Marburg University

Department of Computer Science
Prof. Dr. Sebastian Wild

Date: 2025-11-28
Version: 2025-12-01 23:05

# Sheet 7 for Effiziente Algorithmen (Winter 2025/26)

***Hand In:*** *Until 2025-12-05 18:00, on ILIAS.*

## Problem 1

30 points

Given an alphabet $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$, a pattern `TCCGA`, and the text

`CATGCACTCTCCAGTATCCGA`

Apply the Rabin-Karp algorithm with the following hash function:

$$h(S) = |S|_{\mathtt{A}} + 2 \cdot |S|_{\mathtt{C}} + 3 \cdot |S|_{\mathtt{T}} + 4 \cdot |S|_{\mathtt{G}}.$$

In each step, state the calculated hash value and indicate which letters are actually being compared.

## Problem 2

60 points

Prove the following *No-Free-Lunch* theorems for lossless compression.

1. *Weak version:* For every compression algorithm $A$ and every $n \in \mathbb{N}_{\geq 1}$, there exists an input $w \in \Sigma^n$ for which $|A(w)| \geq |w|$, i.e., the result of the compression is not smaller than the input.

   **Hint:** Try a proof by contradiction. There are several ways to prove this theorem.

2. *Strong version:* For every compression algorithm $A$ and $n \in \mathbb{N}$, the following holds:

   $$\bigl|\{w \in \Sigma^{\leq n} : |A(w)| < |w|\}\bigr| \quad < \quad \tfrac{1}{2} \cdot \bigl|\Sigma^{\leq n}\bigr| \,.$$

   That is, less than half of all possible input lengths (up to $n$) can be compressed so that they are smaller than the original size.

   **Hint:** First determine $|\Sigma^{\leq n}|$.

The theorems apply to any non-unary alphabet, but you can restrict yourself to the binary case, i.e. $\Sigma = \{\texttt{0}, \texttt{1}\}$.

$\Sigma^\star$ is the set of all (finite) strings over the alphabet $\Sigma$. $\Sigma^{\le n}$ is the set of all strings with length $\le n$. We take the domain of (all) compression algorithms to be the set of (all) injective functions $\Sigma^\star \to \Sigma^\star$, i.e., functions that map every possible input string to an output string (encoding), where no two strings are mapped to the same output.

## Problem 3

40 points

Compress the text $T$ = `HANNAHBANSBANANASMAN` using Huffman coding. Show the following steps as your work:

1. the letter frequencies,
2. a step-by-step construction of the Huffman tree,
3. the Huffman code,
4. the coded text.
5. Specify the compression rate of the result
   (ignore the space required to store the Huffman code).

**Important:** Meticulously follow the tie-breaking rules from class:

> 1. To break ties when **selecting** the two **tries** to merge,
>    first use the trie containing the smallest letter in alphabetical order.
> 2. When combining two tries of **different values**,
>    place the lower-valued trie on the left (corresponding to a `0`-bit).
> 3. When combining tries of **equal value**,
>    place the one containing the smallest letter to the left.

## Problem 4

20 points

The given binary string $C$ is the result of encoding a binary string $S$ using run-length encoding. Decode $C$, i.e., provide $S$ and show the steps $b$, $\ell$, $k$ as presented in the lecture.

$$C = \texttt{000100010011100101}$$