Lecture 16:
NP-Complete Problems:
THEY’RE EVERYWHERE!

6.045
Polynomial Time Reducibility

$f : \Sigma^* \rightarrow \Sigma^*$ is a polynomial time computable function if there is a poly-time Turing machine $M$ that on every input $w$, halts with just $f(w)$ on its tape.

Language $A$ is poly-time reducible to language $B$, written as $A \leq_p B$, if there is a poly-time computable function $f : \Sigma^* \rightarrow \Sigma^*$ so that:

$$w \in A \iff f(w) \in B$$

$f$ is a polynomial time reduction from $A$ to $B$.

Note there is a $k$ such that for all $w$, $|f(w)| \leq k|w|^k$.
Definition: A language B is NP-complete if:

1. B ∈ NP
2. Every A in NP is poly-time reducible to B
   That is, A ≤_p B
   When this is true, we say “B is NP-hard”

The Cook-Levin Theorem:
3SAT is NP-complete
“Simple Logic can encode any NP problem!”
The Cook-Levin Theorem:
3SAT is NP-complete
“Simple Logic can encode any NP problem!”

Today we’ll see many more NP-complete problems: NHALT, 3SAT, CLIQUE, IS, VC, SUBSET-SUM, KNAPSACK, PARTITION, BIN-PACKING, ...
(There are entire classes at MIT on this kind of stuff)

And even more on pset/pests...

For all of these problems, assuming P ≠ NP, they are not in P
There are thousands of *natural* NP-complete problems!

Your favorite topic certainly has an NP-complete problem somewhere in it.

Even the other sciences are not safe: biology, chemistry, physics have NP-complete problems too!
Reading Assignment

Read Luca Trevisan’s notes for an alternative proof of the Cook-Levin Theorem!

Sketch:
1. Define **Circuit-SAT**: Given a logical circuit $C$, is there an input $a$ such that $C(a)=1$?
2. Show that **Circuit-SAT** is NP-hard: The $n^k \times n^k$ tableau for $N$ on $w$ can be simulated using a logical circuit of $O(n^{2k})$ gates.
3. Reduce **Circuit-SAT** to 3SAT in polytime.
4. Conclude 3SAT is also NP-hard.
Theorem (Cook-Levin): 3SAT is NP-complete

Corollary: 3SAT \notin P if and only if P \neq NP

Given a new problem \( L \in NP \), how can we prove it is NP-hard?

Generic Recipe:
1. Take a problem \( L' \) that you know to be NP-hard (e.g., 3SAT)
2. Prove that \( L' \leq_p L \)

Then for all \( A \in NP \), \( A \leq_p L' \) by (1), and \( L' \leq_p L \) by (2)
This implies \( A \leq_p L \). Therefore \( L \) is NP-hard!
L is NP-Complete
Given a graph $G$ and positive $k$, does $G$ contain a complete subgraph on $k$ nodes?

$\text{CLIQUE} = \{ (G,k) \mid G \text{ is an undirected graph with a } k\text{-clique} \}$
The Clique Problem

Given a graph G and positive k, does G contain a complete subgraph on k nodes?

\[
\text{CLIQUE} = \{ (G,k) \mid G \text{ is an undirected graph with a } k\text{-clique} \}
\]

Theorem (Karp): CLIQUE is NP-complete

Why is it in NP?
Theorem: CLIQUE is NP-Complete
3SAT $\leq_p$ CLIQUE

Transform every 3-cnf formula $\phi$ into $(G,k)$ such that

$$\phi \in 3\text{SAT} \iff (G,k) \in \text{CLIQUE}$$

Want transformation that can be done in time that is polynomial in the length of $\phi$

How can we encode a *logic* problem as a *graph* problem?
3SAT $\leq_p$ CLIQUE

We transform any 3-cnf formula $\phi$ into $(G,k)$ such that

$$\phi \in 3SAT \iff (G,k) \in CLIQUE$$

Let $C_1, C_2, \ldots, C_m$ be clauses of $\phi$, let $x_1, \ldots, x_n$ be vars.
Set $k := m$
Make a graph $G$ with $m$ groups of 3 nodes each.

Idea: Group $i$ corresponds to clause $C_i$ of $\phi$

Each node in group $i$ is “labeled” by a literal of $C_i$
(Note these labels do not actually appear in the graph!)

Put edges between all pairs of nodes in different groups, except for pairs of nodes with labels $x_i$ and $\neg x_i$

Put no edges between nodes in the same group
\((x_1 \lor x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2 \lor \neg x_2) \land (\neg x_1 \lor x_2 \lor x_2)\)

\(|V| = 3(\text{number of clauses})\)

\(k = \text{number of clauses}\)
\((x_1 \lor x_1 \lor x_1) \land (\neg x_1 \lor \neg x_1 \lor x_2) \land (x_2 \lor x_2 \lor x_2) \land (\neg x_2 \lor \neg x_2 \lor x_1)\)
Claim: $\phi \in 3\text{SAT} \iff (G,m) \in \text{CLIQUE}$

Claim: If $\phi \in 3\text{SAT}$ then $(G,m) \in \text{CLIQUE}$

Proof: Let $A$ be a SAT assignment of $\phi$.
For each clause $C$ of $\phi$, there is a literal in $C$ set true by $A$.
Let $v_C$ be that literal’s corresponding vertex in $G$.

Claim: $S = \{v_C \mid C \text{ is a clause in } \phi\}$ is an $m$-clique in $G$.

Proof: Let $v_C \neq v_{C'}$ be in $S$. Suppose $(v_C, v_{C'}) \notin E$.
Note $v_C$ and $v_{C'}$ are from different groups. So they must label inconsistent literals, call these literals $x$ and $\lnot x$.
But assignment $A$ cannot set true both $x$ and $\lnot x$!
Contradiction. So $(v_C, v_{C'}) \in E$, for all $v_C, v_{C'} \in S$.
Hence $S$ is an $m$-clique, and $(G,m) \in \text{CLIQUE}$.
Claim: \( \phi \in 3\text{SAT} \iff (G,m) \in \text{CLIQUE} \)

Claim: If \((G,m) \in \text{CLIQUE}\) then \(\phi \in 3\text{SAT}\)

Proof: Let \(S\) be an \(m\)-clique of \(G\).
We’ll construct a satisfying assignment \(A\) of \(\phi\).
Claim: \(S\) contains exactly one node from each group of \(G\).

For each variable \(x\) of \(\phi\), define variable assignment \(A\):\n\[ A(x) := 1, \text{ if there is a vertex in } S\text{ with label } x, \]
\[ A(x) := 0, \text{ if there is a vertex in } S\text{ with label } \neg x, \]
or no vertices in \(S\) are labeled \(x\) or \(\neg x\)

For all \(i = 1,\ldots,m\), one vertex from the \(i\)-th group is in \(S\).
\[ \Rightarrow \text{one literal from the } i\text{-th clause of } \phi \text{ is a vertex in } S \]
So for all \(i = 1,\ldots,m\), \(A\) sets at least one literal true in \(i\)-th clause of \(\phi\). Therefore \(A\) is a satisfying assignment to \(\phi\).
Independent Set is NP-hard

**IS:** Given a graph $G = (V, E)$ and integer $k$, is there $S \subseteq V$ such that $|S| \geq k$ and no pair of vertices in $S$ have an edge?

**CLIQUE:** Given $G = (V, E)$ and integer $k$, is there $S \subseteq V$ such that $|S| \geq k$ and every pair of vertices in $S$ have an edge?

**CLIQUE \leq_p IS:**
Given $G = (V, E)$, output $G' = (V, E')$ where $E' = \{(u,v) \mid (u,v) \not\in E\}$.

$(G, k) \in \text{CLIQUE}$ iff $(G', k) \in \text{IS}$

each $k$-Clique in $G$ is an $k$-IS in $G'$
The Vertex Cover Problem

vertex cover = set of nodes C that cover all edges
For all edges, at least one endpoint is in C
VERTEX-COVER = \{ (G,k) \mid G \text{ is a graph with a vertex cover of size at most } k \}\]

**Theorem:** VERTEX-COVER is NP-Complete

(1) VERTEX-COVER $\in$ NP

(2) IS $\leq_p$ VERTEX-COVER

**Want to transform a graph G and integer k into G’ and k’ such that**

\[(G,k) \in IS \iff (G’,k’) \in \text{VERTEX-COVER}\]
Claim: For every graph $G = (V,E)$, and subset $S \subseteq V$, 

$S$ is an independent set if and only if $(V - S)$ is a vertex cover

Proof: $S$ is an independent set

$\iff (\forall u, v \in V)[ (u \in S \text{ and } v \in S) \implies (u,v) \notin E ]$

$\iff (\forall u, v \in V)[ (u,v) \in E \implies (u \notin S \text{ or } v \notin S) ]$

$\iff (V - S)$ is a vertex cover!

Therefore $(G,k) \in IS \iff (G, |V| - k) \in \text{VERTEX-COVER}$

Our polynomial time reduction: $f(G,k) := (G, |V| - k)$
The Subset Sum Problem

Given: Set $S = \{a_1, ..., a_n\}$ of positive integers and a positive integer $t$

Is there an $A \subseteq \{1, ..., n\}$ such that $t = \sum_{i \in A} a_i$?

SUBSET-SUM = $\{(S, t) \mid \exists S' \subseteq S \text{ s.t. } t = \sum_{b \in S'} b\}$

A simple summation problem!

Theorem (in algs): There is a $O(n \cdot t)$ time algorithm for solving SUBSET-SUM.

But $t$ can be specified in $(\log t)$ bits... this isn’t an algorithm that runs in poly-time in the input!
The Subset Sum Problem

Given: Set $S = \{a_1, \ldots, a_n\}$ of positive integers and a positive integer $t$

Is there an $A \subseteq \{1, \ldots, n\}$ such that $t = \sum_{i \in A} a_i$?

$\text{SUBSET-SUM} = \{(S, t) \mid \exists S' \subseteq S \text{ s.t. } t = \sum_{b \in S'} b\}$

A simple summation problem!

Theorem: $\text{SUBSET-SUM}$ is NP-complete
VC $\leq_p$ SUBSET-SUM

Want to reduce a graph to a set of numbers

Given (G, k), let $E = \{e_0, ..., e_{m-1}\}$ and $V = \{1, ..., n\}$

Our subset sum instance $(S, t)$ will have $|S| = n + m$

“Edge numbers”:
For every $e_j \in E$, put $b_j = 4^j$ in $S$

“Node numbers”:
For every $i \in V$, put $a_i = 4^m + \sum_{j : i \in e_j} 4^j$ in $S$

Set the target number: $t = k \cdot 4^m + \sum_{j=0}^{m-1} (2 \cdot 4^j)$

Think of the numbers as being in “base 4”... as vectors with m+1 components
Claim: If \((G,k) \in \text{VC}\) then \((S,t) \in \text{SUBSET-SUM}\)

Suppose \(C \subseteq V\) is a VC with \(k\) vertices.

Define \(S' = \{a_i : i \in C\} \cup \{b_j : |e_j \cap C| = 1\}\)

\(S' = (\text{node numbers corresponding to nodes in } C) \text{ plus }\)
\(\text{(edge numbers corresponding to edges covered only once by } C)\)

Claim: The sum of all numbers in \(S'\) equals \(t\)

\[
\sum_{i \in C} a_i = k \cdot 4^m + \sum_{i \in C} \left( \sum_{j : i \in e_j} 4^j \right) \\
= k \cdot 4^m + \sum_{j : e_j \text{ covered once by } C} 4^j + \sum_{j : e_j \text{ covered twice by } C} (2 \cdot 4^j) \\
\sum_{j : |e_j \cap C| = 1} b_j = \sum_{j : e_j \text{ covered once by } C} 4^j \quad \text{Total sum is } t
\]
Claim: If \((S,t) \in \text{SUBSET-SUM}\) then \((G,k) \in \text{VC}\)

Suppose \(C \subseteq V\) and \(F \subseteq E\) satisfy
\[
\sum_{i \in C} a_i + \sum_{e_j \in F} b_j = t = k \cdot 4^m + \sum_{j=0}^{m-1} (2 \cdot 4^j)
\]

Claim: \(C\) is a vertex cover of size \(k\).

Proof: Subtract the \(b_j\) numbers from the LHS.

Each \(b_j = 4^j\). So what remains is a sum of the form:
\[
\sum_{i \in C} a_i = k \cdot 4^m + \sum_{j=0}^{m-1} (c_j \cdot 4^j)
\]

where each \(c_j > 0\). But \(c_j = \text{number of nodes in } C\) covering \(e_j\)

Therefore every \(e_j\) is covered by \(C\), so \(C\) is a vertex cover!

Moreover, \(|C| = k\): each \(a_i\) in \(C\) adds \(4^m\) to \(t\)
The Knapsack Problem

Given: $S = \{(v_1, c_1), \ldots, (v_n, c_n)\}$ of pairs of positive integers (items)

- a capacity budget $C$
- a value target $V$

Is there an $S' \subseteq \{1, \ldots, n\}$ such that

$$\left(\sum_{i \in S'} v_i\right) \geq V \text{ and } \left(\sum_{i \in S'} c_i\right) \leq C ?$$

Define: $\text{KNAPSACK} = \{(S, C, V) \mid \text{the answer is yes}\}$

A classic economics/logistics/OR problem!

Theorem: $\text{KNAPSACK}$ is NP-complete
KNAPSACK is NP-complete

KNAPSACK is in NP?

Theorem: SUBSET-SUM $\leq_p$ KNAPSACK

Proof: Given an instance $(S = \{a_1,\ldots,a_n\}, t)$ of SUBSET-SUM, create a KNAPSACK instance:

For all $i$, set $(v_i, c_i) := (a_i, a_i)$

Define $T = \{(v_1, c_1),\ldots,(v_n, c_n)\}$

Define $C := V := t$

Then, $(S,t) \in$ SUBSET-SUM $\iff (T,C,V) \in$ KNAPSACK

Subset of $S$ that sums to $t =$ Solution to the Knapsack instance!