Technische Universität
Kaiserslautern

Fachbereich Informatik
AG Algorithmen und Komplexität

Ausgabedatum: 15.11.2012
Version: 2012-11-14 17:50

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

Abgabe: *Bis Donnerstag, 22.11.2012, 12:00 Uhr, Abgabenkasten vor 48-694.*

29. Aufgabe

	a)	b)
Track ε	1	4
Track AI	1	4

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 Baum mit mindestens neun Knoten, einer Höhe von mindestens vier und mindestens einem Knoten mit mehr als zwei Kindern an.

   Hinweis: Um einen Baum eindeutig aus seiner Pre- und Postorder zu rekonstruieren, muss man schrittweise die Wurzeln seiner Teilbäume identifizieren. Untersuchen Sie also, wie Sie anhand der Pre- und Postorder die Wurzel des Baumes bestimmen können, wie Sie anschließend die Wurzeln aller Teilbäume der Wurzel ablesen können usw. Aus den entsprechenden Beobachtungen ist es dann einfach möglich, ein passendes Verfahren abzuleiten.

30. Aufgabe

	a)	b)
Track ε	3	2
Track AI	3	[2]

a) Sei A die Adjazenzmatrix eines Digraphen G. Beweisen Sie, dass es in G genau dann einen Weg von v_i nach v_j der Länge k gibt, wenn $(A^k)_{i,j} \neq 0$, d. h. wenn der Eintrag an Position i, j in der k-ten Potenz der Adjazenzmatrix ungleich 0 ist.

b) Verwenden Sie die Erkenntnis aus a), um die Anzahl der Wege der Länge n von einem Knoten v zu einem Knoten v' im vollständigen Digraphen $G = (V, \mathcal{K}(V))$ mit $|V| = n$ zu berechnen.

Hinweis: Können Sie eine Eigenschaft von A^n zeigen, die sowohl für a) als auch für b) hilfreich ist?

31. Aufgabe

	a)	b)
Track ε	3	[3]
Track AI	[3]	[3]

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.

32. Aufgabe

Track ε	[2]
Track AI	2

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 $G = (V, E)$ erkennt und in Zeit $o(|E|^2)$ läuft. Welche Zeitkomplexität hat Ihr Algorithmus?

33. Aufgabe

Track ε	[2]
Track AI	2

Wir bezeichnen mit *Durchmesser* d eines Graphen $G = (V, E)$ den längsten kürzesten Pfad in G, also

$$d(G) := \max \{\text{dist}(u, v) \mid u, v \in V\}$$

mit $\text{dist}(u, v)$ der Länge eines kürzesten Pfades zwischen u und v in G.

Entwerfen Sie einen Algorithmus, der in Zeit $\mathcal{O}(|V|)$ den Durchmesser eines beliebigen Baumes berechnet.

34. Aufgabe

Track ε	[1]
Track AI	1

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

Skizzieren Sie, wie Tiefensuche ohne 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.

35. Aufgabe

Track ε [3]
Track AI [3]

Gegeben seien ein Graph mit $n \geq 2$ Knoten und ohne Kanten, sowie folgende Operation auf Graphen:

O: Füge dem Graphen (V, E) einen neuen Knoten v sowie zwei Kanten $\{v, w_1\}$ und $\{v, w_2\}$, $w_1, w_2 \in V$, $w_1 \neq w_2$ hinzu.

Wie oft hintereinander kann die Operation O ausgeführt werden, wenn kein Knoten einen Grad > 3 haben darf? Leiten Sie eine geschlossene Darstellung für die maximal mögliche Anzahl der Ausführungen in Abhängigkeit von n her.

36. Aufgabe

Track ε [1]
Track AI [1]

Es sei der ungerichtete Graph G mit den Knoten $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ und folgender Adjazenzliste gegeben:

$$
\begin{array}{ll}
0 \longmapsto & 2 \to 3 \to 10 \\
1 \longmapsto & 10 \to 7 \\
2 \longmapsto & 9 \to 8 \to 3 \to 0 \to 6 \\
3 \longmapsto & 2 \to 5 \to 0 \\
4 \longmapsto & 10 \\
5 \longmapsto & 3 \to 10
\end{array}
\qquad
\begin{array}{ll}
6 \longmapsto & 9 \to 2 \\
7 \longmapsto & 1 \to 10 \\
8 \longmapsto & 2 \to 9 \\
9 \longmapsto & 6 \to 2 \to 8 \\
10 \longmapsto & 1 \to 4 \to 0 \to 5 \to 7
\end{array}
$$

Berechnen Sie den Aufrufbaum von `Tiefensuche()` für G. An welcher Stelle der Tiefensuche können Sie zum ersten Mal folgern, dass G einfache, geschlossene Wege mit einer Länge ≥ 3 besitzt?

37. Aufgabe

Track ε [1]
Track AI [1]

Es sei der Digraph G mit der Knotenmenge $\{1, 2, 3, 4, 5, 6\}$ und folgender Adjazenzliste gegeben:

$$
\begin{array}{ll}
1 \longmapsto & 2 \to 6 \\
2 \longmapsto & 5 \\
3 \longmapsto & \epsilon
\end{array}
\qquad
\begin{array}{ll}
4 \longmapsto & 3 \to 6 \\
5 \longmapsto & 6 \\
6 \longmapsto & 2
\end{array}
$$

Bestimmen Sie für alle sieben Kanten des Graphen, ob es sich um eine Baum-, Vorwärts-, Rückwärts- oder Querkante handelt. Nehmen Sie dabei an, dass die Tiefensuche im Knoten 1 startet.