Bug 1285848

Related Components

RiceDeltaDecoder

- Input (see safebrowsing.proto here)
  - Rice parameter: int32
  - Encoded data: byte array
  - First value: int64
  - Num of entries (deltas): int32
- Output
  - Decoded data: uint32 array
    - The first value would be included in the array
    - The array could be used to represent 4-byte prefixes (for addition) or removal indices (for removal)
  - Decoding result: boolean

BitBuffer

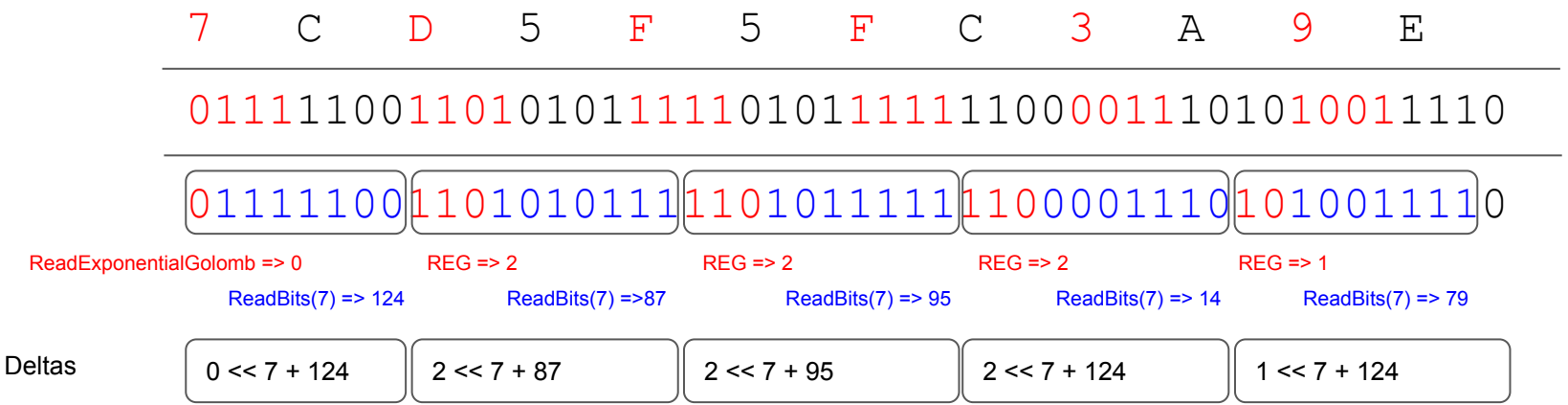
- Copied from webrtc/base/bitbuffer.h/cc
  - There is already a copy in gecko
  - Slightly modified to adapt to safebrowsing codebase.
- In composition with RiceDeltaDecoder
- Only 2 functions would be called by RiceDeltaDecoder
  - ReadExponentialGolomb
  - ReadBits

BitBuffer

- ReadExponentialGolomb
  - Count the number of 1_2 until the first 0_2 appears in the bit stream
  - Would consume all the leading 1's and the first appeared 0
  - E.g. 110_2 => returns 2, 11000_2 => returns 2, 1110111_2 => returns 3
  - The original implementation is to count the number of 0_2 instead of 1_2
    - This is the main part that I made the change to it.
- ReadBits(K)
  - Read in K bits and interpret to uint32 in big endian.
  - Would consume K bits.
  - In the real use case, only one bit would be read at a time bacause of the endianness issue which will be mentioned later

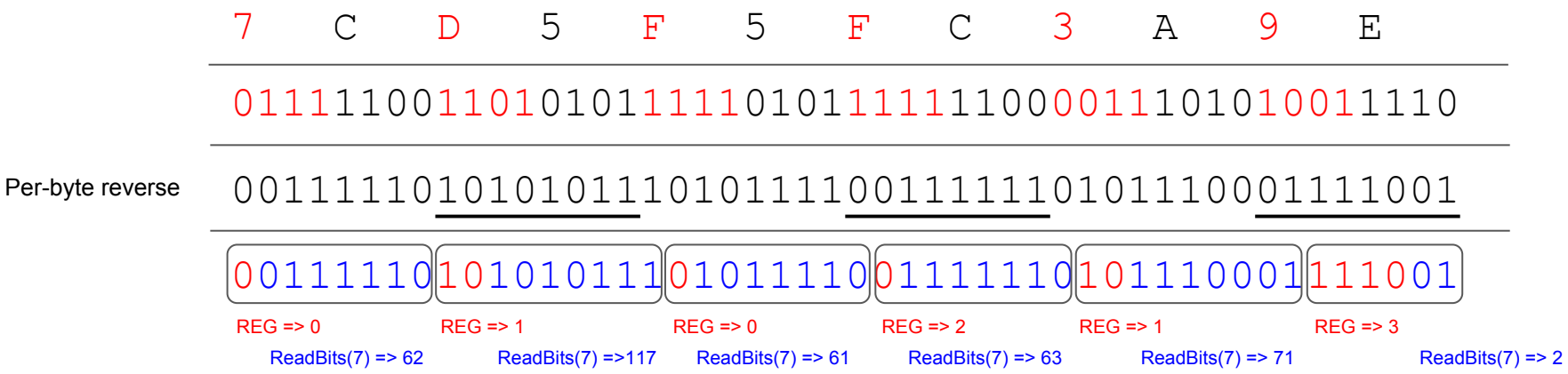
An Example of Decoding

- Rice parameter (K): 7
- # of entries (deltas): 5
- Encoded data: “\x7C\xD5\xF5\xFC\x3A\x9E”



An Example of Decoding

- However, google writes bits “reversely” to the byte stream.
  - The bits needs be read from LSBit to MSBbit per byte !!
- Other than that, the algorithm is the same
  - 1) Read the exponential part
  - 2) Read K bits as the remainder part (but interpret to int32 in little endian)



An Example of Decoding

- The decoded integers are not the actual 4-byte prefixes or removal indices.
- Instead, they are the differences between each adjacent elements.
- For example
  - Assume the removal indices are [4, 10, 77, 100, 124, 125]
  - The actual values to be encoded are [6, 67, 23, 24, 1]
- To re-construct the most original values, we need deltas as well as the first value!