Information Retrieval

Lay Kuan Loh

July 6, 2013

Writing numbers in variable byte notation

Steps

- Write a number in binary
- Pad it with zeros in front until the number of bits is a multiple of seven
- Chunk the bits into groups of seven
- First group has a "1" appended to it.
  Other groups have a "0" appended to it.

Example: Writing 2056 in variable byte notation

- 2056 is 1000 0000 1000 in binary
- chunk it as 10000 0001000
- Pad it as 001 0000 000 1000
- Append the "1"s and "0"s to get: 1001 0000 0000 1000

Exercise: Write the following numbers in variable byte notation

(1) Number: 1

- Binary: 1
- Padded to have a multiple of seven bits: 000 0001
- Append a "1" to the first group: 1000 0001

(2) Number: 1023

- Binary: 11 1111 1111
- Padded to have a multiple of seven bits: 000 0111 111 1111
- Append "1"s and "0"s to get: 0000 0111 0111 1111

(3) Number: 12345678

- Binary: 1011 1100 0110 0001 0100 1110
- Padded to have a multiple of seven bits: 000 0101 111 0001 100 0010 100 1110
- Append "1"s and "0"s to get: 1000 0101 0111 0001 0100 0010 0100 1110

Question

What is the smallest number for which variable byte encoding is less efficient than standard four byte notation?

In other words, for what number n does the variable byte representation of n require more bits than the standard four byte representation?

Answer:

- Idea: Initially, the numbers $1, 2, 3, \ldots$ are stored in variable byte notation and do not take much space. Slowly, as numbers increase, they take more space, until they start taking ≥ 32 bit space, then variable byte encoding is less efficient, as it chews up unnecessary bits by padding the representation.
- Variable byte notation:

  We have a number represented by 28 bits, then we pad it with four bits. Anything bigger, and standard four byte becomes better. That is the binary number: 1111111 1111111 1111111 1111111
- Standard four byte notation: 32 bits needed

Question

11100101111000001011111010110

How many bits would we have needed to encode this with the standard 4 byte integer encoding?

Answer

- Pad to a multiple of seven: 0000011 1001011 1100000 1011111 1010110
- Add bits in front: 10000011 01001011 01100000 01011111 01010110
  5 bytes

γ Encoding

γ codes implement variable-length encoding by splitting the representation of a gap G into a pair of length and offset. Offset is G in binary, but with the leading 1 removed.

For example, for 13 (binary 1101) offset is 101.

Length encodes the length of offset in unary code.

For 13, the length of offset is 3 bits, which is 1110 in unary.

The code of 13 is therefore 1110101, the concatenation of length 1110 and offset 101.

Table 5.5

Number	Unary code	Length	Offset	γ code
0	0			
1	10	0		0
2	110	10	0	10, 0
3	1110	10	1	10, 1
4	1110	110	00	110, 00
9	11 1111 1110	1110	001	1110, 001
13		1110	101	1110, 101
24		1 1110	1000	11110, 1000
511		1 1111 1110	1111 1111	1 1111 1110, 1111 1111
1025		111 1111 1110	00 0000 0001	111 1111 1110, 00 0000 0001

γ codes are relatively inefficient for large numbers (e.g., 1025) as they encode the length of the offset in inefficient unary code. δ codes differ from codes in that they encode the first part of the code (length) in code instead of unary code.

Example: γ code

- 13 in binary: 1101
- offset: 101 (leading one removed)
- length: length of offset in unary code

  Length of offset for 13 is 3 bits, which is 1110 in unary
- γ code of 13: 1110 101

  Concatenation of length 1110 and offset 101

Example: δ code

- Length is coded in γ code, not unary code
- δ code of 7 is 10 0 11
- 7 in binary: 111
- Offset of 7: 11 (leading one removed)
- Length: 2 bits

  Encoded in γ code as

  2 in binary: 10

  Leading one removed: 0

  1 bit in unary: 10

  γ code: 10 1
- In delta code: 100 11

Question

- Compute the δ codes for the other numbers in Table 5.5. For what range of numbers is the δ code shorter than the γ code?

  Example for 24:

In binary: 1 1000

Offset: 1000

Length of offset: 4 bits

Write "4" in γ code as: 110 00

Therefore 24 in δ code is: 11000, 1000

- γ code beats variable byte code in Table 5.6 because the index contains stop words and thus many small gaps. Show that variable byte code is more compact if larger gaps dominate.

  Length of γ code for number x: $2\lfloor \log_2 x \rfloor + 1$

  Length of δ code for number x: $\lfloor \log_2 x \rfloor + 2\lfloor \log_2(\lfloor \log_2 x \rfloor + 1) \rfloor + 1$

  Larger gaps $\Rightarrow$ must write almost everything

- Compare the compression ratios of code and variable byte code for a distribution of gaps dominated by large gaps.

  Variable byte code:

  $8\lceil (\lfloor \log_2 x \rfloor)/7 \rceil$