More

NP

Completeness
Final Exam Details

• Final exam is **Wednesday, December 12** from 12:15 – 3:15PM in **Cubberly Auditorium**.

• Covers material up through and including Wednesday's lecture.

• Exam focuses primarily on material starting with DFAs and NFAs, though there will be at least one midterm-style question on the exam.

• If you need to take the final exam at an alternate time, please contact us as soon as possible so that we can make arrangements.
Exam Review

- Two final exam review sessions this weekend:
  - Saturday, 2PM – 5PM in Gates 104
  - Sunday, 2PM – 5PM in Gates 104
- There is an **extra credit practice final exam** available right now.
  - Worth 5 points extra credit if you make an honest effort to complete all the problems.
  - Due at the time that you take the exam.
  - No solutions released; come talk to us during office hours or the review session if you have questions!
- Second practice exam will be released on Wednesday along with solutions, though not for extra credit.
Previously on CS103...
A language $L$ is called **NP-hard** iff for every $L' \in \text{NP}$, we have $L' \leq_p L$.

A language in $L$ is called **NP-complete** iff $L$ is NP-hard and $L \in \text{NP}$.

The class **NPC** is the set of NP-complete problems.
The Tantalizing Truth

**Theorem:** If *any* NP-complete language is in \( P \), then \( P = NP \).
The Tantalizing Truth

**Theorem:** If *any* NP-complete language is not in P, then P $\neq$ NP.
3-CNF

- A propositional formula is in **3-CNФ** if
  - It is in CNF, and
  - Every clause has *exactly* three literals.

- For example:
  - \((x \lor y \lor z) \land (\neg x \lor \neg y \lor z)\)
  - \((x \lor x \lor x) \land (y \lor \neg y \lor \neg x) \land (x \lor y \lor \neg y)\)
  - But not \((x \lor y \lor z \lor w) \land (x \lor y)\)

- The language **3SAT** is defined as follows:

  \[
  3SAT = \{ \langle \varphi \rangle \mid \varphi \text{ is a satisfiable 3-CNФ formula} \}
  \]

- **Theorem (Cook-Levin):** 3SAT is \textbf{NP}-complete.
The Structure of 3CNF

Each clause must have at least one true literal in it…
The Structure of 3CNF

\[
( x \lor y \lor \neg z ) \land ( \neg x \lor \neg y \lor z ) \land ( \neg x \lor y \lor \neg z )
\]

... subject to the constraint that we never choose a literal and its negation
**NP-Completeness**

**Theorem:** If $L \in \text{NPC}$, $L \leq_p L'$, and $L' \in \text{NP}$, then $L' \in \text{NPC}$. 
Structuring \textbf{NP}-Completeness Reductions
The Shape of a Reduction

- Polynomial-time reductions work by solving one problem with a solver for a different problem.

- Most problems in \( \textbf{NP} \) have different pieces that must be solved simultaneously.

- For example, in 3SAT:
  - Each clause must be made true,
  - but no literal and its complement may be picked.
Reductions and Gadgets

• Many reductions used to show \( \text{NP} \)-completeness work by using gadgets.

• Each piece of the original problem is translated into a “gadget” that handles some particular detail of the problem.

• These gadgets are then connected together to solve the overall problem.
Gadgets in INDSET

Each of these gadgets is designed to solve one part of the problem: ensuring each clause is satisfied.
Gadgets in INDSET

These connections ensure that the solutions to each gadget are linked to one another.
Gadgets in INDSET
A More Complex Reduction
A 3-coloring of a graph is a way of coloring its nodes one of three colors such that no two connected nodes have the same color.
The 3-Coloring Problem

• The **3-coloring problem** is

  Given an undirected graph $G$, is there a legal 3-coloring of its nodes?

• As a formal language:

  $$3\text{COLOR} = \{ \langle G \rangle \mid G \text{ is an undirected graph with a legal 3-coloring.} \}$$

• This problem is known to be **NP-complete** by a reduction from 3SAT.
3COLOR ∈ NP

- We can prove that 3COLOR ∈ NP by designing a polynomial-time nondeterministic TM for 3COLOR.

M = “On input ⟨G⟩:

- **Nondeterministically** guess an assignment of colors to the nodes.
- **Deterministically** check whether it is a 3-coloring.
- If so, accept; otherwise reject.”
A Note on Terminology

- Although 3COLOR and 3SAT both have “3” in their names, the two are very different problems.
  - 3SAT means “there are three literals in every clause.” However, each literal can take on only one of two different values.
  - 3COLOR means “every node can take on one of three different colors.”

**Key difference:**
- In 3SAT variables have two choices of value.
- In 3COLOR nodes have three choices of value.
Why Not Two Colors?

- It would seem that 2COLOR (whether a graph has a 2-coloring) would be a better fit.
  - Every variable has one of two values.
  - Every node has one of two values.
- Interestingly, 2COLOR is known to be in \( \mathbf{P} \) and is conjectured not to be \( \mathbf{NP} \)-complete.
  - Though, if you can prove that it is, you've just won $1,000,000!
From 3SAT to 3COLOR

• In order to reduce 3SAT to 3COLOR, we need to somehow make a graph that is 3-colorable iff some 3-CNF formula $\phi$ is satisfiable.

• **Idea**: Use a collection of gadgets to solve the problem.
  • Build a gadget to assign two of the colors the labels “true” and “false.”
  • Build a gadget to force each variable to be either true or false.
  • Build a series of gadgets to force those variable assignments to satisfy each clause.
Gadget One: Assigning Meanings

These nodes must all have different colors.

The color assigned to T will be interpreted as "true." The color assigned to F will be interpreted as "false." We do not associate any special meaning with O.
Gadget Two: Forcing a Choice

\[(x \lor y \lor z) \land (\neg x \lor y \lor z) \land (\neg x \lor y \lor \neg z)\]
Gadget Three: Clause Satisfiability

\((x \lor y \lor \neg z)\)

This node is colorable iff one of the inputs is the same color as T
Putting It All Together

• Construct the first gadget so we have a consistent definition of true and false.

• For each variable \( v \):
  • Construct nodes \( v \) and \( \neg v \).
  • Add an edge between \( v \) and \( \neg v \).
  • Add an edge between \( v \) and \( O \) and between \( \neg v \) and \( O \).

• For each clause \( C \):
  • Construct the earlier gadget from \( C \) by adding in the extra nodes and edges.
Putting It All Together

C_1 \quad C_2 \quad ... \quad C_n

T \quad F \quad O

\bar{x}_1 \quad x_1 \quad \bar{x}_k \quad x_k
Analyzing the Reduction

- How large is the resulting graph?
- We have $O(1)$ nodes to give meaning to “true” and “false.”
- Each variable gives $O(1)$ nodes for its true and false values.
- Each clause gives $O(1)$ nodes for its colorability gadget.
- Collectively, if there are $n$ clauses, there are $O(n)$ variables.
- Total size of the graph is $O(n)$. 
Another \textbf{NP}-Complete Problem
Let $U$ be a set of elements (the **universe**) and $S \subseteq \mathcal{P}(U)$. An **exact covering** of $U$ is a collection of sets $I \subseteq S$ such that every element of $U$ belongs to exactly one set in $I$. 

$U = \{1, 2, 3, 4, 5, 6\}$

$\begin{align*}
S &= \{1, 2, 5\}, \{2, 5\}, \{1, 3, 6\}, \\
& \quad \{2, 3, 4\}, \{4\}, \{1, 5, 6\}\end{align*}$
Applications of Exact Covering

\[
\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{array}
\]

\[
\{C, 1, 4, 5\} \\
\{C, 1, 2, 4\} \\
\{C, 1, 2, 5\} \\
\{C, 2, 4, 5\} \\
\{M, 1, 4, 7\} \\
\{M, 2, 5, 8\} \\
\{M, 3, 6, 9\}
\]
Exact Covering

• Given a universe $U$ and a set $S \subseteq \mathcal{P}(U)$, the exact covering problem is

  Does $S$ contain an exact covering of $U$?

• As a formal language:

  \[
  \text{EXACT-COVER} = \{ \langle U, S \rangle \mid S \subseteq \mathcal{P}(U) \text{ and } S \text{ contains an exact covering of } U \}
  \]
EXACT-COVER ∈ NPC

- We will prove that EXACT-COVER is NP-complete.
- To do this, we will show that
  - EXACT-COVER ∈ NP, and
  - 3COLOR ≤_p EXACT-COVER
- Note that we're using the fact that 3COLOR is NP-complete to establish that EXACT-COVER is NP-hard.
EXACT-COVER ∈ NP

- Here is a polynomial-time verifier for EXACT-COVER:

  - $V = \text{“On input } \langle U, S, I \rangle:\$
    - Verify that every set in $S$ is a subset of $U$.
    - Verify that every set in $I$ is an element of $S$.
    - Verify that every element of $U$ belongs to an element of $I$.
    - Verify that every element of $U$ belongs to at most one element of $I$.”
3COLOR $\leq_p$ EXACT-COVER

- We now reduce 3-colorability to the exact cover problem.
- A graph is 3-colorable iff
  - Every node is assigned one of three colors, and
  - No two nodes connected by an edge are assigned the same color.
- We will construct our universe $U$ and sets $S$ such that an exact covering
  - Assigns every node in $G$ one of three colors, and
  - Never assigns two adjacent nodes the same color.
\{ W, R_w, R_y, R_z \}
\{ W, G_w, G_y, G_z \}
\{ W, B_w, B_y, B_z \}
\{ X, R_x, R_z \}
\{ X, G_x, G_z \}
\{ X, B_x, B_z \}
\{ Y, R_y, R_w, R_z \}
\{ Y, G_y, G_w, G_z \}
\{ Y, B_y, B_w, B_z \}
\{ Z, R_z, R_w, R_y \}
\{ Z, G_z, G_w, G_y \}
\{ Z, B_z, B_w, B_y \}
\{ W, R_W, R_Y, R_Z \} \\
\{ W, G_W, G_Y, G_Z \} \\
\{ W, B_W, B_Y, B_Z \} \\
\{ X, R_X, R_Z \} \\
\{ X, G_X, G_Z \} \\
\{ X, B_X, B_Z \} \\
\{ Y, R_Y, R_W, R_Z \} \\
\{ Y, G_Y, G_W, G_Z \} \\
\{ Y, B_Y, B_W, B_Z \} \\
\{ Z, R_Z, R_W, R_Y \} \\
\{ Z, G_Z, G_W, G_Y \} \\
\{ Z, B_Z, B_W, B_Y \}
Nothing covers this element, since \(X\) has no blue neighbors.
Two sets cover this element, since Z has two green neighbors.

\[
\{ \{ W, R_w, R_y, R_z \} \}
\{ \{ W, G_w, G_y, G_z \} \}
\{ \{ W, B_w, B_y, B_z \} \}
\{ \{ X, R_x, R_z \} \}
\{ \{ X, G_x, G_z \} \}
\{ \{ X, B_x, B_z \} \}
\{ \{ Y, R_y, R_w, R_z \} \}
\{ \{ Y, G_y, G_w, G_z \} \}
\{ \{ Y, B_y, B_w, B_z \} \}
\{ \{ Z, R_z, R_w, R_y \} \}
\{ \{ Z, G_z, G_w, G_y \} \}
\{ \{ Z, B_z, B_w, B_y \} \}
\]
Correction 1: Filling in Gaps
Correction 2: Avoiding Duplicates
\{ W, R_{WY}, R_{WZ} \} \quad \{ R_{WY} \}
\{ W, G_{WY}, G_{WZ} \} \quad \{ R_{WZ} \}
\{ W, B_{WY}, B_{WZ} \} \quad \{ R_{XZ} \}
\{ X, R_{XZ} \} \quad \{ R_{YZ} \}
\{ X, G_{XZ} \} \quad \{ G_{WY} \}
\{ X, B_{XZ} \} \quad \{ G_{WZ} \}
\{ Y, R_{WY}, R_{YZ} \} \quad \{ G_{XZ} \}
\{ Y, G_{WY}, G_{YZ} \} \quad \{ G_{YZ} \}
\{ Y, B_{WY}, B_{YZ} \} \quad \{ B_{WY} \}
\{ Z, R_{WZ}, R_{XZ}, R_{YZ} \} \quad \{ B_{WZ} \}
\{ Z, G_{WZ}, G_{XZ}, G_{YZ} \} \quad \{ B_{XZ} \}
\{ Z, B_{WZ}, B_{XZ}, B_{YZ} \} \quad \{ B_{YZ} \}
\{ W, R_{wy}, R_{wz} \} \quad \{ R_{wy} \}
\{ W, G_{wy}, G_{wz} \} \quad \{ R_{wz} \}
\{ W, B_{wy}, B_{wz} \} \quad \{ R_{xz} \}
  \quad \{ X, R_{xz} \} \quad \{ R_{yz} \}
\{ X, G_{xz} \} \quad \{ G_{wy} \}
\{ X, B_{xz} \} \quad \{ G_{wz} \}
\{ Y, R_{wy}, R_{yz} \} \quad \{ G_{xz} \}
\{ Y, G_{wy}, G_{yz} \} \quad \{ G_{yz} \}
\{ Y, B_{wy}, B_{yz} \} \quad \{ B_{wy} \}
\{ Z, R_{wz}, R_{xz}, R_{yz} \} \quad \{ B_{wz} \}
\{ Z, G_{wz}, G_{xz}, G_{yz} \} \quad \{ B_{xz} \}
\{ Z, B_{wz}, B_{xz}, B_{yz} \} \quad \{ B_{yz} \}
The Construction

• For each node $v$ in graph $G$, construct four elements in the universe $U$:
  • An element $v$.
  • Elements $R_v$, $G_v$, and $B_v$.

• For each edge $\{u, v\}$ in graph $G$, construct three elements in the universe $U$:
  • Elements $R_{uv}$, $G_{uv}$, $B_{uv}$

• Total size of the universe $U$: $O(|V| + |E|)$. 
The Construction

- For each node $v$ in graph $G$, construct a set belonging to $S$ containing
  - The element $v$,
  - Each $R_{uv}$ for each edge $\{u, v\}$ in the graph.
- Repeat the above for colors G and B.
- Add singleton sets containing each individual element except for elements corresponding to nodes.
- Total size of all sets is $O(|V| + |E|)$
  - Counts each node three times and each edge six times.
The Story So Far

3SAT

INDSET

3COLOR

EXACT-COVER
Another NP-Complete Problem
Given a set \( S \subseteq \mathbb{N} \) and a natural number \( k \), the \textbf{subset sum problem} is to find a subset of \( S \) whose sum is exactly \( k \).
MY HOBBY:
EMBEDDING NP-COMPLETE PROBLEMS IN RESTAURANT ORDERS

CHOTCHKIES RESTAURANT

APPETIZERS

- Mixed Fruit  $2.15
- French Fries  $2.75
- Side Salad  $3.35
- Hot Wings  $3.55
- Mozzarella Sticks  $4.20
- Sampler Plate  $5.80

SANDWICHES

- Barbecue  $6.55

WE'D LIKE EXACTLY $15.05 WORTH OF APPETIZERS, PLEASE.

...EXACTLY? UHH...

HERE, THESE PAPERS ON THE KNAPSACK PROBLEM MIGHT HELP YOU OUT.

LISTEN, I HAVE SIX OTHER TABLES TO GET TO—

AS FAST AS POSSIBLE, OF COURSE. WANT SOMETHING ON TRAVELING SALESMAN?
Subset Sum

• Given a set $S \subseteq \mathbb{N}$ and a natural number $k$, the subset sum problem is

  Is there a subset of $S$ with sum exactly $k$?

• As a formal language:

\[
\text{SUBSET-SUM} = \{ \langle S, k \rangle \mid S \subseteq \mathbb{N}, k \in \mathbb{N} \text{ and there is a subset of } S \text{ with sum exactly } k \} \]
SUBSET-SUM ∈ NPC

• We will prove that SUBSET-SUM is NP-complete.

• To do this, we will show that
  • SUBSET-SUM ∈ NP, and
  • EXACT-COVER ≤_p SUBSET-SUM

• Again, we're using our new NP-complete problem to show other languages are NP-complete.
Here is a nondeterministic polynomial-time algorithm for SUBSET-SUM:

N = “On input \langle S, k \rangle:

- **Nondeterministically** guess a subset \( I \subseteq S \).
- **Deterministically** verify whether the sum of the elements of \( I \) is equal to \( k \).
- If so, accept; otherwise reject.”
EXACT-COVER $\leq_p$ SUBSET-SUM

• We now reduce exact cover to subset sum.
• The exact cover problem has a solution iff
  • Every element of the universe belongs to at least one set, and
  • Every element of the universe belongs to at most one set.
• We will construct our set $S$ and number $k$ such that
  • Each number corresponds to a set of elements, and
  • $k$ corresponds to the universe $U$. 
$S = \left\{ \{1, 2, 5\}, \{2, 5\}, \{1, 3, 6\}, \{2, 3, 4\}, \{4\}, \{1, 5, 6\} \right\}$

$U = \left\{ 1, 2, 3, 4, 5, 6 \right\}$
\[ S = \{1, 2, 5\}, \{2, 5\}, \{1, 3, 6\}, \{2, 3, 4\}, \{4\}, \{1, 5, 6\} \]

\[ U = \{1, 2, 3, 4, 5, 6\} \]

\[ S' = \{110010, 010010, 101001, 011100, 000100, 100011\} \]

\[ k = 1111111 \]
The Basic Intuition

- Suppose there are $n$ elements in the universe and $k$ different sets.
- Replace each set $S$ with a number that is 1 in its $i$th position if $i \in S$ and has a 0 in its $i$th position otherwise.
- Set $k$ to a number that is $n$ copies of the number 1.
A Slight Complexity

• To ensure that the columns don't overflow, write the numbers in base \((B + 1)\) where \(B\) is the total number of sets.

• That way, the columns can't overflow from one column into the next.
The Story So Far

3SAT

INDSET

3COLOR

EXACT-COVER

SUBSET-SUM
Yet Another $\textbf{NP}$-Complete Problem
Given a set \( S \subseteq \mathbb{N} \), the **partitioning problem** is to find a way to split \( S \) into two sets with equal sum.
Partitioning

• Given a set $S \subseteq \mathbb{N}$, the partitioning problem is

  **Can $S$ be split into two sets whose sums are the same?**

• As a formal language:

  $\text{PARTITION} = \{ \langle S \rangle \mid S \subseteq \mathbb{N}, \text{ and there is a way to split } S \text{ into two sets with the same sum. } \}$
\textbf{PARTITION} \in \textbf{NPC}

- We will prove that \textit{PARTITION} is \textbf{NP}-complete.

- To do this, we will show that
  - \textit{PARTITION} \in \textbf{NP}, and
  - \textit{SUBSET-SUM} \leq_{p} \textit{PARTITION}

- Sense a pattern? 😊
PARTITION ∈ NP

- Here is a polynomial-time verifier for PARTITION:

  \( V = \text{"On input } \langle S, S_1, S_2 \rangle:\) 
  
  - Check that \( S_1 \cup S_2 = S \) and that \( S_1 \cap S_2 = \emptyset \).
  - Check that the sum of the elements in \( S_1 \) equals the sum of the elements in \( S_2 \).
  - If so, accept; otherwise, reject."
**SUBSET-SUM \( \leq_p \) PARTITION**

- We now reduce subset sum to partitioning.
- The subset sum has a solution iff
  - Some subset of the master set \( S \) is equal to \( k \).
- We will construct our new set \( S' \) such that
  - If a subset of \( S \) has total \( k \), we can add in a new element to make up the difference to half the total sum.
\{137, 42, 271, 103, 154, 16, 3\}

\[k = 452\]

Total of all elements in this set: 726

\[726 - 452 = 274\]

\[452 - 274 = 178\]

\{137, 42, 271, 103, 154, 16, 3, 178\}
The General Idea

- Add in a new element to the set such that a subset with the appropriate sum also forms a partition.
- The new element added in might need to go in the subset that originally added to $k$, or it might have to go in the complement of that set.
The Story So Far

3SAT

INDSET

3COLOR

EXACT-COVER

SUBSET-SUM

PARTITION
One Final NP-Complete Problem
Given a set $J$ of jobs that take some amount of time to complete and $k$ workers, the job scheduling problem is to minimize the total time required to complete all jobs (called the makespan).
Job Scheduling

• Given a set $J$ of jobs of different lengths, a number of workers $k$, and a number $t$, the job scheduling problem is

  Can the jobs in $J$ be assigned to the $k$ workers such that all jobs are finished within $t$ units of time?

• As a formal language:

  $JOB\text{-}SCHEDULING = \{ \langle J, k, t \rangle \mid \text{The jobs in } J \text{ can be assigned to the } k \text{ workers so all jobs are completed within } t \text{ time} \}$
JOB-SCHEDULING $\in$ NPC

- We will prove that JOB-SCHEDULING is NP-complete.
- To do this, we will show that
  - JOB-SCHEDULING $\in$ NP, and
  - PARTITION $\leq_p$ JOB-SCHEDULING
$$\text{JOB-SCHEDULING} \in \text{NP}$$

- Here is a polynomial-time NTM for JOB-SCHEDULING:

  - \(N = \text{“On input } \langle J, k, t \rangle:\)
    - \textbf{Nondeterministically} guess an assignment of the jobs in \(J\) to the \(k\) workers.
    - \textbf{Deterministically} find the maximum amount of time used by any worker.
    - If it is at most \(t\), accept; otherwise, reject.”
**PARTITION \leq_p JOB-SCHEDULING**

- We now reduce partitioning to job scheduling.
- The reduction is actually straightforward:
  - Given a set of numbers to partition, create one task for each number.
  - Have two workers.
  - See if the workers can complete the tasks in time at most half the total time required to do all jobs.
\( \text{PARTITION} \leq_p \text{JOB-SCHEDULING} \)

\[ \{2, 3, 4, 5, 10\} \]

Total time: 24

12 Time Units
The Story So Far

3SAT

INDSET

3COLOR

EXACT-COVER

SUBSET-SUM

PARTITION

JOB-SCHEDULING
A Historical Note
A Feel for **NP**-Completeness

- We have just seen **NP**-complete problems from
  - Formal logic (3SAT)
  - Graph theory (3-colorability)
  - Set theory (exact cover)
  - Number theory (subset sum / partition)
  - Operations research (job scheduling)

- **You will encounter NP-complete problems in the real world.**
Next Time

• **Approximation Algorithms**
  • Can we *approximate* **NP**-hard problems within polynomial time?

• **P, NP, and Cryptography**
  • How can we use hard problems to our advantage?