

Grundlagenpraktikum: Rechnerarchitektur

Gruppe 134 – Abgabe zu Aufgabe A404

Wintersemester 2022/23

Arman Habibi

Larissa Manalil

Andrei Stoica

1 Einleitung

Die von David A. Huffman, im Jahr 1952, entwickelte Huffmankodierung dient zur verlustfreien Datenkompression. Um Redundanz zu minimieren, werden, nach einer Häufigkeitsanalyse der Symbole, diese in einem Binärbaum angeordnet, sodass der entstehende Binärkode, mit welchem das Symbol im Baum zu erreichen ist, präfixfrei ist. Dies bedeutet, dass kein Binärkode Teil eines anderen sein darf und ermöglicht es die Daten ohne Trennzeichen zu speichern. Die Symbole, die häufiger in der Quelldatei auftauchen, befinden sich weiter oben im Baum und brauchen daher auch weniger Bits zum abspeichern. Das Verfahren ist soweit Optimal, da es keinen anderen symbolbasierten Weg gibt, der einen kürzeren Code erzeugen könnte, insofern die Häufigkeiten der Symbole bekannt sind. [1] Im folgenden gilt es diesen Algorithmus in der Programmiersprache C zu implementieren und ein Rahmenprogramm zu erstellen, welches es ermöglicht Dateien zu komprimieren und zu dekodieren.

2 Lösungsansatz

2.1 Encode

Zuerst wird die Häufigkeitsanalyse der Symbole durchgeführt. Dafür wird ein Array mit Pointern auf Nodes für jedes ASCII Zeichen angelegt, welche die Frequenz des Zeichens speichern, sowie das Zeichen für die spätere Verwendung. Das Zählen erfolgt hierbei mit einer Effizienz von $\mathcal{O}(1)$, da der Array-Index dem ASCII-Code entspricht. Zusätzlich wird Speicher gespart, indem die Nodes erst erstellt werden sobald das zugehörige ASCII Zeichen das erste mal gelesen wird.

Als Beispiel wird der String *ABRACADABRAB* komprimiert, welcher die folgenden Häufigkeiten enthält. Nicht vorhandene Zeichen wurden vernachlässigt.

A	B	C	D	R
5	3	1	1	2

Anschließend wird ein Min-Heap erstellt und die erstellten Nodes darin eingefügt. Hier fängt der von Huffman entwickelte Algorithmus an. [2] Solange mehr als ein Node im Min-Heap ist, werden die beiden Nodes mit der geringsten Frequenz aus dem Min-Heap entfernt und an einem neuen Node angehängt (mit einem Null Zeichen), welcher die Summe der beiden Frequenzen als Wert erhält. Dieser wird wieder in den Min-Heap

eingefügt. Am Ende bleibt nur noch ein Node im Min-Heap übrig, welcher den fertigen Huffman-Baum enthält.

2.2 Decode

3 Korrektheit/Genauigkeit

4 Performanzanalyse

5 Zusammenfassung und Ausblick

Literatur

- [1] David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.
- [2] Alistair Moffat. Huffman coding. *ACM Comput. Surv.*, 52(4), aug 2019.