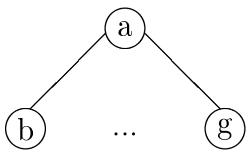
Prof. Andreas Goerdt
Frank Schädlich

TU Chemnitz
Sommersemester 2003

Theoretische Informatik III
3. Übung - Lösungsvorschläge

1. Aufgabe

(a) Der Baum sieht, nach Ausführung der `Find(`v`)`-Operationen in beliebiger Reihenfolge, wie folgt aus:



(b) Eine Folge von `Find(`v`)`-Operationen mit minimaler benötigter Zeit existiert im engeren Sinne nicht. Vielmehr hat jede solche Folge einen Zeitbedarf von 18 Einheiten, die sich wie folgt aufteilen:

- Für jeden Nichtwurzelknoten $v \neq a$ werden bei Ausführung von `Find(`v`)` mindestens zwei Zeiteinheiten (die Knoten a und v werden durchlaufen) benötigt. Zusätzlicher Aufwand entsteht nur, wenn sich auf dem Weg zwischen diesen beiden Knoten weitere befinden. Aufgrund der Wegkompression (wir identifizieren jeden solchen Knoten mit einer umzuhängenden Kante) kann dieser Fall aber höchstens fünfmal auftreten (jede Kante zwischen zwei Nichtwurzelknoten wird genau einmal auf die Wurzel umgebogen - und es gibt fünf solche Kanten). Zählen also einmalig insgesamt 5 Einheiten für Knoten auf dem Weg zwischen a und beliebigem v.
- Hinzu kommen die $6 \cdot 2 = 12$ Einheiten für `Find(`v`)`-Operationen mit $v \neq a$.
- Schließlich wird noch eine Einheit für `Find(`a`)` selbst benötigt.

2. Aufgabe

Sei u die Anzahl der `Union(`v`,`w`)`-Operationen, die vor allen f `Find(`v`)`-Operationen auf der Ausgangsstruktur beginnend ausgeführt werden, wobei $n = u + f$ ist. Eine solche `Union(`v`,`w`)`-Operation hat einen Zeitaufwand von $O(1)$, denn es wird lediglich eine neue Kante erzeugt. Somit haben u `Union(`v`,`w`)` einen Aufwand von $O(u)$.

Nach der Ausführung der u `Union(v,w)`-Operationen mit `Union-By-Size` liegt ein Baum mit maximaler Tiefe von $\log_2 u$ vor. Ohne diese Heuristik kann schlimmstenfalls ein Baum der Tiefe u entstehen. Die für diesen (den schlechteren) Fall, benötigte Zeit für alle `Find(v)` läßt sich aus Aufgabe 1 verallgemeinern:

$$\begin{aligned} \text{Zeit}(f\text{–mal } \texttt{Find(v)}) &\leq u - 1 \text{ (Wegkompression)} + 2 \cdot f \text{ (Knoten)} \\ &= 2 \cdot f + u - 1 \quad \in \quad O(u + f) \end{aligned}$$

Der Gesamtaufwand ist also mit $O(u) + O(u+f) = O(u+f) = O(n)$ linear in der Länge der Folge. Übrigens kann auch ein Potential für die Struktur definiert werden: $\Phi(\mathrm{S}) =$ Anzahl der Knoten, die einen Großvater haben. Dann hat man nach u-mal `Union(v,w)` ein Potential $\Phi = b \leq u - 2$ und somit den Aufwand $O(u+b) = O(u)$. In den `Find(`v`)` hängt die Wegkompression nur Knoten um, die einen Großvater haben. Dafür wird die vom Potential Φ bereitgestellte Zeit verbraucht und man erhält schließlich auch die obige Gleichung, also die Gesamtlaufzeit $O(n)$.

3. Aufgabe

Bei Anwendung von `Union-By-Size` gilt für jeden Baum T in der Struktur: Tiefe(T) $\leq \log_2 |T|$. Diese Tatsache muß auch noch nach dem Löschen eines Knotens Gültigkeit besitzen. Eine einfache Möglichkeit, das zu gewährleisten ist folgende: Man löscht einen Knoten zunächst nicht physisch, sondern markiert in nur als gelöscht.

4. Aufgabe

Der erste Baum kann nicht durch die Heuristik `Union-By-Size` entstanden sein: Sei r die Wurzel, s deren linker Sohn und t der rechte. Kanten werden nur durch `Union(v,w)` erzeugt. Mit `Union-By-Size` ist es zwar durchaus möglich, daß t unter r gehangen wurde. Dann kann aber vorher niemals s unter r gehangen worden sein, denn r ist nur ein einzelner Knoten, während s die Wurzel eines 3-knotigen Teilbaumes ist. Ein solches `Union(v,w)` widerspräche der Heuristik.

Für den zweiten Baum kann leicht eine Folge von Operationen gefunden werden, die ihn erzeugen.