Technische Universität Braunschweig



Algorithmen und Datenstrukturen 2

Arne Schmidt

Recap

1. Ziffer: Man muss Objekt 1 oder 2 wählen, aber nicht beide.

2. Ziffer: Man muss Objekt 3 oder 4 wählen, aber nicht beide.

3. Ziffer: Man muss Objekt 5 oder 6 wählen, aber nicht beide.

4. Ziffer: Man muss Objekt 1, 3 oder 6 wählen.

5. Ziffer: Man muss Objekt 1, 4 oder 6 wählen.

6. Ziffer: Man muss Objekt 2, 3 oder 5 wählen.

$\rightarrow (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$

$n = 12,\ z_i = p_i,\ Z = 111444$

$$
\begin{aligned}
z_1 = p_1 &= 100110 \\
z_2 = p_2 &= 100001 \\
z_3 = p_3 &= 10101 \\
z_4 = p_4 &= 10010 \\
z_5 = p_5 &= 1001 \\
z_6 = p_6 &= 1110 \\
z_7 = p_7 &= 200 \\
z_8 = p_8 &= 100 \\
z_9 = p_9 &= 20 \\
z_{10} = p_{10} &= 10 \\
z_{11} = p_{11} &= 2 \\
z_{12} = p_{12} &= 1
\end{aligned}
$$

Kapitel 5.3 – 3SAT

3SAT - Definition

Definition 5.6
Gegeben:
- n Boolesche Variablen $x_1, \dots, x_n$ aus denen wir Literale ℓ_i der Form x_k oder $\bar{x}_k$ bilden können.
- m Klauseln, jede zusammengesetzt aus genau drei Literalen $C_j = (\ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3})$ mit $1 \leq j \leq m$

Gesucht:
Eine alle m Klauseln erfüllende (engl. satisfying) Wahrheitsbelegung der n Variablen.

Beispiel:

$$(x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee x_3 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4)$$

Erfüllende Belegung: $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0$

Nicht-erfüllende Belegung: $x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 0$

Beobachtung 5.7



Wenn man mir eine erfüllende Belegung gibt, kann ich das leicht verifizieren…

Wenn es keine gibt, ist das nicht so einfach nachzuweisen…

Milleniumsfrage

Problem 5.8
Gibt es für 3SAT einen Algorithmus, der für jede Instanz I in polynomieller Laufzeit in n und m entscheidet, ob I erfüllbar ist?

Oder anders ausgedrückt: Ist 3SAT $\in P$?

Satz 5.9
Ist 0-1 Knapsack $\in P$, dann gilt 3SAT $\in P$.

Beweisidee:
Reduziere 3SAT auf 0-1 Knapsack ähnlich zur letzten Vorlesung:
Für jede Variable konstruiere große Zahlen in deren die letzten m Ziffern ausdrücken ob die Variable als positives/negatives Literal in der entsprechenden Klausel vorkommt.
Für jede Klausur erstelle noch Auffüllobjekte, um zu prüfen, ob die Klausel erfüllt wurde.

Reduktion 3SAT auf Knapsack in Pseudocode

$$
\begin{aligned}
z_1 = p_1 &= 100110\\
z_2 = p_2 &= 100001\\
z_3 = p_3 &= 10101\\
z_4 = p_4 &= 10010\\
z_5 = p_5 &= 1001\\
z_6 = p_6 &= 1110\\
z_7 = p_7 &= 200\\
z_8 = p_8 &= 100\\
z_9 = p_9 &= 20\\
z_{10} = p_{10} &= 10\\
z_{11} = p_{11} &= 2\\
z_{12} = p_{12} &= 1
\end{aligned}
$$

Algorithmus 5.10 3SATtoKnapsack

Eingabe: Instanz I_{3SAT} von 3SAT

Ausgabe: Instanz I_{KN} von 0-1-KNAPSACK

1: $Z := \sum_{i=0}^{n-1} 10^{m+i} + \sum_{i=0}^{m-1} 4 \cdot 10^i$
2: $P := Z$
3: Erstelle Objekte $z_1, \ldots, z_{2n+2m}$ und $p_1, \ldots, p_{2n+2m}$
4: **for** $i = 1$ **to** n **do** ▷ Initialisiere Gewichte für Variablen
5: $\quad p_{2i-1} := z_{2i-1} := 10^{n+m-i}$
6: $\quad p_{2i} := z_{2i} := 10^{n+m-i}$
7: **for** $j = 1$ **to** m **do** ▷ Betrachte jede Klausel
8: $\quad$ **for** $i = 1$ **to** n **do** ▷ Setze Bit von Variablen, wenn in Klausel präsent
9: $\quad\quad$ **if** x_i taucht als positives Literal in C_j auf **then**
10: $\quad\quad\quad p_{2i-1} := z_{2i-1} := z_{2i-1} + 10^{m-j}$
11: $\quad\quad$ **else if** x_i taucht als negatives Literal in C_j auf **then**
12: $\quad\quad\quad p_{2i} := z_{2i} := z_{2i} + 10^{m-j}$
13: $\quad p_{2n+2j-1} := z_{2n+2j-1} := 2 \cdot 10^{m-j}$ ▷ Gewichte um auf 4 aufzufüllen,
14: $\quad p_{2n+2j} := z_{2n+2j} := 1 \cdot 10^{m-j}$ ▷ wenn Klausel erfüllt wurde.
15: **return**

Reduktion 3SAT auf Knapsack in Pseudocode

$$(x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$

$$n = 12,\ z_i = p_i,\ Z = 111444$$

$$\begin{aligned}
z_1 = p_1 &= 100110 \\
z_2 = p_2 &= 100001 \\
z_3 = p_3 &= 10101 \\
z_4 = p_4 &= 10010 \\
z_5 = p_5 &= 1001 \\
z_6 = p_6 &= 1110 \\
z_7 = p_7 &= 200 \\
z_8 = p_8 &= 100 \\
z_9 = p_9 &= 20 \\
z_{10} = p_{10} &= 10 \\
z_{11} = p_{11} &= 2 \\
z_{12} = p_{12} &= 1
\end{aligned}$$

3SAT und Knapsack

Satz 5.11
Wenn 3SAT $\in P$, dann gilt $P = NP$.

Beweisidee:
Zeige: Jede Instanz eines Problems in NP lässt sich als äquivalente 3SAT-Instanz codieren, welches höchstens polynomiell größer ist.

Korollar 5.12
Wenn 0-1 Knapsack $\in P$, dann gilt $P = NP$.

Folgt aus der Reduktion 3SAT auf 0-1 Knapsack, sowie der Reduktion von allen Problem aus NP auf 3SAT.

NP-schwer und -Vollständigkeit

Definition 5.13

1. Ein Problem Π heißt NP-schwer, wenn eine Polynomialzeitreduktion von jedem NP-Problem auf Π existiert.
2. Ein Problem Π heißt NP-vollständig, wenn $\Pi \in NP$ und Π NP-schwer ist.

Bemerkung:
Für den ersten Teil genügt es, ein bereits bekanntes NP-schweres Problem auf Π zu reduzieren!

Korollar 5.14

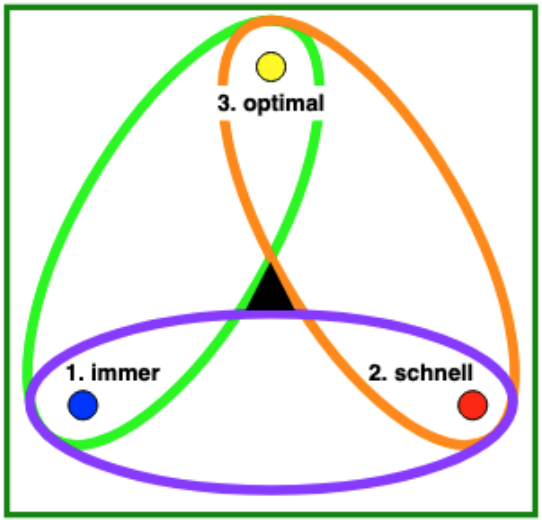
1. 3SAT ist NP-vollständig
2. 0-1 Knapsack ist NP-vollständig
3. Maximum Knapsack ist NP-schwer

Frage: Warum ist nicht klar, ob Maximum Knapsack in NP liegt?

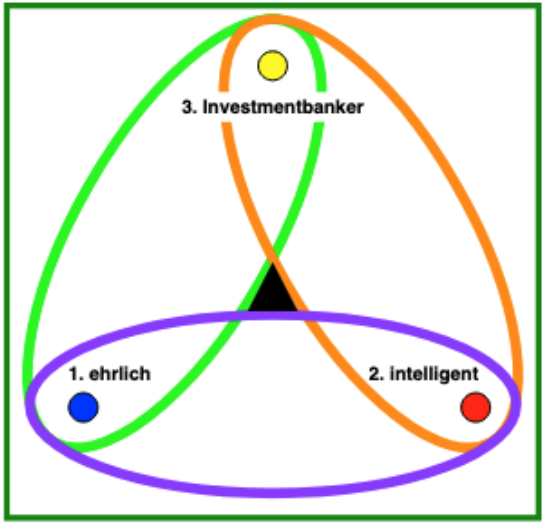
Kapitel 5.5 – Konsequenzen

Algorithmen vs. Investmentbanker

Ziele: Ein “perfekter” Algorithmus liefert…
1. immer
2. schnell
3. eine optimale Lösung.



Ziele: Ein “perfekter” Finanzberater ist ein…
1. ehrlicher
2. intelligenter
3. Investmentbanker.

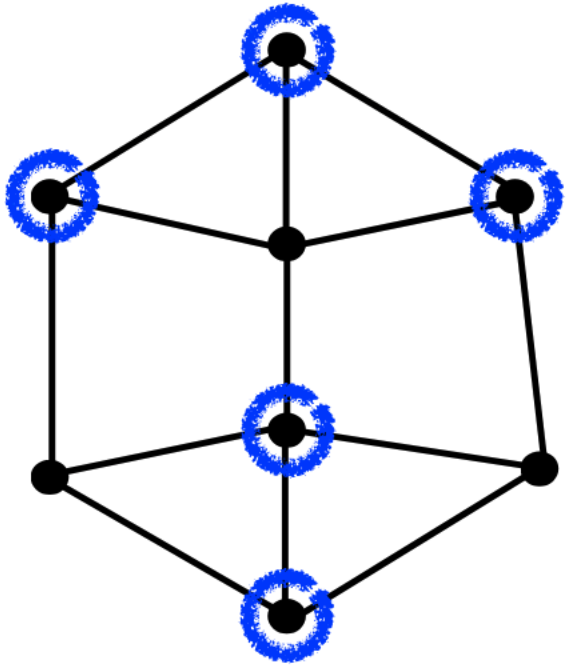


Kapitel 5.6 – Graphenprobleme

Vertex Cover

Problem Vertex Cover (Optimierungsvariante)
Gegeben:
- Ein Graph $G = (V, E)$

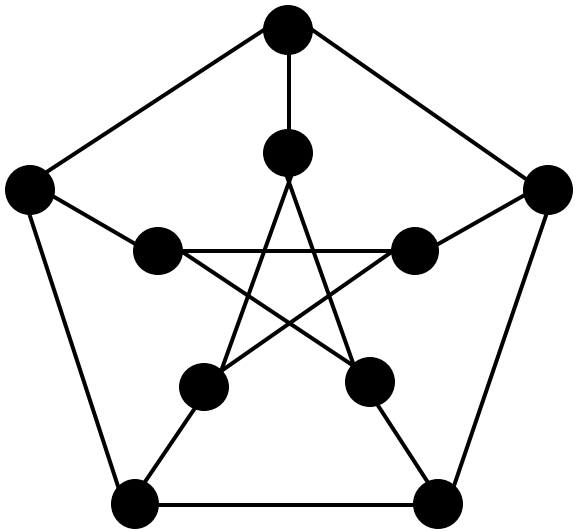
Gesucht:
- Knotenmenge $S \subseteq V$ möglichst kleiner Kardinalität, sodass $u \in S$ oder $v \in S$ für jedes $e = \{u, v\} \in E$ gilt.



Für die Entscheidungsvariante ist zusätzlich ein $k \in \mathbb{N}$ gegeben, und man fragt, ob es ein Vertex Cover der Größe höchstens k gibt.

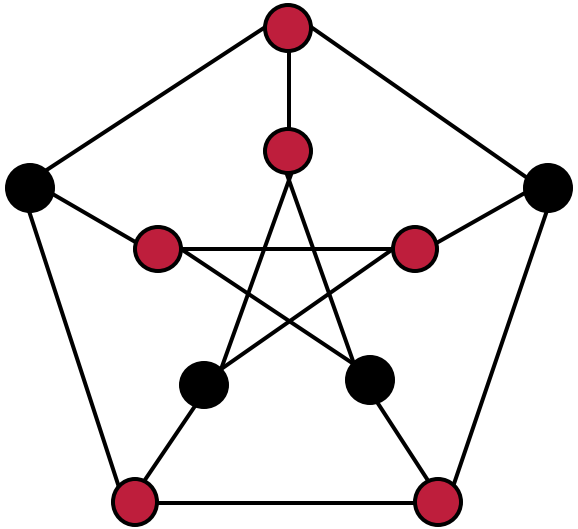
Eine Lösung können wir auch schnell überprüfen!
→ Vertex Cover ist in NP.

Petersengraph



Welche Größe hat ein kleinstes Vertex Cover?

Petersengraph

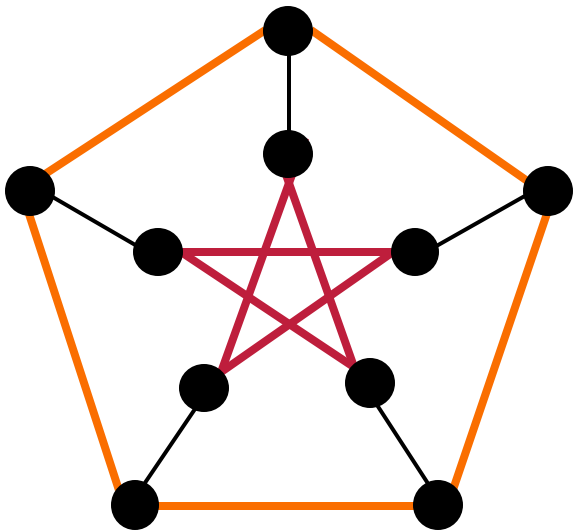


Welche Größe hat ein kleinstes Vertex Cover?

Wir finden schnell eines der Größe 6.



Petersengraph



Welche Größe hat ein kleinstes Vertex Cover?

- Zwei unabhängige Kreise der Länge 5 (rot und orange)
- Schnell zu sehen: Ein Kreis der Länge n benötigt $\left\lceil \frac{n}{2} \right\rceil$ Knoten im Vertex Cover.
- Roter und Orangener Kreis benötigen je 3 Knoten.
- Gesamter Graph benötigt also mindestens 6.

Das kleinste Vertex Cover besitzt also 6 Knoten.

Verschiedene Graphenklassen

	Pfad	Kreise	Bäume	Vollständige Graphen	Beliebig
Größe VC	$\left\lceil \frac{n-1}{2} \right\rceil$	$\left\lceil \frac{n}{2} \right\rceil$	?	$n-1$	?
Polyzeit Algorithmus?	Ja!	Ja!	Ja!	Ja!	?

Frage: Wie schwer ist es zu entscheiden, ob es ein Vertex Cover der Größe $\leq k$ gibt?

Versuchen wir von 3SAT auf Vertex Cover zu reduzieren!

Überlegungen

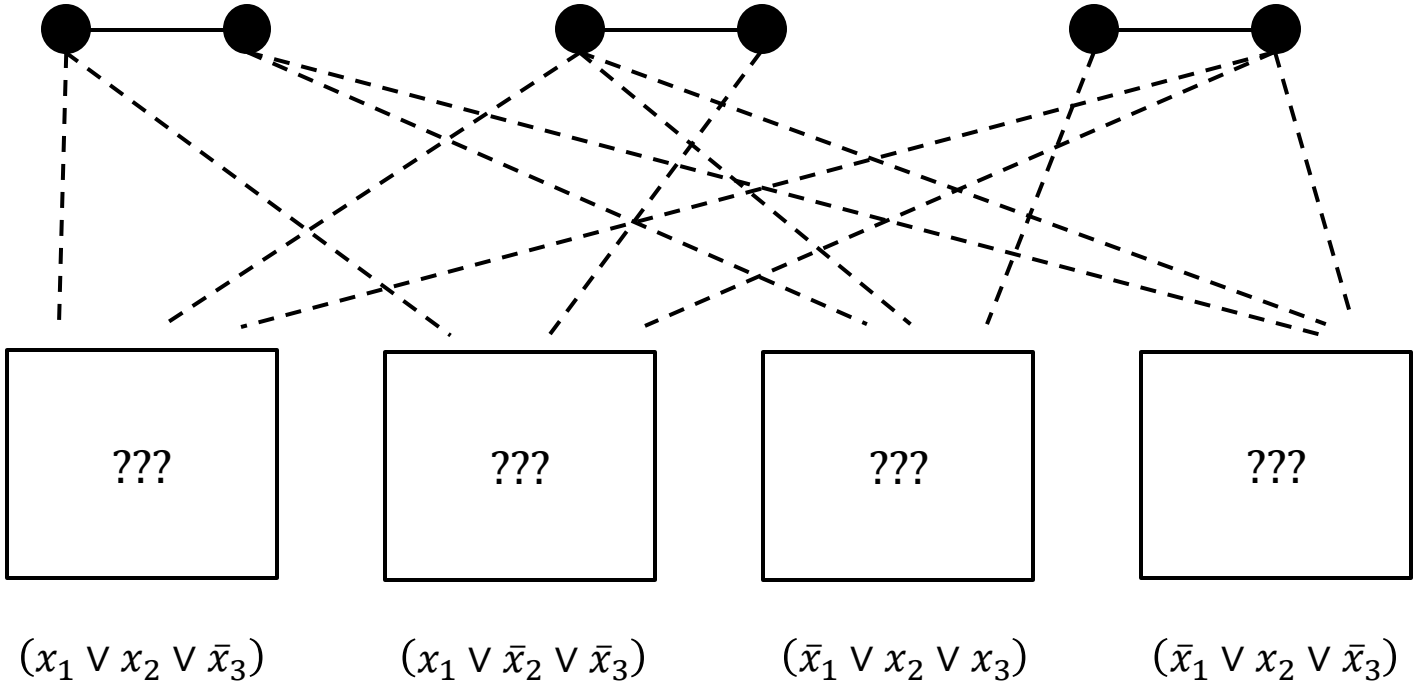


Für Variablen benötigen wir ein **Gadget**, welches die Auswahl (true/false) simuliert.

Für Klauseln benötigen wir ein **Gadget**, in welchem eins von drei Literalen true sein soll.

Wie codieren wir die Variablen-Klauseln-Logik?

Reduktion 3SAT auf Vertex Cover

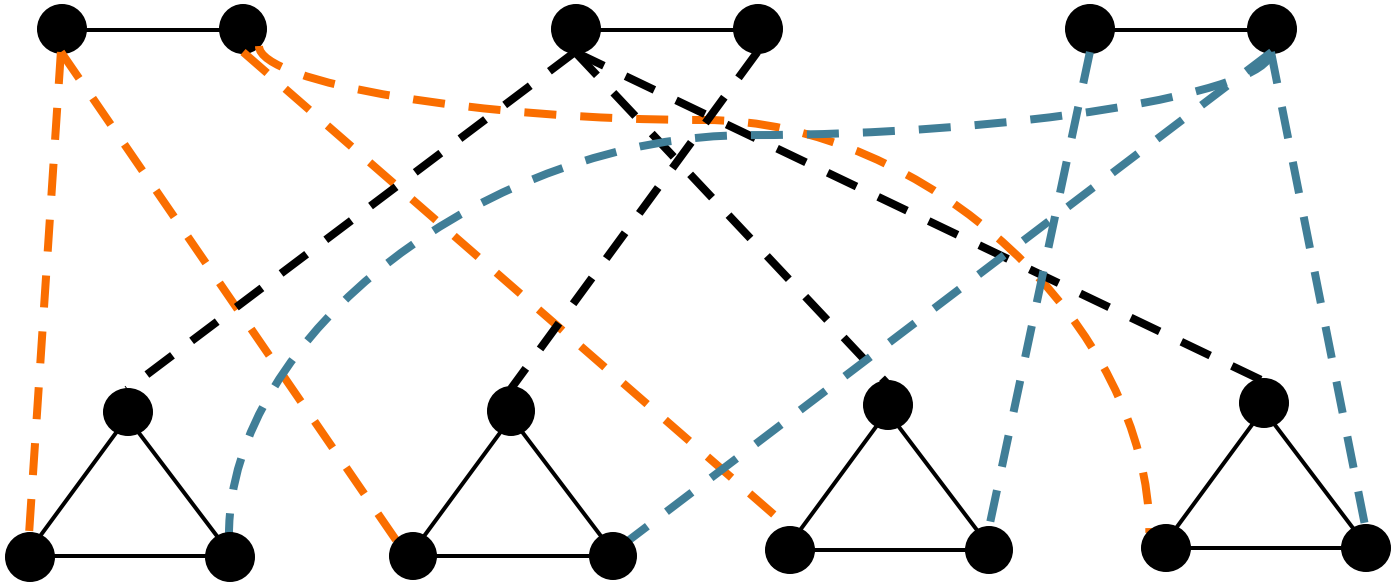


Variablen-gadgets

Variablen-Klausel-Logik

Klausel-gadgets

Reduktion 3SAT auf Vertex Cover



$(x_1 \vee x_2 \vee \bar{x}_3)$ $(x_1 \vee \bar{x}_2 \vee \bar{x}_3)$ $(\bar{x}_1 \vee x_2 \vee x_3)$ $(\bar{x}_1 \vee x_2 \vee \bar{x}_3)$

Variablen-gadgets

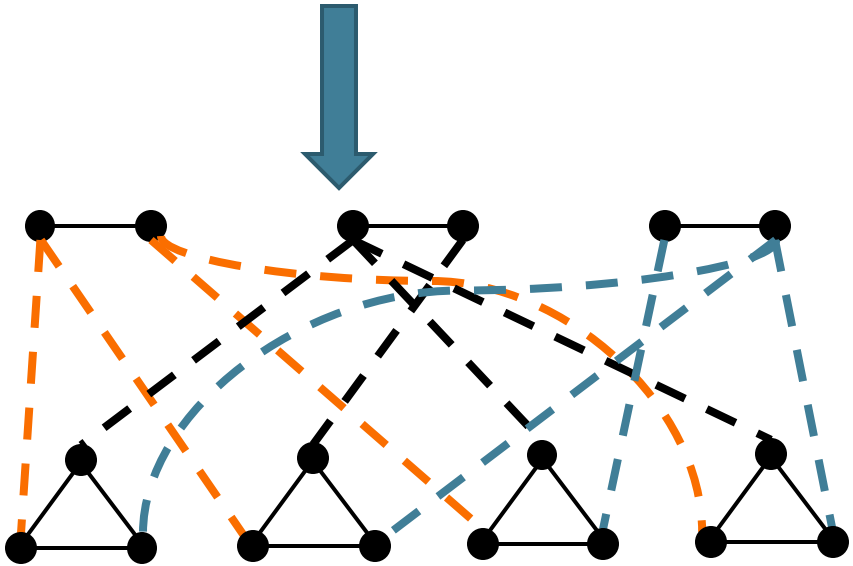
Variablen-Klausel-Logik

Klausel-gadgets

Reduktion 3SAT auf Vertex Cover

$\varphi := (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$

G



φ ist genau dann erfüllbar, wenn G ein Vertex Cover der Größe 11 besitzt.

Reduktion im Allgemeinen

Konstruiere den Graph wie folgt:

- Für jede Variable x_i erstelle zwei Knoten $v_{i,0}$ und $v_{i,1}$
- Für jede Klausel $C_j = (\ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3})$ erstelle drei Knoten $w_{j,1}, w_{j,2}, w_{j,3}$

- Verbinde Kanten entsprechend:
  - $v_{i,0}$ mit $v_{i,1}$
  - $w_{j,1}, w_{j,2}, w_{j,3}$ vollständig miteinander
  - $w_{j,k}$ mit $v_{i,0}$ falls die i-te Variable **negativ** in Klausel j als k-tes Literal auftaucht
  - $w_{j,k}$ mit $v_{i,1}$ falls die i-te Variable **positiv** in Klausel j als k-tes Literal auftaucht

Damit ist recht offensichtlich:
Die Reduktion benötigt polynomielle Laufzeit (und Platz).
Wir müssen nun noch zeigen, dass die Reduktion korrekt ist!

Korrektheit Reduktion

Satz
Sei φ eine 3SAT-Formel mit n Variablen und m Klauseln und $G_\varphi = (V, E)$ der aus der Reduktion entstehende Graph. Dann gilt:

$$\varphi \text{ ist erfüllbar} \Leftrightarrow G_\varphi \text{ besitzt Vertex Cover der Größe} \leq n + 2m$$

Resultate

Satz
Vertex Cover ist NP-vollständig.

Satz
Es existiert ein 2-Approximationsalgorithmus für Vertex Cover.

Beweis: Siehe Übung

Nächste Woche

