Algorithmen und Datenstrukturen SoSe25

-Assignment 9-

Moritz Ruge
Matrikelnummer: 5600961
Lennard Wittenberg
Matrikelnummer: —

Juni 2025

1 Problem: Suchen in Zeichenketten I

Implementieren Sie den naiven Algorithmus und den Algorithmus von Rabin-Karp zur Suche in Zeichenketten. Finden Sie dann heraus, wie oft das Wort whale in Moby Dick vorkommt (ignorieren Sie dabei Groß- und Kleinschreibung). Wie schneiden Ihre Implementierungen im Vergleich ab? Hinweis: Den Roman Moby Dick finden Sie unter
http://www.gutenberg.org/files/2701/2701-0.txt.

2 Problem: Suchen in Zeichenketten II

2.1 Rabin-Karp mit mehreren Suchmustern

Der Algorithmus von Rabin-Karp lässt sich leicht auf mehrere Suchmuster verallgemeinern. Gegeben eine Zeichenkette s und Suchmuster $t1, \ldots, tk$, bestimme die erste Stelle in s, an der eines der Muster $t1, \ldots, tk$ vorkommt. Beschreiben Sie, wie man den Algorithmus von Rabin-Karp für diese Situation anpassen kann. Was ist die heuristische Laufzeit Ihres Algorithmus (unterder Annahme, dass Kollisionen selten sind)?

2.1.1 Probelmstellung:

Gegeben:

- Eine Zeichenkette s (Text) der Länge n
- Ein Suchmuster k mit $t_1, t_2, \ldots, t_k$ der gleichen Länge m

2.1.2 Gesucht:

- Ein Algorithmus: der die erste Position in s (Text), an der irgendeins der Muster $t_1, \ldots, t_k$ vorkommt.
- Die Laufzeit des Algorithmus (unter der Annahme, dass Kollisionen selten sind)

2.1.3 Lösung:

Rabin-Karp vorgehen:

- Wir berechnen den Hashwert des Musters t
- Wir Iterieren über den Text s mit einem Fenster/Bereich der Länge m
- Berechne den Hashwert des aktuellen Fensters $s[i \ldots i + m - 1]$
- Wenn die Hashwerte übereinstimmen, vergleichen wir den Text-ausschnitt und Muster direkt, um Kollisionen zu umgehen

Rabin-Karb für mehrere Muster: [1]

- Wir berechnen den Hashwert für alle Suchmuster $t_1, t_2, \ldots, t_k$
  - Diese Hashwerte speichern wir in einer Datenstruktur (HashSets)
  - Dadurch ist die Überprüfung, ob ein Hashwert zu einem Muster gehört, in $O(1)$ möglich
- Wiederholen des normalen Algorithmus, Iterieren über alle Teilstrings der Länge m im Text s
- Berechnen des aktuellen Hashwertes vom Fenster
- Vergleichen, ob dieser Hashwert in der Menge der HashSets zu finden ist
- Wenn die Hashwerte übereinstimmen, verlgeichen wir wieder direkt

Eigenschaften eines HashSets in Scala: Eine HashSet-Struktur ist eine Datenstruktur, die eine Menge von eindeutigen Werten speichert und sehr schnelle Einfüge-, Such- und Löschoperationen erlaubt - im Schnitt in konstanter Zeit $o(1)$

- HashSets haben die Eigentschaft keine Duplikate zu erlauben, d.h. jeder Wert wird nur einmal gespeichert, doppelte werden ignoriert
- Schnelle Suche von Werten (sofern keine Kollision)
- Ein HashSet verwendet intern eine Hashfunktion um die Werte zu speichern und zu finden

Um HashSets zu benutzten importieren wir folgende Bibliothek:

```
import scala.collection.mutable.HashSet
```

Pseudocode könnte wie folgt aussehen:

```
import scala.collection.mutable.HashSet

// Hashfunktion mit Rolling Hash
def hash(s: String): Int = ...

val musterHashes = HashSet[Int]() // Initialisiere HashSet
val muster = List("bob", "tim", "leo") // Suchmuster s[i ... i+m-1]
val m = muster.length // m = Laenge des Musters bei unterschiedlicher musterlaenge
    sollte m die wenigsten charaktere haben

// Rabin-Karb vorgehen fuer mehrere Muster:
// 1. Wir berechnen den Hashwert fuer alle Suchmuster t1,t2,...,tk
for muster <- muster do
  musterHashes.add(hash(muster)) //speicher die Hashwerte im Set

// wir Iterieren ueber den Text s mit der Fenstergroesse von m
for i <- 0 to s.lenght - m do
  val fenster = s.fenster(i, i + m)
  cal fensterHash = hash(fenster)

  if musterHashes.contains(fensterHash) then
    // Direkte kontrolle ob die muster(string) und Text uebereinstimmen
    if muster.contains(fenster) then
      return i
```

Heuristische Laufzeit: TODO

2.2 Implimentierung des Algorithmus

Implementieren Sie Ihren Algorithmus aus 2.1. Beantworten Sie sodann folgende Frage: Was kommt öfter in dem Roman Sense & Sensibility vor: sense oder sensibility/sensible?
Hinweis: Siehe http://www.gutenberg.org/files/161/161-0.txt.

3 Problem: Suche in Zeichenketten III

Sei $\sum = C, G, T, A$. Sei $s = CTTGGATTA$ und $t = TTA$.

3.1 Naiver Algorithmus

Verwenden Sie den naiven Algorithmus, um festzustellen, ob/wo das Muster t in der Zeichenkette s vorkommt. Zeigen Sie die einzelnen Schritte.

3.2 Rabin-Karp Algorithmus

Verwenden Sie den Algorithmus von Rabin-Karp, um festzustellen, ob/wo das Muster t in der Zeichenkette s vorkommt. Verwenden Sie A : 0, T : 1, G : 2, C : 3 und die Primzahl 5 als Modulus für die Hashfunktion. Zeigen Sie die einzelnen Schritte.

3.3 Knuth-Morris-Pratt Algorithmus

Verwenden Sie den Algorithmus von Knuth-Morris-Pratt, um festzustellen, ob/wo das Muster t in der Zeichenkette s vorkommt. Zeigen Sie die einzelnen Schritte.

References

[1] Nick Dandoulakis-User. *Using Rabin-Karp to search for multiple patterns in a string.* URL: https://stackoverflow.com/questions/1318126/using-rabin-karp-to-search-for-multiple-patterns-in-a-string. (accessed: 25.06.2025).