Notes, 11(a)

ECE 606

Some (Karp) reductions

First problem proved to be **NP**-hard: Circuit-SAT.

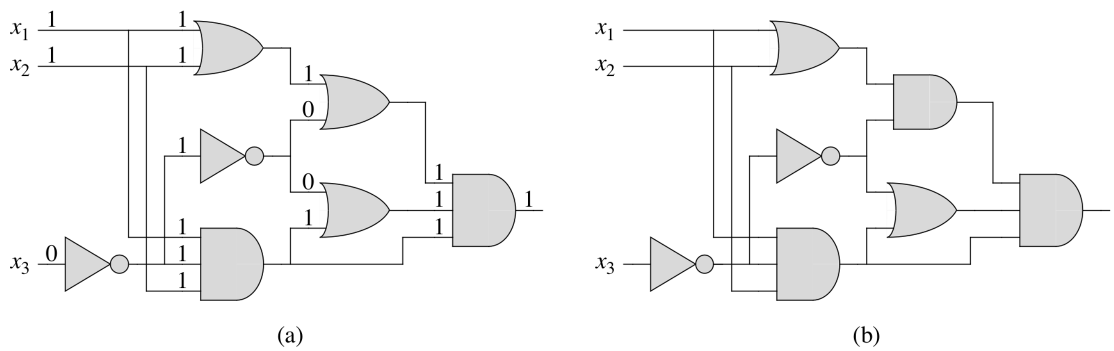


Figure 34.8 Two instances of the circuit-satisfiability problem. **(a)** The assignment $\langle x_1 = 1, x_2 = 1, x_3 = 0\rangle$ to the inputs of this circuit causes the output of the circuit to be 1. The circuit is therefore satisfiable. **(b)** No assignment to the inputs of this circuit can cause the output of the circuit to be 1. The circuit is therefore unsatisfiable.

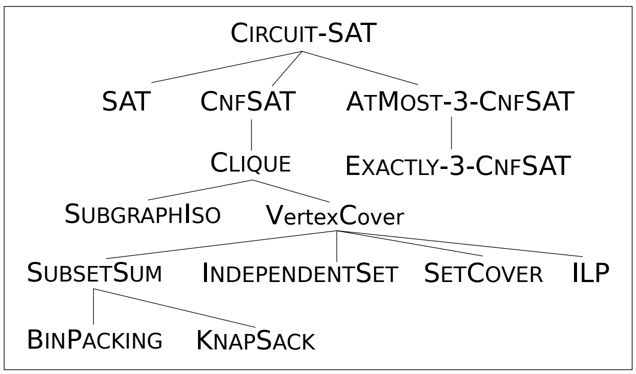
Claim 1. Circuit-SAT *is **NP**-hard.*

Proof: outside the scope of the course.

Idea: given an instance of a problem p in **NP**, encode a search for a certificate/witness for it as an acyclic boolean circuit that returns 1 if and only if such a certificate exists.

Claim 2. *If $p \in \boldsymbol{NP}$-hard and $p \leq_k q$, then $q \in \boldsymbol{NP}$-hard.*

Use CIRCUIT-SAT as an "anchor" problem from which to prove a bunch of other problems to be **NP**-hard. In your textbook:



Before we do that: note that CIRCUIT-SAT is **NP**-complete, i.e., not only **NP**-hard, but also in **NP**. Indeed, all of these problems we consider a in **NP**, which should be easy to prove.

Claim 3. SAT *is* ***NP****-hard.*

Recall what the decision problem SAT is from Lecture (9):

SAT stands for "Boolean satisfiability." Given n propositional variables $p_1, p_2, \ldots, p_n$ and a formula in them with only the operators $\wedge, \vee, \neg$ and parenthesis ( ), does there exist an assignment of true or false to each of $p_1, \ldots, p_n$ that causes the formula to evaluate to true?

For example, the formula

$$((p_1 \wedge \neg p_2) \vee \neg p_1) \wedge p_3 \wedge \neg(p_4 \wedge p_2)$$

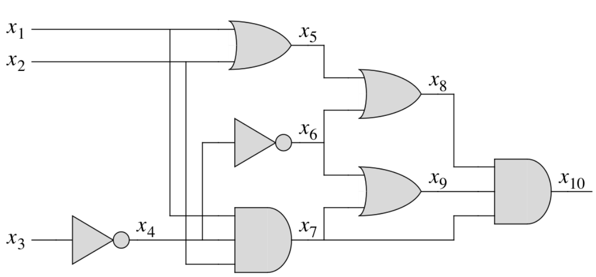
is satisfiable. A satisfying assigment is

$$p_1 = p_4 = 0, p_2 = p_3 = 1$$

The following formula is not satisfiable.

$$((p_1 \wedge \neg p_2) \vee (p_2 \wedge \neg p_1)) \wedge ((\neg p_1 \wedge \neg p_2) \vee (p_1 \wedge p_2))$$

Proof for the claim is from Circuit-SAT $\leq_k$ SAT.



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

From logic problems to graph problems.

Claim 4. CNFSAT *is **NP**-hard.*

Claim 5. CLIQUE *is **NP**-hard.*

Recall what CLIQUE is: given $\langle G, k\rangle$ where G is an undirected graph and k is an integer, does G contain a complete subgraph of k vertices?

Proof in your textbook, which is from CLRS, is by showing that CNFSAT $\leq_k$ CLIQUE.

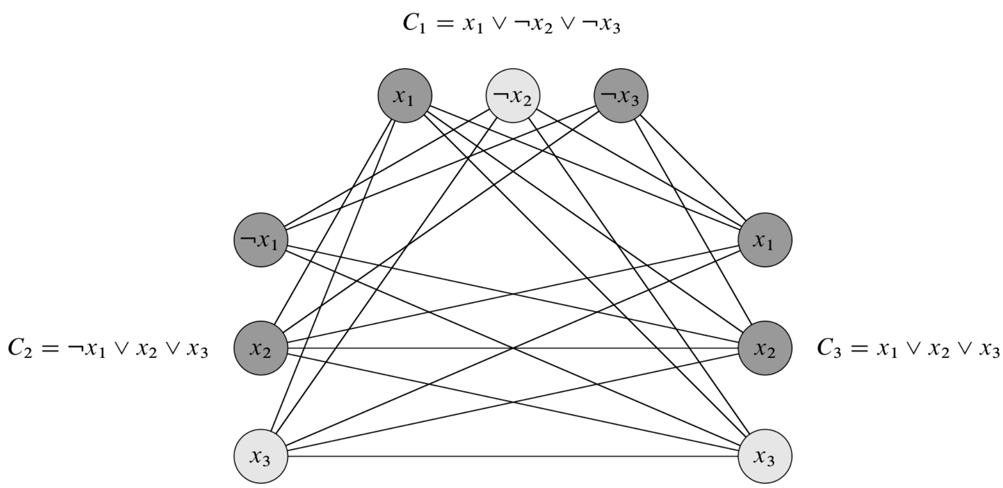


Figure 34.14 The graph G derived from the 3-CNF formula $\phi = C_1 \wedge C_2 \wedge C_3$, where $C_1 = (x_1 \vee \neg x_2 \vee \neg x_3)$, $C_2 = (\neg x_1 \vee x_2 \vee x_3)$, and $C_3 = (x_1 \vee x_2 \vee x_3)$, in reducing 3-CNF-SAT to CLIQUE. A satisfying assignment of the formula has $x_2 = 0$, $x_3 = 1$, and x_1 either 0 or 1. This assignment satisfies C_1 with $\neg x_2$, and it satisfies C_2 and C_3 with x_3, corresponding to the clique with lightly shaded vertices.

VertexCover: given input an undirected graph G and an integer k, does G have a vertex cover of size k?

Define: the complement of an undirected graph $G = \langle V, E \rangle$ is the graph $\overline{G} = \left\langle V, \overline{E} \right\rangle$, where $\overline{E} = \{\langle u, v \rangle \in V^2 \mid \langle u, v \rangle \notin E \wedge u \neq v\}$. That is, $\overline{G}$ has none of the edges that G has, and has all of the edges that G does not.

Claim 6. *$C \subseteq V$ is a clique of $G = \langle V, E \rangle$ if and only if $V \setminus C$ is a vertex cover of $\overline{G} = \left\langle V, \overline{E} \right\rangle$.*

Before we prove the claim, we observe that we can immediately leverage the claim to prove the following claim.

Claim 7. *Clique $\leq_k$ VertexCover.*

Proof. Given an instance $\langle G, k \rangle$ of Clique, the mapping computes and outputs $\left\langle \overline{G}, |V| - k \right\rangle$. $\square$

We now prove the first claim above.

For the "only if": suppose $C \subseteq V$ is a clique in $G = \langle V, E \rangle$. Then, for distinct $u, v \in V$:

$$\begin{aligned} u \in C \wedge v \in C &\iff u \notin V \setminus C \wedge v \notin V \setminus C \\ &\implies \langle u, v \rangle \in E && \because C \text{ is a clique} \\ &\iff \langle u, v \rangle \notin \overline{E} \end{aligned}$$

$$\begin{aligned} \text{So: } u \notin V \setminus C \wedge v \overset{V}{\notin} \setminus C &\implies \langle u, v \rangle \notin \overline{E} \\ \langle u, v \rangle \in \overline{E} &\implies u \in V \setminus C \vee v \in V \setminus C && \because \text{contrapositive} \\ &\implies V \setminus C \text{ is a vertex cover for } \overline{E} \end{aligned}$$

For the "if": see textbook.

Integer Linear Programming (ILP)

Given as input:

- An $m \times n$ integer matrix $\mathbf{A}$, and,
- an $m \times 1$ integer matrix $\mathbf{b}$.

Does there exist an $n \times 1$ bit matrix $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} \leq \mathbf{b}$?

This is equivalent to being given m equations each of the form:

$$a_{i,1}x_1 + a_{i,2}x_2 + \ldots + a_{i,n}x_n \leq b_i$$

where each $a_{i,j}, b_i \in \mathbb{Z}$, and we require each $x_j \in \{0, 1\}$.

Claim 8. $\textsc{VertexCover} \leq_k$ ILP.

We can encode the constraints in $\textsc{VertexCover}$ in the form of ILP.

Given an instance $\langle G, k\rangle$ of $\textsc{VertexCover}$, let $G = \langle V, E\rangle$ where $V = \{1, 2, \ldots, n\}$. In our corresponding ILP instance, we introduce an x_j for every $\overset{j}{\cancel{i}} \in \{1, \ldots, n\}$ with the intent that in a solution to our ILP instance, $x_j = 1$ if vertex j is in the vertex cover, and $x_j = 0$ if it is not.

We adopt the constaints:

$$\begin{aligned} x_a + x_b &\geq 1 && \forall \langle a, b\rangle \in E \\ x_1 + x_2 + \ldots + x_n &\leq k \end{aligned}$$

So resultant ILP instance has $|E| + 1$ equations, with $|V|$ unknowns.