Game plan for today (maximum optimistic):

- Reducing Knapsack to Partition (pg 305)
- The definition of \(NP\)-completeness (pg 308)
- The Bounded Tiling problem (pg 310)
- The Circuit-SAT problem (CLRS pg 1070–1077)
- The map of problems (pg 317)
- Reducing SAT to Exact Cover (pg 318)
- Reducing Exact Cover to HamCycle (pg 320)
- Reducing HamCycle to HamPath (Ex 7.3.3)
Example 7.1.2: Reducing Knapsack to Partition

Knapsack: Given a set $S$ of $n$ integers and capacity $k$, is there [find] a subset of $S$ that sum exactly to $k$?

Partition: Given a set $S$ of $n$ integers, can they be partitioned exactly in half (in terms of their sum)?

Let $S = \{a_1, a_2, \ldots, a_n\}$, $k$ be an instance of Knapsack.

Let $H = \frac{1}{2} \sum_{a_i \in S} a_i$ and make set $S_2 = S \cup \{2H + 2k, 4H\}$. This is an instance of Partition.

Suppose a partition exists for $S_2$, call it $P \cup \{4H\}$ and $(S - P) \cup \{2H + 2k\}$ for some $P \subseteq S$. Then

$$4H + \sum_{a_i \in P} a_i = 2H + 2k + \sum_{a_i \in S - P} a_i$$
$$4H + 2\sum_{a_i \in P} a_i = 2H + 2k + \sum_{a_i \in S} a_i = 2H + 2k + 2H = 4H + 2k$$
$$\sum_{a_i \in P} a_i = k$$

And so $P$ is our solution to Knapsack.

Conversely, suppose there exists $P \subseteq S$, a solution to Knapsack, that is, $\sum_{a_i \in P} a_i = k$. Work backwards algebraically . . .
Definition 7.1.2.: A language $L \subseteq \Sigma^*$ is $\mathcal{NP}$-complete if

1. $L \in \mathcal{NP}$
2. For every language $L' \in \mathcal{NP}$, there is a polynomial reduction from $L'$ to $L$ [L is $\mathcal{NP}$-hard].

Let $\mathcal{NPC}$ be the class of $\mathcal{NP}$-complete languages.

Theorem 7.1.1: $\mathcal{P} = \mathcal{NP}$ iff $\exists L \in \mathcal{NPC}$ such that $L \in \mathcal{P}$.

Proving that a problem is $\mathcal{NP}$-complete shows that it is at least as hard as all the other problems shown to be $\mathcal{NP}$-complete.
**Bounded tiling:** Like the original tiling problem, but we are given the entire first row, and we need to tile only a certain portion, an $s \times s$ square.

The $\mathcal{NP}$-completeness proof:

*Bounded-Tiling is in class $\mathcal{NP}$:* The certificate is the $s \times s$ square. We can check that the square is legal in $O(s^2)$ time. This is polynomial in the size of the input, since the size of the input is $\Omega(s)$.

Now, suppose $L \in \mathcal{NP}$. Then there exists $M$, a nondeterministic Turing machine that decides $L$ in $p(|x|)$ for some polynomial $p$, where $x$ ranges over the candidate strings for $L$.

(Very informal:) Base $s$ on $p(|x|)$, and set up a tiling system analogous to the proof that the original tiling problem is undecidable. A tiling exists iff a computation that accepts $x$ exists (and hence $x \in L$). \qed
A. Prove \( L \in \mathcal{NP} \)

1. Describe a certificate.
2. Demonstrate that the certificate can be used to check a string/solution in polynomial time.
3. Demonstrate that the certificate itself is succinct (polynomial in size) usually easy for our problems—ok to do briefly/informally

B. Prove \( L \) is \( \mathcal{NP} \)-hard

1. Choose a known \( \mathcal{NP} \)-complete problem \( L_2 \).
2. Describe a reduction \( \tau \) from \( L_2 \) to \( L \).
3. Demonstrate \( \tau \) can be computed in polynomial time. (Also usually easy.)
4. Demonstrate that \( x \in L_2 \) iff \( \tau(x) \in L \)
Reducing Sat to Exact Cover:
Suppose \( \{c_1, c_2, \ldots c_\ell\} \) is an instance of Sat.
Define the following instance of Exact Cover:

\[
U = \bigcup \{x_i\} \text{ for each variable } i \\
\text{ } \bigcup \{c_j\} \text{ for each clause } j \\
\text{ } \bigcup \{p_{jk}\} \text{ for each position } k \text{ in clause } j
\]

\[
F = \{ \forall j, k \{p_{jk}\} \\
\forall i \quad T_{i\top} = \{x_i\} \cup \{p_{jk} \mid \lambda_{jk} = \sim x_i\} \\
\forall i \quad T_{i\bot} = \{x_i\} \cup \{p_{jk} \mid \lambda_{jk} = x_i\} \\
\forall j, k \quad \{c_j p_{jk}\} \}
\]


\[
\text{At least one of } T_{i\bot} \text{ or } T_{i\top} \text{ for each } i \text{ must be in the cover, which stands for the truth assignment.}
\]

\[
\text{At least one of } \{c_j p_{jk}\} \text{ must be in the cover, which stands for which literal satisfies clause } j.
\]

\[
\text{The extra } \{p_{jk}\} \text{ sets can be chosen as necessary to account for literals not used in satisfying the formula.}
\]
Proof that HamiltonPath is $\mathcal{NP}$-Complete

**Proof.** [HamiltonPath is $\mathcal{NP}$.] Suppose $G = (V, E)$ is a graph, an instance of the HamiltonPath. Let $p = ⟨u_1, u_2, \ldots, u_n⟩$ be a sequence of vertices from $V$, a proposed Hamilton path in $G$. With any reasonable representation of $G$, one can check that each vertex in $V$ appears uniquely in $p$, and that for any pair of vertices $u_i, u_{i+1}$ as they appear in $p$, the edge $(u_i, u_{i+1})$ is in $E$. Moreover, the path $p$ is smaller than the representation of $G$, so it is succinct.

[HamiltonPath is $\mathcal{NP}$-hard.] Next, suppose $G = (E, V)$ is an instance of HamiltonCycle. Let $v_1 \in V$ be an arbitrary vertex. Let $G' = (V', E')$ be a new graph such that $v_1$ is removed and four new vertices are added, that is, $V' = V - \{v_1\} \cup \{v_a, v_b, v_c, v_d\}$; and every edge that is incident on $v_1$ is replaced with two analogous edges incident on $v_b$ and $v_c$, and edges $(v_a, v_b)$ and $(v_c, v_d)$ are added, that is

$$E' = (E - \{(v_1, v_x) \mid (v_1, v_x) \in E\})$$
$$\cup \{(v_b, v_x), (v_c, v_x) \mid (v_1, v_x) \in E\}$$
$$\cup \{(v_a, v_b), (v_c, v_d)\}$$
This reduction is accomplished by one pass over the edges, which is polynomially computable.

Now, suppose $G$ has a Hamilton cycle, call it $(v_1, v_2, \ldots v_{|V| - 1}, v_1)$. (As a cycle, it has an arbitrary starting/ending point, so we are free to choose $v_1$ as the starting point when naming the cycle.) Then $G'$ has a Hamiltonian path $(v_a, v_b, v_2, \ldots, v_{|V| - 1}, v_c, v_d)$.

Conversely, suppose $G'$ has a Hamiltonian path. Based on how we constructed $G'$ (for example, the only edge going out of $v_a$ is $(v_a, v_b)$, and the only edge going into $v_d$ is $(v_c, v_d)$), that path must be in the form $(v_a, v_b, v_2, \ldots, v_{|V| - 1}, v_c, v_d)$. Then $G$ has a Hamiltonian cycle $(v_1, v_2, \ldots v_{|V| - 1}, v_1)$.

Therefore Hamilton Path is $\mathcal{NP}$-complete. □
Proof that \textsc{Longest Cycle} is $\mathcal{NP}$-Complete

\textbf{Proof.} [\textsc{Longest Cycle is $\mathcal{NP}$.}] Suppose $(G = (V, E), K)$ is an instance of \textsc{Longest Cycle} and $p$ is a path that is a proposed cycle of length $K$. An algorithm to check that $p$ is consistent with $E$, has no repeated vertices, and has length at least $K$, is polynomial with any reasonable representation of $G$. Moreover, since $p$ is no larger than the representation of $G$, it is succinct.

[\textsc{Longest Cycle is $\mathcal{NP}$-hard.}] Suppose $(G = (V, E))$ is an instance of \textsc{Hamilton Cycle}. Then make an instance of \textsc{Longest Cycle} by letting $K = |V|$, which obviously can be done in polynomial time. Since $K = |V|$, any cycle of length (at least) $K$ must be a Hamilton cycle, and any Hamilton cycle must have length $K$.

Therefore \textsc{Longest Cycle} is $\mathcal{NP}$-complete. \qed
Proof that **Subgraph Isomorphism** is $\mathcal{NP}$-Complete

**Proof.** [Subgraph Isomorphism is $\mathcal{NP}$.] Suppose $(G_1 = (V_1, E_1), G_2 = (V_2, E_2))$ is an instance of **Subgraph Isomorphism** and $f$ is a function $V_1 \rightarrow V_2$ (expressed as a list of pairs where $(v_1,a, v_2,b)$ indicates $v_1,a \in V_1$, $v_2,b \in V_2$, and $f(v_1,a) = v_2,b$) proposed as an isomorphism. An algorithm to check that $f$ is a one-to-one function and that for all $(v_1,a, v_1,b) \in E_1$, $(f(v_1,a), f(v_1,b)) \in E_2$, is polynomial with any reasonable representation of $G$. Moreover, since $|f| = O(V_1)$, it is succinct.

[Subgraph Isomorphism is $\mathcal{NP}$-hard.] Suppose $(H = (W, F))$ is an instance of **Hamilton Cycle**. Then construct a graph $G = (V, E)$ such that $|V| = |W|$ and $E = \{(w_1,w_2), (w_2,w_3), \ldots (w_{|V|}, w_1)\}$. An algorithm to construct this graph takes $O(V)$ time.

Note that $E$ has only those edges that make a Hamiltonian cycle. Thus $G$ is isomorphic to a subgraph of $H$ iff $H$ has a Hamiltonian cycle. **Therefore** **Subgraph Isomorphism** is $\mathcal{NP}$-complete. $\square$
Reduction from UHC to TSP (LP pg 324).

Differences between UHC and TSP:

- The graph in TSP is *weighted* (interpreted as distances)
- The graph in TSP is *complete*
- A TSP problem has a *budget*

Suppose we have an instance of UHC, an undirected graph $G = (V, E)$. Construct a graph with the same vertices but complete in its edges and with distances

$$d_{i,j} = \begin{cases} 
0 & \text{if } i = j \\
1 & \text{if } (v_i, v_j) \in E \\
2 & \text{otherwise}
\end{cases}$$

Set the budget to $|V|$. 

Reduction from **Exact Cover** to **Knapsack** (LP pg 325).

Given an instance of **Exact Cover** \((\mathcal{U}, \mathcal{F} \subseteq \mathcal{P}(\mathcal{U}))\), construct an instance of **Knapsack** \((S, K)\):

- \(S = \{1, 2, \ldots |\mathcal{U}|\}\)
- \(K = 2^{|\mathcal{U}|} - 1 = \sum_{i=0}^{|\mathcal{U}|-1} 1\)

Interpret each set in \(\mathcal{P}(S)\) as a bit vector.

Problem: Consider \(S = \{1, 2, 3, 4\}\) and proposed cover \(\{\{1, 3\}, \{1, 4\}, \{1\}\}\).
**Independent Set** problem: Given a graph, is there a set of vertices of size $k$ with none adjacent to each other?

Reduction from **3Sat** to **Independent Set** (LP pg 326–327.)

Suppose we have an instance of **3Sat**, $F = C_1 \land C_2 \land \cdots \land C_m$. WOLOG, suppose each clause has exactly three literals. Construct an instance of **Independent Set**, $(G, K)$ where $K = m$ and $G = (V, E)$ such that

- There is a vertex in $V$ for each literal occurrence (or clause position) $c_{i,j}$.
- $(c_{i,j}, c_{x,y}) \in E$ if either
  - $i = x$ (they are positions in the same clause; this makes a triangle of vertices), or
  - the literals $c_{i,j}$ and $c_{x,y}$ are negations of each other.

Suppose an independent set of size $K$ exists in $G$. It must include exactly one vertex in each triangle. Make a truth assignment that makes each literal in the set true. Suppose a satisfying truth assignment exists. Then for each triangle, pick one vertex corresponding to a true literal.