Technische Universität
Kaiserslautern

Fachbereich Informatik
AG Algorithmen und Komplexität

Ausgabedatum: 22.11.2012
Version: 2012-11-21 16:49

7. Übungsblatt zur Vorlesung Entwurf und Analyse von Algorithmen, WS 12/13

Abgabe: *Bis Donnerstag, 29.11.2012, 12:00 Uhr, Abgabenkasten vor 48-694.*

38. Aufgabe

Track ε [2]
Track AI 2

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.

39. Aufgabe

Track ε 2
Track AI [2]

Bei der Baumdarstellung der Partitionen zur Lösung des UNION/FIND-Problems sind wir davon ausgegangen, dass wir die Größen eines jeden Baumes kennen, und zur Implementierung der UNION-Operation den kleineren Baum zum Sohn der Wurzel des größeren machen.

In dieser Aufgabe gehen wir davon aus, dass wir keine Größen kennen, sondern einen jeden Knoten x mit einem Gewicht $g(x)$ versehen. Wird eine Partition mit nur einem Element als Einknotenbaum initialisiert, so wird das Gewicht dieses Knotens auf 0 gesetzt. Die UNION-Operation macht nun die Wurzel mit einem höheren Gewicht zum Vater der Wurzel mit dem kleineren Gewicht; haben beide Wurzelknoten dasselbe Gewicht, so wählen wir zufällig einen der Wurzelknoten als Vater aus und erhöhen sein Gewicht um 1. FIND-Operationen belassen alle Gewichte unverändert.

Beweisen Sie, dass für $|T_x|$ die Anzahl der Knoten in einem derart konstruierten Baum mit Wurzel x stets $|T_x| \geq 2^{g(x)}$ gilt.

40. Aufgabe

Track ε	3
Track AI	3

Erweitern Sie die aus der Vorlesung bekannte Datenstruktur für Partitionen, die Bäume und Pfadkomprimierung benutzt, um eine Operation `SET(x)`. Für ein beliebiges Element `x` soll `SET(x)` die Partition P_x, in der `x` enthalten ist, als (neue) lineare Liste zurückgeben; die Reihenfolge der Elemente im Ergebnis spielt keine Rolle.

Die Laufzeit von `SET(x)` soll in $\mathcal{O}(|P_x|)$ liegen. Weiterhin sollen die asymptotischen Laufzeiten der anderen Operationen nicht beeinträchtigt werden.

Begründen Sie Korrektheit und Effizienz Ihres Entwurfs.

Hinweis: Es genügt, jeden Knoten der Datenstruktur um ein Attribut zu erweitern.

41. Aufgabe

Track ε	2
Track AI	2

Sei T ein erweiterterter Binärbaum mit n inneren Knoten. Beweisen Sie, dass stets $\mathsf{EPL}(T) = \mathsf{IPL}(T) + 2n$ gilt.

42. Aufgabe

	a)	**b)**	**c)**
Track ε	2	3	[4]
Track AI	2	3	[4]

a) Wie sieht eine geschlossene Darstellung der Erzeugendenfunktion für die über die (inhomogene) Rekursionsgleichung

$$\begin{aligned} n_1 &= 1, \\ n_2 &= 2, \\ n_i &= n_{i-1} + n_{i-2} + 1, \quad i \geq 3, \end{aligned}$$

definierte Zahlenfolge aus?

b) Lösen Sie die inhomogene lineare Rekursionsgleichung

$$\begin{aligned} a_0 &= 1, \\ a_i &= 3 \cdot a_{i-1} + 2^i \quad , \quad i \geq 1, \end{aligned}$$

mittels Erzeugendenfunktionen.

c) Lösen Sie die inhomogene lineare Rekursionsgleichung

$$\begin{aligned} b_0 &= 1, \\ b_1 &= 1, \\ b_{i+2} &= 3 \cdot b_{i+1} - 2 \cdot b_i + i, \quad i \geq 0, \end{aligned}$$

mittels Erzeugendenfunktionen.

43. Aufgabe

Track ε	[1]
Track AI	[1]

Beweisen oder widerlegen Sie: Es gibt erweiterte binäre Bäume mit einer geraden Anzahl von Knoten $n \geq 2$.

44. Aufgabe

Track ε	[2]
Track AI	[2]

Konstruieren Sie aus der Schlüsselfolge $91, 11, 49, 85, 56, 77, 32$ den zugehörigen AVL-Baum. Starten Sie dazu mit dem leeren Baum, in den Sie den Schlüssel 91 einfügen. Fügen Sie in den resultierenden AVL-Baum den Schlüssel 11 ein usw.

Stellen Sie jeden so entstehenden Baum einzeln graphisch dar, wobei an jedem Knoten sein Balancegrad notiert werden soll. Wird nach dem Einfügen eine Rotation notwendig, so geben Sie an, welcher Typ von Rotation an welchem Knoten angewendet werden muss; der aus der Rotation resultierende Baum soll dann als neue Graphik dargestellt werden.

Geben Sie auch den aus der Schlüsselfolge $91, 11, 49, 85, 56, 32, 77$ resultierenden Baum an. Was fällt beim Vergleich der beiden Bäume auf?

45. Aufgabe

	a)	**b)**	**c)**
Track ε	[2]	[2]	[1]
Track AI	[2]	[2]	[1]

a) Entwerfen Sie eine Datenstruktur, die Paare aus $\mathbb{N} \times \mathbb{N}$ so speichert, dass ein beliebiges Element in $\mathcal{O}(\log n)$ Zeit gefunden bzw. eingefügt werden kann, wenn n Elemente bereits enthalten sind.

   Begründen Sie Korrektheit und Effizienz Ihres Entwurfs.

b) Beschreiben Sie, wie man aus Ihrer Datenstruktur aus a) – evtl. mit kleinen Modifikationen, die die in Teil a) geforderten Eigenschaften nicht zerstören – in Zeit $\mathcal{O}(m + \log n)$ eine Liste aller m Elemente $(k, \_)$ in beliebiger Reihenfolge für festes k erzeugen kann.

   Begründen Sie Ihre Behauptungen.

c) Nehmen Sie an, dass auf einer festen Instanz eines Wörterbuchs eine (lange) Folge von Suchen nach den Schlüsseln $(k_0, l_0), (k_1, l_1), \ldots$ ausgeführt werden soll. Die Elemente sind nicht – wie üblicherweise angenommen – uniform gezogen, sondern verhalten sich lokal in der ersten Komponente, oder formal

$$\Pr[k_i = k_{i+1}] = 1 - \varepsilon$$

   für ein $\varepsilon \in (0, 1)$, wobei wir kleine ε, also das Szenario $\varepsilon \to 0$ untersuchen wollen. Sie dürfen außerdem annehmen, dass es sowohl „genügend viele“ Schlüssel also auch „genügend viele“ verschiedene erste bzw. zweite Komponenten gibt.

   Ermöglicht es Ihre Datenstruktur aus a) bzw. b) – evtl. wieder mit kleinen, den allgemeinen Fall nicht störenden Änderungen – aus diesem Wissen Kapital zu schlagen und die mittlere Suchzeit zu verbessern?

   Begründen Sie Ihre Behauptung.

	a)	b)
Track ε	[2]	[2]
Track AI	[2]	[2]

46. Aufgabe

Sei $T = (V, E)$ ein erweiterter Binärbaum mit Wurzel $r \in V$, und seien $u, v \in V$. Die Distanz $d(u, v)$ zwischen u und v ist gleich der Anzahl an Kanten auf dem kürzesten Weg von u nach v, wobei wir annehmen, dass wir die Kanten im Baum auch entgegen ihrer Richtung traversieren dürfen (Vater- und Nachfolgerzeiger). Mit dieser Bezeichnung gilt für die bereits bekannten Pfadlängen IPL und EPL

$$\mathsf{IPL}(T) = \sum_{v \in \mathcal{I}(T)} d(r, v) \text{ bzw.}$$
$$\mathsf{EPL}(T) = \sum_{v \in \mathcal{L}(T)} d(r, v) \; .$$

Zur Wiederholung: $\mathcal{I}(T)$ bezeichnet die Menge alle inneren Knoten, $\mathcal{L}(T)$ die Menge aller (fiktiven) Blätter von T.

Wir führen die *interne freie* und die *interne-externe freie* Pfadlängen IFPL und IEFPL wie folgt ein:

$$\mathsf{IFPL}(T) := \frac{1}{2} \cdot \sum_{u,v \in \mathcal{I}(T)} d(u, v) \text{ bzw.}$$
$$\mathsf{IEFPL}(T) := \sum_{u \in \mathcal{I}(T),\ v \in \mathcal{L}(T)} d(u, v) \; .$$

Wir modifizieren die Struktur eines binären Suchbaums, indem wir in jedem Knoten v außer dem Schlüssel noch den größten und den kleinsten Schlüssel speichern, der im gesamten (Teil)suchbaum mit Wurzel v vorkommt. Mit dieser Information kann man direkt das im Teilbaum mit Wurzel v vorkommende *Schlüsselintervall* ablesen. Die Operation `find(x,T)` realisieren wir nun wie folgt.

Wir beginnen unsere Suche bei einem beliebigen Knoten v von T. Dort überprüfen wir, ob x in dem durch den Teilbaum mit Wurzel v repräsentierten Intervall liegt. Falls ja, setzen wir an dieser Stelle die Suche nach dem üblichen Verfahren fort (d. h. wir entscheiden durch einen Schlüsselvergleich, ob wir in den linken oder rechten Teilbaum laufen müssen usw.). Falls nein, dann gehen wir zum Vaterknoten von v und starten die Suche nach dem modifizierten Verfahren erneut (d. h. wir testen, ob x im repräsentierten Intervall liegt usw.).

Sei T_e der zu T gehörende erweiterte Baum. Zeigen Sie, dass für oben beschriebenen Algorithmus (bei Gleichwahrscheinlichkeit der zu suchenden Schlüssel und der Startknoten) für die mittlere Laufzeit (wir betrachten Schlüsselvergleiche als Elementaroperationen) einer erfolgreichen Suche $\mathsf{CS}_n(T)$ bzw. einer erfolglosen Suche $\mathsf{CU}_n(T)$ in einem binären Suchbaum T mit n Knoten gilt:

a) $\mathbb{E}[\mathsf{CS}_n(T)] = \dfrac{2}{n^2} \cdot \mathsf{IFPL}(T_e) + 1$

b) $\mathbb{E}[\mathsf{CU}_n(T)] = \dfrac{1}{n(n+1)} \cdot \mathsf{IEFPL}(T_e)$