

# Von Berechenbarkeit zu Komplexität

- **Berechenbarkeit** in Bezug auf Entscheidbarkeit und Aufzählbarkeit betrachtet nur prinzipielle Lösbarkeit von Problemen auf Computern.
- Prinzipiell lösbare Probleme können praktisch nicht lösbar sein, weil jeder Algorithmus/DTM zur Lösung des Problems zu viel Zeit und/oder Platz benötigt.
- **Komplexitätstheorie** versucht Probleme gemäß des Zeit- und Platzbedarfs der besten DTMs zu ihrer Lösung zu klassifizieren.
- Konzentrieren uns auf Zeitbedarf und die Klassen P und NP.

# Vier Beispielprobleme

## Minimumbestimmung

**Gegeben:**  $a_1, a_2, \dots, a_n \in \mathbb{N}$ .

**Gesucht:** minimales  $a_i$ .

## Sortieren

**Gegeben:**  $a_1, a_2, \dots, a_n \in \mathbb{N}$ .

**Gesucht:** sortierte Reihenfolge  $a_{\pi(1)} \leq a_{\pi(2)} \leq \dots \leq a_{\pi(n)}$  .

# Vier Beispielprobleme

## Travelling Salesman Problem

**Gegeben:** Städte  $s_1, s_2, \dots, s_n$  und paarweise Distanzen  $d_{ij}$ ,  
 $1 \leq i, j \leq n$ .

**Gesucht:** Rundreise durch alle Städte minimaler  
Gesamtlänge.

# Vier Beispielprobleme



# Vier Beispielprobleme

## Rucksackproblem

**Gegeben:**  $n$  Gegenstände mit Gewichten  $g_1, g_2, \dots, g_n$  und Werten  $w_1, w_2, \dots, w_n$  sowie zulässiges Gesamtgewicht  $G$ .

**Gesucht:** Teilmenge  $S \subseteq \{1, \dots, n\}$  mit  $\sum_{i \in S} w_i$  ist maximal unter der Bedingung  $\sum_{i \in S} g_i \leq G$ .

# Aufwand zur Lösung der Probleme

<b>Problem</b>	<b>Laufzeit elementarer Algorithmen</b>	<b>Bemerkung</b>
<b>Minimum</b>	$O(n)$	<b>Anzahl Vergleiche</b>
<b>Sortieren</b>	$O(n \cdot \log(n))$	<b>Anzahl Vergleiche</b>
<b>TSP</b>	$O(n!)$	<b>Vergleiche &amp; arithmetische Operationen</b>
<b>Rucksack</b>	$O(2^n)$	<b>Vergleiche &amp; arithmetische Operationen</b>

# Laufzeitvergleiche

T(n)	Maximale Größe von n bei vorgegebener Rechenzeit			
	0,01 Sekunden	1 Sekunde	1 Minute	1 Stunde
n	10	1000	60000	3600000
n·log(n)	4	140	4893	204094
n <sup>2</sup>	3	31	244	1897
2 <sup>n</sup>	3	9	15	21
n!	3	6	8	9

bei 1000 Operationen pro Sekunde

# Laufzeitvergleiche

T(n)	Maximale Eingabelänge		Bemerkungen
	vor	nach	
	Erhöhung der Rechengeschwindigkeit		
n	m	10·m	
n·log(n)	m	(fast) 10·m	
n <sup>2</sup>	m	3.16·m	10 <sup>1/2</sup> ≈ 3.16
2 <sup>n</sup>	m	m+3.3	log(10) ≈ 3.3
n!	m	≈ m	

Erhöhung der Rechenleistung um den Faktor 10



# O-Notation

1.  $f = O(g) \Leftrightarrow \exists c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0: f(n) \leq c \cdot g(n)$

f wächst asymptotisch höchstens so schnell wie g

2.  $f = \Omega(g) \Leftrightarrow g = O(f)$

f wächst mindestens so schnell wie g

3.  $f = \Theta(g) \Leftrightarrow f = O(g) \text{ und } f = \Omega(g)$

f und g wachsen asymptotisch gleich schnell

4.  $f = o(g) \Leftrightarrow \forall c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0: f(n) \leq c \cdot g(n)$

f wächst asymptotisch langsamer als g

5.  $f = \omega(g) \Leftrightarrow g = o(f)$

f wächst asymptotisch schneller als g

# Laufzeit einer DTM

**Definition 3.1** DTM  $M (Q, \Sigma, \Gamma, \delta)$  halte bei jeder Eingabe.

- Für  $w \in \Sigma^*$  ist  $T_M(w)$  die Anzahl der Rechenschritte von  $M$  bei Eingabe  $w$ .
- Für  $n \in \mathbb{N}$  ist  $T_M(n) := \max \{T_M(w) \mid w \in \Sigma^{\leq n}\}$ .
- $T_M : \mathbb{N} \rightarrow \mathbb{N}$  heißt **Zeitkomplexität** oder **Laufzeit** der DTM  $M$ .
- $M$  hat Laufzeit  $\mathcal{O}(f(n))$ , wenn  $T_M(n) = \mathcal{O}(f(n))$ .

# DTM $M_1$ für $L = \{0^n 1^n \mid n \geq 1\}$

$M_1$  bei Eingabe  $w \in \{0,1\}^*$  :

1. Durchlaufe die Eingabe. Falls eine 0 nach einer 1 auftaucht, lehne ab. Sonst zurück zum Bandanfang.
2. Wiederhole den folgenden Schritt, solange noch eine 0 *und* eine 1 auf dem Band steht.
3. Durchlaufe das Band und streiche die erste 0 und die letzte 1. Gehe zum Bandanfang zurück.
4. Falls noch eine 0, aber keine 1 oder eine 1, aber keine 0 auf dem Band steht, lehne ab. Sonst akzeptiere.

# Klassen für Zeitkomplexität

**Definition 3.2** Sei  $t : \mathbb{N} \rightarrow \mathbb{N}$  eine monoton wachsende Funktion. Die Klasse  $\text{DTIME}(t(n))$  ist dann definiert als

$$\text{DTIME}(t(n)) := \left\{ L \mid \begin{array}{l} L \text{ ist eine Sprache, die von einer DTM} \\ \text{mit Laufzeit } O(t(n)) \text{ entschieden wird.} \end{array} \right\}$$

# DTM $M_2$ für $L = \{0^n 1^n \mid n \geq 1\}$

$M_2$  bei Eingabe  $w \in \{0,1\}^*$  :

1. Durchlaufe die Eingabe. Falls eine 0 nach einer 1 auftaucht, lehne ab. Sonst zurück zum Bandanfang.
2. Wiederhole die folgenden zwei Schritte, solange noch eine 0 und eine 1 auf dem Band steht.
3. Durchlaufe das Band, falls Anzahl 0 gerade und Anzahl 1 ungerade, oder umgekehrt, lehne ab.
4. Beginnend mit der ersten 0 und ersten 1, streiche jede zweite 0 und jede zweite 1.
5. Falls noch eine 0, aber keine 1 oder eine 1, aber keine 0 auf dem Band steht, lehne ab. Sonst akzeptiere.

# 2-DTM $M_3$ für $L=\{0^n1^n|n\geq 1\}$

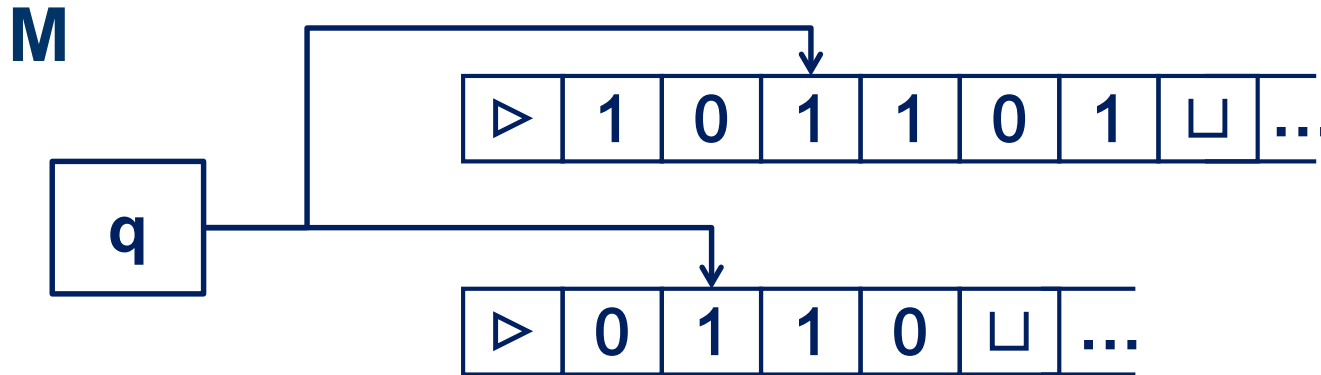
$M_3$  bei Eingabe  $w\in\{0,1\}^*$  :

1. Durchlaufe die Eingabe. Falls eine 0 nach einer 1 auftaucht, lehne ab. Sonst zurück zum Bandanfang.
2. Durchlaufe das erste Band bis zur ersten 1, kopiere dabei jede 0 auf das zweite Band.
3. Für jede 1 auf Band 1 streiche eine 0 auf Band 2. Steht zu einem Zeitpunkt keine 0 aber noch eine 1 oder keine 1 aber noch eine 0 zur Verfügung, lehne ab.
4. Ist keine 1 mehr auf Band 1 und keine 0 mehr auf Band 2, akzeptiere.

# Klassen für Zeitkomplexität

**Satz 3.3** Sei  $t: \mathbb{N} \rightarrow \mathbb{N}$  eine monoton wachsende Funktion mit  $t(n) \leq n$  für alle  $n$ . Jede Mehrband DTM mit Laufzeit  $t(n)$  kann durch eine 1-Band DTM mit Laufzeit  $O(t(n)^2)$  simuliert werden.

# Speicherung von 2 Bändern durch 1 Band



$[p_1, (q, 1, 1)]$

$p_1$ : Startzustand für Simulation von Schritt von M

