Laufzeitanalyse von Algorithmen

Ziel: Algorithmen vergleichen, um vorherzusagen, welcher besser, schneller ist.

Möglichkeit 1, Empirisch:

Algorithmen implementieren, auf einem oder mehreren Rechnern auf diversen Eingaben laufen lassen und Laufzeiten vergleichen.

Problem: wenig Vorhersagekraft für andere Eingaben und andere Rechner.

Laufzeitanalyse von Algorithmen

Ziel: Algorithmen vergleichen, um vorherzusagen, welcher besser, schneller ist.

Möglichkeit 2, Analytisch:

Algorithmen bzgl. eines ***Laufzeitmodells*** analysieren und die ***Laufzeitfunktionen*** vergleichen

Probleme: Welches ***Laufzeitmodell*** ?
Wie vergleicht man ***Laufzeitfunktionen*** ?

Laufzeitanalyse von Algorithmen

Ziel: Algorithmen vergleichen, um vorherzusagen, welcher besser, schneller ist.

Möglichkeit 2, Analytisch:

Algorithmen bzgl. eines ***Laufzeitmodells*** analysieren und die ***Laufzeitfunktionen*** vergleichen

Probleme: Welches ***Laufzeitmodell*** ?
Wie vergleicht man ***Laufzeitfunktionen*** ?

Einfaches, abstraktes *Laufzeitmodell*:

Jede „primitive" Rechenoperation (arithmetische Operation wie +,*,-,/,mod,*etc.*
logische Opertaion wie **and**, **or**, *etc.,*
Speicherzugriff,
(bedingter) Sprungbefehl,
etc.)

braucht eine Zeiteinheit.

Laufzeitanalyse von Algorithmen

Ziel: Algorithmen vergleichen, um vorherzusagen, welcher besser, schneller ist.

Möglichkeit 2, Analytisch:

Algorithmen bzgl. eines ***Laufzeitmodells*** analysieren und die ***Laufzeitfunktionen*** vergleichen

Probleme: Welches ***Laufzeitmodell*** ?
Wie vergleicht man ***Laufzeitfunktionen*** ?

Laufzeitfunktion für primitives Laufzeitmodell

A Algorithmus E Eingabe

$T_A(E)$ = Anzahl der primitiven Operationen, die Algorithmus A braucht bei Abarbeitung von Eingabe E

Laufzeitanalyse von Algorithmen

Ziel: Algorithmen vergleichen, um vorherzusagen, welcher besser, schneller ist.

Möglichkeit 2, Analytisch:

Algorithmen bzgl. eines ***Laufzeitmodells*** analysieren und die ***Laufzeitfunktionen*** vergleichen

Probleme: Welches ***Laufzeitmodell*** ?
Wie vergleicht man ***Laufzeitfunktionen*** ?

Laufzeitfunktion für primitives Laufzeitmodell

A Algorithmus   E Eingabe

$T_A(E)$ = Anzahl der primitiven Operationen,
die Algorithmus A braucht bei Abarbeitung von Eingabe E

Problem: Wie vergleicht man Funktionen $T_A()$ und $T_B()$ für verschiedene Algorithmen A und B ?

Laufzeitanalyse von Algorithmen

Problem: Wie vergleicht man Funktionen $T_A()$ und $T_B()$ für verschiedene Algorithmen A und B ?

1. Menge der Eingaben in Klassen einteilen, z.B. nach Größe **n**
2. Innerhalb einer Klasse Funktionen zusammenfassen, z.B, nach
   schlechtesten Fall
   „durchschnittlichen" Fall
3. Die Zusammenfassungsfunktionen **asymptotisch** betrachten, also nach dem Wachstumsverhalten bei sehr großem Argument

WC-T_A(n) = max{ T_A(E) | E ist Eingabe der Größe n }

"Worst-case Laufzeit", also Laufzeit im schlechtesten Fall

Laufzeitanalyse von Algorithmen

Problem: Wie vergleicht man Funktionen $T_A()$ und $T_B()$ für verschiedene Algorithmen A und B ?

1. Menge der Eingaben in Klassen einteilen, z.B. nach Größe **n**
2. Innerhalb einer Klasse Funktionen zusammenfassen, z.B, nach
   schlechtesten Fall
   „durchschnittlichen“ Fall
3. Die Zusammenfassungsfunktionen **asymptotisch** betrachten, also nach dem Wachstumsverhalten bei sehr großem Argument

$\text{WC-}T_A(n) = \max\{ T_A(E) \mid E \text{ ist Eingabe der Größe } n \}$

“Worst-case Laufzeit”, also Laufzeit im schlechtesten Fall

$\text{WC-}T_A(n) = \text{Average}\{ T_A(E) \mid E \text{ ist Eingabe der Größe } n \}$

“Durchschnittliche Laufzeit”, also Laufzeit im "typischen Fall"

Aber: bezügliche welcher Verteilung? Begründung?

Laufzeitanalyse von Algorithmen

Problem: Wie vergleicht man Funktionen $T_A()$ und $T_B()$ für verschiedene Algorithmen A und B ?

1. Menge der Eingaben in Klassen einteilen, z.B. nach Größe **n**
2. Innerhalb einer Klasse Funktionen zusammenfassen, z.B, nach
   schlechtesten Fall
   „durchschnittlichen“ Fall
3. Die Zusammenfassungsfunktionen **asymptotisch** betrachten, also nach dem Wachstumsverhalten bei sehr großem Argument

Für Algorithmen *A* und *B* sind $WC\text{-}T_A(n)$ und $WC\text{-}T_B(n)$ einfache Funktionen $N \to R$

Welche Funktion ist "kleiner" ?

Asymptotischer Vergleich: Welche Funktion wächst langsamer?
Welche Funktion ist kleiner für sehr großes **n**?

Def: $f,g : \mathbb{N} \to \mathbb{R}$

$f \leq_{fü} g$: für alle n, bis auf endlich viele Ausnahmen gilt $f(n) \leq g(n)$

$(\exists n_0 \in \mathbb{N} \; \forall n \geq n_0 : f(n) \leq g(n))$

Def: $g \in \text{FÜP} = \{ h : \mathbb{N} \to \mathbb{R} \mid h \geq_{fü} 0 \}$

$$O(g) = \{ f \in \text{FÜP} \mid \exists c>0 : f \leq_{fü} c \cdot g \}$$

$$o(g) = \{ f \in \text{FÜP} \mid \forall c>0 : f \leq_{fü} c \cdot g \}$$

$$\Omega(g) = \{ f \in \text{FÜP} \mid \exists c>0 : f \geq_{fü} c \cdot g \}$$

$$\omega(g) = \{ f \in \text{FÜP} \mid \forall c>0 : f \geq_{fü} c \cdot g \}$$

$$\Theta(g) = O(g) \cap \Omega(g)$$

Def: $g \in \text{FÜP} = \{ h : \mathbb{N} \to \mathbb{R} \mid h \geq_{\text{fü}} 0 \}$

$$O(g) = \{ f \in \text{FÜP} \mid \exists c>0 : f \leq_{\text{fü}} c \cdot g \}$$

$$o(g) = \{ f \in \text{FÜP} \mid \forall c>0 : f \leq_{\text{fü}} c \cdot g \}$$

$$\Omega(g) = \{ f \in \text{FÜP} \mid \exists c>0 : f \geq_{\text{fü}} c \cdot g \}$$

$$\omega(g) = \{ f \in \text{FÜP} \mid \forall c>0 : f \geq_{\text{fü}} c \cdot g \}$$

$$\Theta(g) = O(g) \cap \Omega(g)$$

$f \in O(g)$ entspricht f "$\leq$" g

$f \in o(g)$ entspricht f "$<$" g

$f \in \Omega(g)$ entspricht f "$\geq$" g

$f \in \omega(g)$ entspricht f "$>$" g

$f \in \Theta(g)$ entspricht f "$=$" g

Wichtige Regeln

$f \in O(g) \Leftrightarrow g \in \Omega(f)$

$f \in o(g) \Leftrightarrow g \in \omega(f)$ $\qquad$ $f \in O(g) \wedge g \in O(h) \Rightarrow f \in O(h)$

$f \in \Theta(g) \Leftrightarrow g \in \Theta(f)$ $\qquad$ $f \in o(g) \wedge g \in o(h) \Rightarrow f \in o(h)$

Wenn $f_1 \in O(g_1)$, $f_2 \in O(g_2)$

dann $\qquad f_1 + f_2 \in O(\max\{g_1, g_2\})$

$\qquad\qquad f_1 \cdot f_2 \in O(g_1 \cdot g_2)$

$0<a<b \quad 0<\alpha<\beta \quad 1<A<B$

$\log^\alpha n$ "<" $\log^\beta n$ "<" n^a "<" $n^a \log^\alpha n$ "<" n^b "<" A^n "<" $n^a A^n$ "<" B^n

Wichtige Regeln

$f, g \in$ FÜP

$$\lim_{n\to\infty} f(n)/g(n) = 0 \quad \Rightarrow \quad f \in o(g)$$

$$\lim_{n\to\infty} f(n)/g(n) = \infty \quad \Rightarrow \quad f \in \omega(g)$$

$$\exists\, 0<c<\infty : \lim_{n\to\infty} f(n)/g(n) = c \quad \Rightarrow \quad f \in O(g)$$