• **DISCLAIMER:** These notes are not necessarily an accurate representation of what I said during the class. They are mostly what I intend to say, and have not been carefully edited.

• Last time:
  1. Examples of reductions: this lecture.
  2. Why is $P$ a set? Problem, $\Pi$, can be viewed as a set containing inputs that return TRUE.

We start by showing that integer programming, which is linear programming where the variables are required to be integers, is NP-complete. It can be described as:

**Definition 0.1.**

\[
\text{INTEGER-PROGRAM} = \{\text{Set of linear constraints that have satisfiable integer solutions}\}
\]

We will show that INTEGER-PROGRAM is an NP complete problem. To do so, we start with the easier direction:

**Lemma 0.2.**

\[
\text{INTEGER-PROGRAM} \in NP.
\]

*Proof.* The verifier can take a set of values for each of the variables, after which the constraints can be checked in linear time. \qed

We now reduce 3-SAT to INTEGER-PROGRAM.

**Lemma 0.3.**

\[
3\text{-SAT} \rightarrow \text{INTEGER-PROGRAM}.
\]

*Proof.* We will create a function that takes any 3-SAT instance, and output a INTEGER-PROGRAM instance that’s true iff the 3-SAT instance is satisfiable.

Given a 3-SAT instance with clauses $c_1 \ldots c_m$ and variables $x_1 \ldots x_n$, we create a integer program on $y_1 \ldots y_n$ such that

\[
0 \leq y_i \leq 1
\]
with 0 representing false, and 1 representing true. For a variable \( y_i \), \( \neg y_i \) is now simply \( 1 - y_i \). Then if a clause can be expressed as the constraint

\[
z_{i1} + z_{i2} + z_{i3} \geq 1,
\]

where \( z_{ij} \) is the \( j \)th literal in the clause, aka either \( y_{ij} \) or \( 1 - y_{ij} \). Requiring these clauses to be simultaneously fulfilled then gives the integer program that corresponds to 3-SAT. \( \Box \)

Combining these two means that INTEGER-PROGRAM is an NP-complete problem. If we write this out in equations, the Cook-Levin theorem gives:

\[
INTEGER - PROGRAM \rightarrow 3 - SAT.
\]

Chaining this together with what we proved above gives

\[
3 - SAT \rightarrow CLIQUE \rightarrow 3 - SAT.
\]

So in the sense of polynomial-time reductions, CLIQUE is equivalent to INTEGER-PROGRAM. This is another way to interpret the class of NP-complete problems.

In this lecture, we exhibit some examples of the large number of NP-complete problems.

We now move onto graphs, and show that maximum clique is NP-complete. The main idea is that the structure of 3-SAT is rich enough for the literals/clauses to be interpreted as (groups) of vertices. This then allows us to convert instances of 3-SAT to instances of graph theoretic problems.

We start with max-clique. The optimization version of maximum clique asks for the maximum subset of vertices such that every pair is connected by an edge. Its decision version can be formalized as follows:

**Definition 0.4.**

\[
CLIQUE = \{(G, k) : \text{graph } G \text{ has a clique of size } k\}
\]

We will show that CLIQUE is an NP complete problem. To do so, we start with the easier direction:

**Lemma 0.5.**

\[
CLIQUE \in NP.
\]

**Proof.** The verifier can take an indicator vector for the set of vertices in the clique, \( y \). Checking whether the size of \( y \) is at least \( k \) can be done in \( O(n) \) time, and checking whether all pairs of vertices in \( y \) are connected can be done in \( O(n^2) \) time. \( \Box \)

We now reduce 3-SAT to CLIQUE.

**Lemma 0.6.**

\[
3 - SAT \rightarrow CLIQUE.
\]
Proof. We will create a function that takes any 3-SAT instance, and output a CLIQUE instance that’s true iff the 3-SAT instance is true. Given a 3-SAT instance with clauses \( c_1 \ldots c_m \) and variables \( x_1 \ldots x_n \), we create a graph with \( 3m \) vertices as follows:

1. For each clause \( C_r = l^r_1 \cup l^r_2 \cup l^r_3 \), create one vertex for each of \( l^r_1 \), \( l^r_2 \) and \( l^r_3 \).
2. Place an edge between two vertices \( l^r_i \) and \( l^s_j \) if and only if:
   - \( r \neq s \), the literals are from different clauses, and
   - \( l^r_i \neq \neg l^s_j \), aka. they are consistent.

Since each the 3 vertices corresponding to each clause are not connected from each other, at most one of them can be in a clique at a time. This means the maximum clique size is at most \( m \). We now show that a clique of size \( m \) exist if and only if the 3-SAT instance is satisfiable.

1. If we have such a clique, then take the vertices involved. Since none of them conflict with each other, we can set the variables accordingly. As we have one such literal per clause, each clause would then have a satisfied literal, making this a satisfying assignment.
2. If we have an satisfying assignment, then each clause has one literal that’s satisfied. Since none of these literals conflict with each other, there are edges between them, giving the clique of size \( m \).

Recall the vertex cover problem asks for the minimum number of vertices such that every edge has at least one endpoint in the set.

Note that this version is optimizing over a parameter, the number of vertices in the cover. So we need to first transform it into a decision version.

Definition 0.7.

\[
\text{Vertex} \rightarrow \text{Cover} = \{(G, k) : \text{graph } G \text{ has a vertex cover of size } k\}
\]

Recall from the last lecture that this decision version is equivalent to the optimization version by binary searching on the parameter \( k \).

To show that \( \text{Vertex} \rightarrow \text{Cover} \in P \), we can let \( y \) for the verification algorithm be the subset of vertices that form the cover. Checking whether \( y \) cover all edges, and \( |y| \leq k \) are both straightforward.

Lemma 0.8. \( \text{Vertex Cover is in NP} \)

For the harder direction, we need the notion of a complement graph.
Definition 0.9. The complement graph of $G = (V, E)$, $\bar{G} = (V, \bar{E})$ is the graph with all edges that are not in $E$.

The following fact shows the close connection between CLIQUE and VERTEX-COVER.

Lemma 0.10. $G$ has a clique of size $k$ if and only if $\bar{G}$ has a vertex cover of size $|V| - k$.

Proof. We will prove the two directions separately, although the two directions of the proof look very similar: they are based on the fact that $S$ is a clique in $G$ if and only if $V \setminus S$ is an independent set in $\bar{G}$.

For the if direction, let $S$ be a vertex cover in $\bar{G}$, then we claim $V \setminus S$ is a clique in $G$. The fact that $S$ is a vertex cover means that there are no edges in $\bar{G}$ between two vertices in $V \setminus S$. Since $\bar{E}$ is the complement of $E$, this means all edges are present between edges in $V \setminus S$ in $G$.

The other direction follows similarly. Let $S$ be a clique in $G$, then we can show that $V \setminus S$ is a vertex cover in $\bar{G}$. The fact that $\bar{E}$ is the complement of $E$ means that there are no edges between vertices in $S$ in $\bar{G}$. Therefore $V \setminus S$ is a vertex cover in $\bar{G}$.

\[\square\]

Corollary 0.11.

$$\text{CLIQUE} \rightarrow \text{VERTEX} \rightarrow \text{COVER}$$

Note that this Lemma also implies $\text{VERTEX} \rightarrow \text{COVER} \rightarrow \text{CLIQUE}$, which gives an alternate proof that $\text{CLIQUE} \in \text{NP}$. However, because the proof of a problem is in NP can usually be done in directly (as in this case), we usually only focus on ONE side of the reduction when using such structural results.