Tećnicas de diccionario

Fernando Martínez
fernando.martinez@upc.edu

Departament de Matemàtiques • Universitat Politècnica de Catalunya

8 de marzo de 2022

Capítulo 5 de Introduction to Data Compression **Sayood, Khalid** Morgan Kaufmann, 2012, 4th ed.

1. LZ77
   - DEFLATE Compressed Data Format Specification, RFC 1951
2. LZ78
   - LZW (Welch 1984)

LZ77

Ziv, Jacob; Lempel, Abraham (May 1977). *A Universal Algorithm for Sequential Data Compression*. IEEE Transactions on Information Theory. 23 (3): 337–343. CiteSeerX 10.1.1.118.8921. doi:10.1109/TIT.1977.1055714.

Ejemplo: Mensaje: patadecabraabracadabrapaaaaaaaaaaaatadecabr

Ventana de búsqueda: $S = 16$, ventana de texto avanzado: $L = 8$.

[literal, $\widetilde{match - length}$, $< offset >$]

||patadeca|braabracadabrapaaaaaaaaaaaatadecabr ['p', 0, 0]
|p|atadecab|raabracadabrapaaaaaaaaaaaatadecabr ['a', 0, 0]
|pa|tadecabr|aabracadabrapaaaaaaaaaaaatadecabr ['t', 0, 0]
|pat|adecabra|abracadabrapaaaaaaaaaaaatadecabr ['d', 1, 2]
|patad|ecabraab|racadabrapaaaaaaaaaaaatadecabr ['e', 0, 0]
|patade|cabraabr|acadabrapaaaaaaaaaaaatadecabr ['c', 0, 0]
|patadec|abraabra|cadabrapaaaaaaaaaaaatadecabr ['b', 1, 4]
|patadecab|raabraca|dabrapaaaaaaaaaaaatadecabr ['r', 0, 0]
|patadecabr|aabracad|abrapaaaaaaaaaaaatadecabr ['a', 1, 3]
|patadecabraa|bracadab|rapaaaaaaaaaaaatadecabr ['c', 3, 4]

LZ77

Ejemplo: Mensaje: patadecabraabracadabrapaaaaaaaaaaaatadecabr

Ventana de búsqueda: $S = 16$, ventana de texto avanzado: $L = 8$.

[...]
|patadecabraa|bracadab|rapaaaaaaaaaaaatadecabr ['c', 3, 4]
|patadecabraabrac|adabrapa|aaaaaaaaaaatadecabr ['a', 2, 13]
pat|adecabraabracada|brapaaaa|aaaaaaatadecabr ['p', 3, 7]
patadec|abraabracadabrap|aaaaaaaa|aaatadecabr ['a', 2, 13]
patadecabr|aabracadabrapaaa|aaaaaaaa|tadecabr ['t', 8, 1]
patadecabraabracada|brapaaaaaaaaaaat|adecabr| ['d', 1, 2]
patadecabraabracadabr|apaaaaaaaaaaatad|ecabr| ['e', 0, 0]
patadecabraabracadabra|paaaaaaaaaaatade|cabr| ['c', 0, 0]
patadecabraabracadabrap|aaaaaaaaaaatadec|abr| ['b', 1, 4]
patadecabraabracadabrapaa|aaaaaaaaatadecab|r| ['r', 0, 0]
['EOF', 0, 0]

Código:
['p', 0, 0], ['a', 0, 0], ['t', 0, 0], ['d', 1, 2], ['e', 0, 0], ['c', 0, 0], ['b', 1, 4], ['r', 0, 0], ['a', 1, 3], ['c', 3, 4], ['a', 2, 13], ['p', 3, 7], ['a', 2, 13], ['t', 8, 1], ['d', 1, 2], ['e', 0, 0], ['c', 0, 0], ['b', 1, 4], ['r', 0, 0], ['EOF', 0, 0]

- Los valores de S y L afectan a:
  - la ratio de compresión: bits a guardar *vs* posibilidad de encontrar cadenas más largas,
  - el tiempo de ejecución: mayor tiempo de búsqueda *vs* posibilidad de encontrar cadenas más largas.
- Comprimir es mucho más lento que descomprimir.
- Se acostumbra a combinar con un código de Huffman o una codificación aritmética.

DEFLATE Compressed Data Format Specification, RFC 1951

Usado por defecto en GZIP file format specification, RFC 1952

Trabaja con bloques aunque la ventana de búsqueda puede traspasar bloques.

Los tres alfabetos iniciales:

- literal: bytes 0...255 más 256 EOB (end of block)
- $\widetilde{match - length}$: 3-258
- $< offset >$: 1–32768 (2^{15})

se reducen a dos (ver RFC 1951 página 11).

DEFLATE (i)

- literal+ $\widetilde{match - length}$: 286 símbolos

Code	Extra Bits	Length(s)	Code	Extra Bits	Lengths	Code	Extra Bits	Length(s)
257	0	3	267	1	15,16	277	4	67-82
258	0	4	268	1	17,18	278	4	83-98
259	0	5	269	2	19-22	279	4	99-114
260	0	6	270	2	23-26	280	4	115-130
261	0	7	271	2	27-30	281	5	131-162
262	0	8	272	2	31-34	282	5	163-194
263	0	9	273	3	35-42	283	5	195-226
264	0	10	274	3	43-50	284	5	227-257
265	1	11,12	275	3	51-58	285	0	258
266	1	13,14	276	3	59-66			

The extra bits should be interpreted as a machine integer stored with the most-significant bit first, e.g., bits 1110 represent the value 14.

DEFLATE (ii)

- $< offset >$: 1–32768 (2^{15}): 30 símbolos + bits extras

Extra Code	Bits	Dist	Extra Code	Bits	Dist	Extra Code	Bits	Distance
0	0	1	10	4	33-48	20	9	1025-1536
1	0	2	11	4	49-64	21	9	1537-2048
2	0	3	12	5	65-96	22	10	2049-3072
3	0	4	13	5	97-128	23	10	3073-4096
4	1	5,6	14	6	129-192	24	11	4097-6144
5	1	7,8	15	6	193-256	25	11	6145-8192
6	2	9-12	16	7	257-384	26	12	8193-12288
7	2	13-16	17	7	385-512	27	12	12289-16384
8	3	17-24	18	8	513-768	28	13	16385-24576
9	3	25-32	19	8	769-1024	29	13	24577-32768

DEFLATE (iii)

Los diferentes alfabetos se pueden codificar con:

- Huffman estático:
  - literal+ $\widetilde{match - length}$: 286 símbolos

Lit Value	Bits	Codes
0 - 143	8	00110000 through 10111111
144 - 255	9	110010000 through 111111111
256 - 279	7	0000000 through 0010111
280 - 287	8	11000000 through 11000111

  - $< offset >$ 30 símbolos + bits extras: 5 bits (+ los bits extra)
- Huffman dinámico: Se envían las longitudes (que a su vez se codifican con un Huffman preestablecido)

También se puede usar *Lazy Matching* consistente en codificar el mejor resultado de la ventana actual y la ventana corrida una posición. En caso de ser mejor la segunda se envía un carácter y se corre la ventana una posición.

DEFLATE (iv)

Ejemplo: `D`, $\widetilde{4}$, $< 7 >$, `R`, $\widetilde{5}$, $< 3 >$, `A`, `A`, $\widetilde{16}$, $< 1026 >$

Traduzcamos estas *tripletas* a símbolos del alfabeto correspondiente:

68, 258_{+0bits}, $\mathbf{5}_{+1bit}$, 82, 259_{+0bits}, $\mathbf{2}_{+0bits}$, 65, 65, 267_{+1bit}, $\mathbf{20}_{+9bits}$

Codifiquemos los símbolos según el Huffman estático:

01110100	0000010	00101 0	10000010	0000011	00010	01110001	01110001
68	258_{+0bits}	5_{+1bit}	82	259_{+0bit}	2_{+0bits}	65	65

0001011 1	10100 000000001
267_{+1bit}	20_{+9bit}

El resultado final sería:

011101000000010001010100000100000011000100111000101110001000101111010000000001

- Después de un `símbolo del primer alfabeto mayor que 256` (257–285) ($\widetilde{match - length}$) siempre viene un `símbolo del segundo alfabeto` ($< offset >$).
- Después de un `símbolo del primer alfabeto menor que 256` (0–255) siempre viene otro `símbolo del primer alfabeto`.

LZ78

Ziv, Jacob; Lempel, Abraham (September 1978).*Compression of Individual Sequences via Variable-Rate Coding*. IEEE Transactions on Information Theory. 24 (5): 530–536. CiteSeerX 10.1.1.14.2892. doi:10.1109/TIT.1978.1055934.

Ejemplo: mississippi mississippi river

-Diccionario: []
|mississippi mississippi river [0, 'm']
-Diccionario: ['m']
m|ississippi mississippi river [0, 'i']
-Diccionario: ['m', 'i']
mi|ssissippi mississippi river [0, 's']
-Diccionario: ['m', 'i', 's']
mis|sissippi mississippi river [3, 'i']
-Diccionario: ['m', 'i', 's', 'si']
missi|ssippi mississippi river [3, 's']
-Diccionario: ['m', 'i', 's', 'si', 'ss']
mississ|ippi mississippi river [2, 'p']
-Diccionario: ['m', 'i', 's', 'si', 'ss', 'ip']
mississip|pi mississippi river [0, 'p']
-Diccionario: ['m', 'i', 's', 'si', 'ss', 'ip', 'p']
mississipp|i mississippi river [2, ' ']
-Diccionario: ['m', 'i', 's', 'si', 'ss', 'ip', 'p', 'i ']

LZ78

Ejemplo: mississippi mississippi river

-Diccionario: [’m’, ’i’, ’s’, ’si’, ’ss’, ’ip’, ’p’, ’i ’]
mississippi |mississippi river [1, ’i’]
-Diccionario: [’m’, ’i’, ’s’, ’si’, ’ss’, ’ip’, ’p’, ’i ’, ’mi’]
mississippi mi|ssissippi river [5, ’i’]
-Diccionario: [’m’, ’i’, ’s’, ’si’, ’ss’, ’ip’, ’p’, ’i ’, ’mi’, ’ssi’]
mississippi missi|ssippi river [10, ’p’]
-Diccionario: [’m’, ’i’, ’s’, ’si’, ’ss’, ’ip’, ’p’, ’i ’, ’mi’, ’ssi’, ’ssip’]
mississippi mississip|pi river [7, ’i’]
-Diccionario: [’m’, ’i’, ’s’, ’si’, ’ss’, ’ip’, ’p’, ’i ’, ’mi’, ’ssi’, ’ssip’, ’pi’]
mississippi mississippi| river [0, ’ ’]
-Diccionario: [’m’, ’i’, ’s’, ’si’, ’ss’, ’ip’, ’p’, ’i ’, ’mi’, ’ssi’, ’ssip’, ’pi’, ’ ’]
mississippi mississippi |river [0, ’r’]
-Diccionario: [’m’, ’i’, ’s’, ’si’, ’ss’, ’ip’, ’p’, ’i ’, ’mi’, ’ssi’, ’ssip’, ’pi’, ’ ’, ’r’]
mississippi mississippi r|iver [2, ’v’]
-Diccionario: [’m’, ’i’, ’s’, ’si’, ’ss’, ’ip’, ’p’, ’i ’, ’mi’, ’ssi’, ’ssip’, ’pi’, ’ ’, ’r’, ’iv’]
mississippi mississippi riv|er [0, ’e’]
-Diccionario: [’m’, ’i’, ’s’, ’si’, ’ss’, ’ip’, ’p’, ’i ’, ’mi’, ’ssi’, ’ssip’, ’pi’, ’ ’, ’r’, ’iv’, ’e’]
mississippi mississippi rive|r [14, ’EOF’]
-Diccionario: [’m’, ’i’, ’s’, ’si’, ’ss’, ’ip’, ’p’, ’i ’, ’mi’, ’ssi’, ’ssip’, ’pi’, ’ ’, ’r’, ’iv’, ’e’, ’r’]
Código: [0, ’m’], [0, ’i’], [0, ’s’], [3, ’i’], [3, ’s’], [2, ’p’], [0, ’p’], [2, ’ ’], [1, ’i’], [5, ’i’], [10, ’p’], [7, ’i’], [0, ’ ’], [0, ’r’], [2, ’v’], [0, ’e’], [14, ’EOF’]

LZW (Welch 1984)

Elimina la necesidad de enviar la letra siguiente inicializando un diccionario, usualmente con los 256 bytes.

A la hora de codificar vamos acumulando letras en una cadena P mientras está en el diccionario. En el momento en que $P\|s$ no parece en el diccionario:

I) Enviamos el índice de P.

II) Añadinos $P\|s$ al diccionario.

III) Inicializamos $P = s$.

Ejemplo LZW (i)

Ejemplo: $a^1b^2a^3b^4a^5b^6a^7b^8b^9a^{10}$

```
Diccionario                                                    Salida
1. a
2. b
      (1) leemos  P=a,
      (2) leemos b, P=ab NO está en el diccionario              1
3. ab
      P=b
      (3) leemos a, P=ba NO está en el diccionario              2
4. ba
      P=a
      (4) leemos b, P=ab
      (5) leemos a, P=aba NO está en el diccionario             3
5. aba
      P=a
      (6) leemos b, P=ab
      (7) leemos a, P=aba
      (8) leemos b, P=abab NO está en el diccionario            5
6. abab
      P=b
```

Ejemplo LZW (ii)

Ejemplo: $a^1b^2a^3b^4a^5b^6a^7b^8b^9a^{10}$

```
Diccionario                                                    Salida
1. a
2. b
3. ab
4. ba
5. aba
6. abab
      P=b
      (9) leemos b, P=bb NO está en el diccionario              2
7. bb
      P=b
     (10) leemos a, P=ba
     leemos EOF, P=baEOF NO está en el diccionario              4
```

Código: 1,2,3,5,2,4
Diccionario inicial: a, b

Ejemplo LZW (iii)

Ejemplo: Código: 1,2,3,5,2,4
Diccionario inicial: a, b

Diccionario	Salida acumulada
1. a	
2. b	
leemos 1	a
3. a?	
leemos 2	a b
3. ab	
4. b?	
leemos 3	ab ab
4. ba	
5. ab?	
leemos 5, NO está completo pero sabemos las primeras letras Podemos completar la 5ª entrada del diccionario	abab ab?
5, aba	
Completo la letra del mensaje que me faltaba	abab aba
6. aba?	

Ejemplo LZW (iv)

Ejemplo: Código: 1,2,3,5,2,4
Diccionario inicial: a, b

```
Diccionario                                   Salida acumulada
1. a
2. b
3. ab
4. ba
5. aba
                                                 abababa
6. aba?
        leemos 2                                 abababa b
6. abab
7. b?
        leemos 4                                 abababab ba
7. bb
```

Compress

Diccionario dinámico con 2^9 entradas ($\Rightarrow$ 9 bits para codificar el índice de la entrada) de las cuales 256 están inicializadas. Al llenarse aumenta a 2^{10} entradas (10 bits por entrada), así sucesivamente hasta 2^{16} entradas (16 bits por entrada).

A partir de este momento se vuelve un diccionario estático mientras que la ratio de compresión no disminuya. En ese instante se reinicializa el diccionario.

En el caso de **GIF** al llenarse el diccionario se reinicializa.

gzip vs Compress

Table 9.3: *Performance of* gzip *vs* compress.

File	Kbyte	Encode (10K/s) [compression (% remaining)] *compress*	*gzip* (-1)	*gzip* (-3)	*gzip* (-6)	Decode (10K/s) *compress*	*gzip*
bib	109	58 [41.8]	51 [39.4]	39 [35.7]	18 [31.5]	95	158
book1	751	42 [43.2]	42 [47.5]	28 [43.8]	12 [40.8]	102	167
geo	100	45 [76.0]	27 [68.2]	14 [67.9]	6 [66.9]	70	107
obj1	21	44 [65.3]	37 [49.8]	34 [49.2]	23 [48.0]	56	62
pic	501	125 [12.1]	101 [12.8]	84 [12.2]	36 [11.0]	187	313
progc	39	52 [48.3]	46 [39.0]	38 [36.6]	22 [33.5]	76	107
Average		70 [47.8]	51 [42.8]	39 [40.9]	20 [38.6]	98	152

Darrel Hankerson Greg A. Harris Peter D. Johnson, Jr. *Introduction to Information Theory and Data Compression* Second Edition, pág 243