Run-Length Encoding

In class, we used Elias-gamma-codes to encode a list of integers (as part of RLE). But there are other options!

1. Develop a run-length encoding scheme where the input is a bitstring, the output uses 4 different characters, and a run-length of \( k \) is encoded with \( \lfloor \log k \rfloor + 1 \) characters. As always, the encoding must be lossless, i.e., you need to be able to decode uniquely.

2. Which encoding scheme has the better compression ratio, the original RLE or the one from 1)?

3. It turns out that binary encoding is not the best one can do! Consider the following method of encoding positive integers:

\[
\begin{align*}
1 & \quad 2 & \quad 3 & \quad 4 & \quad 5 & \quad 6 & \quad 7 & \quad 8 & \ldots \\
A & \quad B & \quad AA & \quad AB & \quad BA & \quad BB & \quad AAA & \quad AAB & \ldots
\end{align*}
\]

(a) How many characters does this encoding scheme use to encode \( k \)?

(b) Combine this idea with the approach from 1) to obtain a version of RLE that has better compression ratio than the one from class.

LZW Encoding

1. Perform LZW-encoding for the string “barbarbarara”. Decode the result.

2. Perform decoding for the LZW-encoding: 98, 97, 114, 128, 114, 97, 131, 134, 129, 101, 110

3. Let \( a^n \) denote a string \( S = “aa…a” \) such that \(|S| = n\). Show that \( a^n \) can be encoded using LZW in \( \sqrt{2n} \) code-numbers.

Burrows-Wheeler Transform

Given the Burrows-Wheeler transform \( C \) for some string \( S \). Describe how you would get the suffix array for \( S \) in linear time.