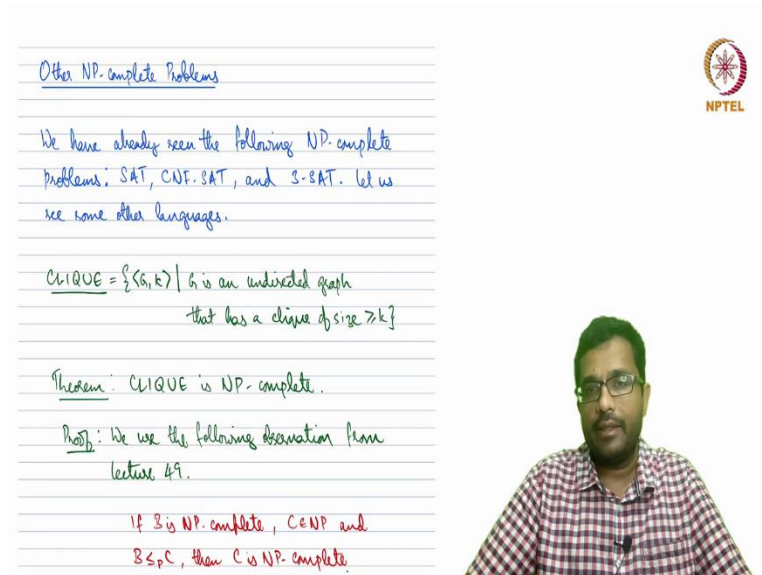


Theory of Computation
Professor Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Lecture 52

CLIQUE and VERTEX-COVER is NP-Complete

(Refer Slide Time: 00:16)



Other NP-complete Problems

We have already seen the following NP-complete problems: SAT, CNF-SAT, and 3-SAT. Let us see some other languages.

$$\text{CLIQUE} = \{ \langle G, k \rangle \mid G \text{ is an undirected graph that has a clique of size } \geq k \}$$

Theorem: CLIQUE is NP-complete.

Proof: We use the following observation from lecture 49.

If B is NP-complete, $C \leq_P B$ and $B \leq_P C$, then C is NP-complete.


Hello, and welcome to lecture 52 of the course Theory of Computation. This is also the first lecture in week 11 of the course. In week 10, we saw the notion of NP completeness, we saw Cook–Levin theorem, which showed that SAT was NP complete. We also saw some other languages being NP complete, which was CNF SAT and 3 SAT. We also saw properties of NP completeness and some other properties.

In this week, week 11, we are mostly going to be seeing other NP complete problems. So, we will we will make use of what we have already learned. And using those properties, we will see other NP complete problems. Hopefully the idea is, you understand different types of NP complete problems like how so many different types of problems can be NP complete. And also, different ways in which we can build gadgets in order to show NP completeness reductions.

(Refer Slide Time: 01:31)

no more even languages.

$CLIQUE = \{ \langle G, k \rangle \mid G \text{ is an undirected graph that has a clique of size } \geq k \}$



Theorem: CLIQUE is NP-complete.

Proof: We use the following observation from lecture 49.

If B is NP-complete, $C \in NP$ and $B \leq_p C$, then C is NP-complete.

Setting 3-SAT = B and CLIQUE = C , it is enough to show the following:

- (1) CLIQUE $\in NP$, and
- (2) 3-SAT \leq_p CLIQUE.



Other NP-complete Problems

We have already seen the following NP-complete problems: SAT, CNF-SAT, and 3-SAT. Let us see some other languages.

$CLIQUE = \{ \langle G, k \rangle \mid G \text{ is an undirected graph that has a clique of size } \geq k \}$

Theorem: CLIQUE is NP-complete.

Proof: We use the following observation from lecture 49.

If B is NP-complete, $C \in NP$ and $B \leq_p C$, then C is NP-complete.



So, the first problem that we'll see is CLIQUE. In fact, all the ingredients to show that CLIQUE is NP complete, we have already covered. So, it is just about stating them and putting them together and seeing that it is NP complete. So, CLIQUE is the problem where we are given 2 things 1 is a graph G and a number K . And this is a yes instance, if the graph has a CLIQUE of size K or bigger.

This is NP complete. So, CLIQUE is a subgraph. Let us say K is 4 means there are 4 vertices, such that they are adjacent in all possible ways. So, there are 6 possible ways in which 4 vertices can be adjacent to each other and all these possibilities exist. If there is a 5 CLIQUE, there is a group of

5 vertices which are adjacent in all possible ways. So again, my drawing is terrible, and it is messy, but you get the idea. So, this is NP complete. So, how are you going to show that this NP complete?

So, we are going to make use of the following observation that we saw from the previous week. So, to show that SAT is NP complete, we had to show that all languages in NP are reducible to SAT. However, from now on, we are not going to be taking that route because we already have one NP complete language.

In fact, we already have SAT, CNF SAT and 3 SAT all of them are NP complete. So, what we are going to be using is this result like we saw that. Suppose B is NP complete, C is in NP, and B is reducible to C in polynomial time, then all of these 3 together imply that C is NP complete.

The way we are going to do this is take an existing or taking known NP complete language B and let C be the language that we want to show is NP complete. So, then all that we have to do is choose a known NP complete language B, then show that B reduces to C where C is the language that we want to show is NP complete.

And then show that C is an NP or we could change the order we could show that C is in NP and show that B is reducible to C for some NP complete language B. So, the things that we have to do is. To show that CLIQUE is NP complete. We have to show that CLIQUE is NP CLIQUE is an NP and then show that an existing known NP complete language reduces to CLIQUE. So here we choose 3 SAT. So, we will show that CLIQUE is an NP and 3 SAT reduces to CLIQUE.

(Refer Slide Time: 04:11)

to show the following:

- (1) $CLIQUE \in NP$, and
- (2) $3\text{-SAT} \leq_p CLIQUE$.

In lecture 46, Theorem 7.32, we have already seen that $3\text{-SAT} \leq_p CLIQUE$. All that remains is to show $CLIQUE \in NP$. We have already seen the below decider in lecture 50.

NFM decider for CLIQUE.

For $i=1$ to n

Non-deterministically select 1 unit vertex i .

Each selected vertex is written on the tape.

If the number of selected vertices $\neq k$, reject.

seen that $3\text{-SAT} \leq_p CLIQUE$. All that remains is to show $CLIQUE \in NP$. We have already seen the below decider in lecture 50.

NFM decider for CLIQUE.

For $i=1$ to n

Non-deterministically select 1 unit vertex i . $\left. \begin{array}{l} \text{Non-deterministically select 1 unit vertex } i. \\ \text{Each selected vertex is written on the tape.} \end{array} \right\} O(n)$

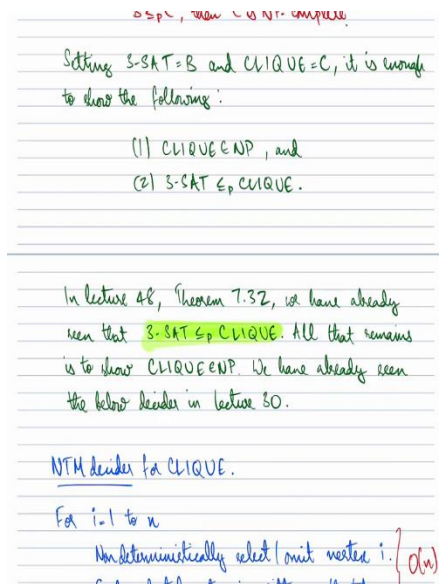
Each selected vertex is written on the tape.

$O(n^2)$ $\left\{ \begin{array}{l} \text{If the number of selected vertices } \neq k, \text{ reject.} \\ \text{Else, for each selected pair } (i, j), \text{ check if } A_{ij} = 1. \\ \text{If } A_{ij} = 0 \text{ for any such pair, reject.} \\ \text{Accept if not rejected.} \end{array} \right.$

In any computation path, this takes at most

$O(n^2) = O(k^2)$ time. \leftarrow $CLIQUE \in NP$





In fact, both of these we have already done in lecture 48, again in the previous week. We showed that 3 SAT reduces to CLIQUE (full reduction). So, when we saw NP complete polynomial time reductions. So, this shows that 3 SAT is in CLIQUE. So now all that remains is to show that CLIQUE is in NP.

In fact, even this we saw in lecture 30. We showed CLIQUE has a non-deterministic decider. We guess a subset of vertices (non-deterministically select a set). For each vertex, we non-deterministically decide to select or skip it.

And at the end you have a subset of vertices. And now you just check does a set have size K. If set has size K, we continue if it does not have size K we reject it. This is first step. So, we are choosing a set non deterministically if it does not have the desired size we reject.

If it has a desired size we check whether it is a K CLIQUE or it is a CLIQUE like we already checked whether it has size K, we check whether it is K CLIQUE we for each pair of vertices check whether there is an edge. So, here A_{ij} is 1, A here is the adjacency matrix .

So, for any such ij , A_{ij} is 0 or in other words the there is no edge between i and j then we reject. If all of them are adjacent we accept. So, this we had seen then and it is clear that this is non deterministic process. So, the time taken to non-deterministically select or omit each vertex is the number of vertices n and then once we have that, then for each pair we check whether there is an edge.

So, that takes K^2 time, so K^2 is at most n^2 . Hence, this whole process take at most $O(n)^2$ time. So, this takes order n time, maybe I will mark that here, this takes order n time and this takes order K^2 time. And so, the whole time the time complexity of the center process is $O(n)^2$, which is polynomial.

Hence, so, we had already seen that this decider for CLIQUE, so I am not going through the characters, but this shows that the running time is also polynomial. So, this is the correctness we have already seen in the running time being polynomial is very clear. And hence, this is a non-deterministic polynomial time decider, hence CLIQUE is an NP.

And earlier in the last week, we showed that 3 SAT reduces to CLIQUE together these 2 imply that CLIQUE is NP complete. So, that is 1 language 1 more language that we know now to be NP complete. Now, if we want to show a new language is NP complete, we can reduce from CLIQUE. We can add CLIQUE to the arsenal of languages that we have and so, that we can reduce from CLIQUE.

(Refer Slide Time: 08:23)

Vertex Cover: Given a graph $G=(V,E)$, a subset $U \subseteq V$ is a vertex cover if $\forall e \in E$, there exists an end point of e in U .

VERTEX-COVER = $\{ \langle G, k \rangle \mid G \text{ is an undirected graph that has a vertex cover of size } k \}$

The next thing is what is covered, what is vertex cover? So, in an undirected graph a vertex cover is a subset of vertices. So, here the vertex cover is a subset U of vertices such that all the edges there one of the endpoints is in U . So, if you take this graph here in the left side, the circle vertices form a vertex cover.

Because you take this edge there is endpoint in this, this edge both endpoints are there, this edge 1 endpoint, this edge 1 endpoint, this edge 1 endpoint, this edge 1 endpoint and this edge also 1 endpoint. So, usually we are interested in the smallest vertex cover and in the largest CLIQUE. So, vertex cover means it is a covering problem we want to cover as many as possible, but we want to reduce the number of vertices used in covering.

So, this is a vertex cover now, we can add 1 more vertex also that is also a vertex cover. So, we can add any vertex because already this covering. So, 1 more vertex will continue to cover. So, that is the vertex cover. And here and the right side we have a complete bipartite graph, 3 vertex on the left side and 4 vertices on the right side. So, any half we can take or the right side or left side and that forms a vertex cover. So, these 3 vertices for a vertex cover. These 4 vertices also would form a vertex cover or form a vertex cover but then I just mark this because this is smaller in number.

Looking at the graph, 2 vertices wouldn't suffice. Covering the left triangle alone requires 2, leaving the other edge exposed. We need a third vertex, so $(G, 2)$ is a no instance.

Vertex Cover deals with pairs (G, K) where G is a graph with a vertex cover of size K . Similar to accepting $(G, 3)$ for graphs with a size-3 cover (which implies a size-4 cover as well), we reject $(G, 2)$.

This highlights another NP-complete problem, distinct from those based on Boolean formulas (like 3-SAT) or clique existence in graphs. Vertex Cover shows how different problem types can be NP-complete.

(Refer Slide Time: 12:19)

Theorem 7.44: VERTEX-COVER is NP-complete.

Proof: As per above approach that we used for CLIQUE, we need to show the following two:

(1) VERTEX-COVER \in NP

(2) 3-SAT \leq_p VERTEX-COVER.

For (1), we can use a 'guess & verify' approach.

We guess k vertices and check if all the edges are "covered" by these vertices.

We focus on (2), for the rest of the proof.

Given a 3-CNF formula ϕ , we will produce

$\langle G, k \rangle$ such that

$\phi \in 3\text{-SAT} \Leftrightarrow \langle G, k \rangle \in \text{VERTEX-COVER}$



So, we'll follow a similar approach to CLIQUE. We'll first show Vertex Cover is in NP, then show a known NP-complete language reduces to it. The known language will be 3-SAT (from Theorem 7.44).

The first part is easy: Vertex Cover is in NP. The hard part, as usual, is showing an NP-complete language reduces to it. Proving C is NP-complete usually involves a reduction from a known NP-complete language. Here, showing Vertex Cover is in NP is simpler.

Similar to CLIQUE, we can use a guess-and-verify approach. We guess a subset of K vertices and check if they form a vertex cover (cover all edges). This straightforward algorithm is clearly non-deterministic and can be verified in polynomial time, making Vertex Cover in NP. We'll skip further details on this part.

What we'll spend time on is the reduction from 3-SAT, which is interesting. Given a 3-SAT instance ϕ (ϕ), we want to build a vertex cover instance (G, K) such that ϕ is satisfiable if and only if G has a vertex cover of size K . (This captures the satisfiability equivalence) The process of constructing (G, K) from ϕ must be done in polynomial time. (This ensures the reduction's efficiency)

In other words, we want to create a graph G and a size K such that ϕ being satisfiable is equivalent to having a vertex cover of size K in G . Additionally, this construction process from ϕ to (G, K) needs to be achievable in polynomial time.

equations. It effectively explains the concept of the reduction from 3-SAT to Vertex Cover, highlighting the two key points. Equivalence: A satisfiable 3-SAT formula (ϕ) corresponds to a graph (G) having a vertex cover of size K . Conversely, if G has a K -sized vertex cover, then ϕ is satisfiable. Polynomial Time: The process of constructing the graph (G) and size (K) from the 3-SAT formula (ϕ) should be achievable in polynomial time with respect to the size of ϕ .

Overall, the passage effectively conveys the essence of the reduction without requiring further modifications.

(Refer Slide Time: 15:05)

Let ϕ have n variables and m clauses.
We will demonstrate through an example.

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

The three triangles form the clause gadgets, while the two "dumb bells" form variable gadgets.



We consider a 3-SAT instance ϕ (ϕ) with n variables and m clauses. Each clause in ϕ is a disjunction (OR) of 3 literals.

To illustrate the reduction, we'll use a simple example. While this example may have repeated variables and literals, the overall procedure is generalizable. We can construct a corresponding graph for any 3-CNF formula following these steps.

The construction of the graph G from the 3-SAT formula ϕ (phi) involves two types of vertices: clause gadgets and variable gadgets. Let ϕ have n variables (X_1, X_2, \dots, X_n) and m clauses (C_1, C_2, \dots, C_m) .

Clause Gadgets: Each clause C_i in ϕ is represented by a corresponding clause gadget G_i in G . This gadget is a triangle with three vertices, denoted as $\{v_i^1, v_i^2, v_i^3\}$. Each vertex is labeled with a literal from the clause. For example, if $C_i = (X_1 \text{ OR } \neg X_2 \text{ OR } X_3)$, then G_i might have vertices labeled $\{v_i^1: X_1, v_i^2: \neg X_2, v_i^3: X_3\}$.

Variable Gadgets: Each variable X_j in ϕ has a corresponding variable gadget V_j in G . This gadget consists of two vertices: $\{u_j, \neg u_j\}$, representing the variable itself and its complement ($\neg X_j$). These two vertices are connected by an edge in G .

Edges within Gadgets: All three vertices within a clause gadget are connected by edges, forming a complete triangle. This can be represented mathematically as for each clause gadget G_i : $E(G_i) = \{\{v_i^1, v_i^2\}, \{v_i^1, v_i^3\}, \{v_i^2, v_i^3\}\}$.

Edges between Gadgets: The crucial connections between clauses and variables are established through edges between gadgets. For each clause C_i and its corresponding gadget G_i , we connect each literal vertex v_i^j in G_i to the corresponding variable vertex (or its complement) in the variable gadget. If the literal in C_i is X_k , we connect v_i^j to u_k . If the literal is $\neg X_k$, we connect v_i^j to $\neg u_k$. This ensures that a variable can only "satisfy" a clause if the corresponding literal vertex in the clause gadget is connected to the appropriate variable vertex in the variable gadget (either the variable itself or its negation).

(Refer Slide Time: 18:46)

while the two "dummy bells" form variable gadgets.

Clause gadget: Three vertices that form a clique - one corresponding to each literal of that clause.

Variable gadget: Two adjacent vertices, corresponding to x_i and \bar{x}_i .

Edges: In addition to all the edges described above, we add edges between the vertices in the clause and variable gadget that have the same label.

Φ has n variables and m clauses.

Clause gadget: Three vertices that form a clique - one corresponding to each literal of that clause.

Variable gadget: Two adjacent vertices, corresponding to x_i and \bar{x}_i .

Edges: In addition to all the edges described above, we add edges between the vertices in the clause and variable gadget that have the same label.

Φ has n variables and m clauses $\Rightarrow G$ has $3m+2n$ vertices
(and $4m+2n$ edges)



corresponding to x_i and \bar{x}_i .

Edges: In addition to all the edges described above, we add edges between the vertices in the clause and variable gadget that have the same label.

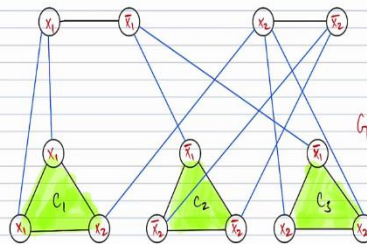
ϕ has n variables and m clauses $\Rightarrow G$ has $3m+2n$ vertices (and $6m+2n$ edges)
CHECK!

We set $k = 2m+n$.

The construction takes $O(m+n)$ time. Now we need to show the following:

Let ϕ have n variables and m clauses. We will demonstrate through an example.

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_2)$$



The three triangles from the clause gadgets, while the two "dumb bells" form variable gadgets.

So, this explaining whatever I just said 3 vertices form a CLIQUE and 2 adjacent vertices the variable gadget and these are the new edges we add edges between the vertices in the clause and the variable gadget that have the same label. And as I said earlier, if ϕ has n variables and m clauses, the variables being X_1 to X_n and clauses being C_1 to C_m , then G has $3m$ plus $2n$ vertices $3m$ plus $2n$ because each clause gadget contains 3 vertices each. So, m clauses gives $3m$ vertices and each variable gadget contains 2 vertices sorry 2 vertices, so, $2n$ vertices in all.

So, we get $3m$ plus $2n$. So, total number of vertices is $3m$ plus $2n$, total number of edges is also not very different we have each clause gadget contains 3 edges. Each variable gadget contains 1 edge so, $3m$ edges.

Because each clause contains 3 edges, so, $3m$ then n edges 1 for each variable gadget so $3m$ plus n and then again $3m$ edges going across. So, $6m$ plus n I think maybe I will see check, so, that you can check and verify this. So, this is the graph so graph is polynomial size $3m$ plus $2n$, $6m$ plus n the procedure is fairly simple like given the formula the process for constructing the graph is simple. So, hence, the construction of the instance is also in polynomial time. So, you can write a for loop or something for the clause gadget for the variable gadget and for the edges going across. So, the we can construct the graph in polynomial time.

(Refer Slide Time: 21:17)

(and $6m+n$ edges)
CHECK!

We set $k = 2m+n$.

The construction takes $O(m+n)$ time. Now we need to show the following:

Φ is satisfiable $\Leftrightarrow G$ has a vertex cover of size $2m+n$.

(\Rightarrow) Φ is satisfiable \Rightarrow

Select a satisfying assignment. Choose the true literal from each clause gadget, and add them to S .



We will demonstrate through an example.

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

$x_1 = F$
 $x_2 = T$

The three triangles from the clause gadgets, while the two "dumb bells" from variable gadgets.

Clause gadget: Three vertices that form a



The next step is to define the value of K in the vertex cover instance (G, K) . Here, K is set to $2m + n$, which is less than the total number of vertices in G ($3m + 2n$). As mentioned earlier, the construction process can be achieved in polynomial time ($O(m + n)$) using loops to iterate through clauses and variables.

Now we need to show the critical correspondence between the satisfiability of the 3-SAT formula ϕ and the existence of a vertex cover of size K ($2m + n$) in G . This works in two directions.

Satisfiable Formula implies Vertex Cover of size K : If ϕ is satisfiable, a truth assignment exists for the variables that makes all clauses true. We can leverage this assignment to construct a vertex cover of size K in G . For example, consider a formula with three clauses. While an initial assignment might not satisfy all clauses, a different approach can work. By strategically assigning truth values (like $X_2 = \text{True}$ and $X_1 = \text{False}$), we can ensure a satisfying assignment. Translating this into a vertex cover, we include all literal vertices connected to true variables and one vertex from each clause gadget. This guarantees that every edge in G is covered, resulting in a vertex cover of size $2m + m$ (less than or equal to K) for any satisfiable formula.

Vertex Cover of size K implies Satisfiable Formula: Conversely, if G has a vertex cover of size K ($2m + n$), we can show that ϕ is satisfiable (though proving this direction is slightly more complex and will be explained separately).

(Refer Slide Time: 23:29)

ψ is satisfiable \leftarrow n was a vertex
 case of huge 2- n .



$(\Rightarrow) \phi$ is satisfiable \Rightarrow

Select a satisfying assignment. Choose the true literal from each variable gadget, and add them to S .

This covers the edges in the variable gadget.

Choose a true literal from each clause gadget and add the other two vertices into S .

This covers the edges in the clause gadget.

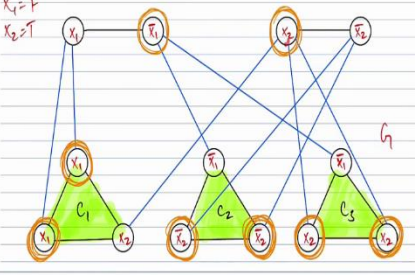



$\psi \in \text{SAT} \Leftrightarrow \text{3-SAT} \Leftrightarrow \text{VERTEX-COVER}$



Let ϕ have n variables and m clauses.
 We will demonstrate through an example.

$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_2)$

$x_1 = F$
 $x_2 = T$



The three triangles from the clause gadgets, while the two "dumb bells" from variable

So, first let us go see the forward direction. If the formula is satisfiable. The graph has a vertex cover of the set size. So, in this case the formula is satisfiable X_1 is false and X_2 is true. Now, since X_1 is so now I will explain how to pick the satisfying assignment, so maybe I will use as a different color to pick them. Maybe I will use this color. So, X_1 is false. So, we pick this X_1 complement from the variable gadget. X_2 is true, so we pick X_2 from the variable gadget, so we pick these 2. And for each clause, we know of 1 at least 1 true literal.

So, in the case of clause 1, we know X_2 is true. Clause 2, we know X_1 complement is true, clause 3 we know both are true, let us say X_1 complement is true. So what we do is we select the vertices

which are not the true which are not the ones that I just mentioned. So, in clause 1 X_2 is true. So we pick the other 2.

So, the what I am building is the vertex cover. So the formula is satisfiable and from the satisfying assignment, I am building the vertex cover and these circle vertices are going to be part of the vertex cover clause 2 X_1 complement is a true literal. So, we select the other 2. Clause 3 in fact, all the literals are true, but let me pick X_1 complement as the true literal and the other 2 are the ones that I pick. So, the claim is that this constitutes a vertex cover. So, let us see why.

So, 1 thing is that for from each clause gadget each clause gadget we have picked 2. So, whatever I am saying is I am saying on the basis of this particular example, but it is general even if you construct any from any formula ϕ if we from any 3 C in a formula 5 if you construct this reduction and we follow these rules, whatever I am saying applies for any such instance So, we have picked 2 vertices from each clause gadget.

So, the claim is that this clause gadget has 3 edges and these 2 vertices any 2 vertices in this triangle covers the edges. So, all the 3 edges here are covered by these 2 vertices. So, all the vertices in this clause gadget is covered and all the vertices in this clause gadget is also covered and all the clause gadgets all the vertices in the all the clause gadgets are covered sorry all the edges in all the clause gadgets are covered by the 2 vertices that we picked from each clause gadget because it the 2 vertices cover this triangle.

Now, all the edges in the variable gadget variable gadget just consists of 2 vertices and 1 edge that is covered by the single vertex that we picked from each variable gadget. So, all the edges in the variable gadgets are also covered. So, what remains to argue is the edges that are going across.

So, for each clause gadgets, there are 3 edges coming out of it once one from each end point one from each vertex of the clause gadget. So, 2 vertices are anyway covered because they are anyway sorry 2 edges coming out of the clause gadget to the variable gadget are anyway covered because should we take C_1 these 2 edges are anyway covered because these X_1 and X_1 are part of the vertex cover.

So, the only thing that remains to be argued is this X_2 or this edge, how is that covered? This edge is covered because this X_2 is covered and in a general setting we have one edge that is to be covered, but how did that edge or how was this vertex chosen or how was this vertex X_2 in the

clause gadget chosen to be not included in the vertex cover. This was chosen to be not included in the vertex cover because it was a true literal. So, we looked for in every clause we looked for 1 literal that is true. So, this was a true literal and hence it was not part of the vertex cover. So, which means, it is adjacent to the X_2 in the variable gadget, which is a true literal and which will be selected.

So, if this X_2 is true, then it would be adjacent to a vertex in the clause gadget sorry, in the variable gadget that is selected as part of the vertex cover, hence, that come that explains how this edge is also going to be covered. Because this X_2 is not part of the vertex cover only because it is a true literal. And because it is a true literal, this endpoint is going to be part of the vertex cover hence these edges are also covered.

So, I have said the same thing here. Choose a 2 literal from each variable gadget and add them to the vertex cover. So, S is the vertex cover the 2 literal from the variable gadgets cover the edges in the variable gadget. Choose a true literal from each clause gadget and add the other 2 vertices into S . And because we are picking 2 vertices in from each clause gadget this cover the edges in the clause gadget

(Refer Slide Time: 30:04)

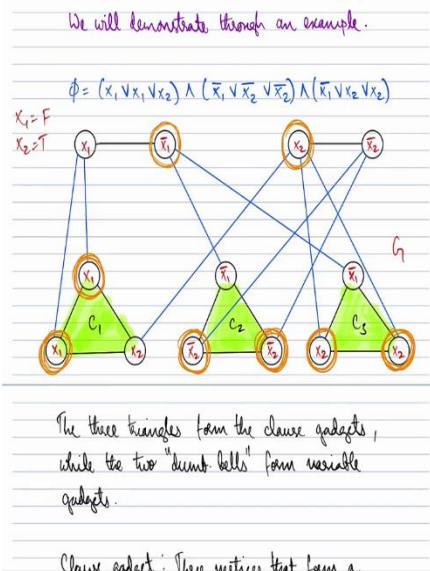
What remains are the edges going across variable and clause gadgets.

For each clause, the vertices from the clause gadgets, and the true literal from the variable gadget cover these.

$\Rightarrow S$ is a vertex cover of size $2m+n$.

(\Leftarrow) Suppose G has a vertex cover of size $2m+n$.





What remains is the edges going across from each clause, the vertices from the clause gadget and the true literal from the variable gadgets cover them. So, if you look at this clause there are 3 edges coming out going to the variable gadget, these 2 are anyway covered what remains is this and this x_1 complement was not picked because it was a true literal, which means the other endpoint must be selected.

So, that covers all the edges. Hence, the selected edges is selected vertices form a vertex cover, and how many vertices did we select? 2 from each clause gadget. So, $2m$ and 1 from each variable gadget so n , so we pick $2m$ plus n vertices and that form a that forms a vertex cover. So, we started with the assumption that is a satisfying assignment and we got a vertex cover of size $2m$ plus n . So this completes 1 direction. Now for the other direction.

(Refer Slide Time: 31:19)

(\Leftarrow) Suppose G has a vertex cover of size $2m+n$.

Let S be the vertex cover.

\Rightarrow To cover the edges in the clause gadget, we need ≥ 2 vertices from each clause gadget.



To cover the edges in the variable gadget, we need ≥ 1 vertex from each variable gadget.



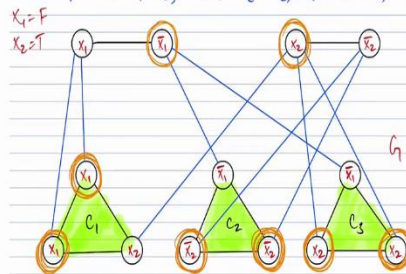
gadget.

$\Rightarrow S$ contains exactly 2 from each clause gadget and 1 from each variable gadget.

Let Φ have n variables and m clauses.

We will demonstrate through an example.

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_2)$$



The three triangles form the clause gadgets, while the two "dumbbells" form variable gadgets.



Suppose the graph has a vertex cover of size $2m$ plus n . Suppose the graph has a vertex cover of size $2m$ plus n . Now we need to show that there is a satisfying assignment. So let us try to see this again. This graph has a vertex cover of size $2m$ plus n . Now notice the structure of this graph. There are these m clause gadgets. Now, if I need to pick it, if I need to cover a triangle, if I need to cover a triangle, something like this maybe I will do it below if I need to cover a triangle. Sorry, if I need to cover a triangle like this, these 3 edges, we need to pick 2 out of these 3 vertices.

So, any clause gadget we need 2 vertices at least in the vertex cover. So any clause gadget we need at least 2 vertices coming out of it and any variable gadget it is something like this. To cover this

edge, we need to pick 1 of these endpoints. So we need 2 vertices from each clause gadget. And we need 1 vertex from each variable gadget at least to form a vertex cover we need 2 vertices from each clause gadget and 1 vertex from each variable gadget. So, this itself tells us that we need minimum $2m$ plus n vertices in a vertex cover.

But here the assumption was that there is a vertex cover of size $2m$ plus n . Now this and now we are saying that every clause gadget has to have at least 2 vertices and every variable gadget has to have at least 1 vertex. Now, the only way we can do this is to pick exactly 2 from each clause gadget and exactly 1 from each variable gadget. Because if we pick more from 1 clause gadget, that means some other clause gadget or variable gadget will not get this number. And hence, there will be something that is not covered in that clause gadget or variable gadget.

(Refer Slide Time: 33:48)

$\Rightarrow S$ contains exactly 2 from each clause gadget and 1 from each variable gadget.

\Rightarrow We set the selected literals from each variable gadget to true. Since S is a vertex cover, it follows that for each clause one of the cross edges must be covered by a vertex from the variable gadget.

Set $x_i = T$

Set $x_j = F$

\Rightarrow This indicates that the chosen assignment is a satisfying assignment.

$\Rightarrow \phi \in 3\text{-SAT}$.

Exercise: Instead of showing $3\text{-SAT} \leq_p \text{VERTEX-COVER}$,



Let ϕ have n variables and m clauses.
 We will demonstrate through an example.

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_2)$$

$x_1 = F$
 $x_2 = T$

The three triangles form the clause gadgets, while the two "dumb-bells" form variable gadgets.



So, this implies that the vertex cover contains exactly 2 vertex vertices from each clause gadget and exactly 1 vertex from each variable gadget, because if you pick if you tried to pick more from some clause gadget, some other gadget is going to suffer because of that and that will hence it will not be a vertex cover. Which means we pick some every cross gadget contributes to and every variable gadget contributes 1. Now, it is kind of the same argument that we saw in the other direction of the proof but in the reverse direction.

Now, how is this going to be a vertex cover? So, now every clause gadget picks 2. So, the edges in the clause gadget are covered and adjacent the variable gadgets are also covered by the 1 vertex. Now, how are we covering the edges going across. So, if you pick a clause 2 edges are covered because 2 vertices are covered, the third vertex is covered because, but, the third vertex is covered sorry third edge is covered because from the other side from the variable gadget because we only have exactly 2 vertices from each clause gadget in the vertex covered.

So, the third edge going across is covered by the from the literals side sorry from the variable gadget which means, if you set all these selected literals to be true, so, X in the case of X_1 suppose the vertex cover X_1 complement then we set X_1 to be false. In the case of X_2 suppose we select X_2 , so, then we set X_2 to be true. So, we look at each variable gadget and see which literally said to which literal is selected for the vertex cover.

So, if X_i is selected to the vertex cover you set X_i to true. If X_i compliment is selected to the vertex cover from the variable gadget we set X_i to be false and this ensures that this assignment that I just

described. So, you look at the selected variables selected literals from each variable gadget. So, maybe I will just write it again here. So, suppose some X_i , X_i compliment here X_i is picked, so, we set X_i to true. Now suppose some others let us say X_j , X_j compliment, then here we set X_j another color just for set X_j to false.

Because yes set X_j to false. So, depending on which 1 we pick, and as I just explained, this will ensure that the selected vertices will form a vertex cover sorry selected vertices will form is a vertex cover so the selected literals the way if we set it this way, this assignment will form a satisfying assignment this is because every so, we need to argue that every clause has a true literal.

So, the way to argue this is that the look at the variable or look at the vertex that is not picked into the vertex cover from the clause. The claim is that this will certainly be true because this is not being covered from the clause vertices it is being this edge is being covered from the other side which means, the other side means this X_2 is connected to X_2 .

So, the other side means X_2 should be picked which means X_2 should be set to true. Hence, each clause is satisfied hence the formula satisfied. So, this gives us that each the assignment that is constructed like this form a satisfying assignment and since ϕ has a satisfying assignment, this implies that this is a yes instance of 3 SAT.

(Refer Slide Time: 38:56)


For (1), we can use a "guess & verify" approach.
 We guess k vertices and check if all the edges are "covered" by these vertices.
 We focus on (2), for the rest of the proof.
 Given a 3-CNF formula ϕ , we will produce $\langle G, k \rangle$ such that


$$\phi \in 3\text{-SAT} \Leftrightarrow \langle G, k \rangle \in \text{VERTEX-COVER}$$

Let ϕ have n variables and m clauses.
 We will demonstrate through an example.

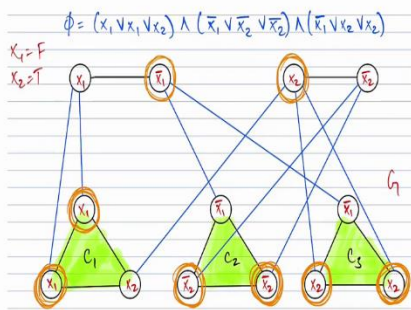
$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_2)$

$x_1 = F$
 $x_2 = T$





Let Φ have n variables and m clauses.
We will demonstrate through an example.



The three triangles form the clause gadgets, while the two "dumb-bells" form variable gadgets.



cover of size k }

Theorem 7.44: VERTEX-COVER is NP-complete.

Proof: As per above approach that we used for CLIQUE, we need to show the following two:

- (1) VERTEX-COVER \in NP
- (2) 3-SAT \leq_p VERTEX-COVER.

For (1), we can use a "guess & verify" approach. We guess k vertices and check if all the edges are "covered" by these vertices.

We focus on (2), for the rest of the proof.

Given a 3-CNF formula Φ , we will produce



problems, NP , NP-SAT , and 3-SAT . Let us see some other languages.

$\text{CLIQUE} = \{ \langle G, k \rangle \mid G \text{ is an undirected graph that has a clique of size } \geq k \}$

Theorem: CLIQUE is NP-complete.

Proof: We use the following observation from lecture 49.

If B is NP-complete, $C \in \text{NP}$ and $B \leq_p C$, then C is NP-complete.

Setting $3\text{-SAT} = B$ and $\text{CLIQUE} = C$, it is enough to show the following:

- (1) $\text{CLIQUE} \in \text{NP}$, and
- (2) $3\text{-SAT} \leq_p \text{CLIQUE}$.



Vertex Cover is first shown to be in NP using a guess-and-verify approach. Then, we reduce 3-SAT to Vertex Cover. Given a 3-SAT formula ϕ with n variables and m clauses, we construct a graph G and size K . The formula is satisfiable if and only if G has a vertex cover of size K ($2m + n$).

This construction involves two types of gadgets. Clause Gadgets (m total): Each clause has a corresponding triangle gadget with 3 vertices labeled with literals from the clause. Variable Gadgets (n total): Each variable has a corresponding gadget with 2 vertices, one for the variable and its complement, connected by an edge.

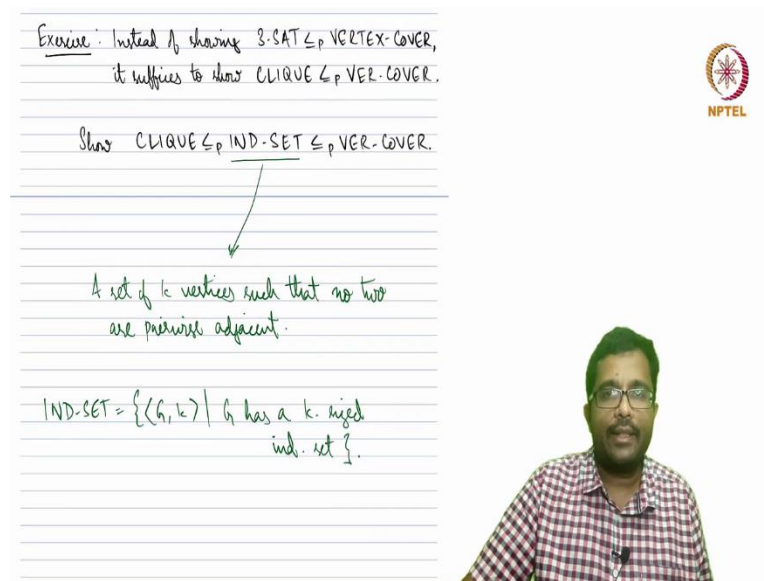
Crucially, we connect clause gadget vertices to corresponding variable gadget vertices based on literal labels (e.g., X_i connects to u_i).

The correspondence between satisfiability and vertex cover size works in two directions.

Satisfiable Formula implies Vertex Cover ($2m + n$): A satisfying assignment for ϕ can be used to construct a vertex cover in G by including literal vertices connected to true variables and one vertex from each clause gadget. This ensures all edges are covered in a vertex cover of size less than or equal to K . Vertex Cover ($2m + n$) implies Satisfiable Formula: If G has a vertex cover of size K ($2m + n$), it can be shown (explained separately) that ϕ has a satisfying truth assignment.

This reduction demonstrates that 3-SAT reduces to Vertex Cover, implying that Vertex Cover is NP-complete (since 3-SAT is already known to be NP-complete). It's important to remember that Vertex Cover can also be reduced from another NP-complete problem like Clique, highlighting the interconnectedness of these problems within the NP-complete class.

(Refer Slide Time: 42:05)



Exercise: Instead of showing $3\text{-SAT} \leq_p \text{VERTEX-COVER}$,
it suffices to show $\text{CLIQUE} \leq_p \text{VER-COVER}$.

Show $\text{CLIQUE} \leq_p \text{IND-SET} \leq_p \text{VER-COVER}$.

A set of k vertices such that no two
are pairwise adjacent.

$\text{IND-SET} = \{ \langle G, k \rangle \mid G \text{ has a } k\text{-sized} \\ \text{ind. set} \}$.

The slide features the NPTEL logo in the top right corner and a video feed of a speaker in the bottom right corner. The speaker is a man with glasses and a beard, wearing a checkered shirt.

The lecture concludes by mentioning an alternative reduction possibility. While the lecture focused on reducing 3-SAT to Vertex Cover, it's also possible to show that Clique reduces to Vertex Cover.

Here's a brief explanation of the alternative reduction chain. Clique to Independent Set: Clique and Independent Set are complementary graph properties. A Clique is a complete subgraph (all vertices connected), while an Independent Set has no edges between any two vertices in the subset. We can construct a reduction from Clique to Independent Set by flipping the graph edges. A graph with a Clique of size K will have an Independent Set of the same size in the flipped graph (and vice versa). Independent Set to Vertex Cover: Independent Set and Vertex Cover are also related. In a Vertex Cover, every edge must be "covered" by a vertex in the cover. An Independent Set, by definition, has no edges, so all vertices in an Independent Set of size K automatically act as a valid Vertex Cover of size K .

Therefore, by demonstrating reductions from Clique to Independent Set and Independent Set to Vertex Cover, we can establish that Clique also reduces to Vertex Cover, reinforcing the NP-completeness of Vertex Cover.

The lecture concludes by highlighting that both Clique and Vertex Cover are NP-complete problems related to graphs, not Boolean formulas. The next lecture will explore another NP-complete problem based on graphs, focusing on finding specific paths within the graph structure.