Max Wisniewski , Alexander Steen

Tutor: Lena Schlipf

Aufgabe 1 Quake Heaps: Details

Beschreiben Sie die Details von Quake Heaps.

Lösung:

Bei Quake Heaps werden die Einträge der einzelnen Turnierbäume in einer verketteten Strucktur von Knoten gespeichert, im gegensatz zu Binärheaps, bei denen man meißt auf Arrays zurück greift. Diese Knoten speichern zum einen ihren Wert, danach einen Verweis auf das Blattelement, dass ihren Wert trägt und zuletzt enthält das Blattelement noch einen Verweis auf das höchste Vorkommen dieses Wertes.

Wird nun ein insert ausgeführt, wird ein neuer Knoten eines Turnierbaumes angelegt, die beiden erwähnten Kanten zeigen auf diesen Knoten selbst, der Wert wird gesetzt und der ganze Knoten wird in das Hauptarray eingetragen, in T[0] und n[0] wird um 1 inkrementiert.

Decrease-Key bekommt einen Zeiger auf das Blattelement das gelöscht werden soll. Dieses kann von Insert herrausgegeben werden. Ein Delete kann dieses auch nicht ändern. Vom Blatt kann in den höchsten Knoten gegangen werden, der den Schlüssel noch enthält. Sind wir keine Wurzel, steht über uns noch ein kleinerer Wert. In diesem Fall führen wir ein cut durch und ändern im obersten Knoten den Wert. Da die anderen diesen alle über das Blatt referenzieren, können alle den Wert statt ihrem eigenen benutzen.
Wir haben weder die Knoten pro Ebene noch die Bäume pro Ebene verändert.

Decrease-key sucht sich als erstes das kleinste Element aus der obersten Liste herraus. Dieser Wert kann am Ende ausgegeben werden und wird nun ersteinmal entlang des Pfades nach unten gelöscht. Dabei entstehen neue Bäume, n muss auf dem ganzen Weg nach unten um eins vermindert werden pro Stufe. Haben wir die ganzen cut's hinter uns, führen wir als nächstes die Konsolidierung durch. Dabei mergen wir von kleinen Bäumen angefangen je 2 Bäume, wenn es mehr als einen Baum auf der Stufe gibt. Sind wir einmal durch, gibt es pro Ebene nur noch einen Baum. Immer wenn wir mergen, muss T angepasst werden. Indem von der Ebene aus der wir mergen die beiden Bäume entfernt werden und in T[i+1] wieder ein neuer eingefügt wird. Da wir einen neuen Knoten mit dem Minimum anlegen, müssen wir n[i+1] um eins inkrementieren. Das Element vom neuen Minimum muss den Zeiger von seinem Kind (mit gleichem Wert) auf das Blatt übernehmen und im Blatt den Zeiger auf den höchsten Vorgänger auf sich selber Ändern. Zuletzt kommt das Erdbeben.
Dabei gucken wir n von klein nach groß durch und gucken wann die Bedingung $n[i+1] \leq \frac{3}{2}n[i]$ Verletzt ist. Haben wir das i gefunden, gehen wir in T von i aus nach oben und löschen die Knoten nach oben hin.
Dies ist kein cut, aber wir können von der Wurzel die Pfade nach unten laufen bis wir die Ebene erreicht haben, auf dem Weg, zählen wir n auf der Ebene um 1 nach unten. Die Wurzel können wir aus T getrost löschen. Haben wir die richtigen Ebenen erreicht,

so fügen wir in T neue Knoten auf Ebene i hinzu, und Daten zuletzt noch die Zeiger in den Blättern der zugehörigen Wurzelknoten so ab, dass es nun auf die neue Wurzel verweist. Dies geht, da jeder Knoten entlang eines Pfades auf sein Blatt verweist.

Aufgabe 2 Quake Heaps: Analyse

Der Schritt *delete-min*, der dafür sorgt, dass es zu jeder Höhe höchstens einen Baum gibtm wird *Konsolidierung* genannt. In dieser Aufgabe sollen Sie sich über verschiede Varianten der Konsolidierung Gedanken machen.

(a) Nehmen Sie an, wir überspringen in *delete-min* den Konsolidierungsschritt. Wie ändert sich daduch die Laufzeitanalyse?

Lösung:

Die Analyse wird dadurch sehr viel leichter. Wenn wir nie den Konsolidierungsschritt durchführen, werden wir nie Turnierbäume zusammenfassen. Da wir aber immer nur Bäume der Höhe 0 einfügen, d.h. einzelne Knoten. Werden wir im Endeffekt nur eine Liste von einzelnen Werten haben.

In eine Liste einfügen kann man, wie vorher in $O(1)$.
Decrease-Key, hat nicht viel zu tun, da nichts umgebaut werden muss, daher ist es $O(1)$.
Delete-Min muss nun allerdings eine Liste von n Elementn nach dem Minimum durchsuchen. Daher sind die Kosten $O(n)$. Dies sollte sowohl amortisiert, erwartet und auch im Durchschnitt rauskommen, da wir keine Zusatzstruckturen haben, die Kosten verteilen könnten, oder uns das suchen erleichtern.

(b) Nehmen Sie an, dass wir am Ende von `delete-min` eine weitere Konsolidierung durchgeführen, falls ein Erdbeben stattgefunden hat. Zeigen Sie, dass sich dadurch die amoritsierte Laufzeit asymptotisch nicht ändert.

Lösung:

Das ganze sollte kein Problem darstellen. Die Laufzeit der zweiten Konsolidierung kann getragen werden, in dem die beim Buchhalter hinterlegten kosten verdoppelt werden, die hierfür aufgebraucht werden. Da es sich um den Faktor 2 handelt, ändert es asymptotisch nichts.

Danach müssen wir uns überlegen, dass die Konsolidierung nach dem Erdbeben unsere Eigenschaft $\forall i \ : \ n[i+1] \leq \frac{3}{2} n[i]$ nicht falsifizieren kann, die wir nach dem Erdbeben hergestellt haben.

Dabei kann man sich überlegen, dass was uns die Eigenschaft kaputt gemacht hat einzelne Pfade waren, die nur beim löschen entstehen können. Solange wir Vollständige Bäume konstruieren erzeugen wir auf 2 Knoten 1 Konten auf der Ebene darüber. Dies kann die Bedingung also nicht verletzen. Da wir beim Merge immer diese Strucktur erhalten, wird die Bedingung nicht verletzt. Damit sind die Vorraussetzungen die selben, wie beim in der VL gezeigten Algorithmus und die Analyse

ändert sich bis um den konstanten Faktor für die Konsolidierung nicht.

Damit bleibt die Laufzeit gleich.

Aufgabe 3 Potentialfunktionen

Sei $\Phi : \mathfrak{D} \to \mathbb{N}_0$ eine Potentialfunktion, die jedem möglichen Zustand $D \in \mathfrak{D}$ der Datenstrucktur eine natürliche Zahl zuweist. Dabei gilt, für den initialen Zustand $\Phi(D_0) = 0$.

Die *amortisierten Kosten* einer Operation X sind folgender Maßen definiert:
$\hat{c}_X = c_X + \Phi(D') - \Phi(D)$, wobei D' der Folgezustand und D der Ausgangszustand ist. c_X sind die tatsächlichen Kosten von X.

(a) Geben Sie eine Interpretation der Potentialfunktion und erklären Sie die Intuition hinter der obigen Definition der amortisierte Kosten.

Lösung:

Bei der Buchhaltermethode werden für jede Operation statische Kosten angelegt, die gesparrt werden. Bei der Potentialmethode können die Kosten für jede Ausführung anders vergeben werden.

Die *amortisierte Kosten* müssen dabei immer die eigentlichen Kosten tragen, und besteht aus der Differenz der beiden Potentiale. Die Potentiale stellen prinzipiell das Konto aus der Buchhaltermethode dar. Die Differenz der beiden, stellt dar, wie viel Geld wir bei der Operation auf das Konto einzahlen (wenn es positiv ist) oder wir abheben (wenn es negtiv ist).

(b) Sei $X_1, X_2, \cdots, X_n$ eine Folge von Operationen, die auf der initialen Datenstrucktur D_0 ausgeführt wird. Zeigen Sie, dass $\sum_i x_{X_i} \leq \sum_i \hat{c}_{X_i}$ ist. Interpretieren Sie das Ergebnis.

Lösung:

Gelten weiterhin die obrigen Definitionen.

$$\begin{aligned}
\sum_{i=1}^{n} \hat{c}_{X_i} &= \sum_{i=1}^{n} c_{X_i} + \Phi(X_i) - \Phi(X_{i-1}) \\
&= \sum_{i=1}^{n} c_{X_i} + \sum_{i=1}^{n} \Phi(X_i) - \sum_{i=1}^{n} \Phi(X_{i-1}) \\
&= \sum_{i=1}^{n} c_{X_i} + \Phi(X_n) + \sum_{i=1}^{n-1} \Phi(X_i) - \Phi(X_0) - \sum_{i=1}^{n-1} \Phi(X_i) \\
&= \sum_{i=1}^{n} c_{X_i} + \Phi(X_n) - \Phi(X_0) \\
&\geq \sum_{i=1}^{n} c_{X_i}
\end{aligned}$$

Dies gilt, da $\Phi(X_0) = 0$ ist und $Im(\Phi) = \mathbb{N}_0$, also kann das wegfallen dieser beiden Terme, den gesammten Term nur größer machen.

Das gute ist, dass wenn wir nun eine Funktion $A : K \to \mathbb{R}$ finden, wobei K unsere Eingabegröße war, und es gilt :
$\forall 1 \leq i \leq n : \hat{c}_{X_i} \leq A(m)$, dann können wir die Kosten der Folge abschätzen durch:
$\sum_{i=1}^{n} c_{X_i} \leq n \cdot A(X)$.

Die amortisierten Kosten pro Operation X habe ist also $A(X)$.

(c) Bestimmen Sie die amortisierten Kosten des Binärzählwerks mit der Potentialmethode erneut.

Lösung:

Wir müssen zunächst Φ definieren. Wenn man sich die Buchhaltermehtode für die Aufgabe einmal überlegt hat, dann erkennt man, dass auf dem Konto immer so viel Geld sein muss, wie die Zahl einsen hat. Sei $x_1x_2...x_n$ der Zustand unseres Binärzählwerks, wobei $x_i \in \{0,1\}$ $\Phi(x_1x_2x_3..x_n) = x_1 + x_2... + x_n$ ist.

Nun können wir die Kosten für einen Überführung definieren. Die Überführung tut zunächst folgendes:
$x_1x_2...x_n \to x'_1x'_2...x'_n$.
$c(x_1x_2...x_n) = |x'_1 - x_1| + |x'_2 - x_2| + ... + |x'_n - x_n|$
Die Anzahl der Bits die wir geändert haben.

Nun können wir die amortisierten Kosten aufschreiben:

$$\begin{aligned}
\hat{c}(x_1x_2...x_n) &= c(x_1x_2...x_3) + \Phi(x'_1x'_2...x'_n) - \Phi(x_1x_2...x_n) \\
&= \sum_{i=1}^{n} |x'_i - x_i| + \sum_{i=1}^{n} x'_i - \sum_{i=1}^{n} x_i \\
&= \sum_{i=1}^{n} (|x'_i - x_i| + x'_i - x_i)
\end{aligned}$$

Für die Bedingungen $(|x'_i - x_i| + x'_i - x_i)$ haben wir nur 4 verschiedene Möglichkeiten, wie sie ausgewertet werden kann. Wenn die Werte sich nicht geändert haben $(x_i = x'_i = 0 \vee 1)$, dann haben wir keine Kosten. Wenn wir $x'_i = 0$ und $x_i = 1$ haben, dann ergibt die Formel 0. Der letzte Fall, wenn wir das Bit von 0 auf 1 ändern ergibt eine 2.

Die amortisierten Kosten sind nun also $\hat{c}(x_1x_2...x_n) = \#$ Änderung $0 \to 1$.
Nun überlegt man sich kurz, dass pro Inkrement, nur maximal eine 1 hinzukommen kann, dann ergbit sich daraus, dass
$\hat{c}(x_1x_2...x_n) \leq 2$ gilt.

Nach *b)* können wir nun aussagen, dass
$\sum \hat{c}_{X_i} \leq 2n$ gilt.

Damit sind die amortisierten Kosten, wie schon auf dem 8. Übungszettel gezeigt $\hat{c}_X = 2 = O(1)$.