11. Aufgabenblatt zur Vorlesung

Algorithmen und Programmierung III

Wolfgang Mulzer WiSe 2013/14

Abgabe am 16. Januar 2014 vor der Vorlesung in die jeweiligen Tutorenfächer

Aufgabe 1 Editierabstand *10 Punkte*

Der *Editierabstand* zwischen zwei Zeichenketten s und t ist die minimale Anzahl von *Editieroperationen*, um s nach t zu überführen. Es gibt drei Editieroperationen: (i) Einfügen eines Zeichens; (ii) Löschen eines Zeichens; und (iii) Ersetzen eines Zeichens durch ein anderes. Zum Beispiel beträgt der Editierabstand zwischen "APFEL" und "PFERD" drei: Lösche A, ersetze L durch R, füge D ein.

Beschreiben Sie einen Algorithmus, der den Editierabstand zwischen zwei Zeichenketten s und t in $O(k\ell)$ Zeit berechnet, wobei s Länge k und t Länge ℓ hat. Erklären Sie außerdem, wie man eine optimale Folge von Editieroperationen findet.

Hinweis: Benutzen Sie dynamisches Programmieren analog zum LCS-Problem: betrachten Sie das jeweils letzte Zeichen in s und t und unterscheiden Sie drei Möglichkeiten: (a) überführe s nach t' und füge dann ein Zeichen an; (b) überführe s' nach t und lösche dann ein Zeichen; (c) überführe s' nach t' und ersetze dann ein Zeichen, falls nötig. Hierbei bezeichnen s' und t' jeweils s und t ohne den letzten Buchstaben.

Aufgabe 2 Suchen in Zeichenketten *10 Punkte*

(a) 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.txt`.

(b) Der Algorithmus von Rabin-Karp lässt sich leicht auf *mehrere* Suchmuster verallgemeinern. Gegeben eine Zeichenkette s und Suchmuster $t_1, \ldots, t_k$, bestimme die erste Stelle in s, an der eines der Muster $t_1, \ldots, t_k$ vorkommt. Beschreiben Sie, wie man den Algorithmus von Rabin-Karp für diese Situation anpassen kann. Was ist die heuristische Laufzeit Ihres Algorithmus (unter der Annahme, dass Kollisionen selten sind)?

Hinweis: Nehmen Sie zunächst an, dass alle Suchmuster die gleiche Länge haben. Der allgemeine Fall lässt sich darauf zurückführen, z.B., indem man zum Hashen alle Suchmuster auf die Minimallänge verkürzt.

(c) (*freiwillig, 5 Zusatzpunkte*) Implementieren Sie Ihren Algorithmus aus (b). 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.txt`.

Aufgabe 3 Tries *10 Punkte*

(a) Zeichnen Sie einen unkomprimierten und einen komprimierten Trie für die Wörter {ALGORITHMUS, TRIE, BAUM, TORUS, BAHN, TORPEDO}.

(b) Entwickeln Sie einen Algorithmus, der alle Wörter in einem unkomprimierten Trie ausgibt und dabei jede Kante höchstens zweimal besucht.