LIÈGE université
School of Engineering

University of Liège
Faculty of applied science
Academic year 2024-2025

ELEN0060-2: Project 2
Source coding, data compression and channel coding

Pierre Lorenzen S203724

Abdelilah Khaliphi S204896

Contents

1 Implementation

1.1 Question 1

In the implementation for this question, we first define a node class to facilitate the creation and management of the binary tree used in the Huffman coding algorithm.

Secondly, iteratively, we sort the nodes based on their probabilities at each iteration, and we merge the two lowest probabilities. This node created by the merge of the two lowest probabilities ones is then added to the tree with a probability equal to the sum of the two lowest probabilities. The children nodes are the two nodes that were merged. The process is repeated until we have a single node left, which is the root of the tree.

Tirdly, we generate the codes for each symbol by traversing the tree. We start at the root and assign a '0' code to the left child and a '1' code to the right child. We continue this process recursively until we reach a leaf node, at which point we store the generated code for that symbol.

Finally, we reorder the symbols to be consistent with the order provided at the input of the function.

To extend the Huffman code generation to any alphabet size q, we can modify the algorithm to merge the q lowest-probabilities nodes at each step and so build a q-ary tree, where each node has up to q children.

To implement this with a number of input symbols n and output alphabet size of $q \geq 2$, we can

1. Take the n symbols with their probabilities
2. Add Dummy Symbols: If $(n-1)\ mod\ (q-1) \neq 0$, add dummy symbols of probability 0 so that: $(n'-1)\ mod\ (q-1) = 0$

   where n' is the total number of symbols (real + dummy).
3. Maintain nodes sorted by probability.
4. Merge q smallest-nodes.
5. Repeat until only one node is left.
6. Label the edges with 0,1,...,q-1.