Übung Algorithmen und Datenstrukturen

Sommersemester 2016

Patrick Schäfer, Humboldt-Universität zu Berlin

Agenda

1. Besprechung des ersten Übungsblatts
2. Fragen zum zweiten Übungsblatt?
3. Beispielaufgaben

Landau Notation

$$O(g) = \{f \mid \exists c > 0 \;\; \exists n_0 > 0 \;\; \forall n \geq n_0 : \;\; f(n) \leq c \cdot g(n)\}$$

$$\Omega(g) = \{f \mid \exists c > 0 \;\; \exists n_0 > 0 \;\; \forall n \geq n_0 : \;\; f(n) \geq c \cdot g(n)\}$$

$$\boldsymbol{\Theta(g)} = \left\{ f \;\middle|\; \begin{array}{c} \exists c_1, c_2 > 0 \;\; \exists n_0 > 0 \\ \forall n \geq n_0 : \;\; c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \end{array} \right\}$$

$$o(g) = \{f \mid \forall c > 0 \;\; \exists n_0 > 0 \;\; \forall n \geq n_0 : \;\; f(n) < c \cdot g(n)\}$$

$$\omega(g) = \{f \mid \forall c > 0 \;\; \exists n_0 > 0 \;\; \forall n \geq n_0 : \;\; f(n) > c \cdot g(n)\}$$

Landau Notation

Zusammenhänge zwischen O, Ω, Θ, o und ω

- $f \in o(g) \Rightarrow f \in O(g)$ $f \in \omega(g) \Rightarrow f \in \Omega(g)$
- $f \in O(g) \Leftrightarrow g \in \Omega(f)$ $f \in o(g) \Leftrightarrow g \in \omega(f)$
- $f \in o(g) \Rightarrow f \notin \Omega(g)$ $f \in \omega(g) \Rightarrow f \notin O(g)$

Grenzwert als hinreichendes Kriterium

- $\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty \Rightarrow f \in O(g)$
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f \in o(g)$

Satz von L'Hôpital: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$, falls f und g differenzierbar sind und ihre Grenzwerte beide gegen 0 oder beide gegen ∞ gehen

Aufgabe: ABC (Stacks & Queues)

Entwerfen Sie einen Algorithmus $\boldsymbol{ABC}(a, b, c, w)$, der bei Eingabe von Zeichen a, b, c und Wort $w = a^k b^l c^m$ $(k, l, m \geq 0)$ testet, ob w die Form $a^n b^n c^n$ für ein $n \geq 0$ hat.

Die Zeichen a, b, c sind vom Datentyp Character. Das Wort w ist als Stack gegeben. Werte des Stacks sind vom Datentyp Character. Das 1. Zeichen des Wortes liegt oben auf dem Stack. Der Stack erlaubt die Operationen:

push(value), pop(), top(), isEmpty().

Als Datenstrukturen in Ihrem Algorithmus darf der gegebene Stack benutzt werden und (maximal) eine Queue. Die Queue erlaubt die Operationen:

enqueue(element), dequeue(), head(), isEmpty().

Ansonsten dürfen keine weiteren Datenstrukturen benutzt werden, auch keine primitiven Datentypen. Ausnahme: Es duürfen Boolsche Werte und Characters benutzt werden.

Aufgabe: Rangieren (Stacks & Queues)

Die Abbildungen zeigen Eisenbahngleise, welche einen Stack bzw. eine Queue mit Überholspur darstellen.

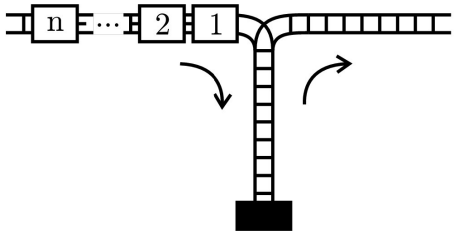

Abbildung 1: Stack

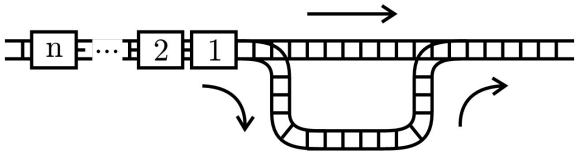

Abbildung 2: Queue

Auf der linken Seite stehen die absteigend nummerierten Waggons n bis 1, die auf dem Rangierwerk so umgestellt werden müssen, dass sie rechts in einer neuen Zusammenstellung herausfahren können.

Gibt es für einen Zug mit n = 5 Waggons (5, 4, 3, 2, 1) Rangiermöglichkeiten, so dass die folgenden Waggon-Zusammenstellungen auf der rechten Seite entstehen?

- 5,4,3,2,1
- 5,3,1,2,4
- 4,2,5,3,1
- 1,4,5,3,2

Aufgabe: Balancierte Klammerausdrücke (Stacks & Queues)

Bestimme, ob die Klammern in einem String balanciert sind.

String	Balanciert
([ ])	Ja
(([ ] [ ]))	Ja
( ]) [	Nein
) (	Nein
(( )	Nein