More Poly-Time Reductions:

1. $3SAT \leq_P \text{DirHamPath}$
2. $3SAT \leq_P \text{IP (not in book)}$
3. Bounded Halting is NPC
4. $3SAT \leq_P \text{3-COL (not in book)}$
5. NP hardness and coNP hardness
6. Decision, search, and optimization problems
7. Coping with NP completeness: Approximation

Sipser, Chapter 7, Sections 7.3, 7.4, 7.5, 10.1, 10.2
Traveling Salesperson

Parameters:

- set of cities $C$
- set of inter-city distances $D$
- goal $k$

(not drawn to scale)
Traveling Salesperson Problem (TSP)

Define $\text{TRAVELING-SALESMAN} = \{ \langle C, D, k \rangle \mid (C, D) \text{ has a TS tour of total distance } \leq k \}$

Remark: Can consider two versions – undirected and directed.
Traveling Salesperson Problem (TSP)

Define $\text{TRAVELING-SALESMAN} = \{\langle C, D, k \rangle \mid (C, D) \text{ has a TS tour of total distance } \leq k \}$

Remark: Can consider two versions – undirected and directed.

Recall
\[ \text{HAMCIRCUIT} = \{\langle G \rangle \mid G \text{ has Hamiltonian circuit} \} \]

Theorem: Directed HAMCIRCUIT is polynomial-time reducible to directed TRAVELING-SALESMAN,

\[ \text{HAMCIRCUIT} \leq_P \text{TRAVELING-SALESMAN} \]
The reduction: Given a directed graph $G = (V, E)$ we construct a directed traveling salesman instance.

- The cities are identical to the nodes of the original graph, $C = V$.
- The distance of going from $v_1$ to $v_2$ is $1$ if $(v_1, v_2) \in E$, and $2$ otherwise.
- The bound on the total distance of a tour is $k = |V|$. 
HAMCIRCUIT $\leq_P$ TSP

Validity of Reduction

$\implies$ Suppose $G$ has a Hamiltonian circuit. The distance assigned by the reduction to all edges in this circuit is 1. Thus in $(C, D)$ there is a traveling salesman tour of total distance $|V| = k$, namely $(C, D, k) \in \text{TRAVELING-SALESMAN}$.

$\impliedby$ Suppose $(C, D)$ has a traveling salesman tour of total distance $|V| = k$. Tour cannot contain any edge of distance 2. Therefore it gives a Hamiltonian circuit in $G$. 
Validity of Reduction

\[ \implies \text{Suppose } G \text{ has a Hamiltonian circuit. The distance assigned by the reduction to all edges in this circuit is } 1. \text{ Thus in } (C, D) \text{ there is a traveling salesman tour of total distance } |V| = k, \text{ namely } (C, D, k) \in \text{TRAVELING-SALESMAN}. \]

\[ \iff \text{Suppose } (C, D) \text{ has a traveling salesman tour of total distance } |V| = k. \text{ Tour cannot contain any edge of distance } 2. \text{ Therefore it gives a Hamiltonian circuit in } G. \]

Efficiency: Reduction in quadratic time (filling up distances for all edges of the complete graph).
The Language SAT (reminder)

**Definition:** A Boolean formula is in **conjunctive normal form** (CNF) if it consists of **clauses**, connected with $\land$'s. Each **clause** is a **disjunction** ($\lor$) of **literals**.

For example $(x_1 \lor \overline{x_2} \lor x_3 \lor x_4) \land (x_3 \lor \overline{x_5} \lor x_6) \land (x_3 \lor \overline{x_6})$

**Definition:** SAT = $\{\langle \phi \rangle \mid \phi$ is satisfiable CNF formula$\}$
3SAT (reminder)

**Definition:** A Boolean formula is in **3CNF form** if it is a **CNF** formula, and each clause has at most **three literals**.

\[
(x_1 \lor \overline{x}_2 \lor \overline{x}_3) \land (x_3 \lor \overline{x}_5 \lor x_6) \land (x_3 \lor \overline{x}_6 \lor x_4)
\]

Define

\[
3SAT = \{\langle \phi \rangle \mid \phi \text{ is satisfiable 3CNF formula}\}
\]

Clearly, if \(\phi\) is a satisfiable 3CNF formula, then for any satisfying assignment of \(\phi\), every clause must contain at least one literal assigned 1.
NPC – Reminder

A language $\mathcal{B}$ is **NP-complete** if it satisfies
A language $\mathcal{B}$ is $\text{NP}$-complete if it satisfies

- $\mathcal{B} \in \text{NP}$, and
A language $B$ is NP-complete if it satisfies

- $B \in NP$, and
- Every $A$ in NP is polynomial time reducible to $B$. 
A language $C$ is coNP-complete if it satisfies

- $C \in coNP$ (namely its complement is in $NP$), and
- For every $D$ in coNP, $D \leq_P C$. 

coNP-Completeness (analog notion)
Cook-Levin (early 70s)

Theorem: \( \text{SAT is NP complete} \).

Prf.: Membership \( (\text{SAT} \in \text{NP}) \) is easy, using satisfying assignment as certificate.

- Hard part is showing that every NP problem reduces to SAT in poly-time.
- Idea: Suppose \( L \in \text{NP} \), and \( M \) is an NTM that accepts \( L \).
- On input \( w \) of length \( n \), \( M \) runs in time \( t(n) = n^c \).
- We consider the \( n^c \)-by-\( n^c \) tableau that describes the computation of \( M \) on input \( w \).
### The Tableau

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>...</th>
<th>t(n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**cell[1,1]**

**cell[1,t(n)]**

![Tableau Diagram]

0 0 1
0
Saw a Few Reductions So Far

- \( \text{SAT} \leq_P \text{3SAT} \) (\( \Rightarrow \) \text{3SAT} is NP-complete)
- \( \text{3SAT} \leq_P \text{Clique} \) (\( \Rightarrow \) \text{Clique} is NP-complete)
- In recitation: \( \text{Clique} \leq_P \text{Independent Set} \) (\( \Rightarrow \) \text{IS} is NP-complete)
- In recitation: \( \text{Clique} \leq_P \text{Vertex Cover} \) (\( \Rightarrow \) \text{VC} is NP-complete)
- \( \text{HamPath} \leq_P \text{HamCircuit} \)
- \( \text{HamCircuit} \leq_P \text{TSP} \)

Will now show \( \text{3SAT} \leq_P \text{DirHamPath} \), thus establishing NP-completeness of DirHamPath, DirHamCircuit, and DirTSP.
Directed Hamiltonian Path

For any 3CNF formula $\phi$, we construct a directed graph $G$ with vertices $s$ and $t$ such that $\phi$ is satisfiable iff there is a directed Hamiltonian path from $s$ to $t$. 
Directed Hamiltonian Path

Here is a 3CNF formula $\phi$:

$$(a_1 \lor b_1 \lor c_1) \land (a_2 \lor b_2 \lor c_2) \land \cdots (a_k \lor b_k \lor c_k) \land$$

where

- each $a_i, b_i, c_i$ is $x_i$ or $\overline{x_i}$
- the $\ell$ clauses are $C_1, \ldots, C_\ell$
- the $k$ variables are $x_1, \ldots, x_k$. 
DirHamPath: NP Completeness Proof

Turn to a separate pdf presentation:

Integer Programming (IP)

Definition: A linear inequality has the form

\[ a_1 x_1 + a_2 x_2 + \ldots + a_n x_n \leq b \]

where \( a_1, \ldots, a_n, b \) are real numbers, and \( x_1, \ldots, x_n \) are real variables.

The Integer Programming (IP) problem:

**Input:** A set of \( m \) linear inequalities with integer coefficients \((a_i, b)\) in \( n \) variables \( x_1, x_2, \ldots, x_n \).

The language \textbf{IP} is the collection of all systems of linear inequalities that have a solution where all \( x_i \) are integers.
Consider the following system of linear inequalities

\[
\begin{align*}
y & \leq 2x & \text{green line} \\
-2x + 1 & \leq y & \text{red line} \\
4x - 2 & \leq y & \text{purple line} \\
0 & \leq x & \leq 1 \\
0 & \leq y & \leq 2
\end{align*}
\]
This set does have a unique solution: the right hand corner of the solid triangle, \((1, 2)\).

But if we change the constraint on \(y\) to \(0 \leq y \leq 1\), then we’d have no solution with integer coordinates, even though there are many solutions with rational, or real, coordinates.

Will now show IP is NP complete.
Membership in NP easy (why?)
\[ \text{SAT} = \{ \langle \varphi \rangle \mid \varphi \text{ is a satisfiable CNF formula} \} \]

For example, the following formula is in SAT:
\[ (x_1 \lor \overline{x}_2 \lor \overline{x}_3 \lor x_4) \land (x_3 \lor \overline{x}_5 \lor x_6) \land (x_3 \lor \overline{x}_6) \]

Let \( \varphi \) be a CNF formula with \( m \) clauses and \( n \) variables \( x_1, \ldots, x_n \) (either \( x_i, \overline{x}_i \), or both, can appear in \( \varphi \)).

Will reduce \( \varphi \) to an IP instance with \( 2n \) variables \( x_1, y_1, \ldots, x_n, y_n \) and \( m + 2n \) linear inequalities, and \( n \) linear equalities (why ?).
Each $x_i$ in $\varphi$ corresponds to the variable $x_i$ in IP.
Each $\overline{x_i}$ in $\varphi$ corresponds to the variable $y_i$ in IP.

For each $i$, we add the inequalities $x_i \geq 0$, $y_i \geq 0$, and the equality $x_i + y_i = 1$
(what do these three express?)

For each clause $k$, we add the inequality
$\sum_{z_j \in \text{Clause}_k} z_j \geq 1$
(what does this inequality express?)

For example, $(x_1 \lor \overline{x_2} \lor \overline{x_3} \lor x_4)$ is translated to
$x_1 + y_2 + y_3 + x_4 \geq 1$.
\( \varphi = (x_1 \lor \overline{x_2} \lor \overline{x_3} \lor x_4) \land (x_3 \lor \overline{x_5} \lor x_6) \land (x_3 \lor \overline{x_6}) \)

translates to

\[
\begin{align*}
x_1 + y_2 + y_3 + x_4 & \geq 1 \\
x_3 + y_5 + x_6 & \geq 1 \\
x_3 + x_6 & \geq 1 \\
x_1 \geq 0, \ y_1 \geq 0, \ x_1 + y_1 & = 1 \\
x_2 \geq 0, \ y_2 \geq 0, \ x_2 + y_2 & = 1 \\
x_3 \geq 0, \ y_3 \geq 0, \ x_3 + y_3 & = 1 \\
x_4 \geq 0, \ y_4 \geq 0, \ x_4 + y_4 & = 1 \\
x_5 \geq 0, \ y_5 \geq 0, \ x_5 + y_5 & = 1 \\
x_6 \geq 0, \ y_6 \geq 0, \ x_6 + y_6 & = 1
\end{align*}
\]
SAT \leq_P IP: Validity (sketch)

Should show
(a) Reduction $g$ is poly-time computable
(b) $\varphi \in \text{SAT} \implies g(\varphi) \in IP$
(c) $g(\varphi) \in IP \implies \varphi \in SAT$. 
SAT $\leq_P$ IP: Validity (sketch)

Should show
(a) Reduction $g$ is poly-time computable
(b) $\varphi \in \text{SAT} \implies g(\varphi) \in \text{IP}$
(c) $g(\varphi) \in \text{IP} \implies \varphi \in \text{SAT}$.

Poly time: easy (verify details!).

Suppose $\varphi \in \text{SAT}$. Take a satisfying assignment.
If $x_i = 1$ assign $x_i = 1, y_i = 0$ in IP.
If $x_i = 0$ assign $x_i = 0, y_i = 1$ in IP.

So "sanity check" constraints satisfied. "Clause constraints" are satisfied due to at least one literal satisfied in each clause., implying $g(\varphi) \in \text{IP}$.

$g(\varphi) \in \text{IP} \implies \varphi \in \text{SAT}$ is similar.
Bounded $A_{TM}$

- **Bounded $A_{TM}$**: Given encoding $\langle M \rangle$ of non-deterministic TM, an input $w$, time bound $1^k$ in unary, does $M$ have an accepting computation of $w$ in $k$ steps or less?

- **Bounded $A_{TM}$** is NP complete, via a “generic” reduction.

- Finding the reduction is easy (check!).

- Proving **Bounded $A_{TM}$** is in NP seems less obvious.
An instance of the problem has the form \( \langle M, w, 1^k \rangle \). The universal NTM, \( U \), decides this language:

- \( U \) does not write anything on input tape.
- Copies \( q_0w \) to second tape.
- Simulates \( M \) step by step, keeping its configuration on second tape.
- Sets up a unary counter on third tape, initialize to zero and increments it for each simulated step of \( M \).
- If counter reaches \( k + 1 \), \( U \) rejects.
- How many steps of \( U \) does it take to simulate \( k \) steps of \( M \)?
Bounded $A_{TM}$ in NP

How many steps $U$ takes to simulate $k$ steps of $M$?

To simulate one step of $M$, NTM $U$ has to find an entry in $M$’s transition function that matches current simulated state and letter under head.

This requires scanning all of $M$’s transition function (on input tape), which takes length of $\langle M \rangle$ steps of $U$.

So to simulate $k$ steps of $M$, NTM $U$ takes $k \cdot |\langle M \rangle|$ steps.

Denote $n = |\langle M, w, 1^k \rangle|$. What is $k \cdot |\langle M \rangle|$ in terms of $n$?

$k \cdot |\langle M \rangle| = \Theta(n^2)$, so $A_{TM} \in NTIME(n^2)$.

Question: What would the NTIME complexity be if $k$ would be encoded in binary?
A legal coloring of an undirected graph is an assignment of colors to the nodes of the graph, such that if two nodes are connected by an edge, they are colored by different colors.
Graph Coloring

- For any specific $k$, a $k$-coloring of $G$ is a coloring that uses at most $k$ colors.
- Of course, a $k$-coloring of $G$ may not exist.
- This raises the question: Given $G$ and $k$, how hard it is to decide if a $k$-coloring of $G$ exists?
- Extreme values of $k$ are easy to decide: E.g. $k = 1$ (does the have any edge?), $k = 2$ (is the graph bipartite? – there is a simple algorithm you can come up with easily), or $k = \text{number of nodes in graph}$ (simply assign each node a different color).
- But what happens beyond trivial cases? For example: "is $G$ 3-colorable".
3-colorability is NP-Complete

Define
3-COL = \{G \mid G \text{ is an undirected 3 colorable graph}\}.
3-COL is NP complete.

Membership of 3-COL in NP is easy:
The certificate is the coloring. The verifier checks this is indeed a coloring (each node assigned exactly one color), and it is legal (nodes connected by edges are assigned different colors).

We will show NP completeness of 3-COL using a polynomial time reduction from 3SAT.
3-colorability is NP-Complete

- Based on formula $\phi$, reduction builds a graph $G$. Each literal from $\phi$ is assigned a node in $G$. The graph contains additional nodes.
- Reduction “forces” each node to be colored either $F$ or $T$ or red.
- Reduction “forces” each literal to be colored legally by either $F$ or $T$.
- Reduction “forces” each $x_i$ and $\overline{x_i}$ to be assigned different colors in a legal coloring.
- For each clause of $\phi$, reduction builds a gadget. Gadget ensures that a 3 coloring of $G$ assigns at least one literal node the “color” $T$.
- Nough said. Time for whiteboard scratching.
3-COL is NP-Complete: Conclusion

- Saw that 3-COL is in NP.
- Saw a poly time reduction from 3SAT to 3-COL (reduction preserves membership and is in poly time).
- Since 3SAT is NPC, so is 3-COL.
Yet More Intractable Problems

- **Subgraph isomorphism** is NP complete (easy reduction from **clique**).

- Graph isomorphism is in NP, seems not to be in P, but we got many good reasons (that are out of scope for this course) to believe it is **not** NP complete.
Chains of Reductions: NPC Problems

- **SAT**
  - IntegerProg
  - **3SAT**
    - Clique
    - 3Color
    - HamPath
      - Scheduling
      - HamCircuit
        - **TRAVELING-SALESMAN**
    - IndepSet
      - VertexCover
        - SetCover
        - 3ExactCover
        - Knapsack
A language $B$ is \textbf{NP-hard} if it satisfies

\begin{itemize}
\item Every $A$ in NP is polynomial time reducible to $B$
\end{itemize}

We do not require $B \in NP$ (membership).

\textbf{Example:} The language

$$\{ \langle \Phi_1, \Phi_2 \rangle \mid \Phi_1 \in SAT, \Phi_2 \notin SAT \}$$

is NP-hard but apparently not NP-complete (why?).
coNP Hardness

A language $B$ is coNP-hard if it satisfies

Every $A$ in coNP is polynomial time reducible to $B$

We do not require $B \in coNP$ (membership).

Example: The language

$$\{\langle \Phi_1, \Phi_2 \rangle \mid \Phi_1 \in SAT, \Phi_2 \notin SAT\}$$

is coNP-hard but apparently not coNP-complete (why?).
Let $R(\cdot, \cdot)$ be a poly time computable predicate.

- **Decision Problem:** Given input $x$, decide if there is some $y$ satisfying $R(x, y)$?
Let $R(\cdot, \cdot)$ be a poly time computable predicate.

- **Decision Problem:** Given input $x$, decide if there is some $y$ satisfying $R(x, y)$?

Using the “certificate” characterization of languages in NP, the decision problem is the same as deciding membership $x \in L$ for $L \in NP$.
Let $R(\cdot, \cdot)$ be a poly time computable predicate.

- **Decision Problem:** Given input $x$, decide if there is some $y$ satisfying $R(x, y)$?

- Using the “certificate” characterization of languages in NP, the decision problem is the same as deciding membership $x \in L$ for $L \in NP$.

- **Search Problem:** Given input $x$, find some $y$ satisfying $R(x, y)$, or declare that none exist.
Let $R(\cdot, \cdot)$ be a poly time computable predicate.

- **Decision Problem**: Given input $x$, decide if there is some $y$ satisfying $R(x, y)$?

- Using the “certificate” characterization of languages in NP, the decision problem is the same as deciding membership $x \in L$ for $L \in NP$.

- **Search Problem**: Given input $x$, find some $y$ satisfying $R(x, y)$, or declare that none exist.

- The search problem seems harder to solve than the decision problem.
Decision, Search, Optimization Problems

- **Search Problem**: Given input $x$, find some $y$ satisfying $R(x, y)$, or declare that none exist.

- Turns out that for **NP complete languages**, search and decision have the “same difficulty”.

- Specifically, given access to an oracle for $L$ (the decision problem), we can solve the search problem in poly time.

- When oracle is successively accessed with queries of decreasing sizes, this technique is known as **self reduction**.

- Examples: SAT and Clique (on board).

- This is applicable to NPC problems but **not** to all of NP.
Decision, Search, Optimization Problems

- **Search Problem:** Given input $x$, find some $y$ satisfying $R(x, y)$, or declare that none exist.

- **Optimization Problem:** Given input $x$, find some $y$ satisfying $R(x, y)$, that is the largest among all solutions (or smallest), or declare that none exist.

- Example 1: Given a graph $G$, find a legal coloring that minimizes number of colors used.

- Example 2: Given a graph $G$, find a clique of largest size possible.
Coping with NP-Completeness

- Approximation algorithms for hard optimization problems.
- Randomized (coin flipping) algorithms.
- Fixed parameter algorithms.
- Heuristics.

These stand in the forefront of current algorithmic research, and could easily fill up to four advanced courses.

(figure from http://wwwbrauer.in.tum.de/gruppen/theorie/hard/vc1.png)
Example: Vertex Cover

Given a graph \((V, E)\)

- find the **smallest** set of vertices \(C\)
- such that for each edge in the graph,
- \(C\) contains at least one endpoint.
Example: Vertex Cover

Given a graph \((V, E)\)

- Find the smallest set of vertices \(C\)
- Such that for each edge in the graph,
- \(C\) contains at least one endpoint.

(figure from www.cc.ioc.ee/jus/gtglossary/assets/vertex_cover.gif)
Vertex Cover

The decision version of this problem is \textbf{NP}-complete by a reduction from \textbf{Independent Set} (proved in recitation).
Vertex Cover

The decision version of this problem is \textbf{NP}-complete by a reduction from \textit{Independent Set} (proved in recitation).

(figure from http://wwwbrauer.in.tum.de/gruppen/theorie/hard/vc1.png)
Approx. Algorithm for VC (Gavril ’74)

- $C := \emptyset$
- while there are edges in $G$
  - choose any edge $(u, v)$ in $G$
  - add $u$ and $v$ to $C$
  - remove them from $G$

**Claim:** This algorithm is a $2$-approximation algorithm for vertex cover.

Meaning $C$ is at most **twice as large** as a minimum vertex cover.
Gavril’s Approximation Algorithm

**Claim:** This is a 2-approximation algorithm.

- Cover $C$ constructed from $|C|/2$ edges of $G$
- no two edges of these share a vertex
- any vertex cover, including the optimum,
- contains at least one node from each of these edges (otherwise an edge would not be covered).

It follows that $OPT(G) \geq |C|/2$

(so $|C|/OPT \leq 2$)

**Remark:** Under some plausible complexity assumption, this factor 2 approximation cannot be improved.