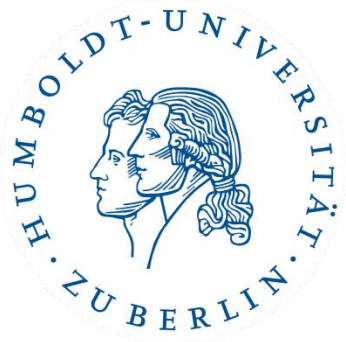
Übung Algorithmen und Datenstrukturen



Sommersemester 2016

Marc Bux, Humboldt-Universität zu Berlin

Organisatorisches

- Literaturempfehlung:
  Ottmann, Thomas, and Peter Widmayer. *Algorithmen und Datenstrukturen*. 5. Auflage, Springer-Verlag, 2011.
- Programmieraufgaben (Java 1.7) sind auf gruenau2 zu testen und in Goya abzugeben (gleicher Termin wie schriftliche Aufgaben)
- Wer sucht noch Anschluss an eine Übungsgruppe?

Agenda

1. Organisatorisches
2. Fragen zum ersten Übungsblatt
3. Landau-Notation (Zusammenfassung)
4. Vorstellung des zweiten Übungsblatts
5. Stacks und Queues
6. Infix- und Postfix-Notation

Landau-Notation (Zusammenfassung)

- $\mathrm{O}(g) = \left\{ f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0} \middle| \begin{array}{l} \exists c \in \mathbb{R} > 0 \quad \exists n_0 \in \mathbb{R} \geq 0 \\ \forall n \geq n_0 : \quad f(n) \leq c \cdot g(n) \end{array} \right\}$
- Zusammenhänge zwischen O, Ω, Θ, o und ω
  - $f \in o(g) \Rightarrow f \in \mathrm{O}(g)$ $\qquad$ $f \in \omega(g) \Rightarrow f \in \Omega(g)$
  - $f \in \mathrm{O}(g) \Leftrightarrow g \in \Omega(f)$ $\qquad$ $f \in o(g) \Leftrightarrow g \in \omega(f)$
  - $f \in o(g) \Rightarrow f \notin \Omega(g)$ $\qquad$ $f \in \omega(g) \Rightarrow f \notin \mathrm{O}(g)$
- Grenzwert als hinreichendes Kriterium
  - $\lim\limits_{n \to \infty} \frac{f(n)}{g(n)} < \infty \Rightarrow f \in \mathrm{O}(g)$
  - $\lim\limits_{n \to \infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f \in o(g)$
- Satz von L'Hôpital: $\lim\limits_{n \to \infty} \frac{f(n)}{g(n)} = \lim\limits_{n \to \infty} \frac{f'(n)}{g'(n)}$, falls f und g differenzierbar sind und ihre Grenzwerte beide gegen 0 oder beide gegen ∞ gehen

Aufgabe: Funktionen ordnen

Sie dürfen voraussetzen, dass $\lim_{n\to\infty} \ln n = \infty$ und $(\ln n)' = \frac{1}{n}$ ist. Beweisen Sie für die Funktionen in den Teilaufgaben (a) bis (d), dass $f \in \mathcal{O}(g)$.

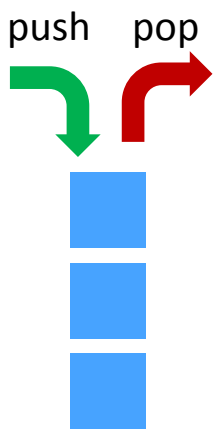
	$f(n)$	$g(n)$
(a)	$n+1$	$n+1000$
(b)	$\sqrt{n}$	$\sqrt[4]{n^3}$
(c)	2^{n+1}	2^n
(d)	2^n	$n!$

Agenda

1. Organisatorisches
2. Fragen zum ersten Übungsblatt
3. Landau-Notation (Zusammenfassung)
4. Vorstellung des zweiten Übungsblatts
5. Stacks und Queues
6. Infix- und Postfix-Notation

Stacks und Queues

- Stack / Stapel: dynamische last-in, first-out (LIFO) Datenstruktur
  - push(value): legt den Wert value oben auf dem Stack ab
  - pop(): entfernt den obersten Wert vom Stack und gibt ihn zurück
  - top() bzw. peek(): liefert den obersten Wert vom Stack ohne ihn dabei zu entfernen
  - isEmpty(): gibt *true* genau dann zurück, wenn der Stack leer ist



- Queue / Warteschlange: dynamische first-in, first-out (FIFO) Datenstruktur
  - enqueue(value) bzw. add(value): hängt den Wert value an das Ende der Queue an
  - dequeue() bzw. remove(value): entfernt den ersten Wert vom Anfang der Queue und gibt ihn zurück
  - head() bzw. peek(value): liefert den ersten Wert vom Anfang der Queue zurück ohne ihn dabei zu entfernen
  - isEmpty(): gibt *true* genau dann zurück, wenn die Queue leer ist

Aufgabe: Algorithmus ABC

Entwerfen Sie einen Algorithmus **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. Elemente des Stacks sind vom Datentyp Character. Das 1. Zeichen des Wortes liegt oben auf dem Stack. Der Stack erlaubt die folgenden Operationen:

- push(element): Legt das Element element oben auf dem Stack ab.
- pop(): Entfernt das oberste Element vom Stack und gibt es zurück. Wenn der Stack leer ist, wird der Wert null zurückgegeben.
- isEmpty(): Gibt true zurück, falls der Stack keine Elemente enthält und false andernfalls.

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

- enqueue(element): Hängt das Element element an das Ende der Queue an.
- dequeue(): Entfernt das erste Element vom Anfang der Queue und gibt es zurück. Wenn die Queue leer ist, wird der Wert null zurückgegeben.
- isEmpty(): Gibt true zurück, falls die Queue keine Elemente enthält und false andernfalls.

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

Agenda

1. Organisatorisches
2. Fragen zum ersten Übungsblatt
3. Landau-Notation (Zusammenfassung)
4. Vorstellung des zweiten Übungsblatts
5. Stacks und Queues
6. Infix- und Postfix-Notation

Infix- und Postfix-Notation

- Infix-Ausdrücke sind wie folgt rekursiv definiert:
  - eine Ziffer $d \in \{0,1,2,3,4,5,6,7,8,9\}$ ist ein Infix-Ausdruck
  - für alle Infix-Ausdrücke a_1, a_2 sind $(a_1 + a_2)$, $(a_1 - a_2)$, $(a_1 * a_2)$ und (a_1 / a_2) Infix-Ausdrücke
  - ungültig: leerer String, $\big((4 - (4 - 4))\big)$, $(3 + (4 - 7) - 2)$
- Postfix-Ausdrücke sind wie folgt rekursiv definiert:
  - eine Ziffer $d \in \{0,1,2,3,4,5,6,7,8,9\}$ ist ein Postfix-Ausdruck
  - für alle Postfix-Ausdrücke a_1, a_2 sind $a_1 a_2 +$, $a_1 a_2 -$, $a_1 a_2 *$ und $a_1 a_2 /$ Postfix-Ausdrücke

Infix-Ausdruck	Postfix-Ausdruck	Eval
$\big((2 - 5) + 6) * 2\big)$	$25 - 6 + 2 *$	6
$\big(3 - (7 + (2 * 9))\big)$	$3729 *+ -$	-22
$\big((5 * 5) + (2 - 5)\big)$	$55 * 25 - +$	22
$\big((4 - (4/2)) * (9 + 4)\big)$	$442/-94 +*$	26

Ausblick: Nächste Woche

- Klärung von Fragen zum zweiten Übungsblatt
- Besprechung der Lösungen zum ersten Übungsblatt
- Fragen?