Übung “Grundbegriffe der Informatik”

Karlsruher Institut für Technologie

Matthias Schulz, Gebäude 50.34, Raum 034

email: schulz@ira.uka.de

Der Warshall-Algorithmus

Matrix quadrieren:

$B = A \cdot A$

.

Der Warshall-Algorithmus

Matrix quadrieren:

$B = A \cdot A$

Initialisiere $B = 0$.

.

Der Warshall-Algorithmus

Matrix quadrieren:

$B = A \cdot A$

Initialisiere $B = 0$.

```
for i=0 to n-1 do
   for j=0 to n-1 do
     for k=0 to n-1 do
     B[i][j] ← B[i][j] + A[i][k] · A[k][j]
    od
  od
od
```

Der Warshall-Algorithmus

Matrix quadrieren:

$B = A \cdot A$

Initialisiere $B = 0$.

```
for k=0 to n-1 do
   for i=0 to n-1 do
     for j=0 to n-1 do
     B[i][j] ← B[i][j] + A[i][k] · A[k][j]
    od
  od
od
```

Variation des Algorithmus:

```
for k=0 to n-1 do
   for i=0 to n-1 do
      for j=0 to n-1 do
      A[i][j] ← A[i][j] + A[i][k] · A[k][j]
    od
  od
od
```

.

Der Warshall-Algorithmus

Variation des Algorithmus

```
for k=0 to n-1 do
   for i=0 to n-1 do
     for j=0 to n-1 do
     A[i][j] ← A[i][j] + A[i][k] · A[k][j]
    od
  od
od
```

Fehler 1: Keine Initialisierung!

.

Der Warshall-Algorithmus

Variation des Algorithmus

```
for k=0 to n-1 do
   for i=0 to n-1 do
     for j=0 to n-1 do
      A[i][j] ← A[i][j] + A[i][k] · A[k][j]
     od
   od
od
```

Fehler 1: Keine Initialisierung!
Fehler 2: Werte, mit denen gerechnet wird, werden geändert!

.

Der Warshall-Algorithmus

Variation des Algorithmus

```
for k=0 to n-1 do
   for i=0 to n-1 do
     for j=0 to n-1 do
     A[i][j] ← A[i][j] + A[i][k] · A[k][j]
    od
  od
od
```

Wann ist $A[i][j]$ am Ende ungleich 0?

.

Der Warshall-Algorithmus

Variation des Algorithmus

```
for k=0 to n−1 do
   for i=0 to n−1 do
     for j=0 to n−1 do
     A[i][j] ← A[i][j] + A[i][k] · A[k][j]
    od
  od
od
```

Wann ist $A[i][j]$ am Ende ungleich 0?
Durchlauf k: $A[i][j]$ ungleich 0, falls $A[i][j]$ vorher ungleich 0,

Der Warshall-Algorithmus

Variation des Algorithmus

```
for k=0 to n-1 do
   for i=0 to n-1 do
     for j=0 to n-1 do
     A[i][j] ← A[i][j] + A[i][k] · A[k][j]
    od
  od
od
```

Wann ist $A[i][j]$ am Ende ungleich 0?
Durchlauf k: $A[i][j]$ ungleich 0, falls $A[i][j]$ vorher ungleich 0,
oder sowohl $A[i][k]$ als auch $A[k][j]$ ungleich 0.

Der Warshall-Algorithmus

Variation des Algorithmus

```
for k=0 to n−1 do
   for i=0 to n−1 do
     for j=0 to n−1 do
     A[i][j] ← A[i][j] + A[i][k] · A[k][j]
    od
  od
od
```

Wann ist $A[i][j]$ am Ende ungleich 0?
Durchlauf k: $A[i][j]$ ungleich 0, falls
$\max\{A[i][j], \min\{A[i][k], A[k][j]\}\}$ ungleich 0.

Der Warshall-Algorithmus

Weitere Variation des Algorithmus

```
for k=0 to n-1 do
   for i=0 to n-1 do
     for j=0 to n-1 do
     A[i][j] ← max{A[i][j], min{A[i][k], A[k][j]}}
    od
  od
od
```

Wann ist $A[i][j]$ am Ende ungleich 0?
Durchlauf k: $A[i][j]$ ungleich 0, falls
$\max\{A[i][j], \min\{A[i][k], A[k][j]\}\}$ ungleich 0.

Weitere Variation des Algorithmus

```
for k=0 to n−1 do
   for i=0 to n−1 do
     for j=0 to n−1 do
     A[i][j] ← max{A[i][j], min{A[i][k], A[k][j]}}
   od
  od
od
```

$\rightarrow$ Zweiter Teil des Warshall-Algorithmus nach Initialisierung!

.

Der Warshall-Algorithmus

Weitere Variation des Algorithmus

```
for k=0 to n-1 do
   for i=0 to n-1 do
     for j=0 to n-1 do
      A[i][j] ← sgn(A[i][j] + A[i][k] · A[k][j])
    od
  od
od
```

$\rightarrow$ Zweiter Teil des Warshall-Algorithmus nach Initialisierung!

.

O-Kalkül

Algorithmus mit Laufzeit in $\Theta(n^3)$

Jede Woche soll der Algorithmus ein Problem lösen.

Wie groß darf Problem sein?

.

O-Kalkül

Algorithmus mit Laufzeit in $\Theta(n^3)$

Jede Woche soll der Algorithmus ein Problem lösen.

Wie groß darf Problem sein? $\rightarrow$ Unbekannt, m

.

O-Kalkül

Algorithmus mit Laufzeit in $\Theta(n^3)$

Jede Woche soll der Algorithmus ein Problem lösen.

Neuer Rechner 8-mal so schnell:

Wie groß darf Problem sein?

.

O-Kalkül

Algorithmus mit Laufzeit in $\Theta(n^3)$

Jede Woche soll der Algorithmus ein Problem lösen.

Neuer Rechner 8-mal so schnell:

Wie groß darf Problem sein? Ungefähr $2m$

.

O-Kalkül

Algorithmus mit Laufzeit in $\Theta(n^d)$

Jede Woche soll der Algorithmus ein Problem lösen.

Neuer Rechner k-mal so schnell:

Wie groß darf Problem sein? Ungefähr $\sqrt[d]{k} \cdot m$

.

O-Kalkül

Algorithmus mit Laufzeit in $\Theta(d^n)$

Jede Woche soll der Algorithmus ein Problem lösen.

Neuer Rechner k-mal so schnell:

Wie groß darf Problem sein? Ungefähr $m + \log_d k$

.

k fest, $f : \mathbb{N}_0 \to \mathbb{N}_0$

$$n \mapsto \begin{cases} \binom{n}{k} & \text{falls } n \geq k \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f \in \Theta(n^k)$

.

O-Kalkül

k fest, $f : \mathbb{N}_0 \to \mathbb{N}_0$

$$n \mapsto \begin{cases} \binom{n}{k} & \text{falls } n \geq k \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f \in O(n^k)$ und $f \in \Omega(n^k)$

.

O-Kalkül

k fest, $f : \mathbb{N}_0 \to \mathbb{N}_0$

$$n \mapsto \begin{cases} \binom{n}{k} & \text{falls } n \geq k \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f \in O(n^k)$:

.

O-Kalkül

k fest, $f : \mathbb{N}_0 \to \mathbb{N}_0$

$$n \mapsto \begin{cases} \binom{n}{k} & \text{falls } n \geq k \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f \in O(n^k)$:

Sei $n \geq k$.

.

k fest, $f : \mathbb{N}_0 \to \mathbb{N}_0$

$$n \mapsto \begin{cases} \binom{n}{k} & \text{falls } n \geq k \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f \in O(n^k)$:

Sei $n \geq k$.

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} = \prod_{i=1}^{k} \frac{n+1-i}{i} \leq \prod_{i=1}^{k} \frac{n}{1} = n^k$$

.

k fest, $f : \mathbb{N}_0 \to \mathbb{N}_0$

$$n \mapsto \begin{cases} \binom{n}{k} & \text{falls } n \geq k \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f \in O(n^k)$:

Sei $n \geq k$.

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} = \prod_{i=1}^{k} \frac{n+1-i}{i} \leq \prod_{i=1}^{k} \frac{n}{1} = n^k$$

$\Rightarrow n_0 = k, c = 1$ funktioniert.

.

O-Kalkül

k fest, $f : \mathbb{N}_0 \to \mathbb{N}_0$

$$n \mapsto \begin{cases} \binom{n}{k} & \text{falls } n \geq k \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f \in O(n^k)$:

Sei $n \geq k$.

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} = \prod_{i=1}^{k} \frac{n+1-i}{i} \leq \prod_{i=1}^{k} \frac{n}{1} = n^k$$

$$f \in \{g : \mathbb{N}_0 \to \mathbb{R}_0^+ \mid \forall n \geq k : g(n) \leq 1 \cdot n^k\}$$

.

O-Kalkül

k fest, $f : \mathbb{N}_0 \to \mathbb{N}_0$

$$n \mapsto \begin{cases} \binom{n}{k} & \text{falls } n \geq k \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f \in O(n^k)$:

Sei $n \geq k$.

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} = \prod_{i=1}^{k} \frac{n+1-i}{i} \leq \prod_{i=1}^{k} \frac{n}{1} = n^k$$

$f \in \{g : \mathbb{N}_0 \to \mathbb{R}_0^+ \mid \forall n \geq k : g(n) \leq 1 \cdot n^k\}$
$\subseteq \{g : \mathbb{N}_0 \to \mathbb{R}_0^+ \mid \exists n_0 \in \mathbb{N}_0 \exists c \in \mathbb{R}_+ :$
$\forall n \geq n_0 : g(n) \leq c \cdot n^k\} = O(n^k)$

k fest, $f : \mathbb{N}_0 \to \mathbb{N}_0$

$$n \mapsto \begin{cases} \binom{n}{k} & \text{falls } n \geq k \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f \in \Omega(n^k)$:

.

k fest, $f : \mathbb{N}_0 \to \mathbb{N}_0$

$$n \mapsto \begin{cases} \binom{n}{k} & \text{falls } n \geq k \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f \in \Omega(n^k)$:

Sei $n \geq k$.

.

k fest, $f : \mathbb{N}_0 \to \mathbb{N}_0$

$$n \mapsto \begin{cases} \binom{n}{k} & \text{falls } n \geq k \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f \in \Omega(n^k)$:

Sei $n \geq k$.

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} = \prod_{i=1}^{k} \frac{n+1-i}{i} \geq \prod_{i=1}^{k} \frac{n+1-k}{k} = \frac{1}{k^k}(n+1-k)^k$$

.

k fest, $f : \mathbb{N}_0 \to \mathbb{N}_0$

$$n \mapsto \begin{cases} \binom{n}{k} & \text{falls } n \geq k \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f \in \Omega(n^k)$:

Sei $n \geq k$.

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} = \prod_{i=1}^{k} \frac{n+1-i}{i} \geq \prod_{i=1}^{k} \frac{n+1-k}{k} = \frac{1}{k^k}(n+1-k)^k$$

Idee: Wenn n hinreichend groß, lässt sich $(n+1-k)$ durch $\frac{n}{2}$ abschätzen.

.

k fest, $f : \mathbb{N}_0 \to \mathbb{N}_0$

$$n \mapsto \begin{cases} \binom{n}{k} & \text{falls } n \geq k \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f \in \Omega(n^k)$:

Sei $n \geq 2k$.

.

k fest, $f : \mathbb{N}_0 \to \mathbb{N}_0$

$$n \mapsto \begin{cases} \binom{n}{k} & \text{falls } n \geq k \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f \in \Omega(n^k)$:

Sei $n \geq 2k$.

$$\Rightarrow k \leq \frac{n}{2} \Rightarrow n + 1 - k \geq n + 1 - \frac{n}{2} = \frac{n}{2} + 1 \geq \frac{n}{2}.$$

.

k fest, $f : \mathbb{N}_0 \to \mathbb{N}_0$

$$n \mapsto \begin{cases} \binom{n}{k} & \text{falls } n \geq k \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f \in \Omega(n^k)$:

Sei $n \geq 2k$.

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} = \prod_{i=1}^{k} \frac{n+1-i}{i} \geq \prod_{i=1}^{k} \frac{n+1-k}{k} = \frac{1}{k^k}(n+1-k)^k$$
$$\geq \frac{1}{k^k}\left(\frac{n}{2}\right)^k = \frac{1}{(2k)^k} n^k$$

.

k fest, $f : \mathbb{N}_0 \to \mathbb{N}_0$

$$n \mapsto \begin{cases} \binom{n}{k} & \text{falls } n \geq k \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f \in \Omega(n^k)$:

Sei $n \geq 2k$.

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} = \Pi_{i=1}^{k} \frac{n+1-i}{i} \geq \Pi_{i=1}^{k} \frac{n+1-k}{k} = \frac{1}{k^k}(n+1-k)^k$$
$$\geq \frac{1}{k^k}(\frac{n}{2})^k = \frac{1}{(2k)^k} n^k$$

$\Rightarrow n_0 = 2k$ und $c = \frac{1}{(2k)^k}$ funktionieren.

.

O-Kalkül

$f(n) = 1,1^n, g(n) = n^{100}$

Zeige: $f(n) \notin O(g(n))$.

.

O-Kalkül

$f(n) = 1,1^n, g(n) = n^{100}$

Zeige: $f(n) \notin O(g(n))$.

Vorgehen: Zeige, dass $\frac{f(n)}{g(n)}$ für große n größer als jedes feste $c \geq 0$ wird.

.

O-Kalkül

$f(n) = 1,1^n, g(n) = n^{100}$

Zeige: $f(n) \notin O(g(n))$.

Annahme: Für $n_0 \in \mathbb{N}_0$ und $c > 0$ gilt

$\forall n \geq n_0 : f(n) \leq cg(n)$

.

O-Kalkül

$f(n) = 1,1^n, g(n) = n^{100}$

Zeige: $f(n) \notin O(g(n))$.

Annahme: Für $n_0 \in \mathbb{N}_0$ und $c > 0$ gilt

$\forall n \geq n_0 : f(n) \leq cg(n)$

Definiere $c_0 = \frac{f(n_0)}{g(n_0)}$

.

O-Kalkül

$f(n) = 1,1^n, g(n) = n^{100}$

Zeige: $f(n) \notin O(g(n))$.

Annahme: Für $n_0 \in \mathbb{N}_0$ und $c > 0$ gilt

$\forall n \geq n_0 : f(n) \leq cg(n)$

$n_{i+1} = 2n_i$
$c_i = \frac{f(n_i)}{g(n_i)}$

.

$f(n) = 1,1^n, g(n) = n^{100}$

Zeige: $f(n) \notin O(g(n))$.

Annahme: Für $n_0 \in \mathbb{N}_0$ und $c > 0$ gilt

$\forall n \geq n_0 : f(n) \leq cg(n)$

$n_{i+1} = 2n_i$
$c_i = \frac{f(n_i)}{g(n_i)}$

$$c_{i+1} = \frac{f(n_{i+1})}{g(n_{i+1})} = \frac{f(2n_i)}{g(2n_i)} = \frac{1{,}1^{2n_i}}{(2n_i)^{100}} = \frac{1{,}1^{n_i} \cdot 1{,}1^{n_i}}{2^{100} \cdot n_i^{100}}$$
$$= \frac{1{,}1^{n_i}}{2^{100}} \cdot \frac{1{,}1^{n_i}}{n_i^{100}} = \frac{1{,}1^{n_i}}{2^{100}} \cdot c_i$$

O-Kalkül

$f(n) = 1,1^n, g(n) = n^{100}$

Zeige: $f(n) \notin O(g(n))$.

Annahme: Für $n_0 \in \mathbb{N}_0$ und $c > 0$ gilt

$\forall n \geq n_0 : f(n) \leq cg(n)$

$\frac{1,1^{n_i}}{2^{100}} > 2$ für $n_i > \log_{1,1} 2^{101}$.

.

O-Kalkül

$f(n) = 1,1^n, g(n) = n^{100}$

Zeige: $f(n) \notin O(g(n))$.

Annahme: Für $n_0 \in \mathbb{N}_0$ und $c > 0$ gilt

$\forall n \geq n_0 : f(n) \leq cg(n)$

$\frac{1{,}1^{n_i}}{2^{100}} > 2$ für $n_i > \log_{1,1} 2^{101}$.

m_0 kleinstes n_i mit $n_i > \log_{1,1} 2^{101}$.
$d_0 = \frac{f(m_0)}{g(m_0)}$

.

O-Kalkül

$f(n) = 1{,}1^n, g(n) = n^{100}$

Zeige: $f(n) \notin O(g(n))$.

Annahme: Für $n_0 \in \mathbb{N}_0$ und $c > 0$ gilt

$\forall n \geq n_0 : f(n) \leq cg(n)$

$m_{i+1} = 2m_i$
$d_{i+1} = \frac{f(m_{i+1})}{g(m_{i+1})} = \frac{1{,}1^{m_i}}{2^{100}} \cdot d_i > 2d_i$

.

$f(n) = 1,1^n, g(n) = n^{100}$

Zeige: $f(n) \notin O(g(n))$.

Annahme: Für $n_0 \in \mathbb{N}_0$ und $c > 0$ gilt

$\forall n \geq n_0 : f(n) \leq cg(n)$

$m_{i+1} = 2m_i$
$d_{i+1} = \frac{f(m_{i+1})}{g(m_{i+1})} = \frac{1{,}1^{m_i}}{2^{100}} \cdot d_i > 2d_i$

Nach Annahme gilt $c > d_0$ und damit $q = \frac{c}{d_0} > 1$ und $\lceil \log_2 q \rceil \geq 1$.

.

O-Kalkül

$f(n) = 1{,}1^n, g(n) = n^{100}$

Zeige: $f(n) \notin O(g(n))$.

Annahme: Für $n_0 \in \mathbb{N}_0$ und $c > 0$ gilt

$\forall n \geq n_0 : f(n) \leq cg(n)$

$m_{i+1} = 2m_i$
$d_{i+1} = \frac{f(m_{i+1})}{g(m_{i+1})} = \frac{1{,}1^{m_i}}{2^{100}} \cdot d_i > 2d_i$

$d_{\lceil \log_2 q \rceil} > 2^{\lceil \log_2 q \rceil} \cdot d_0 \geq 2^{\log_2 q} \cdot d_0 = q \cdot d_0 = \frac{c}{d_0} \cdot d_0 = c$

Dies ist ein Widerspruch!