

2. Versuch einer Definition

Was sind „kombinatorische“ Algorithmen?

Eine mögliche Antwort:

Probleme, die, könnten wir alle Fälle aufzählen, trivial wären, aber eine sehr große Anzahl von Fällen haben.

*Beispiele: Hamiltonscher Kreis (NP-vollständig),
Eulerscher Kreis (P)*

3. Ziel der Vorlesung

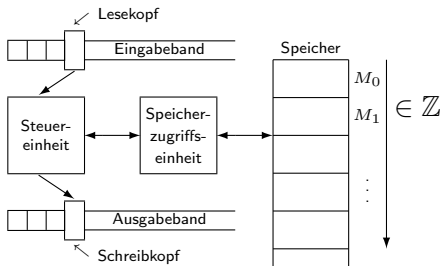
Der Zweck der Vorlesung ist das Studium fundamentaler Konzepte in der Algorithmentheorie. Es werden relevante Maschinenmodelle, grundlegende und höhere Datenstrukturen sowie der Entwurf und die Analyse sequentieller Algorithmen besprochen. Dabei wird eine Reihe verschiedener Analysemethoden (für entsprechend unterschiedliche Anforderungen) eingeführt.

Die betrachteten Problemklassen umfassen eine umfangreiche Auswahl der in der Praxis relevanten kombinatorischen Probleme, wobei die algorithmischen Ansätze sich in dieser Vorlesung jedoch praktisch auf deterministische, sequentielle, exakte Algorithmen beschränken.

Für weiterführende Konzepte und Ansätze (z.B. probabilistische, parallele, approximative Algorithmen) wird auf entsprechende Vorlesungen verwiesen.

4. Maschinenmodelle

- Turingmaschine (TM)
- Registermaschine (RAM)
- Boolesche Schaltkreise
- (Quantencomputer)
- (DNA-Computer)
- ...



Registermaschine



G. Bilardi, K. Ekanadham, P. Pattnaik:

On approximating the ideal random access machine by physical machines.

J.ACM **56**(5), Article 27, ACM, 2009

5. Komplexitätsmaße

Ein **Problem** ist formal eine Sprache $L \subseteq \Sigma^*$. $x \in \Sigma^*$ heißt eine **Probleminstance** für L , wenn wir untersuchen wollen, ob $x \in L$.

Sei M eine (Turing- oder) Registermaschine.

- M **entscheidet** L , falls für alle $x \in \Sigma^*$ M nach endlicher Zeit hält mit

$$\begin{cases} \text{Antwort „ja“, falls } x \in L \\ \text{Antwort „nein“, falls } x \notin L \end{cases}$$

- M **akzeptiert** L , falls für alle $x \in \Sigma^*$ gilt

$$\begin{cases} \text{falls } x \in L : M \text{ hält mit Antwort „ja“} \\ \text{falls } x \notin L : M \text{ hält mit Antwort „nein“ oder hält nicht.} \end{cases}$$

- Berechnung von Funktionen:
Seien Σ, Γ Alphabete. Eine TM (bzw. RAM) M berechnet eine (partielle) Funktion $f : \Sigma^* \rightarrow \Gamma^*$ gdw. für alle x im Definitionsbereich von f gilt:
bei Eingabe x hält M nach endlich vielen Schritten, und zwar mit Ausgabe $f(x)$.

Wir berechnen die **Komplexität** eines Problems in Abhängigkeit von der **Länge** der Eingabe:

Eingaben $x \in \Sigma^n$ haben Länge n .

Insbesondere bei Funktionen oder Problemen, deren Eingabe als „**als aus n Argumenten bestehend**“ interpretiert werden kann, betrachten wir oft auch die **uniforme Eingabelänge** n .

Beispiel 1

Sollen n Schlüssel $\in \Sigma^*$ (vergleichsbasiert) sortiert werden, so nehmen wir als Eingabelänge gewöhnlich n , die Anzahl der Schlüssel, und nicht ihre Gesamtlänge.

Komplexitätsressourcen:

Man betrachtet u.a.

- Rechenzeit
- Speicherplatz
- Anzahl von Vergleichen
- Anzahl von Multiplikationen
- Schaltkreisgröße
- Programmgröße
- Schachtelungstiefe von Laufschleifen
- ...

Komplexität der Ressourceneinheiten:

Wir unterscheiden

- **uniformes** Kostenmodell: Die Kosten jeder Ressourceneinheit sind 1.
- **logarithmisches** Kostenmodell: Die Kosten eines Rechenschritts sind durch die Länge der Operanden bestimmt:
 - 1 Der **Zeitbedarf** eines Rechenschritts ist gleich der größten Länge eines Operanden des Rechenschritts.
 - 2 Der **Platzbedarf** einer Speicherzelle ist gleich der größten Länge eines darin gespeicherten Wertes.

Wir unterscheiden verschiedene **Arten der Komplexität**:

- **worst-case** Komplexität:

$$C_{\text{wc}}(n) := \max\{C(x); |x| = n\}$$

- **durchschnittliche** Komplexität (average complexity):

$$C_{\text{avg}}(n) := \frac{1}{|\Sigma^n|} \sum_{|x|=n} C(x)$$

allgemeiner: Wahrscheinlichkeitsmaß μ

$$C_{\text{avg}}(n) := \sum_{x \in \Sigma^n} \mu(x) \cdot C(x)$$

Wir unterscheiden verschiedene **Arten der Komplexität**:

- **amortisierte** Komplexität:
durchschnittliche Kosten der Operationen in einer Folge von n Operationen
worst-case über alle Folgen der Länge n von Operationen
- **probabilistische** oder **randomisierte** Komplexität:
Algorithmus hat Zufallsbits zur Verfügung. Erwartete Laufzeit (über alle Zufallsfolgen) für feste Eingabe x , dann worst-case für alle $|x| = n$.

Beispiel 2

```
 $r := 2$   
for  $i := 1$  to  $n$  do  $r := r^2$  od  
co das Ergebnis ist  $2^{2^n}$  oc
```

- Zeitbedarf:
 - uniform: n Schritte
 - logarithmisch: $1 + 2 + 4 + \dots + 2^n = 2^{n+1} - 1 = \Theta(2^n)$
- Platzbedarf:
 - uniform: $\mathcal{O}(1)$
 - logarithmisch: 2^n