

Übungsblatt 8 zur Vorlesung Effiziente Algorithmen (Winter 2025/26)

Abgabe: Bis 2025-12-12 18:00, on ILIAS.

1. Aufgabe

15 + 15 Punkte

- a) Geben Sie die Kodierung für den Text $S = \text{TIN}_{\square}\text{TIPTIP}$ (über dem ASCII-Alphabet) mittels der Lempel-Ziv-Welch Encodierung an. Geben Sie als Rechenweg analog zur Vorlesung die Tabelle (nach dem Eintrag 127) an.

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

- b) Decodieren Sie die Codewörter $C = [65, 65, 66, 129, 67, 132, 131]$ welche mit Lempel-Ziv-Welch (und dem ASCII-Alphabet) kodiert wurden. Geben Sie als Rechenweg analog zur Vorlesung die Schrittweise Dekodierung sowie die Tabelle (nach dem Eintrag 127) an.

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

2. Aufgabe

10 + 10 Punkte

- a) Es sei $T = T[0..9) = \text{ABBACBAAA}$ der Eingabetext über dem Alphabet $\Sigma = \{\text{A, B, C}\}$. Verwenden Sie die Move-To-Front Transformation für den Input mit der initialen Liste $Q = [\text{A, B, C}]$.
- Geben Sie als Rechenweg den Zustand von Q nach jedem Schritt und die Ausgabe an. Falls vorhanden, markieren Sie die Runs in der Ausgabe und erklären Sie, wie diese zustande kommen.
- b) Verwenden sie die *inverse* Burrows-Wheeler-Transformation auf dem kodierten Text $C = \text{000\$DOND}$, um den ursprünglichen Text zu erhalten. Es gilt: $\$ < \text{D} < \text{N} < \text{O}$, wobei $\$$ das Ende des Strings markiert.

3. Aufgabe

20 + 30 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 Arithmetic Coding.

4. Aufgabe

10 + 20 + 30 + 10 + 10 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' (statt einem binären Stream) darstellen.

Erstellen Sie eine Klasse `Compression`, mit den folgenden Methoden.

Code muss (zusätzlich zur Ihrer Beschreibung) als separate, kompilierbare .java-Datei eingereicht werden!

- 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) 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 bzip(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.
 - Konkatenieren 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 sehr nützlich sein.