COMP170 Spring 2017 Recitation 8

Show that the following languages are NP-Complete.

\[
\text{DOUBLE-SAT} = \{\langle \phi \rangle \mid \phi \text{ is a 3cnf formula with two satisfying assignments} \}
\]

\[
\text{DOUBLE-CLIQUE} = \{\langle G, k \rangle \mid G \text{ is a graph with two cliques of size } k \}
\]

We show that each language is in NP, and then reduce each language from a known NP-complete language. For DOUBLE-SAT, we reduce from 3SAT: given any formula, we add an extra clause on the end that includes a new variable and its negation. This ensure that the clause is always satisfied by two distinct assignments, regardless of the rest of the formula. For DOUBLE-CLIQUE, we reduce from CLIQUE: given any graph, we add a separate \(k\)-clique that is disconnected from all vertices in the original graph. This ensure that there is at least one \(k\)-clique, regardless of whether or not there is a \(k\)-clique in the original.

Solution: We show that DOUBLE-SAT is in NP and that 3SAT reduces to DOUBLE-SAT. We then show DOUBLE-CLIQUE is in NP, then show that CLIQUE reduces to DOUBLE-CLIQUE.

- **DOUBLE-SAT ∈ NP**
  
  We specify a polynomial non-deterministic decider for DOUBLE-SAT, where \(C(\phi) = \{c_1, c_2, \ldots, c_k\} = \) the set of clauses in \(\phi\)
  
  \(M\) on \(\langle \phi \rangle\)
  
  1. Non-deterministically choose two distinct assignments, \(A\) and \(A'\), for \(\phi\)
  2. For \(i = 0\) to \(|C(\phi)|\)
     
     if \(A\) does not satisfy \(c_i\), REJECT
     
     if \(A'\) does not satisfy \(c_i\), REJECT
  3. If \(\phi\) is not in 3cnf format, REJECT.
  4. ACCEPT

  Correctness conditions:

  ✓ \(M\) runs in polynomial time.
  
  Step 1 can be computed in \(O(n^2)\) time: for each variable, check previous variables to see if it has been assigned. If it hasn’t, non-deterministically assign it ‘true’ or ‘false’. (Checking that the assignments are distinct requires a \(O(n)\) scan through the formula and checking that at least one variable has a different value on each assignment.) Step 2 can be computed in \(O(n)\) time (assuming each literal has been marked as ‘true’ or ‘false’ in step 1). So \(M\) runs in polynomial time. Step 3 runs in \(O(n)\) time by scanning through \(\phi\) and checking its structure.

  ✓ If \(\langle \phi \rangle \in \) DOUBLE-SAT, then \(M\) accepts \(\langle \phi \rangle\).
  
  Suppose \(\langle \phi \rangle \in \) DOUBLE-SAT. Then \(\phi\) is in 3cnf format and there are two satisfying assignments for \(\phi\). So, on some branch of the computation, these assignments will be chosen. On this branch, \(M\) never rejects in the computation of step 2. So \(M\) accepts \(\phi\).
If \( \langle \phi \rangle \not\in \text{double-sat} \), then \( M \) rejects \( \langle \phi \rangle \).

Suppose \( \langle \phi \rangle \not\in \text{double-sat} \). Then either \( \phi \) is not 3cnf format or there are not two satisfying assignments for \( \phi \). In the first case, \( M \) rejects \( \phi \) at step 3. In the second case, on any branch of computation, there is some iteration of step 2 in which one of the chosen assignments does not satisfy a clause. Therefore, in either case, \( M \) rejects \( \phi \).

Therefore, \( \text{double-sat} \in \text{NP} \).

\[ 3\text{sat} \leq^p \text{double-sat} \]

We define the function \( f(\phi) = \phi \land (v \lor \overline{v} \lor v') = \psi \), where \( v \) and \( v' \) are variables that appear nowhere in \( \phi \).

Correctness conditions:

- \( f \) is computable.
  
  We can construct a TM \( M \) to produce \( \psi \) when given \( \phi \). \( M \) iterates through all possible variables, checking if each one is in \( \phi \). When it finds one that is not, it appends \( \phi \) with the above clause and outputs the resulting formula.

- \( f \) is computable in polynomial time.
  
  The above algorithm performs \( O(n^2) \) checks. For each potential new variable, it must check all of the \( O(n) \) clauses of \( \phi \). Since there are, at most, \( O(n) \) variables in \( \phi \), \( M \) performs at most \( O(n) \) checks, and so runs in \( O(n^2) \) time.

- If \( \phi \in 3\text{sat} \) then \( \psi \in \text{double-sat} \).
  
  Suppose \( \phi \in 3\text{sat} \). Then \( \phi \) is a 3cnf formula with at least one satisfying assignment \( \mathcal{A} \). So \( \mathcal{A} \) satisfies each clause of \( \phi \). So \( \mathcal{A} \) satisfies the first \( n \) clauses of \( \psi \). We define \( \mathcal{A}' \) to be identical \( \mathcal{A} \) except that \( \mathcal{A}' \) also includes the assignment ‘\( v \) is TRUE’. We also define \( \mathcal{A}'' \) to be identical with \( \mathcal{A} \) except that \( \mathcal{A}'' \) includes the assignment ‘\( v \) is FALSE’. Since \( v \) does not appear in \( \phi \), and since the final clause of \( \psi \) is satisfied for any assignment of \( v \), both \( \mathcal{A}' \) and \( \mathcal{A}'' \) satisfy \( \psi \). Also, since \( \mathcal{A}' \) and \( \mathcal{A}'' \) differ in their assignment of \( v \), \( \mathcal{A}' \neq \mathcal{A}'' \). Therefore, there are two satisfying assignments of \( \psi \), so \( \psi \in \text{double-sat} \).

- If \( \phi \not\in 3\text{sat} \) then \( \psi \not\in \text{double-sat} \).
  
  Suppose \( \phi \not\in 3\text{sat} \). Then either \( \phi \) is not in 3cnf format, or it has no satisfying assignments. In the first case, \( \psi \) is also not in 3cnf format. In the second case, there is no satisfying assignment for the first \( n \) clauses of \( \psi \). So, there is no satisfying assignment for all clauses of \( \psi \). In either case, \( \psi \not\in \text{double-sat} \).

Therefore, \( 3\text{sat} \leq^p \text{double-sat} \), so \( \text{double-sat} \) is NP-complete.
• **Double-clique** ∈ NP.

We construct a non-deterministic polynomial time decider. (We assume \( G = (V, E) \).)

\( N \) on \( \langle G, k \rangle \)

1. Non-deterministically choose two distinct sets, \( K \) and \( K' \), of \( k \) vertices in \( G \)

2. For all \((i, j) \in V \times V\)
   - if \( i, j \in K \) and \((i, j) \notin E\), reject
   - if \( i, j \in K' \) and \((i, j) \notin E\), reject

Correctness conditions:

✓ \( N \) runs in polynomial time.

Step 1 runs in \( O(n) \) time. An \( O(n) \) scan is sufficient to mark two sets of \( k \) vertices. An \( O(n) \) scan of the vertices can check that the the selected sets are distinct. Step 2 can be computed in \( O(n^3) \) time. All \( O(n^2) \) pairs of vertices are checked, against the \( O(n) \) vertices of \( K \) and \( K' \), and against the \( O(n) \) edges of \( G \).

✓ If \( \langle G, k \rangle \in \text{double-clique} \) then \( M \) accepts \( \langle G, k \rangle \).

Suppose \( \langle G, k \rangle \in \text{double-clique} \). Then \( G \) contains two \( k \)-cliques. So, these two cliques will be chosen on some branch of the computation. So, at step 2, for every pair of vertices in \( K \) and \( K' \), there is an edge between them in \( E \). Therefore, \( M \) accepts \( \langle G, k \rangle \).

✓ If \( \langle G, k \rangle \notin \text{double-clique} \) then \( M \) rejects \( \langle G, k \rangle \).

Suppose \( \langle G, k \rangle \notin \text{double-clique} \). Then \( G \) does not contain two \( k \)-cliques. So, for any sets \( K \) and \( K' \) of \( k \) vertices in \( G \), one of them will lack an edge between two of its vertices. So, there is some iteration of step 2 where \( i, j \in K \) and \((i, j) \notin E\). Therefore, \( M \) rejects \( \langle G, k \rangle \).

Therefore, \( \text{double-clique} \in \text{NP} \).

• **Clique** \( \leq^p \text{Double-clique} \)

We define the function \( g(\langle G, k \rangle) = \langle G', k \rangle \), where \( G' \) is defined as follows. If \( G = \langle V, E \rangle \), then \( G' = \langle V', E' \rangle \), where

\[
\begin{align*}
V' &= V \cup \{v_1, v_2, \ldots, v_k\} \\
E' &= E \cup \{(v_i, v_j) \mid i, j = 1 \text{ or } 2 \text{ or } \ldots \text{ or } k\}
\end{align*}
\]

and none of the \( v_i \) are elements of \( V \).

Correctness conditions:

✓ \( g \) is computable.

We can define a machine \( N \) that outputs \( \langle G', k \rangle \) when given \( \langle G, k \rangle \). \( N \) finds the highest labelled vertex \( v_n \) of \( G \), then adds vertices \( v_{n+1}, v_{n+2}, \ldots, v_{n+k} \) into the vertex set. After each new vertex has been added, \( N \) adds an edge for every pair of new vertices, then outputs the resulting \( \langle G, k \rangle \).

✓ \( g \) is computable in polynomial time

Finding the highest labelled vertex takes \( O(n) \) time. Adding the up to \( n \) new vertices takes \( O(n^2) \) time (if shifting is required). For each new vertex, there are \( O(n) \) edges that are added, so there are \( O(n^2) \) edges added in the last step. Therefore, \( N \) runs in polynomial time.
✓ If $\langle G, k \rangle \in \text{CLIQUE}$, then $\langle G', k \rangle \in \text{DOUBLE-CLIQUE}$.

Suppose $\langle G, k \rangle \in \text{CLIQUE}$. Then $G$ contains a clique of size $k$. Since $G'$ contains $G$ as a subgraph, $G$ also contains a clique of size $k$ in addition to the size $k$ clique that was added. Therefore, $G'$ contains two cliques of size $k$, so $\langle G', k \rangle \in \text{DOUBLE-CLIQUE}$.

✓ If $\langle G, k \rangle \notin \text{CLIQUE}$, then $\langle G', k \rangle \notin \text{DOUBLE-CLIQUE}$.

Suppose $\langle G, k \rangle \notin \text{CLIQUE}$. Then $G$ does not contain a clique of size $k$. Since $G'$ contains $G$ as a subgraph, and since the size $k$ clique that was added is disconnected from $G$ in $G'$, the only size $k$ clique that $G'$ contains is the one that was added. Therefore, $G'$ does not contain two cliques of size $k$, so $\langle G', k \rangle \notin \text{DOUBLE-CLIQUE}$.

Therefore, $\text{CLIQUE} \leq_{\text{P}} \text{DOUBLE-CLIQUE}$, so DOUBLE-CLIQUE is NP-complete.