8. Übungsblatt zu Grundzüge von Algorithmen und Datenstrukturen, WS 15/16

Prof. Markus Bläser
http://www-cc.cs.uni-saarland.de/course/50/

Abgabe: 17. Dezember 2015, 12:00 Uhr

Aufgabe 8.1 Geben Sie eine Methode Binomial_Heap_Decrease_Key(H, x, k) an, die in einem Binomial-Heap H den Schlüssel von Knoten x auf k vermindert. Falls $\text{key}(x) < k$, soll H nicht verändert werden. Die Laufzeit soll $O(\log n)$ sein, wobei n die Anzahl der Knoten in H ist.

Aufgabe 8.2 Ein 2-3-4-Heap ist ein Baum, dessen Knoten folgenden Bedingungen genügen:

- Jeder Knoten ist entweder ein Blatt oder aber ein interner Knoten, der zwei, drei oder vier Kinder hat. Ausnahme: die Wurzel darf auch nur ein Kind haben.
- Alle Blätter haben die gleiche Tiefe.
- Ist x ein Blatt, so enthält x einen Schlüssel, $\text{key}(x)$.
- Ist x ein interner Knoten, so enthält x einen Wert $\text{small}(x)$. Dies ist der kleinste Wert, der in den Blättern unterhalb von x gespeichert ist.

Beschreiben Sie, wie man 2-3-4-Heaps implementieren kann, so dass die folgenden Operationen in Zeit $O(\log n)$ durchgeführt werden können, wobei n die Anzahl aller Knoten in H bzw. in H_1 und H_2 zusammen ist.

- Insert(H, x, k) – fügt Knoten x als Blatt mit Wert k in 2-3-4-Heap H ein.
- Delete(H, x) – löscht Blatt x aus dem 2-3-4-Heap H.
- Extract_Min(H) – entnimmt das Blatt mit dem kleinsten Wert aus H.
- Union(H_1, H_2) – erzeugt einen neuen 2-3-4-Heap, der die Werte von H_1 und H_2 enthält. H_1 und H_2 werden dabei zerstört.

Hinweis: Beim Einfügen könnte man Knoten auf dem Pfad von der Wurzel zum Blatt, die schon vier Kinder haben, „aufspalten“.

Aufgabe 8.3 Wir realisieren einen Stack durch ein Array. Anfangs wird für den Stack ein Array der Größe 1 reserviert. Eine Push-Operation schreibt das Element in den nächsten freien Array-Platz. Ist das Array voll, wird ein doppelt so großes Array angelegt, das alte Array in das neue kopiert und das alte Array freigegeben.

a) Zeigen Sie, dass die amortisierten Kosten für die Push-Operation konstant sind. Dabei sind die Kosten, um eine Element in ein Array zu schreiben gleich 1 und die Kosten, um ein Array neu anzulegen, zu kopieren oder freizugeben gleich der Länge des Arrays.

b) Nun lassen wir zusätzlich Pop-Operationen zu. Dazu wird das letzte Element aus dem Array entfernt. Ist das Array zu weniger als die Hälfte voll, wird es durch ein Array der halben Größe ersetzt. Zeigen Sie, dass in diesem Fall die amortisierten Kosten für Push oder Pop linear sind.

c) Nun halbieren wir die Größe des Array erst, wenn es zu weniger als ein Viertel voll ist. Zeigen Sie, dass nun die Push- und Pop-Operation konstante amortisierte Kosten haben.