Algorithmen und Datenstrukturen

Dr. Martin Moritz Kleine

Aufgabenblatt 4 : Entwurf effizienter Algorithmen

Bei den folgenden Aufgaben geht es darum, effiziente Algorithmen zu entwickeln. Als Programmiersprache verwenden Sie entweder Dafny oder JavaScript.

Dafny: Sichern Sie die Korrektheit Ihrer Implementierung durch aussagekräftige Vor- und Nachbedingungen zu. Die Laufzeitkomplexität ist unter Verwendung eines *ghost* Zählers zu verifzieren. Den Speicherverbrauch Ihrers Codes erläutern Sie im Kommentar an den zu implementierenden Dafny Methoden. Die Abgabe erfolgt wie bei den vorherigen Aufgabenblättern als Einsendung eines Permalinks.

JavaScript: Die Korrektheit Ihrer Implementierung ist durch aussagekräftige Tests abzusichern. Um die Laufzeitkomplexität zu verdeutlichen, führen Sie Zählvariablen mit und stellen die Messwerte graphisch dar. Den Speicherverbrauch Ihrers Codes erläutern Sie im Kommentar an den zu implementierenden JavaScript Funktionen. Die Abgabe erfolgt wie beim ersten Aufgabenblatt als Einsendung einer HTML Datei, basierend auf der bekannten Vorlage.

Übungsaufgabe 4.1: *Summenpaare*

Gegeben sei eine Sequenz S von ganzen Zahlen und eine ganze Zahl K. Zu berechnen ist die Anzahl der Paare von Elementen (P, Q) aus S für die gilt:

$$P + Q = K$$

Die Implementierung soll in $O(|S| \log(|S|))$ laufen und höchstens $O(|S|)$ zusätzlichen Speicherbedarf haben.

Übungsaufgabe 4.2: *Mittelpunkte*

Gegeben sei eine Sequenz S von ganzen Zahlen. Zu berechnen ist ein Index i für den gilt, dass die Summe der Elemente links gleich der Summe der Elemente rechts von i ist:

$$\Sigma_{0 \leq j < i} S[j] = \Sigma_{i < k < |S|} S[k]$$

Der Rückgabewert -1 zeigt an, dass es keinen solchen Index gibt. Die Implementierung soll in $O(|S|)$ laufen und höchstens $O(|S|)$ zusätzlichen Speicherbedarf haben.

Übungsaufgabe 4.3: *Längste Pfade ohne Richtungswechsel*

Gegeben sei ein binärer Baum $T = (V, E)$ und die Funktionen $r, l : V \to V \cup \{nil\}$, die zu einem Knoten das jeweils rechte oder linke Kind bestimmen (nil bedeutet, dass der Knoten kein rechtes bzw. linkes Kind hat). Zu berechnen ist die maximale Distanz zwischen den Knoten v_0, v_n für die gilt, dass sie auf einem Pfad von der Wurzel des Baums zu einem Blatt liegen und zusätzlich gilt

$$\forall i : 0 \leq i < n \bullet l(v_i) \,\texttt{in}\, (v_0, \ldots, v_n) \;\text{ oder }\; \forall i : 0 \leq i < n \bullet r(v_i) \,\texttt{in}\, (v_0, \ldots, v_n)$$

Die Implementierung soll in $O(V)$ laufen und höchstens $O(V)$ zusätzlichen Speicherbedarf haben.

Übungsaufgabe 4.4: *Längste Pfade mit aufsteigendem Gewicht*

Zu berechnen ist die Länge n des längsten Pfades $(v_0, \ldots, v_n)$ eines gerichteten Graphen $G = (V, E)$ mit Gewichtsfunktion $w : E \to \mathbb{R}$ für den gilt, daß die Kantengewichte streng monoton zunehmen:

$$\forall i : 0 < i < n \bullet w(v_{i-1}, v_i) < w(v_i, v_{i+1})$$

Der Graph erlaubt Kanten der Form (v, v).

Die Implementierung soll in $O(V + E \log(E))$ laufen und höchstens $O(V + E)$ zusätzlichen Speicherbedarf haben.