Prof. Dr. Christian Sohler
Nicole Funk, Dr. Matthijs Ebbens, Sebastian Zaun

Sommersemester 2023

7. Übungsblatt

zur Vorlesung

Grundzüge der Informatik I

Abgabe über Ilias bis zum 24.5. 14:00 Uhr.
Besprechung in Kalenderwoche 23.

Aufgabe 1 *Dynamische Programmierung (3 + 2 + 2 + 2 + 2 Punkte)*

Bob hat Angst vorm Klimawandel und betrachtet die Höchsttemperaturen der letzten Jahre. Er möchte die steigende Tendenz der Temperaturen beobachten. Einige kalte Zeiten dazwischen interessieren nicht, da die Tendenz für Bob interessant ist.

Dafür notiert er die Temperaturen in zeitlicher Reihenfolge in einem Array $A = [a_1, \ldots, a_\ell]$. Darin sucht er nach einer längsten aufsteigenden Teilsequenz. Dies ist eine Teilsequenz mit maximaler Länge, deren Elemente aufsteigend sortiert sind. Dabei muss die Sequenz nicht zusammenhängend sein.

a) Es bezeichne $T(i)$ die Länge der längsten aufsteigenden Teilfolge im Teilarray $[a_1, \ldots, a_i]$, $1 \leq i \leq \ell$, die an Position i endet. Geben Sie eine Rekursionsgleichung für $T(i)$ an und erklären Sie diese.

b) Entwerfen Sie basierend auf der Rekursionsgleichung aus a) einen Algorithmus in Pseudocode, der mit dynamischer Programmierung bei Eingabe eines Arrays $A = [a_1, \ldots, a_\ell]$ die Länge einer längsten aufsteigenden Teilsequenz ausgibt. Kommentieren Sie Ihren Algorithmus mit Bezug zu Ihrer Rekursionsgleichung aus a).

c) Analysieren Sie die asymptotische Worst-Case-Laufzeit Ihres Algorithmus.

d) Beweisen Sie die Korrektheit Ihres Algorithmus mithilfe einer Schleifeninvariante.

e) Geben Sie nun einen Algorithmus in Pseudocode an, der die in b) berechnete Lösung rekonstruiert und eine längste aufsteigende Teilsequenz ausgibt. Wandeln Sie dafür Ihren Algorithmus aus b) ab, indem Sie statt der Länge die Teilsequenz zurück geben. Erklären oder kommentieren Sie Ihren Algorithmus.

Aufgabe 2 *Dynamische Programmierung (3 + 2 + 2 + 2 Punkte)*

Seien $x = (x_1, \ldots, x_m) \in \{\texttt{0}, \texttt{1}\}^m$ und $y = (y_1, \ldots, y_n) \in \{\texttt{0}, \texttt{1}\}^n$ zwei Bitstrings. Die *Editierdistanz* zwischen x und y sei definiert als die minimale Anzahl der Operationen, um x in y umzuwandeln: ein Bit einfügen, ein Bit löschen, eine `0` in eine `1` umwandeln oder umgekehrt eine `1` in eine `0` umwandeln.

a) Geben Sie eine Rekursionsgleichung $E(i, j)$ zur Berechnung der Editierdistanz zwischen $(x_1, \ldots, x_i)$ und $(y_1, \ldots, y_j)$, $1 \leq i \leq m$, $1 \leq j \leq n$, an und erklären Sie diese.

Hinweis: Betrachten Sie die aktuellen Bits x_i und y_j.

b) Zeigen Sie die Korrektheit der Rekursionsgleichung aus a).

c) Entwerfen Sie mithilfe der Rekursionsgleichung aus a) einen Algorithmus in Pseudocode, der mit dynamischer Programmierung bei Eingabe zweier Arrays $X = [x_1, \ldots, x_m]$ und $Y = [y_1, \ldots, y_n]$ mit Einträgen in $\{\texttt{0}, \texttt{1}\}$ die Editierdistanz zwischen den Bitstrings x und y ausgibt. Kommentieren Sie Ihren Algorithmus mit Bezug zu Ihrer Rekursionsgleichung aus a).

d) Analysieren Sie die asymptotische Worst-Case-Laufzeit Ihres Algorithmus.