Complexiteit

College 9

Docent: Lieuwe Vinkhuijzen

Mei 2020

Nieuws

- Skype: lieuwe.vinkhuijzen1
- Vragen: `complexiteitleiden@gmail.com`

Vorige keer

1. NP-Hard, NP-Volledig
2. Cook-Levin Stelling: NP-Volledige problemen bestaan
3. Algoritme van Savitch

Deze week

1. Reducties

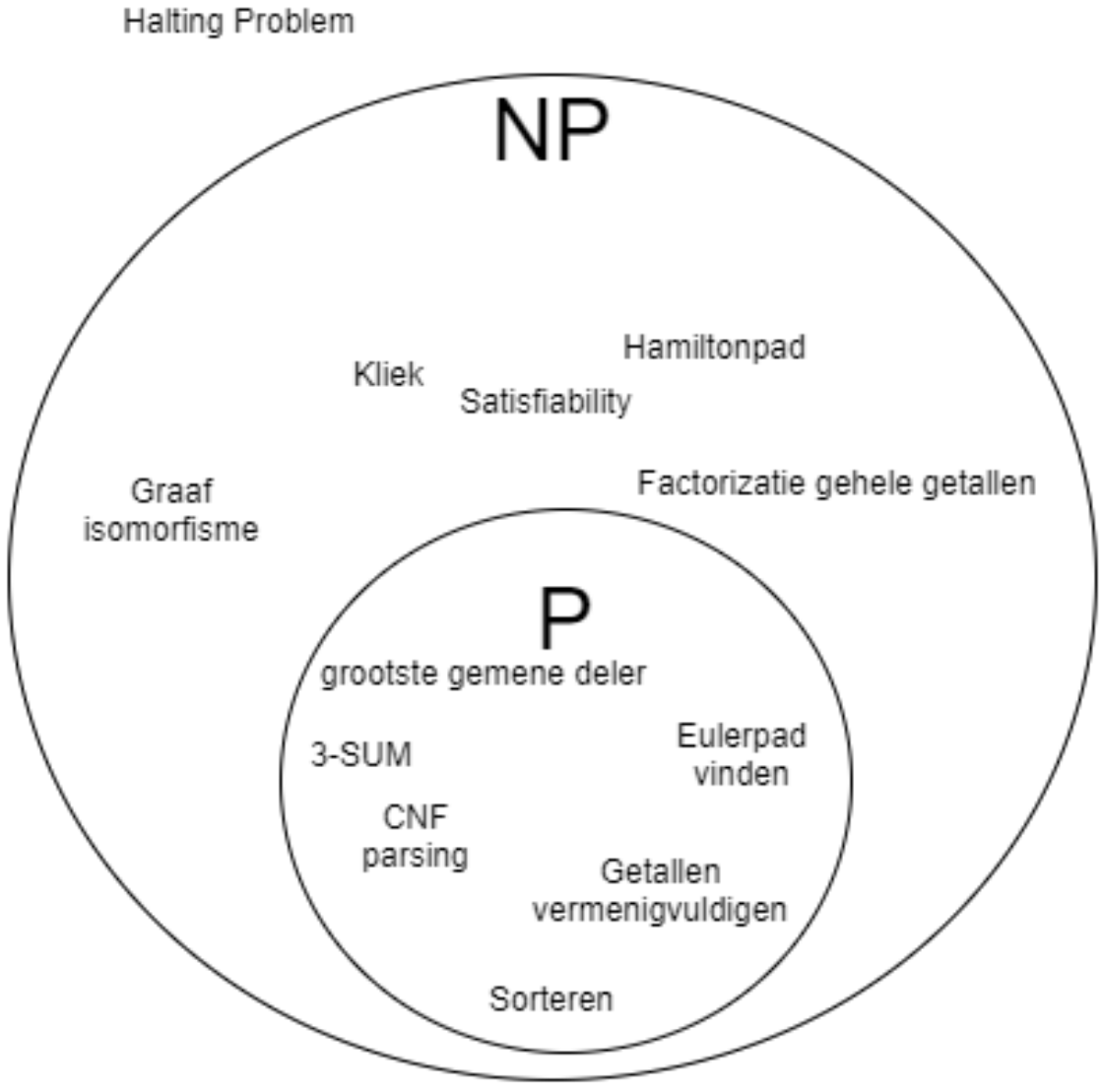
Volgende week

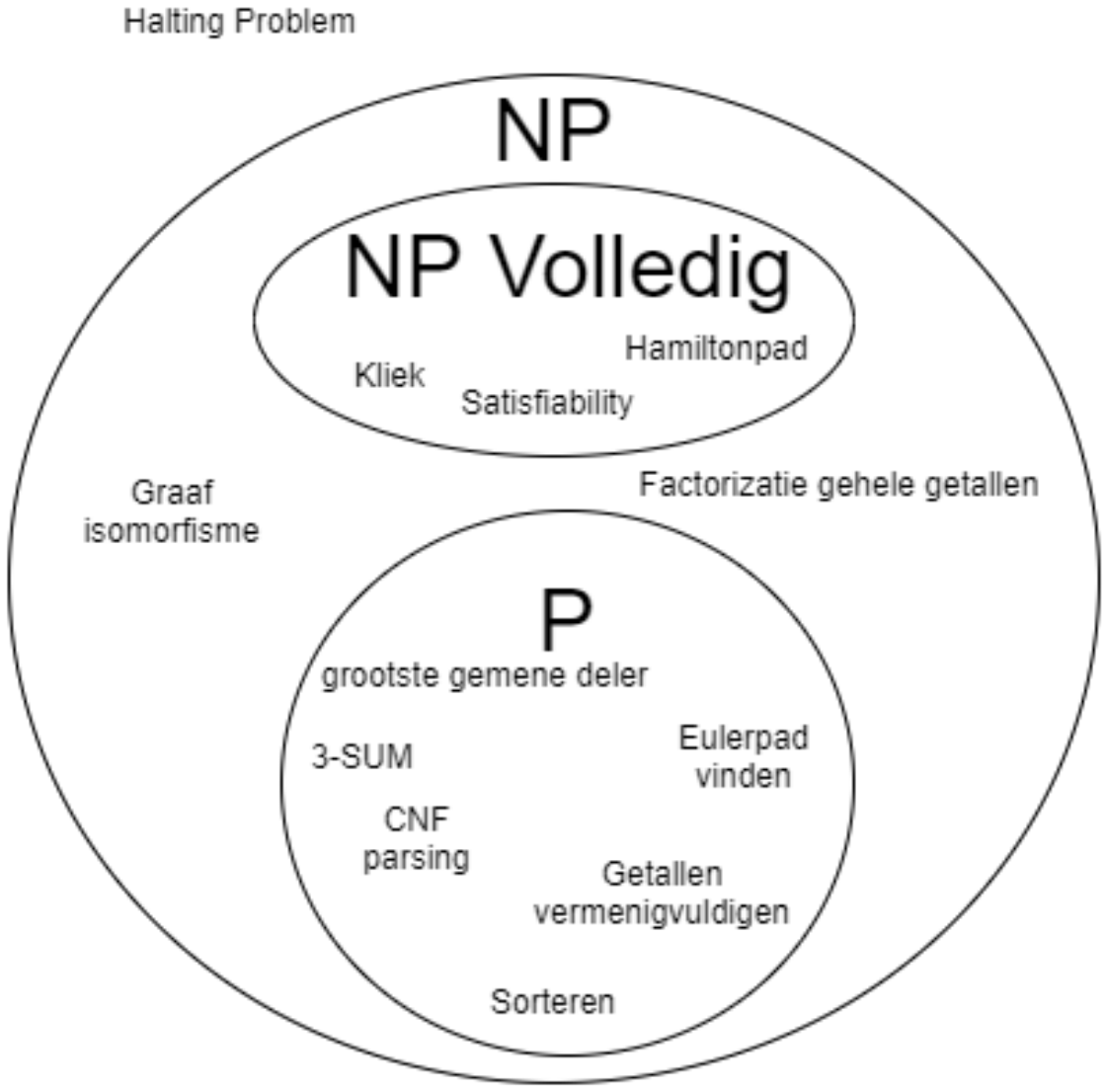
1. Analyse tijdsgebruik recursieve algoritmes

Klassificatie van talen tot nu toe

Taal: Een verzameling $L \subseteq \{0,1\}^*$.

$$\underbrace{\textsc{Regular} \subsetneq \text{CFL}}_{\text{FI 1+2}} \subsetneq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} = \text{NPSPACE} \subseteq \text{EXP} \subsetneq \underbrace{\text{R} \subsetneq \text{RE}}_{\text{FI 3}}$$


Halting Problem
NP
Hamiltonpad
Kliek
Satisfiability
Graaf
isomorfisme
Factorizatie gehele getallen
P
grootste gemene deler
Eulerpad
vinden
3-SUM
CNF
parsing
Getallen
vermenigvuldigen
Sorteren


Halting Problem
NP
NP Volledig
Hamiltonpad
Kliek
Satisfiability
Graaf
isomorfisme
Factorizatie gehele getallen
P
grootste gemene deler
3-SUM
Eulerpad
vinden
CNF
parsing
Getallen
vermenigvuldigen
Sorteren

Reductie: Definitie

Definition (Reductie)

Zij K, L beslissingsproblemen. Een *reductie* van K naar L is een functie f die inputs $x \in K_{ja} \cup K_{nee}$ neemt, en een output $y = f(x) \in L_{ja} \cup L_{nee}$ geeft, zodanig dat

1. Als $x \in K_{ja}$ dan $f(x) \in L_{ja}$
2. Als $x \in K_{nee}$ dan $f(g) \in L_{nee}$

Schrijf: $K \leq L$

Definition (Polynomiale tijd reductie)

f is een *polynomiale tijd reductie* als f een reductie is, en het algoritme dat f berekent heeft polynomiale tijdscomplexiteit.

Schrijf: $K \leq_P L$.

Reducties: Aanleiding

Theorem

Als $K \leq_P L$, en als er een algoritme is dat L oplost in tijd $T_L(|x|)$, dan is er polynoom p, en een algoritme A, z.d.d. A het probleem K oplost in tijd $T_L(p(|x|))$.

```
1: procedure SATISFIABILITY(φ)
2:     (G, k) := REDUCTIE-SAT-NAAR-KLIEK(φ)
3:     answer := SOLVE-KLIEK(G, k)
4:     return answer
5: end procedure
```

Corollary

Als $K \leq_P L$, dan is er een polynomiaal algoritme voor K als er een polynomiaal algoritme is voor K.

Corollary

Als $K \leq_P L$ en $L \leq_P K$ dan is er een polynomiaal algoritme voor beide, of voor geen van beide.

Reducties: Aanleiding

Theorem (Cook-Levin Stelling)

Elke taal $L \in \mathrm{NP}$ *reduceert naar* SATISFIABILITY.

Corollary

Als SATSIFIABILITY *een polynomiaal algoritme heeft, dan heeft elk probleem in* NP *een polynomiaal algoritme.*

Corollary

Als SATISFIABILITY *een polynomiaal algoritme heeft, dan geldt* $\mathrm{P} = \mathrm{NP}$.

Reducties: Aanleiding

Corollary

Als $K \leq_P L$ en $L \leq_P K$ dan is er een polynomiaal algoritme voor beide, of voor geen van beide.

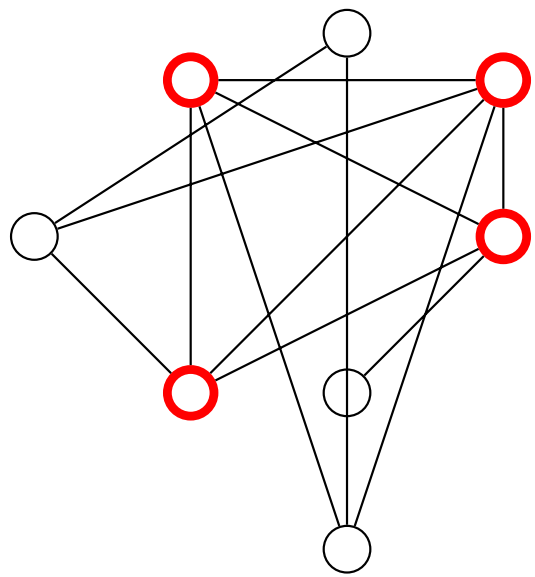
Corollary

Als K en L allebei NP-Volledig zijn, dan is er een polynomiaal algoritme voor beide, of voor geen van beide.

> *We give theorems which strongly suggest, but do not imply, that these problems, as well as many others, will remain intractable perpetually*
>
> Richard Karp, 1971

3-Satisfiability $\leq_P$ Kliek

Input: Een ongerichte graaf $G = (V, E)$ en een getal $k \geq 1$.
Output: Heeft de graaf een kliek van k knopen? Een verzameling $S \subseteq V$ is een kliek als $\forall u \in S, v \in S : (u, v) \in E$.

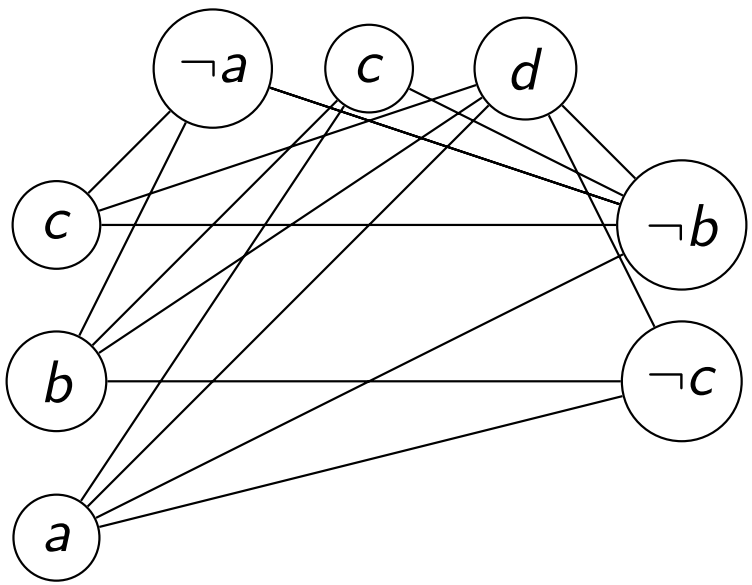


Figuur: Voorbeeld: Een graaf met een kliek van 4 knopen.

3-Satisfiability $\leq_P$ Kliek

Reductie: We maken van een formule, een graaf die een misschien een kliek heeft. Twee variabelen in verschillende clausulen krijgen een tak, tenzij het gaat om x en $\neg x$.

$$\psi = (a \vee b \vee c) \wedge (\neg a \vee c \vee d) \wedge (\neg b \vee \neg c)$$

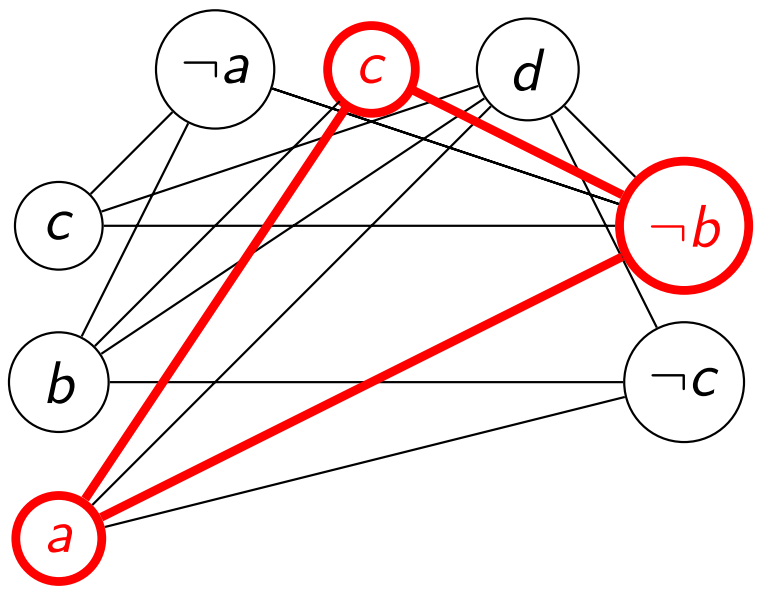


Figuur: De resulterende graaf met een kliek.

3-Satisfiability $\leq_P$ Kliek

We maken van een formule, een graaf die een misschien een kliek heeft.

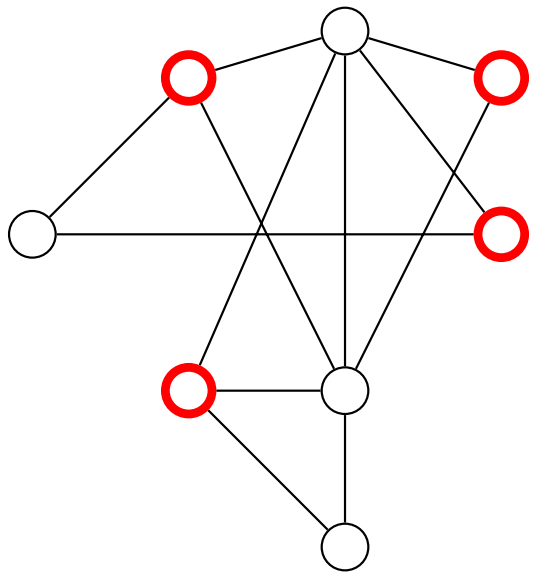
$$\psi = (a \vee b \vee c) \wedge (\neg a \vee c \vee d) \wedge (\neg b \vee \neg c)$$



Figuur: De resulterende graaf met een kliek.

Independent Set

Input: Een graaf $G = (V, E)$ en een getal k
Output: "Ja" desda de graaf een *Independent Set* heeft van grootte k. Een Independent Set is een verzameling vertices $I \subseteq V$ zodanig dat voor alle $i, j \in I : (i, j) \notin E$.

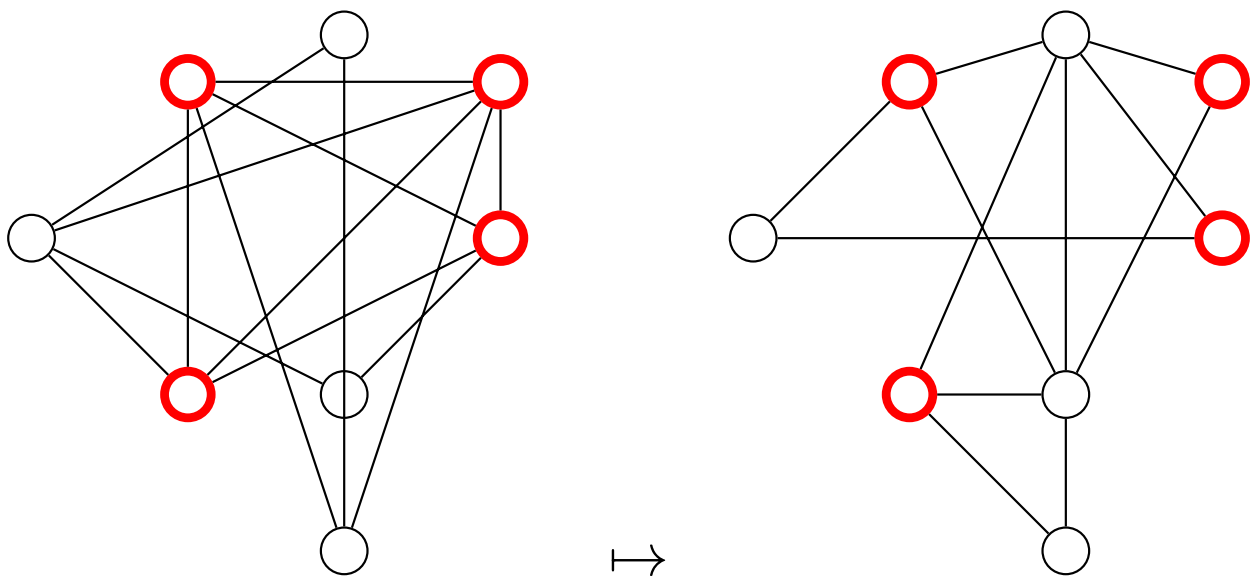


Figuur: Voorbeeld: De rode vertices vormen een independent set.

Kliek $\leq_{\mathrm{P}}$ Independent Set

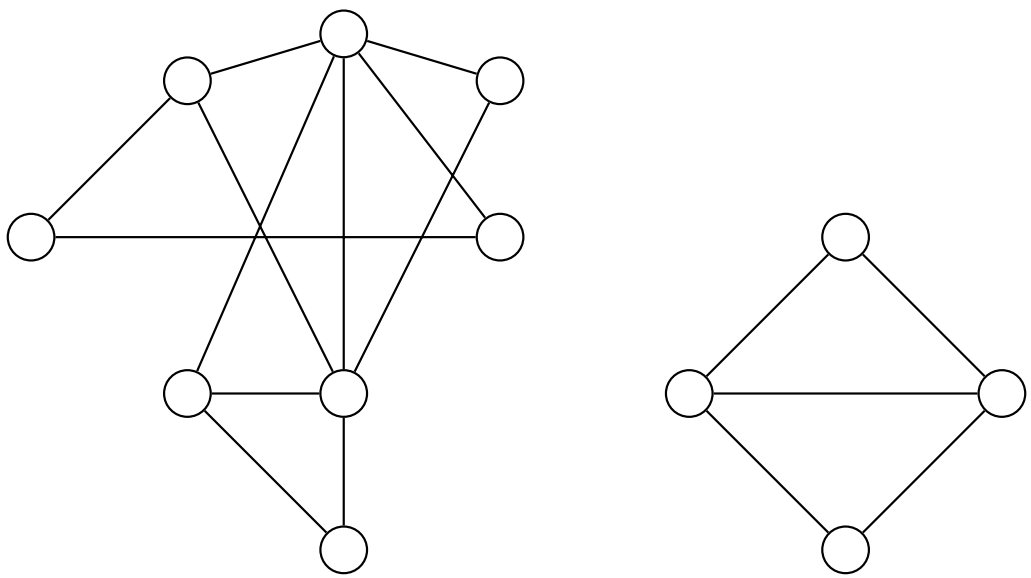
Reductie: Neem het complement van de graaf. Met andere woorden:

$$(G = (V, E), k) \quad \mapsto \quad (G' = (V, \overline{E}), k)$$



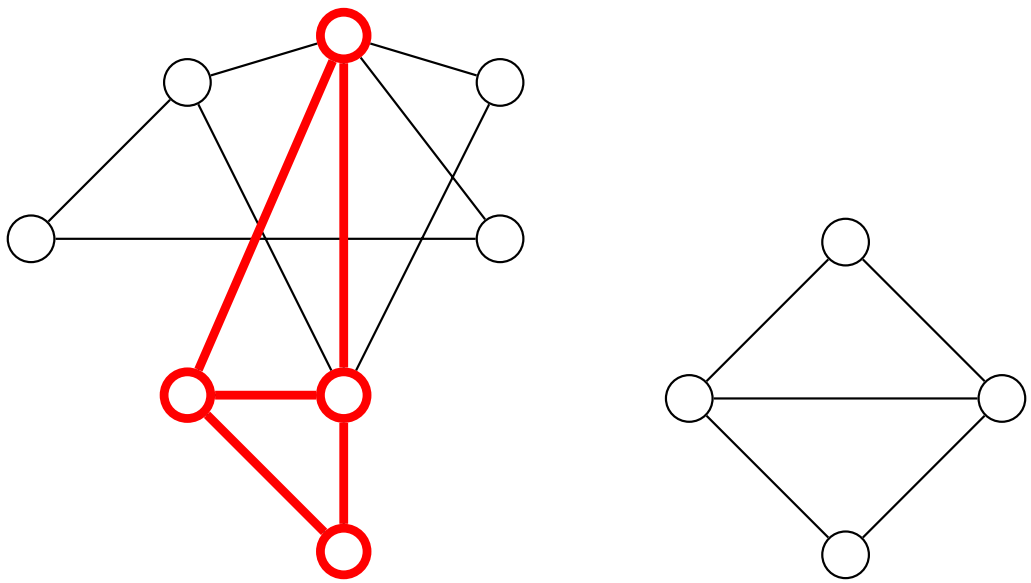
Subgraaf isomorfisme

Input: Twee grafen, $G = (V, E)$ en $H = (F, A)$.
Output: Is er een verzameling knopen $S \subseteq V$ zodanig dat de geïnduceerde subgraaf $G[S]$ gelijk is aan H? (beter gezegd: isomorf is met H)
De geïnduceerde subgraaf $G[S]$ is de graaf $G[S] = (S, \{(u, v) \in S^2 | (u, v) \in E_G\})$.



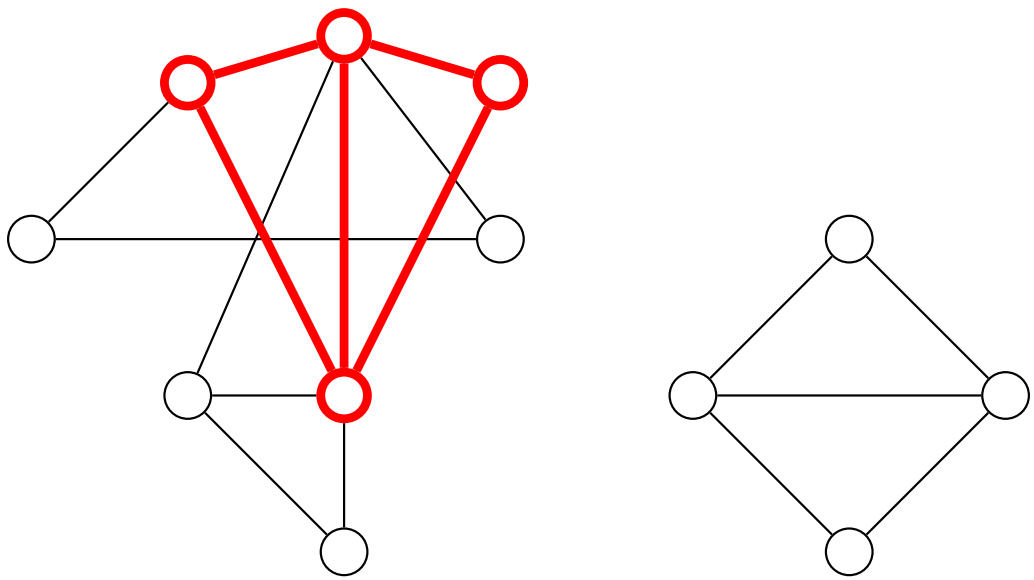
Subgraaf isomorfisme

Input: Twee grafen, $G = (V, E)$ en $H = (F, A)$.
Output: Is er een verzameling knopen $S \subseteq V$ zodanig dat de geïnduceerde subgraaf $G[S]$ gelijk is aan H? (beter gezegd: isomorf is met H)
De geïnduceerde subgraaf $G[S]$ is de graaf $G[S] = (S, \{(u, v) \in S^2 | (u, v) \in E_G\})$.



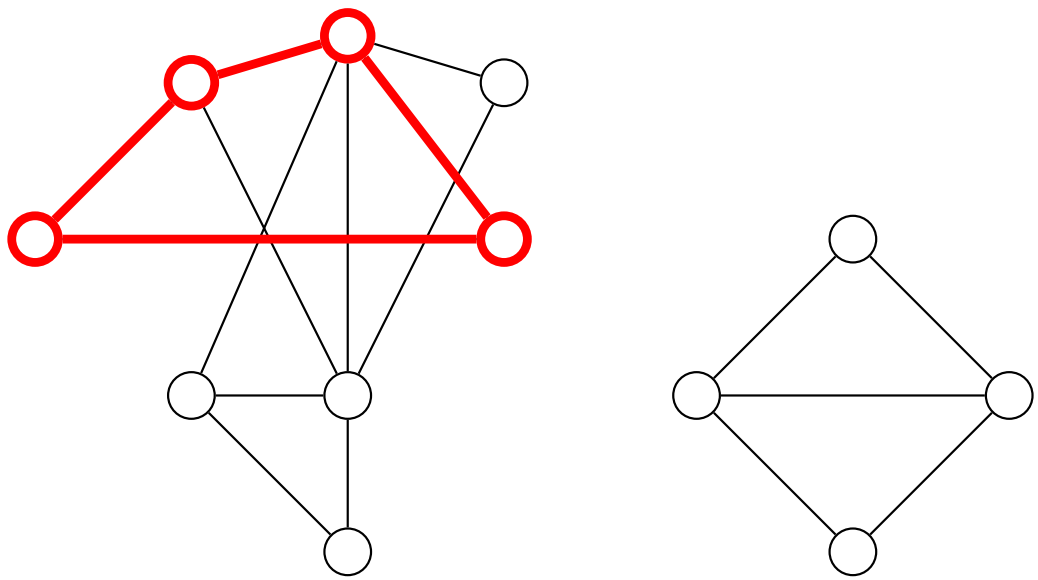
Subgraaf isomorfisme

Input: Twee grafen, $G = (V, E)$ en $H = (F, A)$.
Output: Is er een verzameling knopen $S \subseteq V$ zodanig dat de geïnduceerde subgraaf $G[S]$ gelijk is aan H? (beter gezegd: isomorf is met H)
De geïnduceerde subgraaf $G[S]$ is de graaf $G[S] = (S, \{(u, v) \in S^2 | (u, v) \in E_G\})$.



Subgraaf isomorfisme

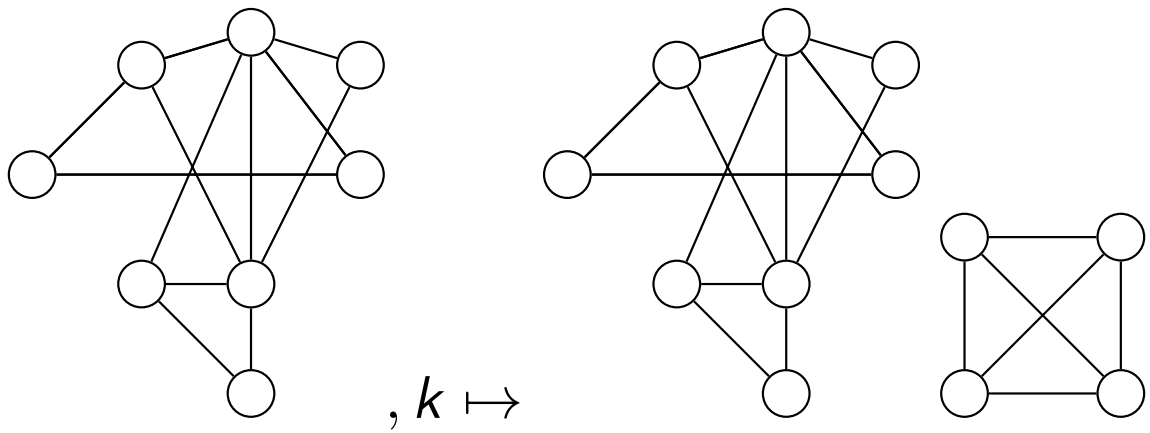
Input: Twee grafen, $G = (V, E)$ en $H = (F, A)$.
Output: Is er een verzameling knopen $S \subseteq V$ zodanig dat de geïnduceerde subgraaf $G[S]$ gelijk is aan H? (beter gezegd: isomorf is met H)
De geïnduceerde subgraaf $G[S]$ is de graaf $G[S] = (S, \{(u, v) \in S^2 | (u, v) \in E_G\})$.



Kliek $\leq_P$ Subgraaf isomorfisme

Reductie: Vraag of er een kliek in de graaf zit, dus

$$(G = (V, E), k) \mapsto (G = (V, E), H = \text{Kliek}(k))$$



Vertex Cover

Input: Een ongerichte graaf $G = (V, E)$ en een getal k
Output: Is er een verzameling knopen $S \subset V$, met hooguit $|S| \leq k$, zodanig dat elke knoop een buur in S heeft?

Reductie vanaf Independent Set. We maken van een instantie $(G = (V, E), k)$ voor Independent Set de instantie $(G' = (V, \overline{E}), |V| - k)$ voor Vertex Cover.
Dan heeft G een Independent Set van grootte minstens k desda G' een vertex cover heeft van grootte hooguit $|S| - k$.

Gericht Hamiltonpad

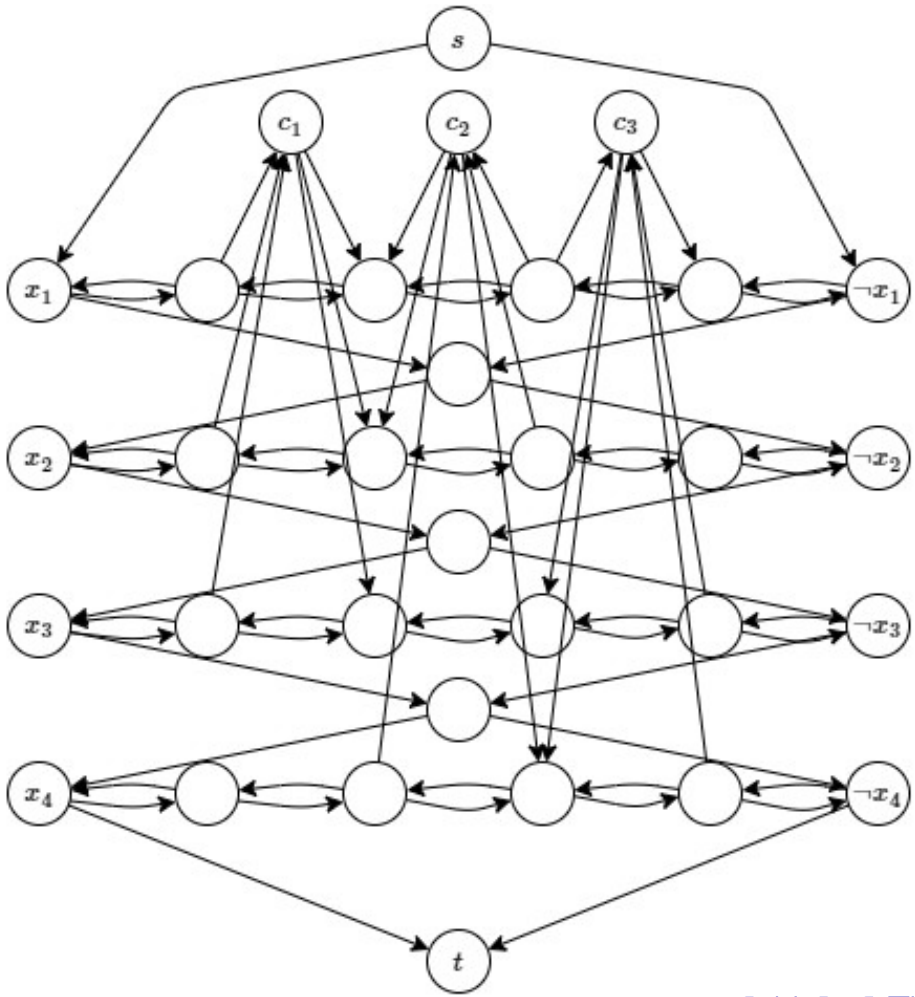
Input: Een gerichte graaf $G = (V, E)$
Output: Is er een pad dat alle vertices precies één keer bezoekt?

Reductie Vanaf 3-CNF-Satisfiability. Gegeven een instantie van 3-SATISFIABILITY, $\phi = \bigwedge_{i=1}^{m}(x_{i,1} \vee x_{i,2} \vee x_{i,3})$ met n variabelen en m clausules.
We maken van de CNF formule ϕ een graaf met $2n + m + n \cdot (m + 1) + 2$ vertices. We zorgen dat er een Hamiltonpad in de graaf is d.e.s.d.a. de formule ϕ vervulbaar is.

3-CNF Satisfiability $\leq_P$ Gericht Hamiltonpad

Voorbeeld.

$\phi = \underbrace{(x_1 \vee x_2 \vee x_3)}_{c_1} \wedge \underbrace{(\neg x_1 \vee \neg x_2 \vee x_4)}_{c_2} \wedge \underbrace{(x_1 \vee \neg x_3 \vee \neg x_4)}_{c_3}$. We maken de volgende instantie voor Gericht Hamiltonpad:



Ongericht Hamiltonpad

Input: Een ongerichte graaf $G = (V, E)$
Output: Is er een pad dat alle vertices precies één keer bezoekt?

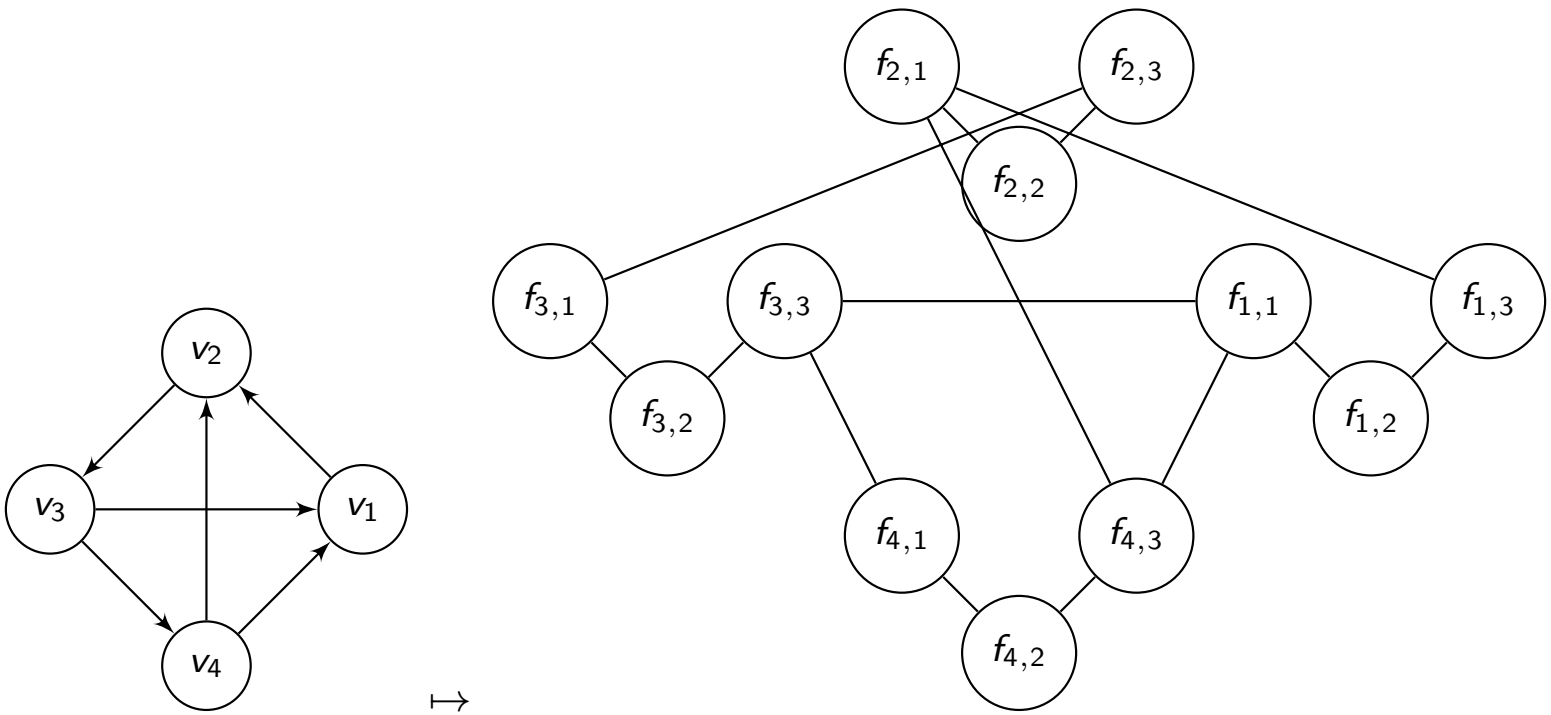
Reductie: Gegeven een graaf $G = (V = \{v_1, \ldots, v_n\}, E)$, maak een nieuwe graaf $H = (F, K)$, met knopen F:

$$F = \{f_{i,1}\}_{i=1}^{n} \cup \{f_{i,2}\}_{i=1}^{n} \cup \{f_{i,3}\}_{i=1}^{n}$$

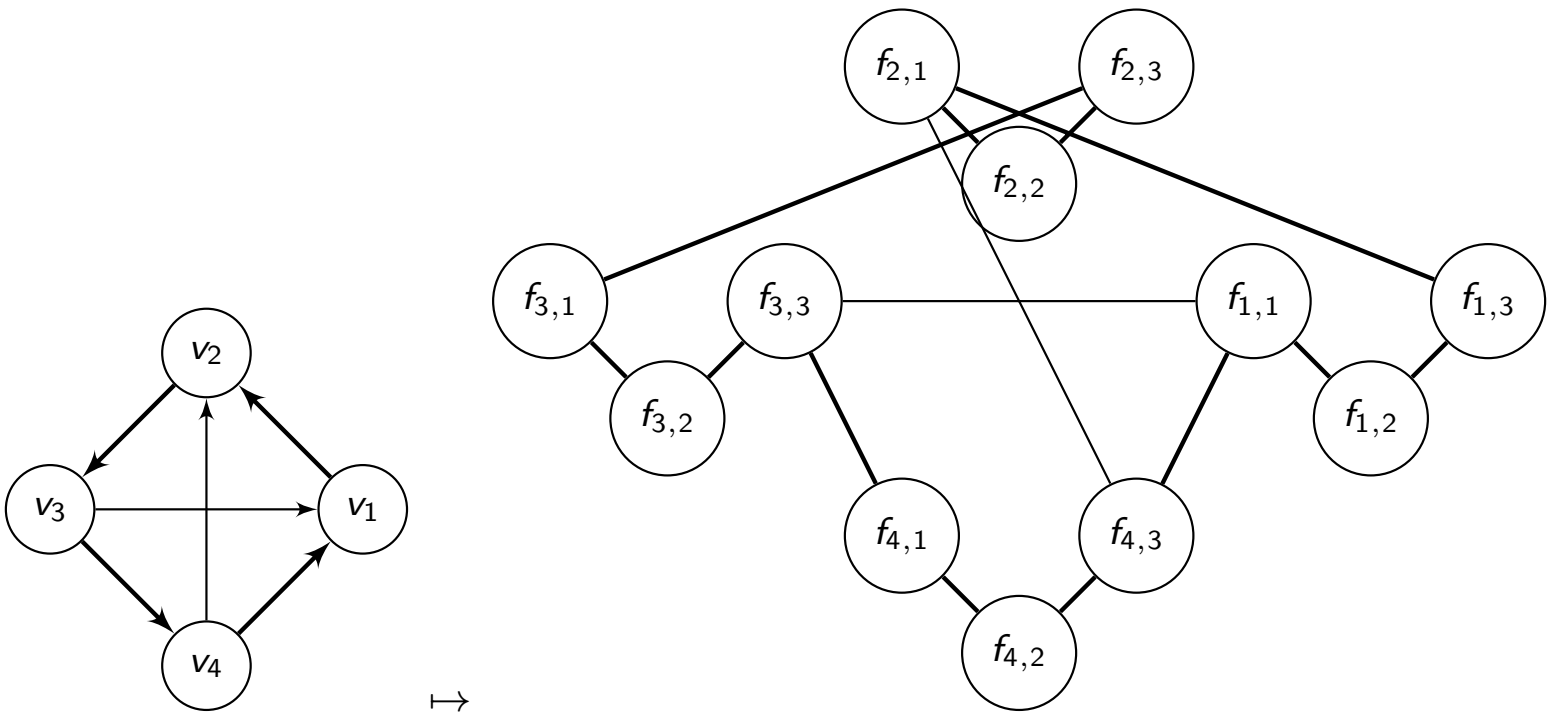
en takken K:

$$\{(f_{i,1}, f_{i,2})\}_{i=1}^{n} \cup \{(f_{i,2}, f_{i,3})\}_{i=1}^{n} \cup \{(f_{i,3}, f_{j,1}) | (v_i, v_j) \in E\}$$

Gericht Hamiltonpad $\leq_P$ Ongericht Hamiltonpad



Gericht Hamiltonpad $\leq_P$ Ongericht Hamiltonpad

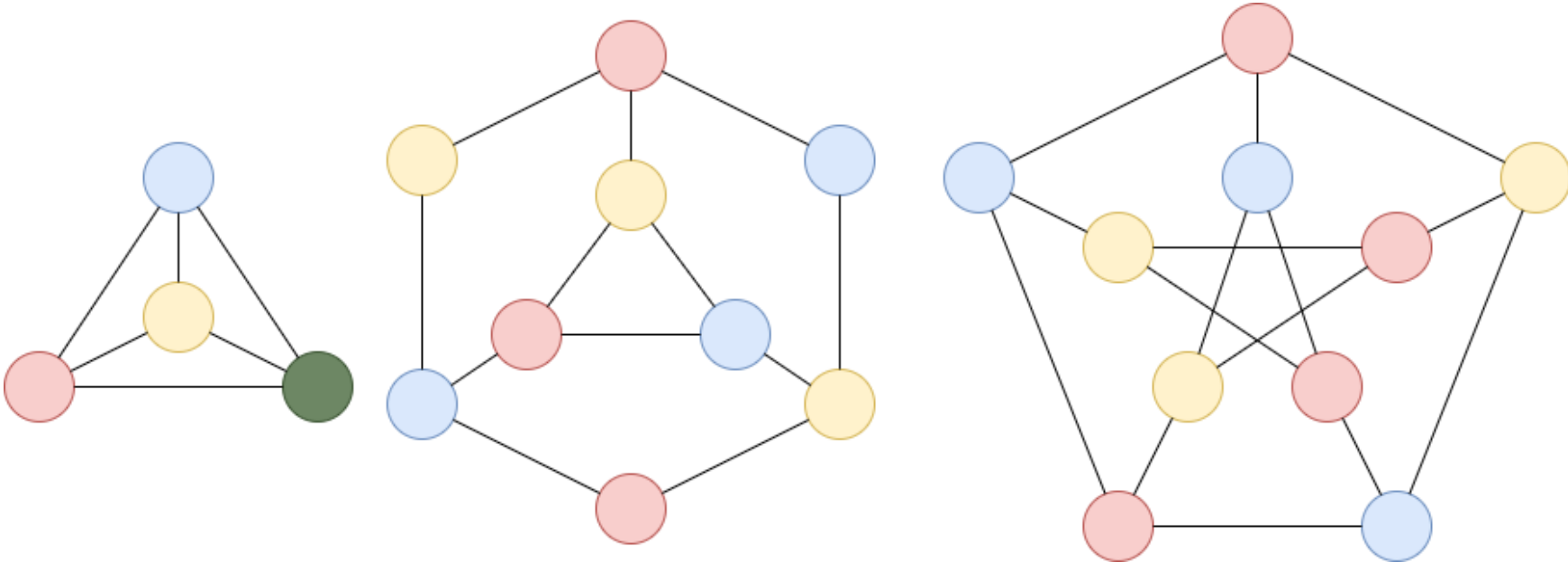


Traveling Salesman

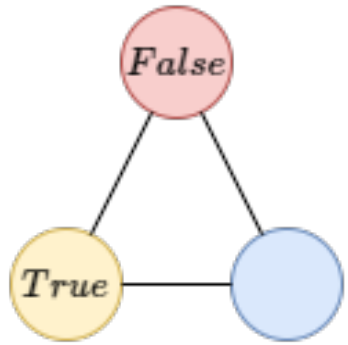
Input: Een ongerichte gewogen graaf $G = (V, E, w)$, met $w : E \to \mathbb{Z}_{\geq 0}$, en een getal k.
Output: Is er een pad dat alle vertices precies één keer bezoekt, en een gewicht heeft van hooguit k?

Ongericht Hamiltonpad is een speciaal geval van Traveling Salesman

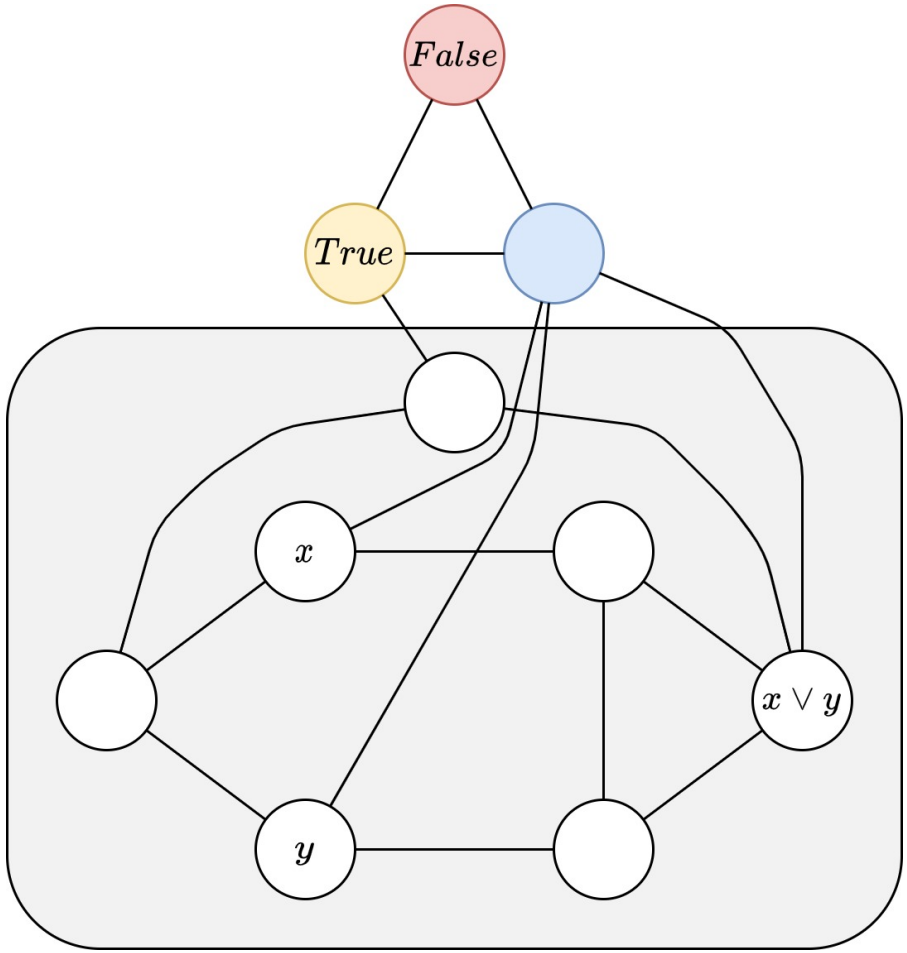
Graafkleuring



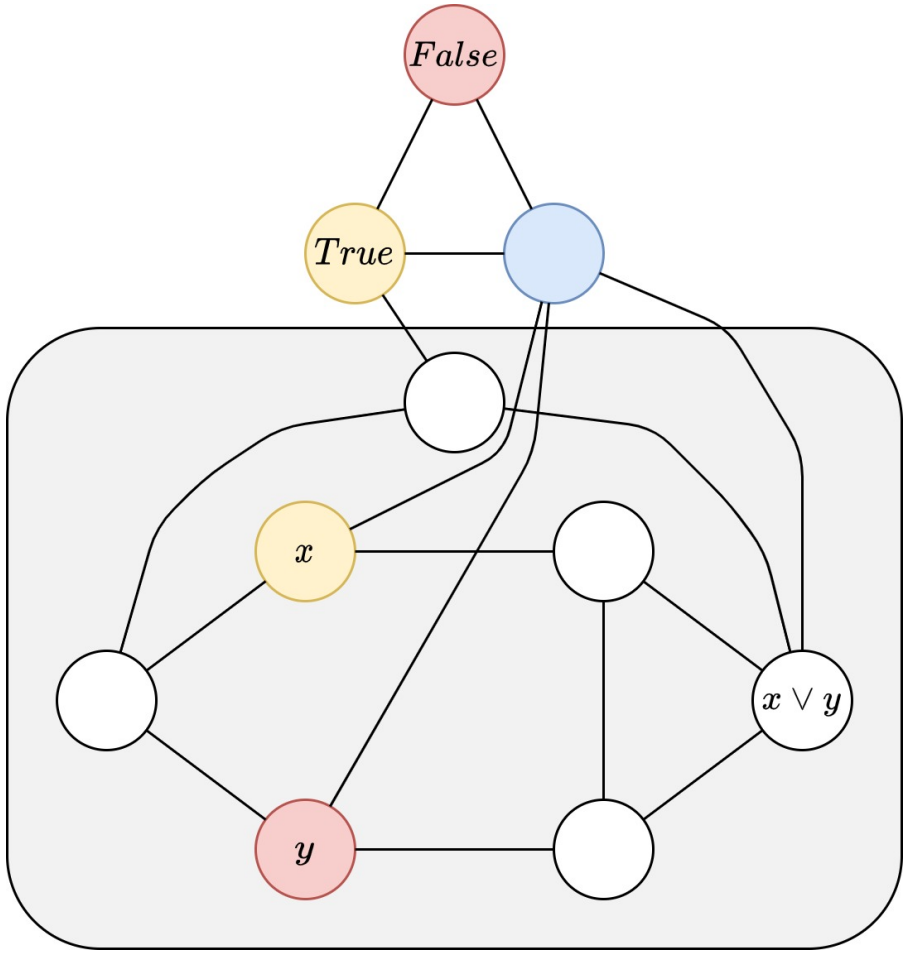
Graafkleuring



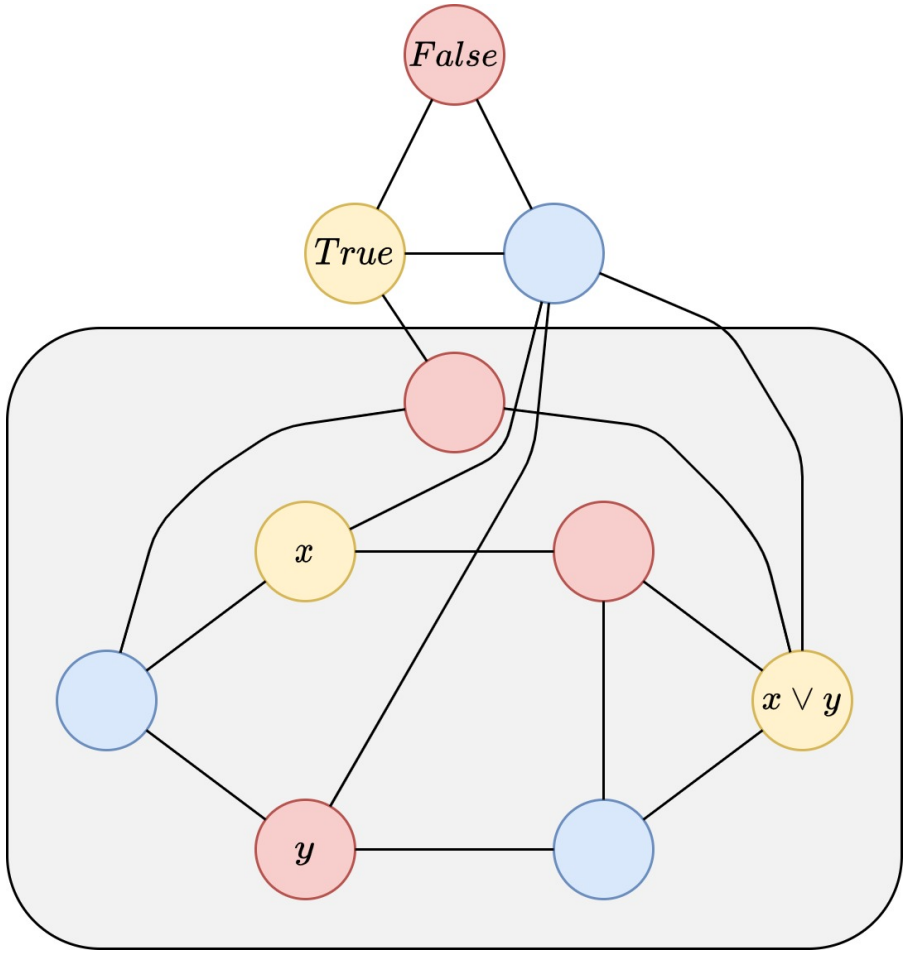
Graafkleuring: OR gadget



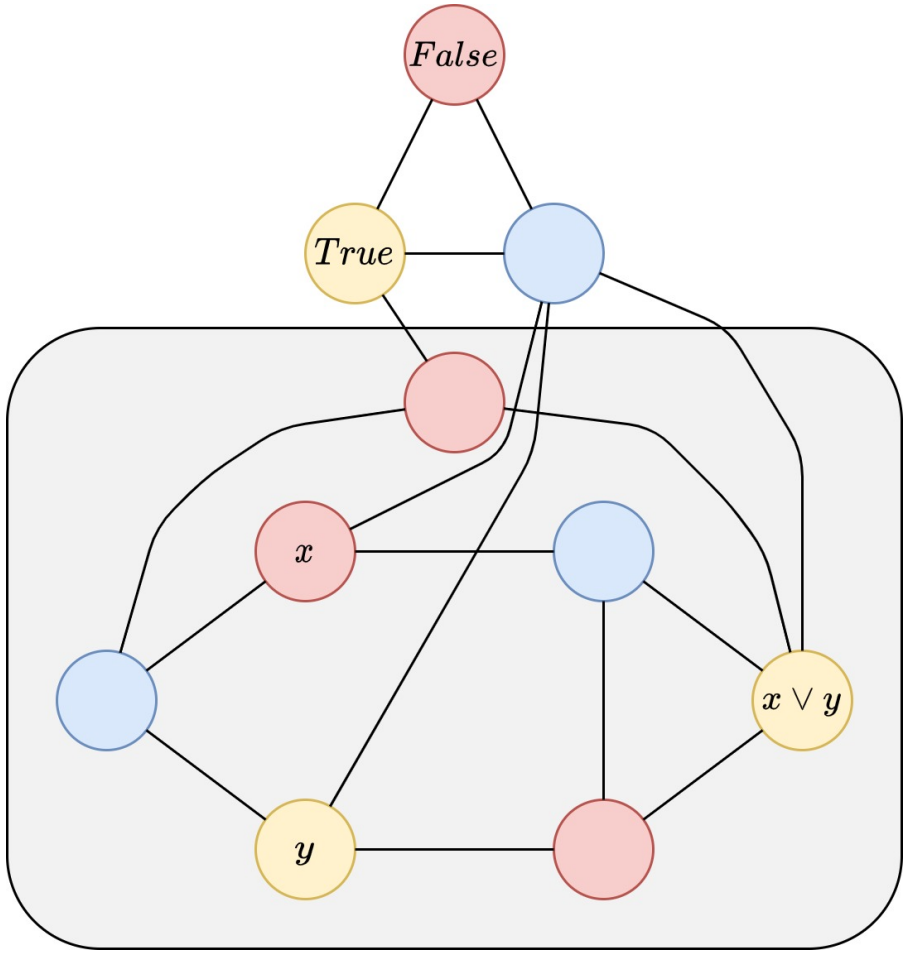
Graafkleuring: OR gadget



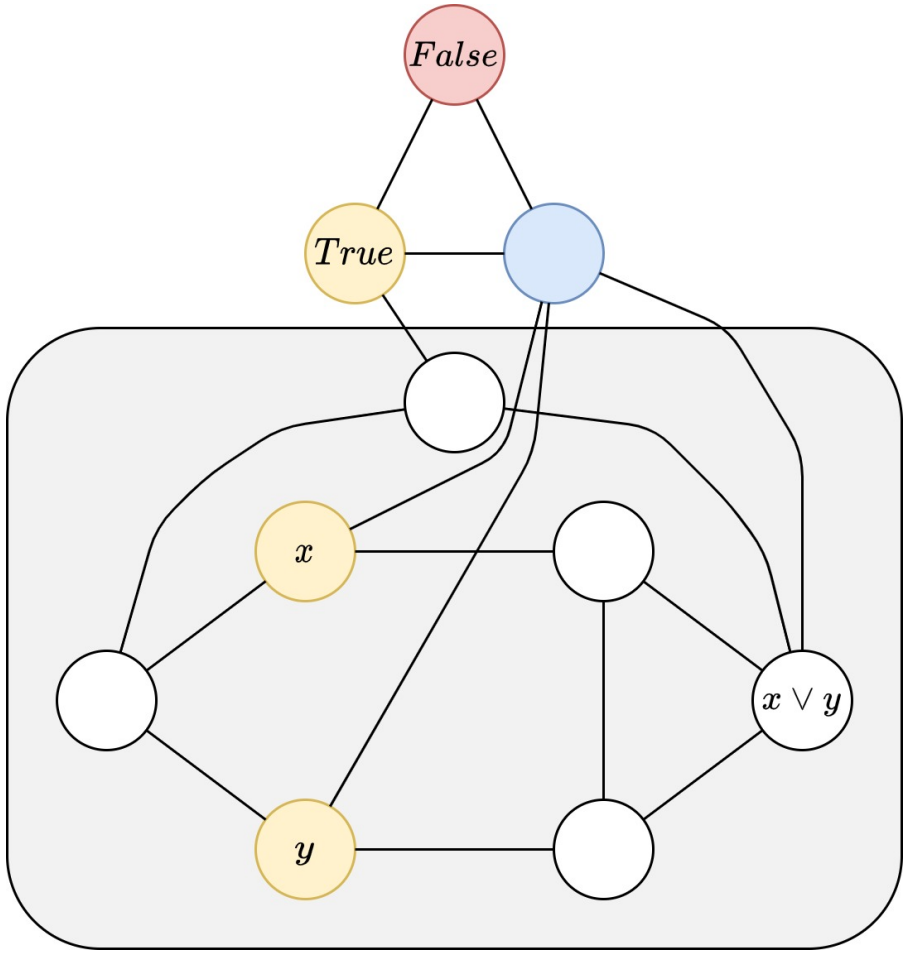
Graafkleuring: OR gadget



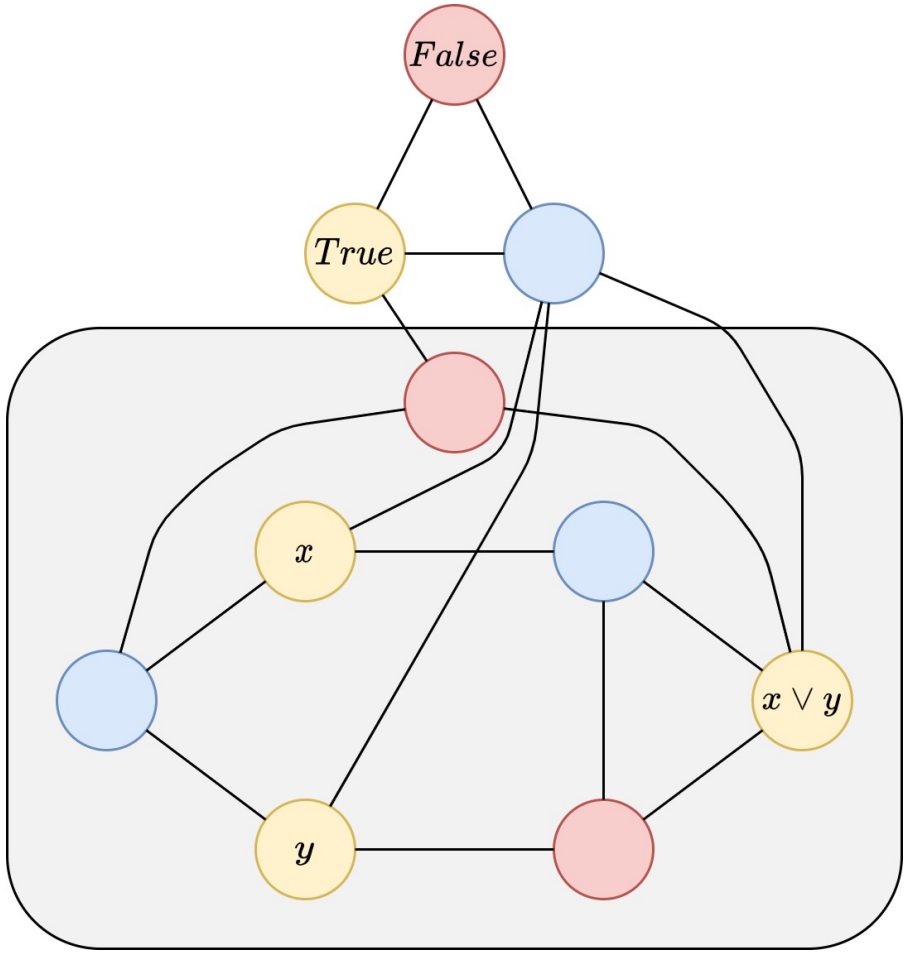
Graafkleuring: OR gadget



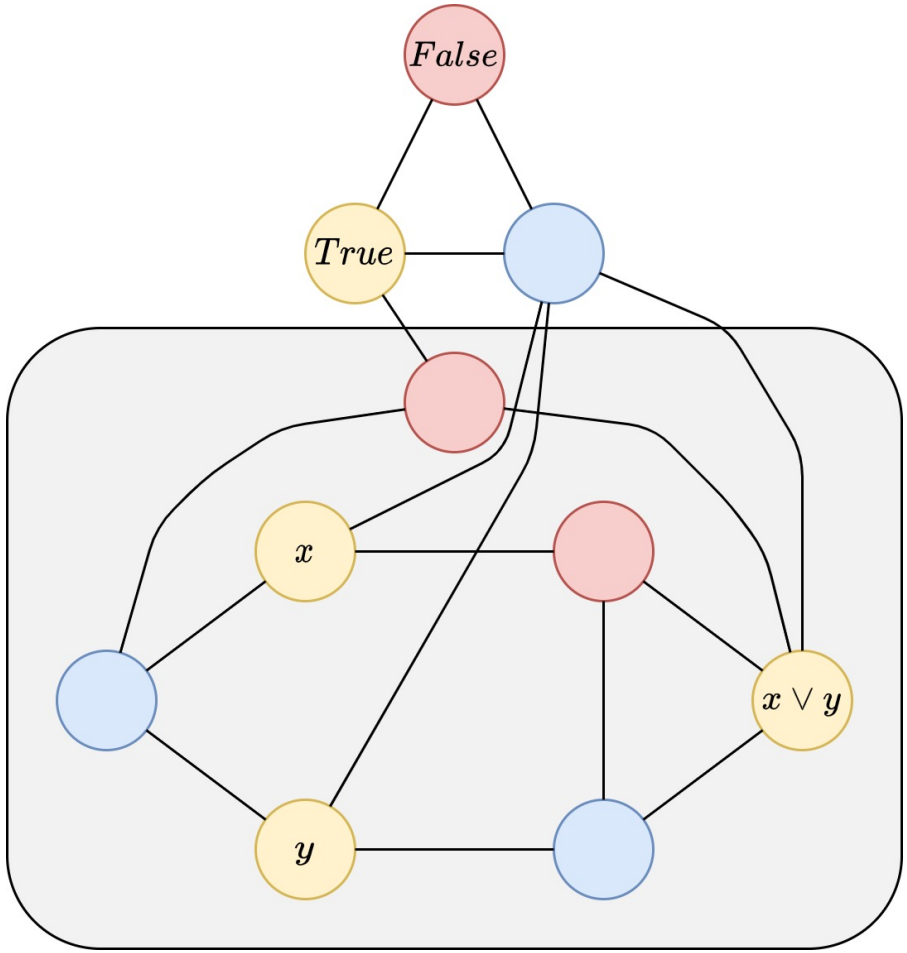
Graafkleuring: OR gadget



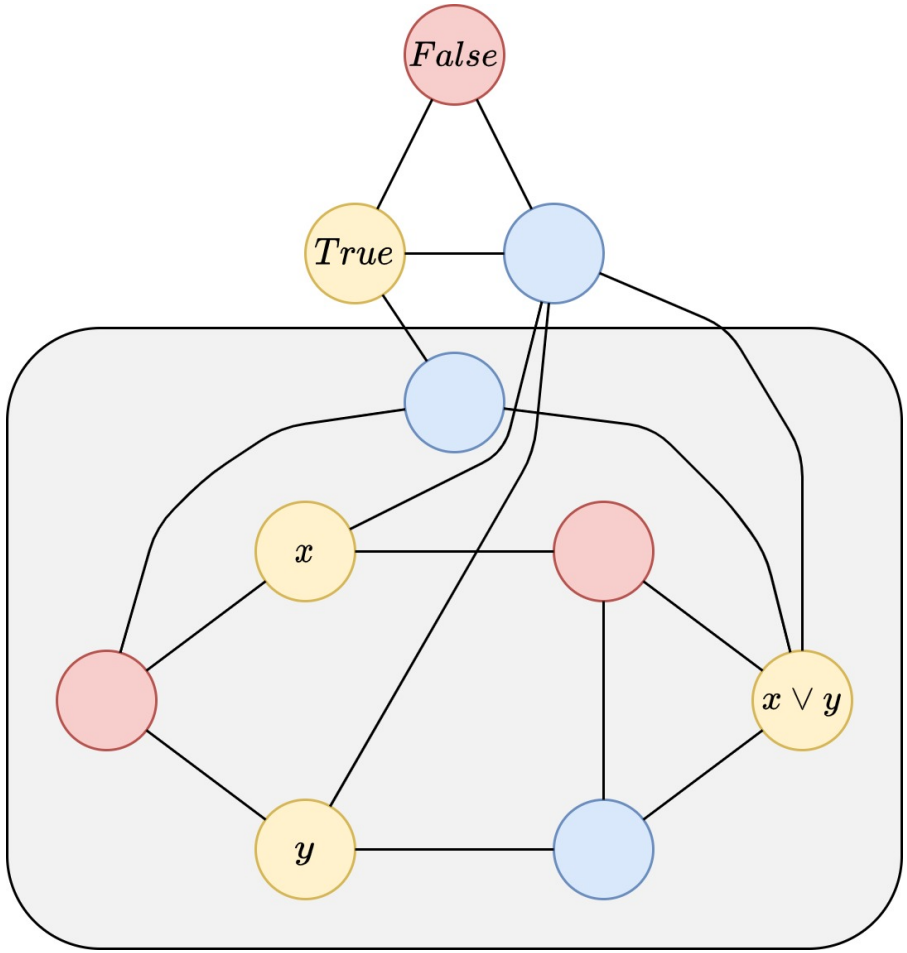
Graafkleuring: OR gadget



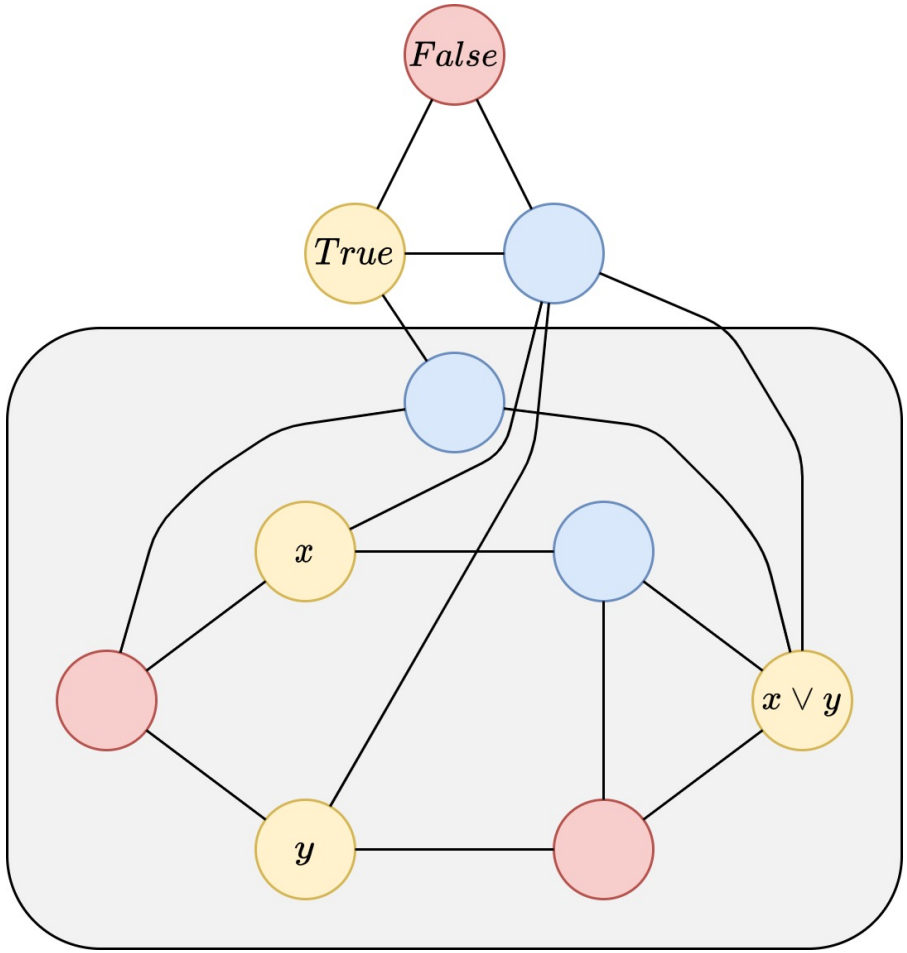
Graafkleuring: OR gadget



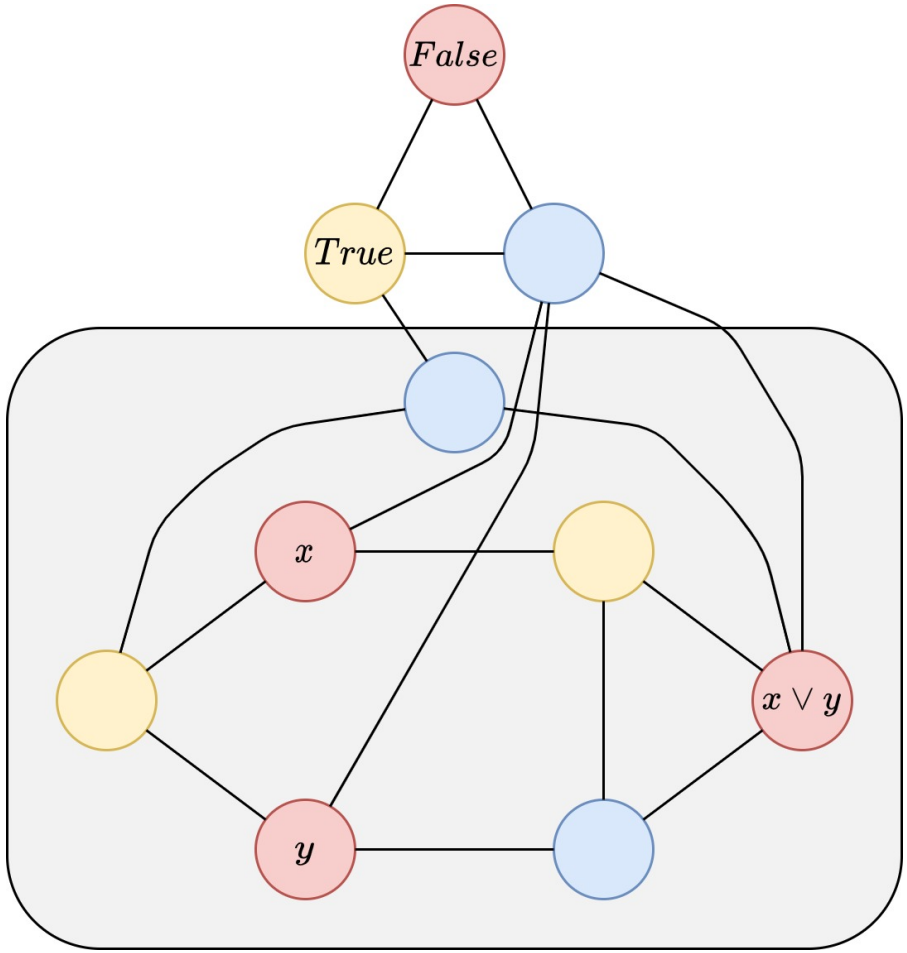
Graafkleuring: OR gadget



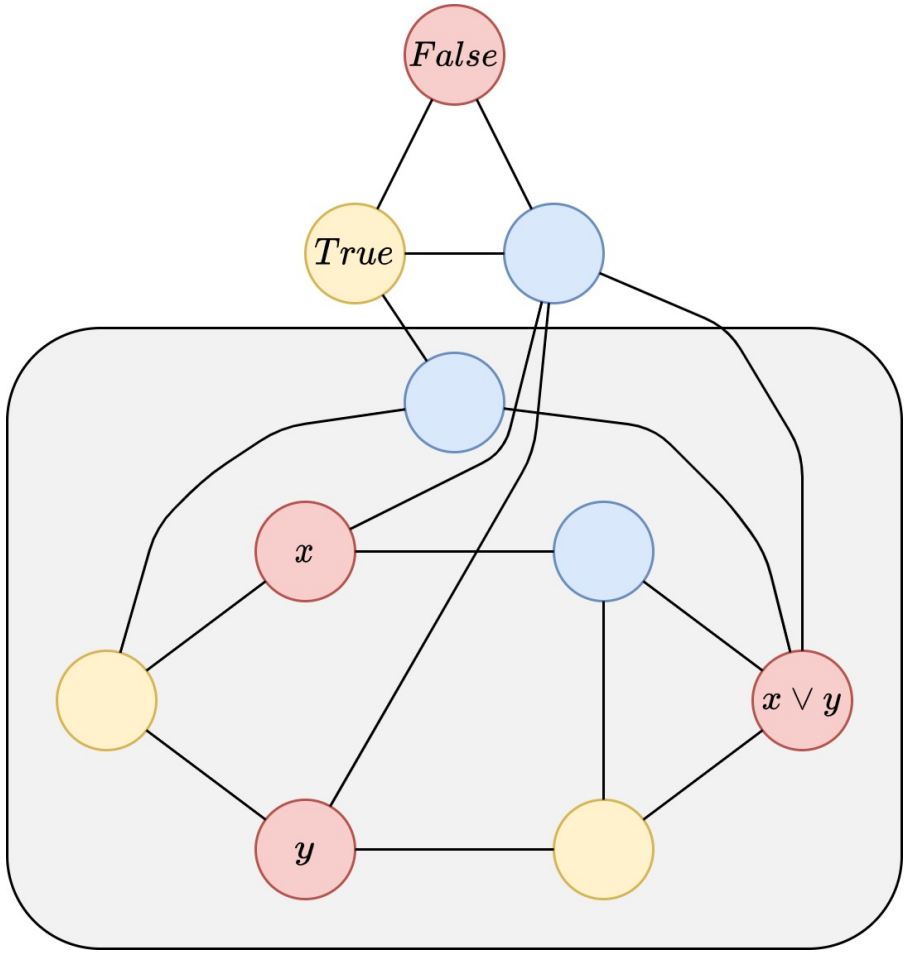
Graafkleuring: OR gadget



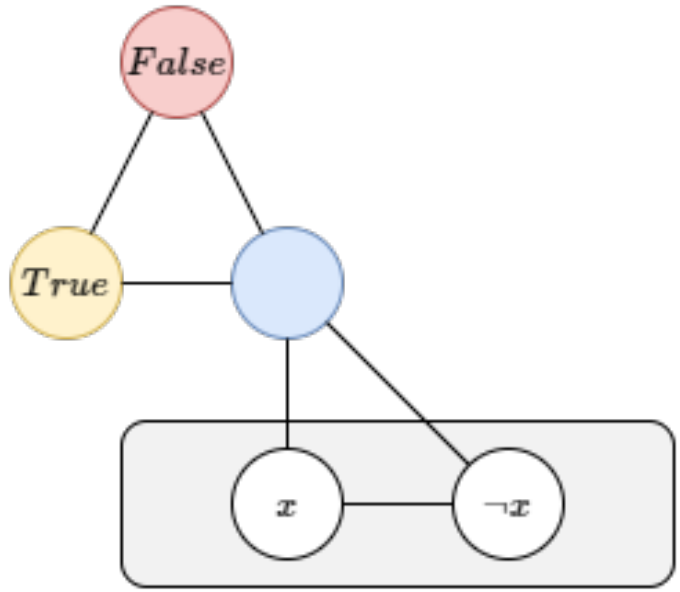
Graafkleuring: OR gadget



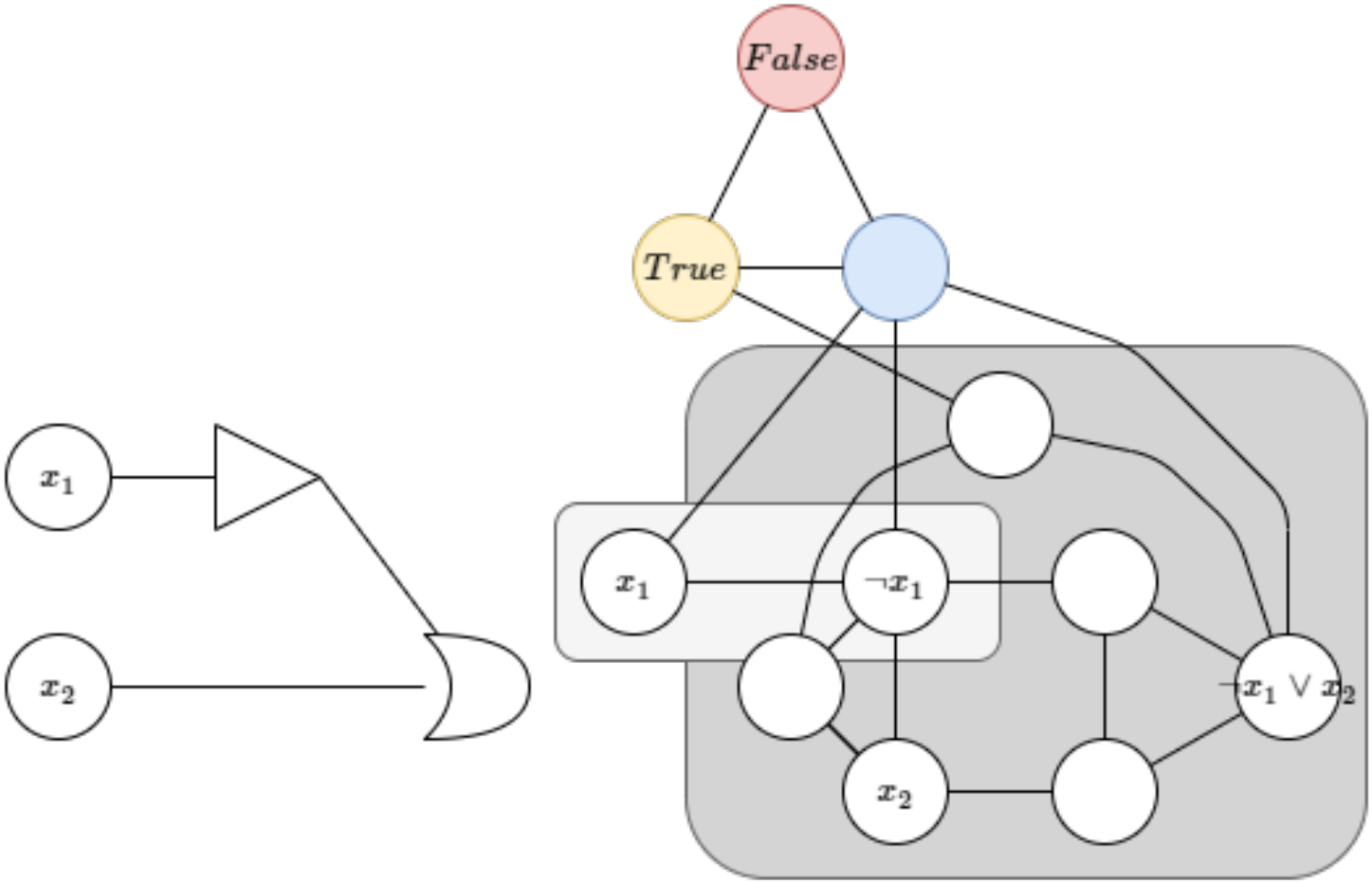
Graafkleuring: OR gadget



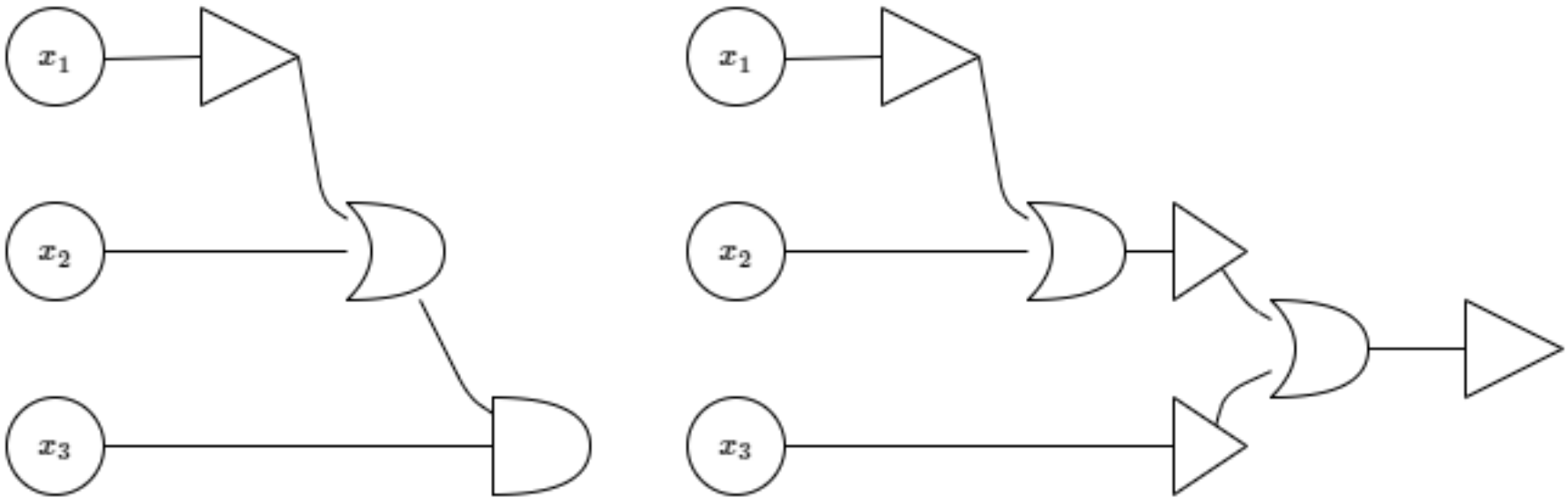
Graafkleuring: NOT gadget



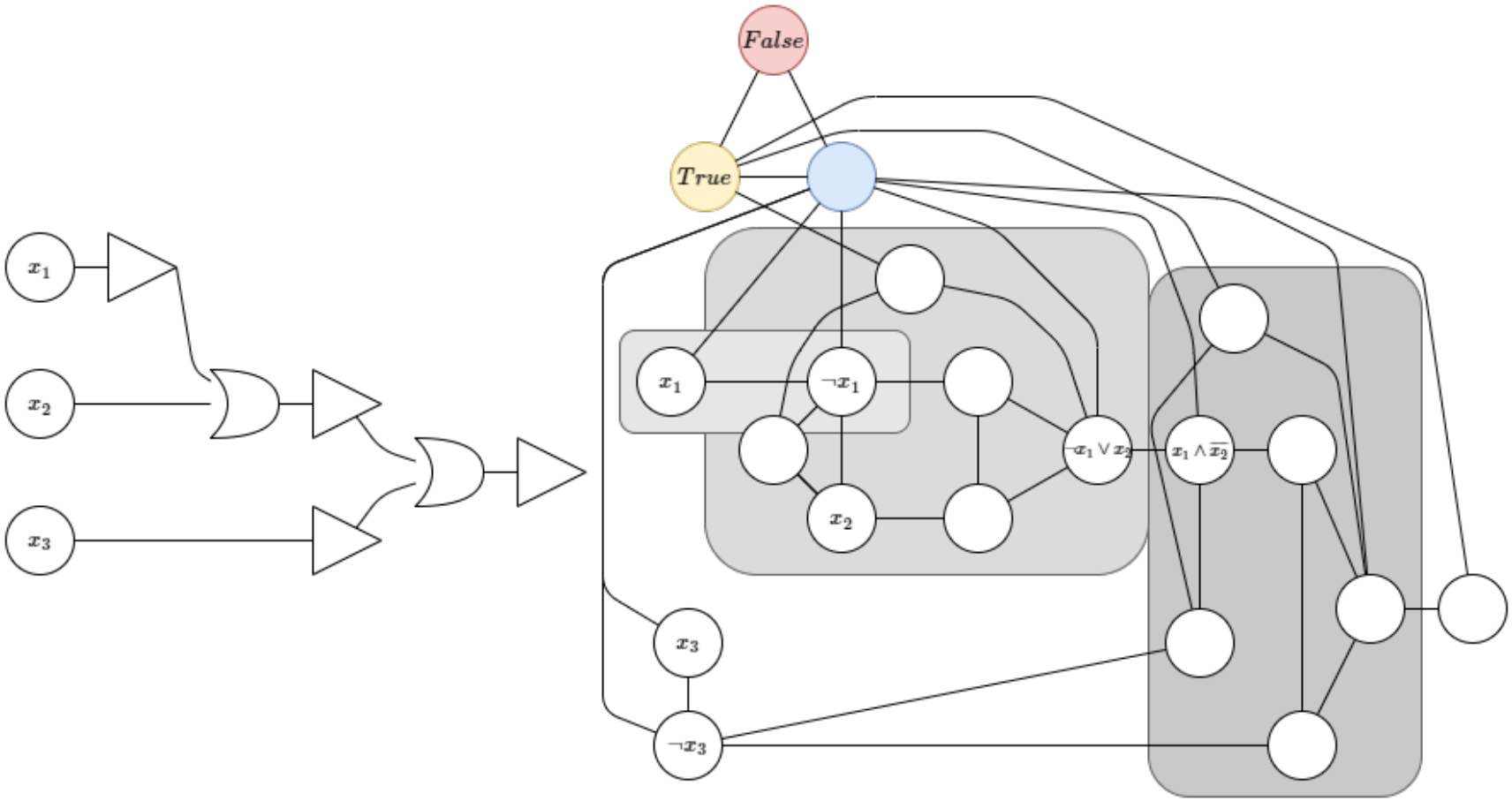
Graafkleuring



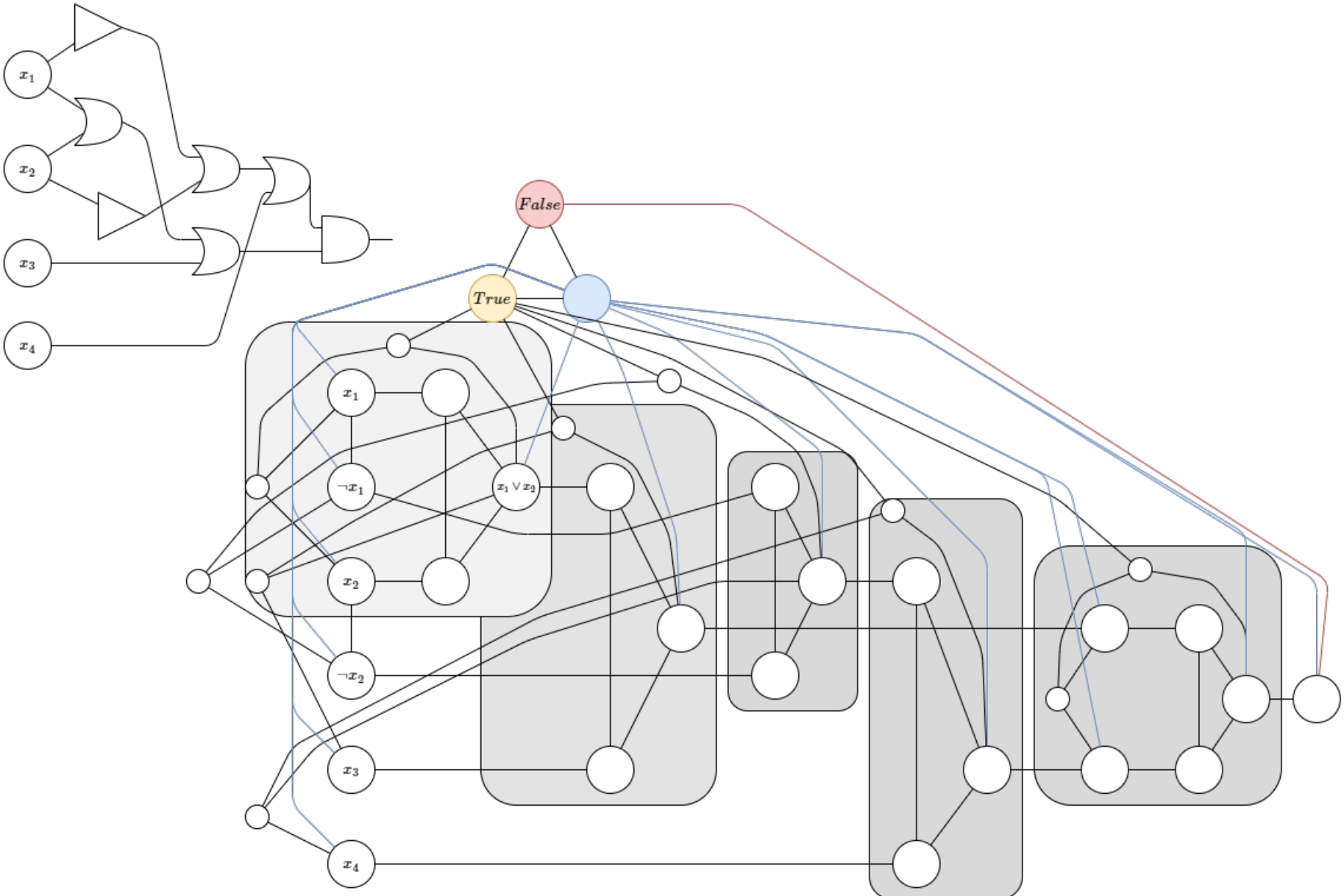
Graafkleuring



Graafkleuring: Klein circuit



Graafkleuring



Exactly-1-Satisfiability

Input: Een aantal Boolse vergelijkingen van de vorm $a + c + b = 1$.
Output: Kunnen de vergelijkingen allemaal worden vervuld?

Voorbeeld.

$$y + z = 1 \tag{1}$$
$$a + x + y = 1 \tag{2}$$
$$a + x + z = 1 \tag{3}$$
$$a + x = 1 \tag{4}$$

(Er is geen oplossing. Dit is dus een nee-instantie van Exactly-1-Satisfiability)

Graafkleuring $\leq_P$ Exactly-1-Satisfiability

Input: Een aantal Boolse vergelijkingen van de vorm $a + c + b = 1$.
Output: Kunnen de vergelijkingen allemaal worden vervuld?

Reductie: Vanaf Graafkleuring. We maken van een ongerichte graaf $G = (V, E)$ een verzameling vergelijkingen zodanig dat de vergelijkingen een oplossing hebben d.e.s.d.a. de graaf een 3-kleuring heeft.
We gebruiken de volgende $3 \cdot |V| + 6 \cdot |E|$ variabelen:

1. $x_{v,c}$ voor knoop $v \in V$ en kleur $c \in \{1, 2, 3\}$ (dus $3|V|$ variabelen)
2. y_{e,c_1,c_2} voor tak $e \in E$ en kleurcombinatie $(c_1, c_2) \in \{(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)\}$ (dus $6|E|$ variabelen)

Graafkleuring $\leq_P$ Exactly-1-Satisfiability

Input: Een aantal Boolse vergelijkingen van de vorm $a + c + b = 1$.

Output: Kunnen de vergelijkingen allemaal worden vervuld?

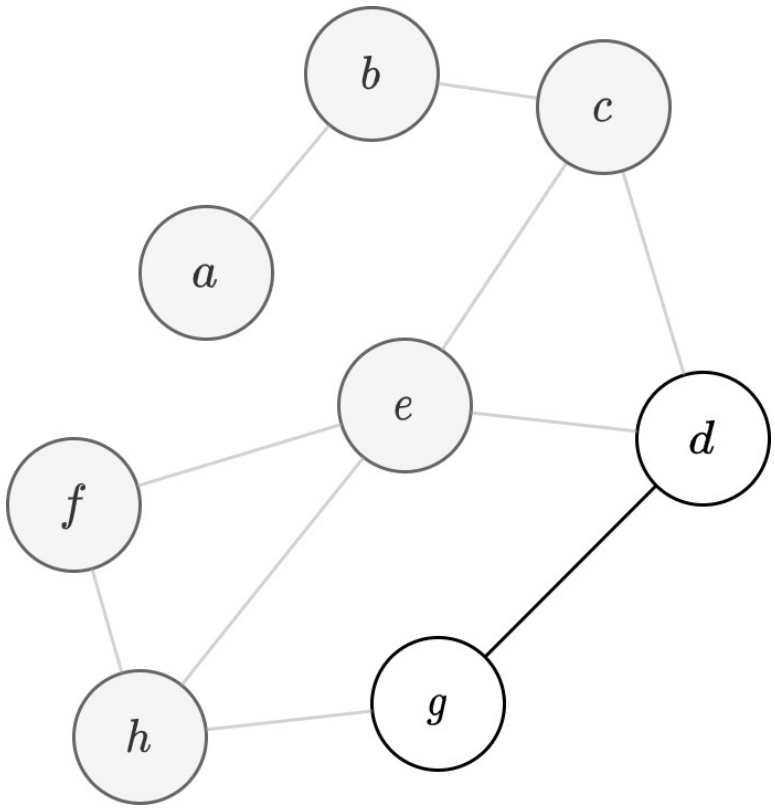
Reductie. (vervolg)

De vergelijkingen vallen uiteen in twee groepen:

1. Elke knoop heeft één kleur: $x_{v,1} + x_{v,2} + x_{v,3} = 1$ voor elke knoop $v \in V$
2. Elke tak heeft een geldige kleurcombinatie:
   $y_{e,1,2} + y_{e,1,3} + y_{e,2,1} + y_{e,2,3} + y_{e,3,1} + y_{e,3,2} = 1$ voor elke tak $e \in E$
3. Twee aangrenzende knopen hebben verschillende kleuren:
   $x_{v,c} + y_{e,c,c+1} + y_{e,c,c+2} + y_{e,c+1,c+2} + y_{e,c+2,c+1} = 1$ voor elke knoop v die grenst aan tak e, voor elke kleur $c \in \{1, 2, 3\}$. (Dit zijn 6 vergelijkingen per tak)

Nu heeft de verzameling vergelijkingen precies de oplossingen die ook 3-kleuringen van de graaf voorstellen.

Graafkleuring $\leq_P$ Exactly-1-Satisfiability



$$x_{d,1} + x_{d,2} + x_{d,3} = 1$$

$$x_{g,1} + x_{g,2} + x_{g,3} = 1$$

$$x_{d,1} + y_{dg,2,1} + y_{dg,2,3} + y_{dg,3,1} + y_{dg,3,2} = 1$$

$$x_{d,2} + y_{dg,1,2} + y_{dg,1,3} + y_{dg,3,1} + y_{dg,3,2} = 1$$

$$x_{d,3} + y_{dg,1,2} + y_{dg,1,3} + y_{dg,2,1} + y_{dg,2,3} = 1$$

$$x_{g,1} + y_{dg,1,2} + y_{dg,1,3} + y_{dg,2,3} + y_{dg,3,1} = 1$$

$$x_{g,2} + y_{dg,1,3} + y_{dg,2,1} + y_{dg,2,3} + y_{dg,3,1} = 1$$

$$x_{g,3} + y_{dg,1,2} + y_{dg,2,1} + y_{dg,3,1} + y_{dg,3,2} = 1$$

Subset Sum

Input: Een verzameling gehele getallen S en een geheel getal t
Output: Is er een deelverzameling $A \subseteq S$ zodanig dat $\sum_{a \in A} a = t$?

Reductie vanaf Exactly-1-Satisfiability. Zij ϕ een instantie met n variabelen en m vergelijkingen, elke vergelijking heeft hooguit 3 variabelen. We maken een instantie (S, t) van Subset Sum die vervulbaar is d.e.s.d.a. ϕ vervulbaar is.
De verzameling heeft $2n$ elementen, namelijk $a_{x,0}$ en $a_{x,1}$ voor elke variabele x in de formule ϕ. Het idee is dat een verzameling A met $t_{x,1} \in A$ correspondeert met een oplossing voor ϕ waarin $x = 1$, en net zo voor $t_{x,0} \in A$; dan geldt $x = 0$.
Nummer de vergelijkingen in ϕ van 1 tot m. Ga elke variabele x van ϕ langs. Als x in vergelijkingen $c_{x,1}, \ldots, c_{x,\ell}$ voorkomt, voeg dan het element $a_{x,1} = 4^{m+x} + \sum_{k=1}^{\ell} 2^{2c_{x,k}}$ toe. Als $\neg x$ in vergelijkingen $c_{\neg x,1}, \ldots, c_{\neg x,\ell}$ voorkomt, voeg het element $a_{x,0} = 4^{m+x} + \sum_{k=1}^{\ell} 2^{2c_{\neg x,k}}$ toe. Zet $t = \sum_{k=1}^{\ell} 4^k + \sum_{k=1}^{n} 4^{m+k}$.

3-SUM

Input: Een lijst met gehele getallen, $S \subseteq \mathbb{Z}$
Output: Bevat S drie getallen $a, b, c \in S$ die voldoen aan $a + b + c = 0$?

Voorbeeld. $S = \{-10, -7, -1, 0, 1, 3, 4, 8\}$. Oplossingen:

$$-7 + 3 + 4 = 0$$
$$-7 - 1 + 8 = 0$$

3-SUM in $\mathcal{O}(n^3)$ tijd

Input: Een lijst met gehele getallen, $S \subseteq \mathbb{Z}$
Output: Bevat S drie getallen $a, b, c \in S$ die voldoen aan $a + b + c = 0$?

Er is een simpel algoritme dat $\mathcal{O}(n^3)$ tijd kost:

```
1: for a, b, c ∈ S do
2:     if a + b + c = 0 then
3:         return ☺
4:     end if
5: end for
6: return ☹
```

Met een handiger algoritme kan het in $\mathcal{O}(n^2)$ tijd, want als je a en b gekozen hebt, dan ligt c vast. Opgave.

Onopgelost: Kan het in $\mathcal{O}(n^{2-\varepsilon})$ tijd, voor een $\varepsilon > 0$?

3-SUM

Een hoop problemen zijn even moeilijk als 3-SUM:

1. Gegeven een verzameling lijnen in het vlak, zijn er drie die in één punt samenkomen?
2. Gegeven een vereniging driehoeken in het vlak, heeft de vereniging van deze oppervlakken een gat?
3. Bereken de oppervlakte van een gegeven verzameling (overlappende) driehoeken
4. Kan één convex polygon binnen een ander convex polygon worden geplaatst?
5. Bereken de Haussdorf Afstand tussen twee verzamelingen lijnen in het vlak.
6. ...

Although this does not prove a lower bound for these problems, there is no hope of obtaining $o(n^2)$ solutions for them unless we can improve the solution for the base problem (3-SUM)

Anka Gajentaan, Mark Overmars, 1995

Drie-lijsten-3-SUM

Input: Drie lijsten met gehele, positieve getallen, $A, B, C \subset \mathbb{Z}_{\geq 0}$.
Output: Zijn er $a \in A, b \in B, c \in C$ die voldoen aan $a + b = c$?

Voorbeeld.

$$A = \{2, 4, 9, 16, 25\} \tag{5}$$
$$B = \{2, 3, 5, 7, 11\} \tag{6}$$
$$C = \{3, 8, 10, 16, 17\} \tag{7}$$

3-SUM $\leq_P$ DRIE-LIJSTEN-3-SUM

We hebben twee verschillende varianten van 3-SUM. Ze zijn "even moeilijk".

Input: Een lijst met gehele getallen, $S \subseteq \mathbb{Z}$
Output: Bevat S drie getallen $a, b, c \in S$ die voldoen aan $a + b + c = 0$?

Input: Drie lijsten met gehele, positieve getallen, $A, B, C \subset \mathbb{Z}_{\geq 0}$.
Output: Zijn er $a \in A, b \in B, c \in C$ die voldoen aan $a + b = c$?

Reductie: We maken van een verzameling S drie lijsten A, B, C.
We zetten $A := S, B := S, C := -S$ $(= \{-s \mid s \in S\})$.
Als $s_1 + s_2 + s_3 = 0$ dan geldt nu $s_1 + s_2 = -(-s_3)$.

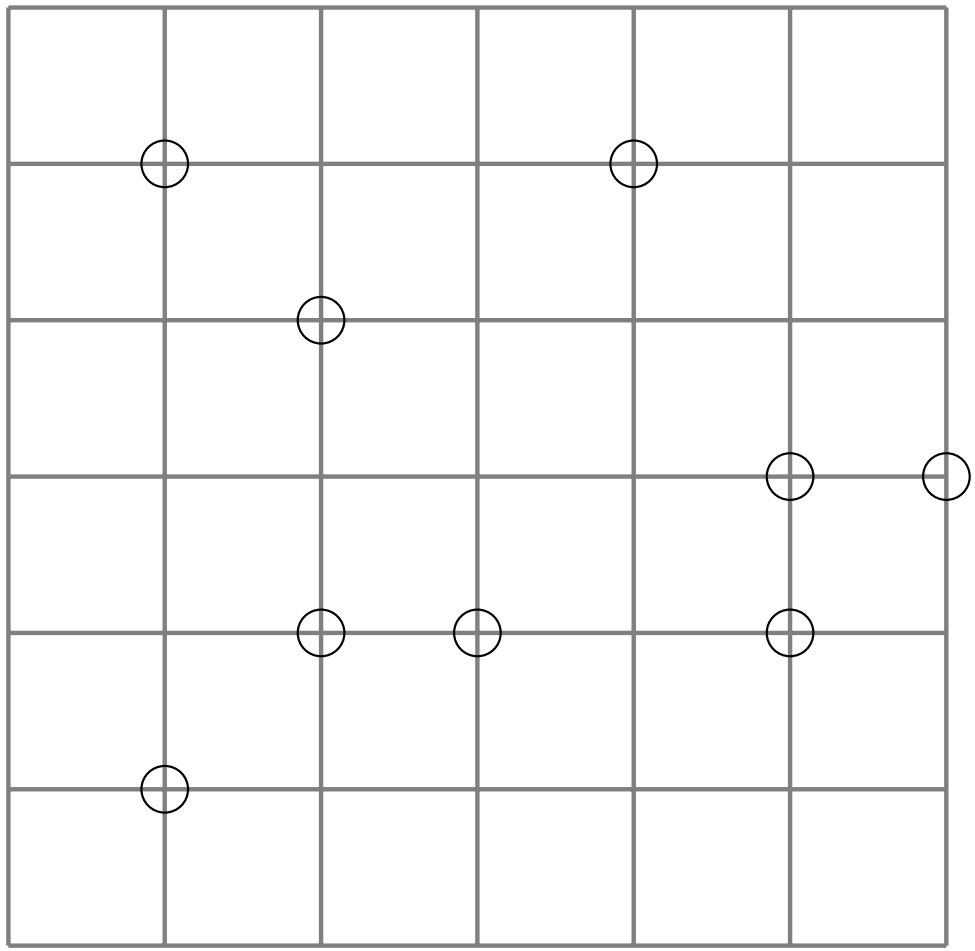
Corollary (Gajentaan, Overmars, 1995)

Als 3-SUM *opgelost kan worden in* $\mathcal{O}(t(n))$ *tijd, dan kan* DRIE-LIJSTEN-3-SUM *ook opgelost worden in* $\mathcal{O}(t(n))$ *tijd.*

3 Punten op een lijn

Input: Een lijst van punten in het vlak met geheeltallige coördinaten, $S \subset \mathbb{Z}^2$.

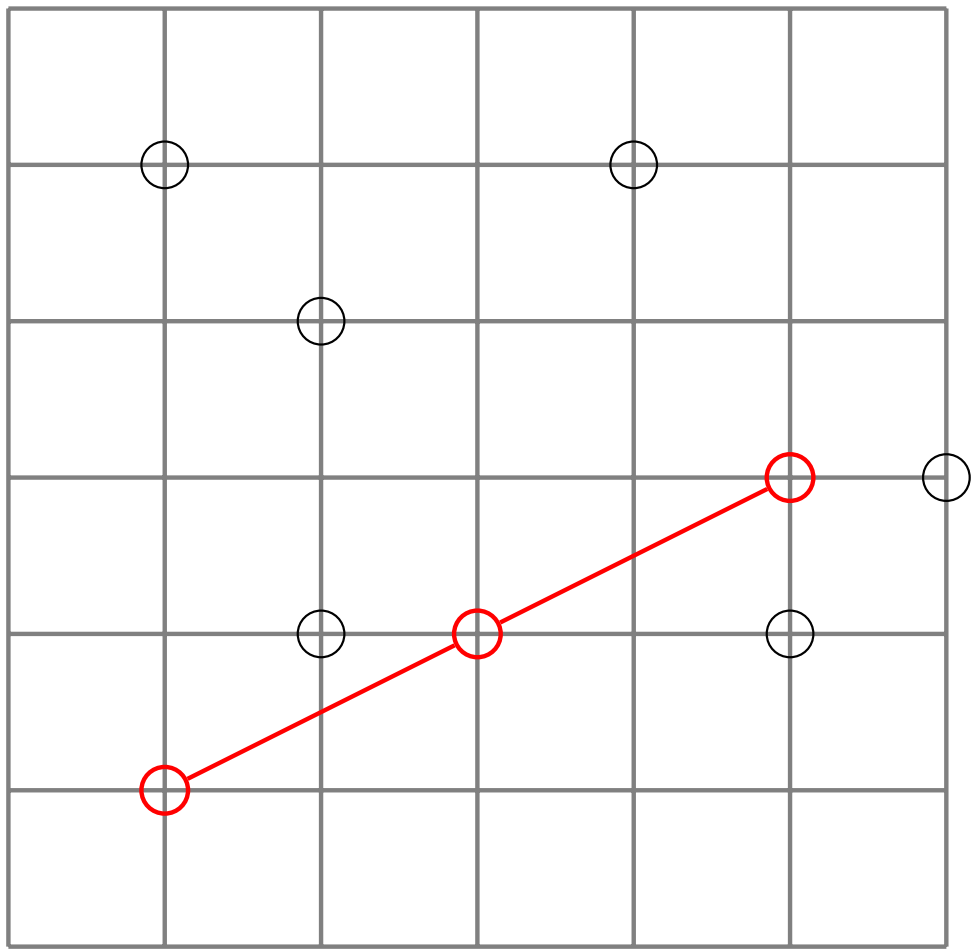
Output: Bevat S drie punten die op één lijn liggen?



3 Punten op een lijn

Input: Een lijst van punten in het vlak met geheeltallige coördinaten, $S \subset \mathbb{Z}^2$.

Output: Bevat S drie punten die op één lijn liggen?



3-SUM $\leq$ 3 PUNTEN OP EEN LIJN

Reductie: Maak voor elk getal $s \in S$ het punt $(s, s^3) \in \mathbb{Z}^2$. Dus

$$S' := \{(s, s^3) \mid s \in S\}$$

Nu geldt: $a + b + c = 0$ d.e.s.d.a. de punten $(a, a^3), (b, b^3), (c, c^3)$ op één lijn liggen. Uitwerken is een opgave.

Corollary (Gajentaan, Overmars, 1995)

Als 3 PUNTEN OP EEN LIJN *opgelost kan worden in* $\mathcal{O}(t(n))$ *tijd, dan kan* 3-SUM *ook opgelost worden in* $\mathcal{O}(t(n))$ *tijd.*

3-SUM: Reducties

Corollary (Gajentaan, Overmars, 1995)

Als 3 PUNTEN OP EEN LIJN *opgelost kan worden in* $\mathcal{O}(n^{2-\varepsilon})$ *tijd, dan kan* 3-SUM *ook opgelost worden in* $\mathcal{O}(n^{2-\varepsilon})$ *tijd.*

Om dit gevolg te impliceren, moet een reductie van 3-SUM naar L:

1. Een algoritme zijn dat 3-SUM oplost, en
2. In $O(n^{2-\varepsilon})$ tijd werken, en
3. Slechts een $O(1)$ hoeveelheid nieuwe instanties van L produceren, en
4. De geproduceerde instanties van L zijn allemaal hooguit $O(n)$ groot.

Waarom reducties?

Reducties zijn een middel, niet een doel. Het doel is om snelle algoritmes te vinden voor problemen.
Vaak gaat een stuk onderzoek zo:

1. Probeer om Probleem A op te lossen
2. Lukt niet
3. Je ontdekt Probleem B. Probleem A lijkt op Probleem B
4. Probleem B deze is aantrekkelijker, want
   - 4.1 Probleem B is conceptueel simpeler
   - 4.2 Er is al een snel algoritme voor Probleem B
   - 4.3 Probleem B is in een vakgebied dat je beter beheerst
   - 4.4 ...
5. Je wilt beweren dat Probleem B oplossen nuttig is voor het oplossen van Probleem A
6. Je maakt een reductie van Probleem A naar Probleem B

Halting Problem