Announcements

- HW solution session on Sunday as usual

New Phrases

- We say a problem is in \( \text{NP} \) if there exists a polynomial time verifier \( TM \) \( V \) and a constant \( k > 0 \) such that for all \( x \in \Sigma^* \):
  
  - if \( x \in L \), then there exists a certificate \( u \) with \( |u| \leq |x|^k \) such that \( V(x, u) \) accepts.
  - if \( x \notin L \), then for all \( u \in \Sigma^* \), \( V(x, u) \) rejects.

- We say there is a polynomial-time many-one reduction from \( A \) to \( B \) if there is a polynomial-time computable function \( f : \Sigma^* \rightarrow \Sigma^* \) such that \( x \in A \) if and only if \( f(x) \in B \). We write this as \( A \leq^P_m B \). (We also refer to these reductions as Karp reductions.)

- A problem \( Y \) is \( \text{NP-hard} \) if for every problem \( X \in \text{NP} \), \( X \leq^P_m Y \).

- A problem is \( \text{NP-complete} \) if it is both in \( \text{NP} \) and \( \text{NP-hard} \).

No Privacy

DOUBLE-CLIQUE: Given a graph \( G = (V, E) \) and a natural number \( k \), does \( G \) contain two vertex-disjoint cliques of size \( k \) each?

Show DOUBLE-CLIQUE is \( \text{NP-complete} \).
This problem is in \textbf{NP}. Here is the description of the poly-time verifier TM.

\texttt{def }\texttt{W}(\texttt{x}, \texttt{u}) : \\
\hspace{1cm} \bullet \text{ if } \texttt{x} \text{ is not an encoding } \langle \texttt{G} = (\texttt{V}, \texttt{E}), \texttt{k} \rangle \text{ of a valid graph } \texttt{G} \text{ and a positive integer } \texttt{k}, \text{ \textbf{REJECT} } \\
\hspace{1cm} \bullet \text{ if } \texttt{u} \text{ is not an encoding of two sets } \texttt{S}, \texttt{T} \subseteq \texttt{V} \text{ of size } \texttt{k} \text{ each, \textbf{REJECT} } \\
\hspace{1cm} \bullet \text{ for each pair of vertices in } \texttt{S}, \text{ if the vertices are not neighbors, \textbf{REJECT} } \\
\hspace{1cm} \bullet \text{ for each pair of vertices in } \texttt{T}, \text{ if the vertices are not neighbors, \textbf{REJECT} } \\
\hspace{1cm} \bullet \text{ for each vertex in } \texttt{S}, \text{ if the vertex is in } \texttt{T}, \text{ \textbf{REJECT} } \\
\hspace{1cm} \bullet \text{ \textbf{ACCEPT} }

Need to show:

(a) If $x \in \text{DOUBLE-CLIQUE}$, there is some proof $u$ of poly-length that makes $W$ accept.

If $x \in \text{DOUBLE-CLIQUE}$, then $x = \langle \texttt{G}, \texttt{k} \rangle$ is a valid encoding and $G$ contains 2 disjoint cliques of size $k$. When $u$ is a valid encoding of a list of the two disjoint cliques, the verifier will accept.

(b) If $x \notin \text{DOUBLE-CLIQUE}$, no matter what $u$ is, $W$ REJECTS.

If $x \notin \text{DOUBLE-CLIQUE}$, then there are 2 options: $x$ is not a valid encoding $\langle \texttt{G}, \texttt{k} \rangle$, or $x$ is valid but $G$ does not contain 2 disjoint cliques of size $k$. In the first case, $W$ rejects. In the second case, $W$ also rejects, as for any encoding $u$, either $u$ is not a valid encoding of two $k$ sized subsets of $V$, these subsets are not disjoint, or they are not valid cliques. Otherwise, if $u$ satisfied these requirements, it would encode 2 disjoint cliques of size $k$ in $G$, which is a contradiction.

(c) $W$ is polynomial-time.

Every step of $W$ is poly-time. Checking if $x$ is a correct encoding of a graph is poly in the size of the graph. Likewise, checking an integer is valid is poly-time. Checking if subsets are of size $k$ is poly in $k$, and checking if they are in $V$ is also poly in the size of $V$ and $k$. Checking if $k$ vertices are neighbors is poly-time. Finally, checking if $k$ vertices are the same as $k$ other ones is poly-time in $k$. So, $W$ is poly-time.

To show that this problem is \textbf{NP-hard}, we will reduce \texttt{CLIQUE} to this problem.

We will first define a map $f : \Sigma^* \rightarrow \Sigma^*$

\texttt{def }\texttt{f}(\texttt{x}) : \\
\hspace{1cm} \text{ if } \texttt{x} \text{ is not an encoding } \langle \texttt{G}, \texttt{k} \rangle \text{ of a graph } \texttt{G} \text{ and int } \texttt{k}: \text{ return } \epsilon \\
\hspace{1cm} \text{ # Create a new graph } \texttt{G'} \text{ that contains two disjoint copies of } \texttt{G} \\
\hspace{1cm} \texttt{V'} = \{v' : v \in \texttt{V}\} \\
\hspace{1cm} \texttt{E'} = \{(u', v') : (u, v) \in \texttt{V} \text{ and } u', v' \in \texttt{V}'\} \\
\hspace{1cm} \text{ return } \langle \texttt{G'} = (\texttt{V} \cup \texttt{V'}, \texttt{E} \cup \texttt{E'}), \texttt{k} \rangle \\

- If $x \in \texttt{CLIQUE}$ then $G$ contains a clique of size $k$. Since $G'$ contains two disjoint copies of $G$ and each of those copies will contain a clique of size $k$, $G'$ contains two disjoint cliques of size $k$. This means that $f(x) \in \text{DOUBLE-CLIQUE}$.

- If $x \notin \texttt{CLIQUE}$ then $x$ is either an invalid encoding in which case $f(x) = \epsilon$ and clearly $\epsilon \notin \text{DOUBLE-CLIQUE}$ or $x$ is a valid encoding and $G$ does not contain a clique of size $k$ in which case the disjoint copies of $G$ in $G'$ will neither contain a cliques of size $k$ so $G'$ does not contain two cliques of size $k$ so $f(x) \notin \text{DOUBLE-CLIQUE}$.

- The function $f$ is polynomial time since verifying the encoding is polynomial time for reasonable encoding schemes and creating $G'$ is also polynomial time.

Since DOUBLE-CLIQUE is both in \textbf{NP} and \textbf{NP-hard}, it is \textbf{NP-complete}.
Edge Cover-Up

Let $G = (V,E)$ be a graph. A vertex covering of $G$ is a set $C \subseteq V$ such that for every edge \{x, y\} \in E, either $x \in C$ or $y \in C$ (a set of vertices such that every edge is incident to at least one vertex in the set). An independent set in $G$ is a set $S \subseteq V$ such that for any $u, v \in S$, \{u, v\} \notin E (a set of vertices such that no edge connects two vertices in the set). Define the following languages:

- **VERTEX-COVER:** $\{(G,k) : G$ is a graph, $k \in \mathbb{N}^+, G$ contain a vertex covering of size $k\}$
- **IND-SET:** $\{(G,k) : G$ is a graph, $k \in \mathbb{N}^+, G$ contains an independent set of size $k\}$

Show that $\text{VERTEX-COVER} \leq_m \text{IND-SET}$ and $\text{IND-SET} \leq_P \text{VERTEX-COVER}$

We will show $\text{VERTEX-COVER} \leq_m \text{IND-SET}$; the other direction is similar (in fact, the same reduction map works).

We will define a map $f: \Sigma^* \rightarrow \Sigma^*$ such that $x \in \text{VERTEX-COVER} \iff f(x) \in \text{IND-SET}$.

```python
def f(x):
    if x is not an encoding $\langle G,k \rangle$ where $G$ is a graph and $k \in \mathbb{N}^+$,
        return $\varepsilon$ (Assuming for our encoding method, $\varepsilon \notin \text{IND-SET}$)
    let $n$ be the number of vertices in $G$
    return $\langle G,n-k \rangle$
```

- If $x \in \text{VERTEX-COVER}$ then $x$ is a valid encoding $\langle G = (V,E),k \rangle$ where $G$ has $n$ vertices. In addition, there exists a vertex cover of size $k$. Let $S$ be the subset of vertices in the vertex cover and consider $V \setminus S$. Observe that $V \setminus S$ is an independent set. To see why, consider any pair of vertices $u, v \in V \setminus S$. Because neither $u$ nor $v$ is in $S$ but $S$ is a vertex cover, we know that $(u,v) \notin E$. In addition, $|V \setminus S| = n-k$, so there exists a independent set of size $n-k$. Hence $f(x) \in \text{IND-SET}$
- If $f(x) \in \text{IND-SET}$, then $G$ must has an independent set of size $n-k$. Let $S$ be the subset of vertices in the independent set. Similar to the previous direction, $V \setminus S$ is a vertex cover. To show this, consider an arbitrary edge $(u,v) \in E$. Because $V \setminus S$ was an independent set, we know one of $u, v \notin S$. Equivilantly, one of $u, v \in V \setminus S$. Thus, all edges are covered by some vertex in $V \setminus S$ so $G$ has a vertex cover of size $n-(n-k) = k$. So $x \in \text{VERTEX-COVER}$.
- $f$ is poly-time since we just need verify that the encoding is valid, count the number of vertices, and then compute $n-k$, which can all be done in poly-time.

What is nondeterministic about NP?

A **nondeterministic TM** (NDTM) is a normal Turing machine where the transition function $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$ is replaced by a transition function of a different type $\delta': Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L,R\})$. This means that if an NDTM is in state $q$ and is reading the letter $\sigma$, there could be any number of choices for the next state, the letter to write, and the direction to move (in this sense, NDTMs are to normal TMs as MFAs are to DFAs). One way to think about how an NDTM computes is that at each step, it makes all the possible choices for the next state/letter/direction and continues simulating the results of all of those choices in parallel. Thus, at any point during its computation, an NDTM has not just one but many active computation paths. An NDTM accepts an input $x$ when any of its computation paths terminate in an accepting state, and it rejects when all of its computation paths terminate in a rejecting state.

Prove that NP is the class of languages decided by some polynomial-time NDTM.
Sketch: Let $S$ be the set of languages decided by some polynomial-time NDTM. ($S \subseteq \text{NP}$) Let $L \in S$ and let $N$ be a poly-time NDTM deciding $L$. Build a verifier that interprets the certificate as the sequence of choices made by $N$ in an accepting computation. The certificate is poly-length because $N$ is poly-time (so its shortest accepting run on any input is poly-length), and the verifier can be poly-time because we need only simulate one particular computation path of $N$ (not all of them).

($\text{NP} \subseteq S$) Given $L \in \text{NP}$ and a verifier $V$ for $L$, build an NDTM $N$ that on input $x$ uses non-determinism to check $V(x, u)$ for all potential certificates $u$ in parallel. There are exponentially many potential certificates, but since we’re checking each one in parallel and each potential certificate is poly-length in $|x|$, $N$ runs in poly-time.

(Extra) Looping Around
Show that the HALTS is NP-hard.

First observe that 3-SAT is decidable. Given an input, we can simply try all $2^n$ possibilities for the variable assignment, then check all the clauses. We’ll present the reduction in a slightly different way: we’ll give a decider $S$ for 3-SAT that returns precisely the result of a black-box call to a decider $H$ for HALTS (make sure you understand why this notion is equivalent). So consider the following machine:

```python
def S(x):
    def HELP(y):
        brute force to check if x is satisfiable
        if x is satisfiable, then halt
        if x is not satisfiable, then loop
        return H(<HELP, "garbage string">)
```

Observe that the input to $H$ is indeed polynomial in the size of the 3-SAT input $x$, as it only requires $x$ to be hard coded once and the remainder of it is constant in size. Further, if it halts, then the expression is satisfiable, and vice versa. Hence, if HALTS can be decided in polynomial time, then 3-SAT can also be decided in polynomial time. Since 3-SAT is NP-complete, it follows that HALTS is NP-hard.

(Bonus) Hard Cut
Define (the decision version of) the MAX-CUT problem as follows:

MAX-CUT: \{\langle G, k \rangle : G’s vertices may be colored with two colors in a way that cuts at least $k$ edges\}.

Prove that MAX-CUT is NP-hard. This is slightly difficult; try reducing from IND-SET.
We'll present a gadget reduction. Given an instance \(\langle G = (V,E), k \rangle\) of \textsc{IND-SET}, we'll output an instance \(\langle G', k' \rangle\) of \textsc{MAX-CUT} such that \(\langle G, k \rangle \in \textsc{IND-SET} \iff \langle G', k' \rangle \in \textsc{MAX-CUT}\). \(G'\) is defined as follows: let \(s\) be a new vertex, and for each \(v \in V\) create a node labeled \(v\) in \(G'\), and draw the edge \(\{v, s\}\). For each edge \(\{u, v\} \in E\), we create the following gadget and insert it in \(G'\): 

\[
\begin{array}{c}
\text{(uv)}_u \\
\text{(uv)}_v \\
\text{(vw)}_v \\
\text{(vw)}_w \\
\end{array}
\]

For example, we transform the following graph as shown (\(G\) on the left, \(G'\) on the right).

\[
\begin{array}{c}
u \\
v \\
w \\
\end{array}
\quad
\begin{array}{c}
n \\
(uv)_u \\
(uv)_v \\
(vw)_v \\
(vw)_w \\
u \\
v \\
w \\
\end{array}
\]

We then output \(\langle G', k + 4E \rangle\). Before proving correctness, first observe that this is constructible in poly-time, since we add one gadget per edge. Also, note that for a given gadget, if neither \(u\) nor \(v\) are colored the same as \(s\), then at most three of the five edges are cut. On the other hand, if at least one of them are colored the same as \(s\), then it is possible to color the intermediary vertices such that four of the five edges are cut. As a corollary, observe that for each gadget, at most four edges can be cut.

\(\Rightarrow\) Now, suppose that there exists an independent set \(S\) of size \(k\) in \(G\). Color all the vertices corresponding to those in \(S\) red, and the vertices corresponding to those in \(V - S\) blue. Also, color \(s\) blue. Since \(S\) is an independent set, for every gadget, at least one of its vertices is blue. Then we may color the intermediary vertices such that four of the five edges in the gadget are cut. The only other edges in \(G'\) are those connecting vertices in \(V\) to \(s\); since \(S\) is of size \(k\), there are \(k\) such cut edges. This coloring therefore achieves \(k + 4E\) cut edges, and so \(\langle G', k + 4E \rangle \in \textsc{MAX-CUT}\), as desired.

\(\Leftarrow\) In the reverse direction, suppose that \(\langle G', k + 4E \rangle \in \textsc{MAX-CUT}\). Since each gadget can only have four of its edges cut, there are at least \(k\) edges cut among the non-gadget edges. Set \(S = \{v : \{v, s\} \text{ is cut}\}\), and write \(S = k + \ell\) for some \(\ell \geq 0\). Then there are at most \(\ell\) edges such that both endpoints are in \(S\); if there were more, then we would not be able to achieve \(k + 4E\) cut edges. For each such edge, delete one endpoint arbitrarily from \(S\). The remaining set is independent and has at least \(k + \ell - \ell = k\) vertices, so we're done.