Übungen zur Vorlesung

Algorithmen und Datenstrukturen

Übungsblatt 05

Cryptoplexity
Cryptography & Complexity Theory
Technische Universität Darmstadt
www.cryptoplexity.de

Arbeitsgruppe Kryptographie und Komplexitätstheorie
Prof. Dr. Marc Fischlin
Moritz Huppert
Tobias Schmalz

Sommersemester 2024
Veröffentlicht: 17.05.2024, 14:00 Uhr MESZ
Abgabe: 31.05.2024, 14:00 Uhr MESZ

G1 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) Verkettete Listen;
(b) Binäre Bäume;
(c) Inorder-, Preorder- und Postorder-Traversierungen;
(d) Binäre Suchbäume.

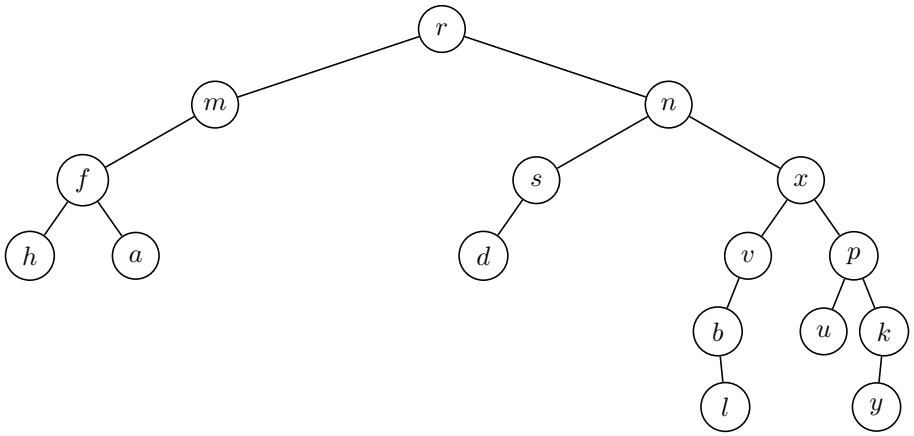
G2 Linked Lists

Eine *einfach verkettete Liste* ist eine Datenstruktur, die ähnlich definiert wird wie eine doppelt verkettete Liste, wo aber jedes Element nur den Zeiger auf das nachfolgende Element in der Liste besitzt (es fehlt also der Zeiger auf das vorhergehende Element).

(a) Vergleichen Sie einfach und doppelt verkettete Listen, indem Sie Vorteile der jeweiligen Datenstruktur beschreiben.
(b) Formulieren Sie einen nicht rektursiven Algorithmus, der eine einfach verkettete Liste mit n Elementen umkehrt. Die Methode soll keinen zusätzlichen Speicher wie extra Listen benutzen (es sind nur temporäre Hilfsvariablen, die unabhängig von der Größe n der Liste sind, erlaubt). Die Laufzeit des Algorithmus soll in $O(n)$ liegen. Beweisen Sie auch die Korrektheit Ihres Algorithmus.

G3 Binäre Bäume

Betrachten Sie den folgenden binären Baum:



Beantworten Sie zunächst die folgenden Fragen zu den Definitionen für binäre Bäume:

(a) Geben Sie die Kinder von n und die Elternknoten von f an.
(b) Welche sind die Geschwister von d?
(c) Welcher Knoten ist die Wurzel des Baumes?
(d) Wie viele innere Knoten und Blätter hat der Baum?
(e) Geben Sie die Vorfahren von v und die Nachkommen von x an.
(f) Was ist die Tiefe von b?
(g) Geben Sie den rechten Teilbaum von x an.
(h) Geben Sie für die Fachbegriffe aus den vorherigen Teilaufgaben die englischen Übersetzungen an.

Führen Sie nun nacheinander folgende Operationen auf dem Baum durch. Benutzen Sie dazu die Algorithmen aus der Vorlesung und zeichnen Sie jeweils den sich ergebenen Baum.

(i) Fügen Sie einen Knoten z ein.
(j) Löschen Sie Knoten v.
(k) Löschen Sie Knoten f.

G4 Binäre Suchbäume

(a) Fügen Sie nacheinander Knoten mit den Schlüsseln $17, 10, 1, 29, 7, 20, 30, 23$ in einen leeren binären Suchbaum ein. Skizzieren Sie den resultierenden Baum.
(b) Traversieren Sie den Baum jeweils nach dem Inorder-, Preorder-, und dem Postorder-Verfahren. Fällt Ihnen bei einem etwas auf?
(c) Entfernen Sie nun nacheinander die Knoten mit den Schlüsseln $1, 23, 29$. Geben Sie den entstehenden Baum, sowie für jeden Löschvorgang den zutreffenden Fall des Löschens an.
(d) Sei 7, 16, 10, 24, 21, 25, 19, 1 die Postorder-Traversierung eines binären Suchbaums. Rekonstruieren und skizzieren Sie diesen Baum.
(e) Beweisen oder widerlegen Sie folgende Aussage, indem Sie einen formalen Beweis oder ein minimales Gegenbeispiel angeben:

> „Alle binären Suchbäume mit eindeutigen Werten und ohne Halbblätter lassen sich allein aus ihrer Inorder-Traversierung eindeutig rekonstruieren."

G5 (Suchpfade in binären Suchbäumen)

Die folgenden Zahlenfolgen sollen Suchpfade bei der Suche nach einem Wert in einem binären Suchbaum darstellen. Bestimmen Sie ob ein binärer Suchbaum existiert, auf dem der angegebene Suchpfad entsteht. Falls nicht, begründen Sie warum dieser nicht existieren kann.

(a) 75, 85, 130, 82, 150, 140, 145
(b) 14, 24, 20, 23, 21, 22

(c) 345, 980, 756, 520, 437, 320, 390
(d) 345, 980, 756, 520, 437, 420, 390
(e) 345, 980, 756, 520, 687, 590, 623

G6* Aussagen über Bäume

(a) Zeigen Sie: Ein Baum mit n Knoten hat immer genau $n - 1$ Kanten. (*Hinweis:* Vollständige Induktion.)
(b) Man kann eine Menge von n Zahlen sortieren, indem man zuerst einen binären Suchbaum aus diesen Zahlen erstellt und diesen dann mittels einer Inorder Traversierung ausgibt. Was sind die Worst-Case und Best-Case Laufzeiten dieses Algorithmus? Begründen Sie Ihre Antwort.
(c) Zeigen Sie, dass die Binäre-Suchbaum-Eigenschaft unter Rechtsrotationen erhalten bleibt. Genauer: Ist B ein binärer Suchbaum und B' der Baum, den man durch eine beliebige Rechtsrotationen von B erhält, dann ist B' auch ein binärer Suchbaum. (Offensichtlich gilt dieselbe Eigenschaft auch für Linksrotationen.)

Hausübungen

In diesem Bereich finden Sie die theoretische Hausübung von Blatt 05. Bitte beachten Sie die allgemeinen Hinweise zu den Hausübungen und deren Abgabe im Moodle-Kurs.

Bitte reichen Sie Ihre Abgabe bis spätestens *Freitag, 31.05.2024, 14:00 Uhr MESZ* ein. Verspätete Abgaben können *nicht* berücksichtigt werden.

H1 Stacks and Queues II (2+2+3+4+4 Punkte)

In der Vorlesung haben Sie die Datenstrukturen Stacks und Queues kennengelernt. Ziel dieser Aufgabe ist es, ihr Verständnis für beide Datenstrukturen zu vertiefen.

(a) Sei Q eine Queue, die auf einem (anfangs leeren) Array der Größe 5 implementiert ist. Stellen Sie das Array nach jeder der folgenden Operationen dar: enqueue$(Q, 8)$; enqueue$(Q, 7)$; enqueue$(Q, 6)$; dequeue(Q); dequeue(Q); enqueue$(Q, 7)$; enqueue$(Q, 2)$; dequeue(Q).

(b) Sei S ein Stack, der auf einem Array der Größe 6 implementiert ist und zu Beginn die folgenden Buchstaben enthält: 'R', 'E', 'G', 'A', 'L', 'E'. Stellen Sie das Array nach jeder der folgenden Operationen dar: pop(S); pop(S); push$(S, \text{'X'})$; pop(S); pop(S); push$(S, \text{'E'})$; push$(S, \text{'N'})$.

(c) Betrachten Sie eine Datenstruktur DoubleStack, welche *zwei* Stacks enthält. Diese soll hier in *einem* Array der Größe $n \in \mathbb{N}$ implementiert werden. Keiner der Stapel soll überlaufen, solange die Gesamtanzahl der Elemente in beiden Stapeln höchstens n beträgt. Implementieren Sie hierfür die Methoden new, push1, push2, pop1, pop2. Beide Push- und Pop-Methoden sollen in konstanter Zeit, d.h. $O(1)$, ablaufen. Die Pop-Methoden sollen ein Fehlersymbol $\perp$ zurückgeben, wenn der zugehörige Stack leer ist. Begründen Sie ihren Entwurf.

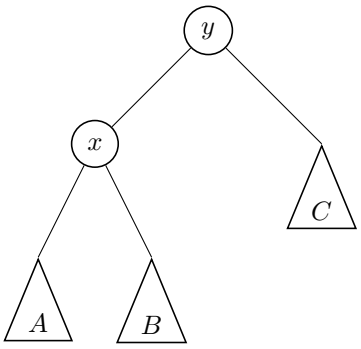
(d) Entwerfen Sie einen Algorithmus, der einen Stack invertiert, d.h. die Reihenfolge der Einträge umkehrt. Sie dürfen (neben dem Eingabestack) als zusätzlichen Speicher ausschließlich *Stacks* verwenden und keine anderen Datenstrukturen. Weiter soll der Algorithmus keinen Rückgabewert haben, sondern die Umkehrung im übergebenen Stack vollziehen. Führen Sie keine unnötigen Kopieroperationen durch. Begründen Sie ihren Entwurf.

(e) Entwerfen Sie einen Algorithmus, der einen Stack invertiert, d.h. die Reihenfolge der Einträge umkehrt. Sie dürfen (neben dem Eingabestack) als zusätzlichen Speicher ausschließlich *Queues* verwenden und keine anderen Datenstrukturen. Weiter soll der Algorithmus keinen Rückgabewert haben, sondern die Umkehrung im übergebenen Stack vollziehen. Führen Sie keine unnötigen Kopieroperationen durch. Begründen Sie ihren Entwurf.

H2 Binäre Suchbäume II (3+4+2+4+3 Punkte)

(a) Fügen Sie nacheinander folgende Knoten mit den Schlüsseln $48, 8, 69, 7, 50, 52, 71, 23, 1, 10$ in einen leeren binären Suchbaum ein. Dokumentieren Sie den Zustand nach jeder Einfügeoperation. Ist der entstandene Baum ein Rot-Schwarz-Baum? Wenn ja, färben Sie ihn ein, wenn nein, begründen Sie kurz warum.

(b) Seien a, b und c beliebige Knoten aus den jeweiligen Teilbäumen A, B und C in folgendem binären Suchbaum T:



Weiter seien d_a, d_b, d_c die Tiefen der drei Knoten. Bestimmen Sie jeweils die neuen Tiefen der drei Knoten nach den Operationen rotateRight(T, y) und rotateLeft(T, y).

(c) Seien x und y beliebige Knoten in einem binären Suchbaum T. Beweisen oder widerlegen Sie folgende Aussage:

Es gilt $\mathsf{delete}(\mathsf{delete}(T, x), y) = \mathsf{delete}(\mathsf{delete}(T, y), x)$, d.h. das Ergebnis zweier Löschoperationen ist unabhängig von deren Reihenfolge.

(d) Sei T ein binärer Suchbaum mit paarweise verschiedenen Werten und v ein beliebiger Knoten in T mit zwei Kindern. Zeigen Sie, dass der Vorgänger (nächstkleinerer Wert) und der Nachfolger (nächstgrößerer Wert) von v Halbblätter oder Blätter sein müssen.
Hinweis: Kann der Vorgänger ein rechtes Kind oder der Nachfolger ein linkes Kind haben?

(e) Argumentieren Sie, dass jeder vergleichsbasierte Algorithmus zum Erstellen eines binären Suchbaums aus einer beliebigen Liste von n Elementen im schlimmsten Fall $\Omega(n \log n)$ Zeit benötigt.