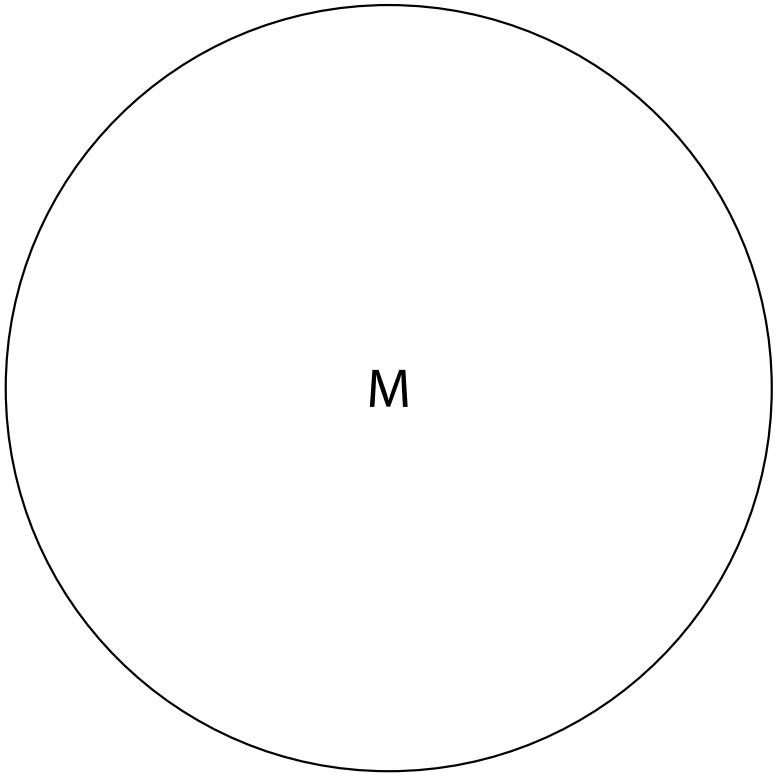
UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR
THEORETISCHE INFORMATIK

Inclusion-Exclusion

Oberseminar

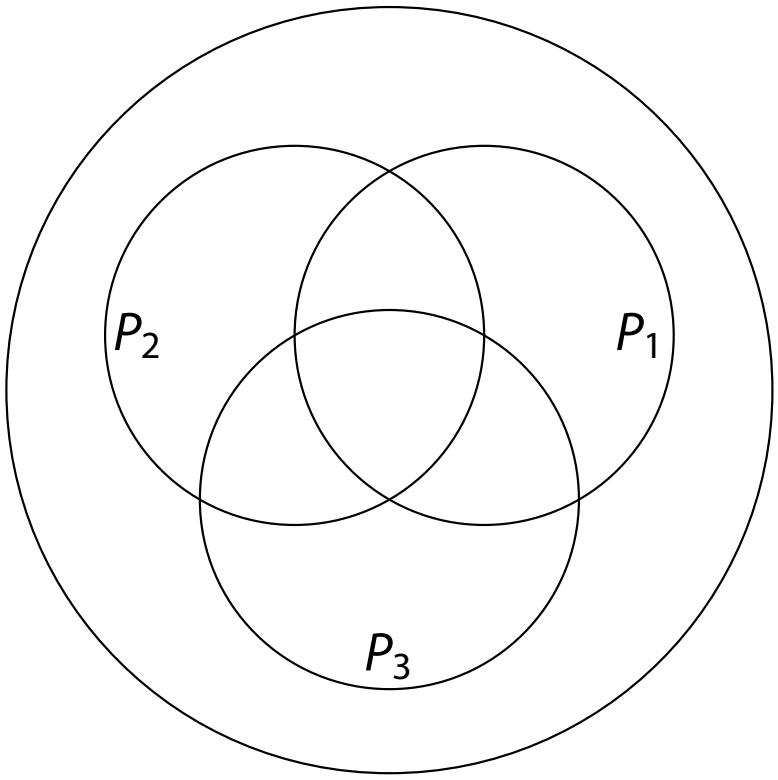
Sebastian Berndt

Ausgangssituation



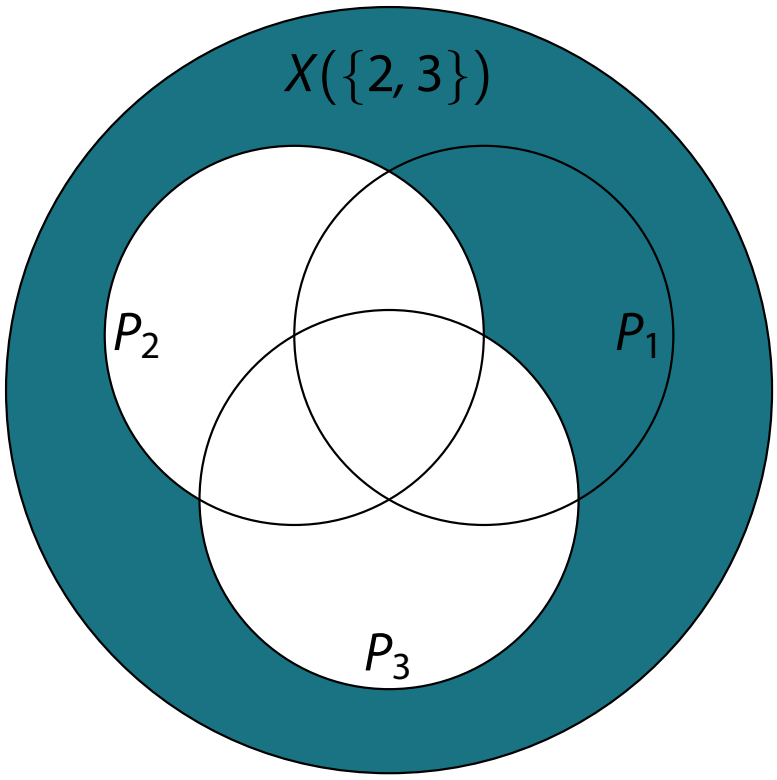
- Endliche Menge *M*

Ausgangssituation



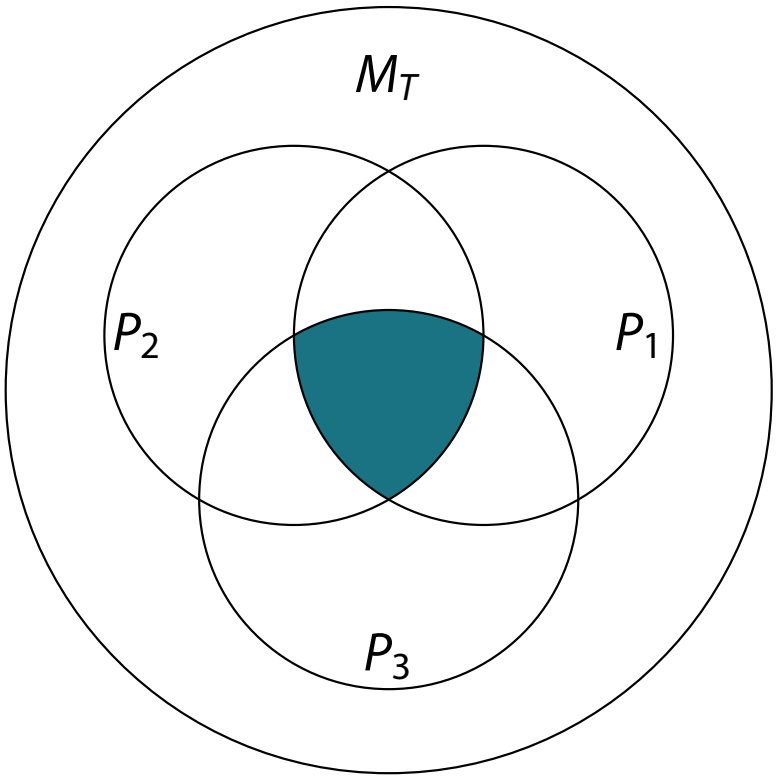
- Endliche Menge M
- Prädikate $P_1, \ldots, P_n$ mit $P_i : M \to \{T, F\}$

Ausgangssituation



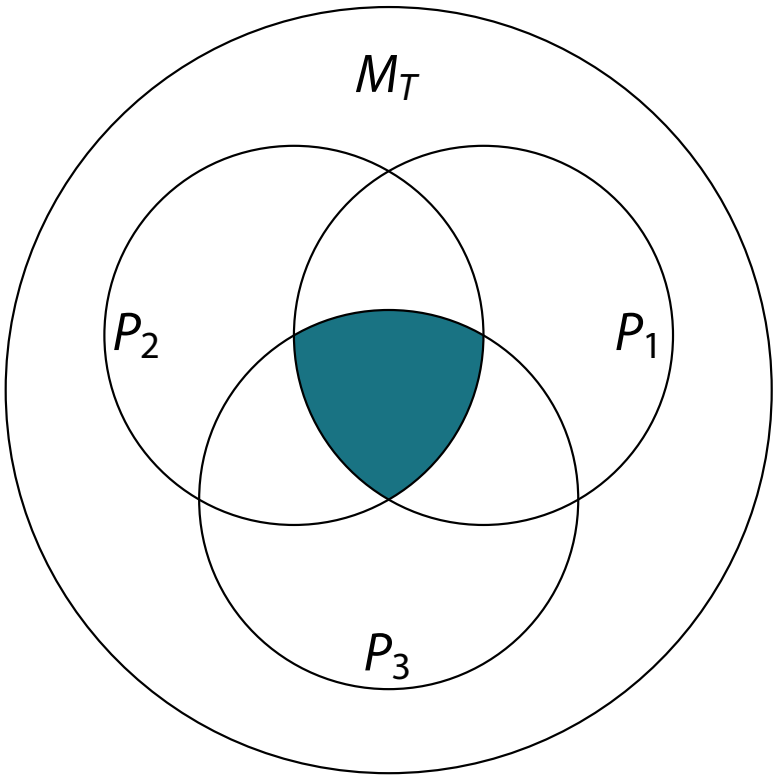
- Endliche Menge M
- Prädikate $P_1, \ldots, P_n$ mit $P_i : M \to \{T, F\}$
- $X(I) = \{m \in M \mid P_i(m) = F \; \forall i \in I\}$

Ausgangssituation



- Endliche Menge M
- Prädikate $P_1, \ldots, P_n$ mit $P_i : M \to \{T, F\}$
- $X(I) = \{m \in M \mid P_i(m) = F \; \forall i \in I\}$
- $M_T = \{m \in M \mid P_i(m) = T \; \forall i \in [n]\}$

Ausgangssituation



- Endliche Menge M
- Prädikate $P_1, \ldots, P_n$ mit $P_i : M \to \{T, F\}$
- $X(I) = \{m \in M \mid P_i(m) = F \; \forall i \in I\}$
- $M_T = \{m \in M \mid P_i(m) = T \; \forall i \in [n]\}$

Frage

Wie groß ist M_T?

Allgemeines Prinzip

Satz

$$|M_T| = \sum_{I \subseteq [n]} (-1)^{|I|} |X(I)|$$

Allgemeines Prinzip

Satz

$$|M_T| = \sum_{I \subseteq [n]} (-1)^{|I|} |X(I)|$$

Anwendung

1. Bestimme M

Allgemeines Prinzip

Satz

$$|M_T| = \sum_{I \subseteq [n]} (-1)^{|I|} |X(I)|$$

Anwendung

1. Bestimme M
2. Bestimme $|X(I)|$ für festes I

Allgemeines Prinzip

Satz

$$|M_T| = \sum_{I \subseteq [n]} (-1)^{|I|} |X(I)|$$

Anwendung

1. Bestimme M
2. Bestimme $|X(I)|$ für festes I
3. Bestimme $|M_T|$

Allgemeines Prinzip

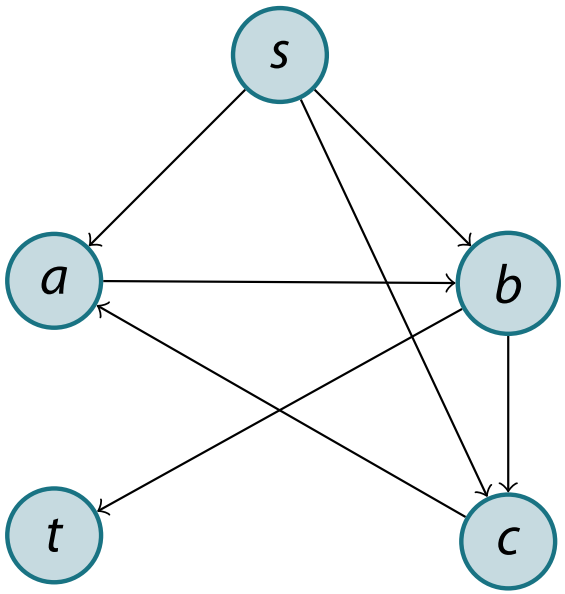
Satz

$$|M_T| = \sum_{I \subseteq [n]} (-1)^{|I|} |X(I)|$$

Anwendung

1. Bestimme M
2. Bestimme $|X(I)|$ für festes I
3. Bestimme $|M_T|$
4. Laufzeit: $\mathcal{O}^*(2^n)$

Ein Beispiel: Hamiltonpfad

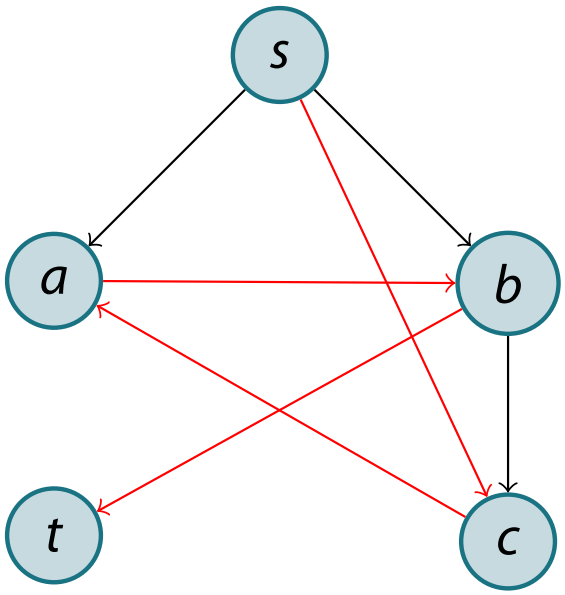


Problemstellung (Hamiltonpfad)

Gegeben *Gerichteter Graph* $G = (V, E)$, $V = \{s, t, v_1, \ldots, v_n\}$

Gesucht *Anzahl der s-t-Pfade, die jeden Knoten genau einmal enthalten*

Ein Beispiel: Hamiltonpfad



Problemstellung (Hamiltonpfad)

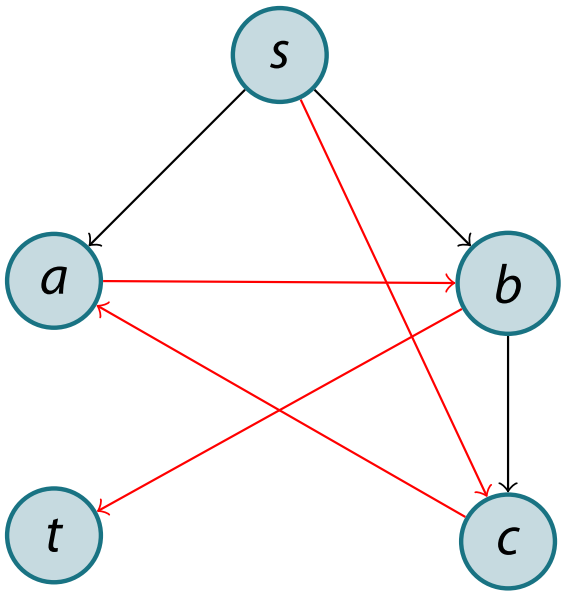
Gegeben *Gerichteter Graph* $G = (V, E)$, $V = \{s, t, v_1, \ldots, v_n\}$

Gesucht *Anzahl der s-t-Pfade, die jeden Knoten genau einmal enthalten*

Satz (Karp 1971)

Das Finden eines Hamiltonpfades ist $\mathcal{NP}$-schwer.

Ein Beispiel: Hamiltonpfad



Problemstellung (Hamiltonpfad)

Gegeben *Gerichteter Graph* $G = (V, E)$, $V = \{s, t, v_1, \ldots, v_n\}$

Gesucht *Anzahl der s-t-Pfade, die jeden Knoten genau einmal enthalten*

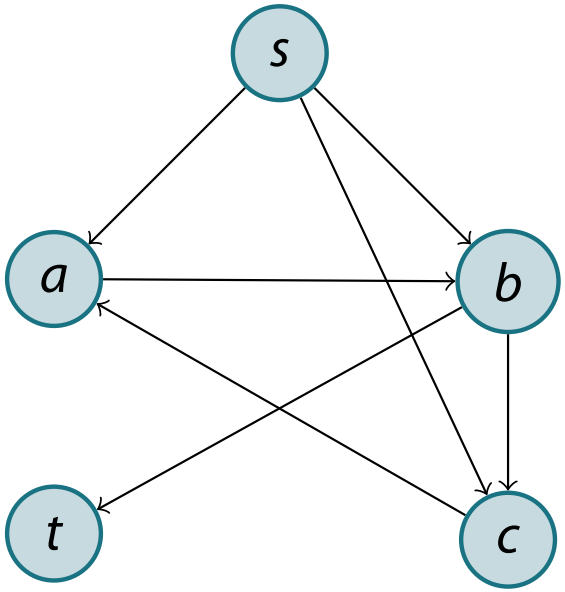
Satz (Karp 1971)

Das Finden eines Hamiltonpfades ist $\mathcal{NP}$-schwer.

Algorithmus

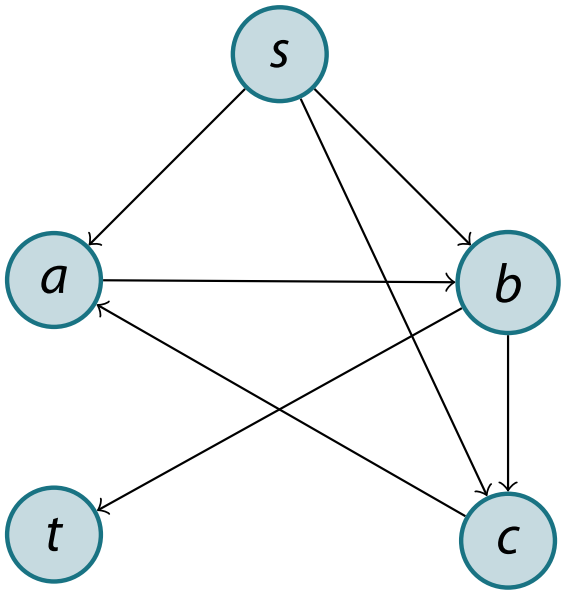
Ausprobieren aller Pfade: $\mathcal{O}^*(n!)$

Lösung per Inclusion-Exclusion



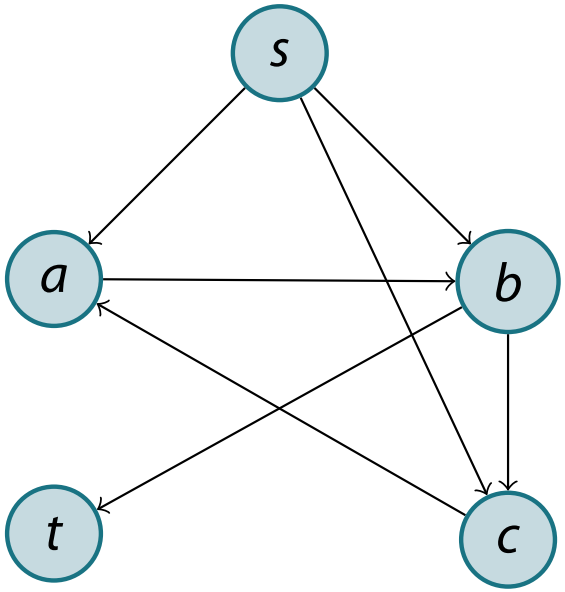
- Definiere $M = \{\Pi_1, \ldots, \Pi_k\}$

Lösung per Inclusion-Exclusion



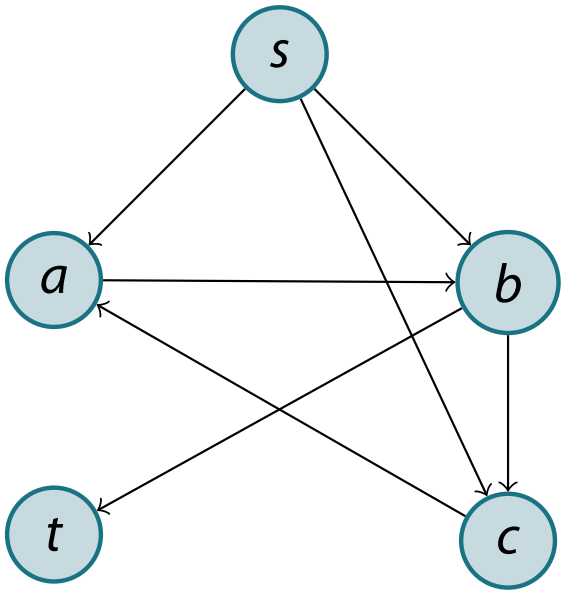
- Definiere $M = \{\Pi_1, \ldots, \Pi_k\}$
- Π_i ist s-t-Weg der Länge $n + 1$

Lösung per Inclusion-Exclusion



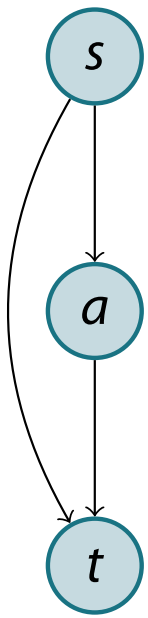
- Definiere $M = \{\Pi_1, \ldots, \Pi_k\}$
- Π_i ist s-t-Weg der Länge $n + 1$
- $P_i(\Pi) = T \Leftrightarrow v_i \in \Pi$

Lösung per Inclusion-Exclusion



- Definiere $M = \{\Pi_1, \ldots, \Pi_k\}$
- Π_i ist s-t-Weg der Länge $n + 1$
- $P_i(\Pi) = T \Leftrightarrow v_i \in \Pi$
- Π ist Hamiltonpfad $\Leftrightarrow \bigwedge_{j=1}^{n} P_j(\Pi) = T$

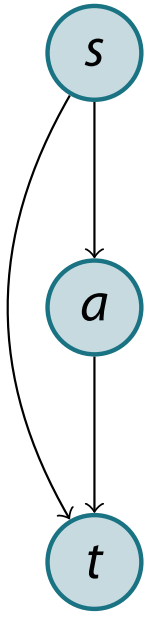
Bestimmung von $X(I)$



Beobachtung

Für $I \subseteq [n]$ ist $X(I)$ Menge aller s-t-Wege der Länge $n + 1$, die nicht über v_i gehen.

Bestimmung von $X(I)$



$A[G]=$

	s	a	t
s	0	1	1
a	0	0	1
t	0	0	0

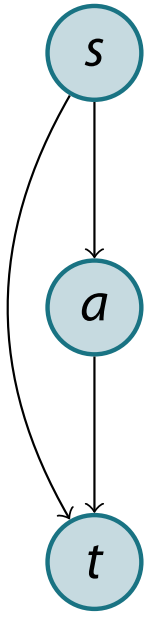
Beobachtung

Für $I \subseteq [n]$ ist $X(I)$ Menge aller s-t-Wege der Länge $n+1$, die nicht über v_i gehen.

Satz

Sei $A[G]$ Adjazenzmatrix zu G. Dann enthält $A[G]^k$ die Anzahl der Wege der Länge $k+1$ zwischen zwei Knoten.

Bestimmung von $X(I)$



$A[G]^2 =$

	s	a	t
s	0	0	1
a	0	0	0
t	0	0	0

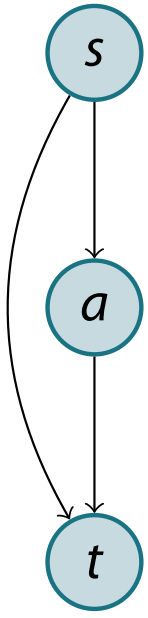
Beobachtung

Für $I \subseteq [n]$ ist $X(I)$ Menge aller s-t-Wege der Länge $n+1$, die nicht über v_i gehen.

Satz

Sei $A[G]$ Adjazenzmatrix zu G. Dann enthält $A[G]^k$ die Anzahl der Wege der Länge $k+1$ zwischen zwei Knoten.

Bestimmung von $X(I)$



$A[G]^2 =$

	s	a	t
s	0	0	1
a	0	0	0
t	0	0	0

Beobachtung

Für $I \subseteq [n]$ ist $X(I)$ Menge aller s-t-Wege der Länge $n+1$, die nicht über v_i gehen.

Satz

Sei $A[G]$ Adjazenzmatrix zu G. Dann enthält $A[G]^k$ die Anzahl der Wege der Länge $k+1$ zwischen zwei Knoten.

Satz

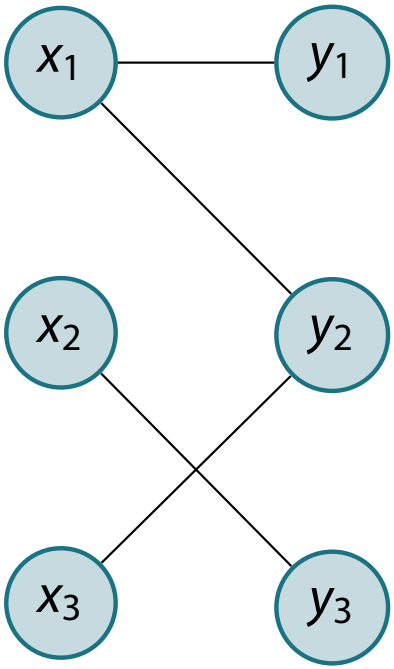
$$|X(I)| = (A[G \setminus I]^{n+1})_{st}$$

Algorithmus

Satz (Kohn, Gottlieb, Kohn 1969)

Anzahl der Hamiltonpfade eines Graphen berechenbar in Zeit $\mathcal{O}^*(2^n)$

Ein Beispiel: Perfekte Matchings



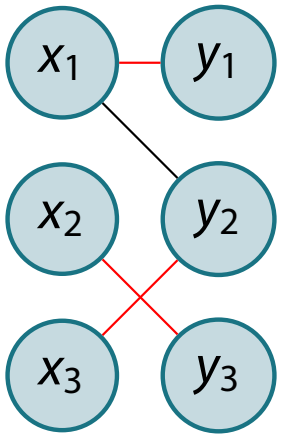
Problemstellung

Perfekte Matchings

Gegeben *Bipartiter Graph* $G = (V, E)$

Gesucht *Anzahl perfekter Matchings von* G

Ein Beispiel: Perfekte Matchings



Problemstellung

Perfekte Matchings

Gegeben *Bipartiter Graph* $G = (V, E)$

Gesucht *Anzahl perfekter Matchings von* G

Erinnerung

Perfektes Matching deckt *alle* Knoten ab

Lösungsansatz

Stelle G als vereinfachte Adjazenzmatrix $A[G] = (a_{ij})$ dar.

Lösungsansatz

Stelle G als vereinfachte Adjazenzmatrix $A[G] = (a_{ij})$ dar.

Beobachtung

$S = \{\{x_1, y_{\pi(1)}\}, \ldots, \{x_n, y_{\pi(n)}\}\}$ ist perfektes Matching gdw.

$$\prod_{i=1}^{n} a_{i\pi(i)} = 1$$

Lösungsansatz

Stelle G als vereinfachte Adjazenzmatrix $A[G] = (a_{ij})$ dar.

Beobachtung

$S = \{\{x_1, y_{\pi(1)}\}, \ldots, \{x_n, y_{\pi(n)}\}\}$ ist perfektes Matching gdw.

$$\prod_{i=1}^{n} a_{i\pi(i)} = 1$$

Satz

Anzahl der perfekten Matchings ist:

$$\sum_{\pi \in S_n} \prod_{i=1}^{n} a_{i\pi(i)} =: \operatorname{perm}(A)$$

Laufzeit: $\mathcal{O}^*(n!)$

Schwierigkeiten

Satz (Valiant 1979)

perm(A) *zu bestimmen ist* $\#\mathcal{P}$*-vollständig.*

Schwierigkeiten

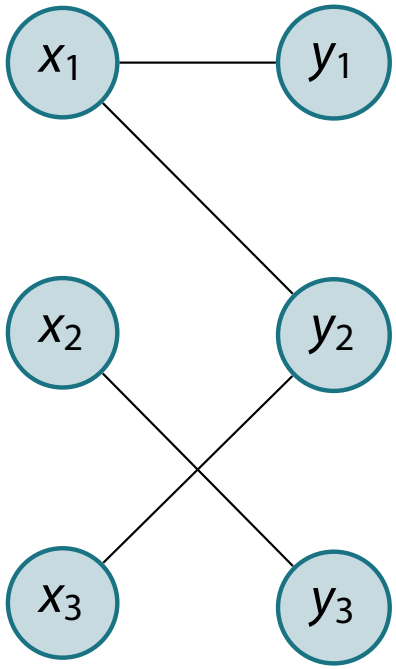
Satz (Valiant 1979)

perm(A) *zu bestimmen ist* $\#\mathcal{P}$*-vollständig.*

Erinnerung

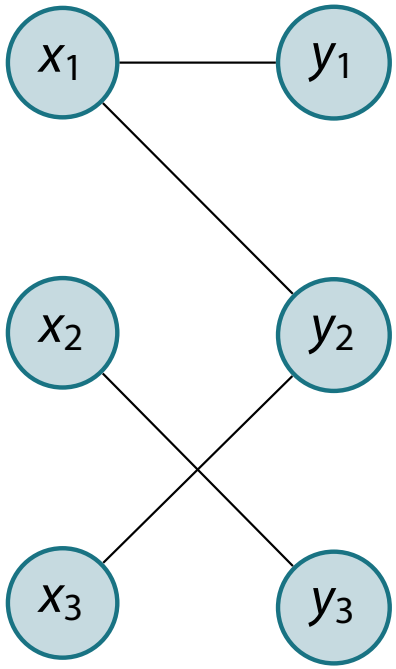
$\#\mathcal{P}$ ist Menge aller Funktionen, die Läufe *nichtdeterministischer* TMs zählen.

Schnelleres Verfahren



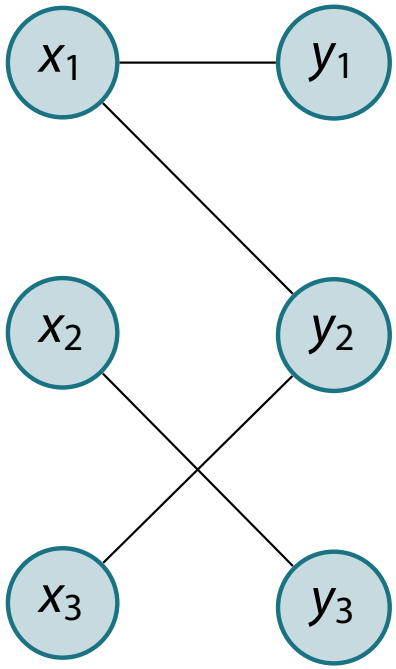
- Definiere $M = \{S_1, \ldots, S_k\}$

Schnelleres Verfahren



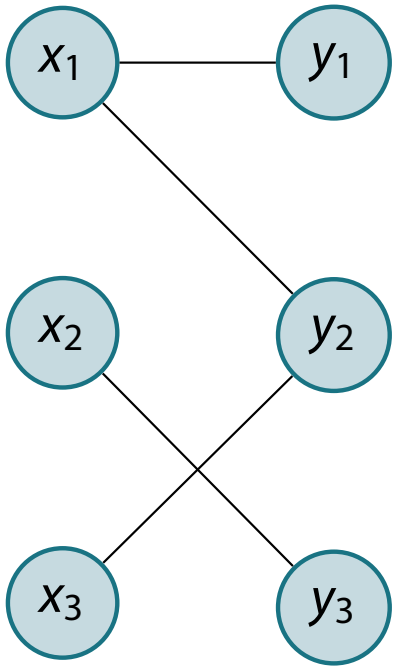
- Definiere $M = \{S_1, \ldots, S_k\}$
- $S_i = \{e_1, \ldots, e_n\}$ mit $\{x_1, \ldots, x_n\} \subseteq \bigcup_{j=1}^{n} e_j$

Schnelleres Verfahren



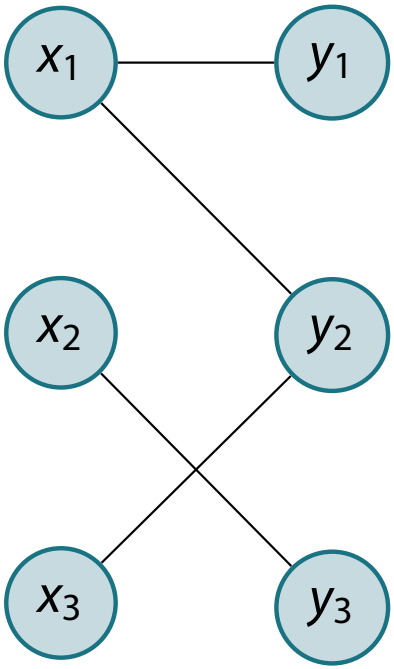
- Definiere $M = \{S_1, \ldots, S_k\}$
- $S_i = \{e_1, \ldots, e_n\}$ mit $\{x_1, \ldots, x_n\} \subseteq \bigcup_{j=1}^{n} e_j$
- $P_j(S) = T \Leftrightarrow y_j$ kommt in S vor

Schnelleres Verfahren



- Definiere $M = \{S_1, \ldots, S_k\}$
- $S_i = \{e_1, \ldots, e_n\}$ mit $\{x_1, \ldots, x_n\} \subseteq \bigcup_{j=1}^{n} e_j$
- $P_j(S) = T \Leftrightarrow y_j$ kommt in S vor
- S ist perfektes Matching $\Leftrightarrow \bigwedge_{j=1}^{n} P_j(S) = T$

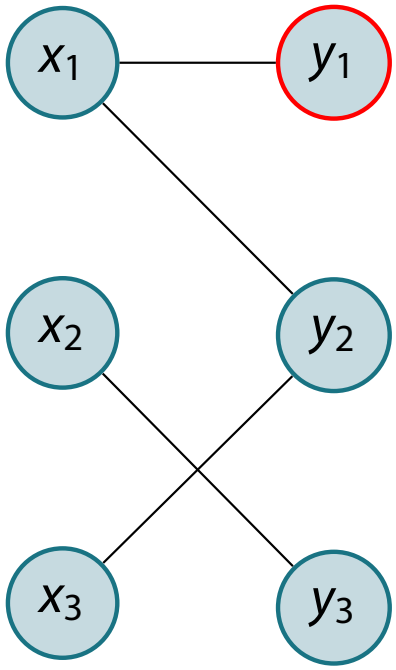
Bestimmung von $X(I)$



Beobachtung

Für $I \subseteq [n]$ ist $X(I)$ Menge aller S, so dass kein y_i in S auftaucht.

Bestimmung von $X(I)$



$X(\{1\})$

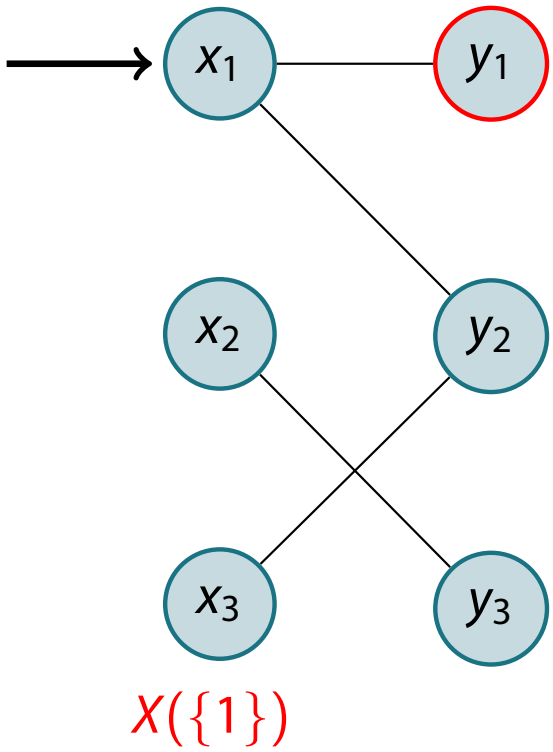
Beobachtung

Für $I \subseteq [n]$ ist $X(I)$ Menge aller S, so dass kein y_i in S auftaucht.

Satz

$$|X(I)| = \prod_{i=1}^{n} \sum_{j \notin I} a_{ij}$$

Bestimmung von $X(I)$



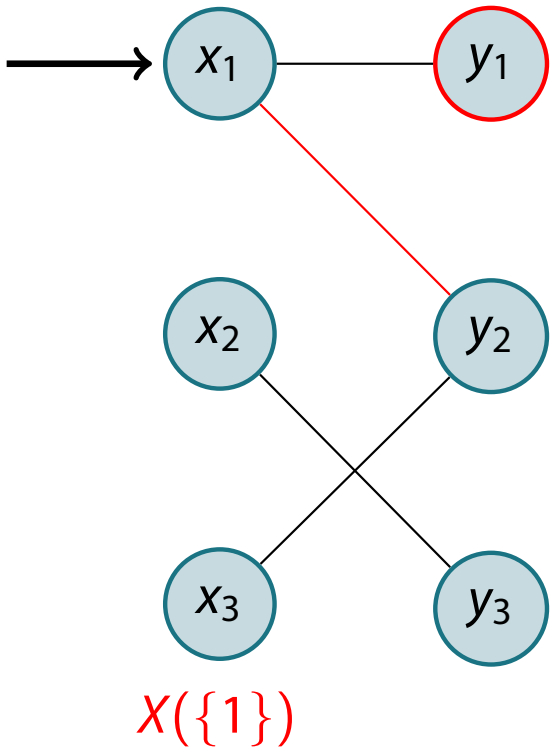
Beobachtung

Für $I \subseteq [n]$ ist $X(I)$ Menge aller S, so dass kein y_i in S auftaucht.

Satz

$$|X(I)| = \prod_{i=1}^{n} \sum_{j \notin I} a_{ij}$$

Bestimmung von $X(I)$



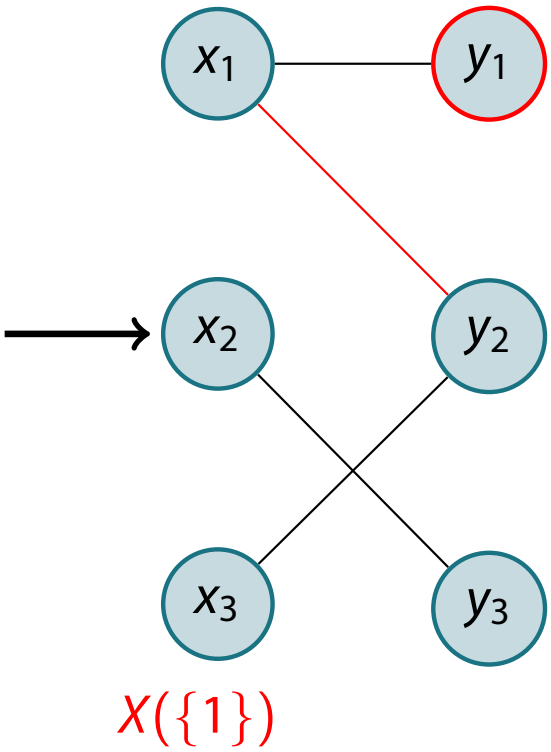
Beobachtung

Für $I \subseteq [n]$ ist $X(I)$ Menge aller S, so dass kein y_i in S auftaucht.

Satz

$$|X(I)| = \prod_{i=1}^{n} \sum_{j \notin I} a_{ij}$$

Bestimmung von $X(I)$



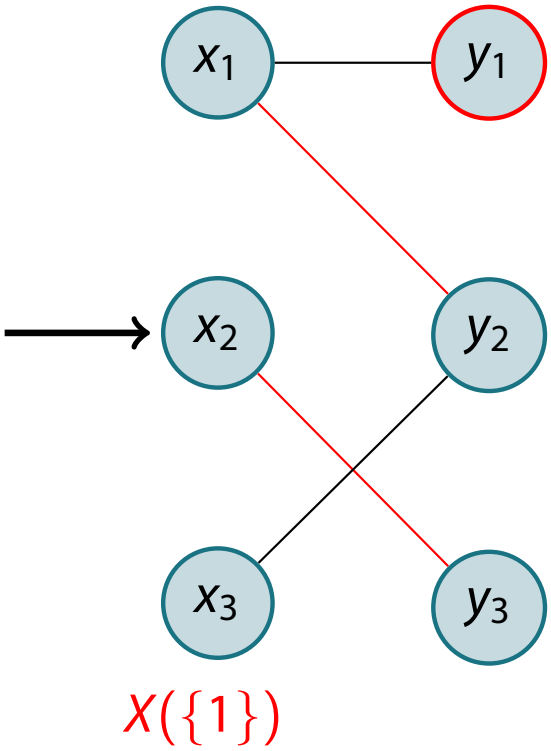
Beobachtung

Für $I \subseteq [n]$ ist $X(I)$ Menge aller S, so dass kein y_i in S auftaucht.

Satz

$$|X(I)| = \prod_{i=1}^{n} \sum_{j \notin I} a_{ij}$$

Bestimmung von $X(I)$



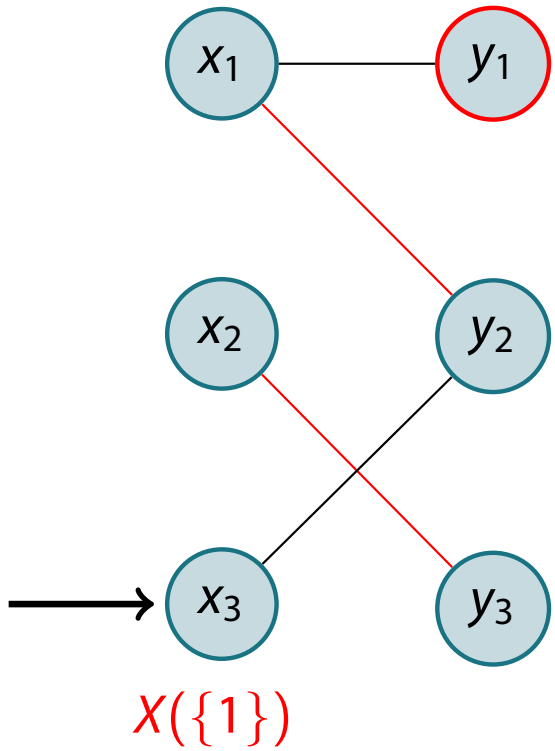
Beobachtung

Für $I \subseteq [n]$ ist $X(I)$ Menge aller S, so dass kein y_i in S auftaucht.

Satz

$$|X(I)| = \prod_{i=1}^{n} \sum_{j \notin I} a_{ij}$$

Bestimmung von $X(I)$



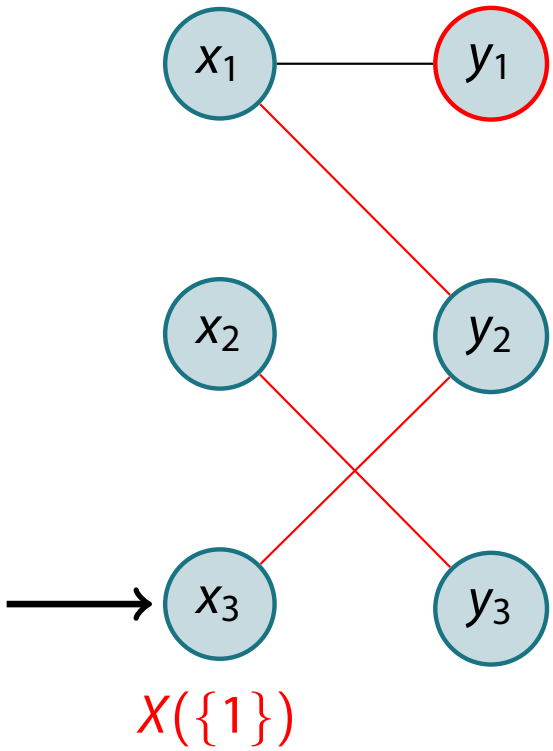
Beobachtung

Für $I \subseteq [n]$ ist $X(I)$ Menge aller S, so dass kein y_i in S auftaucht.

Satz

$$|X(I)| = \prod_{i=1}^{n} \sum_{j \notin I} a_{ij}$$

Bestimmung von $X(I)$



Beobachtung

Für $I \subseteq [n]$ ist $X(I)$ Menge aller S, so dass kein y_i in S auftaucht.

Satz

$$|X(I)| = \prod_{i=1}^{n} \sum_{j \notin I} a_{ij}$$

Algorithmus

Satz (Ryser 1963)

$$\operatorname{perm}(A) = \sum_{I \subseteq [n]} (-1)^{|I|} \prod_{i=1}^{n} \sum_{j \notin I} a_{ij}$$

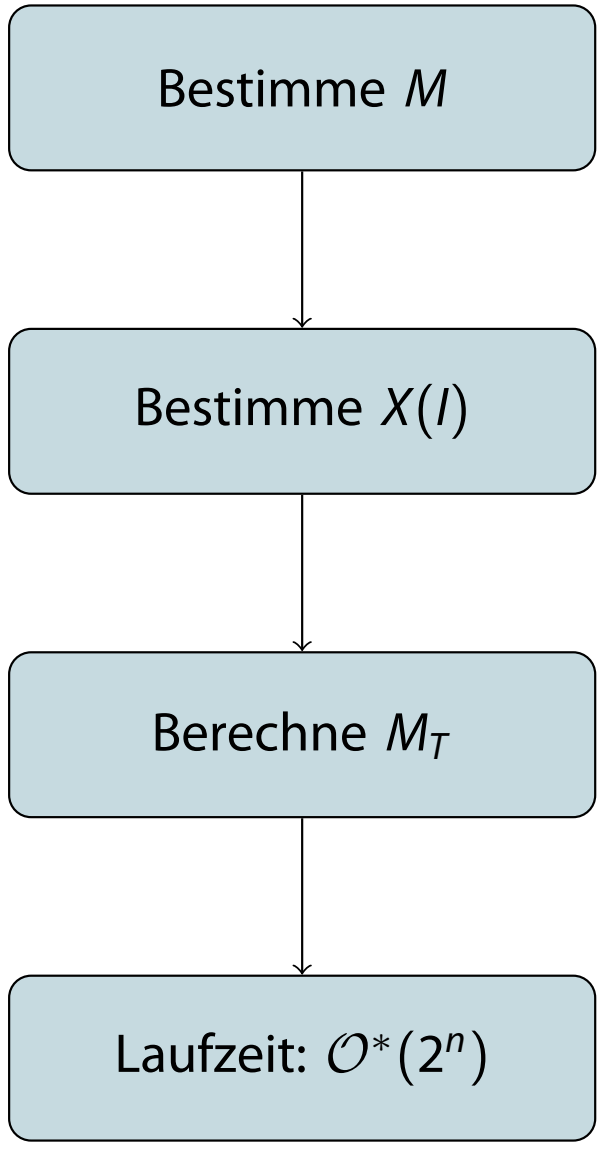
Algorithmus

Satz (Ryser 1963)

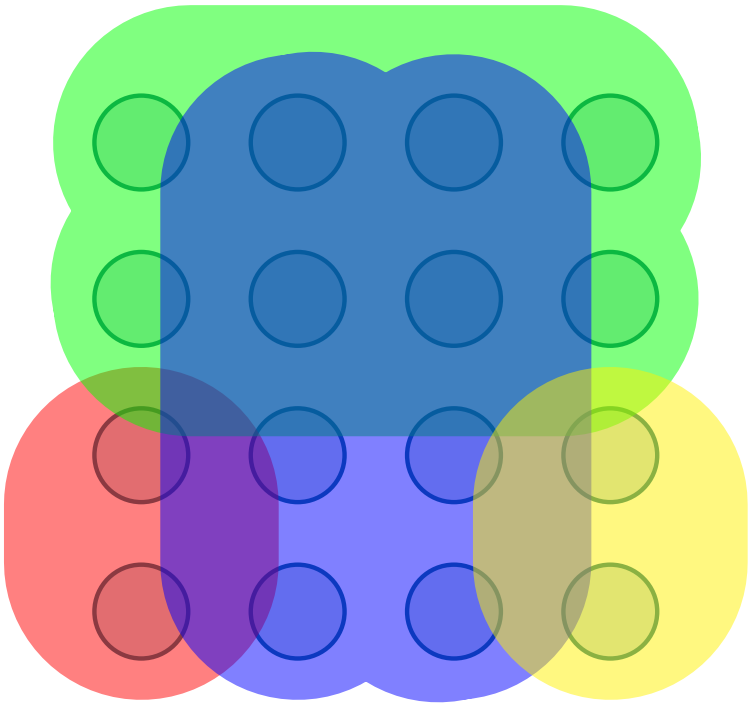
$$\operatorname{perm}(A) = \sum_{I \subseteq [n]} (-1)^{|I|} \prod_{i=1}^{n} \sum_{j \notin I} a_{ij}$$

Laufzeit: $\mathcal{O}^*(2^n)$

Zusammenfassung



Noch ein Beispiel: Überdeckungen



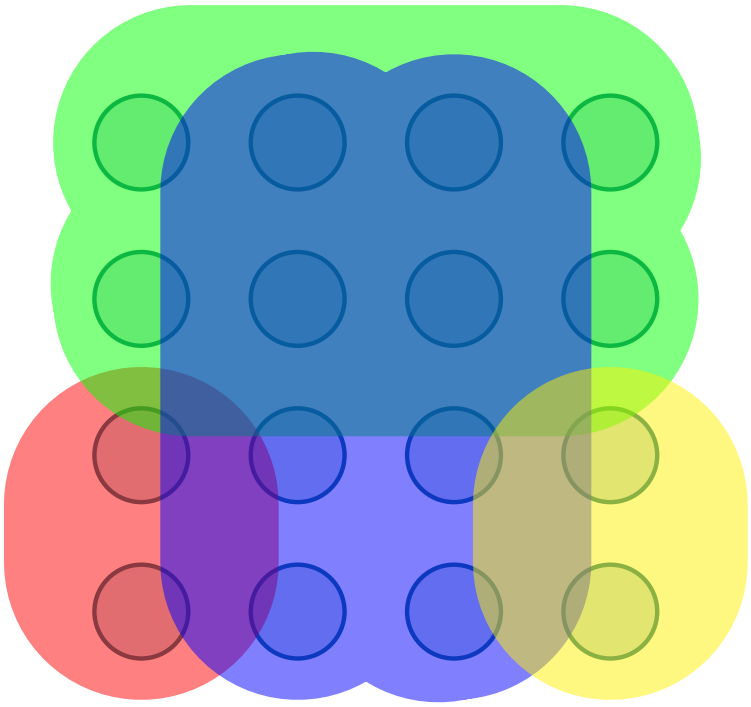
Problemstellung

k-Überdeckungsproblem

Gegeben *Universum* $\mathcal{U} = \{u_1, \ldots, u_n\}$,
Familie $\mathcal{S} = \{S_1, \ldots, S_m\}$ *mit*
$S_i \subseteq \mathcal{U}$

Gesucht $S_{i_1}, \ldots, S_{i_k}$ *mit* $\bigcup_{j=1}^{k} S_{i_j} = \mathcal{U}$

Noch ein Beispiel: Überdeckungen



Problemstellung

k-Überdeckungsproblem

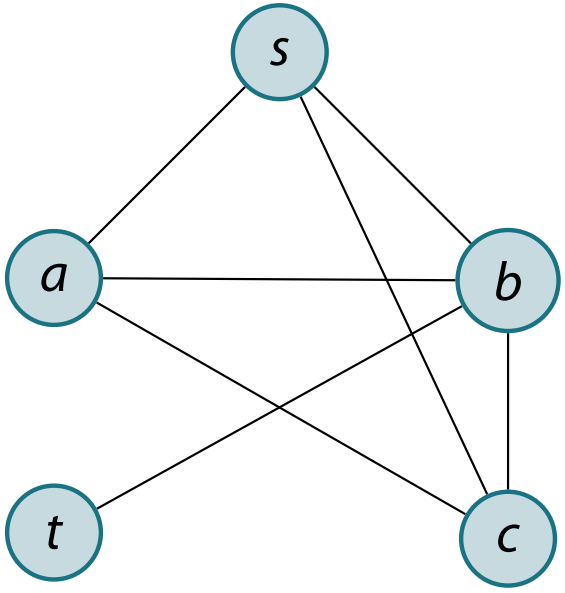
Gegeben *Universum* $\mathcal{U} = \{u_1, \ldots, u_n\}$, *Familie* $\mathcal{S} = \{S_1, \ldots, S_m\}$ *mit* $S_i \subseteq \mathcal{U}$

Gesucht $S_{i_1}, \ldots, S_{i_k}$ *mit* $\bigcup_{j=1}^{k} S_{i_j} = \mathcal{U}$

Satz (Karp 1971)

Das k-Überdeckungsproblem ist $\mathcal{NP}$*-schwer für* $k \geq 3$.

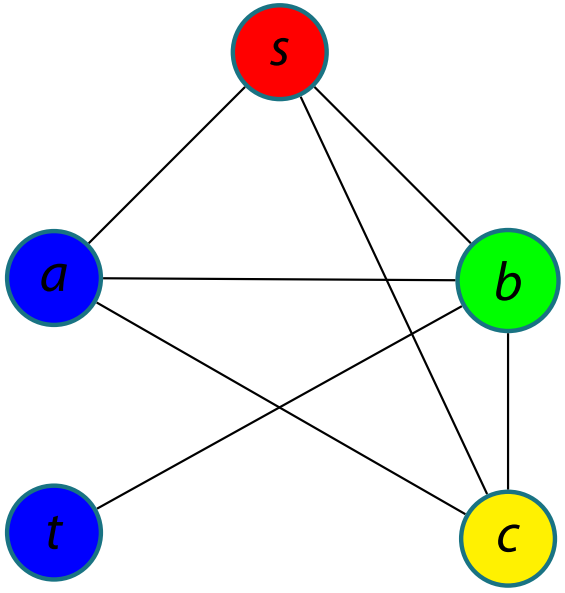
Graphen färben



Gegeben Graph $G = (V, E)$

Gesucht Färbung von G mit $\chi(G)$ Farben

Graphen färben



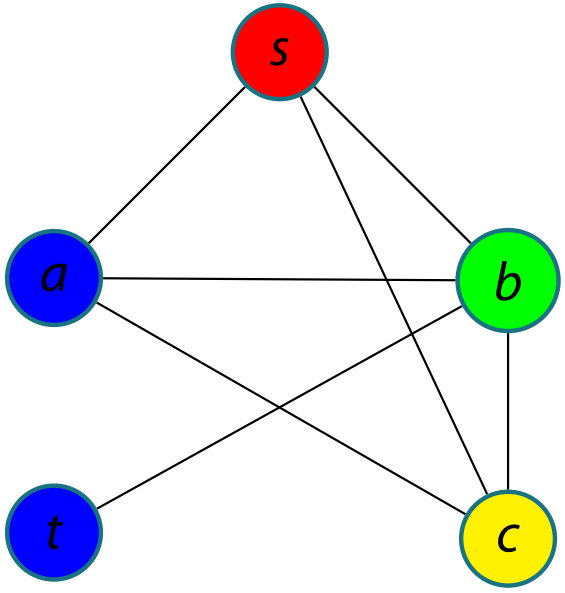
Gegeben Graph $G = (V, E)$

Gesucht Färbung von G mit $\chi(G)$ Farben

Satz

Die Bestimmung von $\chi(G)$ ist $\mathcal{NP}$-schwer.

Graphen färben



Gegeben Graph $G = (V, E)$

Gesucht Färbung von G mit $\chi(G)$ Farben

Satz

Die Bestimmung von $\chi(G)$ ist $\mathcal{NP}$-schwer.

Und nun?

Bestimme k-Überdeckung von $(V, \mathcal{IS})$

$\mathcal{IS}$ ist Menge aller unabhängigen Mengen

Lösungsansatz

Notation

- $c_k(\mathcal{U}, \mathcal{S}) = \#$ *geordneter* k-Überdeckungen von $(\mathcal{U}, \mathcal{S})$.

Lösungsansatz

Notation

- $c_k(\mathcal{U}, \mathcal{S}) = \#$ *geordneter* k-Überdeckungen von $(\mathcal{U}, \mathcal{S})$.
- Für $I \subseteq [n]$ ist $\mathcal{U}[I] = \bigcup_{i \in I} \{u_i\}$

Lösungsansatz

Notation

- $c_k(\mathcal{U}, \mathcal{S}) = \#$ *geordneter* k-Überdeckungen von $(\mathcal{U}, \mathcal{S})$.
- Für $I \subseteq [n]$ ist $\mathcal{U}[I] = \bigcup_{i \in I} \{u_i\}$

Lösungsansatz

Notation

- $c_k(\mathcal{U}, \mathcal{S}) = \#$ *geordneter* k-Überdeckungen von $(\mathcal{U}, \mathcal{S})$.
- Für $I \subseteq [n]$ ist $\mathcal{U}[I] = \bigcup_{i \in I} \{u_i\}$

Beobachtung

k-Überdeckungen von $(\mathcal{U}, \mathcal{S})$ ist $\frac{c_k(\mathcal{U},\mathcal{S})}{k!}$

Lösungsansatz

- Definiere $M = \{O_1, \ldots, O_t\}$

Lösungsansatz

- Definiere $M = \{O_1, \ldots, O_t\}$
- $O_i \in \mathcal{S}^k$

Lösungsansatz

- Definiere $M = \{O_1, \ldots, O_t\}$
- $O_i \in \mathcal{S}^k$
- $P_j(O) = T \Leftrightarrow u_j \in \bigcup O$

Lösungsansatz

- Definiere $M = \{O_1, \ldots, O_t\}$
- $O_i \in \mathcal{S}^k$
- $P_j(O) = T \Leftrightarrow u_j \in \bigcup O$
- O ist k-Überdeckung $\Leftrightarrow \bigwedge_{j=1}^{n} P_j(O) = T$

Bestimmung von $X(I)$

Beobachtung

Für $I \subseteq [n]$ ist $X(I)$ Menge aller O, so dass kein u_i in $\bigcup O$ auftaucht.

Bestimmung von $X(I)$

Beobachtung

Für $I \subseteq [n]$ ist $X(I)$ Menge aller O, so dass kein u_i in $\bigcup O$ auftaucht.

Berechnung von $|X(I)|$

- $g_j(I) = \#$ von $S \in \mathcal{S}$ mit $S \subseteq \mathcal{U} \setminus \mathcal{U}[I]$und $\mathcal{U}[j+1..n] \setminus \mathcal{U}[I] \subseteq S$

Bestimmung von $X(I)$

Beobachtung

Für $I \subseteq [n]$ ist $X(I)$ Menge aller O, so dass kein u_i in $\bigcup O$ auftaucht.

Berechnung von $|X(I)|$

- $g_j(I) = \#$ von $S \in \mathcal{S}$ mit $S \subseteq \mathcal{U} \setminus \mathcal{U}[I]$und $\mathcal{U}[j+1..n] \setminus \mathcal{U}[I] \subseteq S$
- $g_n(I) = \#$ von $S \in \mathcal{S}$ mit $S \subseteq \mathcal{U} \setminus \mathcal{U}[I]$

Bestimmung von $X(I)$

Beobachtung

Für $I \subseteq [n]$ ist $X(I)$ Menge aller O, so dass kein u_i in $\bigcup O$ auftaucht.

Berechnung von $|X(I)|$

- $g_j(I) = \#$ von $S \in \mathcal{S}$ mit $S \subseteq \mathcal{U} \setminus \mathcal{U}[I]$und $\mathcal{U}[j+1..n] \setminus \mathcal{U}[I] \subseteq S$
- $g_n(I) = \#$ von $S \in \mathcal{S}$ mit $S \subseteq \mathcal{U} \setminus \mathcal{U}[I]$
- $|X(I)| = g_n(I)^k$

Berechnung der $g_j(I)$

Dynamische Programmierung

- $g_0(I) \approx \mathsf{lst}\,\mathcal{U} \setminus \mathcal{U}[I] \in \mathcal{S}$?

Berechnung der $g_j(I)$

Dynamische Programmierung

- $g_0(I) \approx \mathsf{lst}\,\mathcal{U} \setminus \mathcal{U}[I] \in \mathcal{S}$?
-

$$g_{j+1}(I) = \begin{cases} g_j(I) & j \in I \\ g_j(I \cup \{j+1\}) + g_j(I) & \text{sonst} \end{cases}$$

Berechnung der $g_j(I)$

Dynamische Programmierung

- $g_0(I) \approx \mathsf{lst}\,\mathcal{U} \setminus \mathcal{U}[I] \in \mathcal{S}$?
-

$$g_{j+1}(I) = \begin{cases} g_j(I) & j \in I \\ g_j(I \cup \{j+1\}) + g_j(I) & \text{sonst} \end{cases}$$

Berechnung der $g_j(I)$

Dynamische Programmierung

- $g_0(I) \approx \mathsf{Ist}\,\mathcal{U} \setminus \mathcal{U}[I] \in \mathcal{S}$?
-

$$g_{j+1}(I) = \begin{cases} g_j(I) & j \in I \\ g_j(I \cup \{j+1\}) + g_j(I) & \text{sonst} \end{cases}$$

Alle $g_j(I)$ berechenbar in Zeit und Platz $\mathcal{O}^*(2^n)$

Algorithmus

Satz (Björklund & Husfeldt, Koivisto 2006)

Anzahl k-Überdeckungen berechenbar in Zeit und Platz $\mathcal{O}^*(2^n)$