Aufgabenstellung zur Bachelorarbeit „Approximation der LZ-Zerlegung“

2. Juni 2020

Es sollen einfache, aber möglichst effiziente Varianten des Algorithmus „Approximating LZ77 via Small-Space Multiple-Pattern Matching“ (ESA 2015) in C++ implementiert und in unser Kompressionsframework tudocomp integriert werden. Zusätzlich soll die Qualität der Approximation (Approximationsgüte) sowie der Platz- und Speicherbedarf der Algorithmen evaluiert werden.

Für den Algorithmus bietet sich folgendes zweistufiges Verfahren an für einen Eingabetext $T[0, n)$:

1. Zunächst soll eine $O(\log n)$-Approximation berechnet werden. Dies kann wie folgt geschehen:
   - Beginne mit Textfenstern der Größe $w := 2^k$ für einen frei wählbaren Parameter k.
   - Erweitere den Text (konzeptuell) mit 0-Bytes am Ende, so dass die Länge von T ein Vielfaches von w ist.
   - Berechne die Rabin-Karp-Fingerprints ϕ_i von $T[i, i+w)$, so dass $i < n$ ein Vielfaches von w ist. Speichere diese Fingerprints zusammen mit der Position i in einer Hashtabelle H.
   - Gehe mit einem Fenster der Länge w über den Text und berechne dabei die Fingerprints von $T[j, j + w)$ für $0 \leq j < n - w$. Prüfe für jeden Fingerprint, ob er in der Hashtabelle an einer Stelle $i > j$ vorkommt. Falls ja, ersetze den Substring $T[i, i + w)$ durch eine w-lange Referenz auf die Stelle j. Dies bildet die finalen Faktoren.
   - Für alle bisher nicht faktorisierten Textzeichen wiederhole den Algorithmus mit halbierter Fenstergröße ($w \leftarrow w/2$).
   - Das Verfahren bricht ab, wenn die Fenstergröße eine gewisse Mindestlänge ℓ (wieder ein Parameter) unterschreitet. Die verbleibenden Zeichen werden naiv kodiert.

   In diesem Schritt sollen unterschiedliche Parameterkombinationen von ℓ und k getestet und die finalen Faktorisierungen mit der originalen LZ77-Faktorisierung sowie weiteren in tudocomp vorhandenen Algorithmen verglichen werden. Außerdem sollen verschiedene Hashfunktionen und Has-

htabellenimplementierungen getestet und die Auswirkungen auf Zeit und den Speicherverbrauch gemessen werden.

2. Im nächsten Schritt soll die obige Approximation durch das Mergen benachbarter Faktoren verbessert werden. Hierfür müssen nur die Textbereiche von einer "Cherry" zur nächsten in Betracht gezogen werden (Details s. Paper). Es sollen auch hier besser einfache, aber dafür praktisch schnelle und platzeffiziente Verfahren entwickelt werden, statt sich stur an die theoretische Beschreibung zu halten.

Insgesamt soll in dieser Arbeit der *Algorithm Engineering*-Ansatz umgesetzt werden (s. z.B. „Peter Sanders: Algorithm Engineering - An Attempt at a Definition. Efficient Algorithms 2009: 321-340"). Dabei wird neben einer theoretisch sauberen Beschreibung der Algorithmen auch eine angemessene, übersichtliche Darstellung der Evaluation erwartet.