



Informatik 2 - Sommersemester 2018

Übungsblatt 10

Abgabe: Freitag, 20. Juli, 14:00 Uhr

Aufgabe 1: Editierdistanz

(13 Punkte)

Seien $A = a_1 \dots a_n, B = b_1 \dots b_m$ zwei Wörter. Für $k \leq n, \ell \leq m$ seien $A_k = a_1 \dots a_k, B_\ell = b_1 \dots b_\ell$ die Präfixe von A und B . Sei wie in der Vorlesung $D_{k,\ell} := D(A_k, B_\ell)$ die Editierdistanz zwischen den beiden Präfixen A_k, B_ℓ . Es gilt $D(A, B) = D_{n,m}$.

- Betrachten Sie die Wörter $A = \text{ananas}$ und $B = \text{bananen}$. Berechnen Sie *alle* $D_{k,\ell}$ bezüglich einheitlichen Kosten von 1 für jede Operation (löschen, einfügen, ersetzen) für $1 \leq k \leq n, 1 \leq \ell \leq m$ mittels dem rekursiven Ansatz und Memoisation, indem Sie wie in der Vorlesung eine entsprechende Tabelle (bottom-up) ausfüllen. Markieren Sie $D(A, B)$. (3 Punkte)
- Wiederholen Sie (a) mit erhöhten Kosten 2 für das Ersetzen. (3 Punkte)
- Sei 3 die Kosten ein Zeichen aus $\{e, n, s\}$ mit einem anderen aus dieser Menge zu ersetzen. Ersetzen-Operationen in denen mindestens ein anderes Zeichen $\notin \{e, n, s\}$ involviert ist, kosten 1. Einfügen und Löschen kostet 2. *Beweisen oder widerlegen* Sie die Korrektheit des dynamischen Programmieransatzes zur Berechnung von $D(A, B)$ für beliebige Wörter A, B für diese Kosten. (5 Punkte)
- Das Löschen eines Zeichens ist aus ungeklärten technischen Gründen nicht mehr möglich. Passen Sie die Rekursionsgleichung für $D_{k,\ell}$ an, um die Editierkosten für die Überführung eines gegebenen Wortes A in ein Wort B ohne Löschen zu berechnen. Gehen Sie davon aus, dass $|A| \leq |B|$ ist und das die Kosten für das Einfügen und Ersetzen 1 betragen. (2 Punkte)

Aufgabe 2: String Matching

(7 Punkte)

Gegeben sei das Muster $P = ABABAA$ und der Text $T = BBABAABABAABABABAABA$.

- Berechnen Sie das Array S des Knuth-Morris-Pratt Algorithmus'. (3 Punkte)
- Finden Sie mithilfe des Knuth-Morris-Pratt Algorithmus alle Vorkommen von P in T . Die Schritte des Algorithmus sollen klar erkennbar sein (analog zum Beispiel aus der Vorlesung). (4 Punkte)

Aufgabe 3: Approximate String Matching

(10 Punkte)

Implementieren Sie den Algorithmus *Approximate String Matching* aus der Vorlesung. Dieser erhält einen Suchtext T und ein Muster P als Eingabe und sucht einen Teilstring von T der die Editierdistanz zu P minimiert. Gehen Sie von Kosten 1 für alle Operationen aus. Führen Sie Ihren Algorithmus mit dem Suchtext `text.txt` und den Mustern aus die in `patterns.txt` gegeben sind (getrennt durch Zeilenumbrüche). Ihr Algorithmus soll für *jedes* Muster, den Substring des Suchtextes für welches die Editierdistanz D minimiert wird in eine Datei `result.txt` schreiben (getrennt durch Zeilenumbrüche).

Hinweis: $E(i, b)$ entspricht der minimalen Editierdistanz zwischen P_i und einem $T_{r,b}$ (vergleiche Vorlesung). Wir empfehlen eine $|P| \times |T|$ -Tabelle $E(i, b)$ bottom-up zu füllen. Die minimale Distanz zwischen Muster P und einem Teilstring des Suchtextes T ist dann $\min_{0 \leq b \leq |T|} E(|P|, b)$.