Encoding Techniques

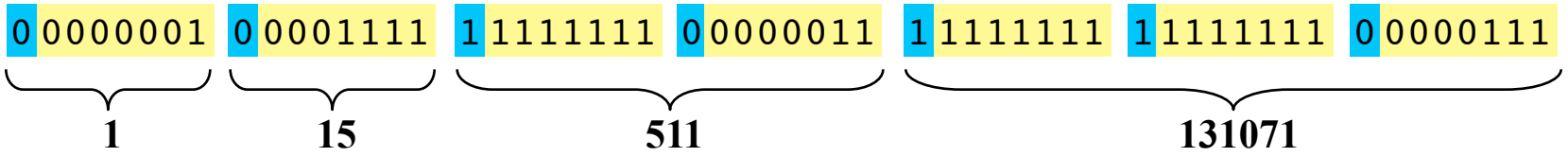
- Bit-level encodings:
  - Unary: *N* '1's followed by a '0'
  - Gamma: $\log_2(N)$ in unary, then floor(log2(*N*)) bits
  - Rice$_K$: floor(N / 2^K) in unary, then N mod 2^K in K bits
    - special case of Golomb codes where base is power of 2
  - Huffman-Int: like Gamma, except $\log_2(N)$ is Huffman coded instead of encoded w/ Unary

- Byte-aligned encodings:
  - varint: 7 bits per byte with a continuation bit
    - 0-127: 1 byte, 128-4095: 2 bytes, ...
  - ...

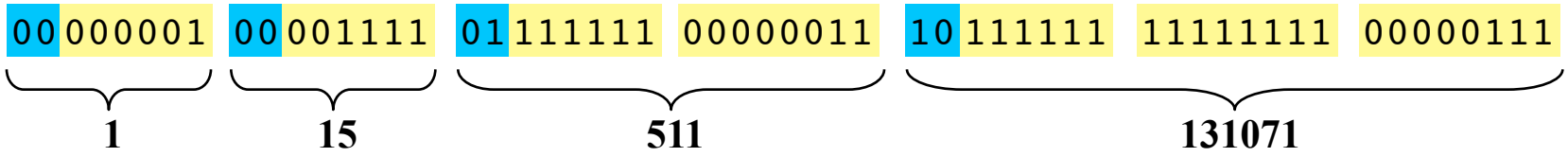
Google

Byte-Aligned Variable-length Encodings

- Varint encoding:
  - 7 bits per byte with continuation bit
  - Con: Decoding requires lots of branches/shifts/masks



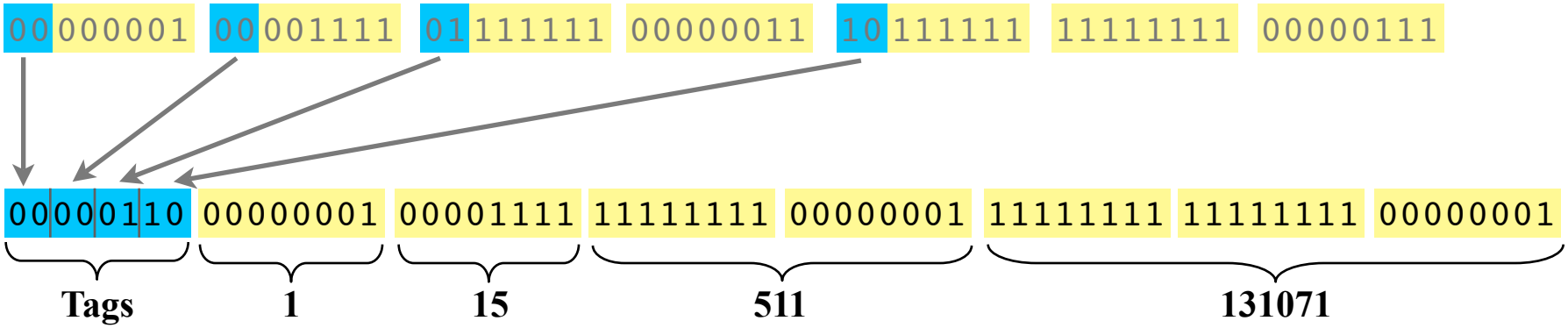
- Idea: Encode byte length as low 2 bits
  - Better: fewer branches, shifts, and masks
  - Con: Limited to 30-bit values, still some shifting to decode



Google

Group Varint Encoding

- Idea: encode groups of 4 values in 5-17 bytes
  - Pull out 4 2-bit binary lengths into single byte prefix



- Decode: Load prefix byte and use value to lookup in 256-entry table:

```
...
00000110 ──► Offsets: +1,+2,+3,+5; Masks: ff, ff, ffff, ffffff
...
```

- Much faster than alternatives:
  - 7-bit-per-byte varint: decode ~180M numbers/second
  - 30-bit Varint w/ 2-bit length: decode ~240M numbers/second
  - Group varint: decode ~400M numbers/second

Google