

Übungen zur Vorlesung
Algorithmen und Datenstrukturen
Übungsblatt 3



ARBEITSGRUPPE KRYPTOGRAPHIE UND KOMPLEXITÄTSTHEORIE

Prof. Dr. Marc Fischlin
Dr. Christian Janson
Patrick Harasser
Felix Rohrbach

Sommersemester 2019

Veröffentlicht am: 03.05.2019
Letzte Änderung am: 08.05.2019
Abgabe am: 17.05.2019, 12:00 Uhr

P1 (Gruppendiskussion)

Nehmen Sie sich etwas Zeit, um die folgenden Fachbegriffe in einer Kleingruppe zu besprechen, sodass sie anschließend in der Lage sind, die Begriffe dem Rest der Übungsgruppe zu erklären:

- (a) Rekursionsgleichungen, Substitutionsmethode und Mastermethode
- (b) Verkettete Listen
- (c) Stacks und Queues

P2 (Substitutionsmethode)

Zeigen Sie mithilfe der Substitutionsmethode folgende Aussagen über Rekursionsgleichungen:

- (a) $T(n) = T(n - 1) + n$ ist in $O(n^2)$
- (b) $T(n) = T(\lceil n/2 \rceil) + 1$ ist in $O(\log_2(n))$
- (c) $T(n) = 2T(\lfloor n/2 \rfloor) + n$ ist in $O(n \log_2(n))$

P3 (Anwendungen des Mastertheorems)

Begründen Sie für jede der folgenden Rekursionsgleichungen $T(n)$, ob Sie das Mastertheorem anwenden können oder nicht. Benutzen Sie gegebenenfalls das Mastertheorem, um eine asymptotische Schranke für $T(n)$ zu bestimmen.

- (a) $T(n) = 3T(n/2) + n^2$;
- (b) $T(n) = 4T(n/2) + n^2$;
- (c) $T(n) = 2^n T(n/2) + n^n$;
- (d) $T(n) = \frac{1}{2}T(n/2) + 1/n$;
- (e) $T(n) = \sqrt{2}T(n/2) + \log(n)$;
- (f) $T(n) = 64T(n/8) - n^2 \log(n)$;
- (g) $T(n) = 4T(n/2) + \frac{n}{\log(n)}$;
- (h) $T(n) = 2T(n/2) + \frac{n}{\log(n)}$;
- (i) $T(n) = 6T(n/3) + n^2 \log(n)$.

P4 (Stacks)

- (a) Sei S ein Stack, der auf einem (anfangs leeren) Array der Größe 6 implementiert ist. Stellen Sie das Array nach jeder der folgenden Operationen dar: $\text{push}(S, 4)$, $\text{push}(S, 1)$, $\text{push}(S, 3)$, $\text{pop}(S)$, $\text{push}(S, 8)$, $\text{pop}(S)$.
- (b) Erklären Sie, wie Sie zwei Stacks auf einem gemeinsamen Array A der Größe n implementieren können, sodass keiner der Stacks überläuft, solange die Summe der Elemente in beiden Stacks nicht größer ist als n . Zudem sollen $\text{push}()$ und $\text{pop}()$ weiterhin in $O(1)$ sein.

Hausübungen

In diesem Bereich finden Sie die theoretischen Hausübungen von Blatt 3. Bitte beachten Sie die allgemeinen Hinweise zu den Hausübungen und deren Abgabe im Moodle-Kurs! **Denken Sie bitte daran, dass Ihre Lösungen nachvollziehbar und entsprechend ausführlich dargestellt und begründet werden sollen.**

Bitte reichen Sie Ihre Abgabe bis spätestens **Freitag, 17.05.2019, um 12:00 Uhr** ein. Verspätete Abgaben können **nicht** berücksichtigt werden.

H1 (Quicksort)

(1+2+2 Punkte)

- Wie ist die Laufzeit von QUICKSORT, wenn alle Elemente des Eingabearrays A den gleichen Wert haben (und immer das letzte Element eines Teilarrys als Pivotelement verwendet wird)? Begründen Sie Ihre Angabe.
- Illustrieren Sie die von QUICKSORT vorgenommene Zerlegung anhand des Arrays $A = [16, 3, 5, 10, 7, 2, 19, 8, 20, 17, 11]$. Verwenden Sie dabei als Pivotelement stets das letzte Element der Teilarrys. Geben Sie auch das fertig sortierte Array an.
- Geben Sie zwei Arrays A, B an, für die die Laufzeit von QUICKSORT dem besten und dem schlechtesten Fall entsprechen. Dabei sollen A und B die Elemente $[16, 3, 5, 10, 7, 2, 19, 8, 20, 17, 11]$ enthalten. Das Pivotelement ist stets das letzte Element eines Arrays. Geben Sie jeweils die Anzahl der Vergleiche an, die QUICKSORT benötigt, um die Arrays zu sortieren.

H2 (Modale Arrays)

(3+1+1 Punkte)

Wir bezeichnen ein Array $A = A[0..n - 1]$ paarweise verschiedener ganzer Zahlen als *modal*, wenn es einen Index $0 \leq m \leq n - 1$ gibt, sodass die Zahlen in $A[0..m]$ aufsteigend sortiert sind (also $A[j] < A[j + 1]$ für alle $0 \leq j < m$) und die Zahlen in $A[m..n - 1]$ absteigend sortiert sind (also $A[j] > A[j + 1]$ für alle $m \leq j < n - 1$).

- Geben Sie einen Algorithmus mit Laufzeit $O(\log n)$ an, der als Eingabe ein modales Array A bekommt und den Index m wie oben ausgibt. Beweisen Sie die Korrektheit Ihres Algorithmus und geben Sie eine asymptotische Abschätzung für dessen Laufzeit.

Ein Polygon ohne Selbstüberschneidungen heißt *konvex*, falls alle Innenwinkel kleiner als 180° sind. Wir repräsentieren ein konvexes Polygon mit n Eckpunkten mittels zwei Arrays $V = V[0..n - 1]$ und $W = W[0..n - 1]$, wobei $(V[i], W[i])$ die ebenen Koordinaten des i -ten Eckpunktes sind ($0 \leq i \leq n - 1$). Nehmen Sie an, dass alle Einträge in V sowie in W paarweise verschieden sind, dass $V[0]$ der kleinste Wert in V ist, und dass die Eckpunkte nach dem Uhrzeigersinn nummeriert sind.

- Geben Sie einen Algorithmus mit Laufzeit $O(\log n)$ an, der als Eingabe die Beschreibung eines konvexen Polygons wie oben bekommt, und den Eckpunkt mit der größten x -Koordinate ausgibt.
- Geben Sie einen Algorithmus mit Laufzeit $O(\log n)$ an, der als Eingabe die Beschreibung eines konvexen Polygons wie oben bekommt, und den Eckpunkt mit der größten y -Koordinate ausgibt.

H3 (Maximale Elemente)

(1+2+2 Punkte)

Betrachten Sie die Menge \mathbb{Z}^2 der Punkte mit ganzzahligen Koordinaten in der reellen Ebene. Man definiere eine Relation \leq auf \mathbb{Z}^2 wie folgt: Für $P, Q \in \mathbb{Z}^2$ mit $P = (x_1, y_1)$ und $Q = (x_2, y_2)$ ist

$$P \leq Q \stackrel{\text{def}}{\iff} (x_1 \leq x_2) \wedge (y_1 \leq y_2).$$

- Handelt es sich bei \leq um eine Ordnungsrelation auf \mathbb{Z}^2 ? Wenn ja, ist \leq eine totale Ordnung?

Ist $M \subseteq \mathbb{Z}^2$ eine Teilmenge von \mathbb{Z}^2 und $P \in M$ ein Element von M , dann ist P ein *maximales* Element von M bezüglich \leq wenn folgendes gilt: Für alle $Q \in M$, wenn $P \leq Q$, dann ist bereits $P = Q$. Mit anderen Worten, es gibt kein Element in M , welches größer als P ist bezüglich \leq .

-
- (b) Geben Sie einen Algorithmus an, der als Eingabe eine endliche Teilmenge $M \subseteq \mathbb{Z}^2$ bekommt, und die Menge der maximalen Punkte im M ausgibt.¹
- (c) Analysieren Sie die Korrektheit und die Laufzeit Ihres Algorithmus.

¹ Hinweis: Versuchen Sie zunächst ein Verständnis dafür zu entwickeln, was es bedeutet, dass ein Punkt $P \in M$ maximal in M bezüglich \leq ist. Um den Algorithmus zu definieren, sortieren Sie zuerst die Punktemenge M auf eine geeignete Art und Weise. Wenden Sie dann das Teile-und-Beherrsche-Paradigma an, indem Sie die Menge zweiteilen und Ihren Algorithmus rekursiv auf beide Teile anwenden. Achten Sie beim Zusammenführen der Ergebnisse darauf, dass es passieren kann, dass einige Punkte vom "linken" Teil plötzlich nicht mehr maximal sind. Diese müssen dann entsprechend ausgeschlossen werden.