5. Aufgabenblatt zur Vorlesung

Algorithmen und Programmierung III

Wolfgang Mulzer WiSe 2013/14

Abgabe am 21. November 2013 vor der Vorlesung in die jeweiligen Tutorenfächer

Aufgabe 1 Binäre Heaps *10 Punkte*

(a) Wie viele Elemente enthält ein binärer Heap der Höhe h mindestens? Wie viele sind es höchstens? Beweisen Sie, dass ein binärer Heap mit n Elementen Höhe $\Theta(\log n)$ hat. (*Achtung:* Wir zählen die Höhe und die Ebenen ab 0.)

(b) Wo kann in einem binären Heap das zweitkleinste Element stehen? Wie ist es mit dem drittkleinsten Element? Geben Sie allgemein an, auf welchen Ebenen eines binären Heaps mit n Elementen sich das k. kleinste Element befinden kann (Sie dürfen annehmen, dass alle Ebenen voll besetzt sind).

(c) Veranschaulichen Sie die bottom-up Heapkonstruktion mit dem Array `[7, 6, 5, 3, 1, 4, 2, 8]`.

(freiwillig, 5 Zusatzpunkte) Angenommen, wir fügen n Elemente nacheinander in einen anfangs leeren Heap ein. Zeigen Sie, dass dies im worst-case $\Omega(n \log n)$ Zeit benötigt.

Achtung: Anfangs braucht das Einfügen wesentlich weniger als $\log n$ Schritte.

Aufgabe 2 Implementierung weiterer Heap-Operationen *10 Punkte*

(a) Programmieren Sie einen binären Heap mit n Elementen, gespeichert als Array, in Java. Implementieren Sie außerdem die folgenden Operationen mit jeweils einer Laufzeit von $O(\log n)$:

- `decreaseKey`(i, k): Setze das Element an Position i auf den Wert k und stelle die Heapeigenschaft wieder her (Vorbedingung: k ist kleiner gleich dem alten Element an Position i).
- `delete`(i): Lösche das Element an der Position i und stelle die Heapeigenschaft wieder her.

Vermeiden Sie Redundanzen, indem Sie bereits vorhandene Funktionen nutzen. Begründen Sie kurz Korrektheit und Laufzeit.

(b) Wir wollen das Interface `PriorityQueue` aus der Vorlesung um die Operationen aus (a) erweitern. Erklären Sie kurz, warum wir zur effizienten Implementierung einen "Zeiger" in die Datenstruktur benötigen (d.h., warum ist es keine gute Idee, an `decreaseKey` nur das alte und das neue Element zu übergeben, selbst wenn jedes Element nur einmal in der Datenstruktur vorkommt?).

Wie können wir solche Zeiger implementieren, so dass das Geheimnisprinzip gewahrt bleibt? Führen Sie eine Interface `Position` ein, das einen Zeiger in

die Datenstruktur repräsentiert. Erläutern Sie knapp, wie man das Interface `PriorityQueue` modifizieren kann, um `Position` zu benutzen. Welche Schwierigkeiten ergeben sich dabei? Skizzieren Sie kurz einen Lösungsansatz. (Bei dieser Teilaufgabe müssen Sie nichts programmieren.)

Aufgabe 3 Elementare Wahrscheinlichkeitsrechnung *10 Punkte*

Auf einem Tisch stehen N Kisten. In diese Kisten werden nacheinander unabhängig voneinander n Bälle geworfen, wobei jede Kiste mit gleicher Wahrscheinlichkeit getroffen wird.

(a) Berechnen Sie die Wahrscheinlichkeit, dass Kiste i leer ist. Sei Y_i die Zufallsvariable, die den Wert 1 annimmt, falls Kiste i leer ist, und 0 sonst. Geben Sie auch den Erwartungswert $E[Y_i]$ an.

(b) Sei X die Zufallsvariable, welche die Anzahl von leeren Kisten angibt. Berechnen Sie den Erwartungswert von X mit Hilfe der Erwartungswerte $E[Y_i]$.

(c) Geben Sie eine möglichst gute Schranke $f(N)$ an, so dass gilt: wenn $n \geq f(N)$, dann ist die Wahrscheinlichkeit, dass eine Kiste mindestens zwei Bälle enthält, größer als $1/2$.

Hinweis: Stellen Sie sich vor, die n Bälle werden nacheinander in die Kisten geworfen. Was ist die Wahrscheinlichkeit, dass der nächste Ball in einer leeren Kiste landet, wenn alle vorherigen Bälle in einer leeren Kiste gelandet sind? Verwenden Sie die ungemein nützliche Abschätzung $1 + x \leq e^x$, welche für alle $x \in \mathbb{R}$ gilt.

(d) Professor Pinocchio hat eine Idee, um Hashtabellen zu vereinfachen. Wenn wir die Zahl N der Plätze in der Hashtabelle im Verhältnis zur Anzahl der zu speichernden Einträge n groß genug wählen, sollte die Wahrscheinlichkeit, dass Kollisionen auftreten, verschwindend gering werden (unter der Annahme, dass sich die Hashfunktion wie eine zufällige Funktion verhält). Dann könnte man auf die Kollisionsbehandlung verzichten. In Anbetracht von (c), was halten Sie von dem Vorschlag? (Wenn Sie Teil (c) nicht gelöst haben, dann bearbeiten Sie diesen Teil unter der Annahme, dass $f(N) = N^{1/3}$ ist. Achtung: Das ist nicht die Lösung für (c).)