UNIVERSIDAD
COMPLUTENSE
MADRID

Tamaño de los índices

Sistemas de Gestión de Datos y de la Información
Enrique Martín - emartinm@ucm.es
Máster en Ingeniería Informática
Fac. Informática

Índices

- Los índices almacenan una lista de entradas por cada palabra del vocabulario.
- Cada entrada puede ser simple (docID) o muy detallada (docID y lista de apariciones).
- Por tanto, los **índices** ocupan una **cantidad elevada de memoria.** Pueden incluso no caber en memoria principal.

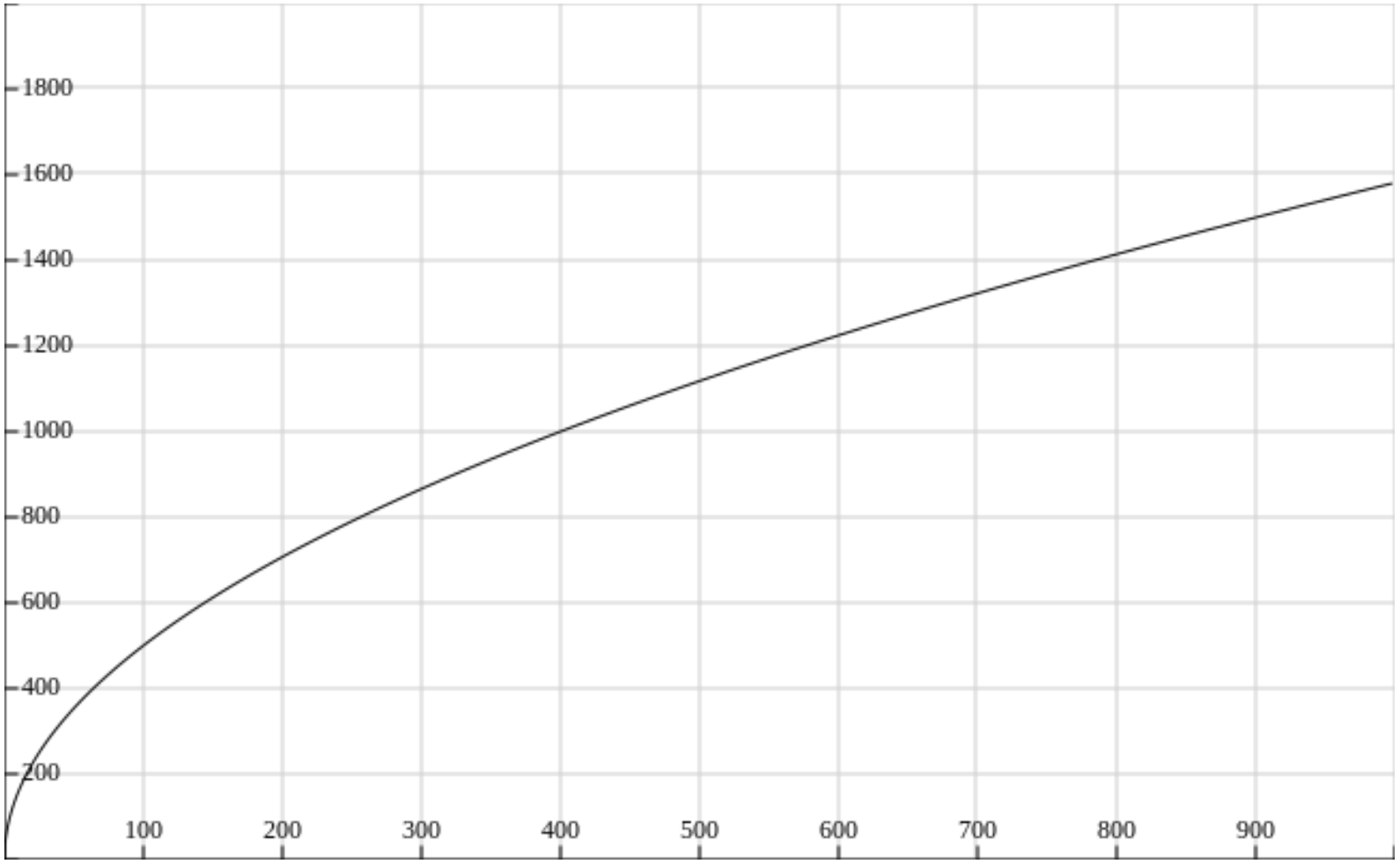
Ley de Heaps

- La **Ley de Heaps** estima el tamaño del vocabulario V en base al número de palabras T en una colección:

$$V = kT^b, \ con \ 30 \leq k \leq 100 \ y \ b \approx .5$$

- Es una ley empírica, con valores concretos para cada colección.
- Muestra que el número de términos crece **proporcional a la raíz del número de palabras**.
- Por tanto, el vocabulario crece de manera controlada según aumenta la colección.

Ley de Heaps



Para k=50 y b=0,5

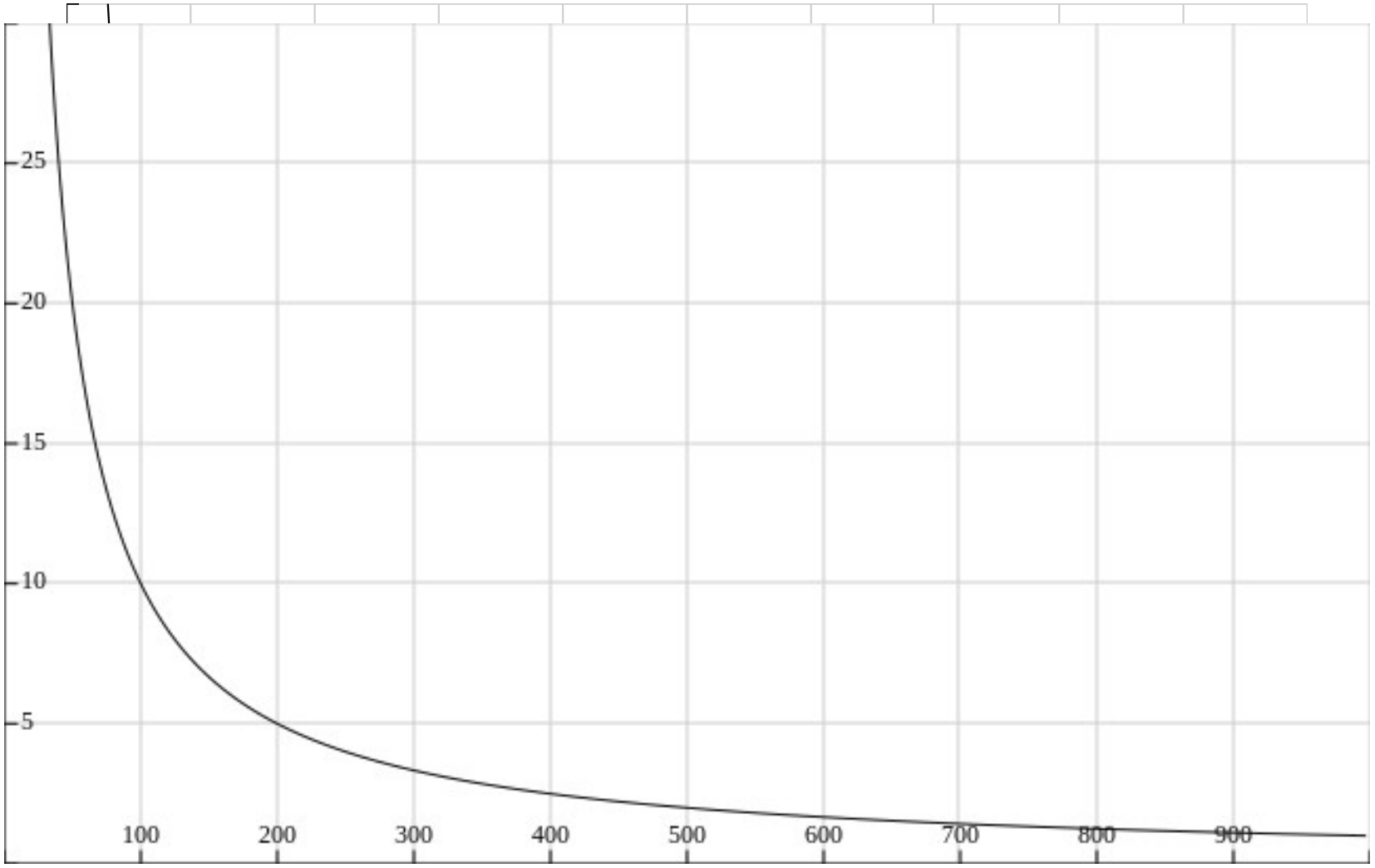
Ley de Zipf

- Por otro lado la **Ley de Zipf** relaciona la repeticiones de cada término.
- Sea cf_i las repeticiones del i-ésimo término más frecuente. Estas repeticiones cf_i son proporcionales a 1/i:

$$cf_i \approx \frac{1}{i}$$

  - El segundo elemento más frecuente aparece 1/2 de veces que el más frecuente.
  - El tercer elemento más frecuente aparece 1/3 de veces que el más frecuente.

Ley de Zipf



Ley de Zipf

- Esta ley empírica nos indica que unos pocos términos se repetirán mucho, por lo que contribuirán mucho al tamaño del índice.
- Estos términos suelen ser **separadores/palabras vacías** *(stopping words)* como artículos, pronombres, conjunciones, etc.
- Descartar separadores reducirá el tamaño del índice. Sin embargo no siempre es posible → búsquedas de frases.

Compresión

- Para reducir el tamaño de un índice se utilizan distintas técnicas de **compresión**.
- Las técnicas que más reducen el tamaño son las que comprimen **listas de apariciones**.
- Representan las apariciones a bajo nivel usando **bits**, por lo que **reducen** el tamaño pero **incrementan** el coste al necesitar procesarlas → compensa si evitas accesos a disco.

Compresión

- Las listas de repeticiones son listas de posiciones donde aparece un término en un documento: [4,16,65,89,134].
- Un primer paso para comprimir es no almacenar posiciones sino sus **diferencias**:
  [4,16-4,65-16,89-65,134-89]
  = [4,12 ,49 ,24 ,45]
- Las diferencias son números más pequeños y por tanto necesitarán **menos bits** para representarse.

Compresión

- Si todos los números necesitan el mismo número de bits no se gana nada al usar diferencias, pues todos ocuparán 32/64 bits.
- Debemos aplicar codificación de **longitud variable**.
- Veremos 4 técnicas:
  - Código unario
  - Elias-gamma
  - Elias-delta
  - Bytes variables

Compresión

- Sin embargo existen más técnicas:
  - Elias-omega
  - Golomb/Rice
  - LLRUN
  - Combinaciones de las anteriores

Código unario

- Muy sencilla: un número **n** se representa como (n-1) **1's** seguido de un **0**.
  - 1: **0**
  - 3: **110**
  - 7: **1111110**
- Una lista de diferencias se puede representar fácilmente como una lista de bits:
  [1,3,7] → **01101111110** (11 bits)
- Es óptimo desde el punto de vista de espacio para distribuciones de probabilidad:

$$P(n) = 2^{-n}$$

Elias-gamma

- La representación Elias-gamma $(\text{Elias-}\gamma)$n número **n** se forma concatenando 2 partes:
  - *offset:* **n** en binario, quitando el primer 1.
  - longitud: representación en **unario** de la longitud de **n** en **binario** $(\lfloor log_2\ n \rfloor + 1)$
- La representación es ***longitud + offset***

Número	Binario	Offset	Longitud (unario)	Representación
1	1		0 (1)	**0**
2	10	0	10 (2)	**100**
3	11	1	10 (2)	**101**
4	100	00	110 (3)	**11000**
5	101	01	110 (3)	**11001**

Elias-delta

- La representación Elias-delta $(\text{Elias-}\delta)$ milar a Elias-gamma, se concatenan 2 partes.
  - *offset:* **n** en binario, quitando el primer 1.
  - longitud: **representación en** $\textbf{Elias-}\gamma$ **ongitud de n** en **binario** $(\lfloor log_2\ n \rfloor + 1)$
- La representación es ***longitud + offset***

Número	Binario	Offset	Longitud Elias-gamma		Representación
1	1		0	(1)	**0**
2	10	0	100	(2)	**1000**
3	11	1	100	(2)	**1001**
4	100	00	101	(3)	**10100**
5	101	01	101	(3)	**10101**

Elias-gamma y Elias-delta

- En general la codificación Elias-gamma es más corta que Elias-delta para número pequeños (hasta 32).
- Según los números van creciendo, la codificación Elias-delta genera codificaciones más cortas.
- En ambos códigos, una lista de repeticiones se codifica concatenando la representación de sus elementos. P. ej. la lista [3,4,5]:
  - Elías-gamma → 101 1100 011001 (13 bits)
  - Elías-delta → 1001 10100 10101 (14 bits)

Bytes variables

- Cada números se representan como secuencias de 1 o varios bytes. Este número de bytes es variable.
- El primer bit de cada byte es el llamado *flag* de continuación:
  - **1**: el byte actual es el último de la secuencia.
  - **0**: el byte actual será continuado con otro byte.
- Los restantes 7 bits almacenan (una parte de) la representación en binario del número.

Bytes variables

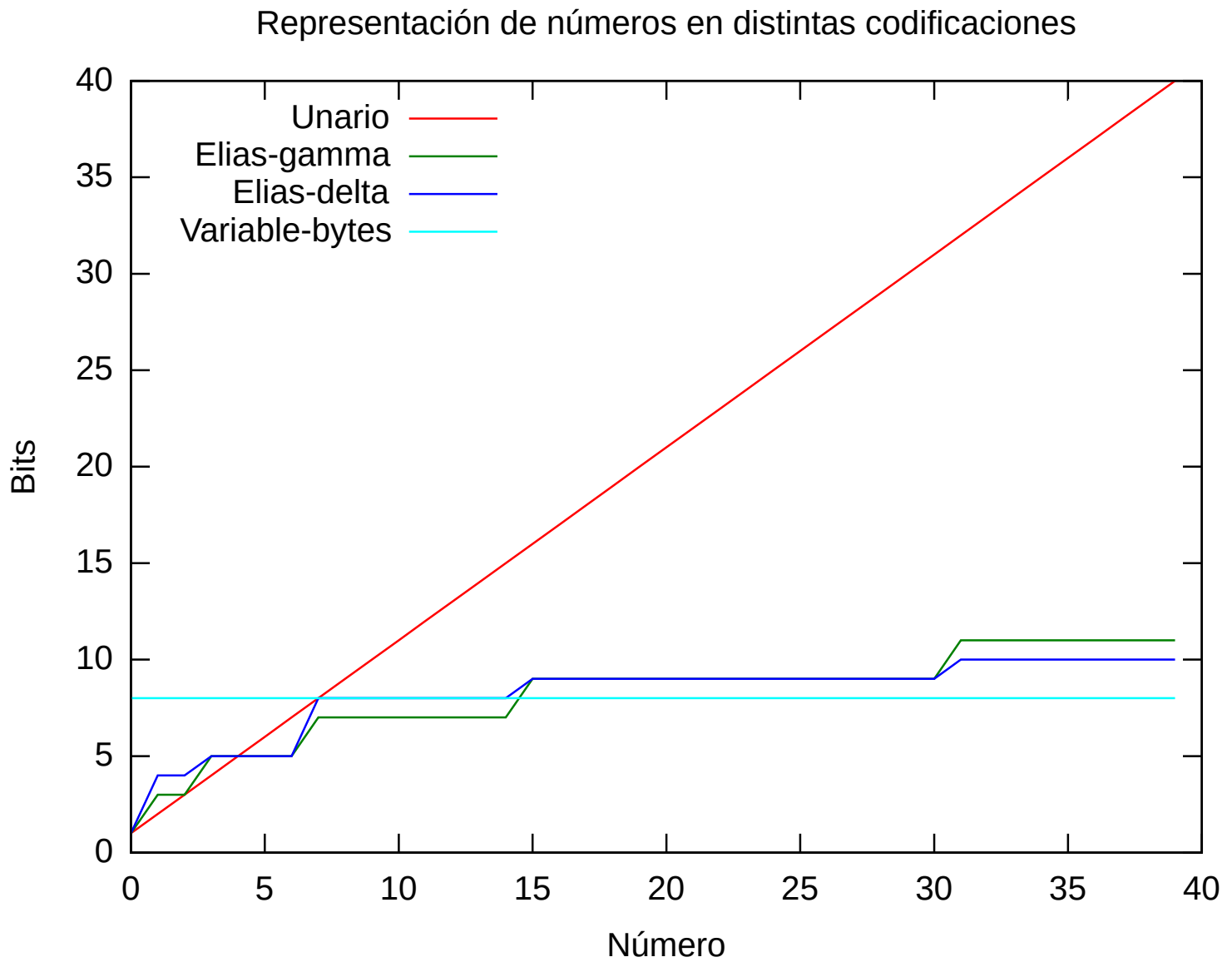
Número	Binario	Variable bytes
1	1	**1**0000001
2	10	**1**0000010
4	100	**1**0000100
5	101	**1**0000101
127	1111111	**1**1111111
315	100111011	**0**0000010 **1**0111011
824	1100111000	**0**0000110 **1**0111000

- Una lista de número se representa como una cadena de bytes:

10000100 **1**0000101 **1**1111111 **0**0000010 **1**0111011

4 5 127 315

Comparación codificaciones



Comparación codificaciones

- Tamaño de la representación de una lista aleatoria de 50 números entre el 1 y el 1000:
  - Enteros en binario: 1600 bits (32.0 bits/número)
  - Unario: 23107 bits (462.14 bits/número)
  - Elias-gamma: 838 bits (16.76 bits/número)
  - Elias-delta: 730 bits (14.6 bits/número)
  - Bytes variables: 744 bits (14.88 bits/número)