• Each subsequent symbol is assigned the next binary number in sequence, ensuring that following codes are always higher in value.

• When you reach a longer codeword, then after incrementing, append zeros until the length of the new codeword is equal to the length of the old codeword.

Parameters

Data type	Name	Comment
itf8[ ]	alphabet	list of all encoded values
itf8[ ]	bit-lengths	array of bit-lengths for each symbol in the alphabet

Byte array coding

Often there is a need to encode an array of bytes. This can be optimized if the length of the encoded arrays is known. For such cases BYTE_ARRAY_LEN and BYTE_ARRAY_STOP codings can be used.

BYTE_ARRAY_LEN

Byte arrays are captured length-first, meaning that the length of every array is written using an additional encoding. For example this could be a golomb encoding. The parameter for BYTE_ARRAY_LEN are listed below:

Data type	Name	Comment
encoding<int>	lengths encoding	an encoding describing how the arrays lengths are captured
encoding<byte>	values encoding	an encoding describing how the values are captured

BYTE_ARRAY_STOP

Byte arrays are captured as a sequence of bytes terminated by a special stop byteFor example this could be a golomb encoding. The parameter for BYTE_ARRAY_STOP are listed below:

Data type	Name	Comment
byte	stop byte	a special byte treated as a delimiter
itf8	external id	id of an external block containing the byte stream

12.3 Choosing the container size

The CRAM format does not constrain the size of the containers and this is an implementation choice. However, the following should be considered when deciding the container size:

~~• Data can be compressed better by using larger containers~~

- Splitting data into defined genomic regions can be problematic when the depth is highly variable, so data size is a more useful metric.

  ~~• Random access performanceis better for smaller containers~~
- Larger containers compress better.

  •
- Smaller containers offer better random access performance.
- Streaming is more convenient for small containers.

  •
- Applications typically buffer containers into memory. This becomes more important if an application is multi-threaded.

We recommend 1MB containers. They are small enough to provide good random access and streaming performance while being large enough to provide good compression. 1MB containers are also small enough to fit into the L2 cache of most modern CPUs.

Some simplified examples are provided below to fit data into 1MB containers.

Unmapped short reads with bases, read names, recalibrated and original quality scores

We have 10,000 unmapped short reads (100bp)with read names, recalibrated and original quality scores. We estimate 0.4 bits/base (read names) + 0.4 bits/base (bases) + 3 bits/base (recalibrated quality scores) + 3 bits/base (original quality scores) =~ 7 bits/base. Space estimate is (10, 000 * 100 * 7) / 8 / 1024 / 1024 =~ 0.9 MB. Data could be stored in a single The optimal size of container will depend on the use case (e.g. active work or archival; streaming or random-access). We recommend application writers consider exposing a series of compression profiles or command-line options to provide user control, defaulting to faster methods and no more than a few megabytes of uncompressed data per container.

Unmapped long reads with bases, read names and quality scores

We have 10, 000 unmapped long reads (10kb) with read names and quality scores. We estimate: 0.4 bits/base (bases) + 3 bits/base (original quality scores) =~ 3.5 bits/base. Space estimate is (10,000 * 10,000 * 3.5) / 8 / 1024 / 1024 =~ 42 MB. Data could be stored in 42 x 1MB containers.

Mapped short reads with bases, pairing and mapping information

We have 250,000 mapped short reads (100bp) with bases, pairing and mapping information. We estimate the compression to be 0.2 bits/base. Space estimate is (250,000 * 100 * 0.2) / 8 / 1024 / 1024 =~ 0.6 MB. Data could be stored in a single container.

Embedded reference sequences

We have a reference sequence (10Mb). We estimate the compression to be 2 bits/base. Space estimate is (10000000 * 2 / 8 / 1024 / 1024) =~ 2.4MB. Data could be written into three containers: 1MB + 1MB + 0.4MBConsider that different instrument types can have vastly different compression ratios per byte stream, for example quality values quantised into 4 values with high correlation with previous qualities, or 60 or more discrete values with minimal predictability. Tools such as *samtools cram-size* can report the space taken up by each data type to aid parameter evaluation.