CS 341: ALGORITHMS

Trevor Brown
trevor.brown@uwaterloo.ca
DC 2338, Office hour M3-4pm
THIS TIME

• Intractability (hardness of problems)
  • Polynomial transformations
    • Example: Clique ≤_p Vertex-Cover
DEFINING NP

Recall from last time

• A decision problem $\Pi$ is **solved** by a poly-time $\text{verify}$ alg. if:
  • for every **yes**-instance $I$, **there exists** a certificate $C$ such that $\text{verify}(I, C)$ returns $\text{true}$, and
  • for every **no**-instance $I$, $\text{verify}(I, C)$ returns $\text{false}$ for every $C$

• **The complexity class NP** denotes the set of all decision problems that can be solved by poly-time $\text{verify}$ algorithms
POLYNOMIAL TRANSFORMATION
THE MORE CONVENIENT DEFINITION

• Let $\Pi_1$ and $\Pi_2$ be decision problems
• $\Pi_1 \leq_P \Pi_2$ iff there exists $f : I(\Pi_1) \to I(\Pi_2)$ such that:
  • $f(I)$ is computable in poly-time, for all $I \in I(\Pi_1)$
  • If $I \in J_{yes}(\Pi_1)$ then $f(I) \in J_{yes}(\Pi_2)$
  • If $f(I) \in J_{yes}(\Pi_2)$ then $I \in J_{yes}(\Pi_1)$

Note: this is the same as saying $(I \in J_{yes}(\Pi_1)) \iff (f(I) \in J_{yes}(\Pi_2))$
Clue
Instance: An undirected graph $G = (V, E)$ and an integer $k$, where $1 \leq k \leq |V|$.
Question: Does $G$ contain a clique of size $\geq k$? (A clique is a subset of vertices $W \subseteq V$ such that $uv \in E$ for all $u, v \in W, u \neq v$.)

Problem 7.9
Vertex Cover
Instance: An undirected graph $G = (V, E)$ and an integer $k$, where $1 \leq k \leq |V|$.
Question: Does $G$ contain a vertex cover of size $\leq k$? (A vertex cover is a subset of vertices $W \subseteq V$ such that $\{u, v\} \cap W \neq \emptyset$ for all edges $uv \in E$.)

Every edge must touch a node in $W$.
These $k$ nodes touch every edge in $G$. 
CLIQUE $\leq_P$ VERTEX-COVER

- Suppose $I = (G, k)$ is an instance of Clique where $G = (V, E), V = \{v_1, \ldots, v_n\}$ and $1 \leq k \leq n$

Want to solve $Clique(G, k)$

Idea: there is a $k$-clique in $G$ iff there is an $(n - k)$ Vertex-Cover in $H$

- Construct instance $f(I) = (H, \ell)$ of Vertex-Cover, where $H = (V, F), \ell = n - k$ and $v_i v_j \in F \iff v_i v_j \notin E$

Reduce to $VertexCover(H, \ell)$ where $\ell = n - k$

Every edge of $G$ is a non-edge of $H$. Every non-edge of $G$ is an edge of $H$.
PROVING THIS IS A POLYNOMIAL TRANSFORMATION

• We denote Clique by $CL$ and Vertex-Cover by $VC$
• $CL \leq_P VC$ iff there exists $f : \mathcal{I}(CL) \rightarrow \mathcal{I}(VC)$ such that:
  • $f(I)$ is computable in poly-time, for all $I \in \mathcal{I}(CL)$
  • If $I \in I_{yes}(CL)$ then $f(I) \in I_{yes}(VC)$
  • If $f(I) \in I_{yes}(VC)$ then $I \in I_{yes}(CL)$
Time to compute $f(I)$?
Assuming adjacency matrix, $\text{Size}(I) = \Theta(n^2 + \log_2 k)$

Constructing $H$ takes $\Theta(n^2)$ time, and computing $\ell$ takes $\Theta(\log n)$ time.

So computing $f(I)$ takes $\Theta(n^2)$ time, which is polynomial in $\text{Size}(I)$.

So this takes poly-time

**COMPLEXITY OF THE TRANSFORMATION**

• Suppose $I = (G, k)$ is an instance of Clique where $G = (V, E), V = \{v_1, ..., v_n\}$ and $1 \leq k \leq n$

Want to solve $\text{Clique}(G, k)$

Reduce to $\text{VertexCover}(H, \ell)$ where $\ell = n - k$

• Construct instance $f(I) = (H, \ell)$ of Vertex-Cover, where $H = (V, F), \ell = n - k$ and $v_i v_j \in F \iff v_i v_j \notin E$
Now let's show this
PROVING: \( I \in I_{\text{yes}}(CL) \Rightarrow f(I) \in I_{\text{yes}}(VC) \)

- Suppose \( I = (G, k) \) is a \textbf{yes}-instance of Clique

- Then there is a set \( W \) of \( k \) vertices in a clique (with \textbf{all-to-all} edges)

- Define \( W' = V \setminus W \). Clearly \( |W'| = n - k = \ell \).

- We \textbf{claim} \( W' \) is a vertex cover of \( H \)

- Consider any edge \( (u, v) \in H \)

- If \( \{u, v\} \cap W' \neq \emptyset \) we are done, so assume \( u, v \notin W' \) to obtain a contradiction

- Then \( u, v \in W \), and \( W \) is a clique in \( G \), so \( (u, v) \in G \)

- But \( (u, v) \in H \) implies \( (u, v) \notin G \). Contradiction!
PROVING THIS IS A POLYNOMIAL TRANSFORMATION

- We denote Clique by $CL$ and Vertex-Cover by $VC$
- $CL \leq_P VC$ iff there exists $f : \mathcal{I}(CL) \rightarrow \mathcal{I}(VC)$ such that:
  - $f(I)$ is computable in poly-time, for all $I \in \mathcal{I}(CL)$
  - If $I \in I_{yes}(CL)$ then $f(I) \in I_{yes}(VC)$
  - If $f(I) \in I_{yes}(VC)$ then $I \in I_{yes}(CL)$

Now let’s show this
PROVING: \( f(I) \in \mathcal{I}_{yes}(VC) \Rightarrow I \in \mathcal{I}_{yes}(CL) \)

• Suppose \( f(I) = (H, \ell) \) is a yes-instance of \( VC \)

• Then there is a set of \( \ell = n - k \) vertices \( W' \) that is a vertex cover of \( H \)

• Define \( W = V \setminus W' \). Clearly \( |W| = k \).

• We claim \( W \) is a clique in \( G \)

• Since \( W' \) is a vertex cover of \( H \), every edge in \( H \) has at least one endpoint in \( W' \)

• Therefore, no edge in \( H \) has two endpoints in \( W \)

• So, in \( G \), there are edges between all pairs of nodes in \( W \). So, \( W \) is a clique in \( G \).

So we have \( CL \leq_p VC \)
**Theorem 7.10**

If $\Pi_1$ and $\Pi_2$ are decision problems, $\Pi_1 \leq_P \Pi_2$ and $\Pi_2 \in \mathbf{P}$, then $\Pi_1 \in \mathbf{P}$.

**Proof.**

Suppose $A$ is a poly-time algorithm for $\Pi_2$, having complexity $O(m^\ell)$ on an instance of size $m$. Suppose $f$ is a transformation from $\Pi_1$ to $\Pi_2$ having complexity $O(n^k)$ on an instance of size $n$. We solve $\Pi_1$ as follows:

1. Given $I \in \mathcal{I}(\Pi_1)$, construct $f(I) \in \mathcal{I}(\Pi_2)$.
2. Run $A(f(I))$.

It is clear that this yields the correct answer. We need to show that these two steps can be carried out in polynomial time as a function of $n = \text{Size}(I)$. Step (1) can be executed in time $O(n^k)$ and it yields an instance $f(I)$ having size $m \in O(n^k)$. Step (2) takes time $O(m^\ell)$. Since $m \in O(n^k)$, the time for step (2) is $O(n^{k\ell})$, as is the total time to execute both steps.
Theorem 7.11

Suppose that \( \Pi_1, \Pi_2 \) and \( \Pi_3 \) are decision problems. If \( \Pi_1 \leq_P \Pi_2 \) and \( \Pi_2 \leq_P \Pi_3 \), then \( \Pi_1 \leq_P \Pi_3 \).

Proof.

We have a polynomial transformation \( f \) from \( \Pi_1 \) to \( \Pi_2 \), and another polynomial transformation \( g \) from \( \Pi_2 \) to \( \Pi_3 \). We define \( h = f \circ g \), i.e., \( h(I) = g(f(I)) \) for all instances \( I \) of \( \Pi_1 \). (Exercise: fill in the details.) \( \square \)
COMPLEXITY CLASS **NP-COMPLETE** (NPC)

The complexity class **NPC** denotes the set of all decision problems $\Pi$ that satisfy the following two properties:

$$\Pi \in \text{NP}$$

For all $\Pi' \in \text{NP}$, $\Pi' \leq_P \Pi$.

**NPC** is an abbreviation for **NP-complete**.

Note that the definition does not imply that NP-complete problems exist!
**Theorem 7.12**

If \( P \cap NPC \neq \emptyset \), then \( P = NP \).

**Proof.**

We know that \( P \subseteq NP \), so it suffices to show that \( NP \subseteq P \). Suppose \( \Pi \in P \cap NPC \) and let \( \Pi' \in NP \). We will show that \( \Pi' \in P \).

1. Since \( \Pi' \in NP \) and \( \Pi \in NPC \), it follows that \( \Pi' \leq_P \Pi \) (definition of NP-completeness).
2. Since \( \Pi' \leq_P \Pi \) and \( \Pi \in P \), it follows that \( \Pi' \in P \) (see slide #180).
Satisfiability and the Cook-Levin Theorem

Problem 7.13

CNF-Satisfiability

Instance: A boolean formula $F$ in $n$ boolean variables $x_1, \ldots, x_n$, such that $F$ is the conjunction (logical “and”) of $m$ clauses, where each clause is the disjunction (logical “or”) of literals. (A literal is a boolean variable or its negation.)

Question: Is there a truth assignment such that $F$ evaluates to true?

Theorem 7.14 (Cook-Levin Theorem)

CNF-Satisfiability $\in$ NPC.
Now that we **have one** NPC problem...

Now, given any NP-complete problem, say $\Pi_1$, other problems in NP can be proven to be NP-complete via polynomial transformations from $\Pi_1$, as stated in the following theorem:

**Theorem 7.15**

Suppose that the following conditions are satisfied:

- $\Pi_1 \in \text{NPC}$,
- $\Pi_1 \leq_P \Pi_2$, and
- $\Pi_2 \in \text{NP}$.

Then $\Pi_2 \in \text{NPC}$. 

More Satisfiability Problems

Problem 7.16
3-CNF-Satisfiability
Instance: A boolean formula $F$ in $n$ boolean variables, such that $F$ is the conjunction of $m$ clauses, where each clause is the disjunction of exactly three literals.
Question: Is there a truth assignment such that $F$ evaluates to true?

Problem 7.17
2-CNF-Satisfiability
Instance: A boolean formula $F$ in $n$ boolean variables, such that $F$ is the conjunction of $m$ clauses, where each clause is the disjunction of exactly two literals.
Question: Is there a truth assignment such that $F$ evaluates to true?

3-CNF-Satisfiability $\in$ NPC, while 2-CNF-Satisfiability $\in$ P.
To show that 3-CNF-SAT is NP-complete, we will give a reduction from CNF-SAT.

2-CNF-SAT can be solved in polynomial time. Suppose we are given an instance $I$ of 2-CNF-SAT on a set of boolean variables $X$.

1. For every clause $x \lor y$ (where $x$ and $y$ are literals), construct two directed edges $\overline{xy}$ and $\overline{yx}$. We get a directed graph on vertex set $X \cup \overline{X}$.
2. Determine the strongly connected components of this directed graph.
3. $I$ is a yes-instance if and only if there is no strongly connected component containing $x$ and $\overline{x}$, for any $x \in X$. 
2-SAT EXAMPLES

- \((p \lor q) \land (\neg p \lor r) \land (\neg r \lor \neg p)\)
  - Satisfiable: \(p = 1, q = 1, r \in \{0,1\}\)
- \((p \lor q) \land (\neg p \lor r) \land (\neg r \lor \neg p) \land (p \lor \neg q)\)

Edges (implications of clauses)...

- \(\neg p \Rightarrow q\)
- \(p \Rightarrow r\)
- \(r \Rightarrow \neg p\)
- \(\neg p \Rightarrow \neg q\)
- \(\neg q \Rightarrow p\)
- \(\neg r \Rightarrow \neg p\)
- \(p \Rightarrow \neg r\)
- \(q \Rightarrow p\)

Can solve this in poly-time. So, 2-SAT is “easy.” It turns out 3-SAT is HARD...

\[q \Rightarrow p \Rightarrow \neg r \Rightarrow \neg p \Rightarrow \neg q\]...
so \(q\) cannot be true

\[-q \Rightarrow p \Rightarrow \neg r \Rightarrow \neg p \Rightarrow q\]...
so \(q\) cannot be false

Therefore the formula cannot be satisfied!
**IS N-SAT HARDER THAN 3-SAT? NOT REALLY...**

**CNF-Satisfiability \( \leq_P \) 3-CNF-Satisfiability**

Suppose that \((X, C)\) is an instance of **CNF-SAT**, where \(X = \{x_1, \ldots, x_n\}\) and \(C = \{C_1, \ldots, C_m\}\). For each \(C_j\), do the following:

**case 1** If \(|C_j| = 1\), say \(C_j = \{z\}\), construct four clauses

\[\{z, a, b\}, \{z, a, \overline{b}\}, \{z, \overline{a}, b\}, \{z, \overline{a}, \overline{b}\}\].

**case 2** If \(|C_j| = 2\), say \(C_j = \{z_1, z_2\}\), construct two clauses

\[\{z_1, z_2, c\}, \{z_1, z_2, \overline{c}\}\].

**case 3** If \(|C_j| = 3\), then leave \(C_j\) unchanged.

**case 4** If \(|C_j| \geq 4\), say \(C_j = \{z_1, z_2, \ldots, z_k\}\), then construct \(k - 2\) new clauses

\[\{z_1, z_2, d_1\}, \{\overline{d_1}, z_3, d_2\}, \{\overline{d_2}, z_4, d_3\}, \ldots, \{\overline{d_{k-4}}, z_{k-2}, d_{k-3}\}, \{\overline{d_{k-3}}, z_{k-1}, z_k\}\].
STOPPED HERE!
Correctness of the Transformation

Suppose $I$ is a yes-instance of **CNF-SAT**. We show that $f(I)$ is a yes-instance of **3-CNFSAT**. Fix a truth assignment for $X$ in which every clause contains a true literal. We consider each clause $C_j$ of the instance $I$.

- If $C_j = \{z\}$, then $z$ must be true. The corresponding four clauses in $f(I)$ each contain $z$, so they are all satisfied.
- If $C_j = \{z_1, z_2\}$, then at least one of the $z_1$ or $z_2$ is true. The corresponding two clauses in $f(I)$ each contain $z_1, z_2$, so they are both satisfied.
- If $C_j = \{z_1, z_2, z_3\}$, then $C_j$ occurs unchanged in $f(I)$.
- Suppose $C_j = \{z_1, z_2, z_3, \ldots, z_k\}$ where $k \geq 3$ and suppose $z_t \in C_j$ is a true literal. Define $d_i = \text{true}$ for $1 \leq i \leq t - 2$ and define $d_i = \text{false}$ for $t - 1 \leq i \leq k$. It is straightforward to verify that the $k - 2$ corresponding clauses in $f(I)$ each contain a true literal.
Correctness of the Transformation (cont.)

Conversely, suppose \( f(I) \) is a yes-instance of 3-CNF-SAT. We show that
\( I \) is a yes-instance of CNF-SAT.

(1) Four clauses in \( f(I) \) having the form \( \{z, a, b\}, \{z, a, \overline{b}\}, \{z, \overline{a}, b\} \)
\( \{z, \overline{a}, \overline{b}\} \) are all satisfied if and only if \( z = \text{true} \). Then the corresponding
clause \( \{z\} \) in \( I \) is satisfied.

(2) Two clauses in \( f(I) \) having the form \( \{z_1, z_2, c\}, \{z_1, z_2, \overline{c}\} \) are both satisfied if and only if at least one of \( z_1, z_2 = \text{true} \). Then the
 corresponding clause \( \{z_1, z_2\} \) in \( I \) is satisfied.

(3) If \( C_j = \{z_1, z_2, z_3\} \) is a clause in \( f(I) \), then \( C_j \) occurs unchanged in \( I \).
Correctness of the Transformation

\[ \{z_1, z_2, d_1\}, \{\overline{d_1}, z_3, d_2\}, \{\overline{d_2}, z_4, d_3\}, \ldots,\]
\[\{\overline{d_{k-4}}, z_{k-2}, d_{k-3}\}, \{\overline{d_{k-3}}, z_{k-1}, z_k\}. \]

(4) Finally, consider the \( k - 2 \) clauses in \( f(I) \) arising from a clause
\( C_j = \{z_1, z_2, z_3, \ldots, z_k\} \) in \( I \), where \( k > 3 \). We show that at least one of
\( z_1, z_2, \ldots, z_k = \text{true} \) if all \( k - 2 \) of these clauses contain a true literal.

Assume all of \( z_1, z_2, \ldots, z_k = \text{false} \). In order for the first clause to contain
a true literal, \( d_1 = \text{true} \). Then, in order for the second clause to contain a
true literal, \( d_2 = \text{true} \). This pattern continues, and in order for the second
last clause to contain a true literal, \( d_{k-3} = \text{true} \).

But then the last clause contains no true literal, which is a contradiction.
We have shown that at least one of \( z_1, z_2, \ldots, z_k = \text{true} \), which says that
the clause \( \{z_1, z_2, z_3, \ldots, z_k\} \) contains a true literal, as required.