UNIVERSIDADE DE AVEIRO

DEPARTAMENTO DE ELECTRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA

Algorithmic Information Theory (2025/26)

Project #1

To deliver: 13 April 2026 (18:00)

To Present: 17 April 2026 (during the class)

Introduction

Modern data compression systems can be viewed as the composition of two tightly coupled components: *modeling* and *coding*. As depicted in Figure 1, the modeling stage seeks regularities in the data and translates them into an estimated probability distribution (or, equivalently, a predictive mechanism) for the symbols to be encoded. The coding stage then uses this distribution to map symbols into bit strings whose expected length approaches the information content of the source, assigning shorter codewords to more probable events.

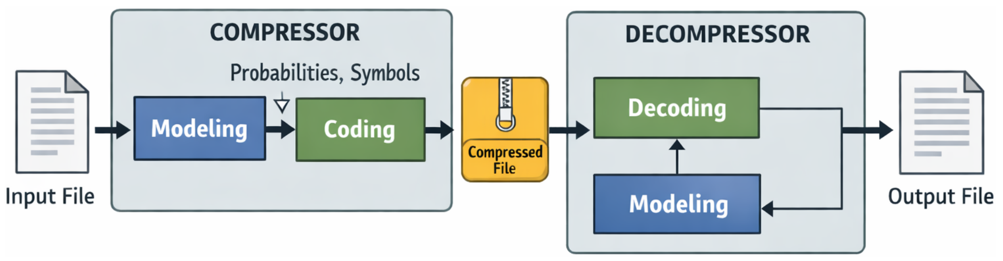

Figure 1: The architecture of a lossless compressor and decompressor.

This separation is fundamental: only with an appropriate model a sequence over an alphabet Σ with length S can be better represented than one obtained from a uniformly random source. Note that any lossless representation may use, in the worst case, $S \log_2 |\Sigma|$ bits to represent the sequence, where $|\Sigma|$ denotes the size of the alphabet. This happens if the sequence is random. Compression is therefore possible only to the extent that the data deviates from total randomness—i.e., to the extent that it contains structure that a model can capture and an encoder can exploit.

The challenge

Within the framework of Algorithmic Information Theory, Project #1 consists of designing and implementing a *lossless* data compression tool. The deliverable must include both a compressor and a corresponding decompressor, and it must employ *arithmetic coding* (or an equivalent *range encoder*) as the coding stage.

The modeling component is left open-ended, provided that all project requirements are satisfied. This flexibility allows each group to explore different trade-offs between compression ratio, computational cost, and implementation complexity. Although not mandatory, an implementation in an efficient systems programming language (e.g., C, C++, or Rust) is strongly recommended, to ensure good performance on realistic inputs.

We will provide a `.zip` archive containing files of different types to be used as a benchmark suite for evaluating the developed compression tool.

Requirements

1. Students will work in groups of three elements.
2. The compressor must be *lossless*: only solutions that reconstruct the original data exactly after decompression will be considered.
3. The coding stage must use *arithmetic coding/decoding* or an equivalent *range coder/decoder.*
4. The implementation must run in Linux, using a maximum of 8 GB of RAM.
5. A benchmark against state-of-the-art compressors (e.g., `gzip`, `bzip2`, `lzma/xz`, `zstd`) must be provided, reporting at least:
   - compressed size (bytes),
   - compression rate per symbol (e.g., bits/symbol),
   - compression time,
   - decompression time.
6. The evaluation will emphasize your ability to synthesize concrete ideas, explanations, comparisons, and results. You must submit a report of at most 6 pages in PDF format, using the following template: template.
7. Each group must give an oral presentation (maximum 10 minutes) supported by slides and a small demonstration. All group members must present a portion of the work (balanced participation is expected).
8. The compression/decompression tool and the report must be submitted via the e-learning platform.

Evaluation

- We will evaluate your feedback/progress during the classes. The idea is for the groups to use the practical class time to, in addition to developing the project, discuss relevant aspects with the teachers.
- We will give extra points to new methodologies/implementations and to high flexibility of the compressor regarding the input data.
- We will evaluate your compression/decompression performance using a custom metric (which will only be shared after delivery). This is a metric that balances in some way an equilibrium between compression capability, compression speed, and decompression speed.
- We will also evaluate your compressor solution (readiness to use, documentation, etc).
- We will evaluate your capability to synthesize the ideas, methodologies, comparisons, and results in the report.

Data

In Moodle, a zipped file is available in the "Project #01" folder:

- **data.zip** — a set of files, with different characteristics, for the benchmark.

FAQ

- **Can we use AI-based tools?** Yes, but we recommend a moderate and responsible use. Keep in mind that *novelty* will be strongly valued throughout the project and during the classes. Moreover, you will be assessed on your ability to clearly explain the implemented approach and the relevant parts of the code.
- **Can we exceed 10 minutes for the presentation or 6 pages for the report (e.g., to include more detail)?** This is *strongly discouraged.* Respecting time and space constraints is part of the evaluation, namely your ability to synthesize and communicate the key ideas efficiently.
- **Can we use code from other authors?** Yes, provided it is properly referenced (author credit is essential). This is particularly relevant for the arithmetic encoder/decoder. The *modeling* module, however, should be implemented by the group.