Grundzüge von Algorithmen und Datenstrukturen, WS 15/16: Lösungshinweise zum 8. Übungsblatt

Markus Bläser

Aufgabe 8.1 Cormen et. al., Kapitel 19, Prozedur Binomial-Heap-Decrease-Key, Seite 470.

Aufgabe 8.2 *Logarithmische Tiefe.* Zunächst begründen wir, dass ein 2-3-4-Heap („Heap“ im folgenden) logarithmische Tiefe in der Anzahl der darin gespeicherten Schlüssel bzw. ebenso in der Anzahl der darin enthaltenen Knoten hat. Daraus folgt dann, dass alle Operationen logarithmische Laufzeit haben, da sie nur ein oder zwei Mal den Heap von oben nach unten bzw. umgekehrt durchlaufen.

Wir zeigen durch Induktion über t: ein Heap der Tiefe t, $t \geq 2$, hat mindestens 2^t Blätter. Für Heaps der Tiefe $t = 2$ begründet man dies, indem man alle solche Heaps einzeln betrachtet. Für einen Heap H der Tiefe $t \geq 3$ gilt: H besteht aus einer Wurzel r und mindestens zwei Unterheaps der Tiefe $t-1$, die nach Induktionsvoraussetzung jeweils $\geq 2^{t-1}$ Blätter haben. Also hat H mindestens $2 \cdot 2^{t-1} = 2^t$ Blätter.

Analog kann man zeigen: ein Heap der Tiefe t hat mindestens $2^{t+1} - 1$ Knoten. Bzw.: da jedes Blatt ein Knoten ist, folgt ohnehin, dass jeder Heap logarithmische Tiefe in der Anzahl der Knoten hat.

small Wir müssen in allen folgenden Algorithmen auch stets den Wert small der einzelnen Knoten gegebenenfalls aktualisieren, etwa, wenn wir Teilbäume umhängen. Dies kann jeweils erfolgen, indem der small-Wert auf das Minimum der small-Werte der Kinder bzw., falls der betrachtete Knoten Blätter als Kinder hat, auf das Minimum der Schlüssel der Kinder gesetzt wird.

In manchen Fällen wird im folgenden das Aktualisieren des small-Wertes erwähnt und beschrieben, in anderen nicht, dann erfolgt es entsprechend dem gerade beschriebenen Verfahren.

Einfügen. Insert(H, x, k) arbeitet so: In jedem Fall sorgen wir dafür, dass x keine Kinder hat und $\text{key}(x) = k$ ist.

Wenn H eine leerer Heap ist, wird x die Wurzel. Falls H aus einer Wurzel r ohne Kinder besteht, geschieht folgendes: Der Heap bekommt einen neue Wurzel r' mit r und x als Kindern. Wir setzen $\text{small}(r') := \min(\text{key}(r), k)$.

Falls die Wurzel r 4 Kinder a, b, c, d hat, so formen wir den Heap so um, dass r zwei Kinder r_1 und r_2 hat und r_1 die Kinder a und b, sowie r_2 die Kinder c und d.

Nun sei z der Knoten im Heap, an dem wir uns gerade befinden, zu Beginn die Wurzel. Wir können nun davon ausgehen, dass z entweder 2 oder 3 Kinder hat.

Falls diese Kinder Blätter sind, fügen wir x als neues Kind von z ein und sind fertig.

Ansonsten setzen wir so fort: Falls es ein Kind w von z gibt, das weniger als 4 Kinder hat setzen wir z auf w und setzen mit diesem z fort wie im letzen Absatz beschrieben. Falls alle Kinder von z 4 Kinder haben, erzeugen wir ein neues Kind von z und verteilen die Enkelbäume von z gleichmäßig auf die nun um eins vergrößerte Menge der Kinder von z. Wir passen small(z) und die small-Felder der Kinder von z an, setzen sie jeweils auf das Minimum der small-Werte ihrer neuen Kindknoten. Nun gibt es ein Kind w von z, das weniger als 4 Kinder hat, wir setzen z auf w und setzen fort wie im vorherigen Absatz beschrieben.

Löschen. Delete(H, x) implementieren wir so: Falls x die Wurzel ist, löschen wir den Heap. Falls x eines von nur zwei Kindern der Wurzel des Heaps ist, wird das andere Kind die Wurzel, die alte Wurzel lassen wir verschwinden, ebenso x.

In allen anderen Fällen gehen wir wie folgt vor: Zunächste wandern wir vom Blatt x zur Wurzel und markieren dabei den Weg, so dass wir im Folgenden, wenn wir von der Wurzel wieder zu x gehen, stets wissen, in welchen Unterbaum wir gehen müssen, um zu x zu gelangen. Dies kann realisiert werden, indem wir für jeden Knoten v des Heaps ein zusätzliches Feld d(v) vorsehen, in das wir nun die Nummer des Kindes eintragen, in dessen Unterbaum sich x befindet.

Nachdem wir den Weg von der Wurzel zu x markiert haben, laufen wir von der Wurzel zu x, wobei z jeweils der Knoten sein soll, bei dem wir uns befinden. Wir werden dafür sorgen, dass z stets mindestens 3 Kinder hat. Wir beginnen mit z bei der Wurzel. Dazu führen wir zunächst folgende Sonderbehandlung der Wurzel durch.

Falls die Wurzel r nur zwei Kinder r_1 und r_2 hat, die wiederum jeweils nur zwei Kinder, a, b und c, d haben, entfernen wir r_1 und r_2 hängen a bis d direkt an r. Falls die Wurzel zwei Kinder hat, von denen eins mindestens drei Kinder hat, gehen wir wie folgt vor: Falls x sich in $T(u)$ befindet, wobei u ein Kind von r ist und mindestens drei Kinder hat, setzen wir z auf u und setzen mit dem nächsten Absatz fort. Ansonsten hängen sei u das Kind von r, das mindestens drei Kinder hat und u' das Kind von r, so dass sich x in $T(u')$ befindet und das folglich nur zwei Kinder hat. Wir hängen nun ein Kind von u an u' und setzen mit $z := u'$ fort.

Nehmen wir nun an, dass wir bei z sind, einem Knoten mit mindestens 3 Kindern und irgendwo in $T(z)$ sich das zu löschende Blatt x befindet. Falls ein Kind von z ist, löschen wir x und sind fertig.

Sei anderenfalls w das Kind von z, so dass sich x in $T(w)$ befindet. Falls w mindestens drei Kinder hat, setzen wir z auf w und setzen fort wie im vorherigen Absatz beschrieben. Falls w nur zwei Kinder hat, löschen wir ein Kind von z und verteilen die bisherigen Enkelbäume von z so auf die um eins verminderte Menge der Kinder von z, dass sich x nun im Unterbaum eines Kindes w' von z befindet, wobei w' mindestens drei Kinder hat. Dann setzen wir z auf w' und setzen fort mit dem vo-

herigen Absatz. Hat w hat nur 2 Kinder, allerdings die beiden Geschwisterkinder 4 Kinder, dann kann man beim Löschen die Enkel nicht verteilen, weil der Platz dafür nicht ausreicht. Dann hängt man vorher ein Kind der Geschwister an w (so dass es mindestens 3 Kinder hat).

Extract Minimum. Extract_Min(H) kann durchgeführt werden, indem in jedem Knoten x in den Unterbaum weitergegangen wird, der das kleinste Blatt hat, was anhand von small(y_i), $y_1, y_2(, y_3(, y_4))$ seien die Kinder von x, ermittelt wird. Sobald man an einem Blatt z angelangt ist, hat man den kleinsten Wert w gefunden, merkt sich diesen, ruft Delete(H, z) auf und gibt w zurück.

Union. Union(H_1, H_2) für Heaps H_1, H_2, die gleiche Tiefe haben, wird durchgeführt, indem eine neue Wurzel erzeugt wird, an die H_1 und H_2 als Teilbäume gehängt werden.

Falls einer der Heaps leer ist, sagen wir H_1, tun wir nichts und H_2 ist das Ergebnis.

Falls H_1 und H_2 unterschiedliche Tiefe haben, sei H_1 der tiefere Heap. Wir gehen nun so vor, also ob wir in H_1 ein Element mit Schlüssel small(r_2) einfügen wollen, wobei r_2 die Wurzel von H_2 sei. Insbesonder spalten wir also, falls nötig, Knoten auf. Wir steigen aber nicht notwendig bis zu den Blättern ab, sondern genau so weit, dass die nun noch verbleibende Tiefe bis zu den Blättern gleich der Tiefe von H_2 ist. Nachdem wir sichergestellt haben, dass der Knoten z, an dem wir uns nun befinden, höchstens 3 Kinder hat, hängen wir r_2 an z.

Aufgabe 8.3 Gab es angeblich letztes Jahr bei Prof. Seidel.