Philipps-Universität Marburg
Fachbereich Mathematik und Informatik
Arbeitsgruppe Algorithmik

Wintersemester 2024/2025

Version 2024-12-07 14:59

Prof. Dr. Sebastian Wild
Dr. Nikolaus Glombiewski

Übungen zur Vorlesung

Effiziente Algorithmen

Abgabe: 13.12.2024,
bis **spätestens** 19:00 Uhr
über die ILIAS Plattform

Übungsblatt 7

Aufgabe 7.1: Lempel-Ziv-Welch (3 Punkte)

a) Geben Sie die Encodierung für den Text `TIN␣TIPTIP` mittels der Lempel-Ziv-Welch Encodierung an. Geben Sie als Rechenweg analog zur Vorlesung die Tabelle (nach dem Eintrag 127) an.

s	#(s)
␣	32
...	...
I	73
...	...
N	78
...	...
P	80
...	...
S	83
T	84
...	...

b) Decodieren Sie die Zahlen [65 65 66 129 67 132 131] welche mit Lempel-Ziv-Welch encodiert wurden. Geben Sie als Rechenweg analog zur Vorlesung die Schrittweise Decodierung sowie die Tabelle (nach dem Eintrag 127) an.

s	#(s)
A	65
B	66
C	67
D	68
...	...

Aufgabe 7.2: Text Transformation (4 Punkte)

a) Es sei $T = T[0..9) =$ `ABBACBAAA` der Eingabe Text mit dem Alphabet $\Sigma = \{\texttt{A}, \texttt{B}, \texttt{C}\}$. Verwenden Sie die Move-To-Front Transformation für den Input mit der initialen Queue $Q = [\texttt{A}, \texttt{B}, \texttt{C}]$.

Geben Sie als Rechenweg den Zustand von Q nach jedem Schritt und die Ausgabe an. Falls vorhanden, markieren Sie jeden Run in der Ausgabe und erklären Sie, wie diese zustande kommen.

b) Verwenden sie die *inverse* Burrows-Wheeler-Transformation auf dem encodierten Text `OOO$DOND`, um den ursprünglichen Text zu erhalten. Es gilt: `$` $<$ `D` $<$ `N` $<$ `O`, wobei \$ das Ende des Strings markiert.

Aufgabe 7.3: Arithemtic Coding (2+3) (5 Punkte)

a) Zeigen Sie, dass in unserem allgemeinen Framework (general stochastic sequence) für jeden String $X_0X_1 \dots X_n =$ \$ gilt $m \leq \lg(1/(P_{0,X_0}P_{1,X_1} \dots P_{n,X_n})) + 2$.

b) Entwerfen Sie einen Algorithmus, der eine Folge von n perfekt uniformen, zufälligen Trits generiert, aber als Eingabe nur (perfekt uniforme) zufällige Bits erhält. Ihr Algorithmus sollte für große n in Erwartung höchstens lg(3) Zufallsbits pro Trit benötigen. Begründen Sie die Korrektheit Ihrer Lösung.

Hinweis: Verwenden Sie Arithemtic Coding.

Aufgabe 7.4: Anwendung von Kompression (1+2+3+1+1) (8 Punkte)

In dieser Aufgabe erstellen Sie schrittweise eine proof-of-concept Implementierung einer Kompressions-Pipeline in Java. Es geht hierbei (ausnahmsweise) nicht um die Laufzeit Ihrer Implementierung; insbesondere müssen Sie nicht versuchen, die Transformationen möglichst effizient zu berechnen. Wir werden außerdem die Ausgabe der Einfachheit halber als `String` mit Zeichen '`0`' und '`1`' darstellen.

Erstellen Sie eine Klasse `Compression`, in welcher Sie die folgenden Methoden implementieren.

a) Implementieren Sie eine Methode `String eliasGammaCode(int i)`, welche für eine ganze Zahl > 0 den Elias Gamma Code wie in der Vorlesung vorgestellt berechnet.

b) Implementieren Sie eine Methode `int[] moveToFront(String text)`, welche für einen String die Move-To-Front Transformation aus der Vorlesung umsetzt.

c) Implementieren Sie eine Methode `String burrowsWheelerTransform(String text)`, welche für einen String die Burrows Wheeler Transformation aus der Vorlesung umsetzt. Wählen Sie als *End-of-Text Character* einen geeigneten Buchstaben, um das Ende des Textes zu markieren und geben Sie diesen kenntlich in der Lösung an (z.B. als Kommentar).

d) Implementieren Sie eine Methode `String compress(String text)`, in welcher der Text wie folgt verarbeitet wird:

Burrows Wheeler Transformation → Move-To-Front Transformation → Elias Gamma Code

Beachten Sie dabei:

- Da der Elias Gamma Code nur für $n > 1$ behandelt wurde, können Sie vor der Eingabe die Zahl um 1 inkrementieren. Für eine Dekodierung würde man entsprechend das Ergebnis um 1 dekrementieren.
- Verketten Sie die Ergebnisse des Elias Gamma Codes zu einem Gesamtergebnis.

e) Wählen Sie Ihr Lieblings-Buch aus dem Project Gutenberg (alternativ ihr am wenigsten geliebtes Buch, oder ein beliebiges), welches im *Plain Text UTF-8* Format verfügbar ist. Machen Sie für den Tutor kenntlich, welches Buch gewählt wurde (als Link oder Datei mit überschaubaren Größen abgeben). Wenden Sie in einer `main`-Methode die Kompressions-Pipeline auf das Buch an. Sie dürfen zur Vereinfachung den Text nach dem Einlesen in Bezug auf Sonderzeichen etwas verändern, falls es zu Problemen mit Ihrem *End-of-Text Character* kommt. Geben Sie die Kompressionsrate mit Begründung an.

Hinweis: Eine Dekodierung ist nicht gefordert, aber kann zur Erkennung von Fehlern insbesondere für die Burrows Wheeler Transformation nützlich sein.