# Algorithm for compression with random access

Shashank Vatedka

July 25, 2021

## Contents



## 1 Algorithm using only zstd but without dictionary compression

Let us build a compressor for (specifically) fasta files.

- Read file into a buffer
- Maintain data structures (for now, arrays) `comm_loc` and `seq_loc` for starting locations of compressed versions of comment and sequences.
- Maintain a buffer `compstring` to store the concatenation of compressed strings
- While not eof, do the following:
  - Read the next comment/sequence, and store this in temporary buffer
  - Compress the string using zstd, and append the compressed string to `compstring`
  - If this string was a comment, then append the starting location of the compressed string in `compstring` to `comm_loc`, else to `seq_loc`
- Write `compstring`, `comm_loc` and `seq_loc` to separate (binary) files.

This is how you would do compression with random access.

- Suppose that we want to reconstruct the i'th comment and sequence. Take this as user input.
- Load `comm_loc` and `seq_loc` to buffer.
- Initialize the file pointer to the file containing `compstring`
- Get the starting locations of the i'th and (i+1)'th compressed sequences from `comm_loc` and `seq_loc`. Using this, compute the lengths of compressed versions of the i'th comment and sequence
- Load the compressed versions to separate buffers
- Decompress the sequences using zstd, and print this/put this in a separate text file.