Tutorium
Grundlagen: Algorithmen und Datenstrukturen

Übungsblatt Woche 5

Aufgabe 5.1 Laufzeitanalyse: Mergesort

Aufgabe 5.1

In dieser Aufgabe machen wir eine Laufzeitanalyse von Mergesort auf drei verschiedene Weisen.

a) Zeigen Sie argumentativ, wie sich die Laufzeit von Mergesort verhält, indem Sie folgende Fragen beantworten: Wie viele Rekursionsebenen gibt es im Allgemeinen bei Mergesort (wobei wir den initialen Aufruf von Mergesort nicht als eigene Rekursionsebene zählen)? In welcher Größenordnung liegt asymptotisch der Aufwand für jede Rekursionsebene? Was ist damit der asymptotische Aufwand für den gesamten Algorithmus?

In der Vorlesung wurde die Laufzeit rekursiv formuliert und das Mastertheorem verwendet, um zu zeigen, dass die Laufzeit von Mergesort in O(n log n) liegt.
Die rekursive Formulierung der Laufzeit lautet

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + \Theta(n)$$

$$T(1) = \Theta(1)$$

Zeigen Sie ohne Verwendung des Master-Theorems, dass die Worst-Case-Laufzeit von Mergesort angewandt auf Zahlenfolgen, deren Längen Zweierpotenzen sind, in $\mathcal{O}(n\ log\ n)$liegt.

b) Verwenden Sie dazu die Methode des iterativen Einsetzens.

c) Verwenden Sie dazu vollständige Induktion (Abschätzung durch z.B. $\mathrm{T(n)} \leq dn \log(dn)$).

Aufgabe 5.1 (a)

Aufgabe 5.1 (b)

Aufgabe 5.1 (c)

Aufgabe 5.2
Betrunkener Übungsleiter

Aufgabe 5.2

Wir betrachten einen torkelnden Übungsleiter an einer Kletterwand, der die folgenden beiden Operationen durchfuhren kann:

- **hoch**
- **runter(int k)**

Die Starthöhe des Übungsleiters beträgt 0 Meter.
Durch die Operation **hoch** steigt der Übungsleiter von seiner aktuellen Position aus genau einen Meter höher. Diese Operation hat die Laufzeit 1.
Durch die Operation **runter(int k)** fällt der Übungsleiter von seiner aktuellen Position aus exakt $min\{h, k\}$ Meter nach unten, wobei h die aktuelle Höhe des Übungsleiters ist. (Das bedeutet, dass der Übungsleiter nie tiefer als seine Starthöhe sinken kann). Wir nehmen hierbei an, dass k stets eine natürliche Zahl (nicht-negativ) ist. Die Operation **runter(int k)** hat die Laufzeit $min\{h, k\}$.

Zeigen Sie mithilfe der Bankkonto-Methode, dass die amortisierten Laufzeiten der Operationen **hoch** und **runter(int k)** in $\mathcal{O}(1)$ liegen.

Aufgabe 5.2

Aufgabe 5.3
Stapelschlange

Aufgabe 5.3

In dieser Aufgabe geht es darum, einen Algorithmus, der eine Queue für Integer-Zahlen mittels zweier Stacks implementiert, hinsichtlich seiner Laufzeit zu untersuchen.

a) Geben Sie die Laufzeitklasse der Worst-Case-Laufzeit eines Aufrufs der ***dequeue()***-Methode in Abhängigkeit der aktuellen Größe der Queue n in Landau-Notation an.

b) Überlegen Sie sich ein Amortisierungsschema, welches die amortisierte Laufzeit der einzelnen Operationen minimiert. Nennen Sie die amortisierten Laufzeitklassen der ***enqueue(int v)***- und ***dequeue()***-Methode, die sich aus Ihrem Amortisierungsschema ergeben. Zeigen Sie die Richtigkeit Ihres Amortisierungsschemas (das Tokenkonto darf niemals negativ werden) und der resultierenden Laufzeitklassen.

```
1  class Stapelschlange {
2    private Stack s1 = new Stack();
3    private Stack s2 = new Stack();
4
5    public void enqueue(int v) {
6      s1.push(v);
7    }
8
9    public int dequeue() {
10     if(s2.isEmpty())
11       while(!s1.isEmpty())
12         s2.push(s1.pop());
13     return s2.pop();
14   }
15 }
```

Methode	Beschreibung	Laufzeitklasse
`void push(int v)`	legt eine Zahl auf den Stack	$\mathcal{O}(1)$
`int pop()`	nimmt die oberste Zahl vom Stack	$\mathcal{O}(1)$
`boolean isEmpty()`	prüft, ob der Stack leer ist	$\mathcal{O}(1)$

Aufgabe 5.3 (a)

Aufgabe 5.3 (b)