Bonus Problem

Benedikt Sosnowksy, Daniel Ulrich, Kharald Anager, Marcel Sinn
die Matrikelnummern: 22107862; 00821114;22100539; 22107861;

June 5, 2022

1 Einleitung

Im folgenden Bericht werden wir darstellen, wie sich Mustererkennungsalgorithmen unterscheiden. Dafür werden wir insbesondere auf Brute-Force, KMP und Boyer-Moore eingehen und verschiedene Vor- und Nachteile aufzeigen.

Das Problem in unserem Beispiel der Mustererkennung ist, dass wir einen bestimmten String in Form eines Textes der Länge n haben und innerhalb dieses Strings einen Muster-String suchen, welcher Teil des Textes sein soll. Unsere Mustererkennungsalgorithmen haben nun die Aufgabe, möglichst schnell und effizient den gesuchten String zu finden. Diese Mustererkennung wird aktuell beispielsweise in PDFs oder verschiedenen Websites angewendet, um die Seite nach bestimmten Wörtern zu durchsuchen. Wie genau diese Suche funktioniert und was die einzelnen Mustererkennungsalgorithmen machen, wird in den folgenden Teilen thematisiert.

2 Algorithmen

2.1 Brute-Force

Das Brute-Force-Entwurfsmuster für Algorithmen ist eine leistungsstarke Technik für den Entwurf von Algorithmen, wenn wir etwas suchen oder eine Funktion optimieren wollen. Bei der Anwendung dieser Technik in einer allgemeinen Situation werden alle möglichen Anordnungen der beteiligten Eingaben aufgezählt und die beste von allen diesen Konfigurationen ausgewählt.

2.1.1 Laufzeit

Die Laufzeit des Brute-Force-Mustervergleichs ist im schlimmsten Fall sehr schlecht, da wir für jeden Kandidatenindex im Text T bis zu m Zeichenvergleiche durchführen können, um herauszufinden, dass P im aktuellen Index nicht mit T übereinstimmt. Die äußere for-Schleife wird bis zu n-m+1 Mal und die innere while-Schleife höchstens bis zu m Mal ausgeführt. Somit ist die Worst-Case Zeitkomplexität der Brute-Force-Methode also O(nm).

2.2 Knuth-Morris-Pratt

Der Knuth-Morris-Pratt (oder "KMP") Algorithmus versucht, durch Vorwissen über den String die nächsten Charaktere im Such-String zu überspringen. So kann, wenn bei der Suche nach "kokosnuss" am Index 5 eine Nicht-Übereinstimmung auftritt, sicher angenommen werden, dass eine Verschiebung des Match-Strings um nur 1 ein "k" mit einem "o" verglichen werden wird. Bei einer Verschiebung um 2 kann eine Nicht-Übereinstimmung nicht mehr ausgeschlossen werden, sodass wir die Suche an dieser Stelle fortsetzen müssen. Diese Vorgehensweise führt zu einer Zeitkomplexität von O(n+m).

2.2.1 Fehlerfunktion

Diese Fehlerfunktion kann für jeden String im vorraus berechnet werden. Diese Funktion gibt für jeden Index k, an dem eine Nicht-Übereinstimmung auftritt, die Anzahl der Zeichen zurück, die übersprungen werden können. Dieser Wert ist jeweils die Länge des grössten Präfixes, dass ein Suffix des Substrings der Länge k ist.

2.3 Boyer – Moore Algorithmus

Der Boyer-Moore Mustererkennungsalgorithmus kann Vergleiche zwischen einem Muster P und einem Bruchteil der Buchstaben in einem Text T vermeiden. In diesem Teil wird eine vereinfachte Version des Original Algorithmus von Boyer und Moore beschrieben. Die Hauptidee des Boyer-Moore Algorithmus ist die Verbesserung der Laufzeit des brute-force Algorithmus, indem zwei möglichst zeitsparende Heuristiken hinzugefügt werden.

Die erste Heuristik ist die Looking-Glass Heuristic. Bei dieser Methode wird auf eine mögliche Platzierung von P gegen T getestet. Der Vergleich wird mit dem Ende von P begonnen und läuft rückwärts an den Beginn von P.

Die zweite Heuristik ist die Character-Jump Heuristic. Während eine mögliche Platzierung von P in T getestet wird, wird eine Nichtübereinstimmung der Buchstaben T[i]=c mit dem korrespondierenden Muster P[k] wie folgt behandelt. Falls c nicht in P vorkommt, wird P komplett bis an T[i] vorbei geschoben. Andernfalls wird P solange verschoben, bis eine Übereinstimmung der Buchstaben c in P in T[i] auftritt.

3 Textbeschreibung

In dieser Untersuchung wurden fünf Textdateien genutzt:

1. bigfile.txt ist eine 10.000.000 Zeichen große Datei an zufälligen Buchstaben und Ziffern.

2. shakespeare.txt enthält die gesammelten Werke desselben in altem Englisch, mit einigen Kommentaren an Beginn und Ende. Die gesamte Länge der Datei beträgt 5,7 Millionen Zeichen.

3. law.txt enthält die Richtlinien zur Durchsuchung und Beschlagnahmung von Computern des US Department of Justice. Das Dokument ist in modernen Englisch abgefasst und enthält ca. 300.000 Zeichen

4. In LI.txt ist ein computergenerierter, pseudolateinischer Text enthalten, der 750.000 Zeichen umfasst.

5. LI-large.txt ist im selben Stil wie LI.txt geschrieben und hat eine Länge von 2.000.000 Zeichen

4 Die Präsentation der Ergebnisse

Bei allen drei Algorithmen wurde die tatsächliche Laufzeit überprüft. Die Ergebnisse in sind der Tabelle 1 zusammengefasst.

	bigfile	law	LI	LI-Large	Shakespeare
brute-force	1647.72 ms	53.01 ms	125.04 ms	348.18 ms	905.94 ms
KMP	1379.80 ms	41.01 ms	112.02 ms	303.05 ms	817.72 ms
Boyer-Moore	1062.24 ms	34.01 ms	97.02 ms	233.07 ms	612.19 ms

Table 1: Ergebnisse der Laufzeit

Aufgrund der erhobenen Angaben können wir schließen, dass der Boyer-Moore Algorithmus die besten Ergebnisse liefert. Der Brute-Force Algorithmus besitzt die schlechteste Laufzeit. Es muss darauf hingewiesen werden, dass KMP Algorithmus kein bedeutender Fortschritt im Vergleich zu Brute-Force ist, da die Laufzeiten sich in der Anwendung nicht signifikant unterscheiden.

5 Fazite und Perspektive

Der Boyer-Moore Algorithmus war bei den verschiedenen Tests immer der Beste. Daraus können wir schließen, dass der Boyer-Moore Algorithmus, abgesehen von der Größe der Textdateien, im Durchschnitt immer schneller ist.

Die Algorithmen lassen sich weiter vergleichen, indem man überprüft, welcher Algorithmus mehr Ressourcen verbraucht, weil der Algorithmus auf einem sehr leistungsschwachen CPU ausgeführt werden kann. Idealerweise vergleicht man die Algorithmen im Zusammenhang mit den allen Parametern. Das heißt: Resourcenverbrauch im Bezug auf Wichtigkeit der Effizienz, Laufzeit je nach Größe der Dateien die geprüft werden sollen und die Wahrscheinlichkeit, wie häufig der "Worstcase" in der Anwendung auftritt.