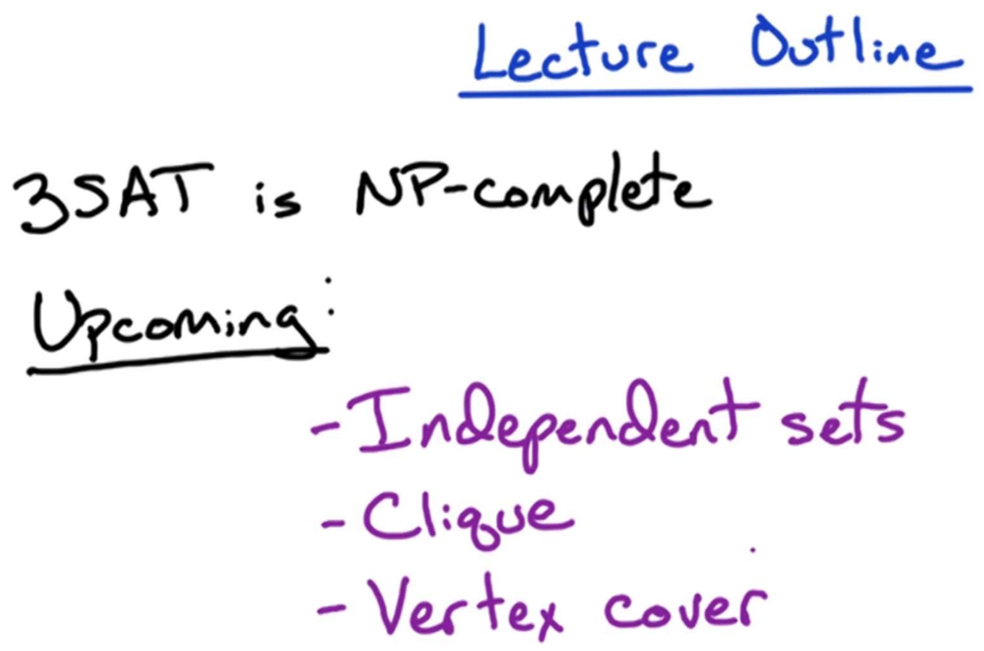
LESSON: NP3: Graph Problems

***

(Slide 1) Graph Problems Lecture Outline



Lecture Outline

3SAT is NP-complete

Upcoming:

- Independent sets
- Clique
- Vertex cover

We've seen how to show that the 3SAT problem is NP-complete. Now let's take a look at a few graph theory examples. We'll start with the Independent sets problem, then will show that the clique problem is also NP-complete. And finally, we'll show that the Vertex cover problem is NP-complete. I'll define for you each of these combinatorial problems as we go along, in case you're not familiar with them. Let's dive into the Independent sets problem.

***

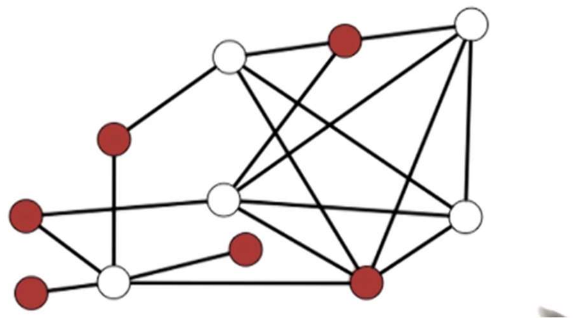
(Slide 2) Independent Set

Independent Set

For undirected $G=(V,E)$,

subset $S \subset V$ is an independent set if no edges are contained in S

(i.e., for all $x,y \in S$, $(x,y) \notin E$)



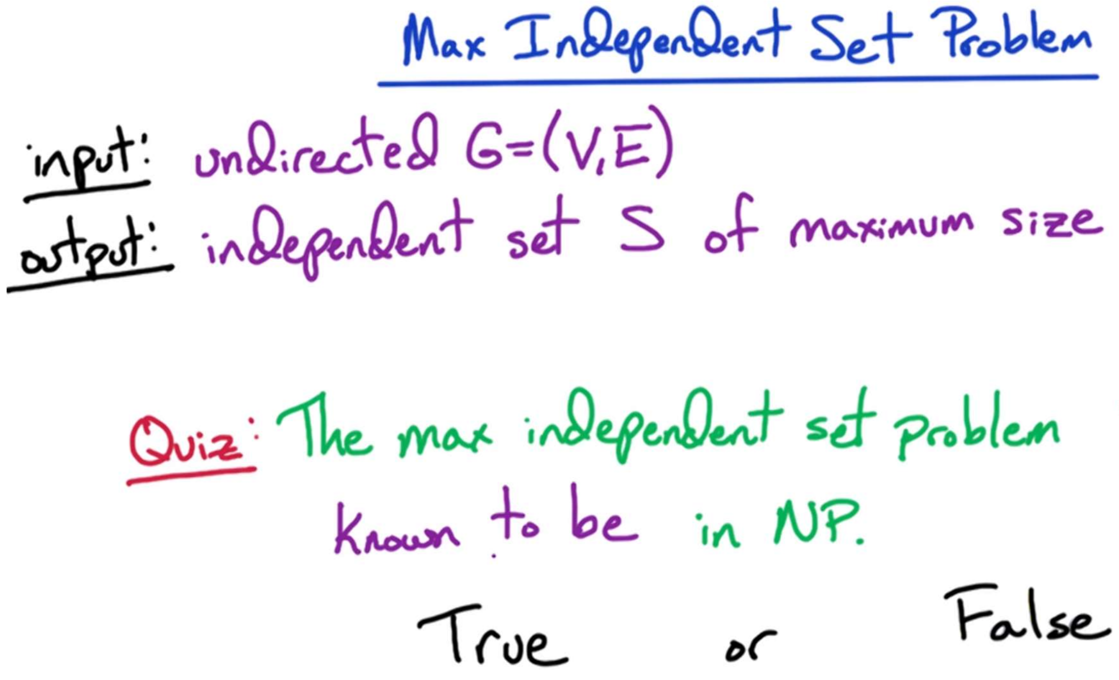
First off, what is an Independent set? Well, let's consider an undirected graph G. Here's an undirected graph. A subset of vertices is an independent set if there are no edges contained in that subset of vertices.

The green vertices here - the three vertices - are an example of an Independent set for this graph. To check that this subset is an Independent set, we can check all pairs of vertices in this subset, and we can check that these pairs are not adjacent.

Here is another example of an Independent set - this one of size six. Now, it's trivial to find very small independent sets. For example, the empty set is an Independent set, or any singleton vertex is an Independent set. The challenging problem is to find the large independent sets.

***

(1st Slide 3) Quiz: Max Independent Set



Max Independent Set Problem

input: undirected $G=(V,E)$

output: independent set S of maximum size

Quiz: The max independent set problem is known to be in NP.

True or False

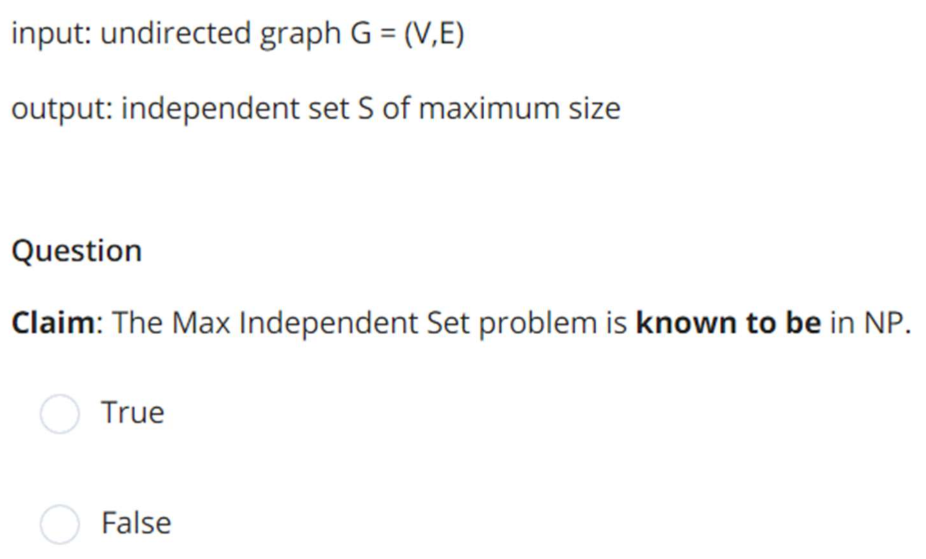
To make sure you understand what's involved in proving that a problem is in the class NP, let's take a short quiz.

The max independent set problem is in the class NP. Is that statement true or false? To be more precise, let's change this statement slightly. So let's say the max independent set problem is known to be in the class NP. Is that statement true or false? I added this qualification, “known to be”, because there are scenarios such as P equals NP that would change things.

So this will be the standard form for our True/False questions of this type.

***

NP3: Graph Problems: Quiz: Max Independent Set



input: undirected graph G = (V,E)

output: independent set S of maximum size

Question

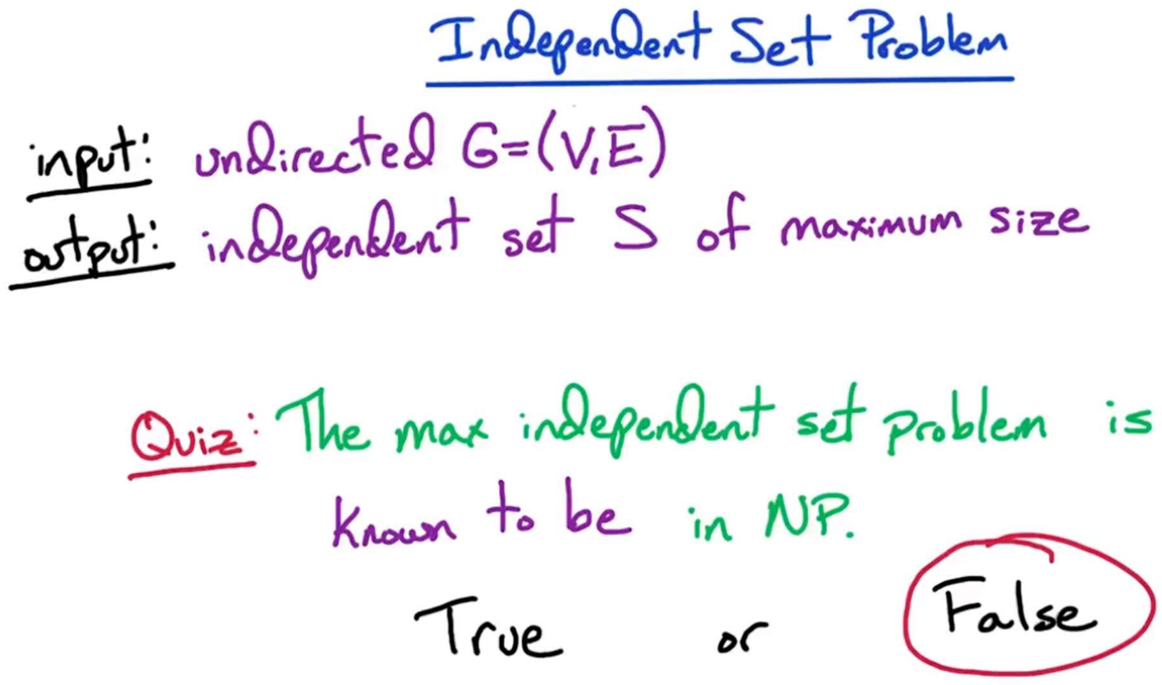
Claim: The Max Independent Set problem is **known to be** in NP.

○ True

○ False

***

(2nd Slide 3) Quiz: Max Independent Set (Answer)



Independent Set Problem

input: undirected $G=(V,E)$

output: independent set S of maximum size

Quiz: The max independent set problem is known to be in NP.

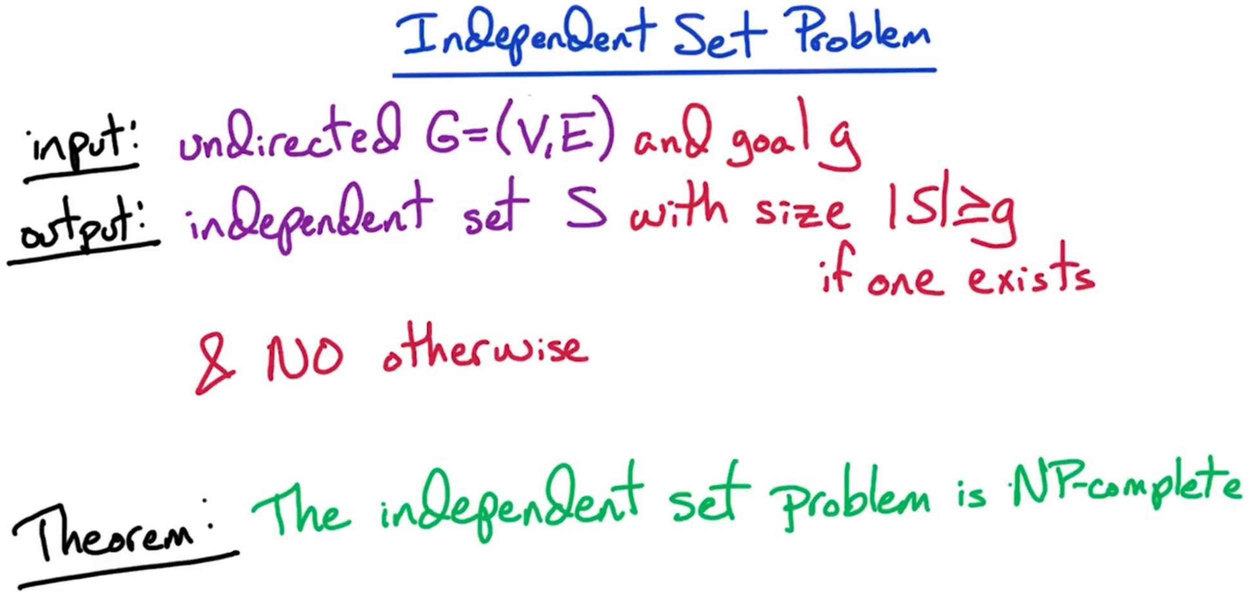
True or [✓ False]

The solution to this quiz is False. Now, why is this problem not in the class NP? Because if you give me a solution, and you claim it's of maximum size, I have no way to verify that it is, in fact, of maximum size. The only way I can verify that it is of maximum size is if I can solve this problem in polynomial time. And that only holds if P equals NP. So assuming P is not equal to NP, then the max independence set problem is not in the class NP.

Now recall that we had a similar scenario for the optimization version of Knapsack problem. When we looked at the version which tried to achieve the maximum value possible, that was not known to be in the class NP. But there was a simple fix so that the problem was in the class NP.   So let's look at the search version of the Independent set.

***

(Slide 4) Search Version



Independent Set Problem

input: undirected $G=(V,E)$ and goal g

output: independent set S with size $|S| \geq g$ if one exists

& NO otherwise

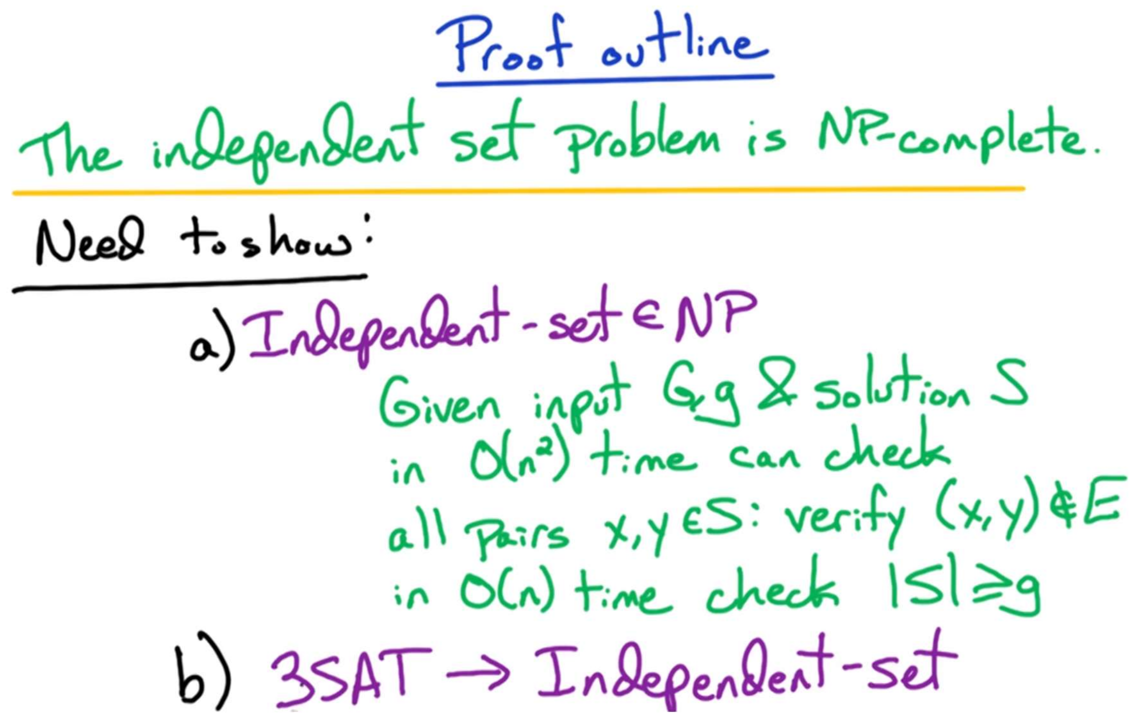
Theorem: The independent set problem is NP-complete

My input is undirected graph as before, and also a goal g. My output is an independent set S with size at least g. I want to output such a set S, if one exists, and if there is no independent set of such a size, then I simply output NO.

It will be straightforward to show that this version of the problem is a search problem and therefore lies in the class NP. And, in fact, we will prove now that the Independent set problem is NP-complete.

***

(Slide 5) Proof Outline



Proof outline

The independent set problem is NP-complete.

Need to show:

a) Independent-set $\in NP$

Given input G, g & solution S

in $O(n^2)$ time can check

all pairs $x,y \in S$: verify $(x,y) \notin E$

in $O(n)$ time check $|S| \geq g$

b) $3SAT \rightarrow$ Independent-set

Now, let's dive into the proof that the independent set problem is NP-complete.

IS ∈NP:

The first step we have to show is that this problem, the Independent set problem, lies in the class NP. To do this, we need to show that we can verify solutions in polynomial time. So given an input to the Independent set problem - this is specified by a graph G and the goal little g and given a proposed solution S, we need to verify that S is in fact the solution to the Independent set problem on this input.

In $O(n^2)$ time, we can consider all pairs of vertices in this subset S. And we can check that this pair of vertices x and y are not adjacent. There's not an edge (x, y) in the input graph. Once we check this for all pairs of vertices in S, then we know that there are no edges contained in the set S and therefore S is an independent set. And, finally, in O(n) time, we can check the size of S and see that the size of S is at least g, our goal. If those two statements are true, then we know that our proposed solution S is in fact a solution to this input.

That proves that the Independent set problem is in the class NP.

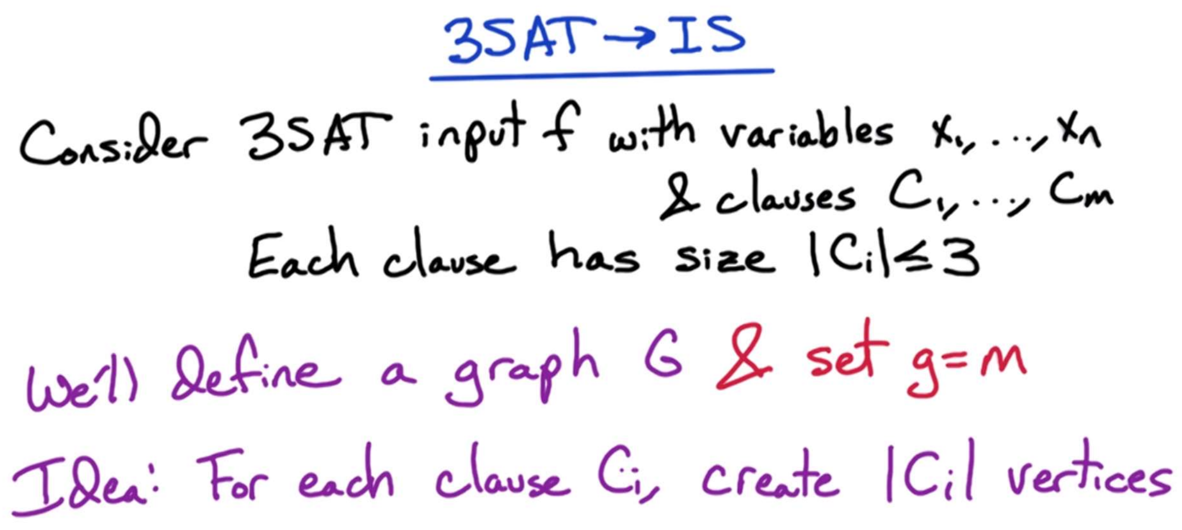
Next, we have to show that the Independent set problem is at least as hard as every problem in the class NP. How do we do that? Well, we take something which is known to be at least as hard as everything in the class NP, such as SAT or 3SAT, and we show a reduction from that known NP-complete problem to this new problem. We're going to show a reduction from the 3SAT problem to the Independent set problem.

Why consider 3SAT instead of SAT? Well, 3SAT is just easier. It's a simpler problem. We know that all the clauses are of size one, two, or three. If we consider a SAT, then the clauses are of arbitrary size.

Now, let's look at the reduction from 3SAT to Independent set.

***

(Slide 6) 3SAT IS



$3SAT \rightarrow IS$

Consider 3SAT input f with variables $x_1, \ldots, x_n$
& clauses $C_1, \ldots, C_m$

Each clause has size $|C_i| \leq 3$

We'll define a graph G & set $g = m$

Idea: For each clause C_i, create $|C_i|$ vertices

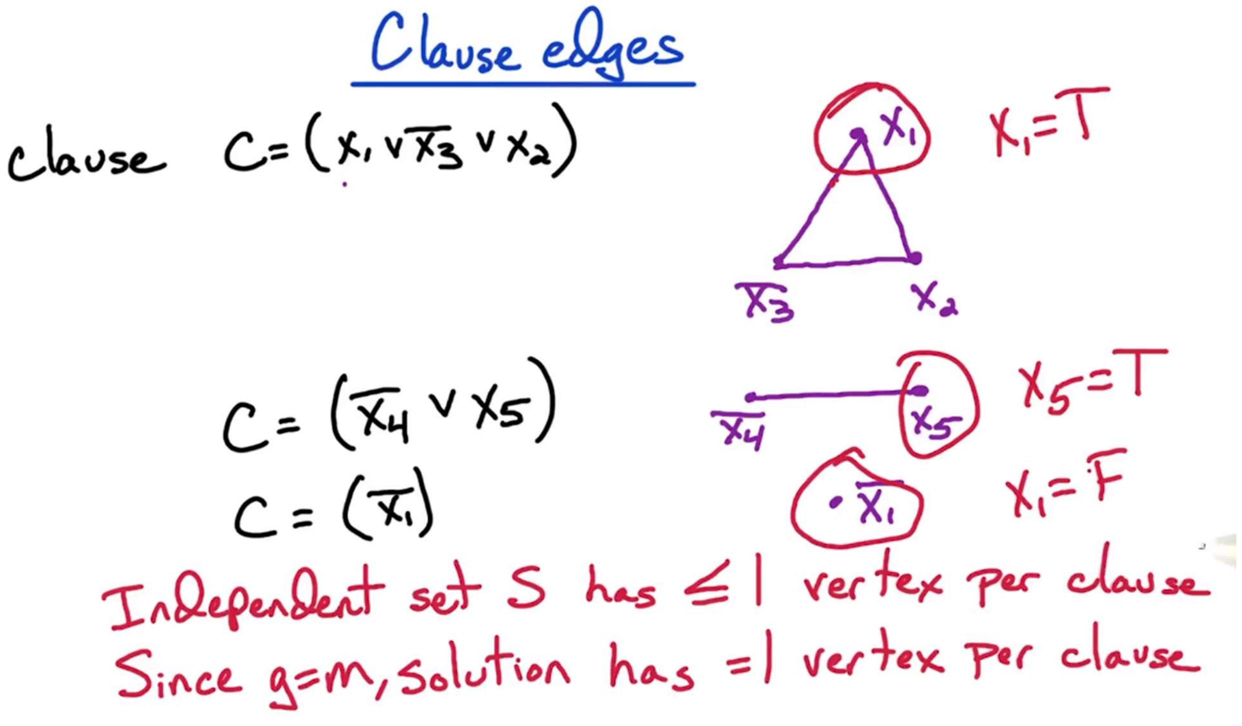
3SAT -> Independent set:

So take an input to the 3SAT problem: it is defined by a formula F, and we'll say the formula F has variables x1 thru xn, and clauses c1 through cm.   So it has n variables and m clauses. And now, we have to find an input to the Independent set problem. And recall since this is 3SAT, each of the clauses has size at most three.   So, it contains one, two or three literals. The input to the Independence set problem is a graph G.   We're going to define a graph G based on this formula F, which is the input to the 3SAT problem. We also need to specify our goal little g. It turns out we're going to set our goal g to be m, the number of clauses.

Here's the high level idea for the construction of our graph G. For each of the clauses, we're going to create a vertex in our graph G corresponding to each of the literals in this clause. So this clause has three literals, then we're going to create three vertices. Since there are m clauses, there's going to be at most 3m vertices in our graph G. And then we're going to add edges to encode this clause. And then we're going to add additional edges between vertices in different clauses in order to ensure a consistent assignment. Let's define it more precisely.

***

(Slide 7) Clause Edges


Clause edges
clause $C=(x_1 \vee \overline{x_3} \vee x_2)$
x_1
$\overline{x_3}$
x_2
$x_1=T$
$C=(\overline{x_4} \vee x_5)$
$\overline{x_4}$
x_5
$x_5=T$
$C=(\overline{x_1})$
$\overline{x_1}$
$x_1=F$
Independent set S has ≤ 1 vertex per clause
Since $g=m$, solution has $=1$ vertex per clause

There's going to be two types of edges. The first type we're going to call "clause edges" because these are defined by the clauses. The second type are "variable edges". Let's begin with the clause edges.

Let's consider a clause of size three. So, here's the clause which is C = (x1$\vee$ $\overline{x3}$ $\vee$ x2). In our graph, we're going to have three vertices corresponding to the three literals. So, this first vertex corresponds to x1. This second vertex corresponds $\overline{x3}$. And the third vertex corresponds to x2. Now, some of these literals might appear in other clauses. For example if there is another clause containing x2, we'll have another vertex corresponding to x2. So, there are multiple vertices corresponding to the same literal.

Now, how do we encode this clause? Well, we simply add a triangle. We add edges between all pairs of these three vertices. In other words, we add the complete graph on these three vertices. What if we had a clause of size 2? For example, ($\overline{x4}$ $\vee$ x5) - then we add two vertices corresponding to these two literals, and we add an edge between them. Finally, if we have a clause of size one - for example, $\overline{x1}$, then we just add a singleton vertex by itself.

This construction, the key property is that an independent set S in this graph is going to have at most one vertex per clause. If you look at these three vertices, we can only include one of them because there's edges between all pairs. Now, recall our goal was to find an Independent set of size at least m. Since every independent set has at most one vertex per clause, in order to achieve our goal of an Independent set of size at least m, this solution has to have exactly one vertex per clause.

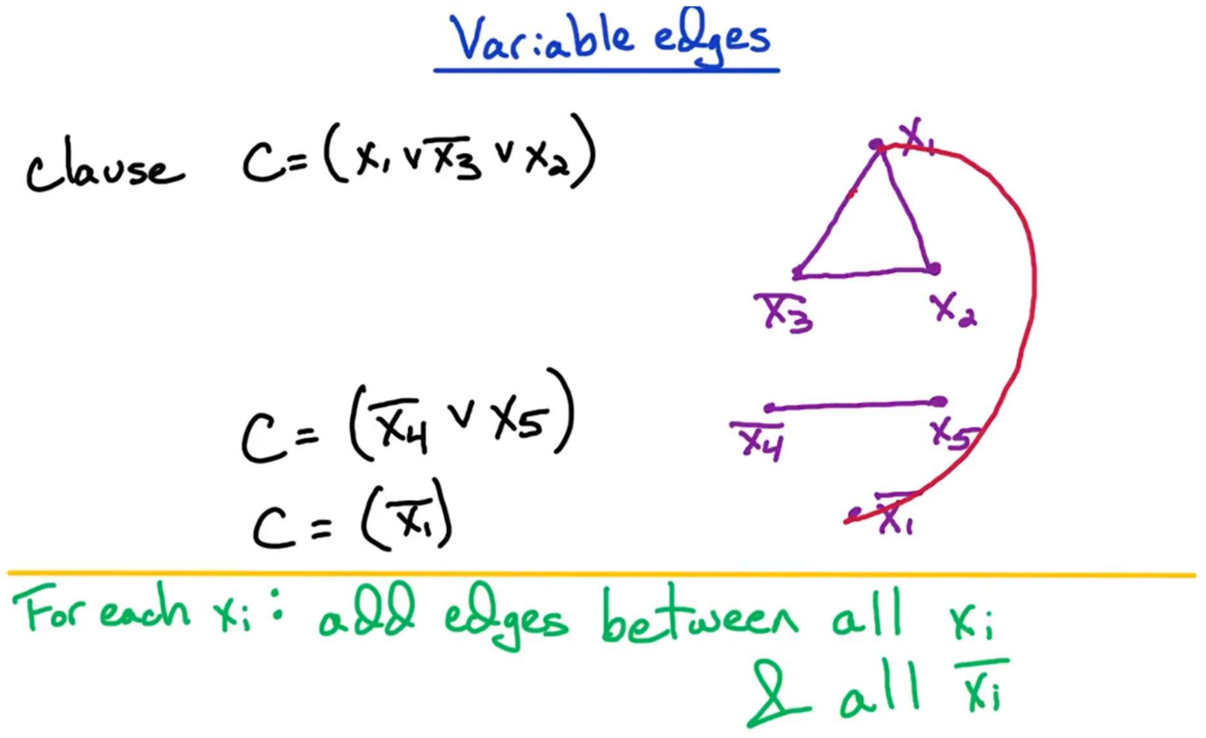
Now, this one vertex per clause will correspond to this satisfied literal in that clause. Now, there may be other satisfied literals in the clause due to other copies of that literal in other clauses, but this property that we have exactly one vertex per clause will ensure that we have at least one satisfied literal in every clause. And therefore, this solution, this Independent set, we will be able to relate it to a satisfying assignment for this original formula.

Now, what happens if I take an Independent set containing this vertex corresponding to x1, and this vertex corresponding to $\overline{x1}$? Well, if this is the graph, this is in fact an Independent set in this graph. And it can take one of these two vertices, and then I have an Independent set of my goal size of three.

Now, a natural way of converting this independent set into an assignment for the original formula is to satisfy each of the corresponding literals. So to satisfy this literal, I set x1 to be true. For this literal, I set x5 to be true. For this literal, I want to set $\overline{x1}$ to be satisfied, which means I set x1 to be false. Now I have a contradiction - this is not a valid assignment because I'm setting x1 to be True, and I'm setting x1 to be False. So I want to ensure that my Independent set corresponds to valid assignments. I never try to set a variable to True and to False.

***

(Slide 8) Variable Edges



Variable edges

clause $C = (x_1 \vee \overline{x_3} \vee x_2)$

$C = (\overline{x_4} \vee x_5)$

$C = (\overline{x_1})$

For each x_i: add edges between all x_i & all $\overline{x_i}$

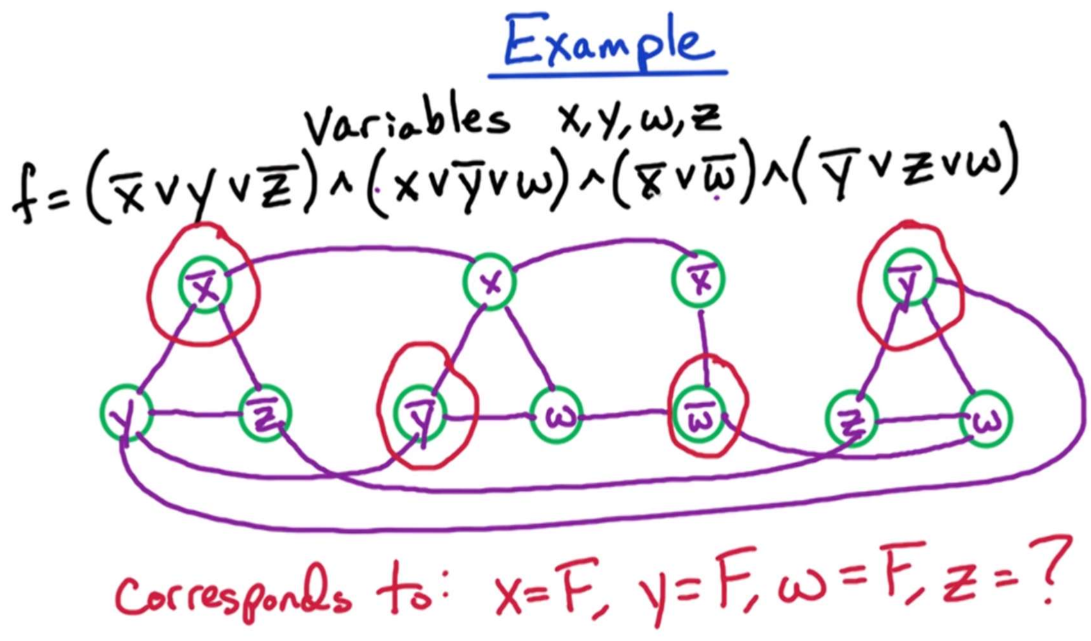
This motivates the notion of variable edges. If I had an edge from this vertex corresponding to x1 to this vertex corresponding to $\overline{x1}$, then I ensure that any independent set contains this vertex or this vertex or neither, but it cannot contain both vertices. And therefore, I'll never try to set x1 to be true and x1 to be false.

In general, for each of the variables xi, I'm going to add edges between all copies corresponding to xi and all copies corresponding to $\overline{xi}$. This will ensure that our independent set corresponds to a valid assignment and then the clause edges will ensure that our Independent set corresponds to an assignment which satisfies all of the clauses.

Let's take a look at a specific example to illustrate this construction.

***

(Slide 9) Example


Example
Variables x, y, w, z
$f = (\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee w) \wedge (\bar{x} \vee \bar{w}) \wedge (\bar{y} \vee z \vee w)$
Corresponds to: x=F, y=F, w=F, z=?

In this example, we'll have four variables which we'll label x, y, w, and z. Now, our 3SAT input is the following formula:

$f = (\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee w) \wedge (\bar{x} \vee \bar{w}) \wedge (\bar{y} \vee z \vee w)$

$(\bar{x} \vee y \vee \bar{z})$ that's the first clause. Second clause is $(x \vee \bar{y} \vee w)$. The third clause is $(\bar{x} \vee \bar{w})$. Fourth clause is $(\bar{y} \vee z \vee w)$.

Now, to start a construction of the graph, for each clause, we construct a vertex corresponding to each of the literals:

- $(\bar{x} \vee y \vee \bar{z})$: So for this clause, we have three vertices corresponding to the literals $\bar{x}$, y, and $\bar{z}$.
- $(x \vee \bar{y} \vee w)$: We have three vertices corresponding to this clause,
- $(\bar{x} \vee \bar{w})$: two vertices for this clause,
- $(\bar{y} \vee z \vee w)$: three vertices for this clause.

Now we add the clause edges:

- $(\bar{x} \vee y \vee \bar{z})$: So we add a triangle on these three vertices,
- $(x \vee \bar{y} \vee w)$: a triangle on these three,
- $(\bar{x} \vee \bar{w})$: an edge on this pair,
- $(\bar{y} \vee z \vee w)$: and a triangle on these three.

Finally, we want to add the variable edges:

- So we run an edge from x to all copies of $\bar{x}$.
- Similarly, from y to all copies of $\bar{y}$.
- From all copies of w to $\bar{w}$.
- Finally, from z to $\bar{z}$.

Now let's look at an example independent set of size four in this graph. Let's say we take $\bar{x}$ from this triangle, $\bar{y}$, $\bar{w}$, and $\bar{y}$. (circled in red on the diagram)

This independent set corresponds to the assignment:

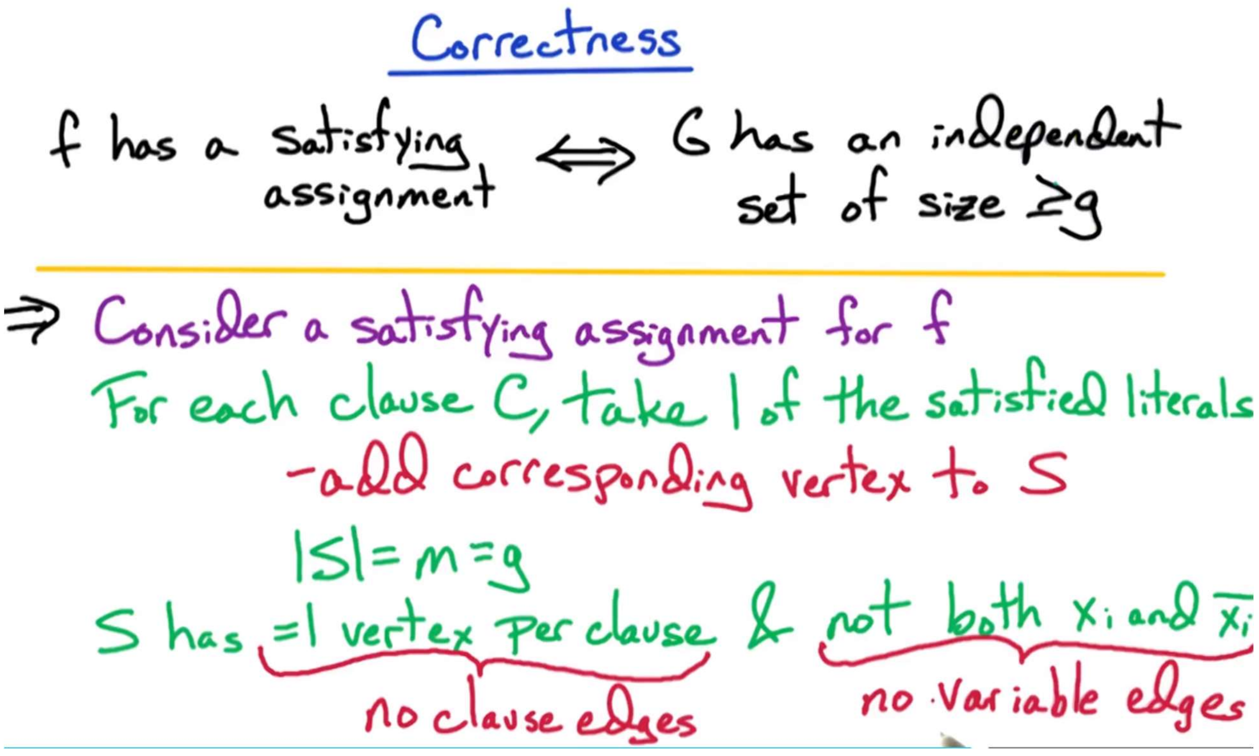
- x = False because of this literal ($\bar{x}$),
- y = False because of this ($\bar{y}$) and this literal ($\bar{y}$),
- w = False because of this literal ($\bar{w}$),
- and z has no constraints.

And notice that such an assignment satisfies this formula.

Let's prove that, in general, that an Independent set of size m in this graph corresponds to a satisfying assignment of this formula, and a satisfying assignment of this formula corresponds to an independent set of size m.

***

(Slide 10) Graph Problems Correctness



Correctness

f has a satisfying assignment $\Leftrightarrow$ G has an independent set of size $\geq g$

$\Rightarrow$ Consider a satisfying assignment for f

For each clause C, take 1 of the satisfied literals

-add corresponding vertex to S

$|S| = m = g$

S has $\underbrace{=1 \text{ vertex per clause}}_{\text{no clause edges}}$ & $\underbrace{\text{not both } x_i \text{ and } \overline{x_i}}_{\text{no variable edges}}$

So, we want to prove the following statement:

> Our 3SAT input f has a satisfying assignment if and only our graph (that we construct) has an independent set of size at least g.

So there's a solution to this Independent set problem if and only if there's a solution to the original 3SAT input.

3SAT => IS:

Let's start with the forward direction. So, consider a satisfying assignment for f, and we'll construct an independent set in this graph of size at least g.

Now, this assignment satisfies the formula f. So what do we know about it? We know for each clause, at least one of the literals in that clause is satisfied. Since there's at least one satisfied literal in every clause, take one of those satisfied literals, exactly one. Now this literal in this clause corresponds to a vertex. We're going to add that vertex into the set S.

Now, what do we know about the size of S? Well, S contains exactly one vertex per clause. So the size of S is m and recall that our goal little g was set to be m. So this set S is of the goal required size.

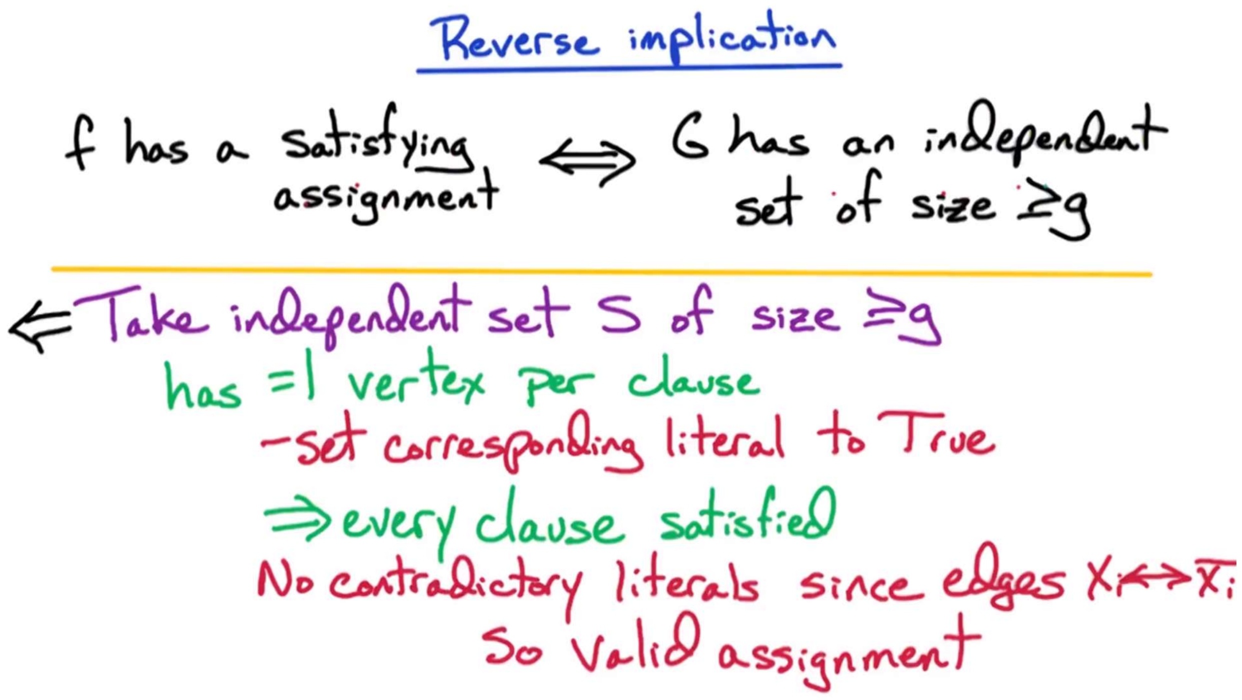
Now we just have to prove that S is an independent set. Now, S contains exactly one vertex per clause, and it never contains both xi and xi_bar. Why does it not contain both xi and xi and xi_bar ? Because it corresponds to an assignment. An assignment either sets xi to be True, in which case we might include copies of xi. Or, we set xi to be False, in which case we might include copies of xi_bar, but we wouldn't include any copies of xi.

Because there is at most one vertex per clause, we know that there are no clause edges contained in this set S. And because we never include a vertex xi and a vertex xi_bar, we know that there are no variable edges contained in this set S. So therefore, there are no edges contained in this set S, so, S is an independent set and it's an independent set of size equal to g - of our goal size.

So we've constructed an independent set of size equal to g in this graph. We've proved the forward direction - we've proven that if we take a satisfying assignment, we can construct an independent set of the desired size in this graph. Now we can try to do the reverse direction.

***

(Slide 11) Reverse Implication



Reverse implication

f has a satisfying assignment $\Longleftrightarrow$ G has an independent set of size $\geq g$

$\Leftarrow$ Take independent set S of size $\geq g$

has =1 vertex per clause

-set corresponding literal to True

$\Rightarrow$ every clause satisfied

No contradictory literals since edges $x_i \leftrightarrow \bar{x}_i$

So valid assignment

IS => 3SAT:

Now let's look at the reverse implication. Now since this independent set has size at least g and g is set to m, then we know that this independent set has exactly one vertex in each of the clauses (actually in each of the triangles corresponding to the clauses). Now this vertex corresponds to a literal in the original formula. So, we set that corresponding literal to True. So, we satisfy that literal. Since every clause has a satisfied literal, then we know that every clause is satisfied and therefore the formula is satisfied.

But does this correspond to an assignment, a valid assignment? But, notice we have no contradictory literals in this set. Why? Because we added edges between xi and xi_bar so we can never include vertex xi and a vertex xi_bar in an independent set. Therefore, we never attempt to set xi to True and xi to False. So, this assignment we constructed corresponds to a valid assignment. This is a valid assignment and it says every clause.

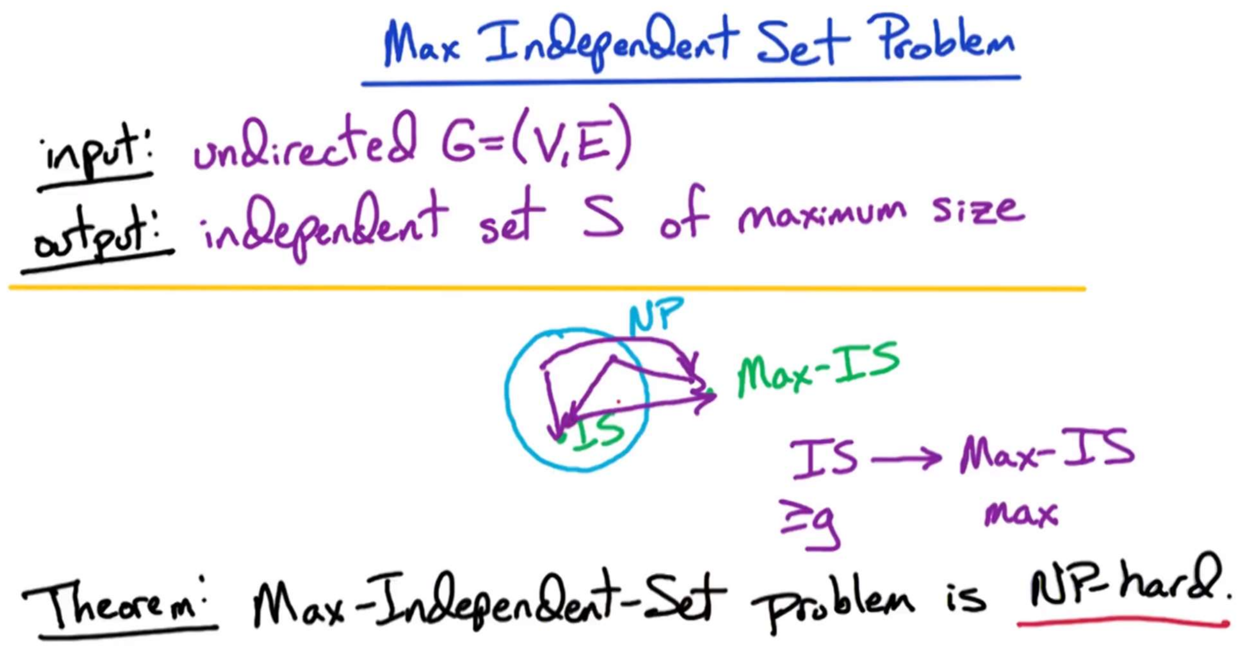
Therefore, it satisfies the formula f, so we've taken an independent set of size at least g and we construct a satisfying assignment. This completes the proof of the reverse implication.

So we've shown this equivalence. This proves that the reduction from 3SAT to Independent set is correct and it shows how to take an independent set and construct a satisfying assignment. And if there is no independent set of size at least g then there is no satisfying assignment.

So this completes the proof that the Independent set problem is NP-complete.

***

(Slide 12) NP-hard



Max Independent Set Problem

input: undirected $G=(V,E)$

output: independent set S of maximum size

$$\begin{tikzcd} \text{IS} \arrow[r] & \text{Max-IS} \end{tikzcd}$$

$\geq g$ max

Theorem: Max-Independent-Set problem is NP-hard.

We just saw that the independent set problem is NP-complete. I want to go back and look at this optimization version - the Max Independent Set Problem, and I want to see what this implies about this problem.

Now we have this class of problems NP, the Independent set problem which we just proved is NP-complete lies in this class - and there is a reduction from every problem in this class to the Independent set problem; It's at least as hard as every problem in this class of NP. Now, the Max Independent set problem is not known to lie in the class NP, so it's outside this class.

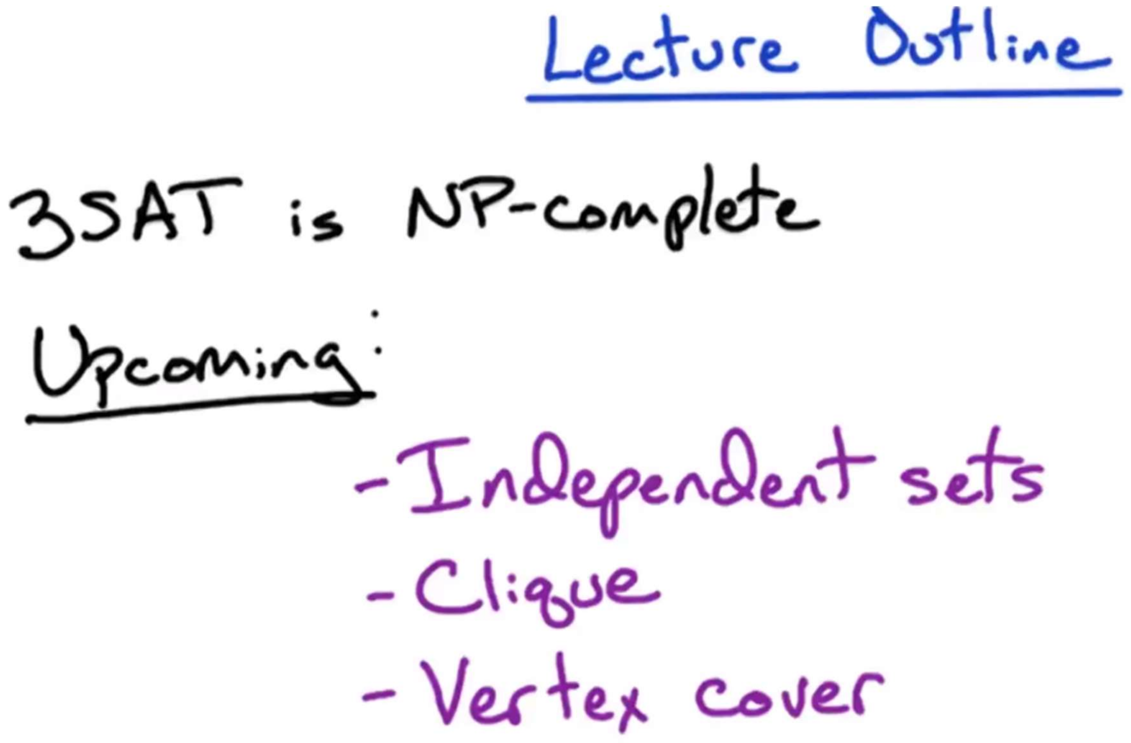
Notice it's quite straightforward to reduce the Independent set problem to the Max Independent set (Max-IS) problem. Here (points at IS), I'm looking for an independent set of size at least g. Here (points at Max-IS), I'm looking for the maximum independent set. If I find the maximum independent set and I check whether that size is at least g, that either gives me a solution, or it tells me there's no solution. So it's quite straightforward to reduce the search version to the optimization version. That means I have a reduction from the Independent set problem to the Max Independent set problem, and, in fact, I have a reduction from every problem in NP, to the Max Independent set problem, either going indirectly through the Independent set problem or directly.

What does that mean? That means the Max Independent set problem is at least as hard as everything in the class NP. Now, if we knew it was in the class NP, then we would know it's NP-complete. But we don't know that. But how do we denote that it's at least as hard as everything in the class NP? We say that the Max Independent set problem is **NP-hard**. Notice it's NP-hard, not NP-complete.

So NP-hard means that it's at least as hard as everything in the class NP. So there is a reduction from everything in the class NP to this problem Max Independent set. So if we can solve Max Independent set in polynomial time, then we can solve everything in NP in polynomial time. To be NP-complete, such as the Independent Set problem, then it has to be NP-hard, and it also has to lie in the set NP. So complete problems are the hardest in the set. Hard problems are at least as hard as everything in the set.

***

(Slide 13) Graph Problems Lecture Outline



Lecture Outline

3SAT is NP-complete

Upcoming:

- Independent sets
- Clique
- Vertex cover

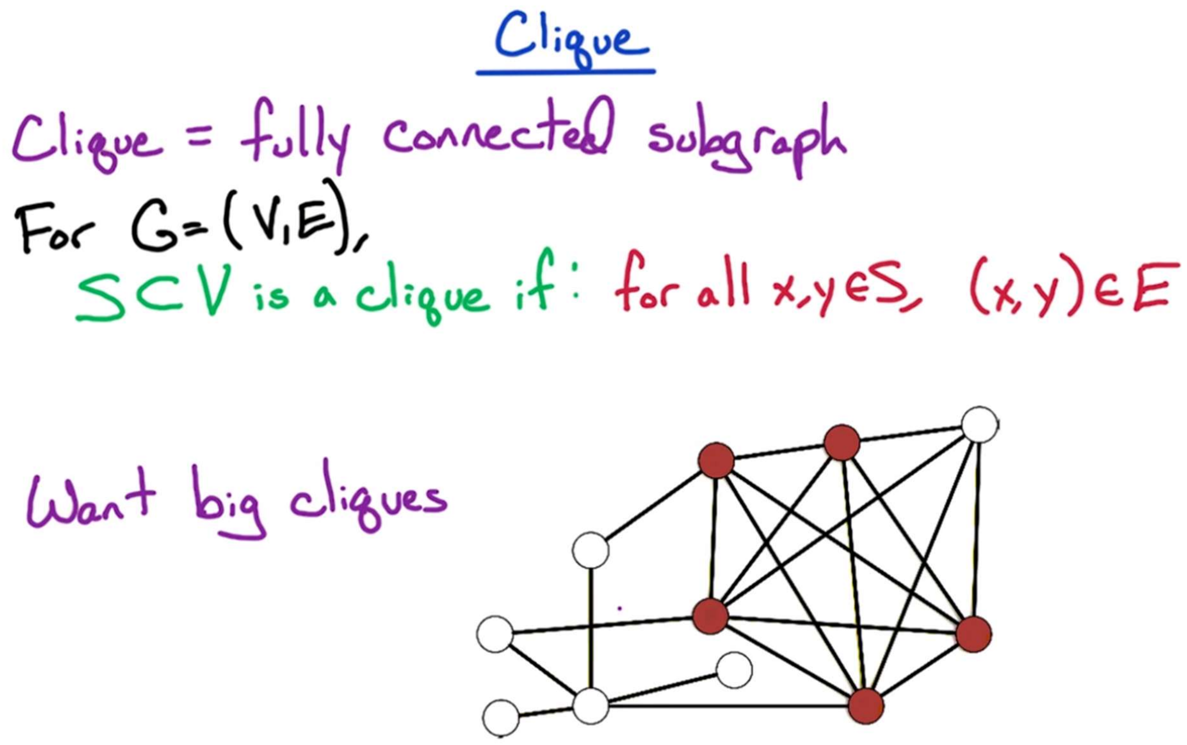
We just saw how to show that the Independent set problem is NP-complete. Now that reduction was quite interesting since it involved a reduction from 3SAT, which is a logic problem, to this graph problem, Independent set.

Now, will show that the Clique problem and Vertex cover problem are NP-complete. These will be much easier since we now have a graph problem, Independent set, to start from.

Let's dive into the Clique problem.

***

(Slide 14) Clique



Clique

Clique = fully connected subgraph

For $G=(V,E)$,

$S \subset V$ is a clique if: for all $x,y \in S$, $(x,y) \in E$

Want big cliques

A clique is a fully connected subgraph. Let me define it more formally for you.

The **Clique** problem is defined for <u>undirected graphs</u>. So, let's consider an undirected graph G.

Here's an example of an undirected graph that we'll use to illustrate a clique.

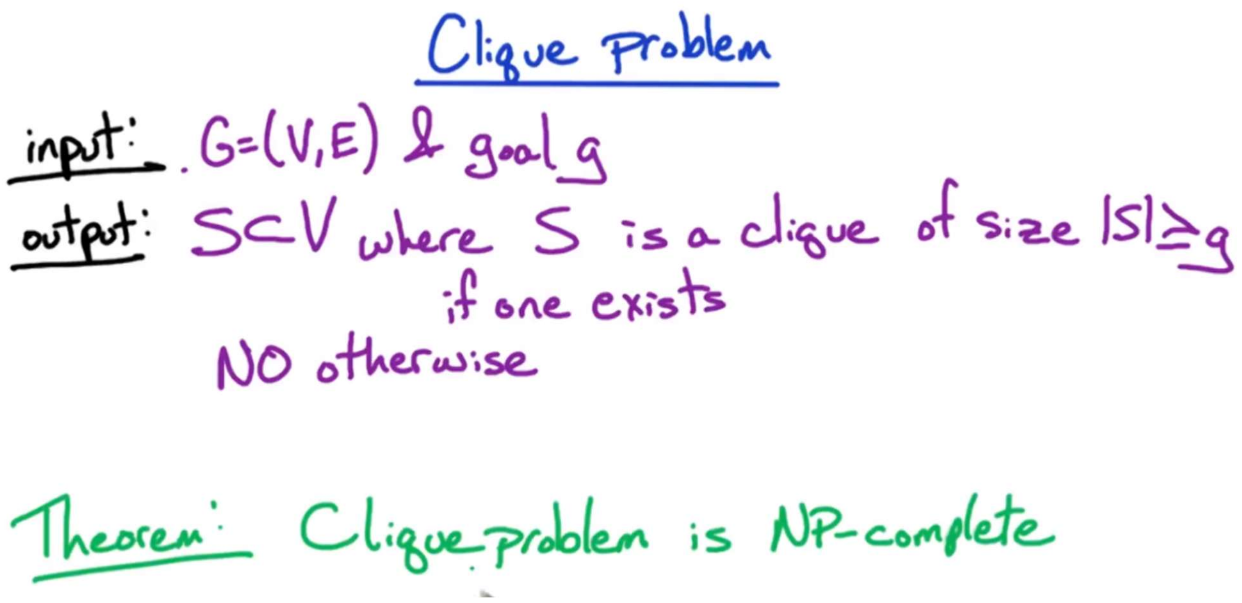
> A subset S of vertices is a **clique** if the following holds: for all pairs of vertices in this subset, say the pair is x and y, then x and y are connected by an edge.

So, all pairs of vertices in this subset S are connected by an edge. Here's an example of a clique of size five in this graph. Notice that all $\binom{5}{2}$ - five choose two - pairs are connected by an edge. And in fact, this is the largest clique in this graph. This vertex can almost be added to the clique but it's not connected to this vertex over here.

The challenging problem is to find large cliques. Why is that the case? Well, very small cliques are trivial to find. For example, the empty set of vertices is always a trivial clique. Moreover, any singleton vertex by itself is also a clique. And if you take any edge, the two endpoints of the edge form a clique. So very small cliques are easy to find. The difficult problem is to find the largest clique or the clique of maximum size.

***

(Slide 15) Clique: Search Version


Clique Problem
input: G=(V,E) & goal g
output: S⊂V where S is a clique of size |S|≥g
if one exists
NO otherwise
Theorem: Clique problem is NP-complete

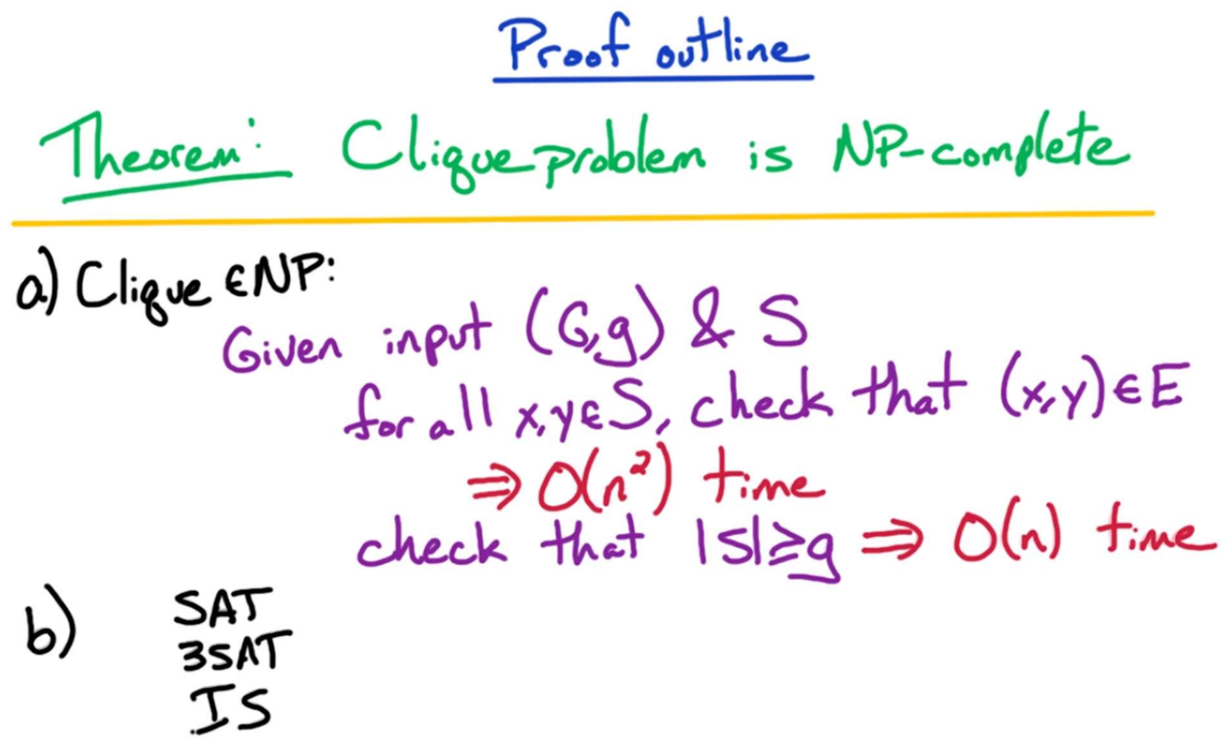
Now let's formally define the Clique problem. The input to the clique problem is an undirected graph G, and it goal little g. The output from the clique problem is a subset of vertices S, which is a clique of size at least g.

Now we're trying to find the largest clique possible. Therefore we're trying to output a clique of size at least g. If we outputted a clique of size at most g, that would be quite trivial because we could always output the empty set. Now we output such a clique if one exists, and if no such clique exists in the graph, then we output NO.

And what we're going to prove now, is that the Clique problem is NP-complete.

***

(Slide 16) Clique: Proof Outline



Proof outline

Theorem: Clique problem is NP-complete

a) Clique $\in$ NP:

Given input (G,g) & S

for all $x,y \in S$, check that $(x,y) \in E$

$\Rightarrow O(n^2)$ time

check that $|S| \geq g \Rightarrow O(n)$ time

b) SAT

3SAT

IS

Let's dive into the proof that the Clique problem is NP-complete. Now, there's two parts to the proof.

First part: Clique $\in$ NP:

The first thing we have to prove is that the Clique problem lies in the class NP. As usual this is quite straightforward.

So consider an input to the Clique problem, a graph G, and a goal little g. And consider our proposed solution S. Now, we have to verify in polynomial time that S is a solution - S is a clique of size at least g. To verify that S is a clique, we consider all pairs of vertices in S. And we check that that a pair of vertices, x and y, are connected by an edge in the input graph G.

Now, how long does this take us to do? It takes us at most $O(n^2)$ time to consider each pair. And then for each particular pair, it takes as the most $O(n)$ time to check whether they're connected by an edge. So this takes in most $O(n^3)$ time by a trivial algorithm. Now that's sufficient for this proof since we just have to show polynomial time. But if you give a little bit of thought, you can easily do it in $O(n^2)$ time as well.

That verifies that S is a clique. Now we have to verify that the size of S is at least little g. That takes us O(n) time, and therefore, it takes us polynomial time to verify that S is a solution to the clique problem on this input instance.

That establishes that the Clique problem lies in the class NP. It's a search problem. We can verify solutions in polynomial time.

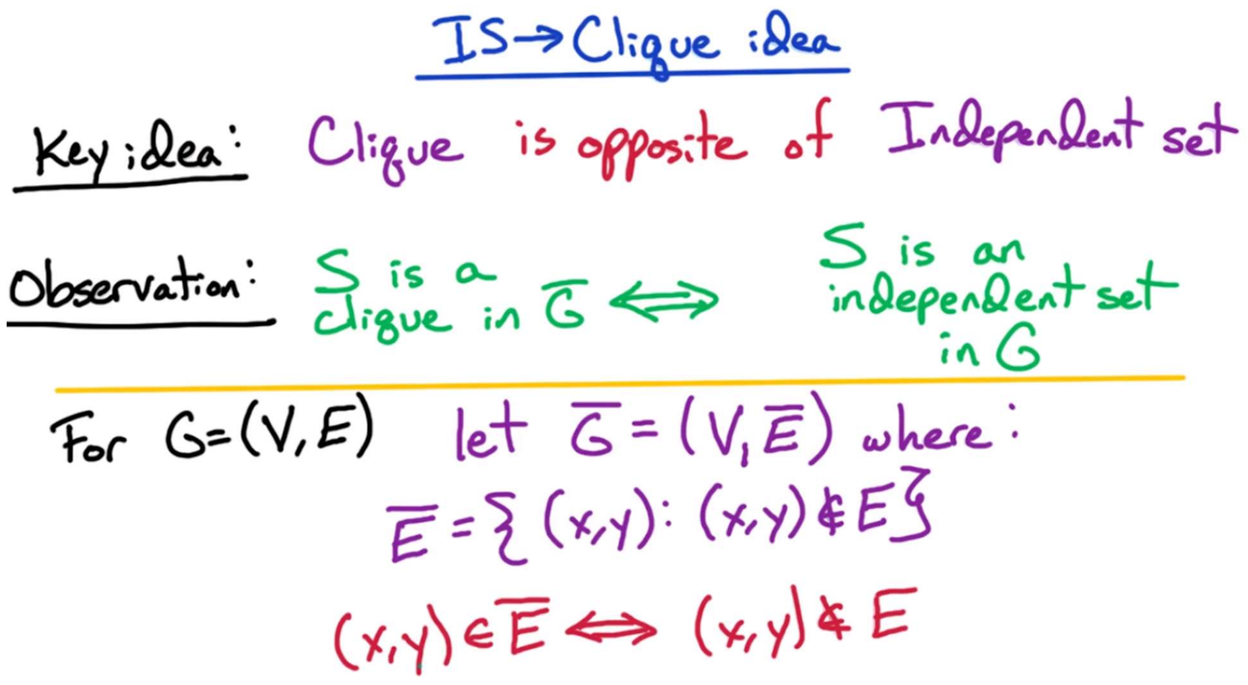
Second part: The second step we have to show is that the Clique problem is at least as hard as every problem in the class NP. How do we do that? Well, we take a known NP-complete problem, and we show a reduction from the known NP-complete problem to the Clique problem.

Which problems do we know are NP-complete? We know the SAT problem is NP-complete. We're going to take one of these. Which one are we going to take? We're going to take the one which is most similar to the Clique problem.

Clique is a graph problem, so we'll take the independent set problem, which is also a graph problem. So, we'll show that the Independent set problem reduces to the Clique problem.

***

(Slide 17) IS Clique Idea



IS → Clique idea

Key idea: Clique is opposite of Independent set

Observation: S is a clique in $\overline{G}$ $\iff$ S is an independent set in G

For $G=(V,E)$ let $\overline{G}=(V,\overline{E})$ where:

$$\overline{E}=\{(x,y):(x,y)\notin E\}$$

$$(x,y)\in\overline{E} \iff (x,y)\notin E$$

Here's a key idea for relating independent sets and cliques in a graph. Let's compare a clique and independent set:

Clique	Independent set
A clique is fully connected which means that I have **all edges** within the pairs of S.	In contrast, an independent set has **no edges** within the set S.
Whereas for clique, for any pair x,y within S, **they are connected** by an edge.	For an independent set, for any pair x,y within S, **they are not connected** by an edge.

So they are opposites of each other. Clique is opposite of independent set and vice versa.

Now we're going to have to take an undirected graph where we want to solve the Independent set problem on this graph. Now, we are assuming that we have an algorithm to solve the Clique problem. So we want to transform this input for the independent set problem into an equivalent input for the clique problem.

What should we do? Well, we should take the opposite of this graph. What exactly do we mean by the opposite? Well, let's formalize it now. We're going to denote the opposite of G as $\overline{G}$ - the complement. It's going to have the same vertex set and the edges are going to be the complement of the edges here. So the edges in $\overline{E}$ are those pairs (x,y) which are not an edge in E.

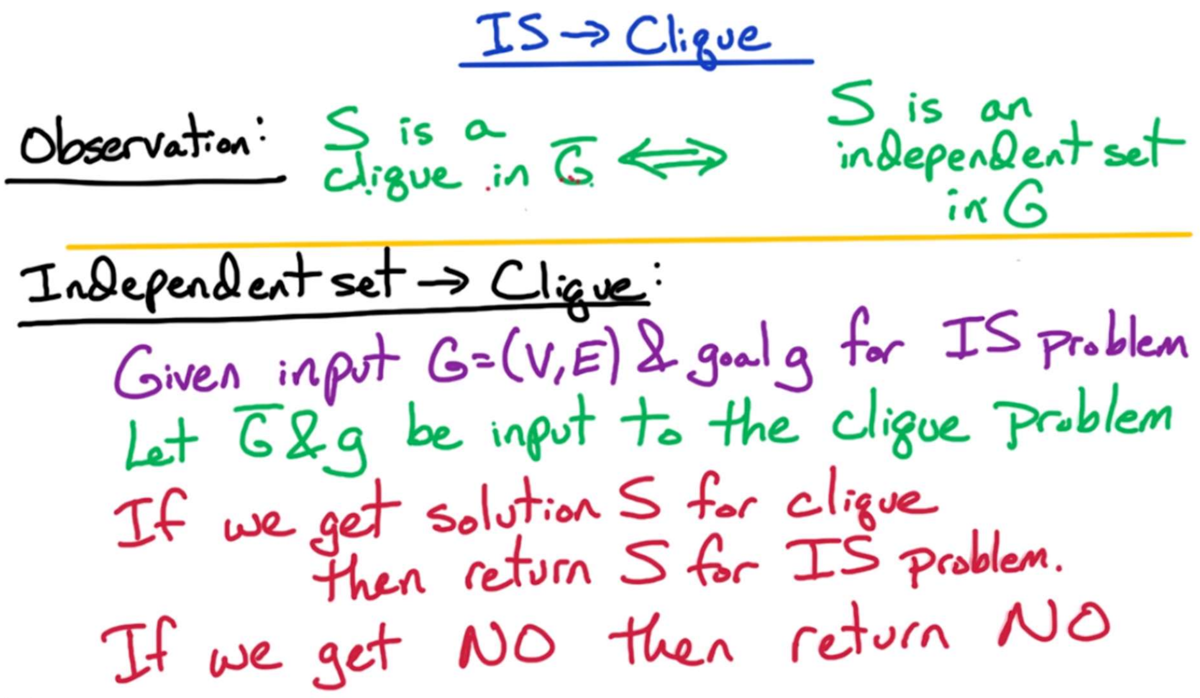
So in other words, a pair (x,y) is an edge in $\overline{G}$ if and only if the pair (x,y) is not an edge in E. So, it's not an edge in E, then it's an edge in $\overline{E}$. If it is an edge in E, then it's not an edge in $\overline{E}$.

Now, we can formalize this key idea - that clique and independent sets are opposites of each other. Now, our observation is that S is an independent set in the original graph G if and only if that same set S is a clique in the complement graph $\overline{G}$. So, if I look at a graph G and its complement graph, its opposite graph, then a set S is an independent set in the original graph, if and only if the set S is a clique in the opposite graph $\overline{G}$.

Now I call it an observation because it doesn't really require a proof after observing this statement. If S is an independent set, that means for all pairs of vertices $x,y \in S$, they are not connected by an edge. And therefore, in $\overline{G}$, they are connected by an edge. And therefore, S is fully connected in $\overline{G}$. And similarly, if S is a clique in $\overline{G}$ that means all pairs x,y are connected by an edge then in the original graph G, all pairs are not connected by an edge. And therefore, it's an independent set. S does not contain any edges in the original graph G.

***

(Slide 18) IS Clique



IS → Clique

Observation: S is a clique in $\overline{G}$ $\Longleftrightarrow$ S is an independent set in G

Independent set → Clique:

Given input G=(V,E) & goal g for IS problem
Let $\overline{G}$ & g be input to the clique problem
If we get solution S for clique then return S for IS problem.
If we get NO then return NO

Given this observation, it's now straightforward to show a reduction from the Independent set problem to the Clique problem.

Now we're doing a reduction from Independent set to Clique, so we have to take an input for the Independent set problem and transform it to an input for the Clique problem.

So consider an input for the Independent set problem. This is defined by a graph G and a goal little g. We take the opposite of the G, that's $\overline{G}$ and we use $\overline{G}$ and the same little g as our input to the Clique problem.

Now we use our observation. If we get a set S which is a clique in the opposite graph $\overline{G}$, then we know that S is an independent set in the original graph G. Now if we get a solution S for the Clique problem, that means that S is a clique of size at least little g in $\overline{G}$. Then we return S as a solution to the Independent set problem, because we know S is an independent set of size at least little g in the original graph capital G.

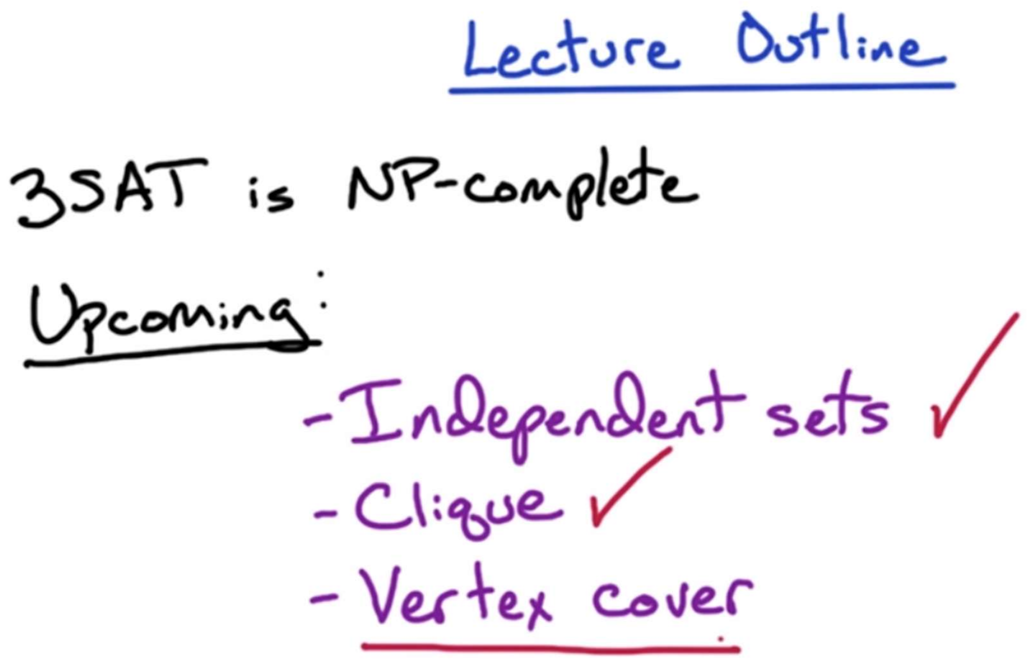
Now if our clique problem returns NO – So, there's no clique of size at least little g in $\overline{G}$ - then we return NO for the Independent set problem because we know that there's no independent set of size at least a little g in the original graph G because if there was an independent set of sufficient size in capital G, then we know there would be a clique of sufficient size in the opposite graph $\overline{G}$.

This completes the definition of the reduction and the proof of correctness follows from this observation.

Therefore, we've shown that the clique problem is NP-complete.

***

(Slide 19) Graph Problems Lecture Outline III



Lecture Outline

3SAT is NP-complete

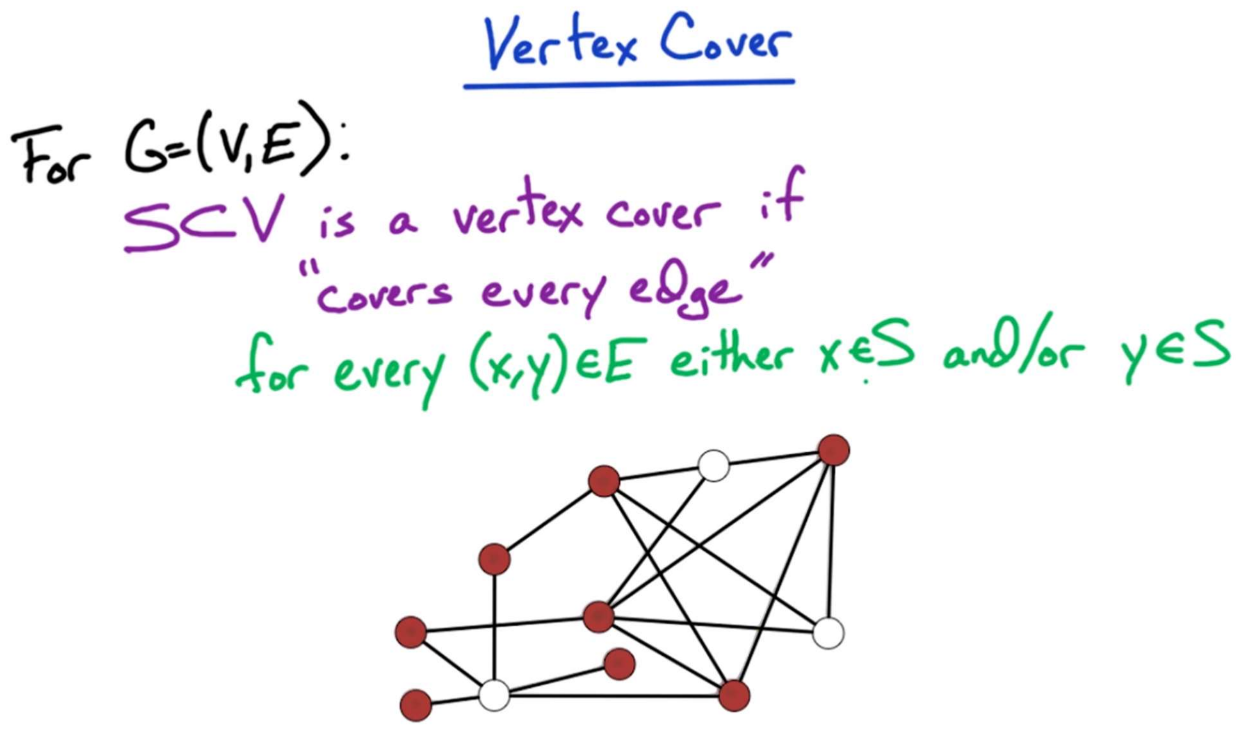
Upcoming:

- Independent sets ✓
- Clique ✓
- Vertex cover

We've shown clique is NP-complete, now let's show that Vertex cover is NP-complete.

***

(Slide 20) Vertex Cover



Vertex Cover

For $G=(V,E)$:

$S \subset V$ is a vertex cover if

"covers every edge"

for every $(x,y) \in E$ either $x \in S$ and/or $y \in S$

What exactly is a vertex cover? Let's formally define it. Once again, we're going to consider undirected graph G. Here's an example graph.

A subset of vertices is a **vertex cover** if it **covers** every edge.

The red vertices in this graph, are a vertex cover for the graph G.

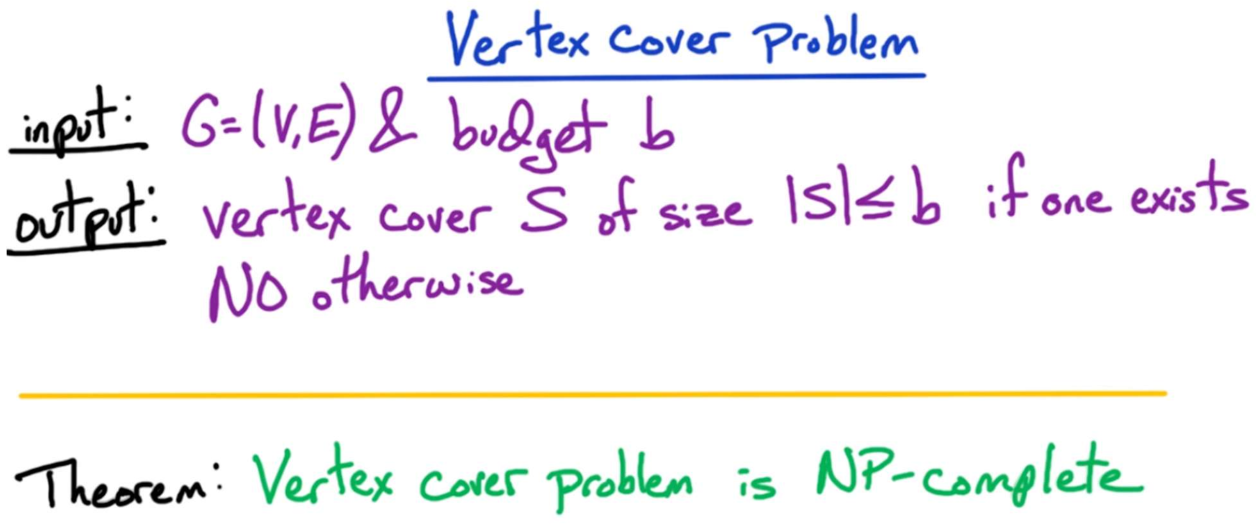
What exactly does it mean that the set S covers every edge? Well, if you look at this example, for every edge, such as this edge, at least one of the two endpoints is in the red set - is in the vertex cover. For this edge, one endpoint is in. For this edge, both endpoints are in the red set.

Formally, to cover every edge, we mean that for every edge of the graph, say (x,y), either endpoint x is in the set S and/or y is in the set S. So, both end points can be in there or at least one of the endpoints is in there.

Now it's easy to find a large vertex cover. For instance, I can include all the vertices in the vertex cover. And, this (the red set) is a large vertex cover for this graph.

***

(Slide 21) VC: Search Version



Vertex cover Problem

input: $G=(V,E)$ & budget b

output: vertex cover S of size $|S| \leq b$ if one exists

NO otherwise

Theorem: Vertex cover problem is NP-complete

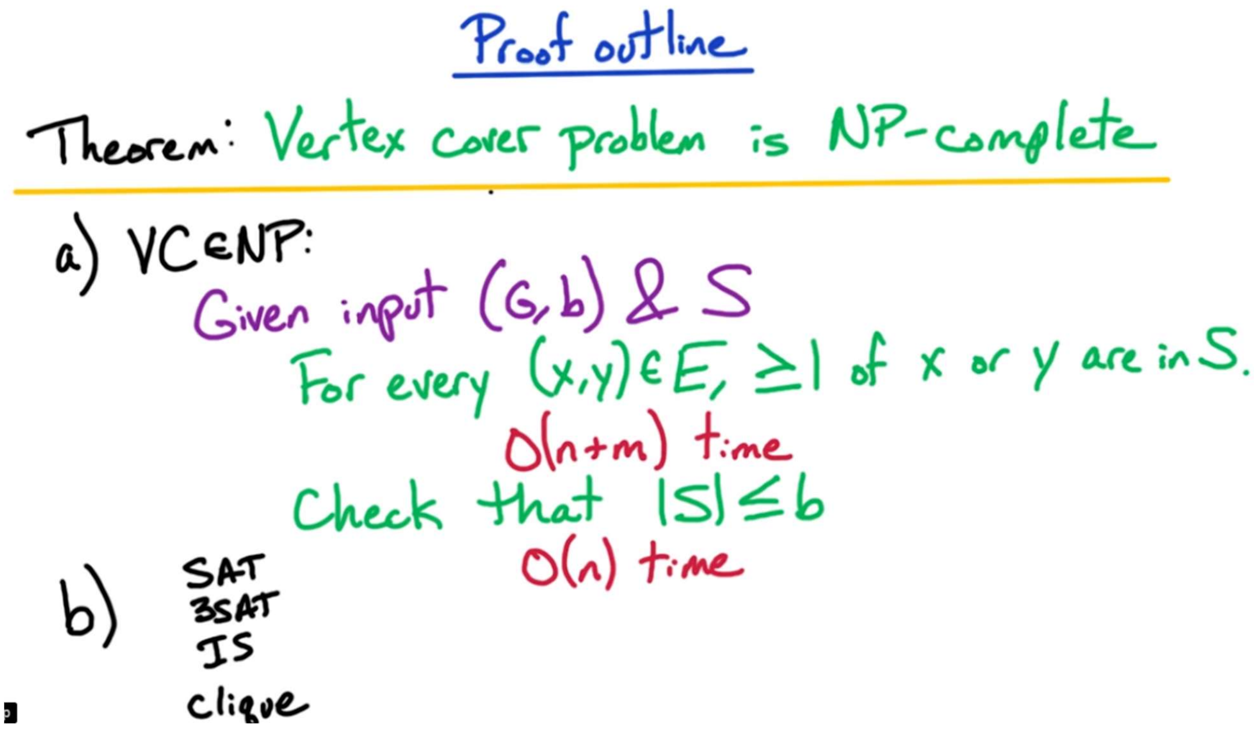
The input to the vertex cover problem is undirected graph G, and the goal which we'll call a budget, little b.

Why do we change a terminology from ‘goal’ to ‘budget’? Because instead of trying to find the maximum size clique or independent set, we're trying to find the minimum size vertex cover. The output to the problem is a vertex cover, S, of size at most b. We're trying to find the smallest possible, if one exists. And if no such vertex cover exists in the graph G, then we simply output NO.

We're going to prove now that the Vertex cover problem is NP-complete.

***

(Slide 22) VC: Proof Outline



Proof outline

Theorem: Vertex cover problem is NP-complete

a) VC $\in$ NP:

Given input (G,b) & S

For every $(x,y) \in E$, ≥ 1 of x or y are in S.

$O(n+m)$ time

Check that $|S| \leq b$

$O(n)$ time

b) SAT

3SAT

IS

clique

To prove that the Vertex cover problem is NP-complete, there are two parts.

VC $\in$ NP:

The first part, once again, is to prove that it's in the class NP. So, consider an input to the Vertex cover problem (i.e., a graph G and a budget b) and consider a proposed solution S (this is a proposed vertex cover in the graph G). Can we verify in polynomial time that S is a vertex cover of requisite size in the input graph G?

First, in order to verify that S is a vertex cover, we have to check every edge of the graph and we have to check that at least one of the two endpoints is in the set S. This can be done in linear time - in O(n+m) time.

Next, we have to check that the set S is sufficiently small. This can be easily done in linear time – in O(n) time.

This show that in polynomial time, we can verify that S is a solution to this input instance.

Now, we can do the second, more non-trivial, aspect of the proof.   Now we have to show that Vertex cover problem is at least as hard as every problem in class NP.

So, we have to take a known NP-complete problem. We now have four problems that we know are NP-complete: SAT, 3SAT, Independent set, and Clique. And we have to reduce one of them to the Vertex cover problem.

It's most natural to take one of the graph problems – either, Independent set or Clique. These are quite similar to each other so it doesn't matter too much. We're going to take Independent set problem and reduce it to the Vertex cove problem.

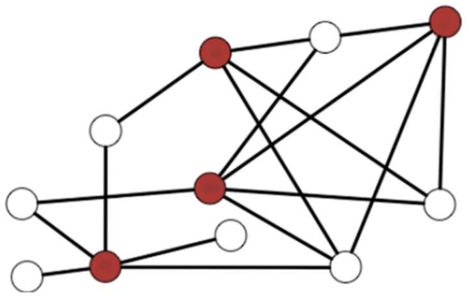
Now, let's do the reduction from the Independent set problem to the Vertex cover problem.

***

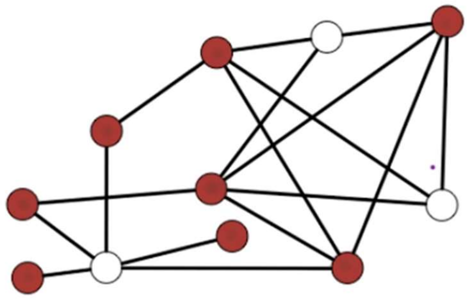
(Slide 23) VC: Reduction Idea

Reduction idea

Claim: S is a vertex cover $\Longleftrightarrow$ $\overline{S}$ is an independent set.

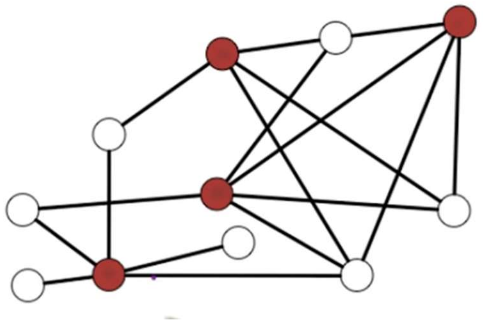


To get an idea for the reduction, let's look at our earlier example of a vertex cover:



The red vertices are our vertex cover in this graph. What do you notice about the vertices that are not red (the white vertices)? They form an independent set in this graph.

Let's take a look at another example of a vertex cover:

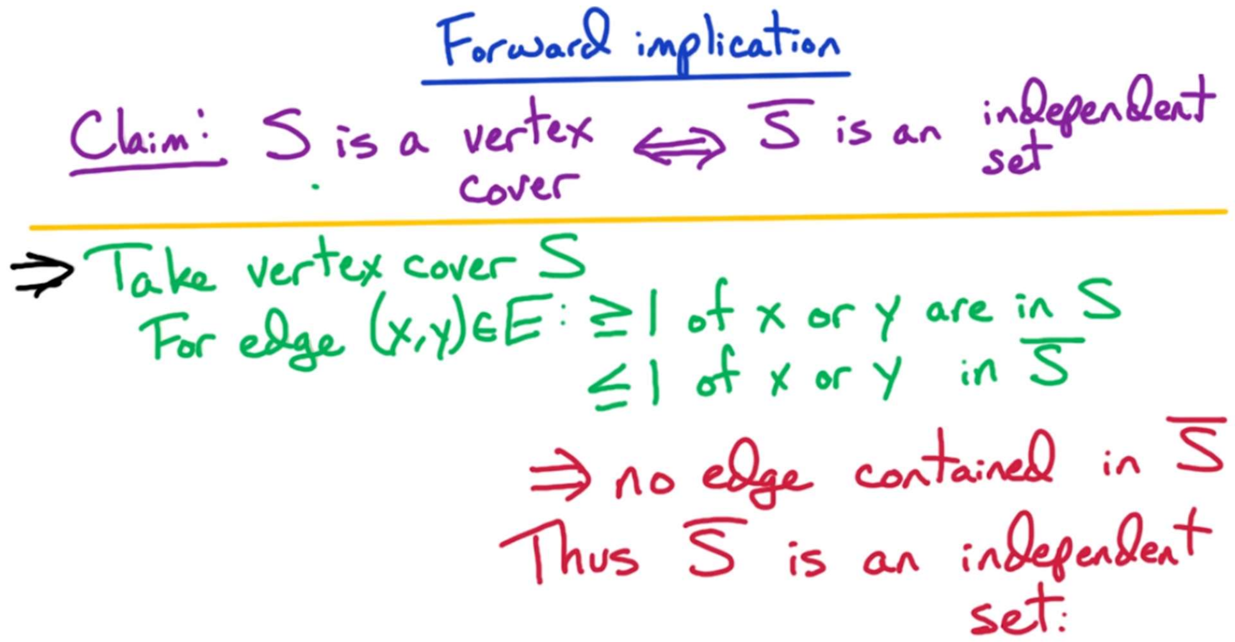


There's a minimum sized vertex cover. What do you notice about the white vertices in this graph? They also are an independent set in this graph. We claim this holds in general: S is a vertex cover in the graph if and only if $\bar{S}$, the complement of S is an independent set. So the red vertices are a vertex cover and the non-red vertices are an independent set and vice versa.

Let's prove this claim and then the reduction from independent set to vertex cover will be straightforward.

***

(Slide 24) Forward Implication



Forward implication

Claim: S is a vertex cover $\Leftrightarrow$ $\overline{S}$ is an independent set

$\Rightarrow$ Take vertex cover S

For edge $(x,y) \in E$: ≥ 1 of x or y are in S

≤ 1 of x or y in $\overline{S}$

$\Rightarrow$ no edge contained in $\overline{S}$

Thus $\overline{S}$ is an independent set.

Let's start with the forward implication of this claim.

S is a vertex cover => S is an independent set:

So let's take a vertex cover S and let's prove that S̄ is an independent set.

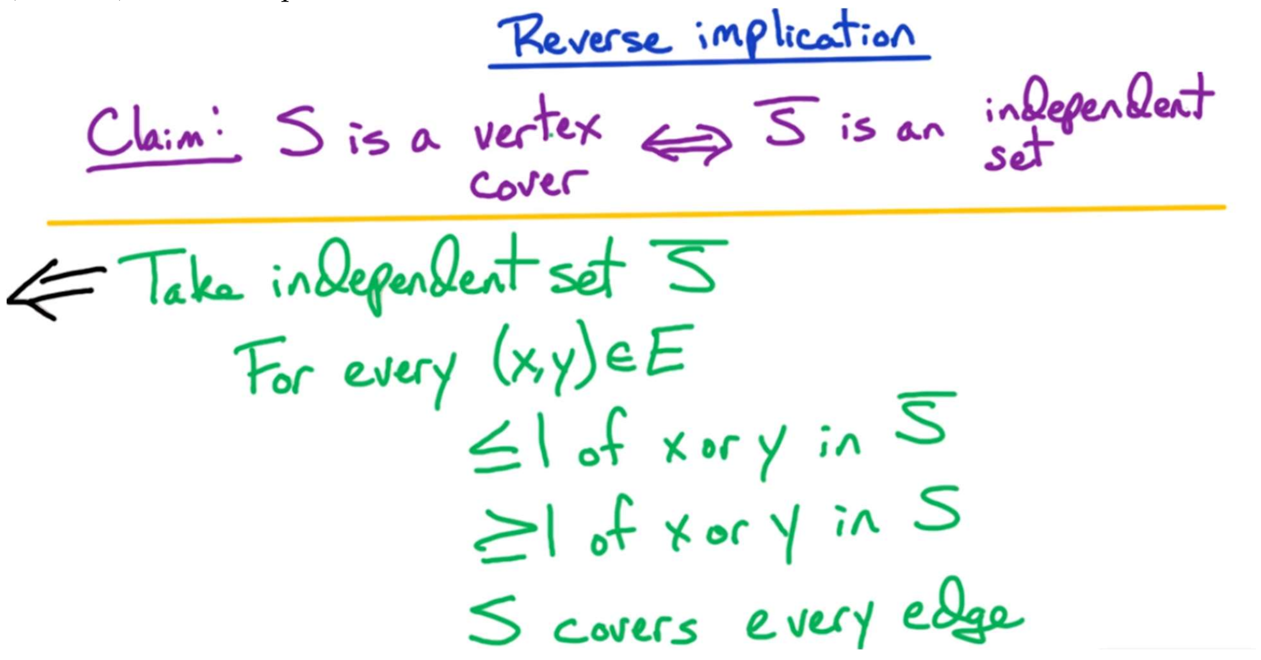
So let's consider this example, the red vertices are in vertex cover S. Now consider an edge of the graph. For instance, this is a particular edge. Notice that at least one of the endpoints is red. Similarly for this edge, at least one of the endpoints is red. In this case both endpoints are red and in general we know for every edge (x,y), at least one of the endpoints is in the set S, because the vertex cover covers every edge. And therefore, at most one of x or y is in S̄, the complement of S. If at least one is in S then at most one is in S̄.

This means that no edge of the graph is contained in S̄. S̄ contains at most one of the endpoints for every edge. So, it doesn't contain both endpoints for any edge. Therefore, S̄ is an independent set.

So we've shown that if S is a vertex cover, then S̄ is an independent set.

***

(Slide 25) Reverse Implication



Reverse implication

Claim: S is a vertex cover $\Longleftrightarrow$ $\bar{S}$ is an independent set

$\Longleftarrow$ Take independent set $\bar{S}$

For every $(x,y) \in E$

≤ 1 of x or y in $\bar{S}$

≥ 1 of x or y in S

S covers every edge

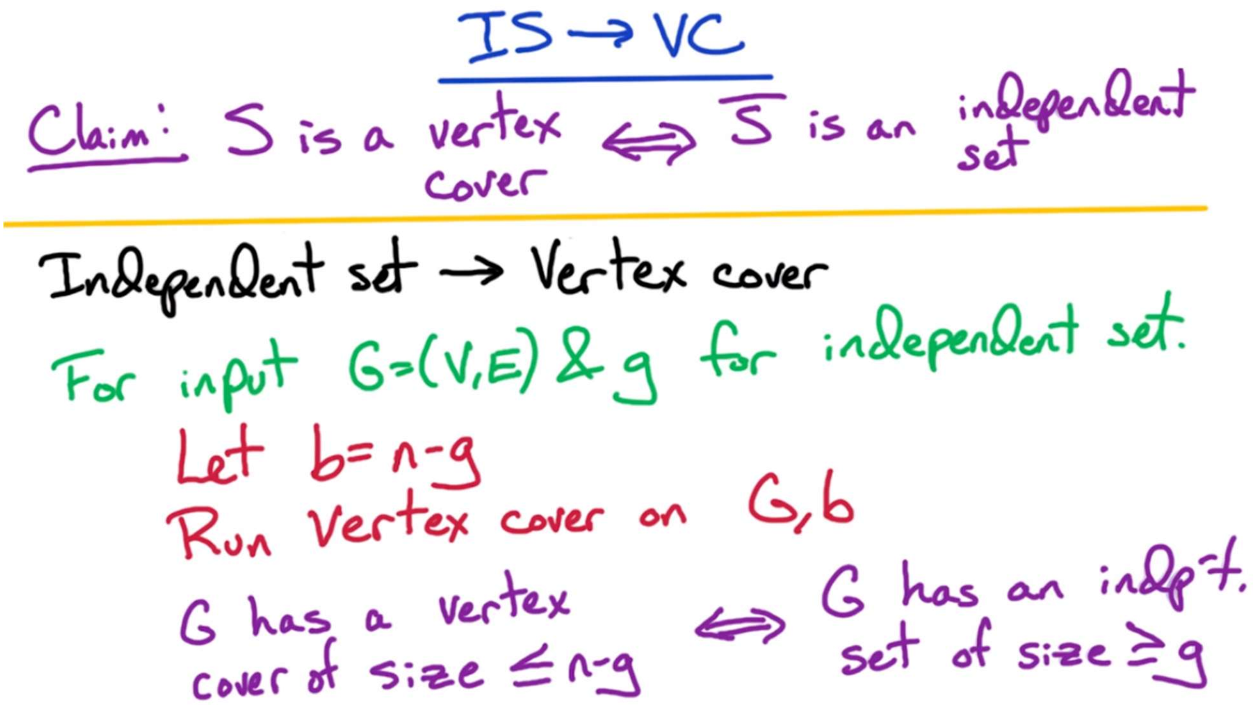
S̄ is an independent set => S is a vertex cover:

> Now, let's prove the reverse implication. Let's take an independent set, S̄, and let's prove that S is a vertex cover.
>
> S̄, we're assuming, is an independent set. So what do we know? We know that no edge is fully contained in this set, S̄. Therefore, we know that at most one of the endpoints, x or y, is in the set S̄. If both endpoints, x and y are in S̄, then that means the edge is fully contained in the set S̄ and therefore, S̄ is not an independent set. But since it's an independent set, at most one of the endpoints or neither of the endpoints is in the set S̄. If at most one is in S̄, then at least one is in S. Thus, S covers every edge of the graph, because for every edge of the graph, at least one of the endpoints is in S. And, therefore, S is a vertex cover. This proves this direction, that if S̄ is an independent set, then S is a vertex cover.

And that completes the proof of the claim. Now, we can do the reduction.

***

(Slide 26) IS VC



IS → VC

Claim: S is a vertex cover $\iff$ $\bar{S}$ is an independent set

Independent set → Vertex cover

For input $G=(V,E)$ & g for independent set:

Let $b=n-g$

Run Vertex cover on G,b

G has a vertex cover of size $\le n-g$ $\iff$ G has an indp't. set of size $\ge g$

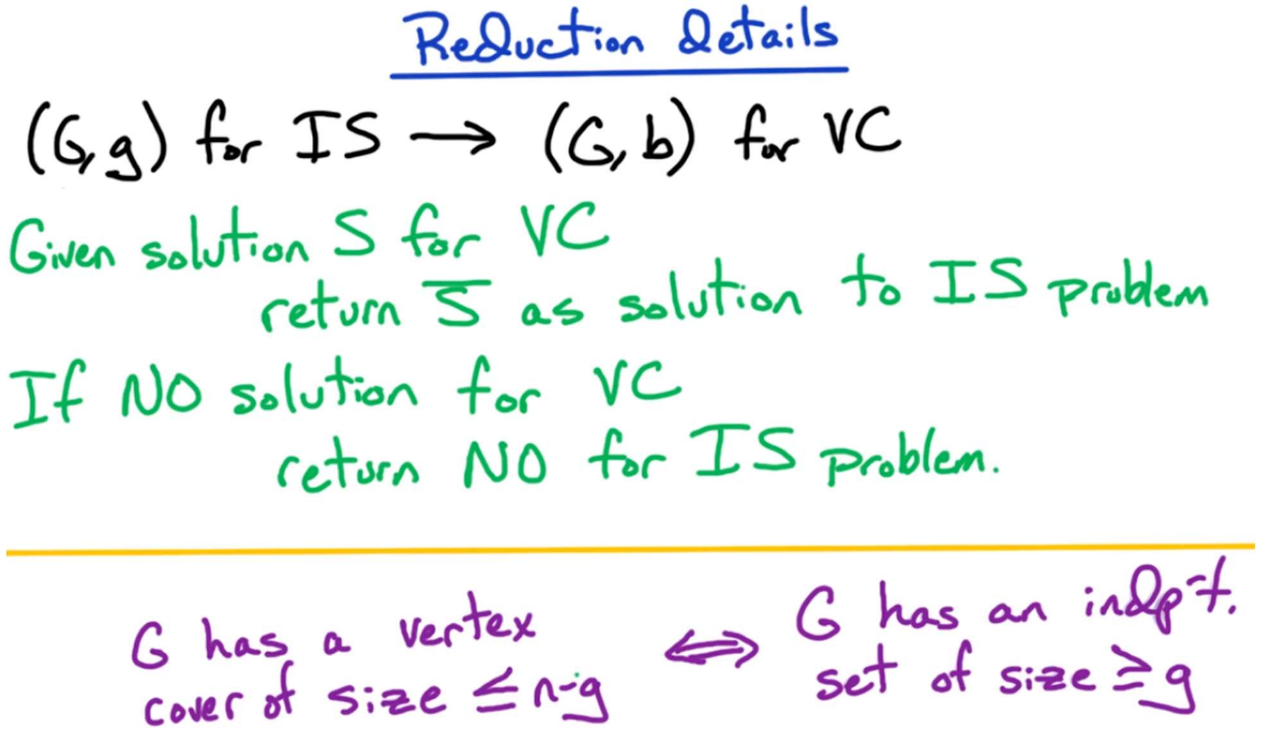
Let's detail the reduction from Independent set problem to the Vertex cover problem.

So, we're trying to find an independent set of size at least little g in the input graph capital G. Now we have to define our input for the vertex cover problem. We let b=n-g. And then we run the Vertex cover problem on the input graph G, the same graph as before, with budget b. Now, that defines the reduction.

Now let's look at the correctness of this reduction. What do we know from the claim? What we know that G has a vertex cover of size at most n-g which is b. So, we have a vertex cover S of this size and we know that in G, $\bar{S}$ is an independent set and this independent set is going to be of size at least little g. So, if we find a vertex cover of the requisite size in G, then its complement set is an independent set of the desired size in G. And if there is no vertex cover of the requisite size, then there is no independent set of the desired size.

***

(Slide 27) IS VC Correctness



Reduction Details

(G,g) for IS $\rightarrow$ (G,b) for VC

Given solution S for VC
return $\overline{S}$ as solution to IS problem

If NO solution for VC
return NO for IS problem.

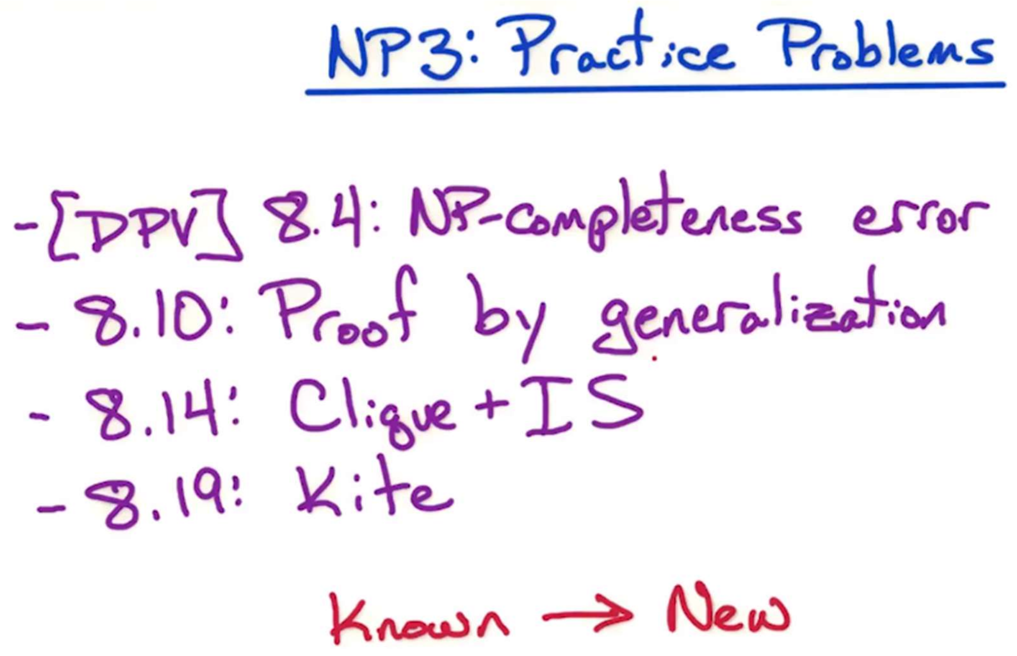
G has a vertex cover of size $\le n-g$ $\Longleftrightarrow$ G has an indep't. set of size $\ge g$

Our reduction took this input instance for the Independent set problem. So we took a graph, a capital G, with a goal, little g, for the Independent set problem, and we reduced it to the Vertex cover problem with input instance, capital G, the same graph, and budget, little b. And little b was defined as n-g. Now, given a solution S for this vertex cover instance – so, this is a vertex cover of size at most little b – then, we return $\overline{S}$ as the solution to the Independent set problem because we know that if S is a vertex cover of size at most little b, then we know that $\overline{S}$ is an independent set of size at least little g. And, if our vertex cover problem comes back with NO, there's no solution, then when we return NO, there's no solution for the Independent set problem because we have this if and only if statement (points at bottom of slide) - if there's no solution here and there's no solution here and vice versa. That completes the reduction and that proves the correctness of the reduction.

That completes the proof that the Vertex cover problem is NP-complete.

***

(Slide 28) Practice Problems



NP3: Practice Problems

- [DPV] 8.4: NP-completeness error
- 8.10: Proof by generalization
- 8.14: Clique + IS
- 8.19: Kite

Known → New

At this point, all of the practice problems at the end of Chapter Eight are relevant. Here are a few of my favorite problems.

- Problem 8.4: In problem 8.4, they give you a proof, a supposed proof of NP-completeness for a problem, and you have to spot the error in that proof.
- Problem 8.10: Now in problem 8.10, there are seven subparts, and you have to prove that each of these problems that they present is NP-complete by doing a proof by generalization. You have to show it's a generalization of a known NP-complete problem.
- Problem 8.14: Some other nice problems are 8.14 where you do the Clique plus Independent set problem. So you're checking whether graph has a large clique and a large independent set.
- Problem 8.19: And 8.19, you consider the Kite problem.

Now, in all these NP-completeness problems, there's two parts: (1) you must first show that the problem is in the class NP, that's usually trivial. And then (2) you have to take a known NP-complete problem and reduce it to this new problem.

Now, what's the known NP-complete problem you should use? Well, it's probably something similar. So, for instance, 8.14 talks about Clique and Independent set, probably you should use Clique or Independent set here as the known problem. If you're talking about a SAT variant, then you should probably use SAT or 3SAT over here.

Now, there are roughly two flavors of NP-completeness reductions:

1. The first is proof by generalization. You can show that this new problem is more general than a known problem. Now, in some sense, we're always showing that the new problem is a generalization of the known problem. But what we mean by this proof by generalization is that you can just set the parameters here so that you get this known problem here.

2. Now, the other type of reduction is you have to do a gadget. We did something like this for the 3-SAT proof. What we do is we take the formula as input over here and we modify it in some way or we take a graph problem over here and input the graph problem here and we modify that graph by adding some small structure to that graph. We call that a gadget that we're adding in.

So, those are the two main approaches. You're going to do proof by gadget. You're going to take the input here and you're going to modify it by adding some gadget to make an input for the new NP-complete problem, or you are going to do a proof by generalization.

This topic is a little like dynamic programming. You have to do a lot of practice problems to get the hang of it. So, good luck and try a lot of practice problems from the text.