



Theoretische Grundlagen der Informatik

Vorlesung am 17.11.2021

Torsten Ueckerdt | 17. November 2021

Letzte Vorlesung

Bearbeitet eine TM \mathcal{M} eine Eingabe w , so gibt es drei Möglichkeiten:

1. \mathcal{M} "läuft" in einen Zustand in F .

$\rightsquigarrow \mathcal{M}$ **akzeptiert** w

2. \mathcal{M} "läuft" in einen Übergang $\delta(q, a) = (q, a, N)$.

$\rightsquigarrow \mathcal{M}$ **lehnt** w **ab**

3. \mathcal{M} "läuft" unendlich lange.

$\rightsquigarrow \mathcal{M}$ **stoppt nicht**

Für eine Turing-Maschine \mathcal{M} und Sprache L definieren wir:

\mathcal{M} hält	1. oder 2.	
\mathcal{M} akzeptiert L , (L semi-entscheidbar)	$\forall w \in L$: 1.	$\forall w \notin L$: 2. oder 3.
\mathcal{M} entscheidet L , (L entscheidbar)	$\forall w \in L$: 1.	$\forall w \notin L$: 2.

Das Halteproblem

Definition.

Das **Halteproblem** ist definiert als folgende Sprache

$$\mathcal{H} := \{w\#v : T_w \text{ hält auf der Eingabe } v\}.$$

Beachte: Wir benutzen ein Trennzeichen $\# \notin \Sigma$.

Satz.

\mathcal{H} ist nicht entscheidbar.

Interpretation:

- Das Problem, ob eine Turing-Maschine auf einer Eingabe w stoppt, ist nicht entscheidbar.

Das Halteproblem

Satz.

$\mathcal{H} = \{w \# v : T_w \text{ hält auf der Eingabe } v\}$ ist nicht entscheidbar.

Beweis:

- Angenommen es existiert eine stets haltende Turing-Maschine, die \mathcal{H} entscheidet.
- Wir konstruieren daraus eine stets haltende Turing-Maschine, die L_d^c entscheidet, mit Widerspruch zum Korollar letzte Vorlesung.

Sei w eine Eingabe, für die wir entscheiden wollen, ob $w \in L_d^c$.

Wir können wie folgt vorgehen:

- Berechne das i , so dass $w = w_i$ ist.
- Betrachte die durch w_i kodierte Turing-Maschine \mathcal{M}_i .
- Wende die Turing-Maschine für \mathcal{H} auf $\langle \mathcal{M}_i \rangle \# w_i$ an.

Erinnerung:

$$w = w_i \in L_d^c$$

$$\Leftrightarrow \mathcal{M}_i \text{ akzeptiert } w_i$$

$$\text{mit } w_i = \langle \mathcal{M}_i \rangle$$

Das Halteproblem

Satz.

$\mathcal{H} = \{w\#v : T_w \text{ hält auf der Eingabe } v\}$ ist nicht entscheidbar.

Sei w eine Eingabe, für die wir entscheiden wollen, ob $w \in L_d^c$.

Wir können wie folgt vorgehen:

- Berechne das i , so dass $w = w_i$ ist.
- Betrachte die durch w_i kodierte Turing-Maschine \mathcal{M}_i .
- Wende die Turing-Maschine für \mathcal{H} auf $\langle \mathcal{M}_i \rangle \# w_i$ an.

Wir machen folgende Fallunterscheidung:

- Falls $\langle \mathcal{M}_i \rangle \# w_i$ nicht akzeptiert wird, dann hält \mathcal{M}_i nicht auf w_i .
- Also ist $w_i \in L_d$ und damit $w_i \notin L_d^c$.
- Falls $\langle \mathcal{M}_i \rangle \# w_i$ akzeptiert wird, dann hält \mathcal{M}_i auf w_i .
- Dann können wir auf der universellen Turing-Maschine die Berechnung von \mathcal{M}_i auf w_i simulieren und so entscheiden, ob \mathcal{M}_i die Eingabe w_i akzeptiert, also ob $w_i \in L_d^c$.

Erinnerung:

$$w = w_i \in L_d^c$$

$$\Leftrightarrow \mathcal{M}_i \text{ akzeptiert } w_i$$

$$\text{mit } w_i = \langle \mathcal{M}_i \rangle$$

Die Universelle Sprache

- Die **universelle Sprache** L_U über $\{0, 1\}$ ist definiert durch $L_U := \{w\#v : v \in L(T_w)\}$.
- L_U ist also die Menge aller Wörter $w\#v$ für die T_w bei der Eingabe v hält und v akzeptiert.

Satz.

Die universelle Sprache L_U ist nicht entscheidbar.

Die Universelle Sprache

- Die **universelle Sprache** L_U über $\{0, 1\}$ ist definiert durch $L_U := \{w\#v : v \in L(T_w)\}$.
- L_U ist also die Menge aller Wörter $w\#v$ für die T_w bei der Eingabe v hält und v akzeptiert.

Satz.

Die universelle Sprache L_U ist nicht entscheidbar.

Beweis:

- Wir zeigen, dass L_U eine Verallgemeinerung von L_D^C ist.
- Wir nehmen an, dass es eine TM gibt, die L_U entscheidet.
- Dann zeigen wir, dass wir damit auch L_D^C entscheiden können:
 - Berechne das i , für das $w = w_i$.
 - Betrachte die durch w_i kodierte Turing-Maschine \mathcal{M}_i .
 - Wende die Turing-Maschine für L_U auf $\langle \mathcal{M}_i \rangle \# w_i$ an.

Wäre L_U entscheidbar, so auch L_D^C im Widerspruch zum Korollar letzte Vorlesung.

Die Universelle Sprache

Satz.

Die universelle Sprache $L_U := \{w\#v : v \in L(T_w)\}$ ist semi-entscheidbar.

Die Universelle Sprache

Satz.

Die universelle Sprache $L_U := \{w\#v : v \in L(T_w)\}$ ist semi-entscheidbar.

Beweis:

Wir benutzen die universelle Turing-Maschine, mit der Eingabe $w\#v$:

- Falls T_w die Eingabe v akzeptiert, geschieht dies nach endlich vielen Schritten und die universelle Turing-Maschine akzeptiert $w\#v$.
- Falls T_w die Eingabe v nicht akzeptiert, wird $w\#v$ von der universellen Turing-Maschine ebenfalls nicht akzeptiert. Dies ist unabhängig davon, ob die Simulation stoppt oder nicht.

Die Universelle Sprache

Satz.

Die universelle Sprache $L_U := \{w\#v : v \in L(T_w)\}$ ist semi-entscheidbar.

Beweis:

Wir benutzen die universelle Turing-Maschine, mit der Eingabe $w\#v$:

- Falls T_w die Eingabe v akzeptiert, geschieht dies nach endlich vielen Schritten und die universelle Turing-Maschine akzeptiert $w\#v$.
- Falls T_w die Eingabe v nicht akzeptiert, wird $w\#v$ von der universellen Turing-Maschine ebenfalls nicht akzeptiert. Dies ist unabhängig davon, ob die Simulation stoppt oder nicht.

Bemerkung: Die Begriffe entscheidbar und semi-entscheidbar unterscheiden sich tatsächlich.

Satz von Rice – Motivation

- Wir haben bisher gezeigt, dass wir kein Programm schreiben können, das für ein Turing-Maschinen-Programm $\langle \mathcal{M} \rangle$ und eine Eingabe w entscheidet, ob \mathcal{M} auf der Eingabe w hält.
- Wir werden im Folgenden sehen, dass wir aus einem Programm im Allgemeinen keine nicht-trivialen Eigenschaften der von dem Programm realisierten Funktion ableiten können.

Satz von Rice

Satz von Rice.

Sei R die Menge der von Turing-Maschinen berechenbaren Funktionen und S eine nicht-triviale Teilmenge von R ($\emptyset \neq S \neq R$). Dann ist die Sprache

$$L(S) := \{ \langle M \rangle : M \text{ berechnet eine Funktion aus } S \}$$

nicht entscheidbar.

Satz von Rice

Satz von Rice.

Sei R die Menge der von Turing-Maschinen berechenbaren Funktionen und S eine nicht-triviale Teilmenge von R ($\emptyset \neq S \neq R$). Dann ist die Sprache

$$L(S) := \{\langle M \rangle : M \text{ berechnet eine Funktion aus } S\}$$

nicht entscheidbar.

Beweisskizze:

- Zeige: $\mathcal{H}_\varepsilon := \{\langle M \rangle : M \text{ hält auf der Eingabe } \varepsilon\}$ ist unentscheidbar
- Zeige: $\mathcal{H}_\varepsilon^c$ ist unentscheidbar
- Führe den Widerspruchsbeweis für die Unentscheidbarkeit von $L(S)$:
- Konstruiere TM für $\mathcal{H}_\varepsilon^c$ unter Benutzung von TM \mathcal{M}' für $L(S)$

Bemerkungen zum Satz von Rice

Der Satz von Rice hat weitreichende Konsequenzen:

Es ist für Programme nicht entscheidbar, ob die durch sie definierte Sprache endlich, leer, unendlich oder ganz Σ^* ist.

Wir haben hier nur die Unentscheidbarkeit von L_d direkt bewiesen.

Die anderen Beweise folgten dem folgenden Schema:

Um zu zeigen, dass ein Problem A unentscheidbar ist, zeigen wir, wie man mit einem Entscheidungsverfahren für A ein bekanntermaßen unentscheidbares Problem B entscheiden kann. Dies liefert den gewünschten Widerspruch.

Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(1, 111), (10111, 10), (10, 0)\}$ hat die Lösung $(2, 1, 1, 3)$, denn:

$$x_2 x_1 x_1 x_3 = 101111110 = y_2 y_1 y_1 y_3$$

$$K = \begin{array}{|c|c|c|} \hline 1 & 10111 & 10 \\ \hline \dots & \dots & \dots \\ \hline 111 & 10 & 0 \\ \hline \end{array} \begin{array}{l} (x_1, y_1) \\ (x_2, y_2) \\ (x_3, y_3) \end{array}$$

Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(1, 111), (10111, 10), (10, 0)\}$ hat die Lösung $(2, 1, 1, 3)$, denn:

$$x_2 x_1 x_1 x_3 = 101111110 = y_2 y_1 y_1 y_3$$

$$K = \begin{array}{|c|c|c|} \hline \begin{array}{c} 1 \\ \dots \\ 111 \end{array} & \begin{array}{c} 10111 \\ \dots \\ 10 \end{array} & \begin{array}{c} 10 \\ \dots \\ 0 \end{array} \\ \hline (x_1, y_1) & (x_2, y_2) & (x_3, y_3) \\ \hline \end{array} \quad \begin{array}{c} \begin{array}{c} 10111 \\ \dots \\ 10 \end{array} \\ 2 \end{array}$$

Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

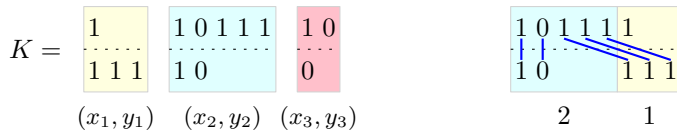
$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(1, 111), (10111, 10), (10, 0)\}$ hat die Lösung $(2, 1, 1, 3)$, denn:

$$x_2 x_1 x_1 x_3 = 101111110 = y_2 y_1 y_1 y_3$$



Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

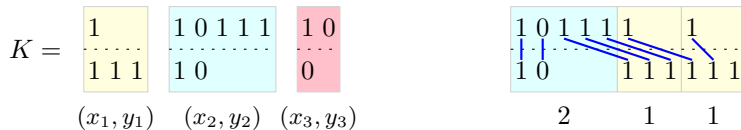
$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(1, 111), (10111, 10), (10, 0)\}$ hat die Lösung $(2, 1, 1, 3)$, denn:

$$x_2 x_1 x_1 x_3 = 101111110 = y_2 y_1 y_1 y_3$$



Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

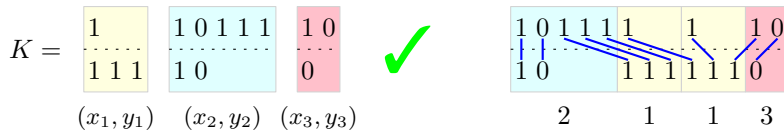
$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(1, 111), (10111, 10), (10, 0)\}$ hat die Lösung $(2, 1, 1, 3)$, denn:

$$x_2 x_1 x_1 x_3 = 101111110 = y_2 y_1 y_1 y_3$$



Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(10, 101), (011, 11), (101, 011)\}$ hat keine Lösung.

$$K = \begin{array}{ccc} \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 1 & 0 & 1 \\ \hline \end{array} & \begin{array}{|c|c|c|} \hline 0 & 1 & 1 \\ \hline 1 & 1 & \\ \hline \end{array} & \begin{array}{|c|c|c|} \hline 1 & 0 & 1 \\ \hline 0 & 1 & 1 \\ \hline \end{array} \\ (x_1, y_1) & (x_2, y_2) & (x_3, y_3) \end{array}$$

Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(10, 101), (011, 11), (101, 011)\}$ hat keine Lösung.

$$K = \begin{array}{ccc} \begin{array}{|c|c|} \hline 1 & 0 \\ \hline \dots & \dots \\ \hline 1 & 0 & 1 \\ \hline \end{array} & \begin{array}{|c|c|c|} \hline 0 & 1 & 1 \\ \hline \dots & \dots & \dots \\ \hline 1 & 1 & \\ \hline \end{array} & \begin{array}{|c|c|c|} \hline 1 & 0 & 1 \\ \hline \dots & \dots & \dots \\ \hline 0 & 1 & 1 \\ \hline \end{array} \\ (x_1, y_1) & (x_2, y_2) & (x_3, y_3) \end{array}$$

$$\begin{array}{|c|c|} \hline 1 & 0 \\ \hline \dots & \dots \\ \hline 1 & 0 & 1 \\ \hline \end{array}$$

$$1$$

Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

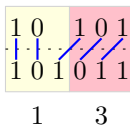
$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(10, 101), (011, 11), (101, 011)\}$ hat keine Lösung.

$$K = \begin{array}{ccc} \begin{array}{|c|c|} \hline 1 & 0 \\ \hline \dots & \dots \\ \hline 1 & 0 & 1 \\ \hline \end{array} & \begin{array}{|c|c|c|} \hline 0 & 1 & 1 \\ \hline \dots & \dots & \dots \\ \hline 1 & 1 & \\ \hline \end{array} & \begin{array}{|c|c|c|} \hline 1 & 0 & 1 \\ \hline \dots & \dots & \dots \\ \hline 0 & 1 & 1 \\ \hline \end{array} \\ (x_1, y_1) & (x_2, y_2) & (x_3, y_3) \end{array}$$



Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

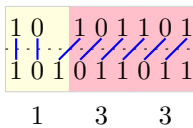
$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(10, 101), (011, 11), (101, 011)\}$ hat keine Lösung.

$$K = \begin{array}{ccc} \begin{array}{|c|c|} \hline 1 & 0 \\ \hline \dots & \dots \\ \hline 1 & 0 & 1 \\ \hline \end{array} & \begin{array}{|c|c|c|} \hline 0 & 1 & 1 \\ \hline \dots & \dots & \dots \\ \hline 1 & 1 & \\ \hline \end{array} & \begin{array}{|c|c|c|} \hline 1 & 0 & 1 \\ \hline \dots & \dots & \dots \\ \hline 0 & 1 & 1 \\ \hline \end{array} \\ (x_1, y_1) & (x_2, y_2) & (x_3, y_3) \end{array}$$



Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(10, 101), (011, 11), (101, 011)\}$ hat keine Lösung.

$$K = \begin{array}{ccc} \begin{array}{|c|} \hline 10 \\ \hline \dots \\ \hline 101 \\ \hline \end{array} & \begin{array}{|c|} \hline 011 \\ \hline \dots \\ \hline 11 \\ \hline \end{array} & \begin{array}{|c|} \hline 101 \\ \hline \dots \\ \hline 011 \\ \hline \end{array} \\ (x_1, y_1) & (x_2, y_2) & (x_3, y_3) \end{array} \quad \times$$

$$\begin{array}{cccc} \begin{array}{|c|} \hline 10 \\ \hline \dots \\ \hline 101 \\ \hline \end{array} & \begin{array}{|c|} \hline 101 \\ \hline \dots \\ \hline 101 \\ \hline \end{array} & \begin{array}{|c|} \hline 101 \\ \hline \dots \\ \hline 101 \\ \hline \end{array} & \begin{array}{|c|} \hline 101 \\ \hline \dots \\ \hline 101 \\ \hline \end{array} \dots \\ 1 & 3 & 3 & 3 \end{array}$$

Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(10, 101), (011, 11), (101, 011)\}$ hat keine Lösung.

$$K = \begin{array}{|c|c|c|} \hline \begin{array}{c} 10 \\ \hline 101 \end{array} & \begin{array}{c} 011 \\ \hline 11 \end{array} & \begin{array}{c} 101 \\ \hline 011 \end{array} \\ \hline (x_1, y_1) & (x_2, y_2) & (x_3, y_3) \end{array} \quad \times$$

$$\begin{array}{|c|c|c|} \hline 011 \\ \hline 11 \\ \hline \end{array} \quad 2$$

Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(001, 0), (01, 011), (01, 101), (10, 001)\}$

$$K = \begin{array}{|c|c|c|c|} \hline 001 & 01 & 01 & 10 \\ \hline 0 & 011 & 101 & 001 \\ \hline \end{array} \begin{array}{l} (x_1, y_1) \\ (x_2, y_2) \\ (x_3, y_3) \\ (x_4, y_4) \end{array}$$

Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(001, 0), (01, 011), (01, 101), (10, 001)\}$

$$K = \begin{array}{|c|c|c|c|} \hline 001 & 01 & 01 & 10 \\ \hline \dots & \dots & \dots & \dots \\ \hline 0 & 011 & 101 & 001 \\ \hline \end{array}$$

(x_1, y_1) (x_2, y_2) (x_3, y_3) (x_4, y_4)

$$\begin{array}{|c|} \hline 01 \\ \hline \dots \\ \hline 101 \\ \hline \end{array}$$

3

Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(001, 0), (01, 011), (01, 101), (10, 001)\}$

$$K = \begin{array}{|c|c|c|c|} \hline 001 & 01 & 01 & 10 \\ \hline \dots & \dots & \dots & \dots \\ \hline 0 & 011 & 101 & 001 \\ \hline \end{array} \begin{array}{l} (x_1, y_1) \\ (x_2, y_2) \\ (x_3, y_3) \\ (x_4, y_4) \end{array}$$

$$\begin{array}{|c|c|} \hline 001 & 01 \\ \hline \dots & \dots \\ \hline 0 & 101 \\ \hline \end{array} \begin{array}{l} 1 \\ 3 \end{array}$$

Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

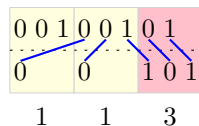
Beispiele

- $K = \{(001, 0), (01, 011), (01, 101), (10, 001)\}$

$$K = \begin{array}{|c|c|c|c|} \hline 001 & 01 & 01 & 10 \\ \hline \dots & \dots & \dots & \dots \\ \hline 0 & 011 & 101 & 001 \\ \hline \end{array}$$

(x_1, y_1) (x_2, y_2) (x_3, y_3) (x_4, y_4)

001	001	01
0	0	101
1	1	3



Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

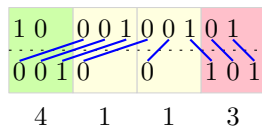
über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(001, 0), (01, 011), (01, 101), (10, 001)\}$

$$K = \begin{array}{|c|c|c|c|} \hline 001 & 01 & 01 & 10 \\ \hline \dots & \dots & \dots & \dots \\ \hline 0 & 011 & 101 & 001 \\ \hline \end{array}$$

(x_1, y_1) (x_2, y_2) (x_3, y_3) (x_4, y_4)



Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

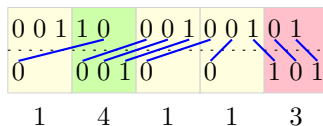
über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(001, 0), (01, 011), (01, 101), (10, 001)\}$

$$K = \begin{array}{|c|c|c|c|} \hline 001 & 01 & 01 & 10 \\ \hline \dots & \dots & \dots & \dots \\ \hline 0 & 011 & 101 & 001 \\ \hline \end{array}$$

(x_1, y_1) (x_2, y_2) (x_3, y_3) (x_4, y_4)



Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

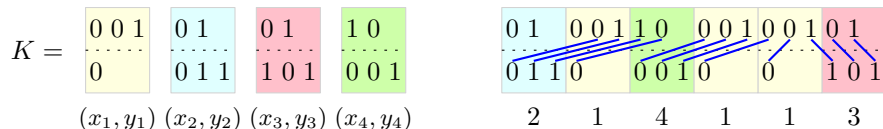
Gegeben ist eine endliche Menge von Wortpaaren

$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(001, 0), (01, 011), (01, 101), (10, 001)\}$



Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblem

Gegeben ist eine endliche Menge von Wortpaaren

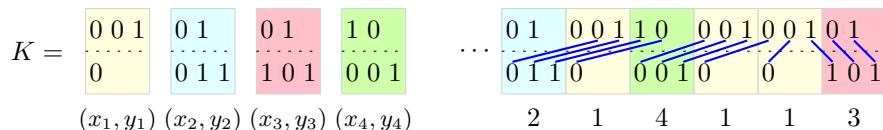
$$K = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

über einem endlichen Alphabet Σ . Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$ gibt, so dass $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ gilt.

Beispiele

- $K = \{(001, 0), (01, 011), (01, 101), (10, 001)\}$

↪ Die kürzeste Lösung hat Länge 66.



Unentscheidbarkeit des PKP

Satz.

Das Post'sche Korrespondenzproblem ist nicht entscheidbar.

Beweisidee:

Dies kann über die Nicht-Entscheidbarkeit des Halteproblems bewiesen werden.

Eigenschaften von (semi-)entscheidbaren Sprachen

- Die entscheidbaren Sprachen sind abgeschlossen unter Komplementbildung, Schnitt und Vereinigung.
- Die semi-entscheidbaren Sprachen sind abgeschlossen unter Schnitt und Vereinigung, aber nicht unter Komplementbildung.

Satz.

Sei $L \subseteq \Sigma^*$ und $L^c = \Sigma^* \setminus L$. Dann gilt

- L entscheidbar $\iff L^c$ entscheidbar.
- L entscheidbar $\iff L$ und L^c semi-entscheidbar.

Beweis: Übung und Tutorien.

■ Komplexitätstheorie

Komplexitätstheorie

Fragestellung bisher:

- Ist eine Sprache L entscheidbar oder nicht?
- Ist eine Funktion berechenbar oder nicht?
- Benutzung von deterministischen Turing-Maschinen.

In diesem Kapitel:

- Wie effizient kann ein Problem gelöst werden?
- Betrachtung von **nichtdeterministischen** Turing-Maschinen.

Frage (P vs. NP):

Gibt es einen wesentlichen Effizienzgewinn beim Übergang von der deterministischen Turing-Maschine zur nichtdeterministischen Turing-Maschine?

Wie sieht ein Problem aus?

Beispiel: Traveling Salesman Problem (TSP)

Gegeben sei ein vollständiger Graph $G = (V, E)$ mit ganzzahligen Kantengewichten c , d.h.

$$\blacksquare V := \{1, \dots, n\} \quad \blacksquare E := \{\{u, v\} : u, v \in V, u \neq v\} \quad \blacksquare c: E \rightarrow \mathbb{Z}^+$$

■ **Optimierungsproblem:**

Gesucht ist eine Tour (Rundreise), die alle Elemente aus V enthält und minimale Gesamtlänge unter allen solchen Touren hat.

■ **Optimalwertproblem:**

Gesucht ist die Länge einer minimalen Tour.

■ **Entscheidungsproblem:**

Gegeben sei zusätzlich auch ein Parameter $k \in \mathbb{Z}^+$. Die Frage ist nun: Gibt es eine Tour, deren Länge höchstens k ist?

Wie sieht ein Problem aus?

- **Optimierungsproblem:**

Gesucht ist eine Tour (Rundreise), die alle Elemente aus V enthält und minimale Gesamtlänge unter allen solchen Touren hat.

- **Optimalwertproblem:**

Gesucht ist die Länge einer minimalen Tour.

- **Entscheidungsproblem:**

Gegeben sei zusätzlich auch ein Parameter $k \in \mathbb{Z}^+$. Die Frage ist nun: Gibt es eine Tour, deren Länge höchstens k ist?

Bemerkung:

- Mit einer Lösung des Optimierungsproblems kann man leicht auch das Optimalwertproblem und das Entscheidungsproblem lösen.
- Mit einer Lösung des Optimalwertproblems kann man leicht auch das Entscheidungsproblem lösen.

Definition: Problem

Ein **Problem** Π ist gegeben durch:

- eine allgemeine Beschreibung aller vorkommenden Parameter;
- eine genaue Beschreibung der Eigenschaften, die die Lösung haben soll.

Eingabe: Graph $G = (V, E)$, Kantengewichtung $c: E \rightarrow \mathbb{Z}^+$, Zahl k

Lösung: zykl. Permutation $x_1 x_2 \cdots x_n$ von V mit $\{x_i, x_{i+1}\} \in E$ für $i = 1, \dots, n - 1$ und

$$\sum_{i=1}^{n-1} c(\{x_i, x_{i+1}\}) \leq k$$

Eine **Instanz** I von Π erhalten wir, indem wir die Parameter von Π festlegen. (**Problembispiel**)

$$V = \{a, b, c, d\}$$

$$E = \{\{a, b\}, \{a, c\}, \{c, d\}, \{b, d\}, \{b, c\}\}$$

$$c(\{a, b\}) = c(\{b, d\}) = c(\{b, c\}) = 1, c(\{a, c\}) = 2, c(\{c, d\}) = 4$$

Definition: Kodierungsschema

- Wir interessieren uns für die **Laufzeit** von Algorithmen.
- Diese wird in der Größe des Problems gemessen.

Die Größe eines Problems ist abhängig von der Beschreibung oder Kodierung der Instanzen.

- Ein **Kodierungsschema** s ordnet jeder Instanz I eines Problems ein Wort oder *Kodierung* $s(I)$ über einem Alphabet Σ zu.

$$V = \{a, b, c, d\}$$

$$E = \{\{a, b\}, \{a, c\}, \{c, d\}, \{b, d\}, \{b, c\}\}$$

$$c(\{a, b\}) = c(\{b, d\}) = c(\{b, c\}) = 1, c(\{a, c\}) = 2, c(\{c, d\}) = 4$$

$$s(I) = 00|01|10|11 \sqcup 00 * 01|00 * 10|10 * 11|01 * 11|01 * 10 \sqcup 1|2|4|1|1$$

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, |, \sqcup, *\}$$

Definition: Kodierungsschema

- Wir interessieren uns für die **Laufzeit** von Algorithmen.
- Diese wird in der Größe des Problems gemessen.

Die Größe eines Problems ist abhängig von der Beschreibung oder Kodierung der Instanzen.

- Ein **Kodierungsschema** s ordnet jeder Instanz I eines Problems ein Wort oder *Kodierung* $s(I)$ über einem Alphabet Σ zu.

$$s(I) = 00|01|10|11 \sqcup 00 * 01|00 * 10|10 * 11|01 * 11|01 * 10 \sqcup 1|2|4|1|1$$

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, |, \sqcup, *\}$$

- Die **Inputlänge** einer Instanz ist die Anzahl der Symbole ihrer Kodierung.
hier: $|s(I)| = 51$

Kodierungsschema

Es gibt verschiedene Kodierungsschemata für ein bestimmtes Problem.

Beispiel:

- Zahlen können dezimal, binär, unär, usw. kodiert werden.
- Die Inputlänge von 5127 beträgt dann 4 für dezimal, 13 für binär und 5127 für unär.

Wir werden uns auf vernünftige Schemata festlegen:

- Die Kodierung einer Instanz soll keine überflüssigen Informationen enthalten.
- Zahlen sollen binär (oder k -är für $k \neq 1$) kodiert sein.

Kodierungsschema

Dies bedeutet, die Kodierungslänge

- einer ganzen Zahl n ist $\lfloor \log_k |n| + 1 \rfloor + 1 =: \langle n \rangle$
(eine 1 benötigt man für das Vorzeichen);
- einer rationalen Zahl $r = \frac{p}{q}$ ist $\langle r \rangle = \langle p \rangle + \langle q \rangle$;
- eines Vektors $X = (x_1, \dots, x_n)$ ist $\langle X \rangle := \sum_{i=1}^n \langle x_i \rangle$;
- einer Matrix $A \in \mathbb{Q}^{m \times n}$ ist $\langle A \rangle := \sum_{i=1}^m \sum_{j=1}^n \langle a_{ij} \rangle$.
- eines Graphen $G = (V, E)$ kann zum Beispiel durch die Kodierung seiner *Adjazenzmatrix*, die eines gewichteten Graphen durch die Kodierung der *Gewichtsmatrix* beschrieben werden.

Äquivalenz von Kodierungsschemata

Zwei Kodierungsschemata s_1, s_2 heißen **äquivalent** bezüglich eines Problems Π , falls es Polynome p_1, p_2 gibt, so dass gilt:

$$|s_1(I)| = n \implies |s_2(I)| \leq p_2(n)$$

und

$$|s_2(I)| = m \implies |s_1(I)| \leq p_1(m)$$

für alle Instanzen I von Π .

Entscheidungsprobleme

- Ein Entscheidungsproblem Π können wir als Familie / Klasse D_{Π} von Instanzen auffassen.
 - Mit festem Kodierungsschema s ist das eine Menge von Wörtern über Σ \rightsquigarrow unsere Eingaben
- Eine Teilmenge dieser Klasse ist $J_{\Pi} \subseteq D_{\Pi}$, die Klasse der **Ja-Instanzen**, d.h. die Instanzen deren Antwort Ja ist.
- Der Rest der Klasse $N_{\Pi} \subseteq D_{\Pi}$ ist die Klasse der **Nein-Instanzen**.

Korrespondenz von Entscheidungsproblemen und Sprachen

Ein Problem Π und ein Kodierungsschema $s: D_\Pi \rightarrow \Sigma^*$ zerlegen Σ^* in drei Klassen:

- Wörter aus Σ^* , die *nicht* Kodierung eines Beispiels aus D_Π sind,
- Wörter aus Σ^* , die Kodierung einer Instanz $I \in N_\Pi$ sind,
- Wörter aus Σ^* , die Kodierung einer Instanz $I \in J_\Pi$ sind.

Die dritte Klasse ist die Sprache, die zu Π im Kodierungsschema s **korrespondiert**.

Die zu einem Problem Π und einem Kodierungsschema s **zugehörige Sprache** ist

$$L[\Pi, s] := \left\{ x \in \Sigma^* : \begin{array}{l} \Sigma \text{ ist das Alphabet zu } s \text{ und } x \text{ ist Kodierung} \\ \text{einer Ja-Instanz } I \text{ von } \Pi \text{ unter } s, \text{ d.h. } I \in J_\Pi \end{array} \right\}$$

Entscheidungsprobleme und Turing-Maschinen

- Wir betrachten im folgenden deterministische Turing-Maschinen mit zwei Endzuständen q_J, q_N , wobei q_J akzeptierender Endzustand ist.
- Dann wird die Sprache $L_{\mathcal{M}}$ akzeptiert von der Turing-Maschine \mathcal{M} , falls
$$L_{\mathcal{M}} = \{x \in \Sigma^* : \mathcal{M} \text{ akzeptiert } x\} .$$
- Eine deterministische Turing-Maschine \mathcal{M} **löst** ein Entscheidungsproblem Π unter einem Kodierungsschema s , falls \mathcal{M} bei jeder Eingabe über dem Eingabe-Alphabet in einem Endzustand endet und $L_{\mathcal{M}} = L[\Pi, s]$ ist.
 - D.h. die Turing-Maschine \mathcal{M} **entscheidet** $L[\Pi, s]$.

Zeitkomplexität

Für eine deterministische Turing-Maschine \mathcal{M} , die für alle Eingaben über dem Eingabe-Alphabet Σ hält, ist die **Zeitkomplexitätsfunktion** $T_{\mathcal{M}}: \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ definiert durch

$$T_{\mathcal{M}}(n) = \max \left\{ m: \begin{array}{l} \text{es gibt eine Eingabe } x \in \Sigma^* \text{ mit } |x| = n, \text{ so dass die Berechnung} \\ \text{von } \mathcal{M} \text{ bei Eingabe } x \text{ } m \text{ Berechnungsschritte (Übergänge) benötigt,} \\ \text{bis ein Endzustand erreicht wird} \end{array} \right\}$$

Zeitkomplexität

Für eine deterministische Turing-Maschine \mathcal{M} , die für alle Eingaben über dem Eingabe-Alphabet Σ hält, ist die **Zeitkomplexitätsfunktion** $T_{\mathcal{M}}: \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ definiert durch

$$T_{\mathcal{M}}(n) = \max \left\{ m: \begin{array}{l} \text{es gibt eine Eingabe } x \in \Sigma^* \text{ mit } |x| = n, \text{ so dass die Berechnung} \\ \text{von } \mathcal{M} \text{ bei Eingabe } x \text{ } m \text{ Berechnungsschritte (Übergänge) benötigt,} \\ \text{bis ein Endzustand erreicht wird} \end{array} \right\}$$

Bemerkungen

- Wenn Eingabe x Länge n hat, so braucht \mathcal{M} höchstens $T_{\mathcal{M}}(n)$ Berechnungsschritte.
- Für ein Entscheidungsproblem Π mit Kodierungsschema s :

Instanz $I \in D_{\Pi}$ von Π	\Leftrightarrow	Kodierung $s(I) \in \Sigma^*$
Kodierungslänge $ s(I) $	\Leftrightarrow	Länge des Wortes
Zeit zum Lösen von I	\Leftrightarrow	Anzahl Berechnungsschritte von \mathcal{M}

Die Klasse \mathcal{P}

Die Klasse \mathcal{P} ist die Menge aller Sprachen L (Entscheidungsprobleme), für die eine deterministische Turing-Maschine existiert, deren Zeitkomplexitätsfunktion polynomial beschränkt ist, d.h. es existiert ein Polynom p mit

$$T_M(n) \leq p(n).$$

- Zum Beispiel $T_M(n) \leq 4n^2 + 42$.
- Sprachen / Entscheidungsprobleme in \mathcal{P} sind “effizient lösbar”.

Schwierigkeit von Entscheidungs- und Optimierungsproblem

Satz.

Falls es einen Algorithmus \mathcal{A} gibt, der das Entscheidungsproblem des TSP in polynomialer Zeit löst, so gibt es auch einen Algorithmus, der das Optimierungsproblem in polynomialer Zeit löst.

Satz.

Falls es einen Algorithmus \mathcal{A} gibt, der das Entscheidungsproblem des TSP in polynomialer Zeit löst, so gibt es auch einen Algorithmus, der das Optimierungsproblem in polynomialer Zeit löst.

Beweis: Algorithmus, der das Optimierungsproblem löst.

Input: $G = (V, E)$, $c_{ij} = c(\{i, j\})$ für $i, j \in V := \{1, \dots, n\}$,
Algorithmus \mathcal{A}

Output: d_{ij} ($1 \leq i, j \leq n$), so dass alle bis auf n der d_{ij} -Werte den Wert $1 + \max\{c_{ij} \mid 1 \leq i, j \leq n\}$ haben. Die restlichen n d_{ij} -Werte haben den Wert c_{ij} und geben genau die Kanten einer optimalen Tour an.

Algorithmus OPT-TOUR (als Beweis) 1/2

- 1 berechne $m := \max\{c_{ij} \mid 1 \leq i, j \leq n\}$;
- 2 setze $L(\text{ow}) := 0$ und $H(\text{igh}) := n \cdot m$; $(L \leq OPT \leq H)$
- 3 **Solange** $H - L > 1$ gilt, führe aus: (binäre Suche nach OPT)
- 4 **Falls** $\mathcal{A}(G, c, \lceil \frac{1}{2}(H + L) \rceil) = \text{“nein”}$, $(OPT > \lceil \frac{1}{2}(H + L) \rceil)$
- 5 setze $L := \lceil \frac{1}{2}(H + L) \rceil + 1$;
- 6 **Sonst** $(OPT \leq \lceil \frac{1}{2}(H + L) \rceil)$
- 7 setze $H := \lceil \frac{1}{2}(H + L) \rceil$;
- 8 **Falls** $\mathcal{A}(G, c, L) = \text{“nein”}$ (hier gilt $H - L \leq 1$)
- 9 setze $OPT := H$;
- 10 **Sonst**
- 11 setze $OPT := L$;

Algorithmus OPT-TOUR (als Beweis) 2/2

Wir kennen den Optimalwert OPT und finden jetzt eine optimale Tour.

- 12 **Für** $i = 1 \dots n$ führe aus
- 13 **Für** $j = 1 \dots n$ führe aus
- 14 setze $R := c_{ij}$; (merke Länge der Kante ij)
- 15 setze $c_{ij} := m + 1$; (mache Kante ij zu lang)
- 16 **Falls** $\mathcal{A}(G, c, OPT) = \text{“nein”}$, (Kante ij in opt. Tour)
- 17 setze $c_{ij} := R$; (Kante ij wie vorher)
- 18 setze $d_{ij} := c_{ij}$;

Bemerkungen zum Algorithmus

Die Schleife der binären Suche bricht ab, und danach ist die Differenz $H - L$ gleich 1 oder 0, denn:

- Solange $H - L > 1$, ändert sich bei jedem Schleifendurchlauf einer der Werte H, L :
 - Für $H - L > 1$ gilt, dass $L \neq \lceil \frac{1}{2}(H + L) \rceil + 1$ und $H \neq \lceil \frac{1}{2}(H + L) \rceil$ ist.
- Die Differenz $H - L$ verkleinert sich mit jedem Durchlauf
- Da H und L ganzzahlig sind, tritt der Fall $H - L \leq 1$ ein.
- Zu jedem Zeitpunkt gilt $H - L \geq 0$:
 - $H - L = 0$ ist möglich, wenn zum Beispiel L auf $\lceil \frac{1}{2}(H + L) \rceil + 1$ erhöht wird und vorher $H - L = 2$ oder $H - L = 3$ war.

Laufzeit des Algorithmus

- $\mathcal{A}(G, c, k)$ wird (für verschiedene k) etwa $\log(n \cdot m)$ -mal aufgerufen.
- $\mathcal{A}(G, c, OPT)$ wird etwa n^2 -mal aufgerufen.
- Es finden also $O(n^2 + \log(nm))$ Aufrufe von \mathcal{A} statt.
- Die Inputlänge ist $O(n^2 \cdot \max \langle c_{ij} \rangle) = O(n^2 \cdot \max \log c_{ij})$.
- Da \mathcal{A} polynomiell ist, ist dies also auch OPT-TOUR.

Zwischenstand Komplexitätstheorie

■ Probleme

- Optimierungsprobleme, Optimalwertprobleme, Entscheidungsprobleme
- Problem Π ist Klasse D_{Π} von Instanzen I .

■ Eingabegrößen

- Kodierungsschema $s: D_{\Pi} \rightarrow \Sigma^*$ über Alphabet Σ^* .
- Kodierung $s(I)$ einer Instanz I ist ein Wort aus Σ^* .
- Inputlänge $|s(I)|$ ist Länge des Wortes.

■ Entscheidungsprobleme

- Ja–Instanzen J_{Π} und Nein–Instanzen N_{Π}
- Sprache $L[\Pi, s]$ der Kodierungen aller Ja–Instanzen
- TM \mathcal{M} löst Π wenn \mathcal{M} Sprache $L[\Pi, s]$ entscheidet.

■ Laufzeiten

- Zeitkomplexitätsfunktion $T_{\mathcal{M}}(n)$ von \mathcal{M} bei Eingaben der Länge n
- Die Klasse \mathcal{P} : Sprachen von \mathcal{M} mit $T_{\mathcal{M}}(n)$ polynomiell in n .

- Am Beispiel TSP: Entscheidung \rightarrow Optimalwert \rightarrow Optimierung

Zwischenstand Komplexitätstheorie

■ Probleme

- Optimierungsprobleme, Optimalwertprobleme, Entscheidungsprobleme
- Problem Π ist Klasse D_{Π} von Instanzen I .

■ Eingabegrößen

- Kodierungsschema $s: D_{\Pi} \rightarrow \Sigma^*$ über Alphabet Σ^* .
- Kodierung $s(I)$ einer Instanz I ist ein Wort aus Σ^* .
- Inputlänge $|s(I)|$ ist Länge des Wortes.

Testen Sie sich!

■ Entscheidungsprobleme

- Ja-Instanzen J_{Π} und Nein-Instanzen N_{Π}
- Sprache $L[\Pi, s]$ der Kodierungen aller Ja-Instanzen
- TM \mathcal{M} löst Π wenn \mathcal{M} Sprache $L[\Pi, s]$ entscheidet.

■ Laufzeiten

- Zeitkomplexitätsfunktion $T_{\mathcal{M}}(n)$ von \mathcal{M} bei Eingaben der Länge n
- Die Klasse \mathcal{P} : Sprachen von \mathcal{M} mit $T_{\mathcal{M}}(n)$ polynomiell in n .

- Am Beispiel TSP: Entscheidung \rightarrow Optimalwert \rightarrow Optimierung