5. Aufgabenblatt zur Vorlesung

Algorithmen, Datenstrukturen und Datenabstraktion WS 17/18

Wolfgang Mulzer, Katharina Klost

Abgabe am 24. November 2017 bis 10 Uhr in die jeweiligen Tutorenfächer

Aufgabe 1 Spezifizieren und Implementieren eines ADT *10 Punkte*

Spezifizieren Sie sinnvoll den abstrakten Datentyp "double ended queue", der das Einfügen und Löschen von Objekten an beiden Enden unterstützt, und zwar auf zwei verschiedene Arten: einmal verbal (versuchen Sie, möglichst genau zu sein) und einmal anhand eines mathematischen Modells.

Formulieren Sie ein entsprechendes Interface in Java, und implementieren Sie den Datentyp auf zwei verschiedene Arten. Vergleichen Sie die Vor- und Nachteile Ihrer beiden Implementierungen (bezüglich Laufzeit, Implementierungsaufwand, Flexibilität, usw.).

Freiwillig, 5 Zusatzpunkte: Verfahren Sie wie in Aufgabe 3.3(c) und erstellen Sie mindestens fünf JUnit-Testcases für Ihre Implementierung.

Aufgabe 2 Untere Schranken für Prioritätswarteschlangen *5 Punkte*

(a) Zeigen Sie, wie man eine Prioritätswarteschlange verwenden kann, um eine Folge von n vergleichbaren Elementen zu sortieren.

(b) Aus *Konzepte der imperativen und objektorientierten Programmierung* wissen Sie, dass jeder vergleichsbasierte Sortieralgorithmus mindestens $\Omega(n \log n)$ Operationen benötigt. Folgern Sie, dass in jeder vergleichsbasierte Implementierung der Prioritätswarteschlange mindestens eine Operation $\Omega(\log n)$ amortisierte Laufzeit haben muss.

Aufgabe 3 Binäre Heaps *15 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]`.

(d) 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.