Forelesning 14

Om du står overfor et NP-komplett problem, så er det viktig å kunne bevise det. Vi har sett at alt i NP kan reduseres til CIRCUIT-SAT direkte, men vi kan også vise kompletthet indirekte, ved å redusere videre fra CIRCUIT-SAT!

Pensum

- ☐ 34. NP-completeness: 34.4 og 34.5

Læringsmål

Fra pensumheftet

- [N_1] Forstå hvordan NP-komplett-het kan bevises ved én reduksjon
- [N_2] Kjenne CIRCUIT-SAT, SAT, 3-CNF-SAT, CLIQUE, VERTEX-COVER, HAM-CYCLE, TSP og SUBSET-SUM
- [N_3] Forstå NPC-bevisene for disse
- [N_4] Forstå at det binære ryggsekkproblemet er NPH
- [N_5] Forstå at *lengste enkle vei* er NPH
- [N_6] Kjenne reduksjonsstrategier og kunne konstruere enkle NPH- og NPC-bevis

mlh@ntnu.no

Forelesningen filmes

s.ntnu.no/video-opptak

Mer algoritmer?

- **Søk studassjobb etter jul**
  `algdat.idi.ntnu.no/studass.html`
- **Ta TDT4125 Algoritmekonstruksjon**
  Bestått/ikke bestått!
- **Data: Velg «Effektive datasystemer»**
  Kanskje integrert PhD?

Forelesning 14

NP-komplette problemer

1. **CIRCUIT-SAT**
2. **SAT**
3. **3-CNF-SAT**
4. **CLIQUE**
5. **VERTEX-COVER**
6. **HAM-CYCLE**
7. **TSP**
8. **SUBSET-SUM**

Litt repetisjon

$L \in \mathbf{NPC}$

Hvordan viser vi at L er **NP**-komplett?

mlh@ntnu.no

- Vis at L $\in$ **NP**

At sertifikat for ja-svar kan verifiseres i pol. tid

mlh@ntnu.no

- Vis at $L \in \mathbf{NP}$
- Velg et kjent **NP**-komplett språk L'

mlh@ntnu.no

- Vis at $\mathrm{L} \in \mathbf{NP}$
- Velg et kjent **NP**-komplett språk L'
- Beskriv en algoritme som beregner en funksjon

$$f : \{0,1\}^* \rightarrow \{0,1\}^*$$

som transformerer instanser av L' til instanser av L

Dette er altså reduksjonen fra L' til L, som viser $\mathrm{L}' \leqslant_{\mathbf{P}} \mathrm{L}$

mlh@ntnu.no

- Vis at $\mathrm{L} \in \mathbf{NP}$
- Velg et kjent **NP**-komplett språk L'
- Beskriv en algoritme som beregner en funksjon

$$f : \{0,1\}^* \to \{0,1\}^*$$

som transformerer instanser av L' til instanser av L

- Vis at

$$x \in \mathrm{L}' \iff f(x) \in \mathrm{L}\,,$$

for alle $x \in \{0,1\}^*$

Vi må sørge for at vi får samme svar for $f(x)$

mlh@ntnu.no

- Vis at $\mathrm{L} \in \mathbf{NP}$
- Velg et kjent **NP**-komplett språk L'
- Beskriv en algoritme som beregner en funksjon

$$f : \{0,1\}^* \to \{0,1\}^*$$

  som transformerer instanser av L' til instanser av L
- Vis at

$$x \in \mathrm{L}' \iff f(x) \in \mathrm{L}\,,$$

  for alle $x \in \{0,1\}^*$
- Vis at algoritimen som beregner f har polynomisk kjøretid

mlh@ntnu.no

En fotnote

om NP-hardhet

- Boka bruker samme type reduksjoner for **NP**-hardhet

Polynomiske avbildningsreduksjoner: $x \in \mathrm{L}' \Leftrightarrow f(x) \in \mathrm{L}$

mlh@ntnu.no

- Boka bruker samme type reduksjoner for **NP**-hardhet
- Det begrenser oss til beslutningsproblemer . . .

(Siden vi ikke får transformere svaret)

mlh@ntnu.no

- Boka bruker samme type reduksjoner for **NP**-hardhet
- Det begrenser oss til beslutningsproblemer . . .
- . . . men gir oss altså $\mathbf{NPC} = \mathbf{NP} \cap \mathbf{NPH}$

mlh@ntnu.no

- Boka bruker samme type reduksjoner for **NP**-hardhet
- Det begrenser oss til beslutningsproblemer . . .
- . . . men gir oss altså $\mathbf{NPC} = \mathbf{NP} \cap \mathbf{NPH}$
- Det er vanlig med mer generelle reduksjoner for **NPH** . . .

mlh@ntnu.no

- Boka bruker samme type reduksjoner for **NP**-hardhet
- Det begrenser oss til beslutningsproblemer . . .
- . . . men gir oss altså $\mathbf{NPC} = \mathbf{NP} \cap \mathbf{NPH}$
- Det er vanlig med mer generelle reduksjoner for **NPH** . . .
- . . . men ikke for **NPC**!

Ukjent om man da ender med flere **NP**-komplette problemer!

mlh@ntnu.no

- Boka bruker samme type reduksjoner for **NP**-hardhet
- Det begrenser oss til beslutningsproblemer . . .
- . . . men gir oss altså $\mathbf{NPC} = \mathbf{NP} \cap \mathbf{NPH}$
- Det er vanlig med mer generelle reduksjoner for **NPH** . . .
- . . . men ikke for **NPC**!
- Løser man et slikt **NPH**-problem, har man uansett $\mathbf{P} = \mathbf{NP}$

Altså et som er **NP**-hardt under generelle polynomiske reduksjoner

mlh@ntnu.no

- Boka bruker samme type reduksjoner for **NP**-hardhet
- Det begrenser oss til beslutningsproblemer . . .
- . . . men gir oss altså $\mathbf{NPC} = \mathbf{NP} \cap \mathbf{NPH}$
- Det er vanlig med mer generelle reduksjoner for **NPH** . . .
- . . . men ikke for **NPC**!
- Løser man et slikt **NPH**-problem, har man uansett $\mathbf{P} = \mathbf{NP}$

(For **NPH** definert slik er det altså ukjent om $\mathbf{NPC} = \mathbf{NP} \cap \mathbf{NPH}$)

mlh@ntnu.no

Se delkapittel 34.5.6, «Reduction strategies» for generelle råd

Her ser vi på noen spesifikke eksempler

Merk: Det kreves ikke grundig forståelse av de ulike NP-kompletthetsbevisene

Det viktigste er å skjønne problemene, og at dere får se eksempler på reduksjoner

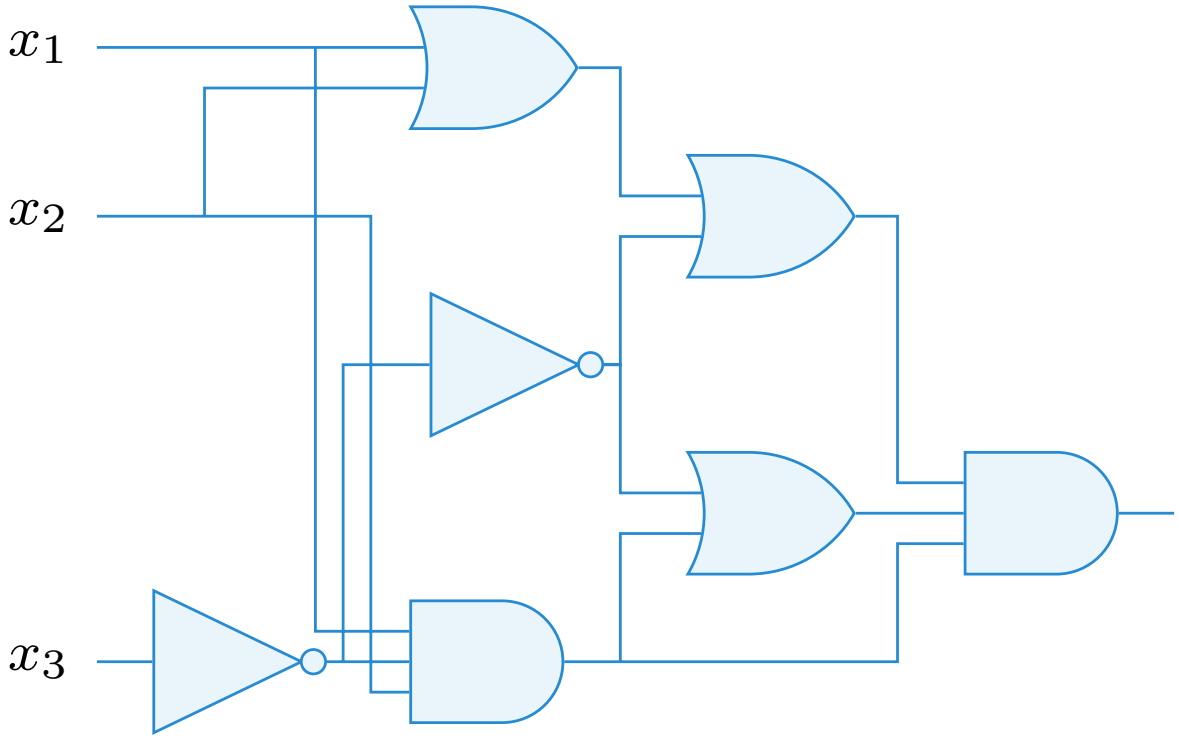
Verifikasjon er stort sett opplagt

Hvis vi lurer på om X finnes, så kan X fungere som sertifikat

Og kan vi sjekke X i polynomisk tid, er problemet altså i NP

1:8

CIRCUIT-SAT



CIRCUIT-SAT

Instans: En krets med logiske porter og én utverdi

Spørsmål: Kan utverdien bli 1?

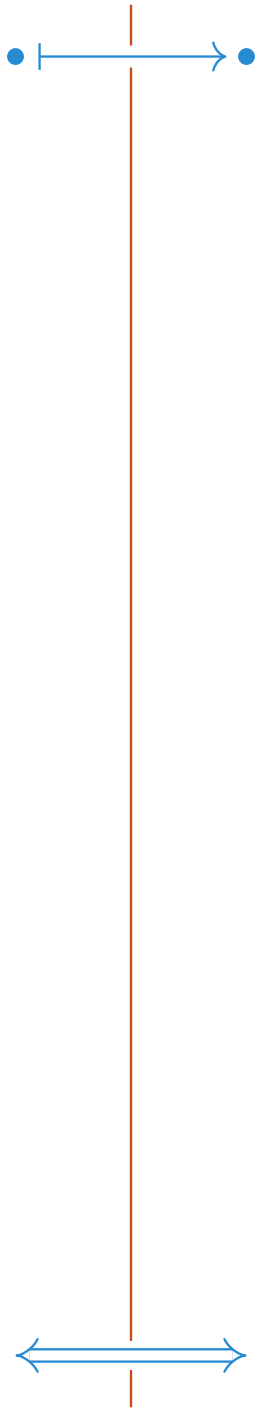
mlh@ntnu.no

Mulig sertifikat: Innverdier

Sjekk at de gir utverdi 1

- Vi har et vilkårlig språk/problem L $\in$ **NP**
- Vi vil redusere dette til CIRCUIT-SAT
- Det eneste vi vet er at $x \in$ L kan verifiseres i polynomisk tid
- Vi simulerer trinnene i verifikasjonsalgoritmen A med kretser!
- Spørsmålet blir: Kan A (for et eller annet sertifikat) svare 1?

$$x \in \{0,1\}^*$$



Er x med i språket L?

$\Longleftrightarrow$

Kan utverdien bli 1?

mlh@ntnu.no

L er i **NP**, så . . .

Det finnes en pol. alg. A, som er slik at

› $x \in \mathrm{L}$

nøyaktig når minst én $y \in \{0,1\}^*$ gir

› $\mathrm{A}(x, y) = 1$,

der $|y| = \mathrm{O}(|x|^c)$, for en eller annen c.

Er x med i språket L? $\Longleftrightarrow$ Kan utverdien bli 1?

mlh@ntnu.no

L er i **NP**, så . . .

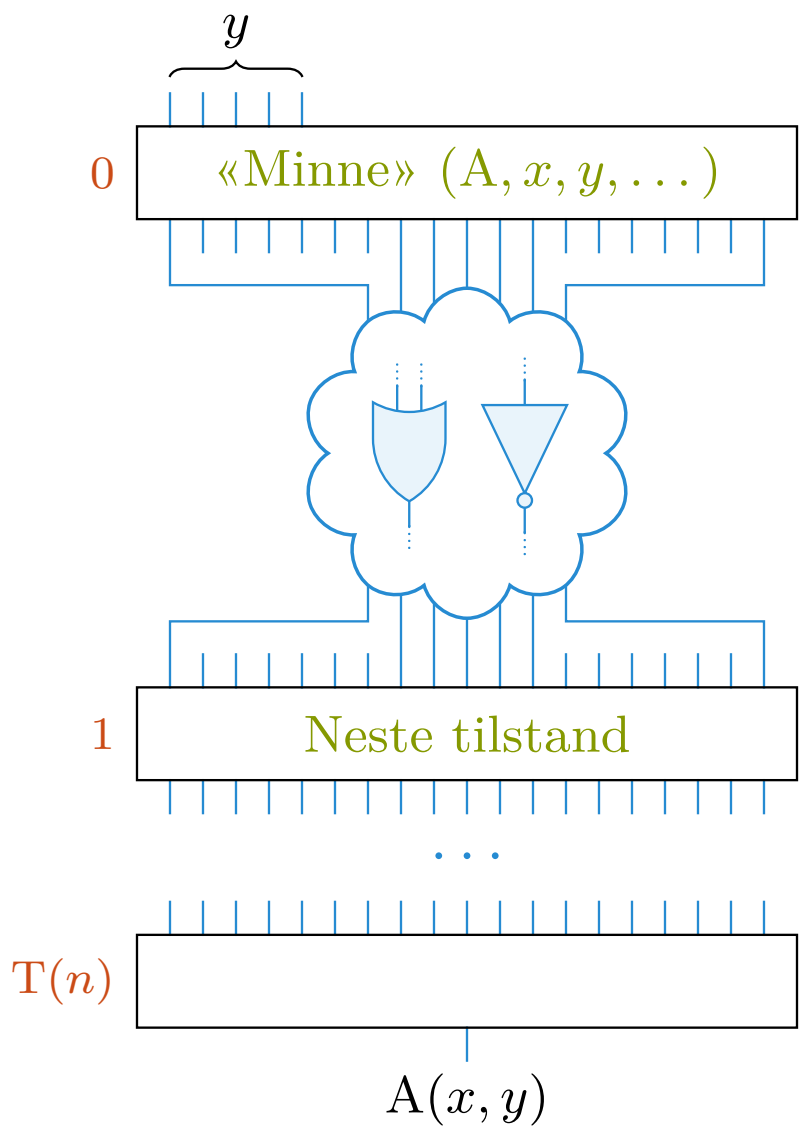
Det finnes en pol. alg. A, som er slik at

› $x \in \mathrm{L}$

nøyaktig når minst én $y \in \{0,1\}^*$ gir

› $\mathrm{A}(x, y) = 1$,

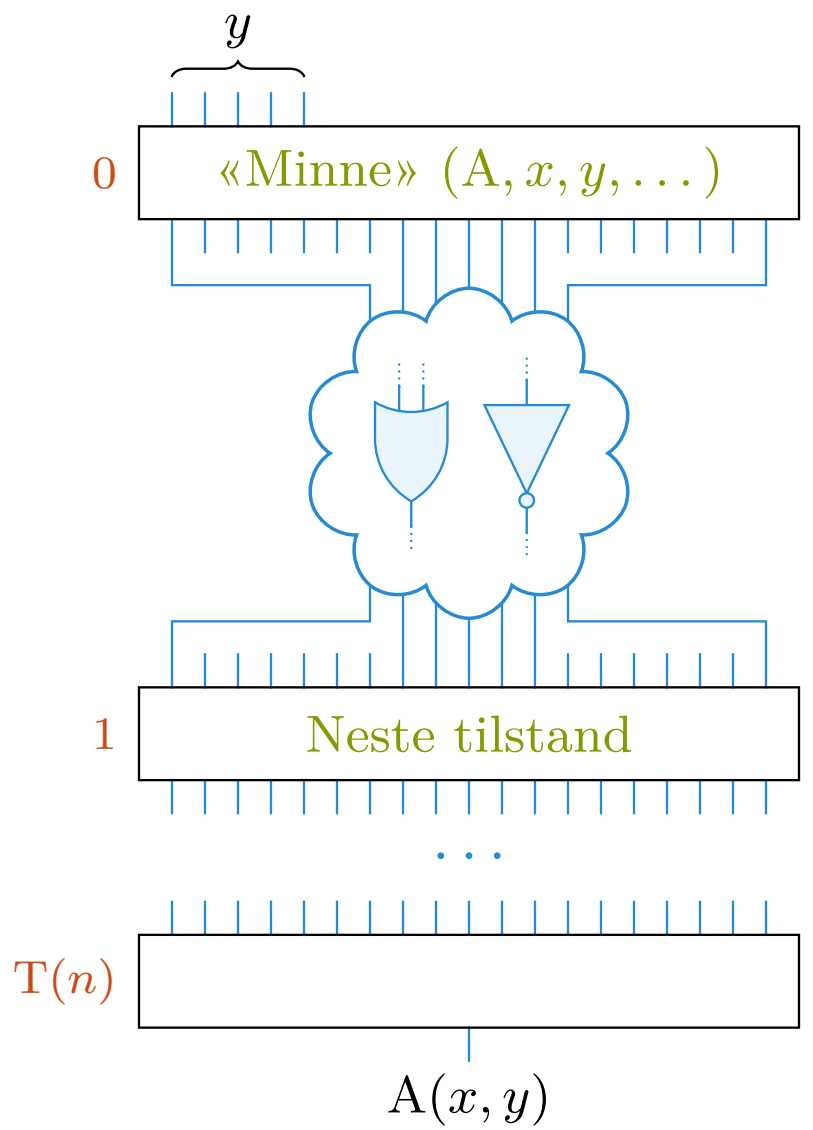
der $|y| = \mathrm{O}(|x|^c)$, for en eller annen c.



Er x med i språket L? ⟺ Kan utverdien bli 1?

mlh@ntnu.no

$$x \in \{0,1\}^*$$



Er x med i språket L? $\Longleftrightarrow$ Kan utverdien bli 1?

mlh@ntnu.no

2:8

SAT

$$\phi = ((x_1 \rightarrow x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$$

SAT

Instans: En logisk formel

Spørsmål: Kan formelen være sann?

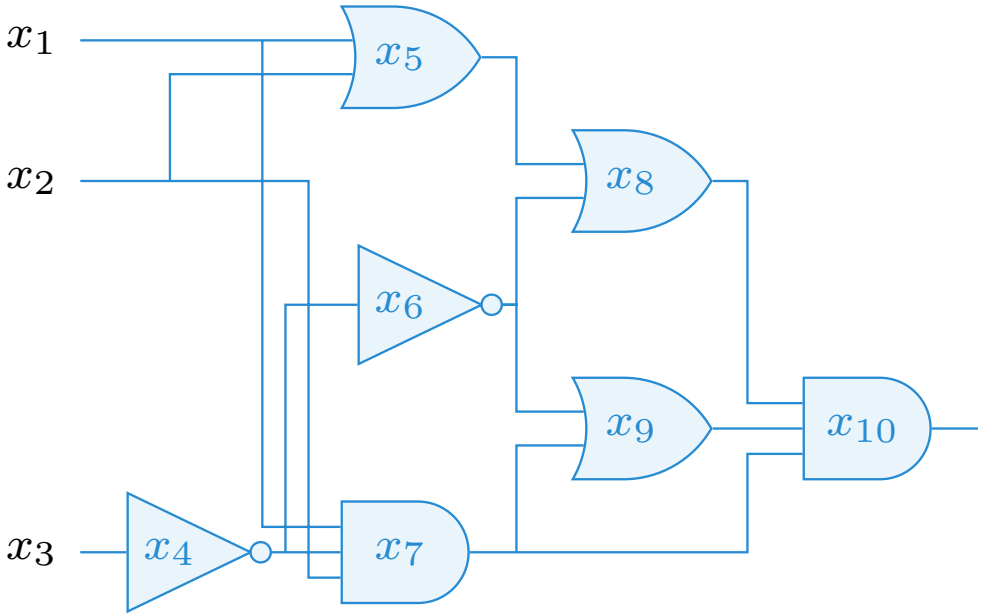
mlh@ntnu.no

Mulig sertifikat: Sannhetsverdier for variable

Sjekk at formelen blir sann

› Direkte oversettelse av logisk krets?

- Direkte oversettelse av logisk krets?
- Kan gi eksponentielt stor formel!



$$\begin{aligned}\phi = x_{10} \;&\wedge\; (x_4 \leftrightarrow \neg x_3)\\ &\wedge\; (x_5 \leftrightarrow (x_1 \vee x_2))\\ &\wedge\; (x_6 \leftrightarrow \neg x_4)\\ &\wedge\; (x_7 \leftrightarrow (x_1 \wedge x_2 \wedge x_4))\\ &\wedge\; (x_8 \leftrightarrow (x_5 \vee x_6))\\ &\wedge\; (x_9 \leftrightarrow (x_6 \vee x_7))\\ &\wedge (x_{10} \leftrightarrow (x_7 \wedge x_8 \wedge x_9))\end{aligned}$$

Kan utverdien bli 1? $\iff$ Kan ϕ være sann?

mlh@ntnu.no

3:8

3-CNF-SAT

CNF-form

Konjunksjon av disjunksjoner av literaler

For eksempel $(x \vee y) \wedge (\neg z \vee \neg y \vee w \vee u) \wedge (\neg v \vee \neg x \vee \neg z)$

mlh@ntnu.no

3-CNF-form

Konjunksjon av disjunksjoner av 3 literaler

For eksempel $(x \vee y \vee \neg z) \wedge (\neg y \vee w \vee u) \wedge (\neg v \vee \neg x \vee \neg z)$

mlh@ntnu.no

$$\begin{aligned}\phi = (\ & x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \\ (\neg & x_1 \vee x_2 \vee x_3\ ) \wedge \\ (\ & x_1 \vee x_2 \vee x_3\ )\end{aligned}$$

3-CNF-SAT

Instans: En logisk formel på 3-CNF-form

Spørsmål: Kan formelen være sann?

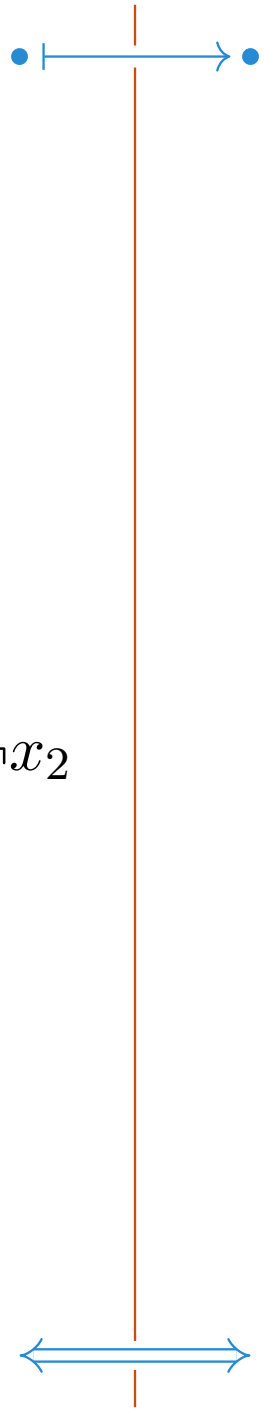
mlh@ntnu.no

Mulig sertifikat: Sannhetsverdier for variable

Sjekk at formelen blir sann

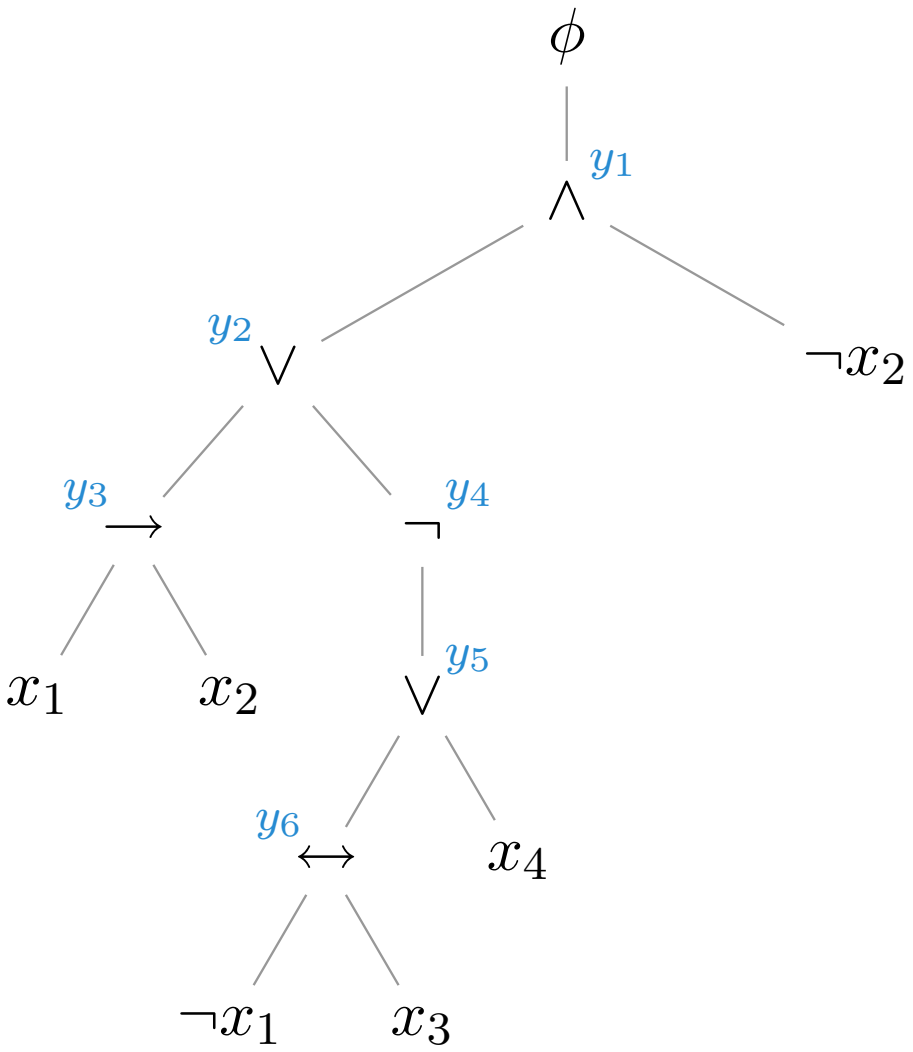
- Vi kan bruke ca. samme reduksjon, på syntakstreet til ϕ!
- Vi får da en formel ϕ' av pol. størrelse
- ϕ' er en konjunksjon av ledd med maks 3 literaler
  - Dvs.: de to argumentene, samt resultatet av operatoren
- Hvert ledd gjøres om til CNF vha. en sannhetstabell
- CNF $\phi'' \rightarrow$ 3-CNF ϕ''':
  - $(x \vee y)$ gjøres om til $(x \vee y \vee z) \wedge (x \vee y \vee \neg z)$
  - Tilsvarende blir (x) til fire nye disjunksjoner

$$\phi = ((x_1 \rightarrow x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$$



Kan ϕ være sann? $\Longleftrightarrow$ Kan ϕ''' være sann?

mlh@ntnu.no



$$
\begin{aligned}
\phi' \;=\; & y_1 \wedge (y_1 \leftrightarrow (y_2 \wedge \neg x_2)) \\
& \wedge (y_2 \leftrightarrow (y_3 \vee y_4)) \\
& \wedge (y_3 \leftrightarrow (x_1 \rightarrow x_2)) \\
& \wedge (y_4 \leftrightarrow \neg y_5) \\
& \wedge (y_5 \leftrightarrow (y_6 \vee x_4)) \\
& \wedge (y_6 \leftrightarrow (\neg x_1 \leftrightarrow x_3))
\end{aligned}
$$

ϕ'' = CNF, vha. sannhetstabeller

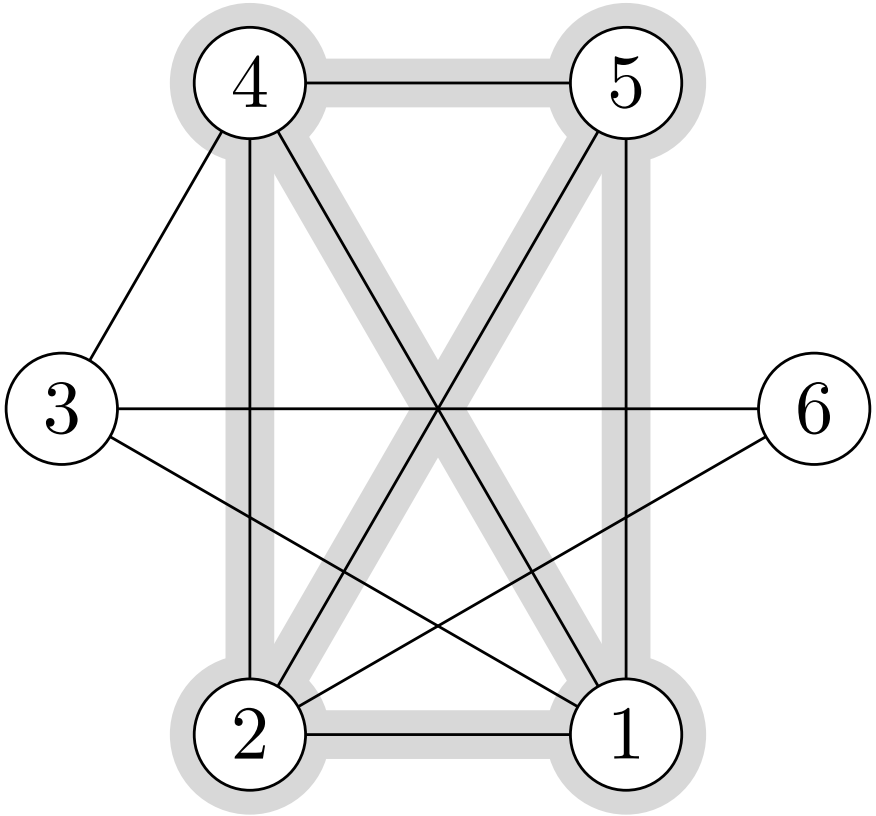
ϕ''' = 3-CNF, vha. dummy-variable

Kan ϕ være sann? $\Longleftrightarrow$ Kan ϕ''' være sann?

mlh@ntnu.no

4:8

CLIQUE



CLIQUE

Instans: En urettet graf G og et positivt heltall k

Spørsmål: Har G en en komplett delgraf med k noder?

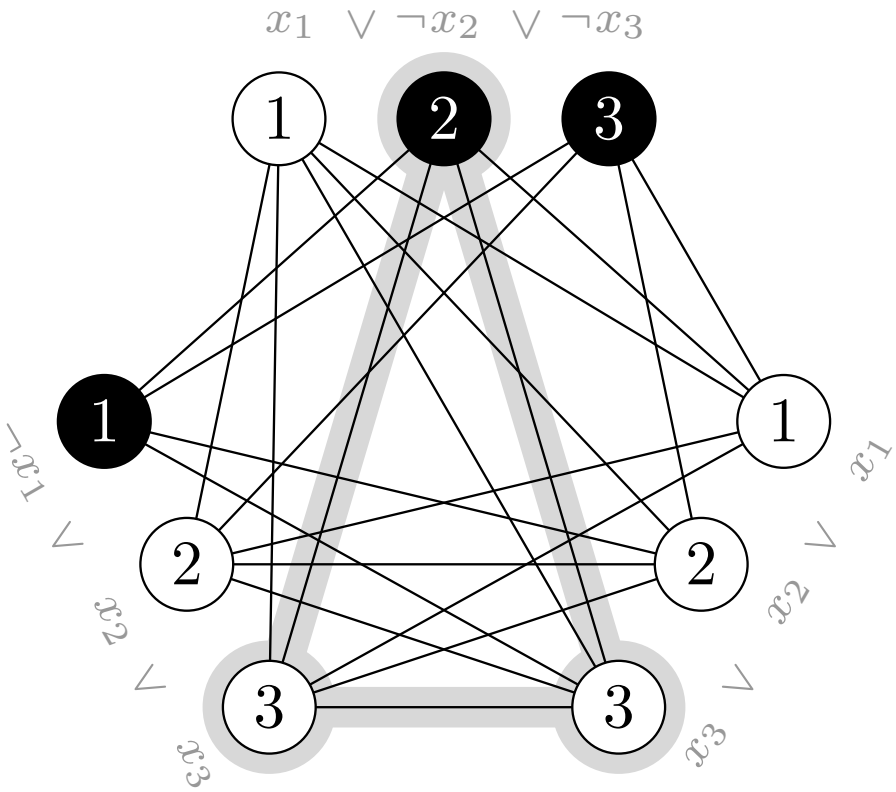
mlh@ntnu.no

Mulig sertifikat: Et sett med k noder

Sjekk at nodene utgjør en klikk

- Vi vil redusere fra 3-CNF-SAT
- Lag én node i G for hver literal i formelen
- Ingen kanter mellom literaler fra samme disjunksjon
- Ellers: Kanter mellom literaler som kan være sanne samtidig
- La k være antall disjunksjoner

$$\phi = (\ x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge (\neg x_1 \vee x_2 \vee x_3\ ) \wedge (\ x_1 \vee x_2 \vee x_3\ )$$



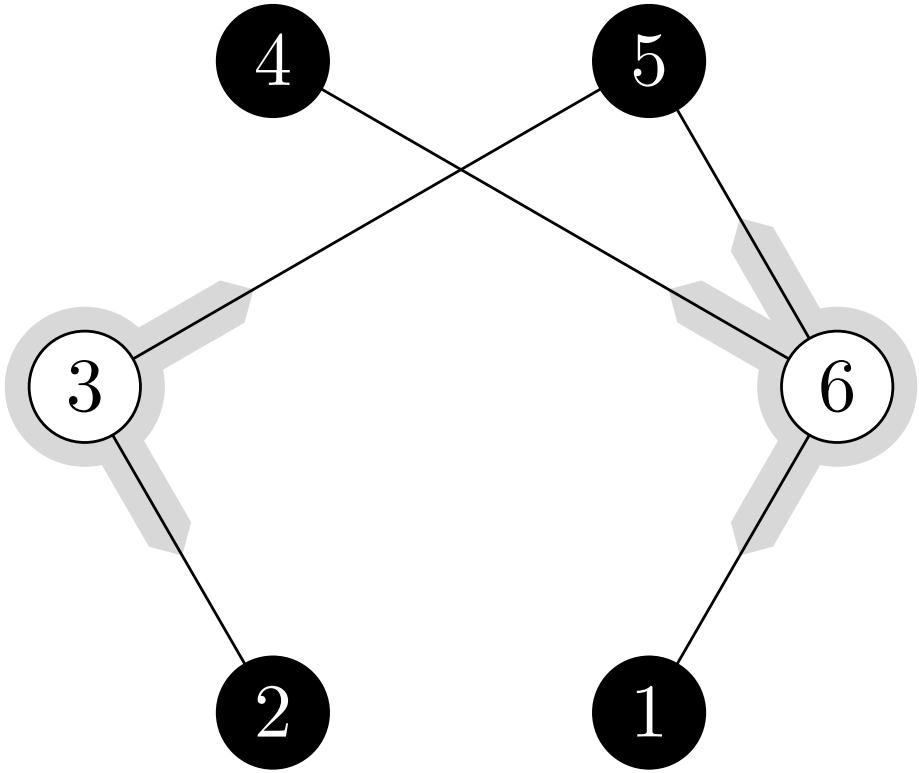
Tilsv. $x_1, x_2, x_3 = -, 0, 1$

Kan ϕ være sann? $\iff$ Finnes en k-klikk?

mlh@ntnu.no

5:8

VERTEX-COVER



VERTEX-COVER

Instans: En urettet graf G og et positivt heltall k

Spørsmål: Har G en et nodedekke med k noder?

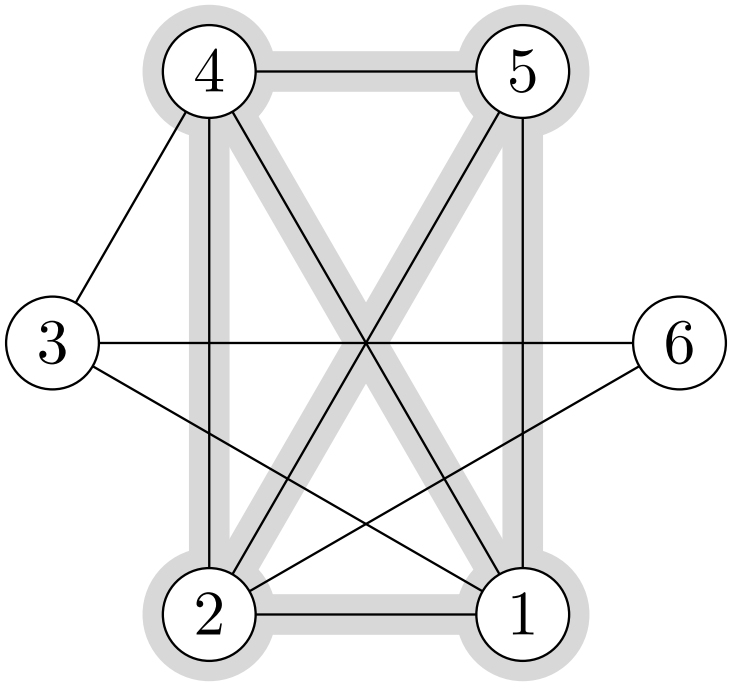
Dvs., k noder som tilsammen ligger inntil alle kantene

mlh@ntnu.no

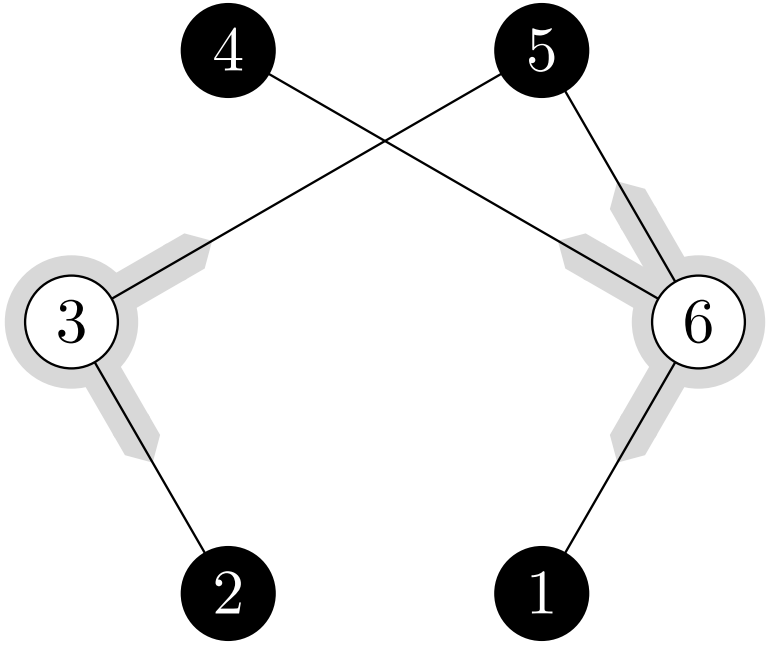
Mulig sertifikat: Et sett med k noder

Sjekk at nodene utgjør et nodedekke

- En klikk er en komplett delgraf
- Tilsvarer en uavhengig mengde (kantfri delgraf) i komplementet $\bar{\mathrm{G}} = (\mathrm{V}, \bar{\mathrm{E}})$
- Nodene utenfor en uavhengig mengde utgjør et nodedekke
- Hvis G har en k-klikk …
  - … så har $\bar{\mathrm{G}} = (\mathrm{V}, \bar{\mathrm{E}})$ en uavh. mengde med k noder …
  - … og dermed også et $(|\mathrm{V}| - k)$-nodedekke
- Samme resonnement holder i motsatt retning



G



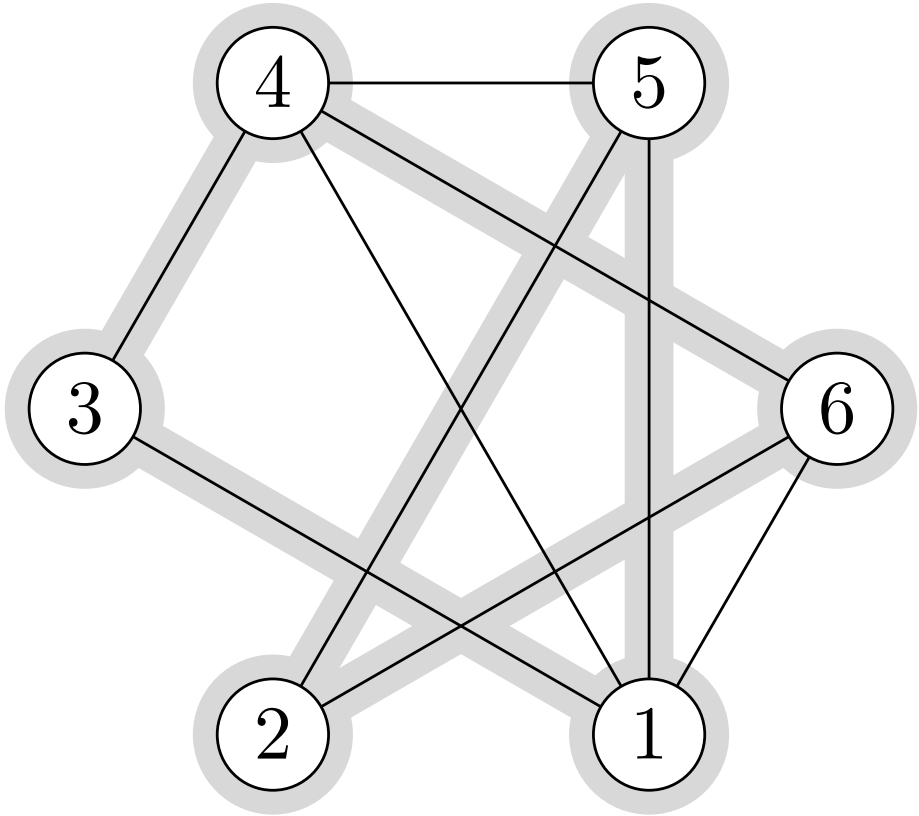
$\bar{\mathrm{G}}$

Finnes en k-klikk? $\Longleftrightarrow$ Finnes et $(|\mathrm{V}|-k)$-dekke?

mlh@ntnu.no

6:8

HAM-CYCLE



HAM-CYCLE

Instans: En urettet graf G

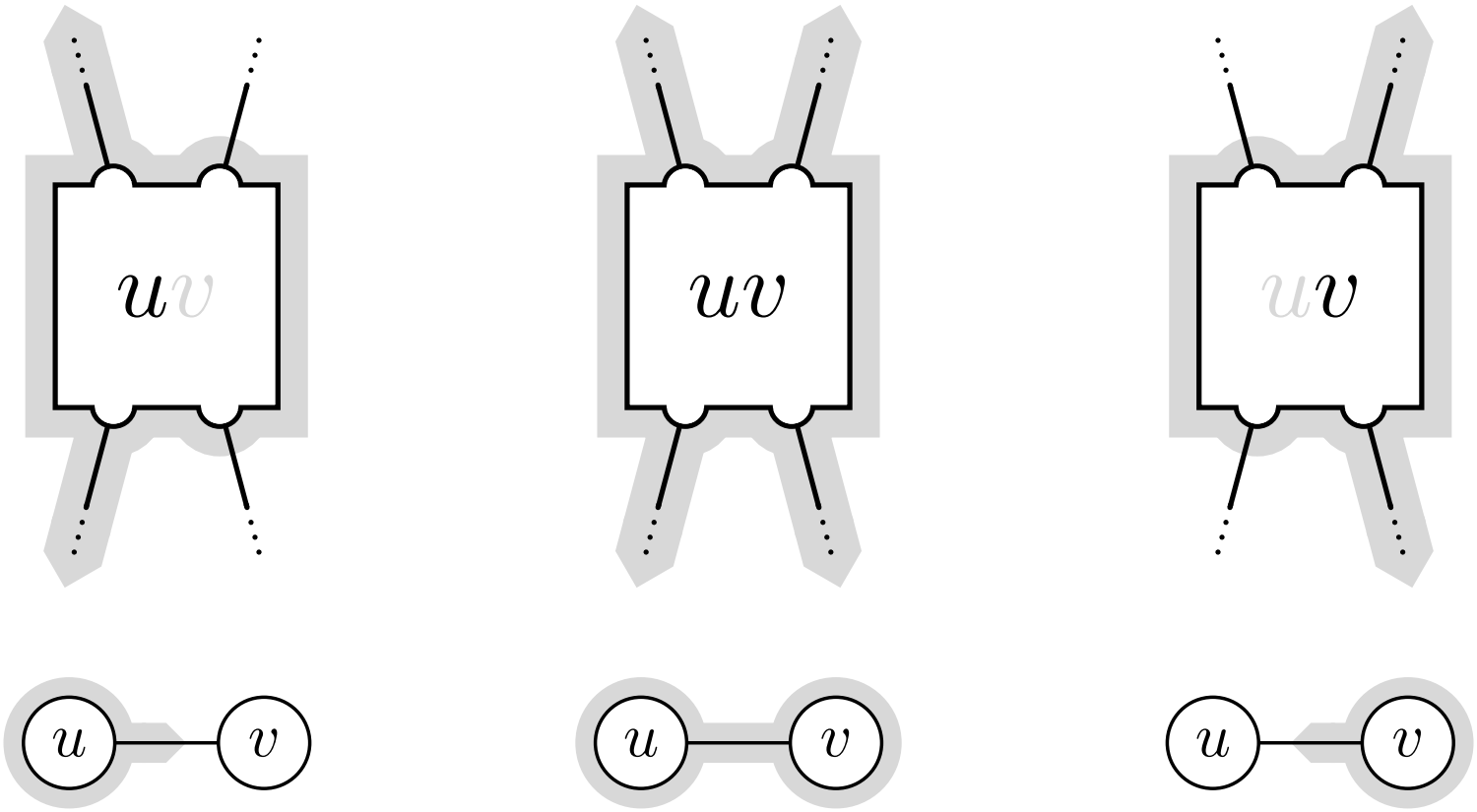
Spørsmål: Finnes det en sykel som inneholder alle nodene?

mlh@ntnu.no

Mulig sertifikat: En permutasjon av nodene

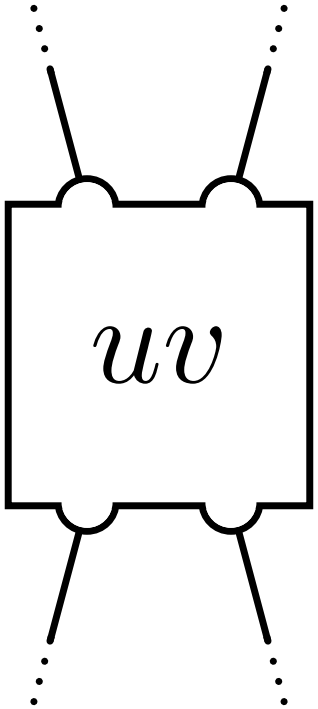
Sjekk at etterfølgende noder, inkl. siste og første, er naboer

- Vi reduserer fra VERTEX-COVER
- Sykelen må kunne velge noder til nodedekket
- En kant (u, v) kan dekkes av u eller v eller begge
- Vi lager oss en innretning (*widget*) som hjelper oss med dette



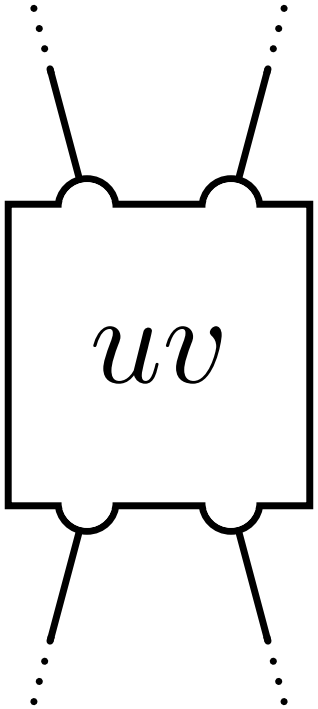
Om sykelen går igjennom på høyre side, velges v

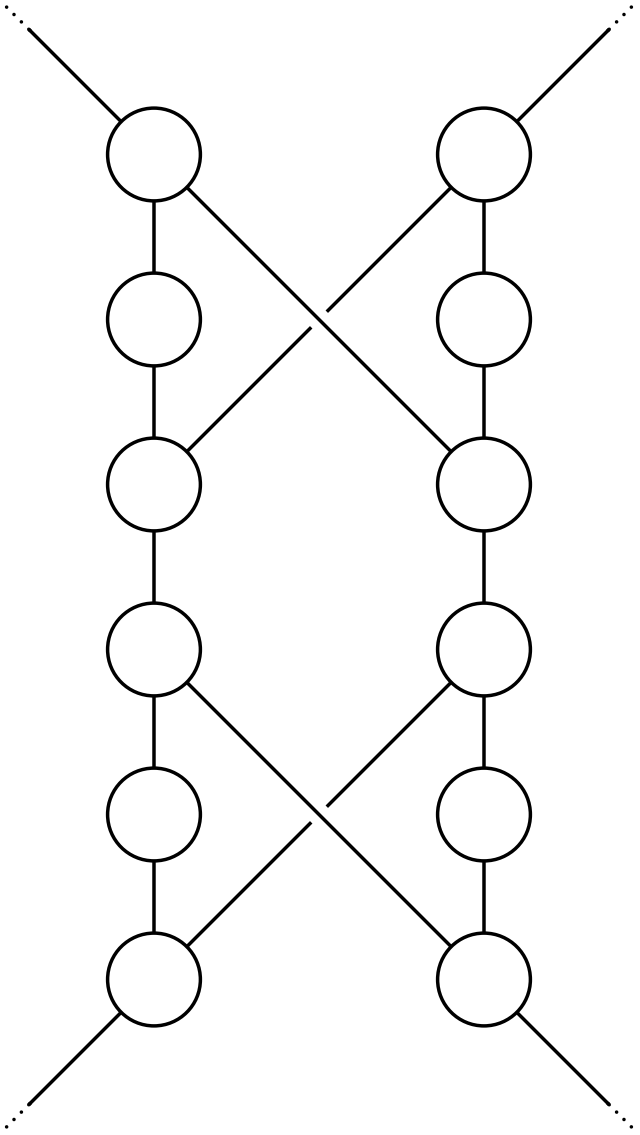
mlh@ntnu.no



Hvordan lager vi disse?

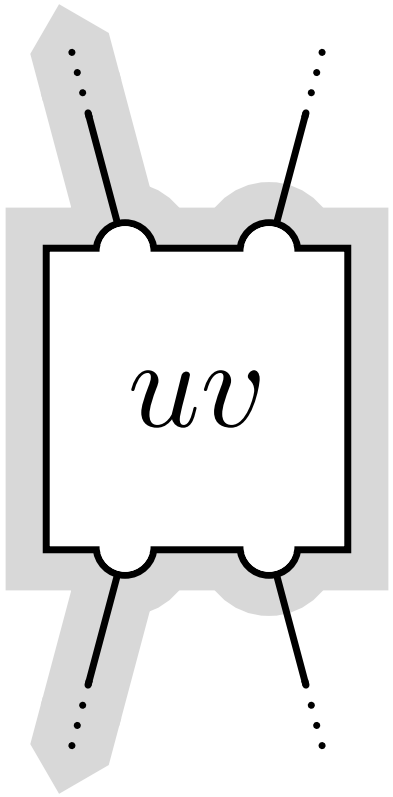
mlh@ntnu.no

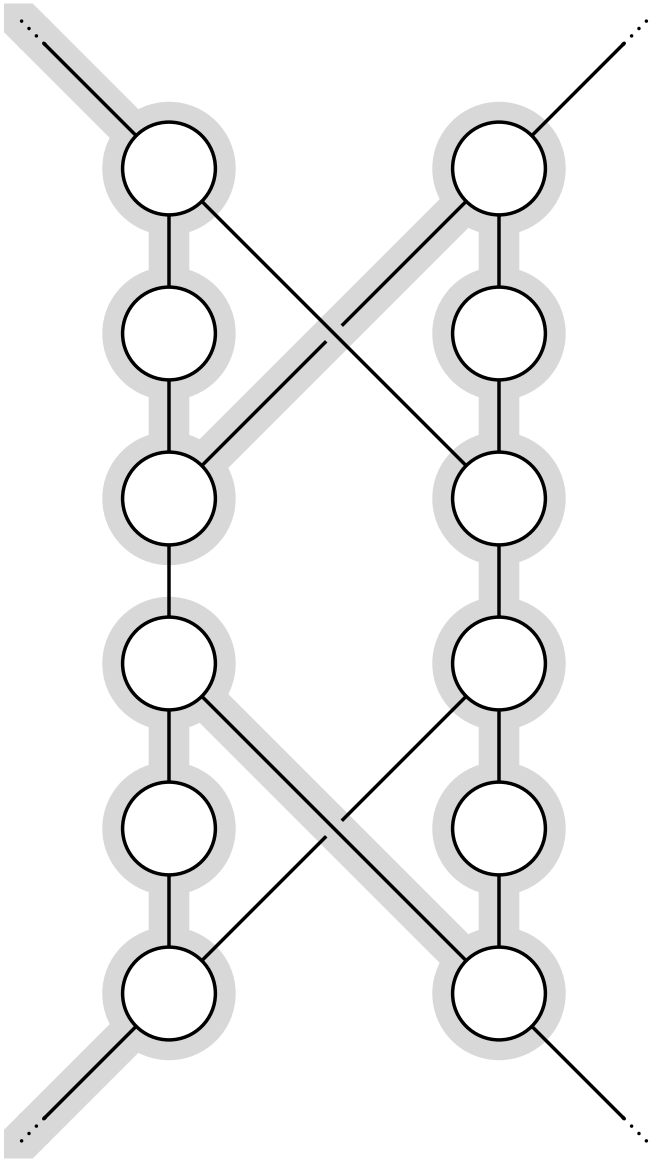




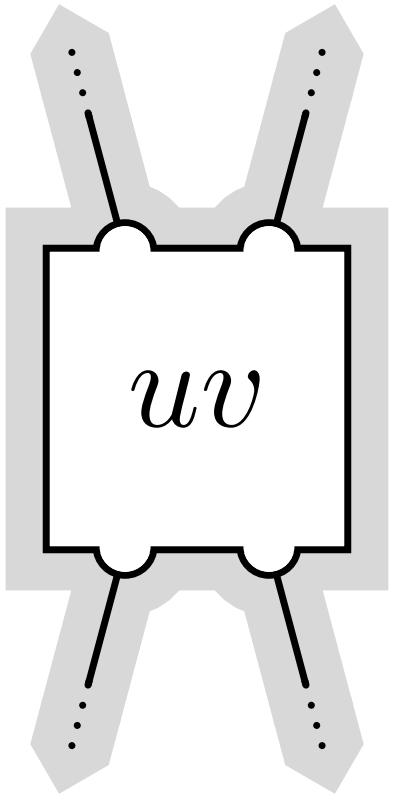
Kan uansett besøke alle

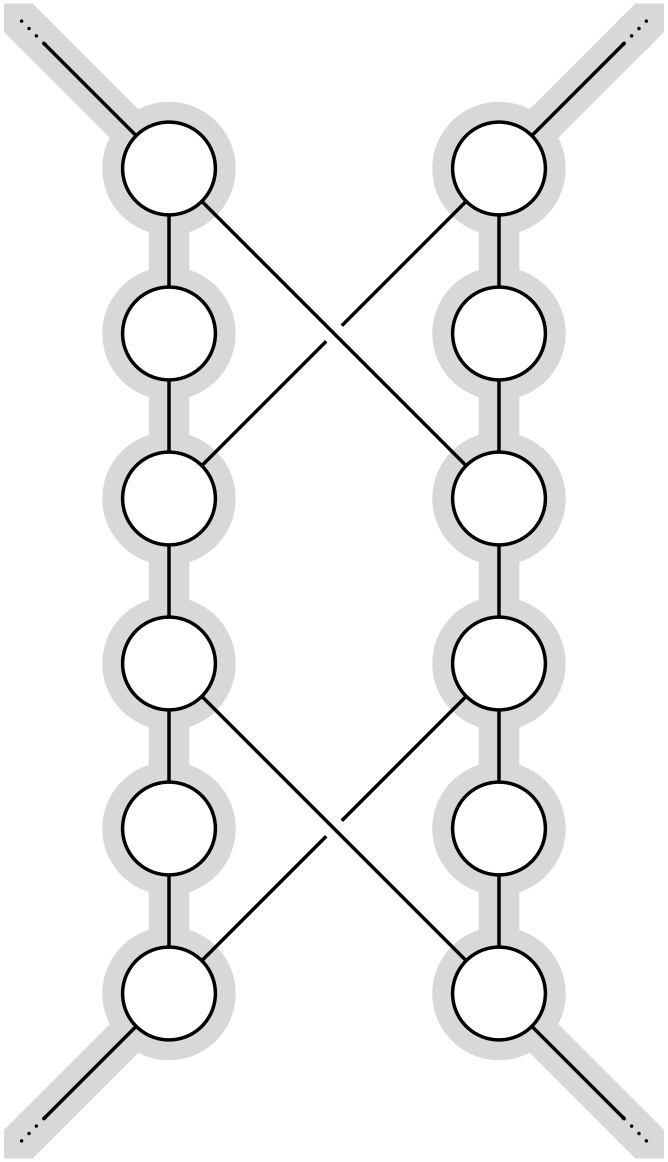
mlh@ntnu.no



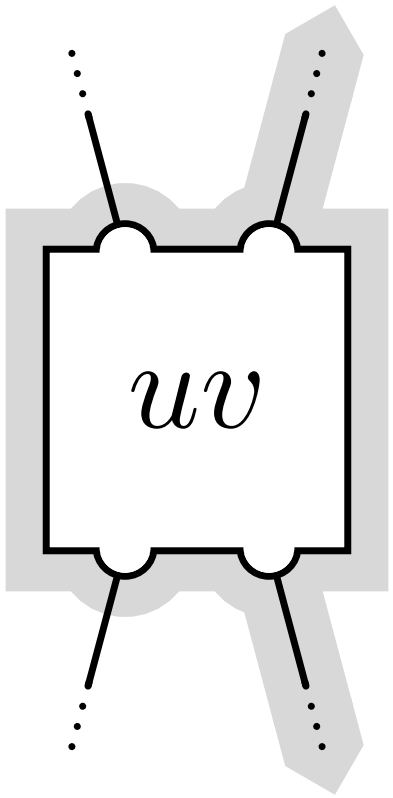


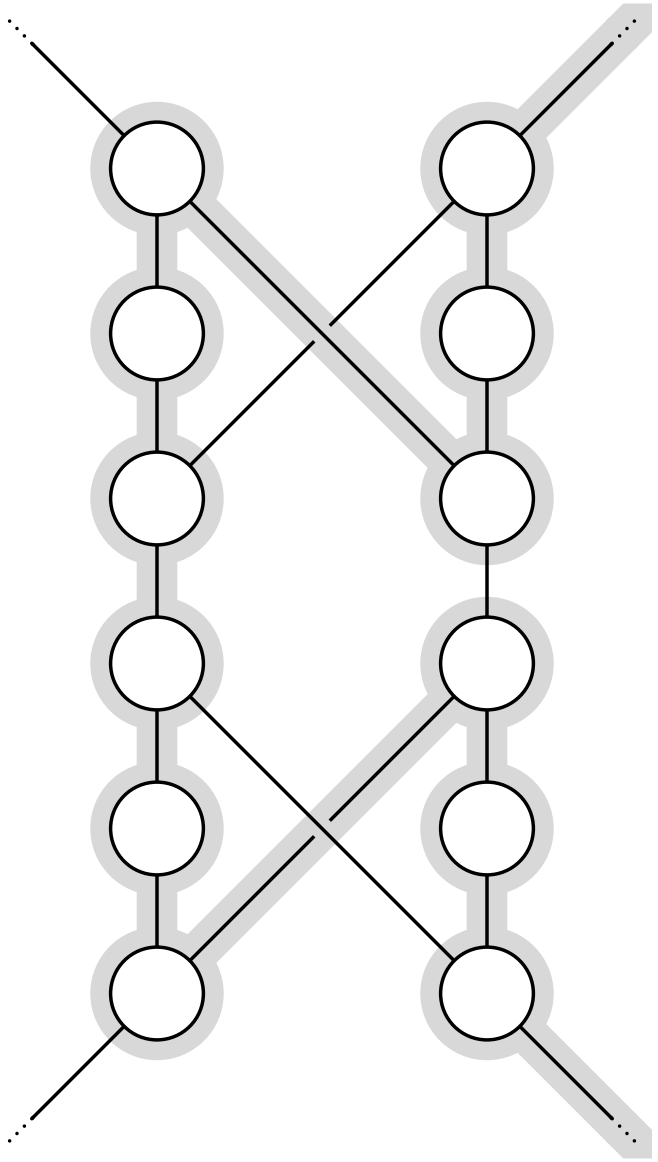
mlh@ntnu.no





mlh@ntnu.no





mlh@ntnu.no

› Lag «nabolister» ved hjelp av innretningene våre

Husk: Hver innretning representerer én av kantene i grafen

- Lag «nabolister» ved hjelp av innretningene våre
  - Ordne utkanter for hver originale node vilkårlig

Vi lager oss en liste med ut-kanter (altså av innretninger) for v

- Lag «nabolister» ved hjelp av innretningene våre
  - Ordne utkanter for hver originale node vilkårlig
  - Koble nederste innretningsnode til øverste for neste

På den siden av innretningen som representerer v

- Lag «nabolister» ved hjelp av innretningene våre
  - Ordne utkanter for hver originale node vilkårlig
  - Koble nederste innretningsnode til øverste for neste
- Koble k «selektornoder» til første øverste og siste nederste

Selektornode i representerer den i-ende noden i nodedekket

- Lag «nabolister» ved hjelp av innretningene våre
  - Ordne utkanter for hver originale node vilkårlig
  - Koble nederste innretningsnode til øverste for neste
- Koble k «selektornoder» til første øverste og siste nederste
- Hver selektornode kan da velge seg en naboliste

Velger node og «dekker» kanter (sender sykel gjennom innretninger)

- Lag «nabolister» ved hjelp av innretningene våre
  - Ordne utkanter for hver originale node vilkårlig
  - Koble nederste innretningsnode til øverste for neste
- Koble k «selektornoder» til første øverste og siste nederste
- Hver selektornode kan da velge seg en naboliste
  - Den «velger» ved å sende hamiltonsykelen til den første, øverste innretningsnoden i nabolista

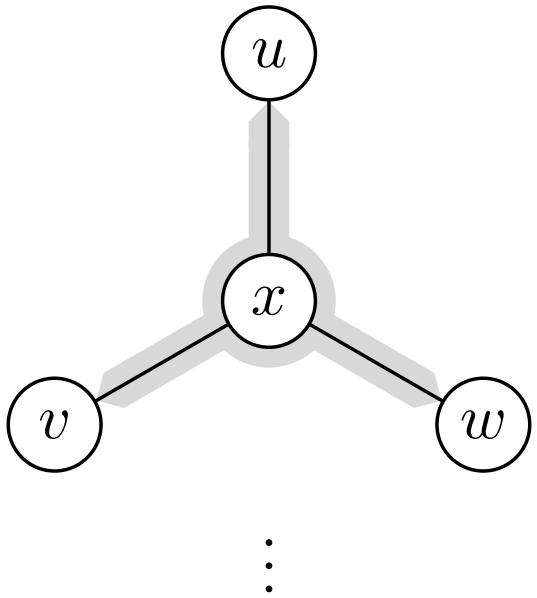
For v sin nabokant-liste: Går gjennom innretninger i lista på v-siden

- Lag «nabolister» ved hjelp av innretningene våre
  - Ordne utkanter for hver originale node vilkårlig
  - Koble nederste innretningsnode til øverste for neste
- Koble k «selektornoder» til første øverste og siste nederste
- Hver selektornode kan da velge seg en naboliste
  - Den «velger» ved å sende hamiltonsykelen til den første, øverste innretningsnoden i nabolista
- Første og siste innretning i lista er koblet til alle k selektornoder

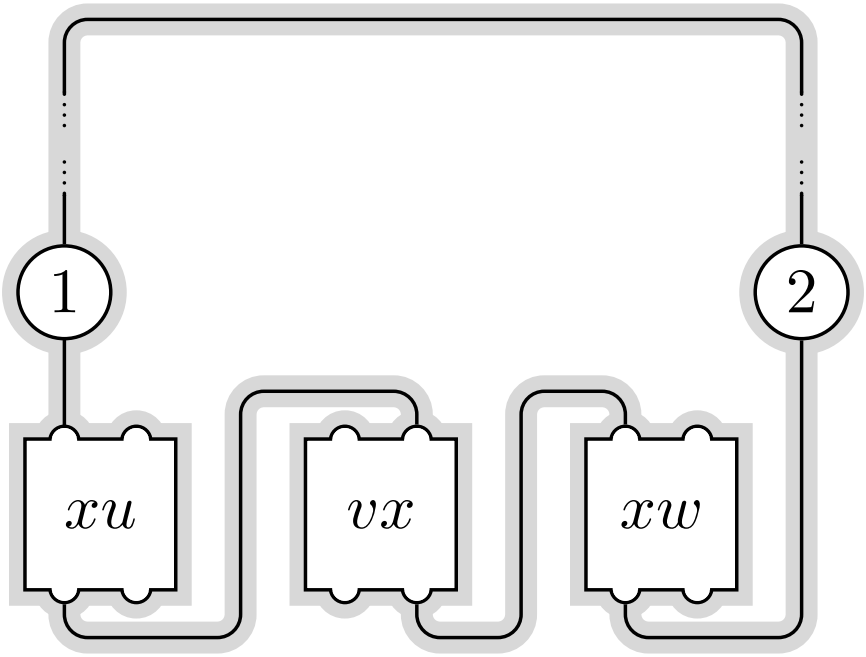
Siste innretning sender sykelen videre til neste (ev. første) selektornode

- Lag «nabolister» ved hjelp av innretningene våre
  - Ordne utkanter for hver originale node vilkårlig
  - Koble nederste innretningsnode til øverste for neste
- Koble k «selektornoder» til første øverste og siste nederste
- Hver selektornode kan da velge seg en naboliste
  - Den «velger» ved å sende hamiltonsykelen til den første, øverste innretningsnoden i nabolista
- Første og siste innretning i lista er koblet til alle k selektornoder
- Vi har en hamiltonsykel hvis og bare hvis hver naboliste velges

Vi er innom selektornodene, og går gjennom nabolister mellom dem

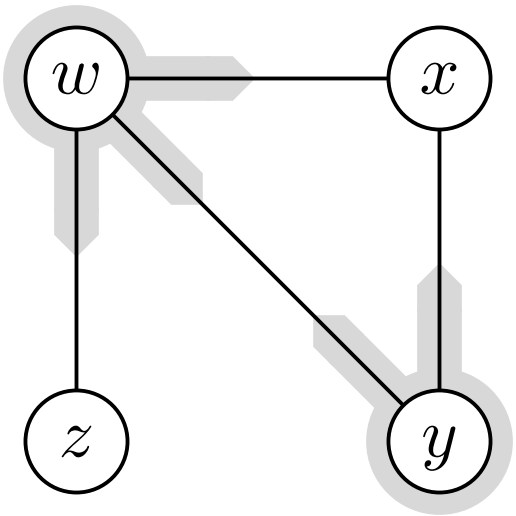


Dekker kanter xu, xv og xw

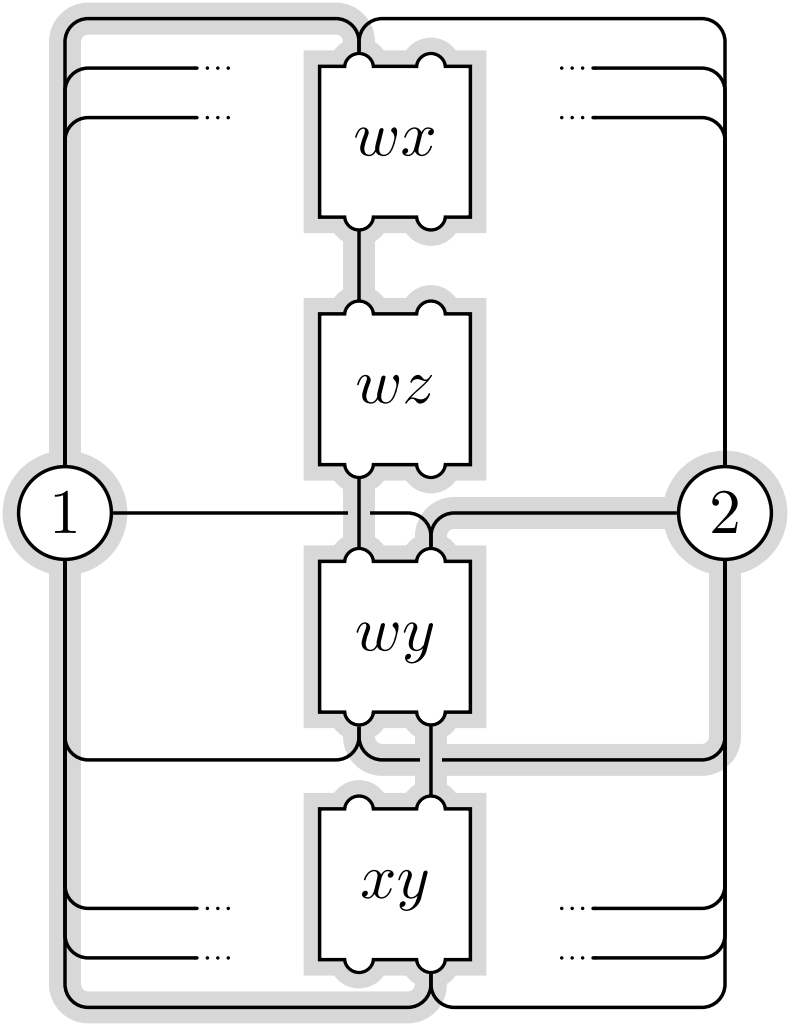


Sykel via tilsvarende widgets

mlh@ntnu.no



$k = 2$



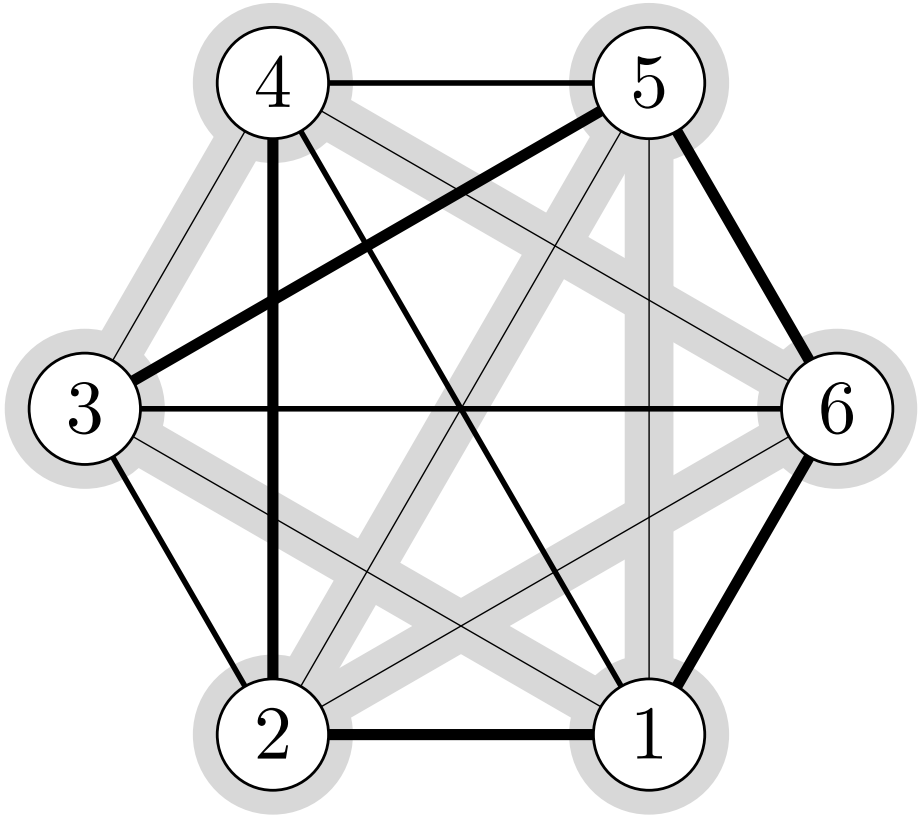
Finnes et k-dekke? $\Longleftrightarrow$ Finnes en hamiltonsykel?

mlh@ntnu.no

Reduksjon videre til lengste enkle vei så vi på forrige gang

7:8

TSP



TSP

Instans: Komplett graf med vekter i $\mathbb{N}$ og et tall $k \in \mathbb{N}$

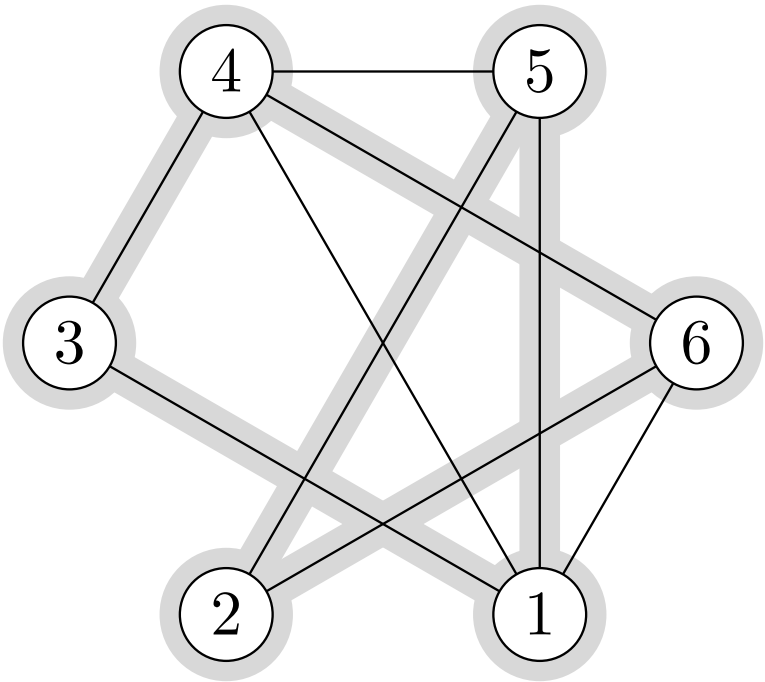
Spørsmål: Finnes det en rundtur med kostnad $\leqslant k$?

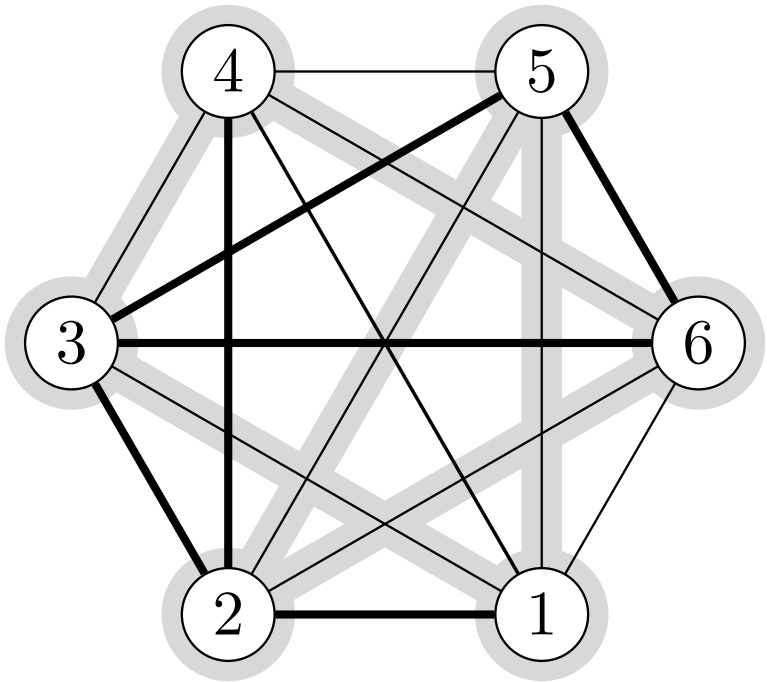
mlh@ntnu.no

Mulig sertifikat: En permutasjon av nodene

Sjekk at kantene mellom etterfølgende noder, inkl. siste og første, summerer til maks k

- Billigste hamiltonsykel!
- Trenger bare gjøre originalgrafen veldig billig





$c = 1$

$c = 0$

Finnes en hamiltonsykel? $\Longleftrightarrow$ Finnes tur m/kostnad 0?

mlh@ntnu.no

Fun fact: Det er ukjent om euklidsk TSP er i NP!

Altså TSP for punkter i planet

Mer generelt: Det er ukjent om vi kan sjekke om en sum av kvadratrøtter er under en viss verdi i polynomisk tid!

Se f.eks. http://blog.computationalcomplexity.org/2022/11/euclidean-tsp-is-np-hard-but-not-known.html

(Dette er altså fra 2022. Det kan jo være noen som har funnet ut av det siden …)

8:8

SUBSET-SUM

SUBSET-SUM

Instans: Mengde positive heltall S og positivt heltall t

Spørsmål: Finnes en delmengde av S med sum t?

mlh@ntnu.no

Mulig sertifikat: En delmengde

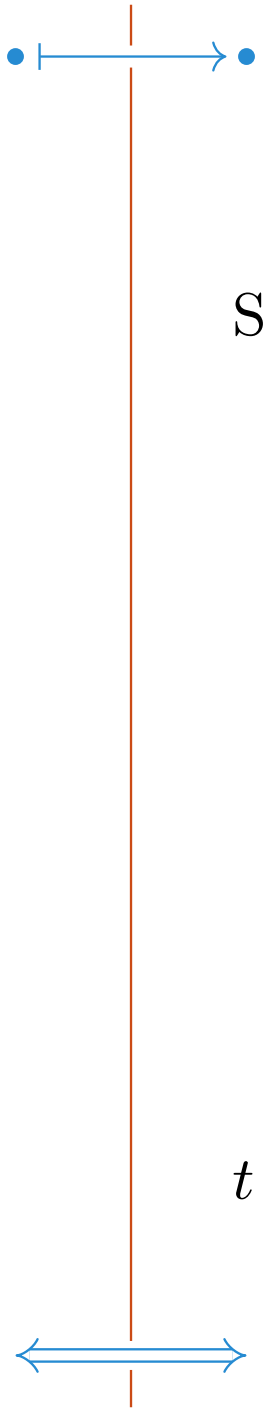
Sjekk at delmengden summerer til t

- Harmløse antagelser til reduksjonen:
  - Ingen disjunksjon inneholder både x og $\neg x$
  - Hver x forekommer i minst én disjunksjon

$$\begin{aligned}\phi = (\ & x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \\ (\ & \neg x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \\ (\ & \neg x_1 \vee \neg x_2 \vee x_3\ ) \wedge \\ (\ & x_1 \vee x_2 \vee x_3\ )\end{aligned}$$

Kan ϕ være sann? $\iff$ Har S en delsum t?

$$\phi = (\ x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge (\neg x_1 \vee \neg x_2 \vee x_3\ ) \wedge (\ x_1 \vee x_2 \vee x_3\ )$$

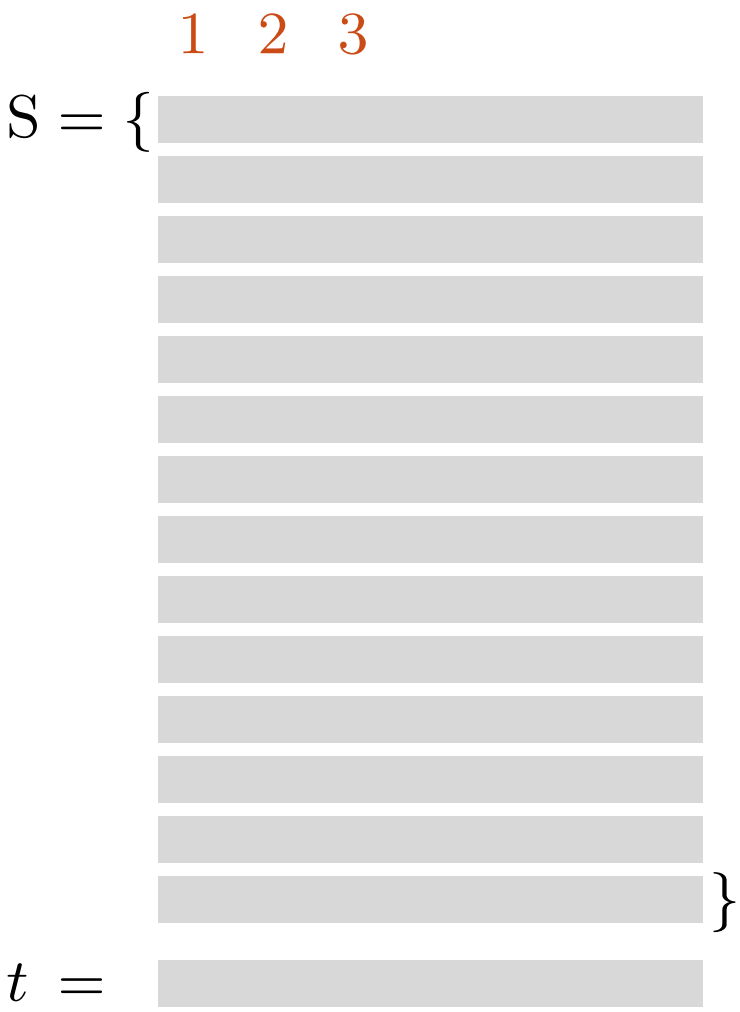


$$S = \{ \quad \}$$

$$t =$$

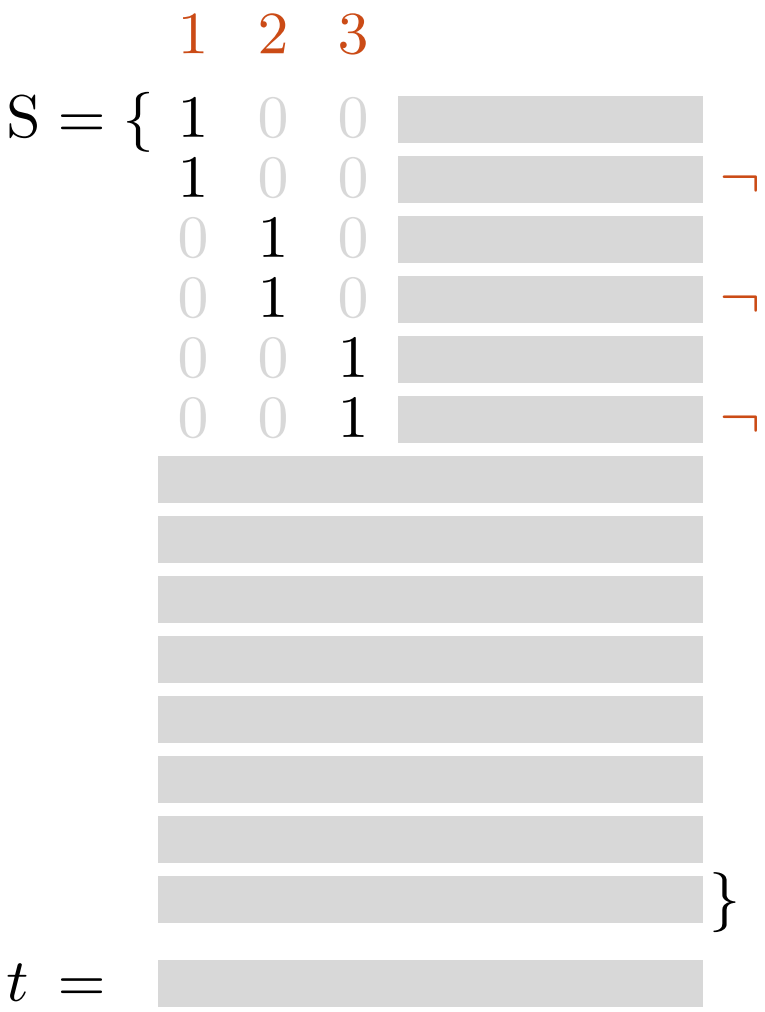
Kan ϕ være sann? $\iff$ Har S en delsum t?

$$\phi = (\ x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge (\neg x_1 \vee \neg x_2 \vee x_3\ ) \wedge (\ x_1 \vee x_2 \vee x_3\ )$$



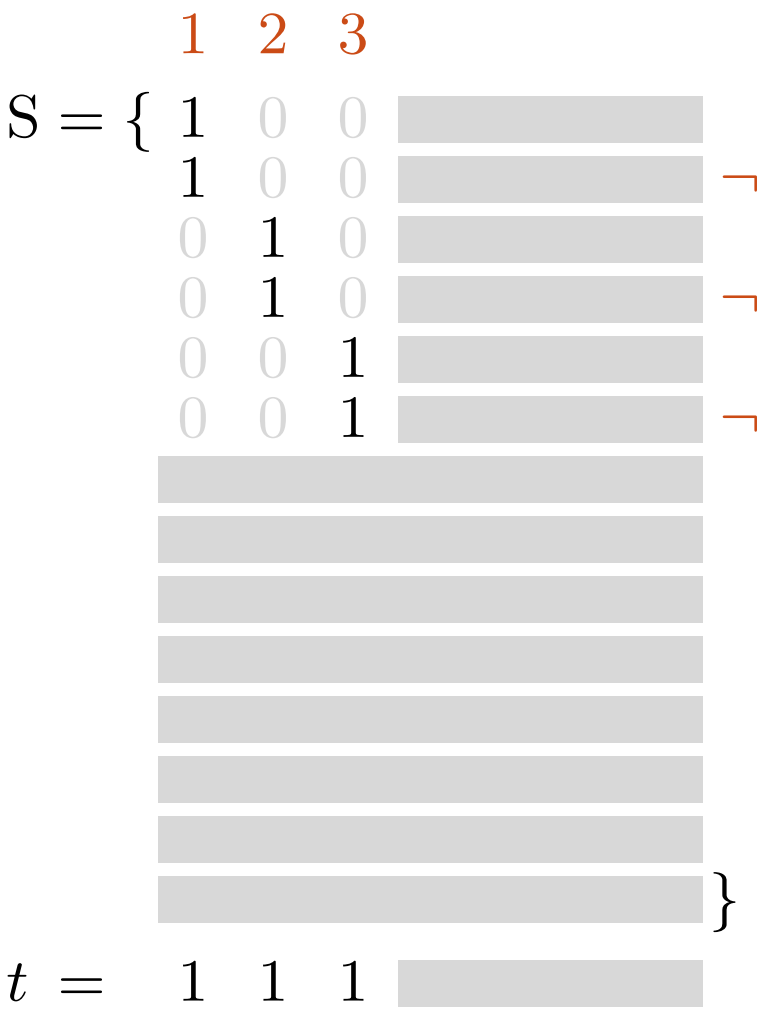
Vi innfører et siffer (en kolonne) per variabel

$$\phi = (\ x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge (\neg x_1 \vee \neg x_2 \vee x_3\ ) \wedge (\ x_1 \vee x_2 \vee x_3\ )$$



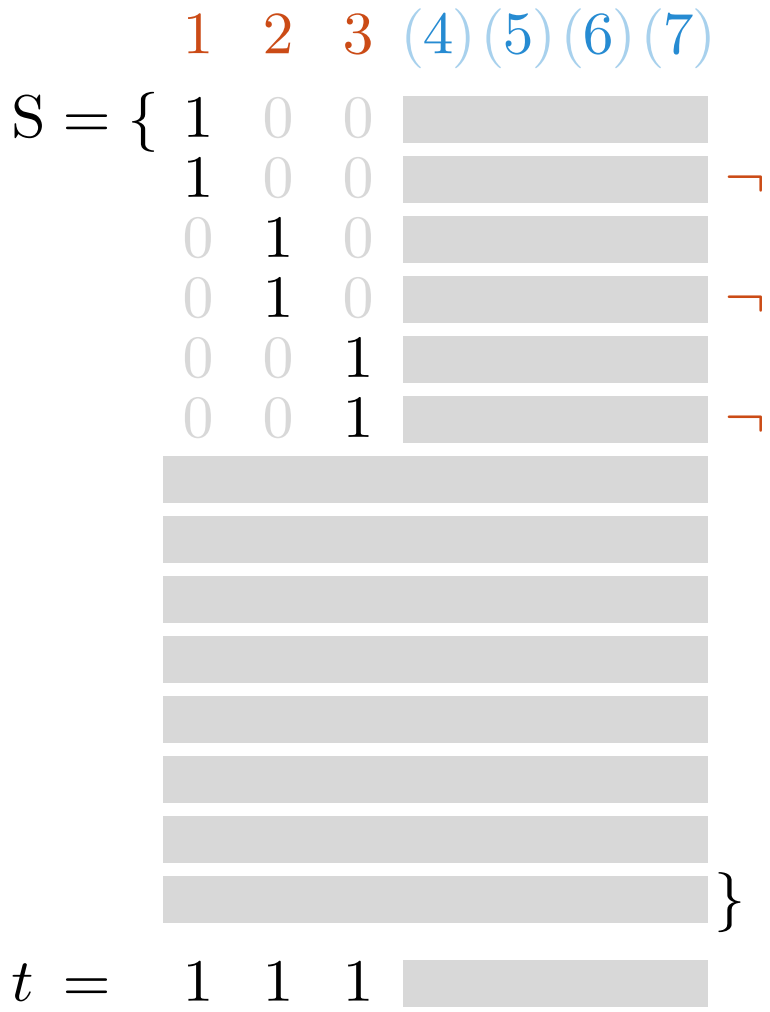
Ett tall per variabel og dens negasjon

$$\begin{aligned}\phi = (\ & x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \\ (\ & \neg x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \\ (\ & \neg x_1 \vee \neg x_2 \vee x_3\ ) \wedge \\ (\ & x_1 \vee x_2 \vee x_3\ )\end{aligned}$$



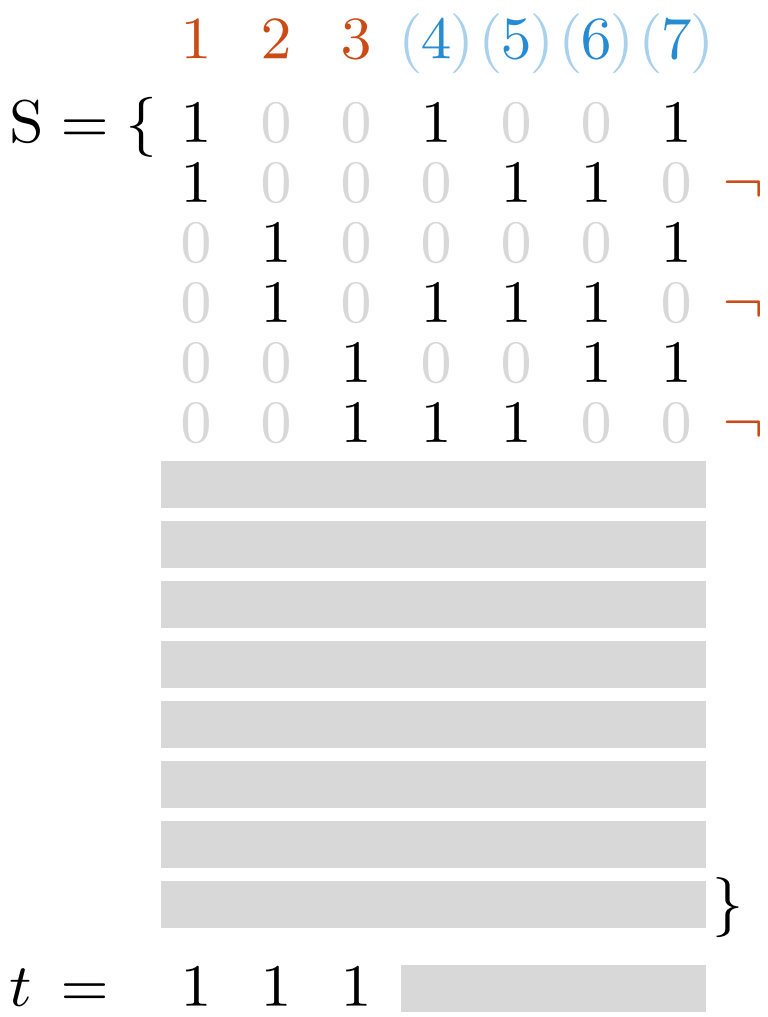
Delsummen tvinger frem nøyaktig én sannhetsverdi per variabel

$$
\begin{aligned}
\phi = (\ & x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge && (4)\\
(\ & \neg x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge && (5)\\
(\ & \neg x_1 \vee \neg x_2 \vee x_3\ ) \wedge && (6)\\
(\ & x_1 \vee x_2 \vee x_3\ ) && (7)
\end{aligned}
$$



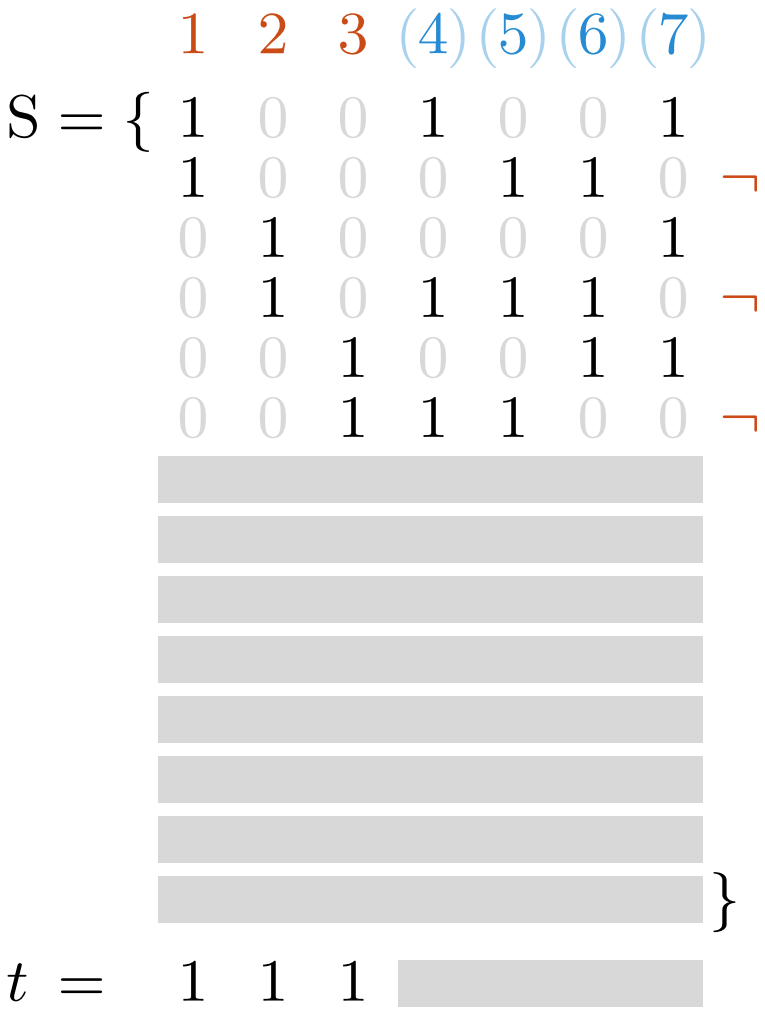
Vi innfører et siffer per disjunksjon

$$\phi = (\ x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \quad (4)$$
$$(\neg x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \quad (5)$$
$$(\neg x_1 \vee \neg x_2 \vee\ \ x_3\ ) \wedge \quad (6)$$
$$(\ x_1 \vee\ \ x_2 \vee\ \ x_3\ ) \quad (7)$$



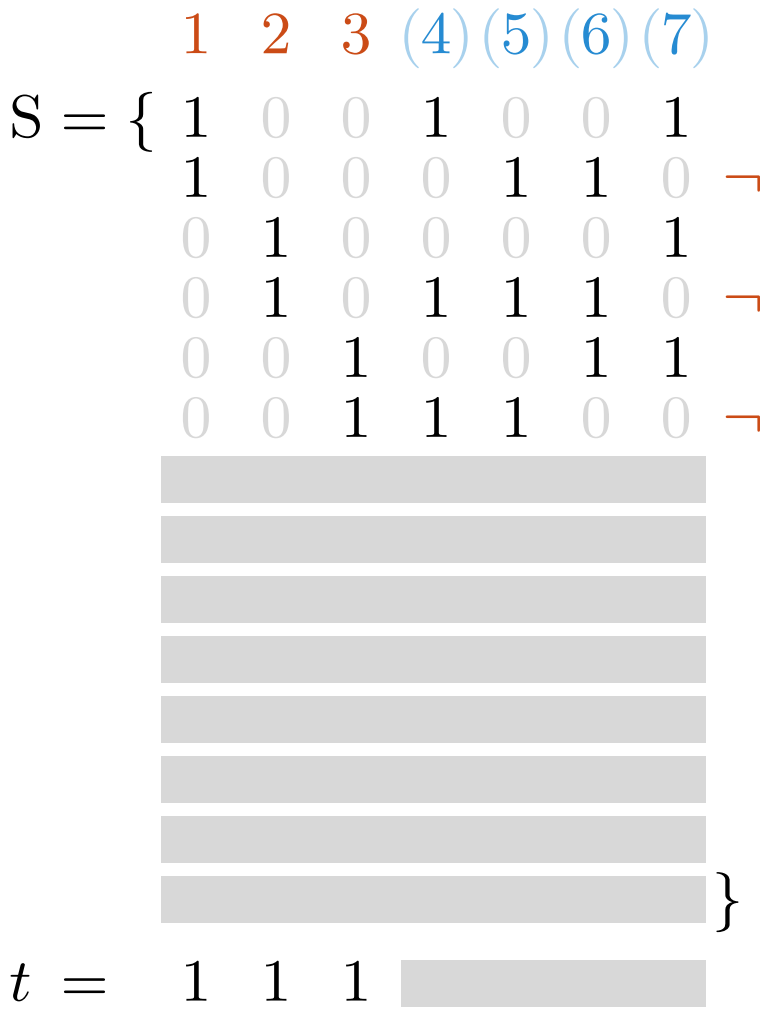
For hver disjunksjon Er variabelen eller negasjonen med?

$$\phi = (\ x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \quad (4)$$
$$(\neg x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \quad (5)$$
$$(\neg x_1 \vee \neg x_2 \vee\ \ x_3\ ) \wedge \quad (6)$$
$$(\ x_1 \vee\ \ x_2 \vee\ \ x_3\ ) \quad (7)$$



Vi vil tvinge frem minst én sann literal per disjunksjon

$$\phi = (\ x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \quad (4)$$
$$(\neg x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \quad (5)$$
$$(\neg x_1 \vee \neg x_2 \vee\ \ x_3\ ) \wedge \quad (6)$$
$$(\ x_1 \vee\ \ x_2 \vee\ \ x_3\ ) \quad (7)$$



Sum per disjunksjonskolonne skal altså være 1, 2 eller 3

$$\phi = (\ x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \quad (4)$$
$$(\neg x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \quad (5)$$
$$(\neg x_1 \vee \neg x_2 \vee\ \ x_3\ ) \wedge \quad (6)$$
$$(\ x_1 \vee\ \ x_2 \vee\ \ x_3\ ) \quad (7)$$

$$
\begin{array}{rccccccccc}
 & 1 & 2 & 3 & (4) & (5) & (6) & (7) & \\
S = \{ & 1 & 0 & 0 & 1 & 0 & 0 & 1 & \\
 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & \neg \\
 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & \\
 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & \neg \\
 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & \\
 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & \neg \\
 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \\
 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & \\
 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \\
 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & \\
 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \\
 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & \} \\
t = & 1 & 1 & 1 & & & & &
\end{array}
$$

Vi innfører slakkverdier, som fyller ut det som mangler

$$\phi = (\ x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \quad (4)$$
$$(\neg x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \quad (5)$$
$$(\neg x_1 \vee \neg x_2 \vee\ x_3\ ) \wedge \quad (6)$$
$$(\ x_1 \vee\ x_2 \vee\ x_3\ ) \quad (7)$$

$$\begin{array}{rccccccccc}
 & 1 & 2 & 3 & (4) & (5) & (6) & (7) & \\
S = \{ & 1 & 0 & 0 & 1 & 0 & 0 & 1 & \\
 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & \neg \\
 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & \\
 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & \neg \\
 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & \\
 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & \neg \\
 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \\
 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & \\
 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \\
 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & \\
 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \\
 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & \} \\
t = & 1 & 1 & 1 & 4 & 4 & 4 & 4 &
\end{array}$$

Men: Nå kan vi få slakk på opptil 3; må kreve at summen er 4

$$\phi = (\ x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \quad (4)$$
$$(\neg x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \quad (5)$$
$$(\neg x_1 \vee \neg x_2 \vee\ \ x_3\ ) \wedge \quad (6)$$
$$(\ x_1 \vee\ \ x_2 \vee\ \ x_3\ ) \quad (7)$$

Kan ϕ være sann?

$\Longleftrightarrow$

Har S en delsum t?

	1	2	3	(4)	(5)	(6)	(7)	
S = {	1	0	0	1	0	0	1	
	1	0	0	0	1	1	0	¬
	0	1	0	0	0	0	1	
	0	1	0	1	1	1	0	¬
	0	0	1	0	0	1	1	
	0	0	1	1	1	0	0	¬
	0	0	0	1	0	0	0	
	0	0	0	2	0	0	0	
	0	0	0	0	1	0	0	
	0	0	0	0	2	0	0	
	0	0	0	0	0	1	0	
	0	0	0	0	0	2	0	
	0	0	0	0	0	0	1	
	0	0	0	0	0	0	2	}
t =	1	1	1	4	4	4	4	

$$\phi = (\ x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \quad (4)$$
$$(\neg x_1 \vee \neg x_2 \vee \neg x_3\ ) \wedge \quad (5)$$
$$(\neg x_1 \vee \neg x_2 \vee\ \ x_3\ ) \wedge \quad (6)$$
$$(\ x_1 \vee\ \ x_2 \vee\ \ x_3\ ) \quad (7)$$

$x_1 = 1,\ \ x_2 = 0$

Kan ϕ være sann?

$\Longleftrightarrow$

Har S en delsum t?

	1	2	3	(4)	(5)	(6)	(7)	
S = {	1	0	0	1	0	0	1	
	1	0	0	0	1	1	0	¬
	0	1	0	0	0	0	1	
	0	1	0	1	1	1	0	¬
	0	0	1	0	0	1	1	
	0	0	1	1	1	0	0	¬
	0	0	0	1	0	0	0	
	0	0	0	2	0	0	0	
	0	0	0	0	1	0	0	
	0	0	0	0	2	0	0	
	0	0	0	0	0	1	0	
	0	0	0	0	0	2	0	
	0	0	0	0	0	0	1	
	0	0	0	0	0	0	2	}
t =	1	1	1	4	4	4	4	

Merk: For at reduksjonen skal fungere, må vi ha større tall enn vi kan håndtere i konstant tid i RAM-modellen

Lett å redusere videre til det binære ryggsekkproblemet

Bare la vekt være lik verdi

Det viser at det binære ryggsekk-problemet er NP-hardt, men gir oss også en pseudopolynomisk løsning på SUBSET-SUM!

1. **CIRCUIT-SAT**
2. **SAT**
3. **3-CNF-SAT**
4. **CLIQUE**
5. **VERTEX-COVER**
6. **HAM-CYCLE**
7. **TSP**
8. **SUBSET-SUM**