COMP 355
Advanced Algorithms
Clique, Vertex Cover, and Dominating Set
Chapter 8 (KT)
Section 34.5 (CLRS)
Recap

Last time we gave a reduction from 3SAT (satisfiability of boolean formulas in 3-CNF form) to IS (independent set in graphs).

Recall that to show that a decision problem (language) \( L \) is NP-complete we need to show:

1. \( L \in \text{NP} \).
2. \( L \) is NP-hard.
Some NP-Complete Problems

**Clique (CLIQUE):** The clique problem is: given an undirected graph \( G = (V,E) \) and an integer \( k \), does \( G \) have a subset \( V' \) of \( k \) vertices such that for each distinct \( u, v \in V' \), \((u, v) \in E\). In other words, does \( G \) have a \( k \) vertex subset whose induced subgraph is complete?

**Vertex Cover (VC):** A vertex cover in an undirected graph \( G = (V,E) \) is a subset of vertices \( V' \subseteq V \) such that every edge in \( G \) has at least one endpoint in \( V' \). The vertex cover problem (VC) is: given an undirected graph \( G \) and an integer \( k \), does \( G \) have a vertex cover of size \( k \)?

**Dominating Set (DS):** A dominating set in a graph \( G = (V,E) \) is a subset of vertices \( V' \) such that every vertex in the graph is either in \( V' \) or is adjacent to some vertex in \( V' \). The dominating set problem (DS) is: given a graph \( G = (V,E) \) and an integer \( k \), does \( G \) have a dominating set of size \( k \)?
INDEPENDENT SET: Given a graph $G = (V, E)$ and an integer $k$, is there a subset of vertices $S \subseteq V$ such that $|S| \geq k$, and for each edge at most one of its endpoints is in $S$?

Ex. Is there an independent set of size $\geq 6$? Yes.
Ex. Is there an independent set of size $\geq 7$? No.
Clique, Independent set, and Vertex Cover

\[ G \]

\[ V' \text{ is a clique of size } k \text{ in } \overline{G} \]

\[ \Leftrightarrow \]

\[ G \]

\[ V' \text{ is a independent set of size } k \text{ in } G \]

\[ \Leftrightarrow \]

\[ G \]

\[ V \setminus V' \text{ is a vertex cover of size } n - k \text{ in } G \]
Lemma: Given an undirected graph $G = (V, E)$ with $n$ vertices and a subset $V' \subseteq V$ of size $k$. The following are equivalent:

i. $V'$ is a clique of size $k$ for the complement, $G$
ii. $V'$ is an independent set of size $k$ for $G$
iii. $V \setminus V'$ is a vertex cover of size $n - k$ for $G$, (where $n = |V|$)

Proof:

(i) $\Rightarrow$ (ii): If $V'$ is a clique for $G$, then for each $u, v \in V'$, $\{u, v\}$ is an edge of $G$ implying that $\{u, v\}$ is not an edge of $G$, implying that $V'$ is an independent set for $G$.

(ii) $\Rightarrow$ (iii): If $V'$ is an independent set for $G$, then for each $u, v \in V'$, $\{u, v\}$ is not an edge of $G$, implying that every edge in $G$ is incident to a vertex in $V \setminus V'$, implying that $V \setminus V'$ is a vertex cover for $G$.

(iii) $\Rightarrow$ (i): If $V \setminus V'$ is a vertex cover for $G$, then for any $u, v \in V'$ there is no edge $\{u, v\}$ in $G$, implying that there is an edge $\{u, v\}$ in $G$, implying that $V'$ is a clique in $G$. 
Theorem: CLIQUE is NP-complete.

**CLIQUE ∈ NP:** We guess the $k$ vertices that will form the clique. We can easily verify in polynomial time that all pairs of vertices in the set are adjacent (e.g., by inspection of $O(k^2)$ entries of the adjacency matrix).

**IS ≤ₚ CLIQUE:** We want to show that given an instance of the IS problem $(G, k)$, we can produce an equivalent instance of the CLIQUE problem in polynomial time. The reduction function $f$ inputs $G$ and $k$, and outputs the pair $(G, k)$. Clearly this can be done in polynomial time. By the above lemma, this instance is equivalent.
Theorem: VC is NP-complete.

**VC ∈ NP:** The certificate consists of the k vertices in the vertex cover. Given such a certificate we can easily verify in polynomial time that every edge is incident to one of these vertices.

**IS ≤_p VC:** We want to show that given an instance of the IS problem \((G, k)\), we can produce an equivalent instance of the VC problem in polynomial time. The reduction function \(f\) inputs \(G\) and \(k\), computes the number of vertices, \(n\), and then outputs \((G, n - k)\). Clearly this can be done in polynomial time.
Dominating Set (Definition)

Problem:

- Dominating-set = Given $<G, k>$, does a dominating set of size (at most) $k$ for $G$ exists?

Let $G=(V,E)$ be an undirected graph.

A dominating set $D$ is a set of vertices that covers all vertices.

- i.e., every vertex of $G$ is either in $D$ or is adjacent to at least one vertex from $D$. 

Dominating Set (Example)

Size-2 example: \{Yellow vertices\}
Dominating Set (Proof Sketch)

Steps:

1) Show that Dominating-set ∈ NP.

2) Show that Dominating-set is not easier than a NPC problem
   • We choose this NPC problem to be Vertex cover
   • Reduction from Vertex-cover to Dominating-set

3) Show the correspondence of “yes” instances between the reduction
It is trivial to see that Dominating-set ∈ NP
- Given a vertex set D of size k, we check whether (V-D) are adjacent to D
- i.e., for each vertex, v, in D, whether v is adjacent to some vertex u in D
Dominating Set - (2) Reduction

Reduction - Graph transformation

• For each edge \((v, w)\) of \(G\), add a vertex \(vw\) and the edges \((v, vw)\) and \((w, vw)\) to \(G'\)

• Furthermore, remove all vertices with no incident edges; such vertices would always have to go in a dominating set but are not needed in a vertex cover of \(G\)
  – We skip the discussion of this subtle part in the followings

\[
G \xrightarrow{T} G' \\
<G, k> \in L \iff <G', k> \in L'
\]

Vertex-cover Dominating-set
A vertex cover, $C$, is a set of vertices that covers all edges

- i.e., each edge is at least adjacent to some node in $C$

\{2, 4\}, \{3, 4\}, \{1, 2, 3\} are vertex covers
Dominating Set: Graph Transformation Example

$G$

$G'$
A dominating set of size $K$ in $G'$ $\iff$ A vertex cover of size $K$ in $G$

Let $D$ be a dominating set of size $K$ in $G'$

- Case 1): $D$ contains only vertices from $G$
  - Then, all new vertices have an edge to a vertex in $D$
  - $D$ covers all edges
  - $D$ is a valid vertex cover of $G$
A dominating set of size $K$ in $G'$ $\iff$ A vertex cover of size $K$ in $G$

Let $D$ be a dominating set of size $K$ in $G'$

- Case 2): $D$ contains some new vertices (vertex in the form of $uv$)
  (We show how to construct a vertex cover using only old vertices, otherwise we cannot obtain a vertex cover for $G$)

For each new vertex $uv$, replace it by $u$ (or $v$)
If $u \in D$, this node is not needed
Then the edge $u-v$ in $G$ will be covered
After new edges are removed, it is a valid vertex cover of $G$ (of size at most $K$)
Dominating Set - (3) Correspondence

A dominating set of size $k$ in $G'$ $\iff$ A vertex cover of size $k$ in $G$

$\Leftarrow$ Let $C$ be a vertex cover of size $k$ in $G$

For an old vertex, $v \in G'$:
  - By the definition of VC, all edges incident to $v$ are covered
  - $v$ is also covered

For a new vertex, $uv \in G'$:
  - Edge $u-v$ must be covered, either $u$ or $v \in C$
  - This node will cover $uv$ in $G'$

Thus, $C$ is a valid dominating for $G'$ (of size at most $k$)
Dominating Set: Graph Transformation Example

$G$

$G'$
Dominating Set - (3)
Correspondence

Vertex-cover in $G$

Dominating-set in $G'$
Vertex Cover to Dominating Set

$\text{VC} \leq_p \text{DS}$

VC: “every edge is incident to a vertex in $V'$”.
DS: “every vertex is either in $V'$ or is adjacent to a vertex in $V'$”.
Translation must somehow map the notion of “incident” to “adjacent”
Correctness of the Reduction

We need to show that $G$ has a vertex cover of size $k$ if and only if $G'$ has a dominating set of size $k'$.

Correctness of the VC to DS reduction (where $k = 3$ and $l = 1$).
NP-Completeness

So far, we have seen:
1. 3-SAT to INDEPENDENT SET (IS)
2. IS to CLIQUE
3. IS to VERTEX COVER
4. VERTEX COVER to DOMINATING SET
5. 3-COLORING to CLIQUE COVER (not the same as CLIQUE)
So your problem is NP-Complete? Now What?

Important: NP-Completeness is not a death sentence, but you need appropriate expectations/strategies

Some Useful Strategies
1. Brute-Force (for small input sizes)
2. Heuristics – Fast algorithms that are not always correct
3. Solve in exponential time but faster than brute-force search
4. Approximation Algorithms