

Komplexität von Berechnungen

Notation: $\ell(x) = \lfloor \log(|x|) \rfloor + 1$: Länge einer ganzen Zahl x

$\mathbf{x} = (x_1, \dots, x_m) \in \mathbb{Z}^m$: Eingabe einer RAM

$\|\mathbf{x}\|_u = m$: *uniforme* Grösse der Eingabe

$\|\mathbf{x}\|_{\log} = \sum_{i=1}^m \ell(x_i)$: *logarithmische* Grösse der Eingabe

Laufzeitkomplexität

$T_M^u(\mathbf{x})$: *uniforme* Zeitkomplexität der RAM M auf Eingabe \mathbf{x} :
Anzahl der ausgeführten Programmschritte

$$T_M^u(n) = \max\{T_M^u(\mathbf{x}) : \|\mathbf{x}\|_u = n\}$$

M heisst *uniform- $t(n)$ -zeitbeschränkt*, wobei $t : \mathbb{N} \rightarrow \mathbb{N}$, wenn

$$T_M^u(n) \leq t(n) \quad (n \in \mathbb{N})$$

Dies ist ein *worst-case* Komplexitätsbegriff

$T_M^{\log}(\mathbf{x})$: *logarithmische* Zeitkomplexität der RAM M auf Eingabe \mathbf{x} :
Summe der logarithmischen Kosten der ausgeführten Programmschritte

$$T_M^{\log}(n) = \max\{T_M^{\log}(\mathbf{x}) : \|\mathbf{x}\|_{\log} = n\}$$

M heisst *logarithmisch- $t(n)$ -zeitbeschränkt*, wobei $t : \mathbb{N} \rightarrow \mathbb{N}$, wenn

$$T_M^{\log}(n) \leq t(n) \quad (n \in \mathbb{N})$$

Dies ist ein *worst-case* Komplexitätsbegriff

average-case Laufzeitkomplexität

$\Phi = (\phi_n)_{n \in \mathbb{N}}$: Familie von Wahrscheinlichkeitsverteilungen,
wobei ϕ_n WV auf Eingaben der (uniformen oder logarithmischen) Grösse n
erwartete Laufzeit auf Eingaben der uniformen Grösse n :

$$\bar{T}_{M,\Phi}^u(n) = \sum_{\|\mathbf{x}\|_u=n} \phi_n(\mathbf{x}) \cdot T_M^u(\mathbf{x})$$

erwartete Laufzeit auf Eingaben der logarithmischen Grösse n :

$$\bar{T}_{M,\Phi}^{\log}(n) = \sum_{\|\mathbf{x}\|_{\log}=n} \phi_n(\mathbf{x}) \cdot T_M^{\log}(\mathbf{x})$$

M heisst *im Mittel uniform- $t(n)$ -zeitbeschränkt*, bzw. *im Mittel logarithmisch- $t(n)$ -zeitbeschränkt* wobei $t : \mathbb{N} \rightarrow \mathbb{N}$, wenn

$$T_{M,\Phi}^u(n) \leq t(n) \quad \text{bzw.} \quad T_{M,\Phi}^{\log}(n) \leq t(n) \quad (n \in \mathbb{N})$$

Grundsätzliche Bemerkungen zur *average-case* Komplexität

- *average-case* Komplexität ist aus praktischer Sicht oft ein realistischeres Kostenmass als *worst-case* Komplexität
- Eine sinnvolle Wahl der Verteilungen $\Phi = (\phi_n)_{n \in \mathbb{N}}$ auf den Eingaben hängt von den à priori Informationen über die erwarteten Eingaben ab — hat man keine solchen Informationen, behilft man sich mit der Annahme, dass jedes ϕ_n die Gleichverteilung auf $\mathbf{X}_u(n) = \{\mathbf{x}; \|\mathbf{x}\|_u = n\}$ bzw. $\mathbf{X}_{\log}(n) = \{\mathbf{x}; \|\mathbf{x}\|_{\log} = n\}$ ist. Dann gilt

$$\bar{T}_{M,\Phi}^u(n) = \frac{1}{\#\mathbf{X}_u(n)} \sum_{\mathbf{x} \in \mathbf{X}_u(n)} T_M^u(\mathbf{x})$$

bzw.

$$\bar{T}_{M,\Phi}^{\log}(n) = \frac{1}{\#\mathbf{X}_{\log}(n)} \sum_{\mathbf{x} \in \mathbf{X}_{\log}(n)} T_M^{\log}(\mathbf{x})$$

- Selbst unter der vereinfachenden Annahme der Gleichverteilung: *average-case* Komplexität ist meist erheblich schwieriger zu bestimmen als *worst-case* Komplexität, sogar bei konzeptuell sehr einfachen Algorithmen
- Bei der berühmt-berüchtigten Problemstellung **P-vs-NP** handelt es sich um eine Fragestellung der *worst-case* Komplexität! Es macht einen guten Sinn, diese Problemstellung (*effizientes Berechnen* vs. *effizientes Verifizieren*) auch im Rahmen der *average-case* zu untersuchen — das ist erst in jüngster Zeit mit einigem Erfolg gelungen.

Randomisierte Algorithmen

Wenn man Algorithmen Zugriff auf Zufallszahlen/bits erlaubt, werden auch bei fester Eingabe Laufzeit und Resultat einer Berechnung variieren und müssen als Zufallsvariable betrachtet werden, für deren mittleres Verhalten (Erwartungswert) man sich interessiert.

Solche Algorithmen können u.U. sehr viel effizienter als deterministische Algorithmen für dasselbe Problem sein — wobei man ein ungünstiges Verhalten (falsches Ergebnis, lange Laufzeit) als seltenes Ereignis in Kauf nimmt.

Eine *randomisierte Registermaschine* hat als zusätzlichen Befehl **RAND**, wobei **RAND** m eine Zahl aus $[0..m-1] = \{0, 1, 2, \dots, m-1\}$ mit Gleichverteilung liefert

$\hat{T}_M^u(\mathbf{x})$: erwartete *uniforme* Zeitkomplexität einer randomisierten RAM M auf Eingabe \mathbf{x}

$$\hat{T}_M^u(n) = \max\{\hat{T}_M^u(\mathbf{x}) : \|\mathbf{x}\|_u = n\}$$

M heisst *uniform- $t(n)$ -erwartet-zeitbeschränkt*, wenn

$$\hat{T}_M^u(n) \leq t(n) \quad (n \in \mathbb{N})$$

$\hat{T}_M^{\log}(\mathbf{x})$: erwartete *logarithmische* Zeitkomplexität einer randomisierten RAM M auf Eingabe \mathbf{x}

$$\hat{T}_M^{\log}(n) = \max\{\hat{T}_M^{\log}(\mathbf{x}) : \|\mathbf{x}\|_{\log} = n\}$$

M heisst *logarithmisch- $t(n)$ -erwartet-zeitbeschränkt*, wenn

$$\hat{T}_M^{\log}(n) \leq t(n) \quad (n \in \mathbb{N})$$

Hinweise

- Normalerweise analysiert man RAM-Programme und ihre Laufzeiten nicht im Detail, sondern orientiert sich an den für einen Algorithmus charakteristischen Operationen (Vergleiche, arithmetische Operationen, Speicheroperationen, ...)
- Relevant ist, ob die Argumente der charakteristischen Operationen von beschränkter Grösse sind (\Rightarrow uniformes Kostenmass) oder sehr gross werden können (\Rightarrow logarithmisches Kostenmass)

Platzkomplexität

uniform: Anzahl der von einer RAM benutzten Register

logarithmisch:

$$S_M^{\log}(\mathbf{x}) = \sum_i \max\{\ell(j) ; j \text{ kommt bei Berechnung in Register } i \text{ vor}\}$$

$$S_M^{\log}(n) = \max\{S_M^{\log}(\mathbf{x}) ; \|\mathbf{x}\|_{\log} = n\}$$

logarithmisch $s(n)$ -platzbeschränkte RAM :

$$S_M^{\log}(n) \leq s(n) \quad (n \in \mathbb{N})$$

entsprechende Definitionen für *average-case* und randomisierte Berechnungen

weitere Komplexitätsbegriffe

- *Schaltkreiskomplexität*: Algorithmen werden realisiert durch (Familien von) Schaltkreisen (= *dags* mit Operationen in den Knoten des Graphen) — relevant ist hierbei neben der Schaltkreisgrösse (=Anzahl der verwendeten Gatter \sim Laufzeit bei *sequentieller* Abarbeitung) auch die Schaltkreistiefe (Laufzeit bei *paralleler* Abarbeitung)
- *Beschreibungskomplexität*
- Komplexität von *Problemen*: welchen Aufwand erfordert jeder Algorithmus (unter einem einheitlichen Komplexitätsmodell) zur Bearbeitung eines Problems?
untere Schranken vs . obere Schranken
 - obere Schranke aus konkreten Algorithmen/Programmen
 - untere Schranken aus theoretischen ÜberlegungenEs ist meist viel schwieriger, Aussagen über untere Schranken für die Komplexität eines Problems zu gewinnen als obere Schranken!