    - Si porque as palabras son a base do procesado dos textos en recuperación de textos e en indexación.
    - Ademais as Leis de Heaps e Zipf amosan que:
      - o vocabulario non medrará en exceso, e que
      - a distribución de frecuencias será sesgada

Introdución. Linguaxe Natural: Leis

- Lei de Heaps

Heaps' law establishes that the relationship between the number of words in a natural language text (N) and the number of different words (n) in that text (that is, words in the vocabulary) is given by the expression $n = \alpha N^{\beta}$, where α and β are free parameters empirically determined. In English text corpora, it typically holds that $10 \leq \alpha \leq 100$ and $0.4 \leq \beta \leq 0.6$.





Introdución. Linguaxe Natural: Leis

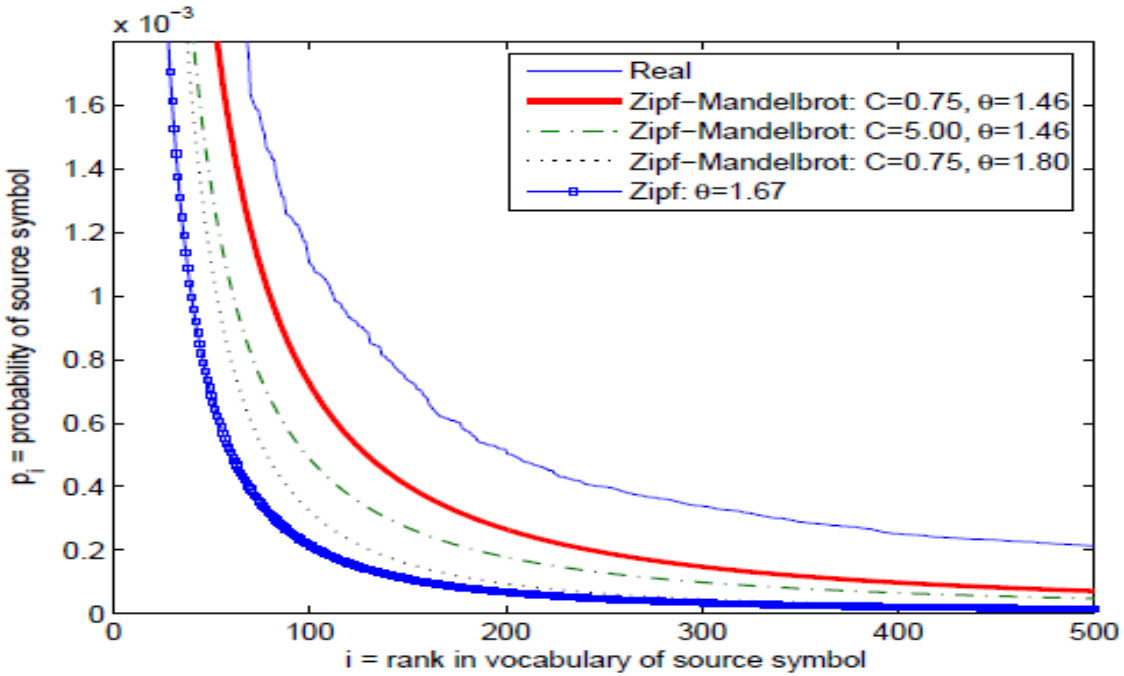
- Lei de Zipf

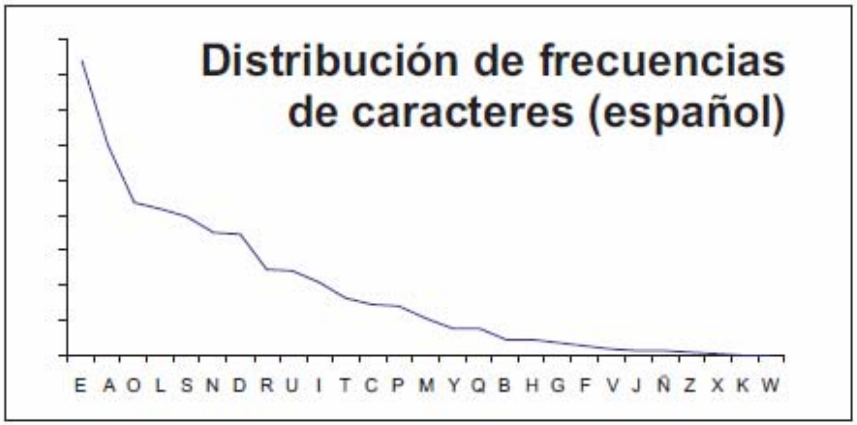
Zipf's Law [Zip49] gives a good estimation for the word frequency distribution in natural language texts. It is well-known [BCW90] that, in natural language, the probability of occurrence of words in a vocabulary closely follows Zipf's Law. That is:

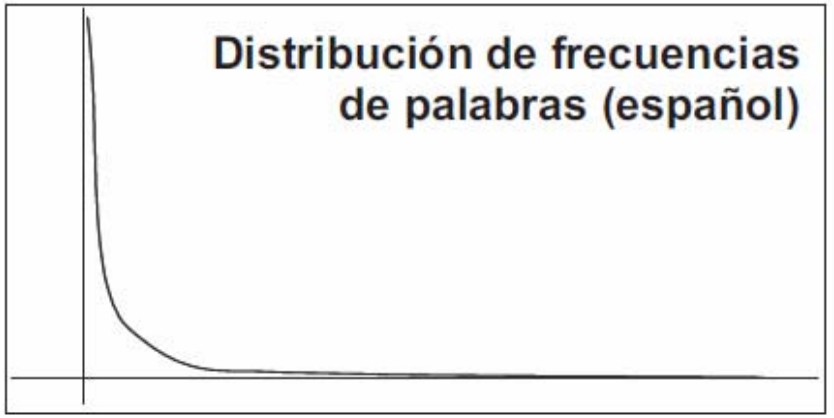
$$p_i = \frac{A_z}{i^\theta}$$

Where i is the rank of a word in the vocabulary ($i = 1 \,..\, n$), θ is a constant that depends on the analyzed text ($1 < \theta < 2$), and $A_z = \frac{1}{\sum_{i>0} 1/i^\theta} = \frac{1}{\zeta(\theta)}$ is a normalization factor[1].

Introdución. Linguaxe Natural: Leis







- Exercicio:
  - Texto de 1Gb
  - Usando palabras (lonxitude media = 5 chars)
  - Obtivemos empíricamente α= 50 β= 0,5
  - Cantas palabras diferentes se obtiveron durante o proceso de compresión?

Introducción. Alfabeto de saída

- Técnicas orientadas a byte Vs orientadas a bit.
  - O texto comprimido estará consituido por secuencias de:
    - Bits: 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 1
    - Bytes: 250 128 10 255 128

- En xeral:
  - Utilizar bits mellora a compresión (pois pode haber códigos de moi poucos bits
  - Utilizar bytes:
    - Incrementa o tamaño do texto comprimido (o código máis pequeno que se lle pode asignar a un símbolo a comprimir é de 1 byte)
    - Fai máis eficiente o procesado do texto

Clasificación dependendo do alfabeto entrada / saída

- O compresor pode elexir como alfabeto de entrada
  - Un número fixo de símbolos (*compresores estatísticos*):
    - Exemplo: 1 carácter, 1 palabra,
  - Un número variable de símbolos (*compresores de diccionario*):
    - Exemplo: A primeira vez que aparece un carácter **a** comprímese el so, mais a segunda vez que aparece codifícase xunto co seguinte carácter **ab**

- Os códigos crearanse sobre símbolos dun alfabeto de saída:
  - Códigos de lonxitude fixa (1bit, 10 bits, 1 byte, 2 bytes,…)
  - Códigos de lonxitude variable (algúns de 1 bit/byte, outros de 2, etc)

- CLASIFICACIÓN: (fixed-to-variable, variable-to-fixed, var-to-var)

Alfabeto entrada	Alfabeto saída: fixo	Alfabeto saída: var
fixo	--	estatísticos
var	diccionario	híbridas

Introducción: 2 principais familias

- **Técnicas baseadas en diccionario (Lempel-Ziv, gzip, compress,…)**
  - Constrúen un diccionario que contén substrings de ***lonxitude variable*** do texto
  - COMPRESIÓN: Cada substring do texto a comprimir é reemplazado por un **código de lonxitude-fixa** (que normalmente depende da súa posición no diccionario).
  - A asociación ***símbolo ←→-código*** varia ao longo do texto

- **Compresores estatísticos (Baseados en Huffman, aritméticos, PPM,…)**
  - Calculan a frecuencia dos símbolos de entrada
    - Un símbolo de entrada é 1 carácter, ou 2 (bigramas), ou 3 (trigramas) etc, ou un símbolo de entrada é una palabra
  - COMPRESIÓN: Asocian un código de lonxitude-variable a cada símbolo de entrada máis curto (lonxitude variable) aos símbolos máis frecuentes
    - Os símbolos que aparecen moitas veces reemplázanse por códigos moi curtos, o que implica que os símbolos moi pouco frecuentes se codifiquen mediante códigos moi largos.

Introducción. Compresores dinámicos Vs estáticos

- Clasificación dependente de como se realiza o proceso de modelado (detectar os símbolos e obter información sobre os mesmos).

  - **Compresión <u>estática</u>**: Un símbolo sempre recibe o mesmo código (p. ex: morse)

  - **Compresión en 2 pasadas ou <u>semi-estática</u>**: Realízase unha primeira pasada para obter información do texto, que permita decidir que código se lle asignan a cada símbolo. Finalmente comprímese o texto. (p.ex: Huffman clásico)

  - **Compresión en 1 pasada ou <u>dinámica</u>**: Vaise procesando o texto símbolo a símbolo, e a medida que o proceso de compresión avanza, axústase o esquema de codificación. Por iso, os códigos asignados a un mesmo símbolo poden variar en distintos momentos.

    (Ex: compresores de diccionario, ziv-Lempel, Huffman dinámico, …)

Exercicio

- Dado o texto: ABABACADAE
- ¿Cal sería a probabilidade coa que se codificaría o 7º símbolo de usar:?
  - Un modelo semistático
  - Un modelo dinámico.

Compresión estatística: Conceptos de Tª da información

- Os compresores estatísticos típicos (p.ex Huffman) son compresores de *orden 0* *(Asumen que os símbolos son independentes entre si) e...*
  - *Tratan de manter a **información** existente no texto (mensaxe) e reducir a **redundancia** do mesmo.*

- Para entender isto primeiro imos falar da teoría da información de Shannon:

  - Cómo medir a información?
    - A cantidade de información é equivalente á cantidade de *sorpresa* na mensaxe.
    - Se che conto algo que xa sabes (“es un alumno do máster”) non che dou información.
    - Se che digo (“isto non entra no exame”), douche máis información, independentemente do número de palabras que utilicei.

Compresión estatística: Conceptos de Tª da información

- Supoñamos que tiramos unha moeda ao aire e queremos saber que saiu...
  - O resultado é **cara ou cruz**, chega 1 bit para representalo.
  - ¿Que saiu? Cara → 0 | Cruz →1
- Isto pode xeneralizarse a moitos problemas da vida real por medio de utilizar máis bits.
  - A cuestión é atopar o número mínimo de si/non necesarios para chegar á resposta.

Compresión estatística: Conceptos de Tª da información

- Supoñamos unha baralla de 64 cartas, numeradas do 1 ao 64.
  - Supoñamos que unha persoa *A* saca unha carta do mazo, e outra *B* ten que adiviñar qué carta é (un número do 1 ao 64)
  - Cántas preguntas “si/non” temos que formular?

    - B pode dividir o intervalo 1-64 en dous, e comeza preguntando se a carta está no intervalo 1-32?
      - Se a resposta é non, está no intervalo 33-64.
    - Agora o intervalo divídese outra vez, e a pregunta sería está no intervalo 33-48?
    - O proceso continúa ata que o intervalo quede reducido a unha carta (número).
    - Polo tanto serían necesarias 6 preguntas, posto que é o número de veces que 64 pode ser divido á metade, ou.

      $$\mathbf{6=\log_2 64}$$

Compresión estatística: Conceptos de Tª da información

- Outra forma de ver o problema é:
  - Dado un número *N,* cantos díxitos son necesarios para expresalo?
    - $\log_{10} N$
    - $\log_2 N$ (se traballamos en base 2)

- Acórdase que o número de bits necesarios para expresar *N* é proporcional á información de *N.*

- Canta información hai nun número *N* de *k* díxitos (en base 10)?
  - Se asumimos que son *x* bits: $10^k-1=2^x-1$
    - 10^k-1 é o máximo número representable con k díxitos
    - 2^x-1 o equivalente en binario

  - En xeral, dado un número de *k* díxitos en base *n,*

    N^k *(en base n)*$=2^x$*(en binario)* ➔ $\log_2 n^k = x$ ➔ $x = k \log_2 n$

    $x=k \log_2 n$ ==número de bits necesario para representar ese número.

Compresión estatística: Conceptos de Tª da información

- En xeral, dado un número N de *k* díxitos en base *n,* $x=k \log_2 n$

  - Expresa que a cantidade de información incluída nun díxito en base *n* é igual á incluída en $\boldsymbol{\log_2 n}$ bits.
  - Podemos pensar que unha mensaxe está formada por símbolos, supoñamos dun alfabeto de *n* símbolos. Podemos pensar en cada un deles como un díxito en base *n.*
  - Se enviamos por un canal ***s*** símbolos (díxitos) por segundo:
    - $H=s \log_2 n$
    - **H:** denota a cantidade de información (medida en bits) enviada por segundo.

Compresión estatística: Conceptos de Tª da información

- Se o símbolo a_i ocorre con probabilidade P_i, *entón ocorre* sP_i veces por unidade de tempo.
  - *Teñamos en conta que:* $P_1+P_2+\ldots+P_n=1$ (probabilidade)

- Se todos os P_i son iguais:
  - $P_i=P$
  - $1 = \sum P_i = nP$
  - $n=1/P$

- Voltando a probabilidades distintas,
  - a_i contribúe a H con $sP_i \log_2(1/P_i) = -sP_i \log_2 P_i$
  - $H = -s \sum P_i \log_2 P_i$

- Como H é a cantidade de información enviada por segundo. A información contida por un símbolo en base-n é H/s ou $-\sum P_i \log_2 P_i$

Compresión estatística: Conceptos de Tª da información

- H/s= $H = -\sum P_i \log_2 P_i$ é a ***entropía*** dos datos enviados pola canle e para a **distribución de frecuencias** dada.
  - É o **límite inferior** da compresión
    - se consideramos que a probabilidade de ocorrencia dun símbolo é independente dos símbolos anteriores. (entropía de orde 0 ou H_0)
    - $H_0, H_1, H_2, \ldots H_k$

- Un esquema de codificación (ou simplemente código):
  - Asignará un número dado de símbolos de saída a cada símbolo de entrada (s_i), tratando de reducir a lonxitude media dos códigos xerados. LMC = $\sum P_i \mid C_i \mid$
  - Debe tratar de minimizar a redundancia:

    R = $\sum P_i \mid C_i \mid - H$ = LMC - H = $\sum P_i \mid C_i \mid - \sum -P_i \log_2 P_i$

Exercicio

- Exercicio:

  - Dado o texto: ABCABADABABCDEABACAD
  - Un esquema de codificación que asigna os códigos:

    A → 0

    B → 10

    C → 110

    D → 1110

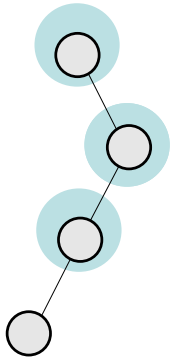
    E → 11110
  - Calcúlese:
    - A lonxitude media dos códigos resultantes
    - O valor da entropía "H" de orde cero.
    - A redundancia da codificación
  - Se a redundancia dunha codificación A é maior que a doutra B Cal delas é mellor?

- Codificacións/códigos:
  - Decodificables unívocamente.
  - Libres de prefixo
  - Libres de prefixo de redundancia mínima

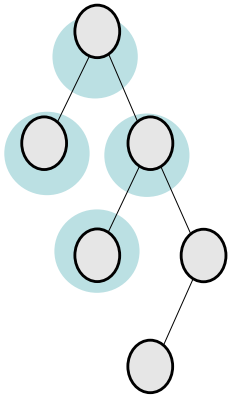
Códigos libres de prefixo

- Códigos libres de prefixo
  - Un código nunca é prefixo de outro máis longo.
  - Ao ler un código xa pode decodificarse sen necesidade de ler máis símbolos. → decodificación máis rápida

- Ejemplo: Compresión do texto: ABCABAC



Código decodificable unívocamente.
Non libre de prefixo



Código **libre de prefixo ou código instantáneo**

Momento tras o que se consigue recoñecer un símbolo descomprimido

1 10 100 1 10 1 10 Texto comprimido

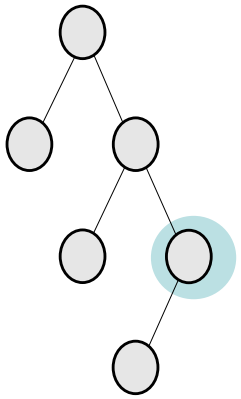
A B C A B A B

0 10 110 0 10 0 110

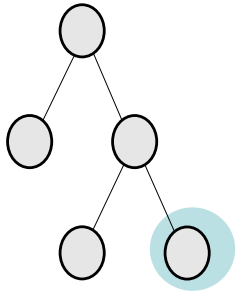
A B C A B A C

Códigos libres de prefixo de redundancia mínima

- Códigos libres de prefixo <u>mínimos</u> ou de <u>redundancia mínima</u>
  - Dado un prefixo propio **x** dun código dado, entón **x<u>a</u>** debe ser:
    - un código ou
    - un prefixo propio dun código, para calquera símbolo **<u>a</u>**, do alfabeto de saída



Código **libre de prefixo <u>non mínimo</u>**



Código **libre de prefijo non mínimo**

Texto comprimido **0 10 110 0 10 0 110**

Texto comprimido **0 10 11 0 10 0 11**

Texto orixinal: ABCABAC

Exercicio

- Dada a codificación:

  A→ 0

  B→10

  C→110

  D→1110

  E→1111

  F→11111

- ¿É un código libre de prefixo de redundancia mínima?

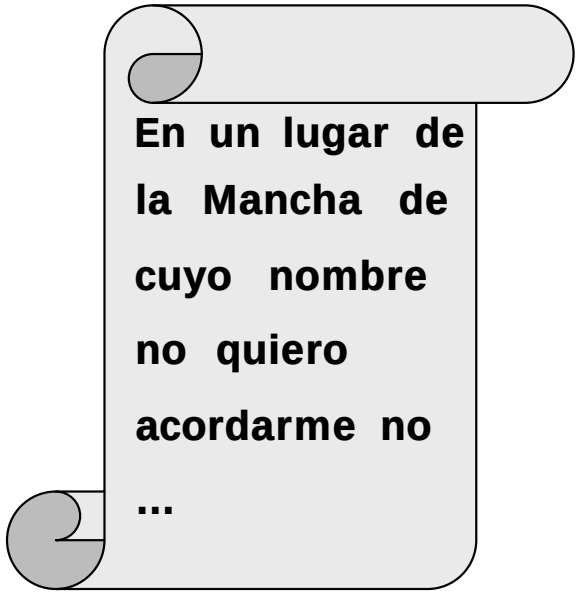
- Compresión estatística semistática
  - Huffman

Compresión estatística semistática

- 2 pasos:
  - Calcular las frecuencias dos símbolos de entrada e codificar
  - Compresión (substitución símbolo → código)

Compresión estatística semistática

1er paso: Procesar texto > Ordenar vocabulario > Generar códigos



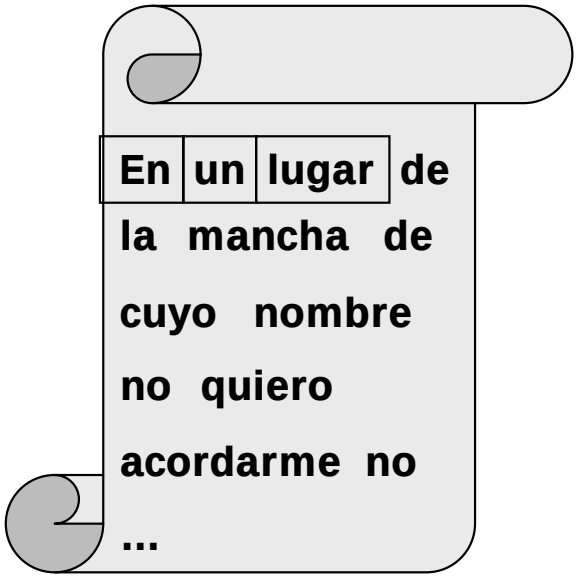
Texto

vocabulario		códigos
de	2	C_1
no	2	C_2
En	1	C_3
un	1	C_4
lugar	1	C_5
la	1	C_6
Mancha	1	C_7
cuyo	1	C_8
nombre	1	C_9
quiero	1	C_{10}
acordarme	1	C_{11}

Símbolo = palabra

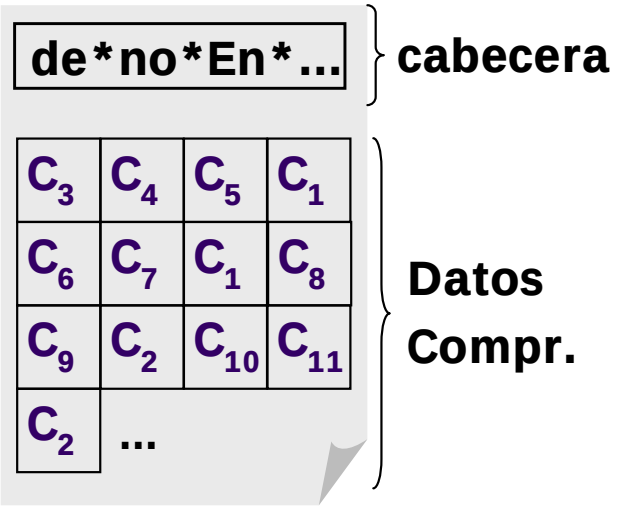
Compresión estatística semistática

2º paso: Substitución palabra ← código



Texto

vocabulario		código
de	2	C_1
no	2	C_2
En	1	C_3
un	1	C_4
lugar	1	C_5
la	1	C_6
Mancha	1	C_7
cuyo	1	C_8
nombre	1	C_9
quiero	1	C_{10}
acordarme	1	C_{11}



Fichero de salida

Compresión estatística semistática

- Compresión estadística
- 2 pasos:
  - Calcular as frecuencias das palabras e codificar
  - Compresión (substitución palabra → código)
- Asociación entre símbolo de entrada ←→ código non cambia ao longo do texto
  - A búsqueda directa é posible → permite Text retrieval
- Método máis representativo: **Huffman**.

- Codificación Huffman orientada a caracteres

D. A. Huffman. A method for the construction of minimum-redundancy codes. In *Proc. Inst. Radio Eng.*, pages 1098–1101, September 1952. Published as Proc. Inst. Radio Eng., volume 40, number 9.

Huffman

- ***Método Huffman* clásico**
  - Código libre de prefixo ***óptimo*** (i.e., menor lonxitude total)
  - Baseado en caracteres (orixinalmente)
    - Os caracteres son os símbolos a codificar
  - Orientado a bits: Os códigos son secuencias de bits
  - Constrúese unha árbore de Huffman para xerar os códigos

Árbore de Huffman

A Huffman tree is built through the following process:

1. A set of nodes is created, one node for each distinct source symbol. Each leaf node stores a source symbol and its frequency.
2. The two least frequent nodes X and Y are taken out of the set of nodes.
3. A new internal node P is added to the set of nodes. P is set as the parent of the nodes X and Y in the tree, and its frequency is computed as the sum of the frequencies of those two nodes.
4. Steps 2) and 3) are repeated while two or more nodes remain in the set of nodes. When the whole process finishes, the unique node that remains in the set is the root of the Huffman tree (and its frequency is the sum of occurrences of all the source symbols).

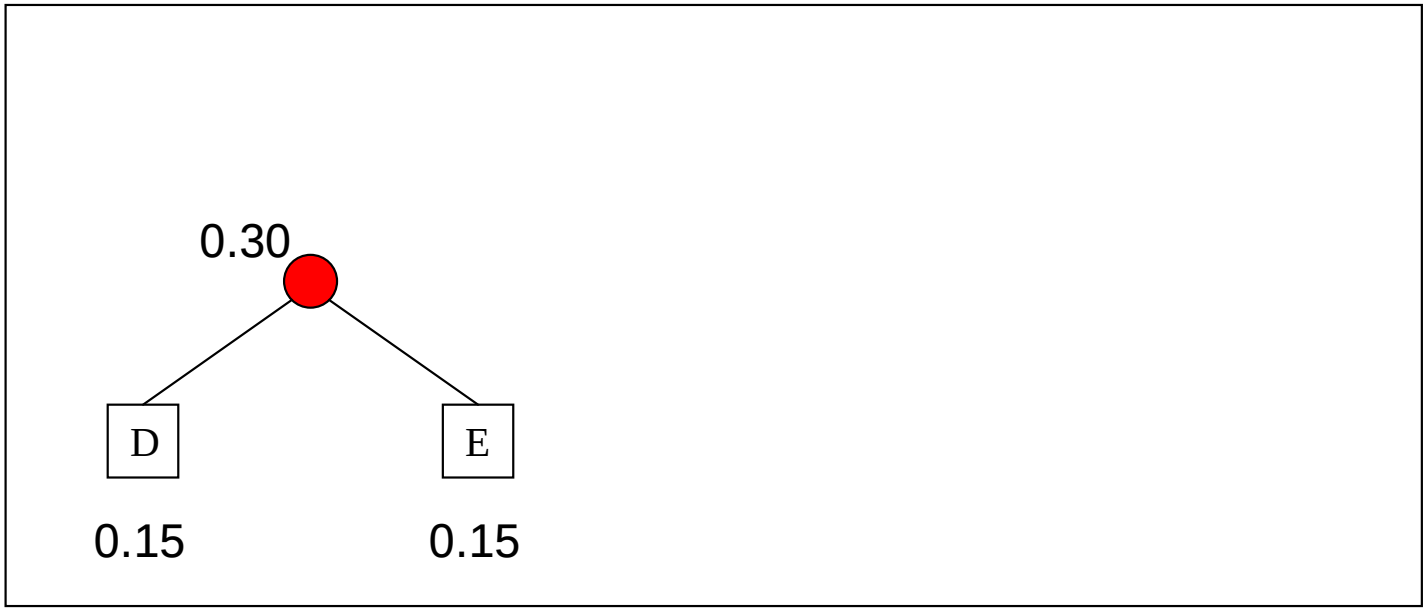
Huffman: construcción

- Ordeación de símbolos por frecuencia

Símbolos	**A**	**B**	**C**	**D**	**E**
Frecuencia / n	0.25	0.25	0.20	0.15	0.15
Código					

Huffman: construcción

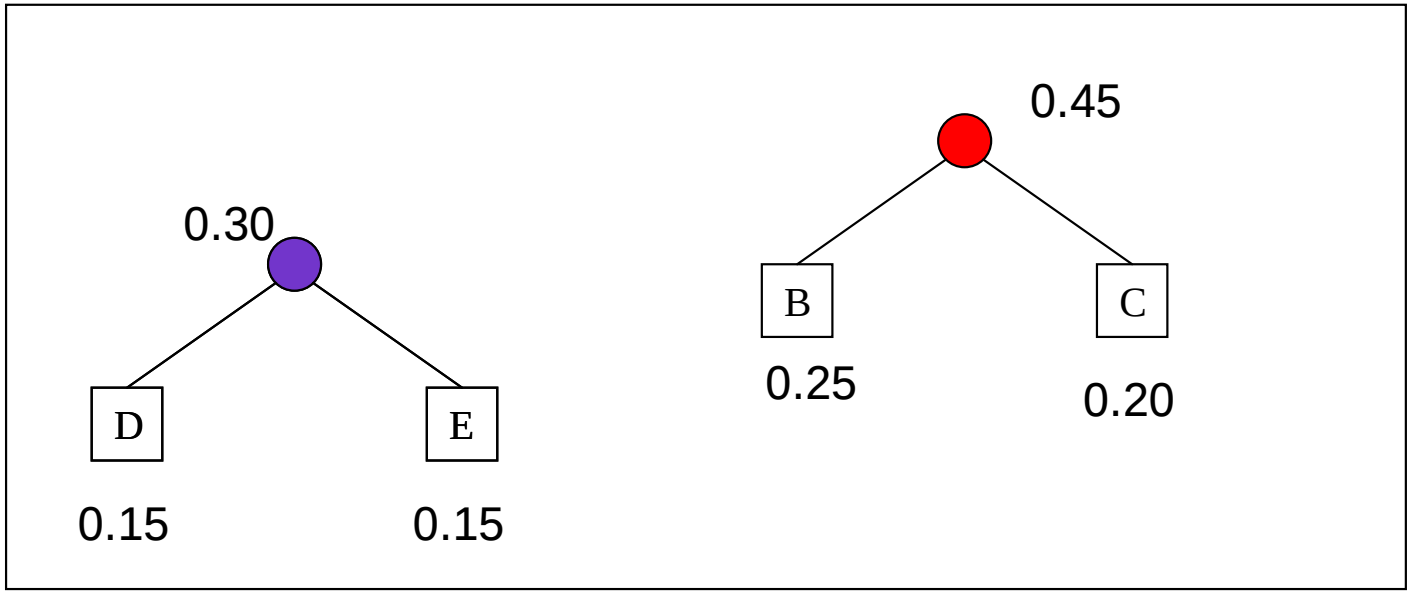
- Construcción de abaixo → arriba



Símbolos	A	B	C	D	E
Frecuencia / n	0.25	0.25	0.20	0.15	0.15
Código					

Huffman: construcción

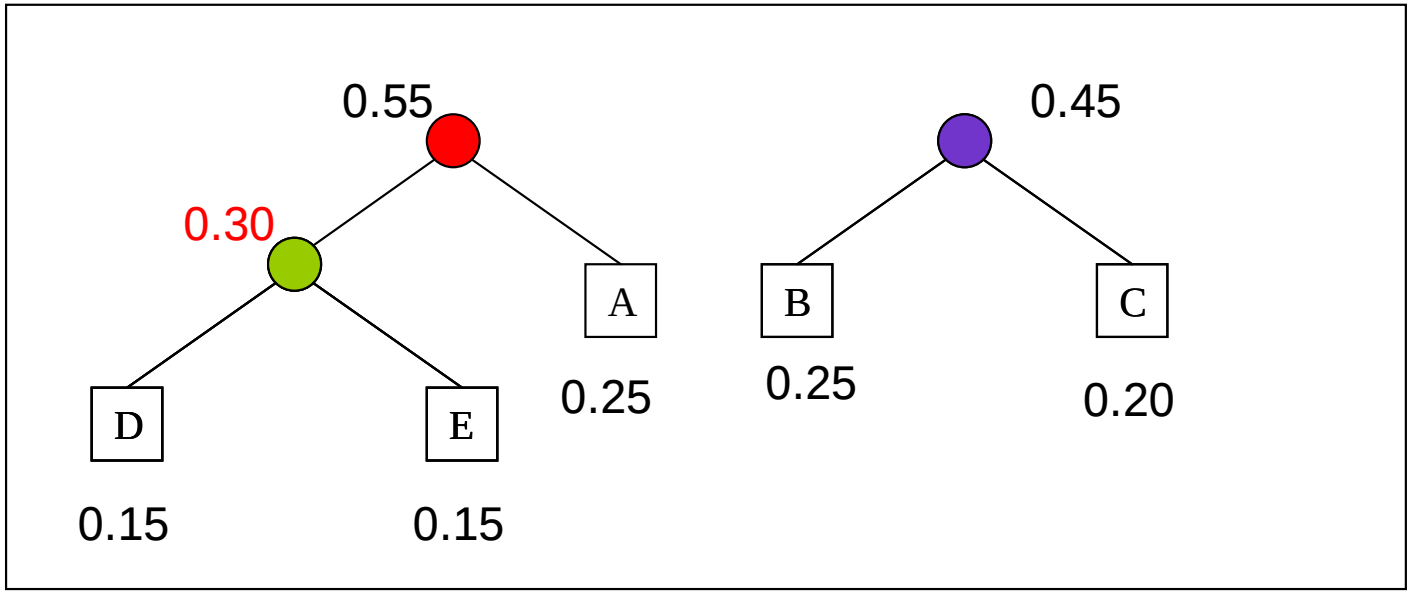
- Construcción de abaixo → arriba



Símbolos	A	B	C	D	E
Frecuencia / n	0.25	0.25	0.20	0.15	0.15
Código					

Huffman: construcción

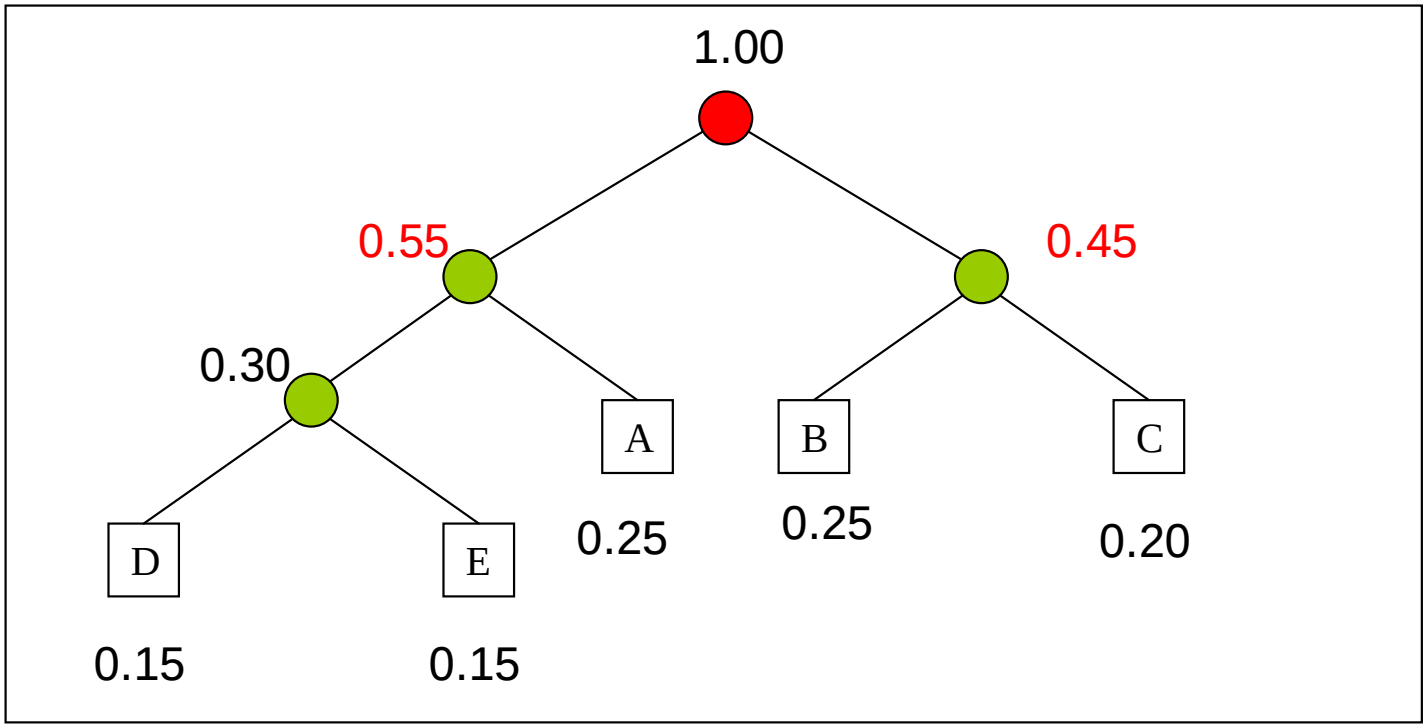
- Construcción de abaixo → arriba



Símbolos	A	B	C	D	E
Frecuencia / n	0.25	0.25	0.20	0.15	0.15
Código					

Huffman: construcción

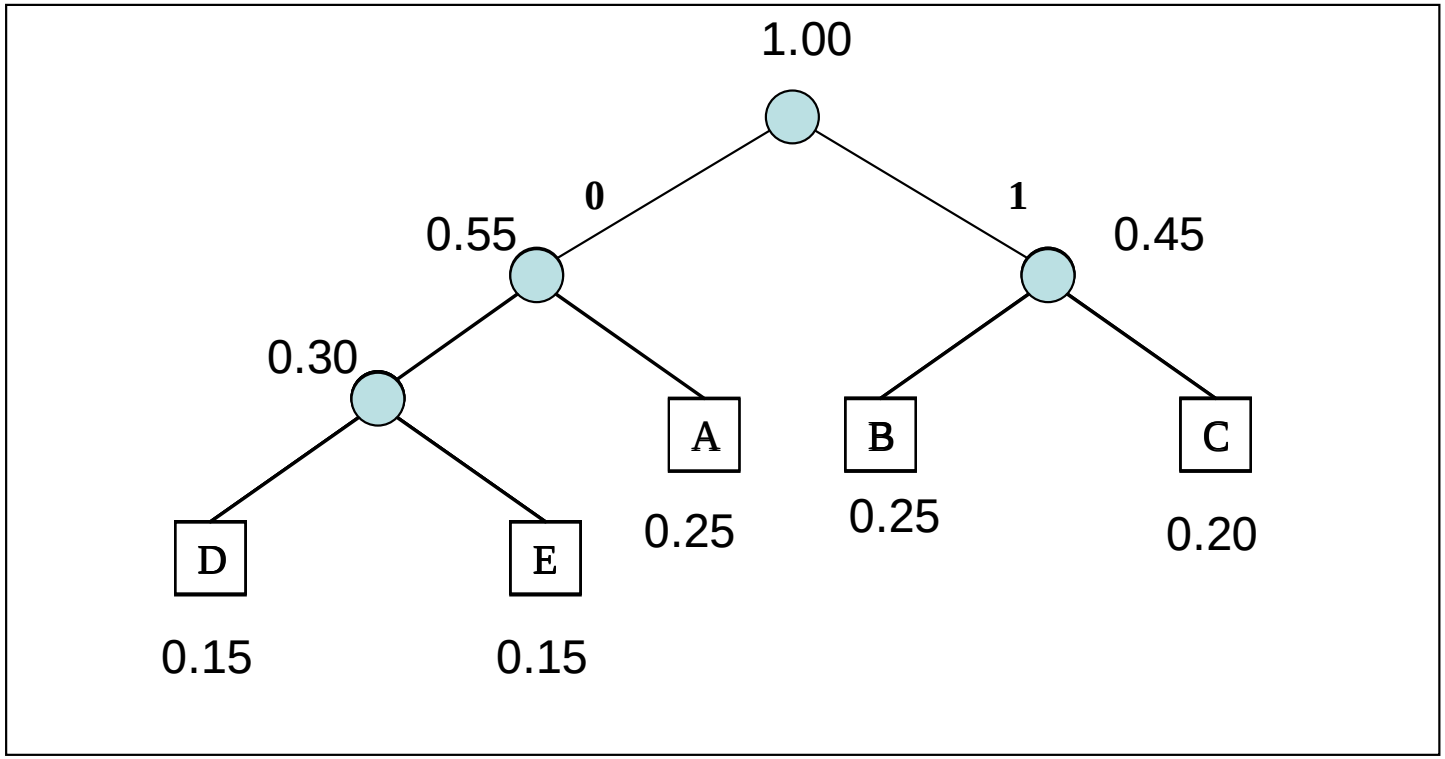
- Construcción de abaixo → arriba



Símbolos	A	B	C	D	E
Frecuencia / n	0.25	0.25	0.20	0.15	0.15
Código					

Huffman: construcción

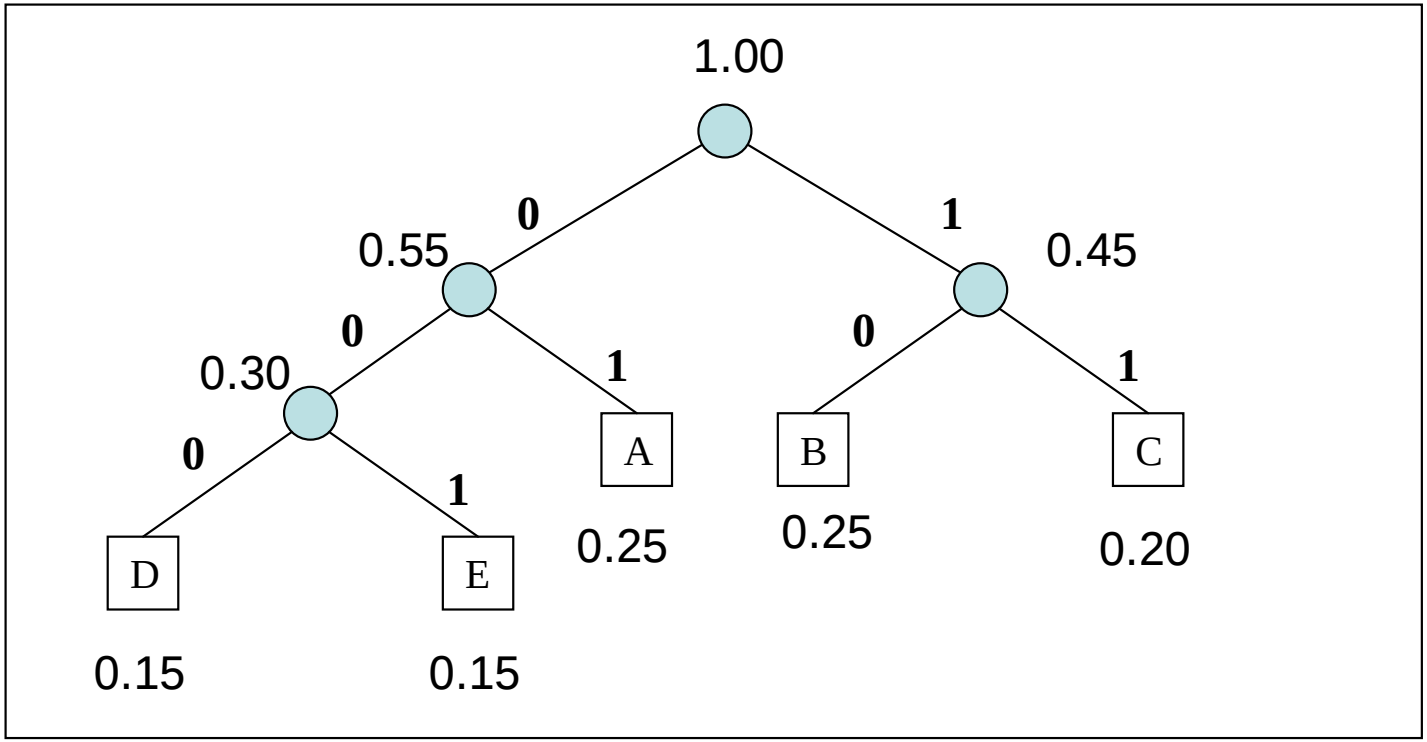
- Etiquetar ramas



Símbolos	A	B	C	D	E
Frecuencia / n	0.25	0.25	0.20	0.15	0.15
Código					

Huffman: construcción

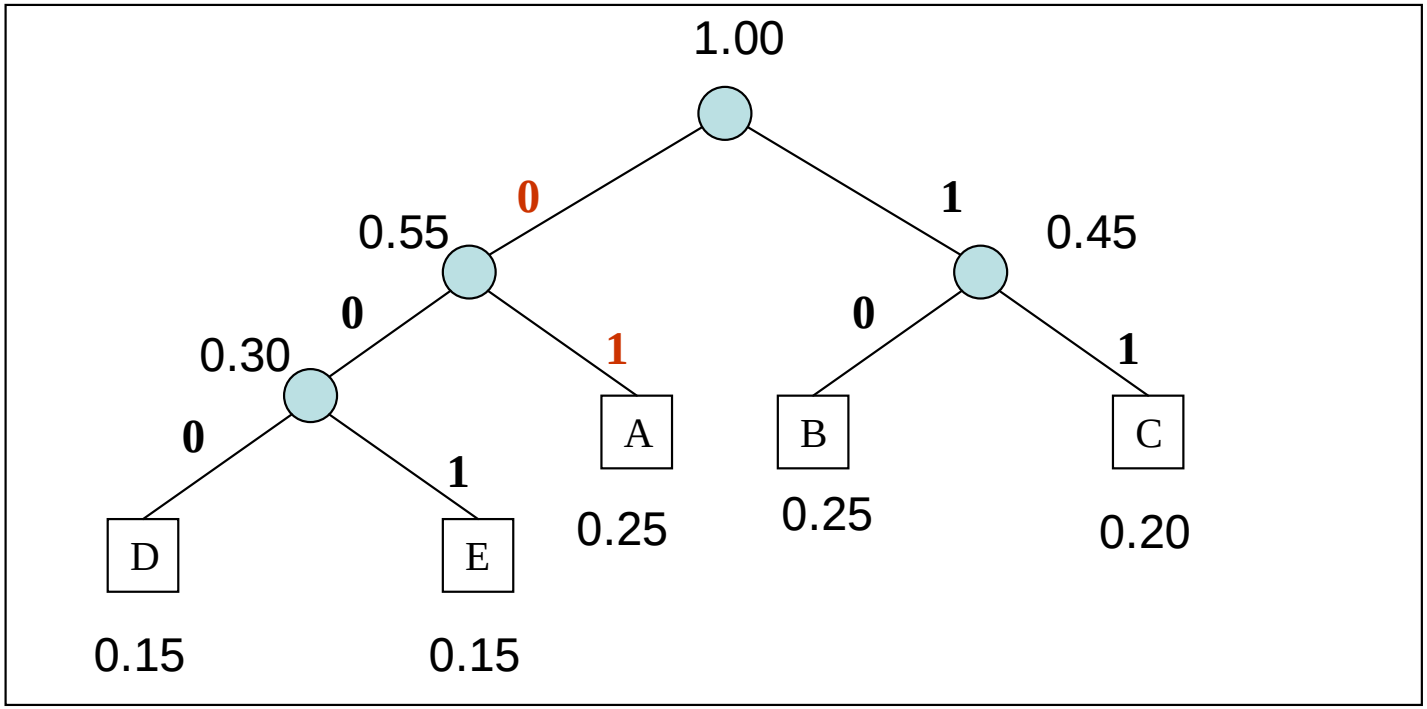
- Etiquetar ramas



Símbolos	A	B	C	D	E
Frecuencia / n	0.25	0.25	0.20	0.15	0.15
Código					

Huffman: construcción

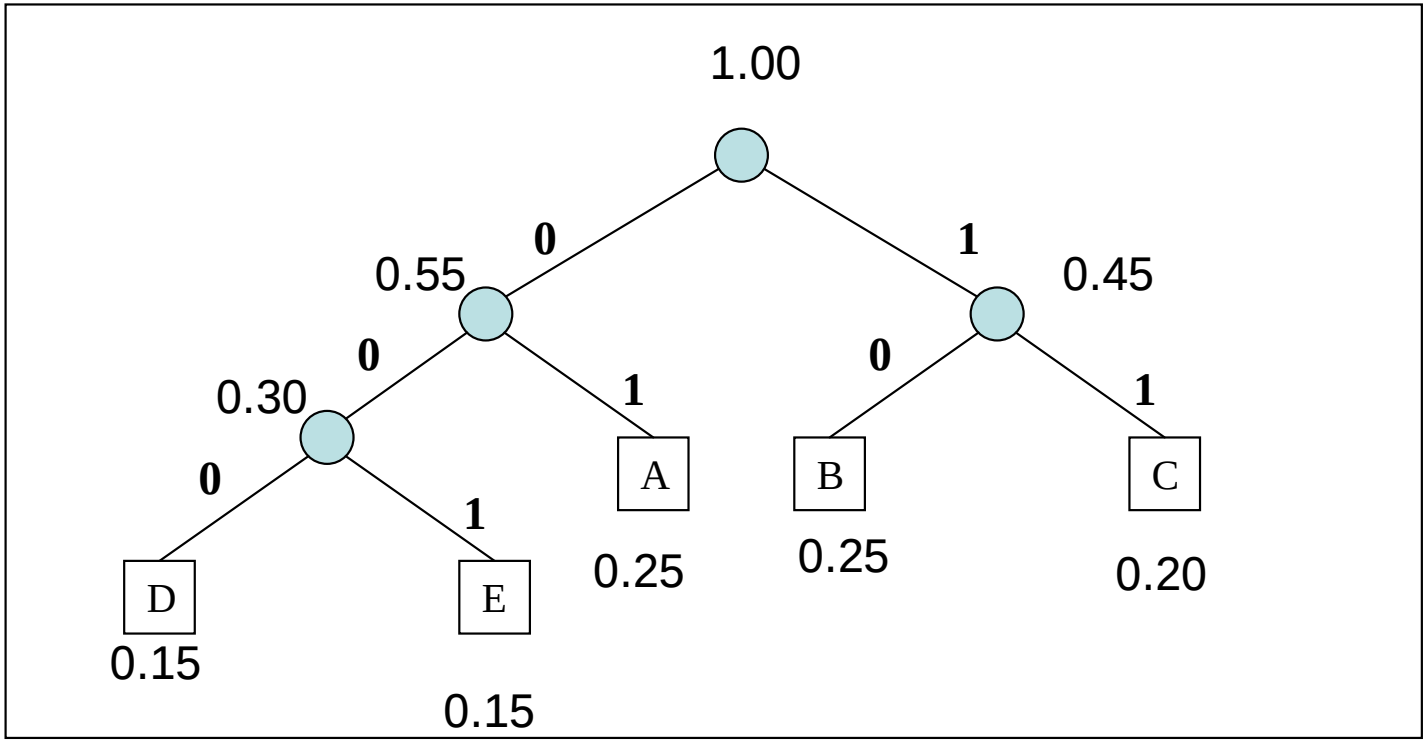
- Asignación de códigos



Símbolos	A	B	C	D	E
Frecuencia / n	0.25	0.25	0.20	0.15	0.15
Código	01	10	11	000	001

Huffman: construcción

- Asignación de códigos



Sempre hai combinacións de bits que non se usan

→ Asegurar libre de prefixo: ex: “00”, “0”, “1”

Código	01	10	11	000	001

Ex.: codificando ADB → 01 000 10

Outro exemplo

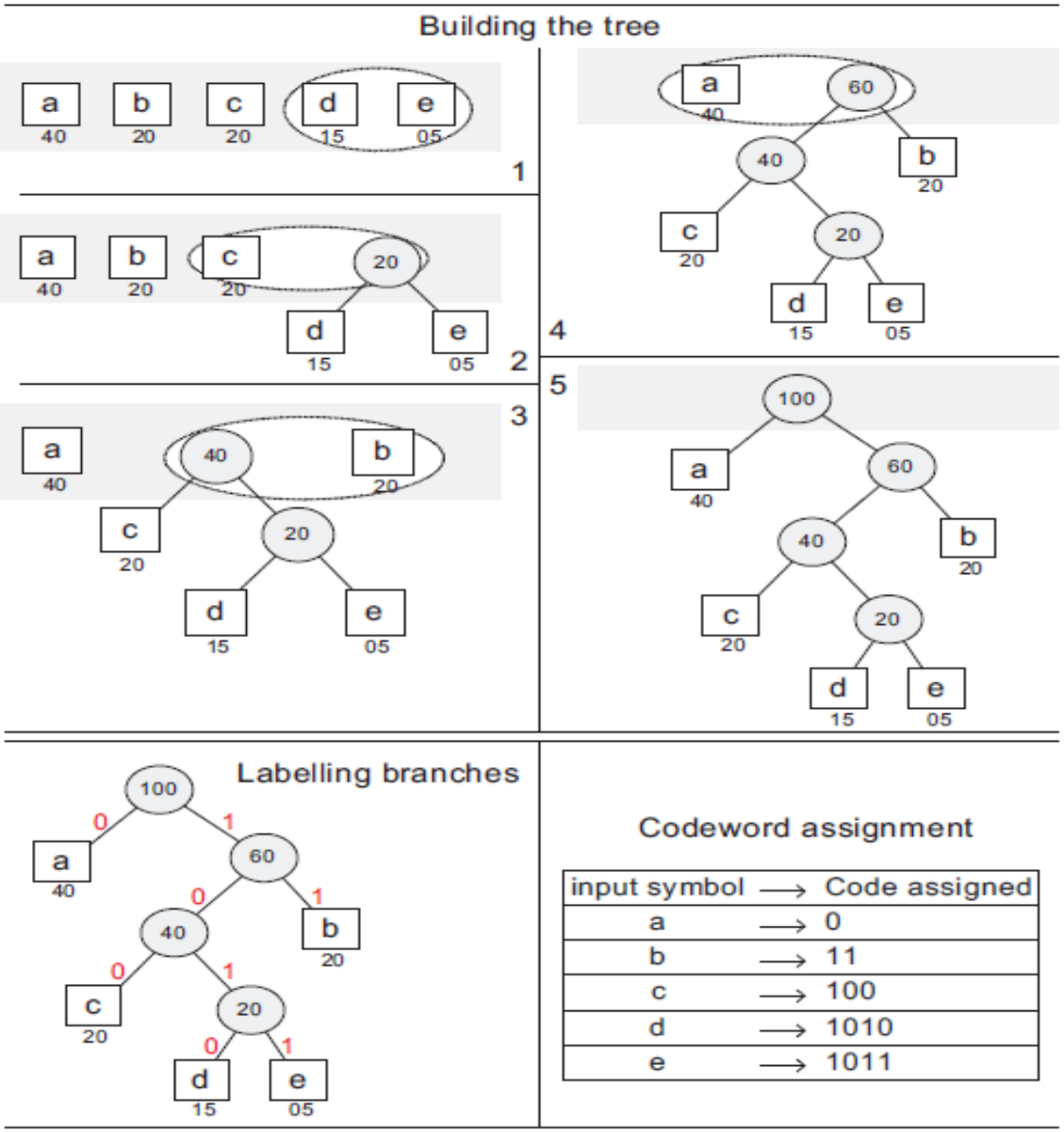


Figure 4.1: Building a classic Huffman tree.

Huffman clásico

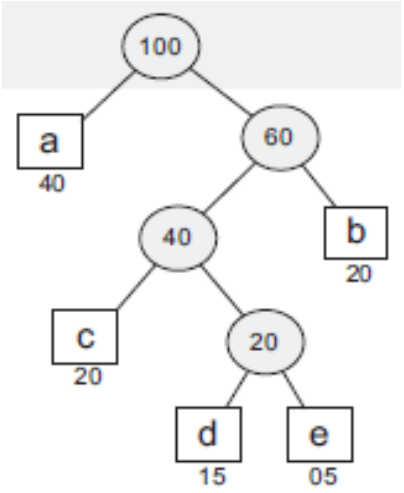
- Código libre de prefixo óptimo
- Baseado en caracteres (orixinalmente)
  - Os caracteres son os símbolos a codificar
- Orientado a bits: Os códigos son secuencias de bits
- Constrúese unha árbore Huffman para xerar os códigos
- A estructura da árbore ten que ser almacenada xunto co texto comprimido
- Ratio de compresión sobre el 60% (pobre no caso de texto en L.N.).

Huffman Canónico

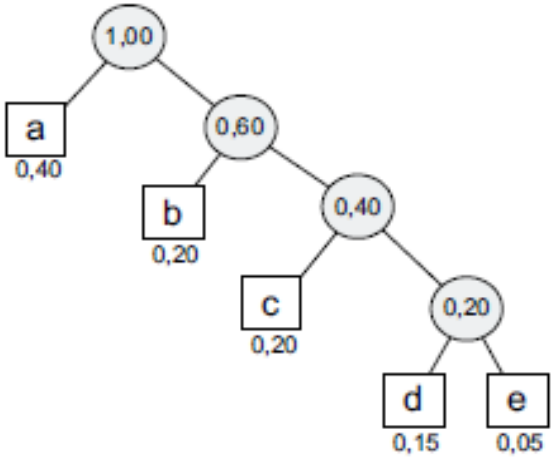
- Propiedades:
  - Os códigos da mesma lonxitude son dados secuencialmente (son números “binarios” consecutivos)
  - O primeiro código C_L de lonxitude |L| obtense a partir do último código C_{L-1} de lonxitude |L-1| como segue:

$$C_L = 2\,(C_{L-1} + 1)$$

- Intuitivamente:
  - No mesmo nivel, primeiro van as follas (se as hai) e despois os nodos internos (se os hai)



Non canónico



Huffman canónico

Almacenarase:

1.Listaxe de símbolos

2.Mín/Máx lonxitude código

3.Num códigos en cada nivel

<a, b, c, d, e><1, 4><1, 1, 1, 2>

Codificación Huffman: Exemplo.

- Exercicio:
  - Dado o texto seguinte: ABABCDABDEFAGAEADHABK
  - Aplica codificación Huffman clásica, e amosa:
    - A árbore Huffman resultante
    - O código que se lle asignará a cada letra (símbolo).
    - ¿é única dita árbore? Xustifica a túa resposta.
    - O texto comprimido resultante e a súa lonxitude en número de bits (non se teña en conta o vocabulario e a forma da árbore Huffman que debería formar tamén parte do ficheiro comprimido).
  - Transforma a árbore resultante previamente para convertila nunha árbore Huffman canónica, e amosa:
    - A árbore canónica resultante.
    - O código que lle corresponderá a cada letra
    - O texto comprimido resultante.
    - A cabeceira que debe ser incluida no xunto co texto comprimido para permitir unha posterior descompresión.