Independent sets and vertex covers
COMS20010 2020, Video 10-1

John Lapinskas, University of Bristol

Examples of NP-hardness

We can prove a problem is NP-hard by reducing from 3-SAT... but we can also do it by reducing from any other NP-complete problem.

There are so many to choose from it's hard to get the scale across, so in the vein of Project Steve I'm only going to list examples from video games:

Examples of NP-hardness

We can prove a problem is NP-hard by reducing from 3-SAT... but we can also do it by reducing from any other NP-complete problem.

There are so many to choose from it's hard to get the scale across, so in the vein of Project Steve I'm only going to list examples from video games:

- Lemmings;
- Pac-Man;
- Minesweeper;
- Tetris;
- Candy Crush;
- Angry Birds;
- Classic Mario games;
- Spelunky;
- Donkey Kong Country 1–3.

Examples of NP-hardness

We can prove a problem is NP-hard by reducing from 3-SAT... but we can also do it by reducing from any other NP-complete problem.

There are so many to choose from it's hard to get the scale across, so in the vein of Project Steve I'm only going to list examples from video games:

- Lemmings;
- Pac-Man;
- Minesweeper;
- Tetris;
- Candy Crush;
- Angry Birds;
- Classic Mario games;
- Spelunky;
- Donkey Kong Country 1–3;

Examples of NP-hardness

We can prove a problem is NP-hard by reducing from 3-SAT... but we can also do it by reducing from any other NP-complete problem.

There are so many to choose from it's hard to get the scale across, so in the vein of Project Steve I'm only going to list examples from video games:

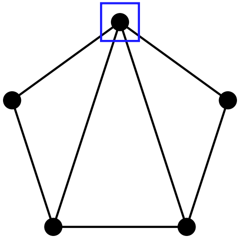
- Lemmings;
- Pac-Man;
- Minesweeper;
- Tetris;
- Candy Crush;
- Angry Birds;
- Classic Mario games;
- Spelunky;
- Donkey Kong Country 1–3;
- Every Legend of Zelda game;
- Every Metroid game;
- Every Fire Emblem game;
- Mainline Pokémon games;
- Mario Kart;
- Desktop Tower Defense;
- Harvest Moon;
- Inventory packing in ARPGs;
- Damage boosting in speedruns.

Independent sets

In a graph $G = (V, E)$, an **independent set** is a subset of V which contains no edges. For example:

Independent sets

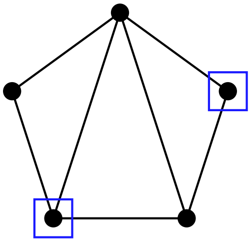
In a graph $G = (V, E)$, an **independent set** is a subset of V which contains no edges. For example:



These sets are independent...

Independent sets

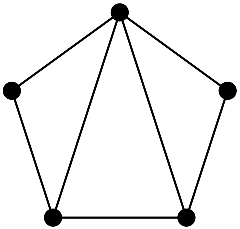
In a graph $G = (V, E)$, an **independent set** is a subset of V which contains no edges. For example:



These sets are independent...

Independent sets

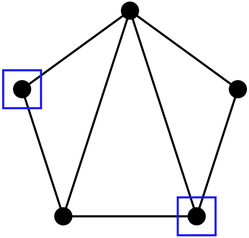
In a graph $G = (V, E)$, an **independent set** is a subset of V which contains no edges. For example:



These sets are independent...

Independent sets

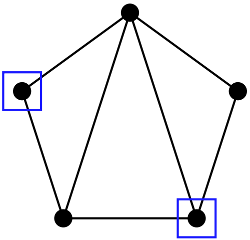
In a graph $G = (V, E)$, an **independent set** is a subset of V which contains no edges. For example:



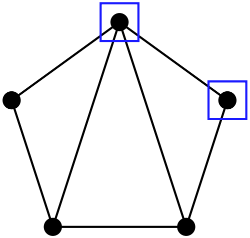
These sets are independent...

Independent sets

In a graph $G = (V, E)$, an **independent set** is a subset of V which contains no edges. For example:



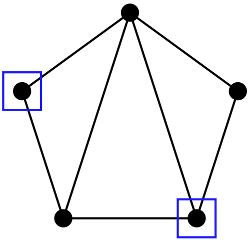
These sets are independent...



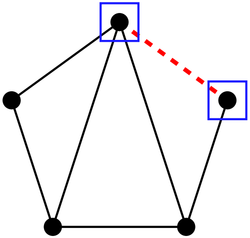
But this set isn't.

Independent sets

In a graph $G = (V, E)$, an **independent set** is a subset of V which contains no edges. For example:



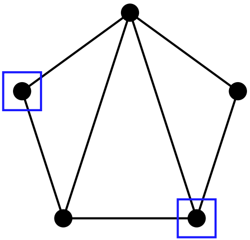
These sets are independent...



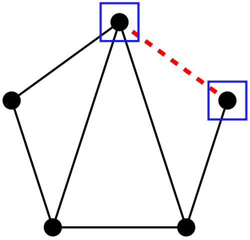
But this set isn't.

Independent sets

In a graph $G = (V, E)$, an **independent set** is a subset of V which contains no edges. For example:



These sets are independent...



But this set isn't.

Independent sets are important in graphs which model **conflicts**.

For example, suppose we are trying to assign frequencies to radio transmitters while avoiding interference. If we join two transmitters by an edge when they are close enough to interfere with each other, then we can safely assign the same frequency to all transmitters in an independent set.

In a graph $G = (V, E)$, an **independent set** is a subset of V which contains no edges.

Independent sets are important in graphs which model **conflicts**.

For example, suppose we are trying to assign frequencies to radio transmitters while avoiding interference. If we join two transmitters by an edge when they are close enough to interfere with each other, then we can assign the same frequency to all transmitters in an independent set.

In a graph $G = (V, E)$, an **independent set** is a subset of V which contains no edges.

Independent sets are important in graphs which model **conflicts**.

For example, suppose we are trying to assign frequencies to radio transmitters while avoiding interference. If we join two transmitters by an edge when they are close enough to interfere with each other, then we can assign the same frequency to all transmitters in an independent set.

We would often like to be able to find a **maximum** independent set, i.e. one which is as big as possible.

In a graph $G = (V, E)$, an **independent set** is a subset of V which contains no edges.

Independent sets are important in graphs which model **conflicts**.

For example, suppose we are trying to assign frequencies to radio transmitters while avoiding interference. If we join two transmitters by an edge when they are close enough to interfere with each other, then we can assign the same frequency to all transmitters in an independent set.

We would often like to be able to find a **maximum** independent set, i.e. one which is as big as possible.

The decision version of this problem, **IS**, asks: Given a graph $G = (V, E)$, and an integer k, does G contain an independent set of size at least k?

Theorem: IS is NP-complete.

In a graph $G = (V, E)$, an **independent set** is a subset of V which contains no edges.

Independent sets are important in graphs which model **conflicts**.

For example, suppose we are trying to assign frequencies to radio transmitters while avoiding interference. If we join two transmitters by an edge when they are close enough to interfere with each other, then we can assign the same frequency to all transmitters in an independent set.

We would often like to be able to find a **maximum** independent set, i.e. one which is as big as possible.

The decision version of this problem, **IS**, asks: Given a graph $G = (V, E)$, and an integer k, does G contain an independent set of size at least k?

Theorem: IS is NP-complete.

We can verify a set is independent in polynomial time, so IS $\in$ NP.

In a graph $G = (V, E)$, an **independent set** is a subset of V which contains no edges.

Independent sets are important in graphs which model **conflicts**.

For example, suppose we are trying to assign frequencies to radio transmitters while avoiding interference. If we join two transmitters by an edge when they are close enough to interfere with each other, then we can assign the same frequency to all transmitters in an independent set.

We would often like to be able to find a **maximum** independent set, i.e. one which is as big as possible.

The decision version of this problem, **IS**, asks: Given a graph $G = (V, E)$, and an integer k, does G contain an independent set of size at least k?

Theorem: IS is NP-complete.

We can verify a set is independent in polynomial time, so IS $\in$ NP.

We will show NP-hardness by reducing from 3-SAT, i.e. proving 3-SAT $\leq_c$ IS. Since we already proved SAT $\leq_c$ 3-SAT, the result follows.

In a graph $G = (V, E)$, an **independent set** is a subset of V containing no edges. **IS** asks: Does G contain an independent set of size at least k?

A CNF formula has width 3 if all its OR clauses contain 3 literals.
3-SAT asks: is the input width-3 CNF formula satisfiable?

Theorem: IS is NP-complete. **Goal:** Prove 3-SAT $\leq_c$ IS.

Let F be an instance of 3-SAT. We'll follow our usual approach: build a graph G whose size-($\geq k$) independent sets correspond to satisfying assignments of F, then apply our IS oracle to G.

In a graph $G = (V, E)$, an **independent set** is a subset of V containing no edges. **IS** asks: Does G contain an independent set of size at least k?

A CNF formula has width 3 if all its OR clauses contain 3 literals.
3-SAT asks: is the input width-3 CNF formula satisfiable?

Theorem: IS is NP-complete. **Goal:** Prove 3-SAT $\leq_c$ IS.

Let F be an instance of 3-SAT. We'll follow our usual approach: build a graph G whose size-($\geq k$) independent sets correspond to satisfying assignments of F, then apply our IS oracle to G.

To simulate the variables of F, we want a gadget that can be in one of two states which will represent `True` and `False`...

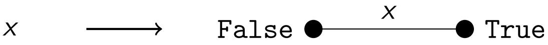
In a graph $G = (V, E)$, an **independent set** is a subset of V containing no edges. **IS** asks: Does G contain an independent set of size at least k?

A CNF formula has width 3 if all its OR clauses contain 3 literals.
3-SAT asks: is the input width-3 CNF formula satisfiable?

Theorem: IS is NP-complete. **Goal:** Prove 3-SAT $\leq_c$ IS.

Let F be an instance of 3-SAT. We'll follow our usual approach: build a graph G whose size-($\geq k$) independent sets correspond to satisfying assignments of F, then apply our IS oracle to G.

To simulate the variables of F, we want a gadget that can be in one of two states which will represent `True` and `False`...



Idea: Use an edge!

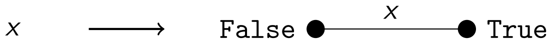
In a graph $G = (V, E)$, an **independent set** is a subset of V containing no edges. **IS** asks: Does G contain an independent set of size at least k?

A CNF formula has width 3 if all its OR clauses contain 3 literals.
3-SAT asks: is the input width-3 CNF formula satisfiable?

Theorem: IS is NP-complete. **Goal:** Prove 3-SAT $\leq_c$ IS.

Let F be an instance of 3-SAT. We'll follow our usual approach: build a graph G whose size-($\geq k$) independent sets correspond to satisfying assignments of F, then apply our IS oracle to G.

To simulate the variables of F, we want a gadget that can be in one of two states which will represent `True` and `False`...



Idea: Use an edge!

An independent set can't contain both vertices, and (if we do everything else right) a **maximum** independent set must contain one of the two vertices.

In a graph $G = (V, E)$, an **independent set** is a subset of V containing no edges. **IS** asks: Does G contain an independent set of size at least k?

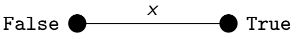
A CNF formula has width 3 if all its OR clauses contain 3 literals.
3-SAT asks: is the input width-3 CNF formula satisfiable?

Theorem: IS is NP-complete. **Goal:** Prove 3-SAT $\leq_c$ IS.

Let F be an instance of 3-SAT. We'll build a graph G whose size-($\geq k$) independent sets correspond to satisfying assignments of F.

Our variable gadget is:

$x \quad \longrightarrow$



We use the same idea to model the clauses of F. We have three literals in the clause, and we want to force one of them to be true, so...

In a graph $G = (V, E)$, an **independent set** is a subset of V containing no edges. **IS** asks: Does G contain an independent set of size at least k?

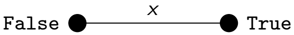
A CNF formula has width 3 if all its OR clauses contain 3 literals.
3-SAT asks: is the input width-3 CNF formula satisfiable?

Theorem: IS is NP-complete. **Goal:** Prove 3-SAT $\leq_c$ IS.

Let F be an instance of 3-SAT. We'll build a graph G whose size-($\geq k$) independent sets correspond to satisfying assignments of F.

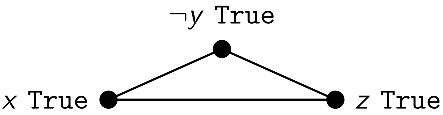
Our variable gadget is:

$x \longrightarrow$



We use the same idea to model the clauses of F. We have three literals in the clause, and we want to force one of them to be true, so...

$x \vee \neg y \vee z \longrightarrow$



We will set things up so that:
Maximum independent set $\Longrightarrow$ exactly one vertex is included.

Joining the gadgets together

So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?

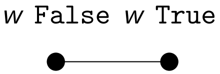
Joining the gadgets together

So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?

We add gadgets for each variable...

Joining the gadgets together

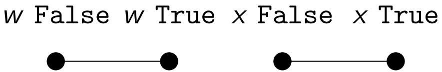
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



We add gadgets for each variable...

Joining the gadgets together

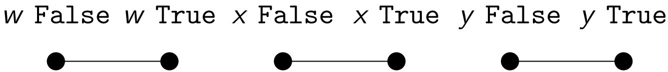
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



We add gadgets for each variable...

Joining the gadgets together

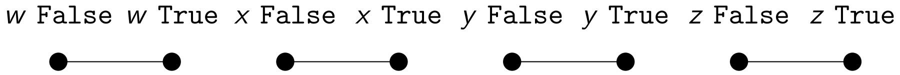
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



We add gadgets for each variable...

Joining the gadgets together

So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



We add gadgets for each variable...

Joining the gadgets together

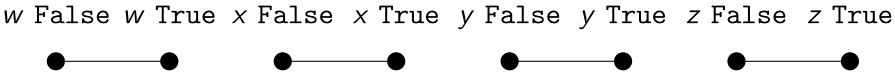
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?

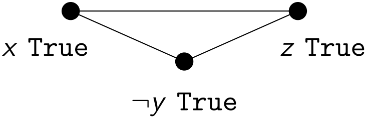
w `False` w `True` x `False` x `True` y `False` y `True` z `False` z `True`

We add gadgets for each variable... and gadgets for each clause...

Joining the gadgets together

So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?

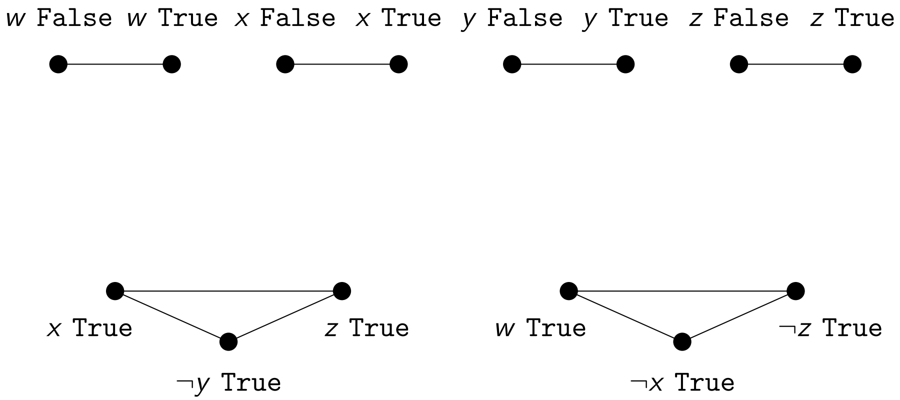




We add gadgets for each variable... and gadgets for each clause...

Joining the gadgets together

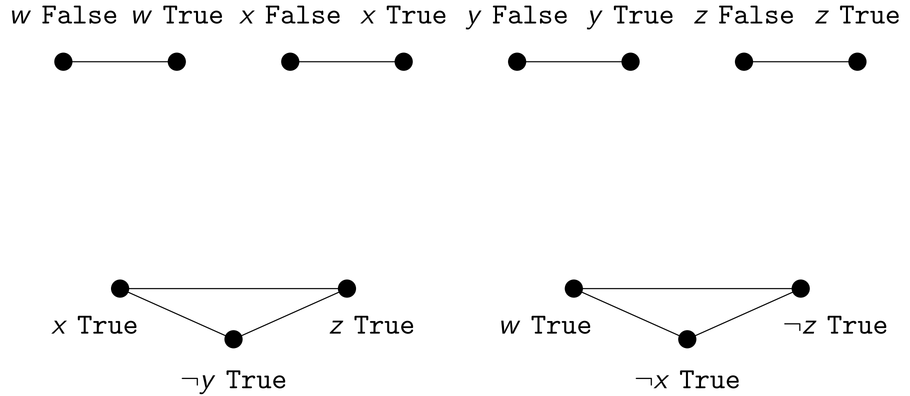
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



We add gadgets for each variable... and gadgets for each clause...

Joining the gadgets together

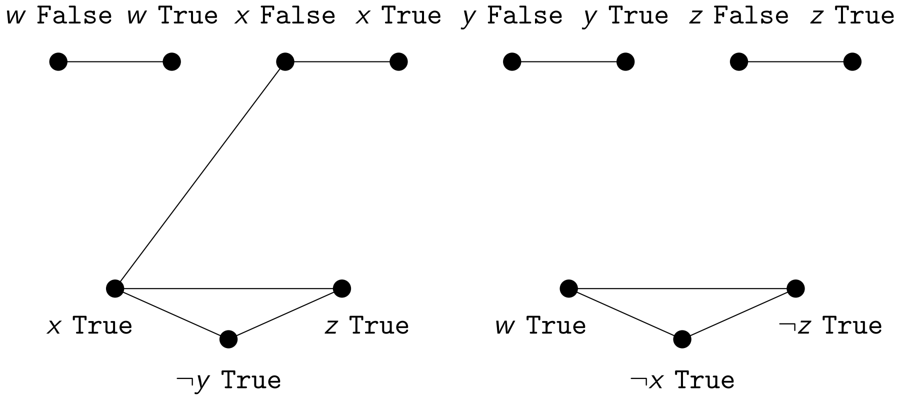
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



And now we need to enforce that "x True" is only in the independent set if x is actually true. Which we can do by joining it to the corresponding "x False" vertex in the variable gadget.

Joining the gadgets together

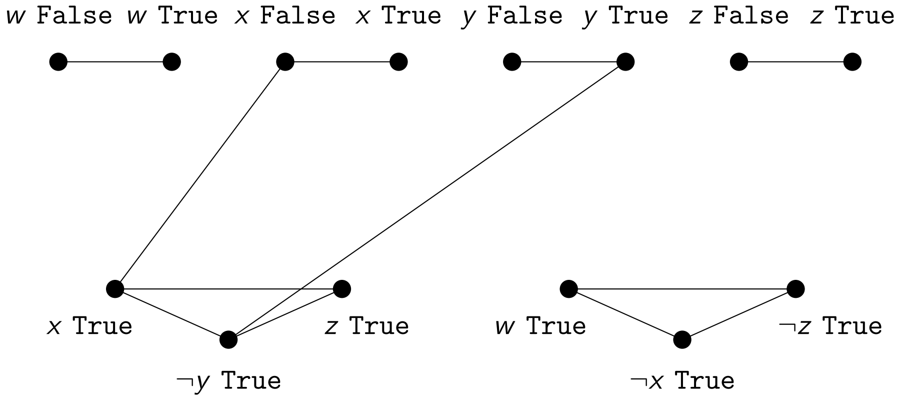
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



And now we need to enforce that "x `True`" is only in the independent set if x is actually true. Which we can do by joining it to the corresponding "x `False`" vertex in the variable gadget.

Joining the gadgets together

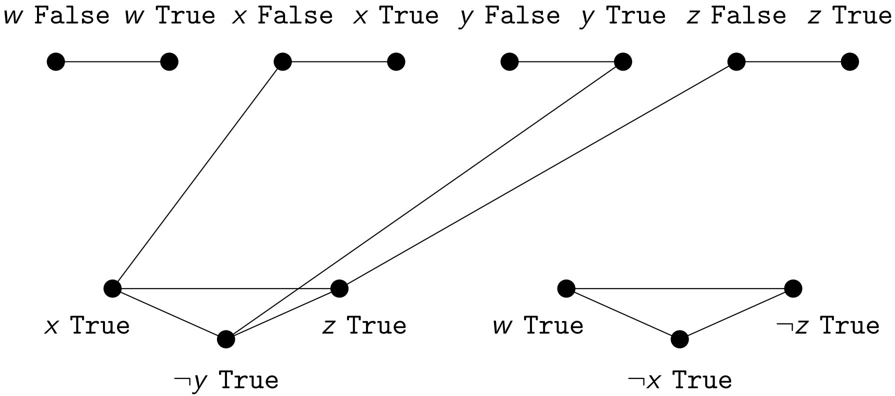
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



And now we need to enforce that "x True" is only in the independent set if x is actually true. Which we can do by joining it to the corresponding "x False" vertex in the variable gadget.

Joining the gadgets together

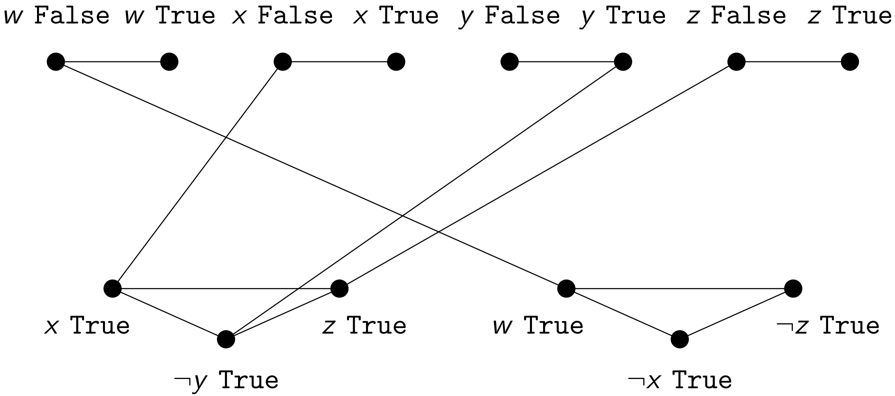
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



And now we need to enforce that "x `True`" is only in the independent set if x is actually true. Which we can do by joining it to the corresponding "x `False`" vertex in the variable gadget.

Joining the gadgets together

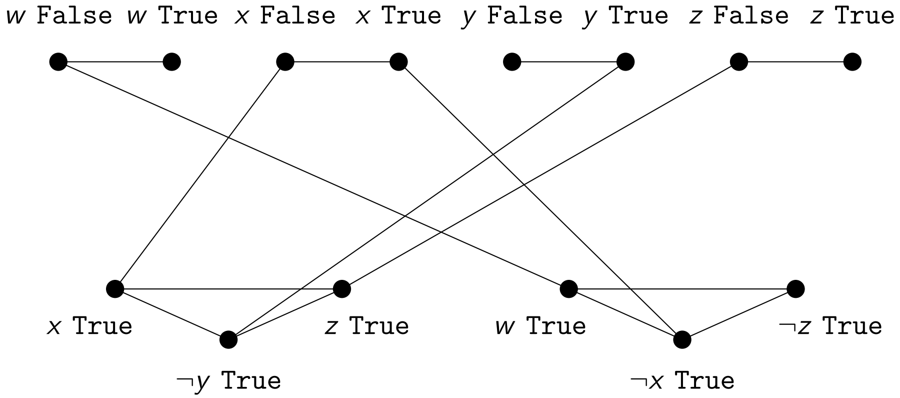
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



And now we need to enforce that "x `True`" is only in the independent set if x is actually true. Which we can do by joining it to the corresponding "x `False`" vertex in the variable gadget.

Joining the gadgets together

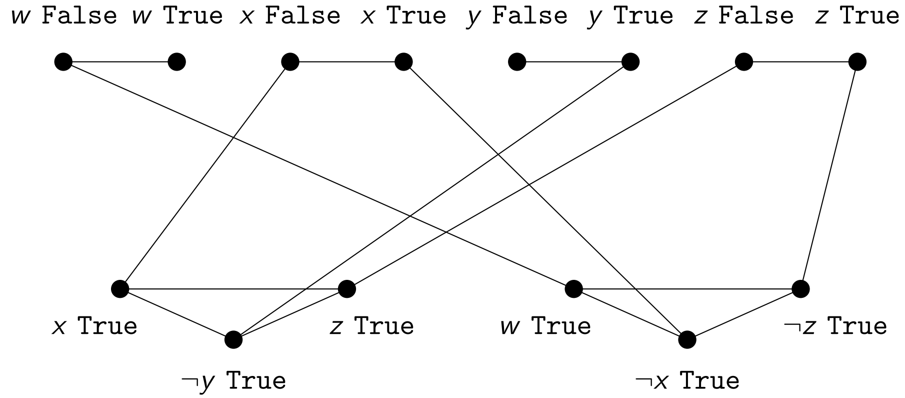
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



And now we need to enforce that "x True" is only in the independent set if x is actually true. Which we can do by joining it to the corresponding "x False" vertex in the variable gadget.

Joining the gadgets together

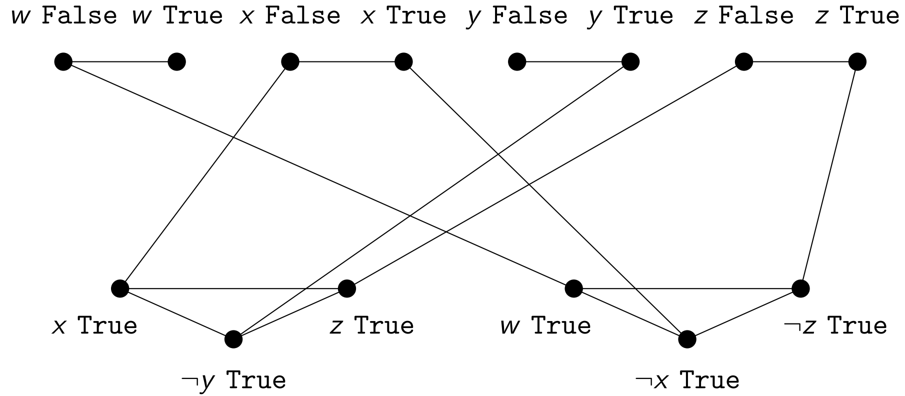
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



And now we need to enforce that "x True" is only in the independent set if x is actually true. Which we can do by joining it to the corresponding "x False" vertex in the variable gadget.

Joining the gadgets together

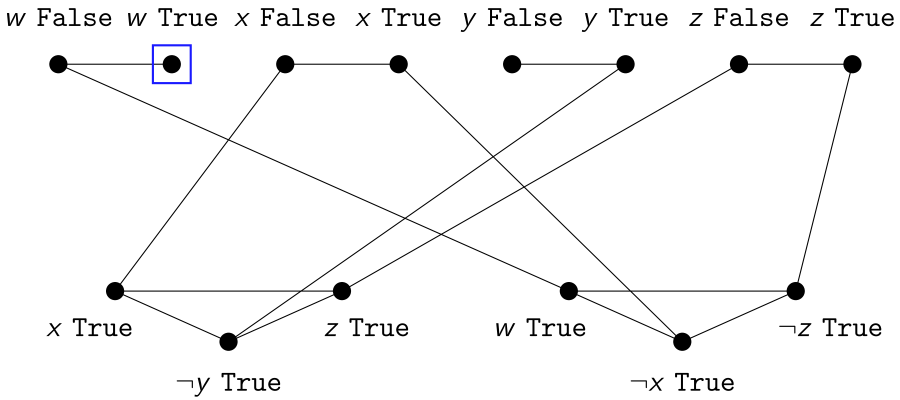
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



Now a satisfying assignment in F, say $w = \texttt{True}$, $x = \texttt{True}$, $y = \texttt{False}$, $z = \texttt{False}$, corresponds to a size-6 independent set in G.

Joining the gadgets together

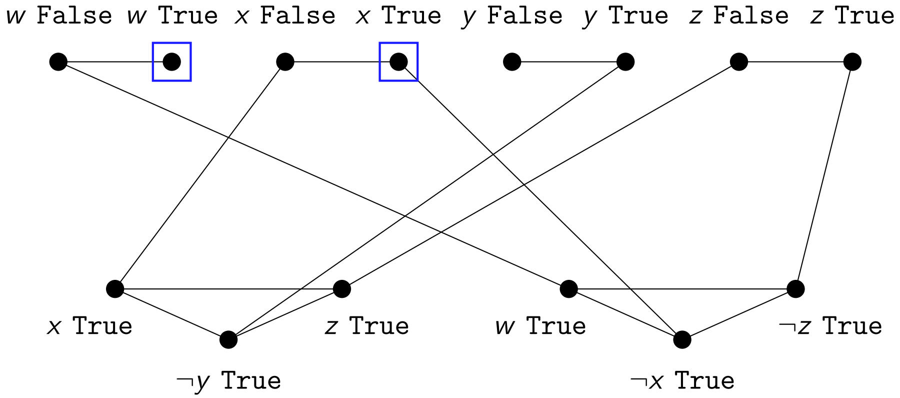
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



Now a satisfying assignment in F, say $\boldsymbol{w} = \texttt{True}$, $x = \texttt{True}$, $y = \texttt{False}$, $z = \texttt{False}$, corresponds to a size-6 independent set in G.

Joining the gadgets together

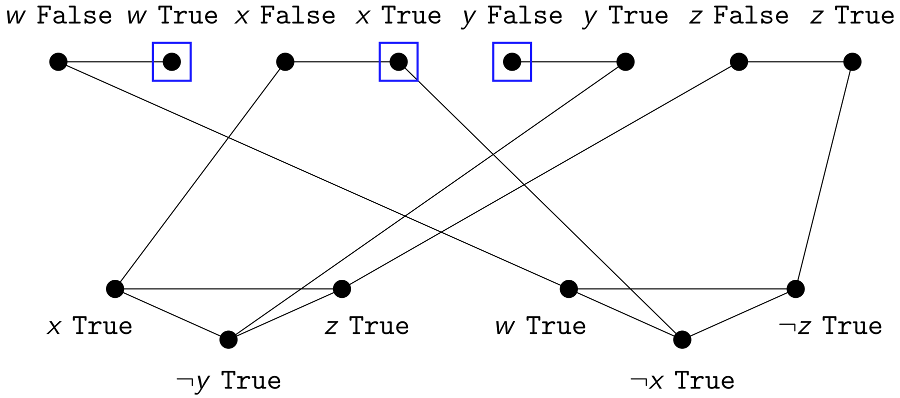
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



Now a satisfying assignment in F, say $w = \texttt{True}$, $\boldsymbol{x} = \texttt{True}$, $y = \texttt{False}$, $z = \texttt{False}$, corresponds to a size-6 independent set in G.

Joining the gadgets together

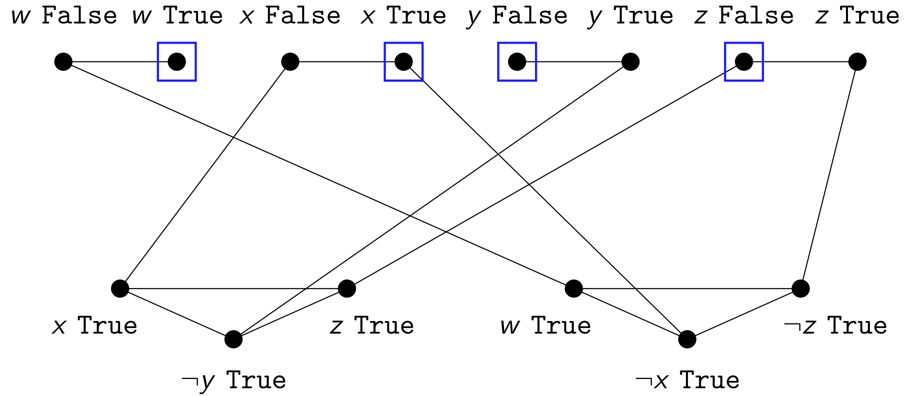
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



Now a satisfying assignment in F, say $w =$ True, $x =$ True, $\boldsymbol{y} =$ **False**, $z =$ False, corresponds to a size-6 independent set in G.

Joining the gadgets together

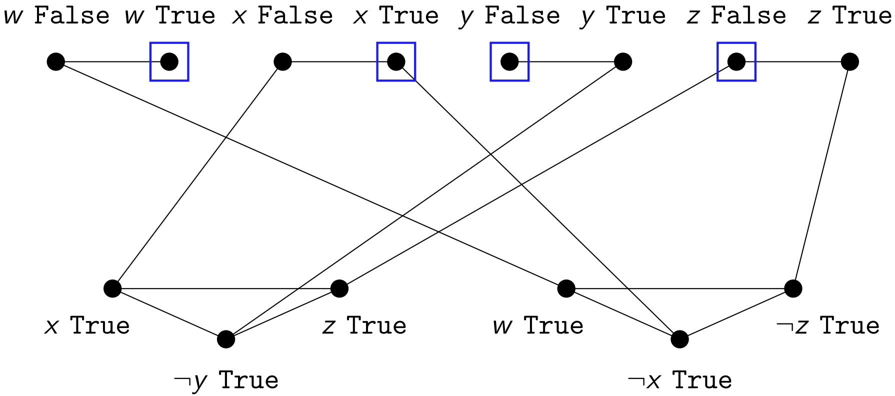
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



Now a satisfying assignment in F, say $w =$ True, $x =$ True, $y =$ False, $z =$ False, corresponds to a size-6 independent set in G.

Joining the gadgets together

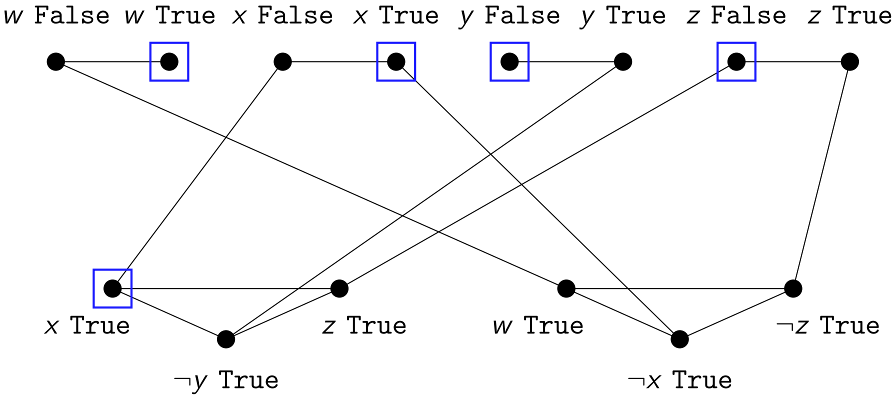
So if $F = (\boldsymbol{x} \vee \boldsymbol{\neg y} \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



Now a satisfying assignment in F, say $w = \texttt{True}$, $x = \texttt{True}$, $y = \texttt{False}$, $z = \texttt{False}$, corresponds to a size-6 independent set in G.

Joining the gadgets together

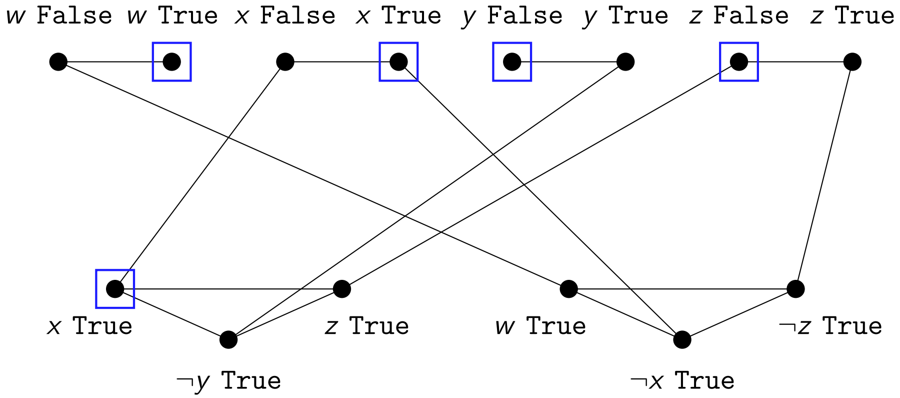
So if $F = (\boldsymbol{x} \vee \boldsymbol{\neg y} \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



Now a satisfying assignment in F, say $w = \texttt{True}$, $x = \texttt{True}$, $y = \texttt{False}$, $z = \texttt{False}$, corresponds to a size-6 independent set in G.

Joining the gadgets together

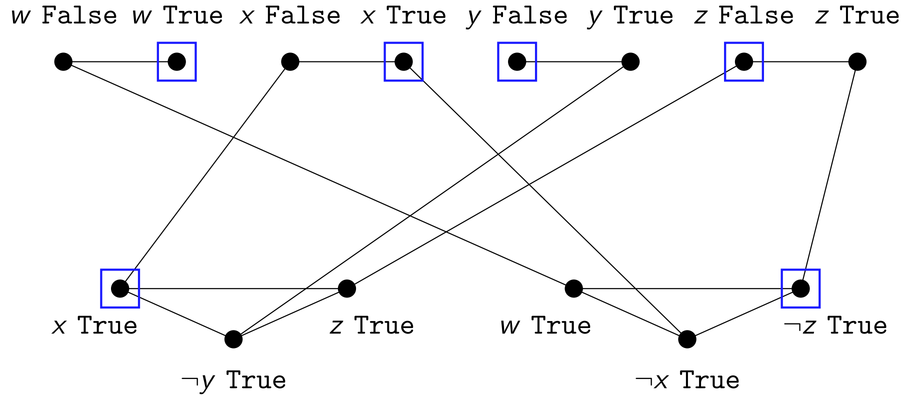
So if $F = (x \vee \neg y \vee z) \wedge (\boldsymbol{w} \vee \neg x \vee \boldsymbol{\neg z})$, say, how do we build G?



Now a satisfying assignment in F, say $w = \texttt{True}$, $x = \texttt{True}$, $y = \texttt{False}$, $z = \texttt{False}$, corresponds to a size-6 independent set in G.

Joining the gadgets together

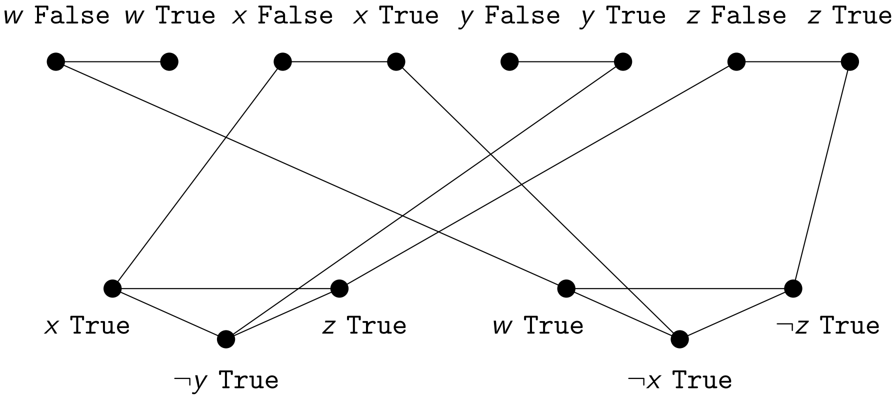
So if $F = (x \vee \neg y \vee z) \wedge (\boldsymbol{w} \vee \neg x \vee \boldsymbol{\neg z})$, say, how do we build G?



Now a satisfying assignment in F, say $w = \texttt{True}$, $x = \texttt{True}$, $y = \texttt{False}$, $z = \texttt{False}$, corresponds to a size-6 independent set in G.

Joining the gadgets together

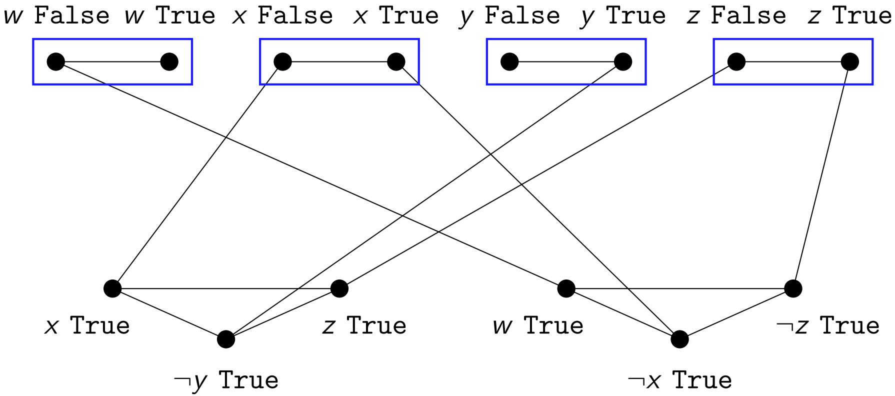
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



Conversely, any independent set in G has at most one vertex from each variable gadget...

Joining the gadgets together

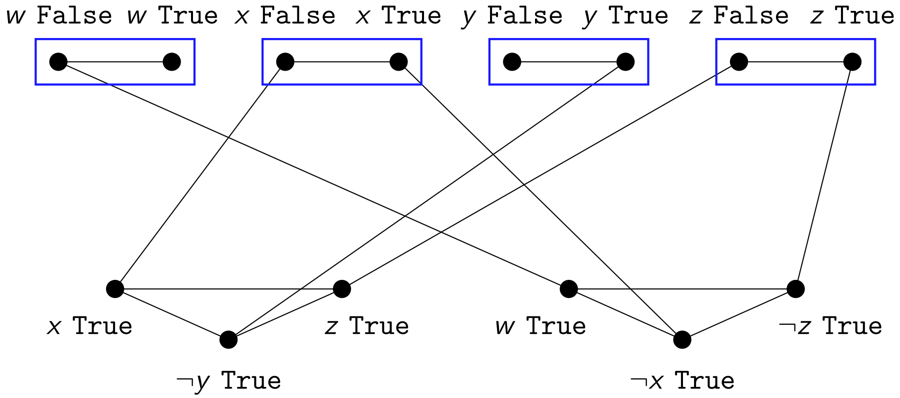
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



Conversely, any independent set in G has at most one vertex from each variable gadget...

Joining the gadgets together

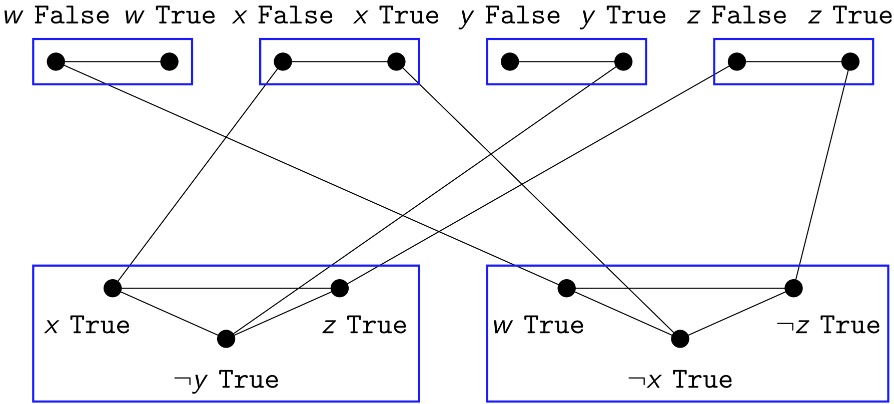
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



Conversely, any independent set in G has at most one vertex from each of these edges... and at most one vertex from each clause gadget, for a total of at most 6 vertices.

Joining the gadgets together

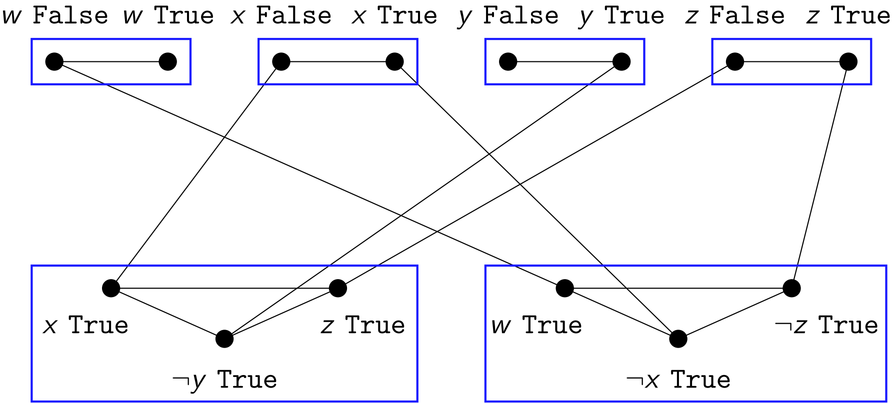
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



Conversely, any independent set in G has at most one vertex from each of these edges... and at most one vertex from each clause gadget, for a total of at most 6 vertices.

Joining the gadgets together

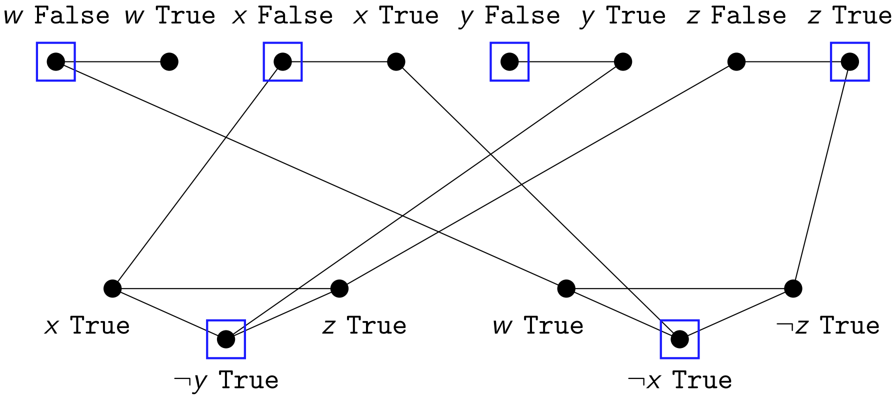
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



So any size-($\geq$6) independent set must have size **exactly** 6, and contain one vertex from each variable gadget and one from each clause gadget.

Joining the gadgets together

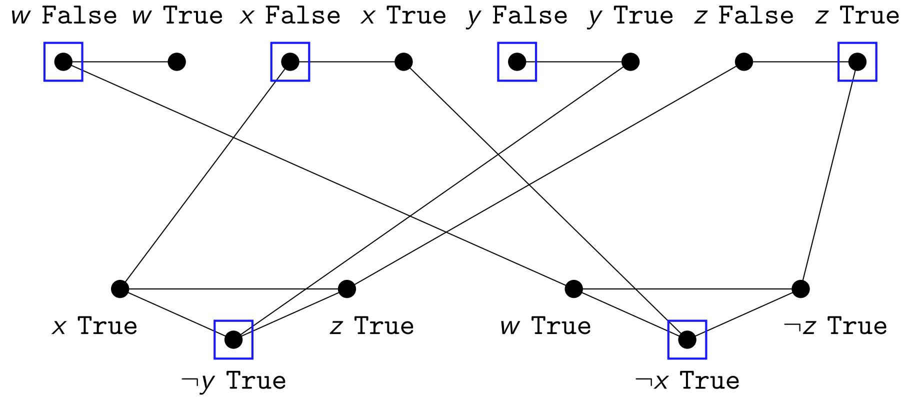
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



So any size-($\geq$6) independent set must have size **exactly** 6, and contain one vertex from each variable gadget and one from each clause gadget.

Joining the gadgets together

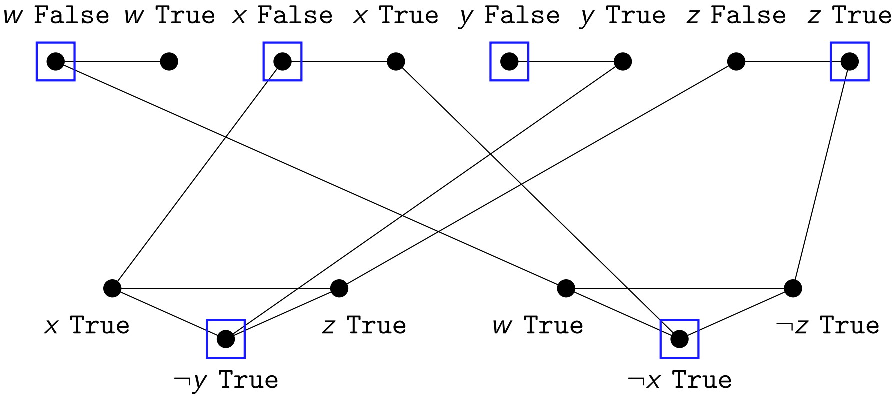
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



Hence any size-($\geq$6) independent set corresponds to an assignment, in this case $w = x = y =$ False, $z =$ True. It must be satisfying because there are no edges between variable vertices and clause vertices.

Joining the gadgets together

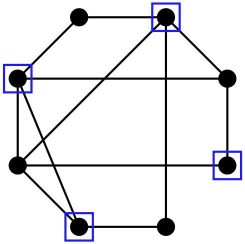
So if $F = (x \vee \neg y \vee z) \wedge (w \vee \neg x \vee \neg z)$, say, how do we build G?



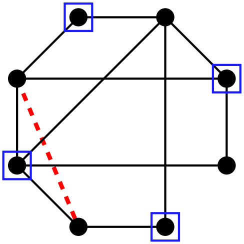
The same construction (and the same correctness proof) works for any instance of 3-SAT. □

Recall from Video 8-2...

A **vertex cover** in a graph $G = (V, E)$ is a set $X \subseteq V$ such that every edge in E has at least one vertex in X.



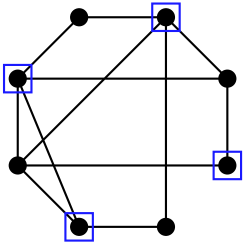
A valid vertex cover.



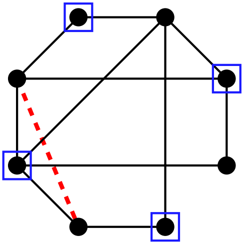
Not a valid vertex cover.

Recall from Video 8-2...

A **vertex cover** in a graph $G = (V, E)$ is a set $X \subseteq V$ such that every edge in E has at least one vertex in X.



A valid vertex cover.

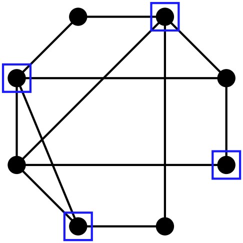


Not a valid vertex cover.

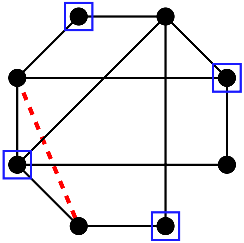
We would like to find the **smallest possible** vertex cover of G.
We claimed this problem was hard to solve exactly (our algorithm was approximate), but we never proved it...

Recall from Video 8-2...

A **vertex cover** in a graph $G = (V, E)$ is a set $X \subseteq V$ such that every edge in E has at least one vertex in X.



A valid vertex cover.



Not a valid vertex cover.

We would like to find the **smallest possible** vertex cover of G.
We claimed this problem was hard to solve exactly (our algorithm was approximate), but we never proved it...

The decision version of this problem (**VC**) asks: Given a graph G, and an integer k, does G contain a vertex cover of size at most k?

A **vertex cover** in a graph $G = (V, E)$ is a set $X \subseteq V$ such that every edge in E has at least one vertex in X.

VC asks: Does G contain a vertex cover of size at least k?

A **vertex cover** in a graph $G = (V, E)$ is a set $X \subseteq V$ such that every edge in E has at least one vertex in X.

VC asks: Does G contain a vertex cover of size at least k?

Theorem: VC is NP-complete.

A **vertex cover** in a graph $G = (V, E)$ is a set $X \subseteq V$ such that every edge in E has at least one vertex in X.

VC asks: Does G contain a vertex cover of size at least k?

Theorem: VC is NP-complete.

We can verify a set is a vertex cover in polynomial time, so VC $\in$ NP.
We'll prove NP-hardness by proving IS $\leq_c$ VC.

A **vertex cover** in a graph $G = (V, E)$ is a set $X \subseteq V$ such that every edge in E has at least one vertex in X.

VC asks: Does G contain a vertex cover of size at least k?

Theorem: VC is NP-complete.

We can verify a set is a vertex cover in polynomial time, so VC $\in$ NP.
We'll prove NP-hardness by proving IS $\leq_c$ VC.

This time though, we'll do it **non-constructively**, without gadgets.

A **vertex cover** in a graph $G = (V, E)$ is a set $X \subseteq V$ such that every edge in E has at least one vertex in X.

VC asks: Does G contain a vertex cover of size at least k?

Theorem: VC is NP-complete.

We can verify a set is a vertex cover in polynomial time, so VC $\in$ NP. We'll prove NP-hardness by proving IS $\leq_c$ VC.

This time though, we'll do it **non-constructively**, without gadgets.

Lemma: X is an independent set if and only if $V \setminus X$ is a vertex cover.

A **vertex cover** in a graph $G = (V, E)$ is a set $X \subseteq V$ such that every edge in E has at least one vertex in X.

VC asks: Does G contain a vertex cover of size at least k?

Theorem: VC is NP-complete.

We can verify a set is a vertex cover in polynomial time, so VC $\in$ NP.
We'll prove NP-hardness by proving IS $\leq_c$ VC.

This time though, we'll do it **non-constructively**, without gadgets.

Lemma: X is an independent set if and only if $V \setminus X$ is a vertex cover.
(Because an edge intersects $V \setminus X$ if and only if it's **not** a subset of X.)

A **vertex cover** in a graph $G = (V, E)$ is a set $X \subseteq V$ such that every edge in E has at least one vertex in X.

VC asks: Does G contain a vertex cover of size at least k?

Theorem: VC is NP-complete.

We can verify a set is a vertex cover in polynomial time, so VC $\in$ NP. We'll prove NP-hardness by proving IS $\leq_c$ VC.

This time though, we'll do it **non-constructively**, without gadgets.

Lemma: X is an independent set if and only if $V \setminus X$ is a vertex cover. (Because an edge intersects $V \setminus X$ if and only if it's **not** a subset of X.)

So G contains an independent set of size at **least** k if and only if G contains a vertex of size at **most** $|V| - k$.

A **vertex cover** in a graph $G = (V, E)$ is a set $X \subseteq V$ such that every edge in E has at least one vertex in X.

VC asks: Does G contain a vertex cover of size at least k?

Theorem: VC is NP-complete.

We can verify a set is a vertex cover in polynomial time, so VC $\in$ NP. We'll prove NP-hardness by proving IS $\leq_c$ VC.

This time though, we'll do it **non-constructively**, without gadgets.

Lemma: X is an independent set if and only if $V \setminus X$ is a vertex cover. (Because an edge intersects $V \setminus X$ if and only if it's **not** a subset of X.)

So G contains an independent set of size at **least** k if and only if G contains a vertex of size at **most** $|V| - k$.

Our reduction just passes the instance $(G, |V| - k)$ to our VC-oracle. $\square$

A **vertex cover** in a graph $G = (V, E)$ is a set $X \subseteq V$ such that every edge in E has at least one vertex in X.

VC asks: Does G contain a vertex cover of size at least k?

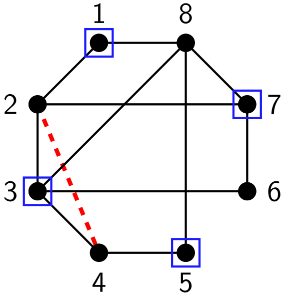
Theorem: VC is NP-complete. $\square$

A **vertex cover** in a graph $G = (V, E)$ is a set $X \subseteq V$ such that every edge in E has at least one vertex in X.

VC asks: Does G contain a vertex cover of size at least k?

Theorem: VC is NP-complete. $\square$

In video 8-2, we expressed finding maximum vertex covers in terms of **integer** linear programming for our approximation algorithm:



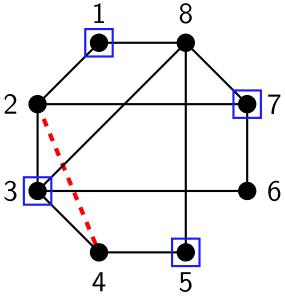
$$\begin{aligned} \textstyle\sum_v x_v &\to \min \text{subject to} \\ x_u + x_v &\geq 1 \text{ for all } \{u, v\} \in E; \\ x_v &\leq 1 \text{ for all } v \in V; \\ x_v &\geq 0 \text{ for all } v \in V; \\ x_v &\in \mathbb{N} \text{ for all } v \in V. \end{aligned}$$

A **vertex cover** in a graph $G = (V, E)$ is a set $X \subseteq V$ such that every edge in E has at least one vertex in X.

VC asks: Does G contain a vertex cover of size at least k?

Theorem: VC is NP-complete. $\square$

In video 8-2, we expressed finding maximum vertex covers in terms of **integer** linear programming for our approximation algorithm:



$$\begin{aligned} \textstyle\sum_v x_v &\to \min \text{ subject to} \\ x_u + x_v &\geq 1 \text{ for all } \{u, v\} \in E; \\ x_v &\leq 1 \text{ for all } v \in V; \\ x_v &\geq 0 \text{ for all } v \in V; \\ x_v &\in \mathbb{N} \text{ for all } v \in V. \end{aligned}$$

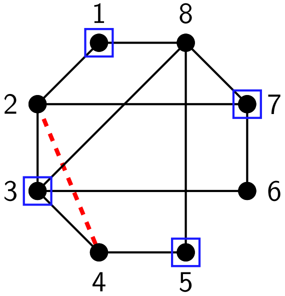
Corollary: Integer linear programming is NP-hard! $\square$

A **vertex cover** in a graph $G = (V, E)$ is a set $X \subseteq V$ such that every edge in E has at least one vertex in X.

VC asks: Does G contain a vertex cover of size at least k?

Theorem: VC is NP-complete. $\square$

In video 8-2, we expressed finding maximum vertex covers in terms of **integer** linear programming for our approximation algorithm:



$$\begin{aligned} \textstyle\sum_v x_v &\to \min \text{ subject to} \\ x_u + x_v &\geq 1 \text{ for all } \{u, v\} \in E; \\ x_v &\leq 1 \text{ for all } v \in V; \\ x_v &\geq 0 \text{ for all } v \in V; \\ x_v &\in \mathbb{N} \text{ for all } v \in V. \end{aligned}$$

Corollary: Integer linear programming is NP-hard! $\square$

Notice we reduced SAT $\leq_c$ 3-SAT $\leq_c$ IS $\leq_c$ VC $\leq_c$ ILP — by proving one problem is NP-hard, we make all our future hardness proofs easier...