Technische Universität
Kaiserslautern

Fachbereich Informatik
AG Algorithmen und Komplexität

Ausgabedatum: 21.11.2013
Version: 2013-11-26 11:43

6. Übungsblatt für Track AI zur Vorlesung Entwurf und Analyse von Algorithmen, WS 13/14

Abgabe: *Bis* ***Freitag****, 29.11.2013, 12:00 Uhr, Kasten im Treppenhaus 48-6.*

Bitte denken Sie an die Anmeldung zur Zwischenklausur!

Der Fehler der Woche

Da Laufzeit in $\Theta(n^{\frac{3}{2}})$, muss Speicherbedarf in $\mathcal{O}(n)$ sein.

Was ist falsch?

Basisaufgaben

B1: Queue aus Stacks bauen

3 Punkte

Nehmen Sie an, Sie dürfen ausschließlich die Datenstruktur `Stack` verwenden, ohne daran Veränderungen vorzunehmen. Wie können Sie dann unter Verwendung zweier Stacks *effizient* eine Queue implementieren?

Effizienz heißt hier, dass jedes Element zwischen Eintritt in die Queue bis zum Verlassen der Queue nur konstant oft *gepusht* oder *gepopt* wird, und zwar unabhängig davon, wie viele Elemente dazwischen in die Queue eingefügt oder aus der Queue entfernt werden.

Formulieren Sie die wesentliche(n) Invarianten, die die Korrektheit ihres Entwurfs sicherstellen. Geben Sie weiterhin Pseudocode für die Initialisierung und Queueoperationen ihrer Datenstruktur an. Zeigen Sie weiterhin, dass Ihre Lösung tatsächlich effizient ist.

B2: Bäume in Links-Sohn-Darstellung

3 Punkte

Wir haben die Links-Sohn-Darstellung für Bäume behandelt, in der die Felder `KRaum` und `ZRaum` verwendet werden, um alle Knoten und die Struktur der Bäume darzustellen. Dabei ist es möglich, diese beiden Felder zu einem Feld mit drei Komponenten (Markierung, linkester Sohn, nächster Bruder) zu verschmelzen.

Wie müssen in dieser verschmolzenen Darstellung die Einträge *linkester Sohn* und *nächster Bruder* verwendet werden, um einen Baum zu repräsentieren?

Beschreiben Sie Ihre Lösung allgemein und illustrieren Sie sie anhand eines Baums, der mindestens sieben Knoten, eine Höhe von mindestens drei und mindestens einen Knoten mit mehr als zwei Kindern hat.

B3: Tiefensuche ohne Rekursion

3 Punkte

Tiefensuche ist – im Gegensatz zur Breitensuche – als rekursive Prozedur realisiert. Prozeduraufrufe erzeugen oft unverwünschten Laufzeit- und Speicheroverhead und sollten daher, sofern möglich und lohnend, vermieden werden.

Skizzieren Sie, wie Tiefensuche ohne rekursive Prozeduraufrufe implementiert werden kann. Diskutieren Sie, inwiefern eine Verbesserung der Laufzeit zu erwarten ist.

Hinweis: Sie *können* gerne Ihre Position mit Laufzeitmessungen untermauern. Stellen Sie in diesem Fall sicher, dass Ihre Ergebnisse belastbar sind.

B4: Breitensuche auf Adjazenzmatrizen

3 Punkte

Im Kontext von Satz 2.8 hatten wir gesehen, dass Breitensuche auf (Di)Graphen in Zeit $\mathcal{O}(|V| + |E_w|)$ möglich ist, wenn der (Di)Graph über Adjazenzlisten gegeben ist.

Implementieren Sie Breitensuche auf (Di)Graphen, die als Adjazenzmatrix gegeben sind! Begründen Sie die Korrektheit ihrer Implementierung und geben Sie eine möglichst scharfe obere Schranke für die Laufzeit an.

Aufbauaufgaben

15. Aufgabe

2 + 5 Punkte

Im Zuge der Diskussion des Traversierens von Bäumen

a) hatten wir festgestellt, dass weder In-, Pre- noch Postorder einen Baum eindeutig festlegt. Finden Sie für jeden der drei Fälle jeweils ein Beispiel, das diese Aussage belegt.

b) hatten wir bemerkt, dass die Pre- und die Postorder gemeinsam verwendet werden können, um den zugehörigen Baum eindeutig zu rekonstruieren. Entwerfen Sie einen Algorithmus, der genau dies leistet. Begründen Sie, warum Ihr Algorithmus die gestellte Aufgabe auch tatsächlich löst.

   Wenden Sie Ihr Verfahren zur Verdeutlichung beispielhaft auf die Pre- und Postorder eines beliebigen Baums mit

   - mindestens neun Knoten,
   - einer Höhe von mindestens vier und
   - mindestens einem Knoten, der mindestens drei innere Knoten als Kinder hat,

   an. Illustrieren Sie den Ablauf Ihres Algorithmus.

   Beachten Sie, dass nach Definition die Knotenlabel paarweise verschieden sind.

16. Aufgabe

5 + 2 Punkte

a) Beweisen Sie Satz 2.8:

> Sei $G = (V, E)$ ein gerichteter oder ungerichteter Graph, der als Adjazenzliste repräsentiert ist. Sei $w \in V$.
>
> i) `Breitensuche(w)` besucht jeden Knoten in V_w genau einmal und sonst keinen anderen Knoten.
>
> ii) `Breitensuche(w)` läuft in Zeit höchstens $\mathcal{O}(|V| + |E_w|)$.

b) Beweisen Sie Satz 2.9:

> Der von `Breitensuche(w)` erzeugte Aufrufbaum T_w ist ein Baum von kürzesten Wegen für G.

17. Aufgabe

5 Punkte

Sei $G = (V, E)$ ein zusammenhängender Graph. Die Kante $e \in E$ heißt *Brücke*, wenn das Entfernen von e dazu führte, dass G in zwei disjunkte Teilgraphen zerfällt.

Entwerfen Sie mit Hilfe der Tiefensuche einen Algorithmus, der die Brücken eines gegebenen Graphen $G = (V, E)$ ausgibt und in Zeit $o(|E|^2)$ läuft. Welche Zeitkomplexität hat Ihr Algorithmus?