Universität Karlsruhe (TH)
Institut für Theoretische Informatik

Prof. Dr. Peter Sanders, Dr. Rob van Stee
G. Veit Batz, Johannes Singler

3. Übungsblatt zur Algorithmentechnik WS 2007/08

http://algo2.iti.uni-karlsruhe.de/algotech.php
{sanders|vanstee|batz|singler}@ira.uka.de

Aufgabe 1 (*Priority-Queues, Schwierigkeit* $1 + 1$)
Abbildung 1 zeigt den inneren Zustand eines Fibonacci-Heap. Das angekreuzte Heap-Item sei dabei besonders markiert (wie es im Rahmen der Cascading-Cuts-Technik üblich ist).

a) Gegeben sei ein Fibonacci-Heap mit einem inneren Zustand wie in Abbildung 1. Wie in der Abbildung dargestellt seien außerdem drei Pointer x, y und z auf Heap-Items gegeben. Nun werde folgende Operationenfolge ausgeführt:

$$\mathit{decreaseKey}(x, 2);\ \mathit{decreaseKey}(y, 7);\ \mathit{decreaseKey}(z, 4);\ \mathit{deleteMin}()$$

Welche inneren Zustände durchläuft der Fibonacci-Heap? Zeichnen Sie abstrakt.

b) Sind folgende Aussagen richtig oder falsch? Begründen Sie jeweils kurz.

1. In einem Pairing-Heap ist jeder Baum von der Form B_i für ein jeweils geeignetes $i \in \mathbb{N}_0$.
2. Die Cascading-Cuts-Technik sorgt dafür, dass die Anzahl der Wurzelknoten in einem Fibonacci-Heap höchstens logarithmisch in n ist (n sei die Anzahl der Elemente im Heap).
3. Für Pairing-Heaps hat die Operation *decreaseKey* einen amortisierten Zeitaufwand von $\mathcal{O}(\log n)$, wobei n die Anzahl der Elemente im Heap sein soll.

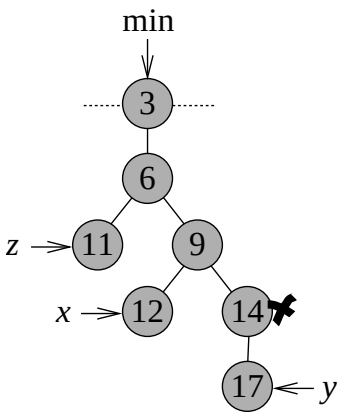

Abbildung 1: Innerer Zustand eines Fibonacci-Heaps.

Aufgabe 2 (*Sortierte Listen mit Rangoperation, Schwierigkeit* $1 + 2 + 2$)
Sortierte Listen werden oft mittels balancierter Binärbäume implementiert, die bei n Elementen eine maximale Höhe von $\mathcal{O}(\log n)$ garantieren. Damit kosten die Operationen `insert`, `remove`, `update` und `locate` mit Hinweis nur $\mathcal{O}(\log \Delta)$ Zeit (es sei Δ der Abstand vom Hinweis zur Fundstelle) und `rangeSearch` kostet $\mathcal{O}(\log \Delta + L)$, wobei L die Größe des Ergebnisses sein soll.

a) Wie ist die asymptotische Laufzeit, wenn man eine bereits sortierte Liste in einen leeren Rot-Schwarz-Baum geschickt einfügt?

So genannte Rank-Select-Dictionaries erlauben die effiziente Abfrage des Elements mit dem Rang k (`select`) und umgekehrt das Herausfinden des Rangs eines gesuchten Elements (`rank`). Dies kann erreicht werden, indem jeder Knoten des Rot-Schwarz-Baums speichert, wie viele Nachfahren er hat.

b) Skizzieren Sie die Algorithmen für `select` und `rank`, sowie die Neuerungen bei `insert`.

c) Was bedeutet das für die Laufzeiten der grundlegenden Operationen einer sortierten Liste, also `insert`, `remove`, `update`, `locate` und `rangeSearch`?

Aufgabe 3 (*Zusammenhang Matrizen und Graphen, Schwierigkeit* $1 + 3 + 1 + 2$)
Betrachten Sie das lineare Gleichungssystem (LGS) $Bx = c$. Die $n \times n$-Matrix B lässt sich als Adjazenzmatrix interpretieren. Der Graph $G = (V, E)$ ist dann festgelegt durch $V = \{1, \ldots, n\}$ und $E = \{(i, j) : B_{ij} \neq 0\}$, d. h. eine Kante zwischen den Knoten i und j existiert genau dann, wenn der zugehörige Eintrag in B ungleich 0 ist.
Das Vertauschen von Zeilen i und j in der Matrix bei gleichzeitigem Vertauschen der entsprechenden Zeilen in c ändert die Lösungsmenge nicht, ebensowenig das Vertauschen von Spalten i und j von B und den entsprechenden Zeilen von x.

a) G sei ein gerichteter azyklischer Graph (DAG). Welche für die Lösung des LGS günstige Form lässt sich nun durch vertauschen erreichen?

b) Die Anzahl der Kanten sei m. Geben Sie einen Algorithmus an, der die Matrix in Zeit $\mathcal{O}(n + m)$ auf die in a) gesuchte Form bringt. Nehmen Sie dabei an, dass die oben erwähnten Vertauschungsoperationen jeweils nur eine Zeiteinheit kosten. Zusätzlich sei eine Adjazenzlisten-Darstellung des Graphen verfügbar.

c) Welcher asymptotische Zeitaufwand ist danach noch nötig, um das LGS zu lösen? Zählen Sie jetzt jede Rechenoperation.

d) Wie kann die Voraussetzung aus b) erreicht werden, dass die Vertauschung von Zeilen und Spalten nur konstanten Zeitaufwand hat?

Aufgabe 4 (*Starke Zusammenhangskomponenten, Schwierigkeit* 2)
Gegeben sei ein gerichteter Graph $G = (V, E)$. Wie schon in der Vorlesung definieren wir

$$v \overset{*}{\longleftrightarrow} w \qquad :\Longleftrightarrow \qquad \text{es gibt Pfade } \langle v, \ldots, w\rangle \text{ und } \langle w, \ldots, v\rangle \text{ in } G$$

für $v, w \in V$. Zeigen Sie nun, dass $\overset{*}{\longleftrightarrow}$ eine Äquivalenzrelation ist.

Aufgabe 5 (*Tiefen- und Breitensuche, Schwierigkeit* $2 + 3$)

a) In der Vorlesung wurde ein Algorithmus zur Tiefensuche vorgestellt. Ändern Sie diesen so ab, dass für jede Kante die entsprechende Klassifizierung „forward“, „backward“, „cross“ oder „tree“ ausgegeben wird.

b) Zeigen Sie: Bei der Breitensuche (siehe Buch von Mehlhorn und Sanders, Abbildung 9.2) gibt es keine „forward“ Kanten.