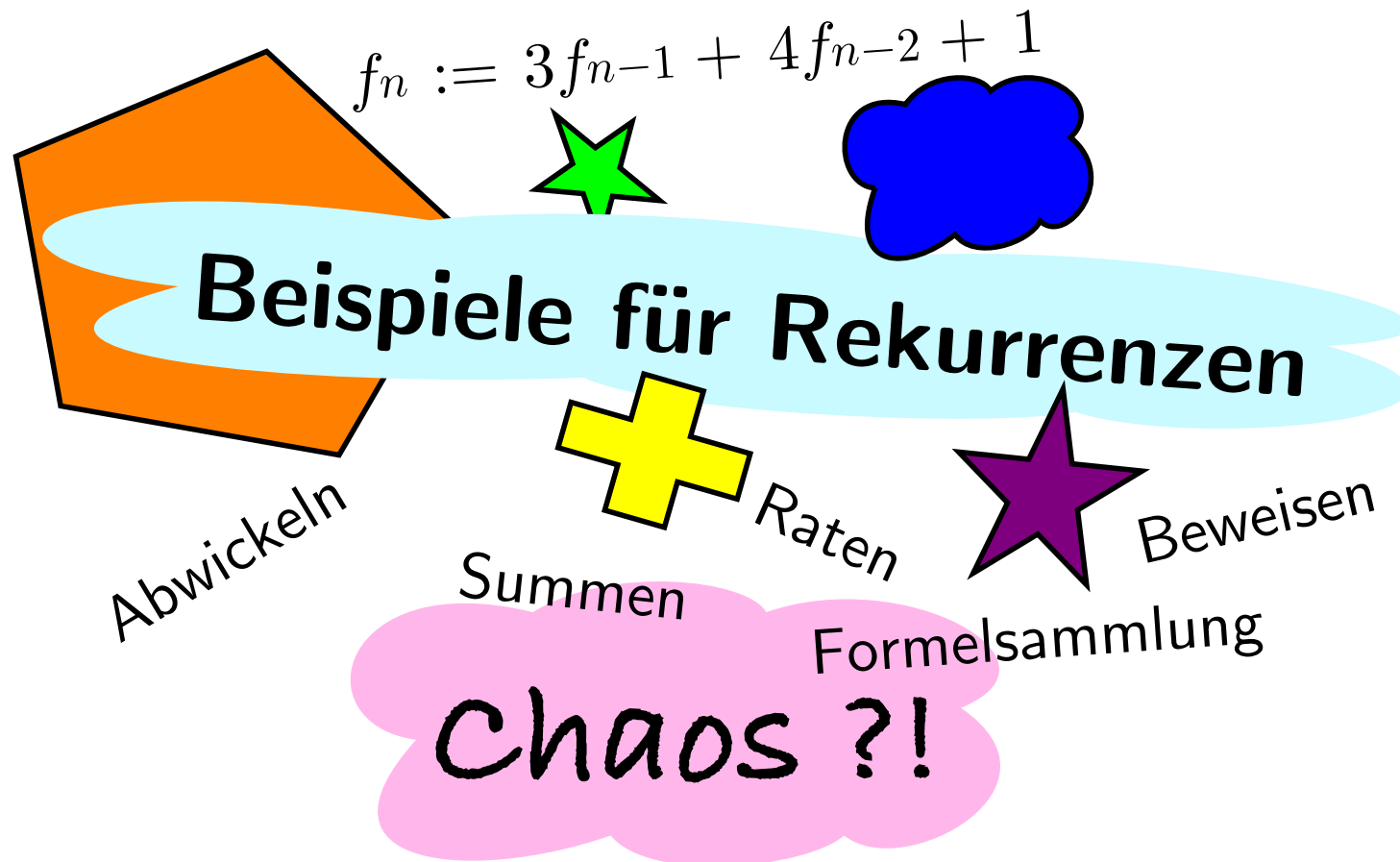


Überblick



➡ Suche nach einer Systematik



Allgemeines Verfahren

Bestimmung einer **geschlossenen Formel** für durch **lineare Rekurrenzgleichung** definierte Folge $\langle g_n \rangle_{n \geq 0}$:

1. Bilde eine **einzigste Gleichung**, in der g_n durch andere Terme der Folge $\langle g_n \rangle_{n \geq 0}$ ausgedrückt wird.
2. Multipliziere die beiden Seiten der Gleichung mit z^n und summiere für alle n . Die linke Seite wird zu $G(z) = \sum_{n=0}^{\infty} g_n z^n$. Man erhält eine **erzeugende Funktion** für $\langle g_n \rangle_{n \geq 0}$.
3. Löse diese Gleichung nach $G(z)$ auf, um eine **geschlossene Formel** für $G(z)$ zu bekommen.
4. Entwickle $G(z)$ in eine **formale Potenzreihe**, um den Koeffizienten von z^n zu bestimmen.



Beispiel

Finde eine geschlossene Formel für die **Fibonacci-Zahlen**! Fibonacci-Zahlen sind durch folgende Rekurrenzgleichung definiert:

$$f_n = \begin{cases} 0 & \text{falls } n \leq 0 \\ 1 & \text{falls } n = 1 \\ f_{n-1} + f_{n-2} & \text{falls } n > 1 \end{cases}$$

Dies ist offensichtlich **keine** geschlossene Formel!

Was kann nun getan werden, um eine solche zu finden?



Prädikate als Zahlenfunktionen

Definition: [K. Iverson, 1962]

Sei $P(n)$ ein **Prädikat**. Dann ist $[P(n)]$, folgendermaßen definiert:

$$[P(n)] = \begin{cases} 1 & \text{falls } P(n) \text{ wahr ist} \\ 0 & \text{falls } P(n) \text{ nicht wahr ist} \end{cases}$$

Beispiel:

$$f_n := \begin{cases} 2 & \text{falls } n = 0 \\ 3 & \text{falls } n = 1 \\ 2f_{n-1} + f_{n-2} & \text{sonst} \end{cases}$$

$$f_n := 2f_{n-1} + f_{n-2} + 2[n = 0] - [n = 1]$$



1. Zusammenfassen

- Bilde eine einzige Gleichung, in der g_n durch andere Terme der Folge $\langle g_n \rangle_{n \geq 0}$ ausgedrückt wird.
- Diese Gleichung soll für alle n gültig sein.
- Wir setzen fest: $g_n = 0$ falls $n < 0$.

Beispiel: Fibonacci-Zahlen

Finde eine **einzige Gleichung** für f_n , die für alle n gilt (auch für $n < 0$!).

$$f_n = f_{n-1} + f_{n-2} + [n = 1]$$



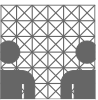
2. Erzeugende Funktion

- Multipliziere Gleichung mit z^n und summiere für n .
- Erzeugende Funktion für $\langle g_n \rangle_{n \geq 0}$: $G(z) = \sum_{n=0}^{\infty} g_n z^n$
- Rechte Seite manipulieren $\Rightarrow$ Ausdruck von $G(z)$.

Beispiel: Fibonacci-Zahlen

Transformiere $f_n = f_{n-1} + f_{n-2} + [n = 1]$:

$$\begin{aligned} F(z) &= \sum_n f_n z^n \\ &= \sum_n (f_{n-1} + f_{n-2} + [n = 1]) z^n \\ &= \sum_n f_{n-1} z^n + \sum_n f_{n-2} z^n + \sum_n [n = 1] z^n \\ &= zF(z) + z^2 F(z) + z \end{aligned}$$



3. Auflösen der Gleichung nach $G(z)$

- Löse diese Gleichung nach $G(z)$ auf, um eine geschlossene Formel für $G(z)$ zu bekommen.

Beispiel: Fibonacci-Zahlen

Löse die Gleichung nach $F(z)$ auf:

$$F(z) = zF(z) + z^2F(z) + z$$

$$\iff F(z) - zF(z) - z^2F(z) = z$$

$$\iff F(z) \cdot (1 - z - z^2) = z$$

$$\iff F(z) = \frac{z}{1 - z - z^2}$$



4. Formale Potenzreihe

- **Entwickle $G(z)$ in eine formale Potenzreihe**, um den Koeffizienten von z^n zu bestimmen.
(Dieser Schritt ist im allgemeinen der schwierigste!)

Beispiel: Fibonacci-Zahlen

- Versuch $\frac{z}{1-z-z^2}$ als Summe zweier Brüche in der Form $z \left(\frac{a}{1-\alpha z} + \frac{b}{1-\beta z} \right)$ zu schreiben.
- Dies bedeutet, den Nenner $(1 - z - z^2)$ in der Form $(1 - \alpha z)(1 - \beta z)$ zu schreiben und danach die Größen a und b zu suchen.

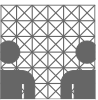


4. Schritt (Forts.)

- Wenn dies geschafft ist, so wissen wir, daß $\frac{a}{1-\alpha z}$ erzeugende Funktion für $a \sum_{n \geq 0} (\alpha z)^n$ ist, also

$$\begin{aligned} F(z) &= z \left(a \sum_{n \geq 0} (\alpha z)^n + b \sum_{n \geq 0} (\beta z)^n \right) \\ &= \sum_{n \geq 0} (a\alpha^{n-1} + b\beta^{n-1}) z^n \end{aligned}$$

- Mit $[z^n]F(z) = f_n = a\alpha^{n-1} + b\beta^{n-1}$ erhalten wir also die n -te Fibonacci-Zahl in **geschlossener Form!**



Faktorisierung des Polynoms

- Die **Faktorisierung** von $(1 - z - z^2)$ in die Form $(1 - \alpha z)(1 - \beta z)$ gelingt vielleicht noch direkt.
- Allgemeine Methode für beliebige Polynome:

Definition: Sei $q(z)$ ein Polynom über $\mathbb{C}$ vom Grad d , so heißt $q^R(z) := z^d \cdot q(\frac{1}{z})$ das **reflektierte Polynom** von q .

Satz: Sei $q(z) = 1 + q_1z + q_2z^2 + q_3z^3 + \dots + q_dz^d$ ein Polynom über $\mathbb{C}$ vom Grad d und $\alpha_1, \alpha_2, \dots, \alpha_d$ die Nullstellen des reflektierten Polynoms $q^R(z)$, dann gilt:

$$q(z) = (1 - \alpha_1 z)(1 - \alpha_2 z) \cdot \dots \cdot (1 - \alpha_d z)$$



Fast am Ziel (Fibonacci)

- Bei der noch zu analysierenden Fibonacci-Zerlegung sind also die Nullstellen von $(1 - z - z^2)^R = z^2 - z - 1 = (z - \alpha)(z - \beta)$ zu finden. Es ergeben sich die Nullstellen

$$\alpha = \frac{1 + \sqrt{5}}{2} \quad \text{und} \quad \beta = \frac{1 - \sqrt{5}}{2}$$

die wir auch schon vom **goldenen Schnitt** kennen.

- Es sind jetzt noch a und b in

$$\frac{1}{(1 - \alpha z)(1 - \beta z)} = \frac{a}{1 - \alpha z} + \frac{b}{1 - \beta z}$$

zu bestimmen.



Noch ein wenig rechnen ...

- Wir bringen die Summe auf den Hauptnenner $(1 - \alpha z)(1 - \beta z)$ und es folgt

$$(a+b) - (a\beta + b\alpha)z = 1 = (a+b) - \frac{a(1 - \sqrt{5}) + b(1 + \sqrt{5})}{2}z.$$

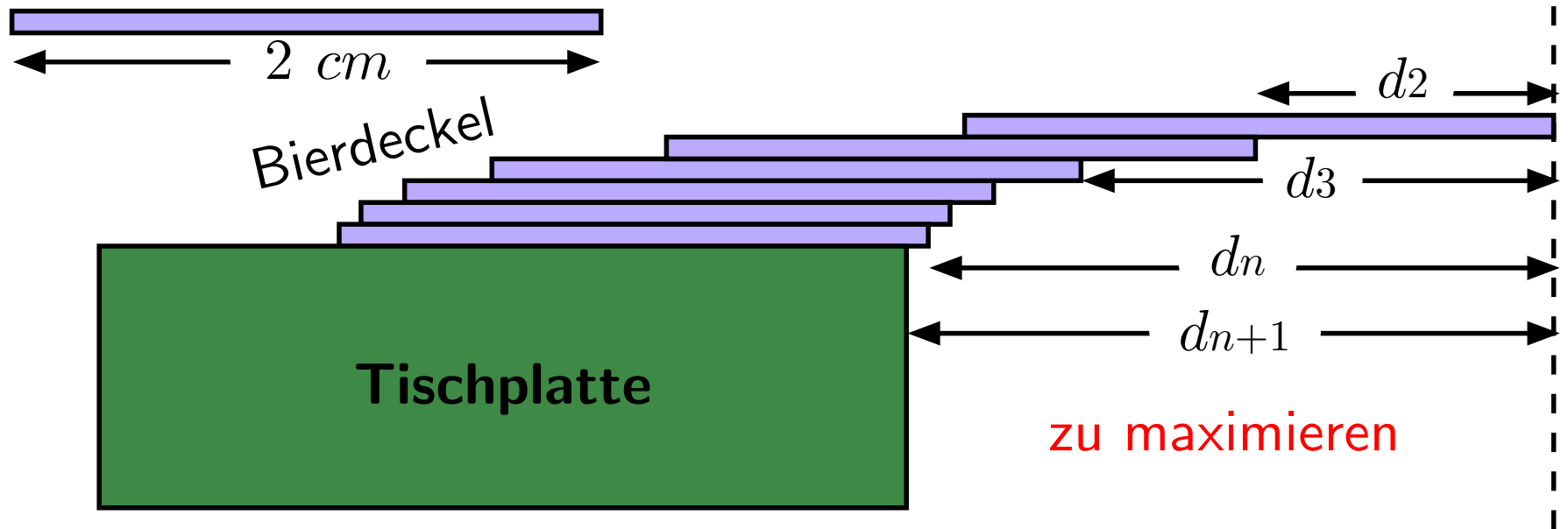
- Also $a + b = 1$ sowie $a\beta + b\alpha = 0$ (*warum?!*), woraus $a = \frac{\alpha}{\sqrt{5}} = \frac{1+\sqrt{5}}{2\sqrt{5}}$ und $b = -\frac{\beta}{\sqrt{5}} = \frac{\sqrt{5}-1}{2\sqrt{5}}$.
- Mit $f_n = a\alpha^{n-1} + b\beta^{n-1}$ finden wir letztendlich

$$f_n = \frac{1}{\sqrt{5}} \left(\left(\frac{1 + \sqrt{5}}{2} \right)^n - \left(\frac{1 - \sqrt{5}}{2} \right)^n \right).$$



Weitere Beispiele ...

... sind im Skript zu finden!



Offensichtlich gilt

• $d_1 = 0$

• $d_2 = 1 = \frac{d_1+1}{1}$

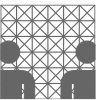
• $d_3 = \frac{(d_1+1)+(d_2+1)}{2}$



• ... und allgemein

$$\begin{aligned}d_{k+1} &= \frac{1}{k} \cdot \sum_{i=1}^k (d_i + 1) \\&= \frac{1}{k} \left(k + \sum_{i=1}^k d_i \right) \\&= 1 + \frac{1}{k} \cdot \sum_{i=1}^k d_i\end{aligned}$$

• Somit ist $k \cdot d_{k+1} = k + \sum_{i=1}^k d_i$.



Bierdeckel (Forts.)

- Einsetzen von $k - 1$ statt k in $kd_{k+1} = k + \sum_{i=1}^k d_i$ ergibt:

$$(k - 1) d_k = k - 1 + \sum_{i=1}^{k-1} d_i$$

- Durch Subtraktion beider Gleichungen:

$$kd_{k+1} - (k - 1) d_k = 1 + d_k$$

- Umgerechnet die einfache Rekurrenz

$$d_{k+1} = d_k + \frac{1}{k}$$

mit der Lösung $d_{k+1} = \sum_{i=1}^k \frac{1}{i}$.



Problem

- **ABER:** Kann man zu jeder Summenformel auch eine geschlossenen Formel finden?
- **Antwort:** Nein, z.B. nicht für das Bierdeckelproblem!
- Bei der Analyse von Algorithmen trifft man oft auf die **harmonischen Zahlen:** (H_n : ist n -te Harmonische Zahl)

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \sum_{i=1}^n \frac{1}{i}$$

- H_n kann für jedes n exakt berechnet werden!
- Das reicht jedoch nicht aus, denn es gibt keine geschlossene Formel für H_n . Deshalb wird eine gute **Approximation** für H_n benötigt.



Abschätzung durch Integrale

- **Summenformeln** können oft nach oben und unten durch **Integrale** abgeschätzt werden.

- **Satz** Für $G(n) := \sum_{i=1}^n g(i)$ mit $g(i) \leq g(i+1)$ gilt:

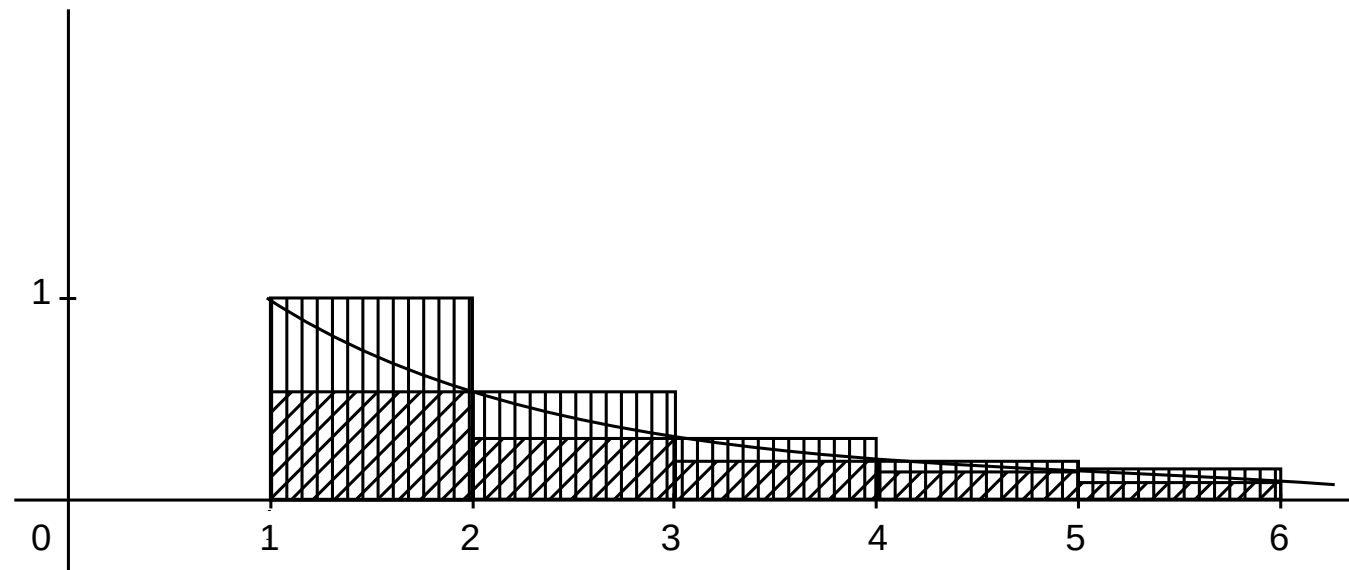
$$\int_0^n g(x) dx \leq G(n) \leq \int_1^{n+1} g(x) dx$$

- Ist eine Funktion $g(x)$ **monoton fallend**, aber gilt stets $g(x) > 0$, so läßt sich das Verfahren der Abschätzung von $G(n) := \sum_{i=1}^n g(i)$ mit Hilfe von Integralen immer noch gut anwenden.



Harmonische Zahlen

- $H_n = \sum_{i=1}^n \frac{1}{i}$, so daß $g(x) = \frac{1}{x}$ ist monoton fallend.
- Wegen $\int \frac{1}{x} dx = \ln(x)$ kann mit dieser Methode eine Abschätzung von H_n vorgenommen werden.
- Für $g(x) = \frac{1}{x}$ ergibt sich:



- Die Rechtecke stellen die Flächen $\frac{1}{x}$ dar.



Approximation von H_n

- Es folgt nun $\int_1^n \frac{1}{x} dx < H_n < 1 + \int_1^n \frac{1}{x} dx$ und somit

$$\ln n < H_n < \ln n + 1 \quad \text{falls } n > 1$$

und diese **Approximation** ist schon nicht schlecht.

- Viel besser aber ist die Approximation

$$H_n = \ln n + 0.5772156649 + \frac{1}{2n} - \frac{1}{12n^2} + \underbrace{\Theta(n^{-4})}_{\text{„wächst wie } n^{-4}\text{“}}$$



Beispiel: schriftliches Multiplizieren

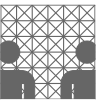
- Wieviel Zeit braucht ein Mensch (ein Computer), um zwei n -stellige ganze Zahlen zu multiplizieren, wenn er den gewöhnlichen Schulalgorithmus benutzt?

$$\begin{array}{ccccccccccc} & & & & a_1 & a_2 & \dots & a_n & \times & b_1 & b_2 & \dots & b_n \\ & & & & \bullet & \bullet & \dots & \bullet & & & & & \\ & & & & & & & & & & & & \\ & & & \bullet & \bullet & \dots & \bullet & & & & & & \\ & & & & & & & & & & & & \\ & & \vdots & & \vdots & & & & & & & & \\ & & & & & & & & & & & & \\ \bullet & \bullet & \bullet & & & \dots & \bullet & \bullet & & & & & \end{array}$$

n Zeilen

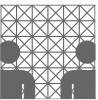
$2n$ Stellen

- Die exakte Antwort ist wenig aussagekräftig, aber es gilt: Für k -mal größere Zahlen wird k^2 -mal mehr Zeit benötigt.



Optimalitätsfrage

- Gibt es einen schnelleren Algorithmus?
- x, y : zwei n -stellige ganze Zahlen (n gerade) in Binärdarstellung.
- $x = x_1 2^{\frac{n}{2}} + x_2$ und $y = y_1 2^{\frac{n}{2}} + y_2$, wobei x_1, x_2, y_1, y_2 $\frac{n}{2}$ -stellige Binärdarstellungen ganzer Zahlen sind
- Dann ist $xy = x_1 y_1 2^n + (x_1 y_2 + x_2 y_1) 2^{\frac{n}{2}} + x_2 y_2$
- **vier Multiplikationen** der $\frac{n}{2}$ -stelligen Zahlen, **drei Additionen** und **zwei Shifts** um n und $\frac{n}{2}$ Positionen.
- Mit dem Schulalgorithmus: Komplexität in $O(n^2)$.



Multiplikation anders

- Andere Methode, die oben benötigten drei Werte x_1y_1 , $x_1y_2 + x_2y_1$ und x_2y_2 zu berechnen:

$$x_1y_1, \quad x_2y_2$$

$$z_1 = (x_1 + x_2)(y_1 + y_2) = x_1y_1 + x_1y_2 + x_2y_1 + x_2y_2$$

$$z_1 - x_1y_1 - x_2y_2 = x_1y_2 + x_2y_1$$

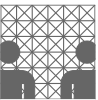
- Reduktion des Problems der Multiplikation n -stelliger Zahlen auf das Problem von **drei Multiplikationen** $\frac{n}{2}$ -stelliger Zahlen in $O(n)$ Zeit.
- Analyse der Zeitkomplexität von *divide-and-conquer*: Zeitkomplexität $O(n^{1.58})$ ist!

Das ist besser als die Schulmethode!!!



Strukturelle Komplexitätstheorie

- systematisches Studium der Komplexität von Problemen (nicht konkreten Algorithmen!)
 - Probleme als Mengen
 - Problemlösung entspricht Akzeptierung von Wörtern
 - Modell ist die Turing-Maschine (DTM vs. NTM)
 - Zeit- und Platzbedarfe werden abgeschätzt
 - untere und obere Schranken
 - in Abhängigkeit der Eingabegröße
- Eingabelänge zählt nicht zum Platzbedarf!



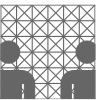
Probleme als Mengen

- Berechenbarkeit einer beliebigen Funktion ist manchmal umständlich zu zeigen!
- Idee: Akzeptierung verwenden, um die Komplexität von Problemen zu untersuchen.
 - intern wird etwas berechnet
 - das Ergebnis muß nicht (standardisiert) ausgegeben werden
 - auch NTM einsetzbar
 - damit ist ein Vergleich zwischen deterministischen und nichtdeterministischen Problemlösungen direkt möglich
- *Wie codiere ich ein Problem als Menge?*



Probleme als Mengen (2)

- Probleme bestehen aus:
 - Fragestellung
 - Eingaben/Voraussetzungen
 - Lösung
- Problemstellung & Lösung(svorschlag) stellen die **Probleminstanz** dar.
- Bestandteile werden als Wörter über einem endlichen Alphabet codiert.
- Genau die Probleminstanzen, die eine Lösung darstellen sollen akzeptiert werden.



Hamilton-Kreis

Gegeben: Ein ungerichteter Graph $G = (V, E)$

Gesucht: Gibt es einen geschlossenen Kreis in G , bei dem jeder Knoten genau einmal durchlaufen wird, d.h. existiert eine Folge von Knoten $v_1, v_2, \dots, v_n$ mit $n = |V|$, $V = \{v_1, v_2, \dots, v_n\}$ und $\{v_i, v_{i+1}\} \in E$ und $\{v_n, v_1\} \in E$ für alle $i \in \mathbb{N}$ mit $1 \leq i < n$?

Antwort: JA / NEIN



Hamilton-Kreis als Menge



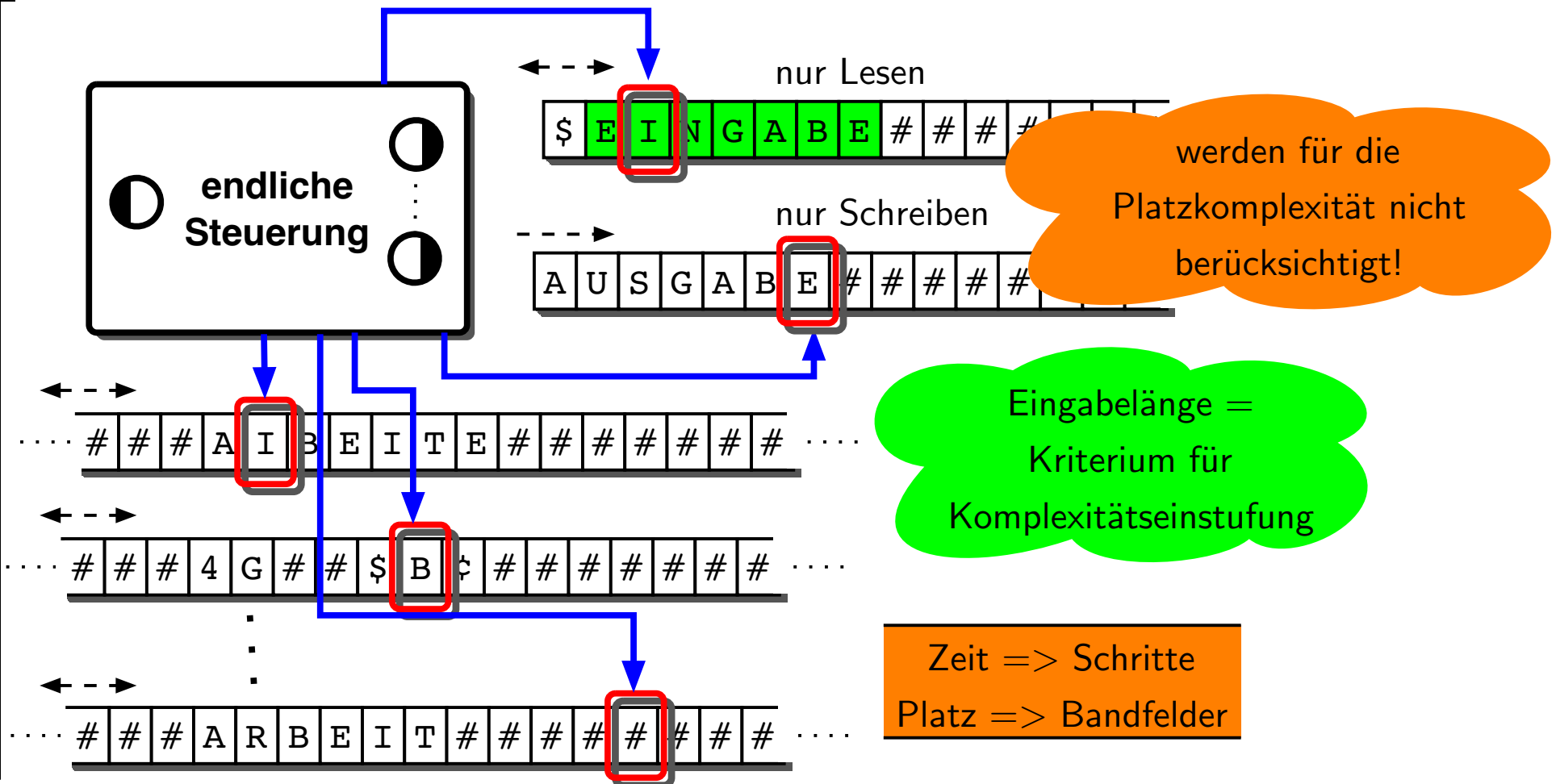
$$G = (\{1, 2, 3, 4\}, \{(1, 2), (2, 3), (3, 4)\})$$

- Mögliche Darstellung: $1, 2, 3, 4(1, 2)(2, 3)(3, 4)$
 - Problem: kein **endliches** Alphabet
- Alternative: $1, 11, 111, 1111(1, 11)(11, 111)(111, 1111)$
- Darstellung mit den Zeilen der **Adjazenzmatrix**:
 $0100\#1010\#0101\#0010$
- Länge ist stets $|V|^2 + |V| - 1$ Zeichen. (obiges Beispiel hat die Länge 19.)



k -Band-offline-Turing-Maschine

- Wir verwenden künftig generell k -Band-offline-TM



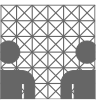
Begriffe: Zeit und Platz

Definition: Für eine konkrete Rechnung *benötigt* eine NTM soviel

- **Zeit**, wie Konfigurationenswechsel auftreten.
- **Platz**, wie maximal Felder auf den Arbeitsbändern besucht werden.

Definition Eine NTM A verarbeitet ein Wort w mit der

- **Zeitbeschränkung** t genau dann, wenn die *kürzeste* Rechnung für w höchstens $\lceil t \rceil$ Zeit benötigt.
- **Platzbeschränkung** $s \in \mathbb{R}$, wenn die Rechnung mit dem geringsten Platzverbrauch für w höchstens $\lceil s \rceil$ Platz benötigt.

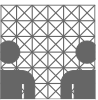


Definition von Komplexitätsklassen

- Zusammenfassen ähnlicher Eingaben
 - über die Art des Problems (Semantik) ist in der Regel nur sehr schwer eine Aussage zu machen
 - Kategorisierung nach **Länge des Eingabewortes**.

Definition: Seien $s : \mathbb{R} \rightarrow \mathbb{R}$, $t : \mathbb{R} \rightarrow \mathbb{R}$. Eine NTM A ist

- **t -zeitbeschränkt** gdw. sie für **jedes akzeptierte** Eingabewort der Länge n mit der Zeitbeschränkung $t(n)$ arbeitet.
- **s -platzbeschränkt** genau dann, wenn sie für **jedes akzeptierte** Eingabewort der Länge n mit der Platzbeschränkung $s(n)$ arbeitet.



Beispiele von Komplexitätsklassen

Definition Für $s, t : \mathbb{N} \rightarrow \mathbb{R}$ mit $t(n) \geq n + 1$ und $s(n) \geq 1$, bezeichne:

$$\mathcal{D}Time(t) := \{L \mid L = L(A) \text{ für eine } t\text{-zeitbeschr. DTM } A\}$$

$$\mathcal{N}Time(t) := \{L \mid L = L(A) \text{ für eine } t\text{-zeitbeschr. NTM } A\}$$

$$\mathcal{D}Space(s) := \{L \mid L = L(A) \text{ für eine } s\text{-platzbeschr. DTM } A\}$$

$$\mathcal{N}Space(s) := \{L \mid L = L(A) \text{ für eine } s\text{-platzbeschr. NTM } A\}$$

$$\mathcal{P} := \{L \mid \text{Es gibt ein Polynom } p : \mathbb{N} \rightarrow \mathbb{R} \text{ und eine } p\text{-zeitbeschränkte DTM } A \text{ mit } L = L(A)\}$$

$$\mathcal{NP} := \{L \mid \text{Es gibt ein Polynom } p : \mathbb{N} \rightarrow \mathbb{R} \text{ und eine } p\text{-zeitbeschränkte NTM } A \text{ mit } L = L(A)\}$$

