Effiziente Algorithmen

1. Vorlesung
05.04.2011
EINFÜHRUNG

Algorithmisches Problem:
Besteht aus: Eingabemenge \( \text{In} \)
Ausgabemenge \( \text{Out} \)
eindeutiger Vorschrift \( f: \text{In} \rightarrow \text{Out} \)
Definition der Problemgröße
\( s: \text{In} \rightarrow \mathbb{N} \)

Beispiele:
① Primzahlproblem
\( \text{In}: \mathbb{N} \) \hspace{1em} \( \text{Out}: \{0, 1, 2\} \)
\[ f(0) = 0 \text{ nicht Primzahl} \]
\[ f(2) = 1 \text{ nicht Primzahl} \]
\[ s(n) = \log(n) \]
② Sortieren
\( \text{In}: \) Folge von Objekten aus einer streng geordneten Menge
\( \text{Out}: \) " "
\[ f(a_1, \ldots, a_k) = (a_1', a_2', \ldots, a_k') \]
Wobei \( (a_1', a_2', \ldots) \) Permutation von \( (a_1, \ldots, a_k) \)
nach \( a_i, a_j \) \( i < j \) \( a_i' < a_j' \) gilt
\[ s(a_1, \ldots, a_k) = k \]
Für Algorithmus A ist ein deterministisches Verfahren, welches für jede Eingabe eines Problems in einer endlichen Anzahl elementarer Schritte die zugehörige Ausgabe berechnet.

- Sei \( t_A(w) \) = \# elementarer Schritte von A bei Eingabe w
  schwierig abzuschätzen

- \( T(w) = \max \ 2 \ t_A(w) \) \( s(w) = n^3 \)
  worst-case Laufzeit

Für die Beschreibung von \( T(w) \) geben wir meist asymptotische Schranken an
\( \geq \) groß-\( O \)-Notation
Schnitte für asymptotisches Wachstum

- Obere Schnitte

In den meisten Fällen wollen wir $T(n)$ von oben beschränken

$$T(n) = O(f(n)) : \Leftrightarrow \exists c > 0 \forall n \geq n_0$$

$T(n) \leq c \cdot f(n)$

Obere Schnitte

- Untere Schnitte

Tauchen weniger häufig auf.
Heißt zum Messen der Komplexität von Problemen, oder bei der Abschätzung wie gut die Analyse eines Algorithmus war.
$T(u) = \Omega (g(u)) : \Rightarrow \exists c_1 > 0 \forall u \geq u_0$

\[ T(u) \geq c \cdot g(u) \]

- Scharfe Schranke

Wenn

$T(u) = O (f(u))$ und

$T(u) = \Omega (f(u))$ schreiben wir:

$T(u) = \Theta (f(u))$

Alternative: $T(u) = \Theta (f(u)) : \Rightarrow \exists c_1 > 0 \exists c_2 > 0 \forall u \geq u_0$

$0 \leq f(u) \leq T(u) \leq c_2 f(u)$

- andere Schranke

Durch Tauschen von $u \leq \Theta (\overline{u})$

wird $O(f(u)) = \Theta (f(u))$ ($\Omega (gf(u)) = \omega (gf(u))$

Bringen wir in diesem Fall ebenfalls nicht...
Ungang und asympt. Schwank.

- Für konstante $c, x : (c > 0)$
  \[ T(n) = c \cdot f(n) + x \implies T(n) = O(c f(n)) \]

- Für $T(n) = a_0 + a_1 n + a_2 n^2 \ldots + a_k n^k$, $a_k > 0$ (Polynom vom Grad $k$)
  \[ \implies T(n) = O(n^k) \]

- Für alle $b > 1, x > 1, \varepsilon > 0$
  \[ (\log_b n)^x = O(n^\varepsilon) \]

- Für jedes $r > 1$ und jedes $d > 0$
  \[ n^d = O(r^n) \]

Konvention: $O(1) = \text{Konstante}$

Funktion $f : \mathbb{N} \to \mathbb{N}$
Was sind eigentlich "elementare Schritte" 
Hochzähler
-> Rechenschema

REGISTERMASCHINE (RAM)

- Daten stehen in Registern & Speicher
- Es gibt keine Nebenläufigkeit
- Es gibt so viele Speicherzellen
- Befehlsatz:
  * arithmetische Op: \( +, -, \times, \div \)
  * Lade-/Speicher Op: LOAD, STORE
  * R < M, R -> M
  * Kontroll Op: bedingte & unbedingte Verzweigungen

- Datenarten: Integers & Floats
- jede Speicherzelle / Register kann
  eine 0-logen Bit große Zahl
  speichern (\( C \geq 1 \))

  (Indizierung des Inputs möglich aber)

  (nicht für große Speicherzellen)
Die Ausführung eines Befehles (opn) auf der RAM kostet eine Einheit
→ Einheitskostenwagnis

• Einige andere Opns sind nicht zulässig, weil sie auf Standard CPUs nicht in \( O(1) \) Takt zeiten berechnet werden können: z.B. \( x^y \)

Aber \( 2^x = 1 \ll x \) ist erlaubt.

Links Shift

Beschreibung von Algorithmen

• Entweder verbal oder in Pseudo-Code

• Pseudo-Code (in der VL)
  -Pascal-Stil, d.h. Englische Bezeichner
    - for, while, if, then, else, foreach

• Zuweisung: \( x \leftarrow y \)

• Vergleich: \( x = y \)

• Kompliziertere Berechnungsvorschriften werden verbal beschrieben

• Häufig Mengennotation

• Attribut von \( X = \{ \text{B. X. Sohn} \} \)
Beispiel

**BUBBLE SORT** (A)

1. FOR i = 1 to A.Länge - 1
2.      FOR j = A.Länge down to i + 1
5.      ENDFOR
6.  ENDFOR
7. RETURN (A)