Recall the boolean satisfiability problem: SAT $= \{\langle\phi\rangle \mid \phi$ is a satisfiable Boolean formula$\}$.

A **boolean formula** is a formula of boolean variables using AND, NOT, and OR operators. A formula is satisfiable if there is an assignation of true/false values (a "TVA": a "truth value assignment") to the boolean variables such that the formula evaluates to true.

We have seen so far the following:

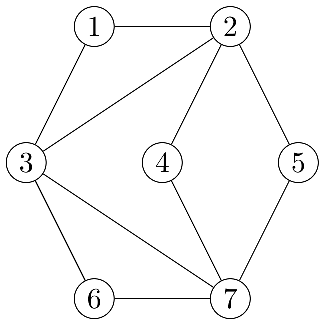
- SAT is in $\mathcal{NP}$. That is:
  - There is a non-deterministic Turing Machine to decide the language in polynomial time
  - There is a deterministic Turing Machine that serves as a verifier
- The Cook-Levin Theorem demonstrated that, for any problem in $\mathcal{NP}$, we can write that as a polynomial-size instance of SAT . Therefore, a polynomial time decider for SAT implies the existence of a polynomial time decider for every problem in $\mathcal{NP}$.

  To some extent, this means that SAT is the "hardest" problem in $\mathcal{NP}$. In fact, any problem that has this property and the first is equally the "hardest" problem in $\mathcal{NP}$. We call these $\mathcal{NP}$-complete.
- Any instance of SAT can be written as 3-SAT, where each *clause* is of three terms. An example of one we saw is $\phi = (x_2 \vee x_3 \vee x_4)(\overline{x_2} \vee x_3 \vee \overline{x_4})(\overline{x_1} \vee x_3 \vee x_5)(\overline{x_1} \vee x_2 \vee x_5)$ $(\overline{x_3} \vee x_4 \vee x_5)(\overline{x_2} \vee \overline{x_4} \vee \overline{x_5})(x_1 \vee \overline{x_2} \vee x_5)(x_3 \vee \overline{x_4} \vee x_5)(x_1 \vee \overline{x_3} \vee \overline{x_5})(\overline{x_2} \vee x_4 \vee \overline{x_5})$ $(x_1 \vee x_2 \vee \overline{x_4})(\overline{x_1} \vee x_2 \vee x_3)(\overline{x_1} \vee \overline{x_2} \vee x_5)(\overline{x_1} \vee x_2 \vee \overline{x_4})$

Independent Set

The INDEPENDENT SET problem is as follows. Given a graph G and an integer k, determine if there is some subset of the vertices V', $|V'| \geq k$ such that no two vertices in V' share an edge.

For example, find an independent set of size $k = 4$ in the following graph:



To prove INDEPENDENT SET is $\mathcal{NP}$-complete, we need to do two things:

- Prove INDEPENDENT SET is in $\mathcal{NP}$
- Show that for any problem in $\mathcal{NP}$, we can write it as a polynomial-size instance of INDEPENDENT SET. The proof that we can do this for SAT was quite involved; just like with the first undecidible problem and subsequent ones, though, there is an easier way via reduction.

Let's prove that INDEPENDENT SET is $\mathcal{NP}$-complete.

- Prove that INDEPENDENT SET is in $\mathcal{NP}$. You may do this either via a NTM polynomial-time decider or via a DTM polynomial-time verifier.

- We want to show that for any problem in $\mathcal{NP}$, we can write it as a polynomial-size instance of INDEPENDENT SET. We already know that this can be done for an instance of 3-SAT – and that's important!

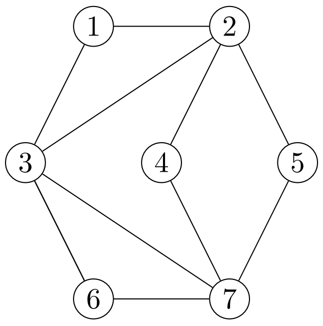
  Suppose we have an instance of the 3-SAT problem and want to encode it as an instance of the INDEPENDENT SET problem.

  – One way to think of 3-SAT is that we need to choose one term from each clause.
  – We must be careful in how we select the terms, so that we never choose x_i in one clause and $\overline{x_i}$ in another clause, for the same value of i.
  – In order to demonstrate that our reduction is correct, we must show that the instance of INDEPENDENT SET created is polynomial in the size of the original 3-SAT instance and that it has an independent set of size k if and only if the original boolean formula has a satisfying assignment.

Vertex Cover

The Vertex Cover problem is as follows. Given a graph G and integer k, determine if there is some subset of the vertices V', $|V'| \le k$ such that each edge is incident to some vertex in V'.

For example, find a vertex cover of size $k = 3$ in the following graph:



Prove that Vertex Cover is $\mathcal{NP}$-complete.

Set Cover

The Set Cover problem is as follows. We are given a series of n sets and a value k, can you select no more than k sets such that every element in the n sets is in at least one element of the k chosen?

For example, can you select three of the following sets in such a way that each letter from 'A' through 'J' is in at least one chosen set?

Set Number	Elements
1	A B
2	A C D E
3	B C F I
4	D G
5	E H
6	I J
7	F G H J

Prove that Set Cover is $\mathcal{NP}$-complete.

3-Color

The 3-Color problem is as follows. Given an undirected graph G, determine whether there is a mapping of vertices in G to 3 distinct colors such that no edge is monochromatic.

Let's prove that 3-Color is $\mathcal{NP}$-complete.

- Prove that 3-Color is in $\mathcal{NP}$.

- We want to show that for any problem in $\mathcal{NP}$, we can write it as a polynomial-size instance of 3-Color. We already know that this can be done for an instance of 3-Sat – and that's important!

  Suppose we have an instance of the 3-Sat problem and want to encode it as an instance of the 3-Color problem.

  - One way to think of 3-Sat is that we must choose a row from the truth table.
  - We must choose a row in such a way that each clause is satisfied (if such a row exists).
  - We must then demonstrate that our instance of the 3-Color problem is of appropriate size and maps correctly to the boolean satisfiability problem.

Directed Hamiltonian Path

A Directed Hamiltonian Path is a simple path that includes every vertex - in a *directed* graph. It turns out this is also $\mathcal{NP}$-complete for undirected graphs and also with cycles that include every vertex. These four variants are all $\mathcal{NP}$-complete.

Let's prove that Directed Hamiltonian Path is $\mathcal{NP}$-complete.

- Prove that Directed Hamiltonian Path is in $\mathcal{NP}$.

- We want to show that for any problem in $\mathcal{NP}$, we can write it as a polynomial-size instance of Directed Hamiltonian Path. We already know that this can be done for an instance of 3-Sat – and that's important!

  Suppose we have an instance of the 3-Sat problem and want to encode it as an instance of the Directed Hamiltonian Path problem.

  – One way to think of 3-Sat is that we must choose a row from the truth table.
  – We must choose a row in such a way that each clause is satisfied (if such a row exists).
  – We must then demonstrate that our instance of the Directed Hamiltonian Path problem is of appropriate size and maps correctly to the boolean satisfiability problem.

Categorizing Complex Problems

Each of the examples in lecture were proven from 3-SAT. Of course, there's no reason you need to do that each time you prove a problem is $\mathcal{NP}$-complete. For example, you saw that VERTEX COVER and SET COVER are $\mathcal{NP}$-complete. Those proofs used INDEPENDENT SET and VERTEX COVER, respectively – and were far easier (for a human to see and write). In general, if you are proving a problem is $\mathcal{NP}$-complete, you likely will want to use a problem of the same general type as what you are proving.

Type	Description	Examples
Packing	You're given a collection of objects and want to choose at least k of them; there are conflicts among subsets of objects (typically pairs).	
Covering	You're given a collection of objects and you want to choose a subset that collectively achieves a certain goal while only selecting k of the objects.	
Partitioning	Divide up a collection of objects into subsets such that each object appears in exactly one of the subsets. Conflicts prevent subsets of objects from going into the same set.	
Sequencing/Permutation	Find a permutation of the n objects. There are restrictions (either absolute or costs) that prevent you from placing certain objects after certain others.	
Numerical		SUBSET SUM KNAPSACK
Constraint Satisfaction		3-SAT

Subset Sum

The Subset Sum problem is as follows. Given a set S of numbers, as well as a number T, determine if there is a subset of S that sums to exactly T.

- Prove that Subset Sum is in $\mathcal{NP}$.

Now let's use a solution to Subset Sum to solve 3-Sat.

As with graph coloring and Hamiltonian Paths, we are going to set up an instance of Subset Sum that first selects a row of the truth table, then we will modify the instance to ensure that each clause is satisfied by the row we choose.

Set up an instance of Subset Sum for a 3-Sat instance with five variables such that any solution to the instance will result in a truth value assignment of variables. How would this look if there were n variables (instead of just five)? What is the corresponding Subset Sum instance?

	x_1	x_2	x_3	x_4	x_5
v_1					
v_1'					
v_2					
v_2'					
v_3					
v_3'					
v_4					
v_4'					
v_5					
v_5'					
T					

Now set up an instance that corresponds to $\phi = (x_2 \vee x_3 \vee x_4)(\overline{x_2} \vee x_3 \vee \overline{x_4})(\overline{x_1} \vee x_3 \vee x_5)$ $(\overline{x_1} \vee x_2 \vee x_5)(\overline{x_3} \vee x_4 \vee x_5)(\overline{x_2} \vee \overline{x_4} \vee \overline{x_5})(x_1 \vee \overline{x_2} \vee x_5)(x_3 \vee \overline{x_4} \vee x_5)$.

Can you generalize this to n variables and k clauses?

	x_1	$\dots$	x_n	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
v_1	1		0								
v_1'	1		0								
v_2	0										
v_2'	0										
v_3	0										
v_3'	0										
v_4	0										
v_4'	0										
v_5	0										
v_5'	0										
s_1	0	0	0								
s_1'	0	0	0								
s_2	0	0	0								
s_2'	0	0	0								
s_3	0	0	0								
s_3'	0	0	0								
$\vdots$											
s_8	0	0	0								
s_8'	0	0	0								
T	1	1	1								