Final Exam Study Questions

This is a “core dump” of potential questions for the post-Midterm 2 material on the Final. This should give you a good idea of the types of questions that we might ask on the exam—in particular, there might be a series of True/False or other Multiple Choice questions—but the actual exam questions may or may not appear in this handout. This list intentionally includes a few questions that are too long or difficult for exam conditions.

Don’t forget to review the study problems for Midterms 1 and 2; the final exam is cumulative!

How to Use These Problems

Solving every problem in this handout is not the best way to study for the exam. Memorizing the solutions to every problem in this handout is the absolute worst way to study for the exam.

What we recommend instead is to work on a sample of the problems. Choose one or two problems at random from each section and try to solve them from scratch under exam conditions—by yourself, in a quiet room, with a 30-minute timer, without your notes, without the internet, and if possible, even without your cheat sheet. If you’re comfortable solving a few problems in a particular section, you’re probably ready for that type of problem on the exam. Move on to the next section.

Discussing problems with other people (in your study groups, in the review sessions, in office hours, or on Piazza) and/or looking up old solutions can be extremely helpful, but only after you have (1) made a good-faith effort to solve the problem on your own, and (2) you have either a candidate solution or some idea about where you’re getting stuck.

If you find yourself getting stuck on a particular type of problem, try to figure out why you’re stuck. Do you understand the problem statement? Are you stuck on choosing the right high-level approach? Are you stuck on the technical details? Or are you struggling to express your ideas clearly? (We strongly recommend writing solutions that follow the homework grading rubrics bullet-by-bullet.)

Similarly, if feedback from other people suggests that your solutions to a particular type of problem are incorrect or incomplete, try to figure out what you missed. For NP-hardness proofs: Are you choosing a good problem to reduce from? Are you reducing in the correct direction? Are you designing your reduction with both good instances and bad instances in mind? You’re not trying solve the problem, are you? For undecidability proofs: If you are arguing by reduction, are you reducing in the correct direction? You’re not using pronouns, are you?

Remember that your goal is not merely to “understand” the solution to any particular problem, but to become more comfortable with solving a certain type of problem on your own. Understanding is a trap; aim for mastery. If you can identify specific steps that you find problematic, read more about those steps, focus your practice on those steps, and try to find helpful information about those steps to write on your cheat sheet. Then work on the next problem!
True or False? (All from previous final exams)

For each of the following questions, indicate every correct answer by marking the “Yes” box, and indicate every incorrect answer by marking the “No” box. Assume $P \not= NP$. If there is any other ambiguity or uncertainty about an answer, mark the “No” box. For example:

- $x + y = 5$
- 3SAT can be solved in polynomial time.
- Jeff is not the Queen of England.
- If $P = NP$ then Jeff is the Queen of England.

1. Which of the following are a good English specifications of a recursive function that could possibly be used to compute the edit distance between two strings $A[1..n]$ and $B[1..n]$?

- $Edit(i, j)$ is the answer for $i$ and $j$.
- $Edit(i, j)$ is the edit distance between $A[i]$ and $B[j]$.
- $Edit[i, j] = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ Edit[i - 1, j - 1] & \text{if } A[i] = B[j] \\ \max \left\{ 1 + Edit[i, j - 1], 1 + Edit[i - 1, j], 1 + Edit[i - 1, j - 1] \right\} & \text{otherwise} \end{cases}$
- $Edit[1..n, 1..n]$ stores the edit distances for all prefixes.
- $Edit(i, j)$ is the edit distance between $A[i..n]$ and $B[j..n]$.
- $Edit[i, j]$ is the value stored at row $i$ and column $j$ of the table.
- $Edit(i, j)$ is the edit distance between the last $i$ characters of $A$ and the last $j$ characters of $B$.
- $Edit(i, j)$ is the edit distance when $i$ and $j$ are the current characters in $A$ and $B$.
- $Edit(i, j, k, l)$ is the edit distance between substrings $A[i..j]$ and $B[k..l]$.

[I don’t need an English description; my pseudocode is clear enough!]
2. Which of the following statements are true for every language $L \subseteq \{0, 1\}^*$?

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>L is non-empty.</td>
<td></td>
</tr>
<tr>
<td>L is infinite.</td>
<td></td>
</tr>
<tr>
<td>L contains the empty string $\varepsilon$.</td>
<td></td>
</tr>
<tr>
<td>$L^*$ is infinite.</td>
<td></td>
</tr>
<tr>
<td>$L^*$ is regular.</td>
<td></td>
</tr>
<tr>
<td>L is accepted by some DFA if and only if $L$ is accepted by some NFA.</td>
<td></td>
</tr>
<tr>
<td>L is described by some regular expression if and only if $L$ is rejected by some NFA.</td>
<td></td>
</tr>
<tr>
<td>L is accepted by some DFA with 42 states if and only if $L$ is accepted by some NFA with 42 states.</td>
<td></td>
</tr>
<tr>
<td>If $L$ is decidable, then $L$ is infinite.</td>
<td></td>
</tr>
<tr>
<td>If $L$ is not decidable, then $L$ is infinite.</td>
<td></td>
</tr>
<tr>
<td>If $L$ is not regular, then $L$ is undecidable.</td>
<td></td>
</tr>
<tr>
<td>If $L$ has an infinite fooling set, then $L$ is undecidable.</td>
<td></td>
</tr>
<tr>
<td>If $L$ has a finite fooling set, then $L$ is decidable.</td>
<td></td>
</tr>
<tr>
<td>If $L$ is the union of two regular languages, then its complement $\overline{L}$ is regular.</td>
<td></td>
</tr>
<tr>
<td>If $L$ is the union of two decidable languages, then $L$ is decidable.</td>
<td></td>
</tr>
<tr>
<td>If $L$ is the union of two undecidable languages, then $L$ is undecidable.</td>
<td></td>
</tr>
<tr>
<td>If $L \not\in \mathcal{P}$, then $L$ is not regular.</td>
<td></td>
</tr>
<tr>
<td>$L$ is decidable if and only if its complement $\overline{L}$ is undecidable.</td>
<td></td>
</tr>
<tr>
<td>Both $L$ and its complement $\overline{L}$ are decidable.</td>
<td></td>
</tr>
</tbody>
</table>
3. Which of the following statements are true for at least one language \( L \subseteq \{0, 1\}^* \)?

<table>
<thead>
<tr>
<th></th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>L is non-empty.</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>L is infinite.</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>L contains the empty string ( \varepsilon ).</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>( L^* ) is finite.</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>( L^* ) is not regular.</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>L is not regular but ( L^* ) is regular.</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>L is finite and ( L ) is undecidable.</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>L is decidable but ( L^* ) is not decidable.</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>L is not decidable but ( L^* ) is decidable.</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>L is the union of two decidable languages, but ( L ) is not decidable.</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>L is the union of two undecidable languages, but ( L ) is decidable.</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>L is accepted by an NFA with 374 states, but ( L ) is not accepted by a DFA with 374 states.</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>L is accepted by an DFA with 374 states, but ( L ) is not accepted by a NFA with 374 states.</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>L is regular and ( L \notin \mathcal{P} ).</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>There is a Turing machine that accepts ( L ).</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>There is an algorithm to decide whether an arbitrary given Turing machine accepts ( L ).</td>
<td>No</td>
</tr>
</tbody>
</table>
4. Which of the following languages over the alphabet \{0, 1\} are regular?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Yes</strong></td>
<td><strong>No</strong></td>
</tr>
<tr>
<td>{0^m 1^n</td>
<td>m \geq 0 \text{ and } n \geq 0}</td>
</tr>
<tr>
<td>All strings with the same number of 0s and 1s</td>
<td><strong>Yes</strong></td>
</tr>
<tr>
<td>Binary representations of all positive integers divisible by 17</td>
<td><strong>Yes</strong></td>
</tr>
<tr>
<td>Binary representations of all prime numbers less than 10^{100}</td>
<td><strong>Yes</strong></td>
</tr>
<tr>
<td>{ww</td>
<td>w \text{ is a palindrome}}</td>
</tr>
<tr>
<td>{wxw</td>
<td>w \text{ is a palindrome and } x \in {0, 1}^*}</td>
</tr>
<tr>
<td>{(M)</td>
<td>M \text{ accepts a regular language}}</td>
</tr>
<tr>
<td>{(M)</td>
<td>M \text{ accepts a finite number of non-palindromes}}</td>
</tr>
</tbody>
</table>

5. Which of the following languages/decision problems are decidable?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Yes</strong></td>
<td><strong>No</strong></td>
</tr>
<tr>
<td>\emptyset</td>
<td></td>
</tr>
<tr>
<td>{0^n 1^{2n} 0^n 1^{2n}</td>
<td>n \geq 0}</td>
</tr>
<tr>
<td>{ww</td>
<td>w \text{ is a palindrome}}</td>
</tr>
<tr>
<td>{(M)</td>
<td>M \text{ accepts } (M) \cdot (M)}</td>
</tr>
<tr>
<td>{(M)</td>
<td>M \text{ accepts a finite number of non-palindromes}}</td>
</tr>
<tr>
<td>{(M, w)</td>
<td>M \text{ accepts } ww}</td>
</tr>
<tr>
<td>{(M, w)</td>
<td>M \text{ accepts } ww \text{ after at most }</td>
</tr>
<tr>
<td>Given an NFA (N), is the language (L(N)) infinite?</td>
<td><strong>Yes</strong></td>
</tr>
<tr>
<td>CIRCUIT\text{Sat}</td>
<td><strong>Yes</strong></td>
</tr>
<tr>
<td>Given an undirected graph (G), does (G) contain a Hamiltonian cycle?</td>
<td></td>
</tr>
<tr>
<td>Given two Turing machines (M) and (M'), is there a string (w) that is accepted by both (M) and (M')?</td>
<td><strong>Yes</strong></td>
</tr>
</tbody>
</table>
6. Recall the halting language $\text{HALTONINPUT} = \{\langle M, w \rangle \mid M \text{ halts on input } w\}$. Which of the following statements about its complement $\text{HALTONINPUT} = \Sigma^* \setminus \text{HALTONINPUT}$ are true?

- [ ] Yes  [ ] No  $\text{HALTONINPUT}$ is empty.
- [ ] Yes  [ ] No  $\text{HALTONINPUT}$ is regular.
- [ ] Yes  [ ] No  $\text{HALTONINPUT}$ is infinite.
- [ ] Yes  [ ] No  $\text{HALTONINPUT}$ is decidable.
- [ ] Yes  [ ] No  $\text{HALTONINPUT}$ is recursively enumerable, but not decidable.
- [ ] Yes  [ ] No  $\text{HALTONINPUT}$ is not recursively enumerable.

7. Suppose some language $A \in \{0, 1\}^*$ reduces to another language $B \in \{0, 1\}^*$. Which of the following statements must be true?

- [ ] Yes  [ ] No  A Turing machine that recognizes $A$ can be used to construct a Turing machine that recognizes $B$.
- [ ] Yes  [ ] No  $A$ is decidable.
- [ ] Yes  [ ] No  If $B$ is decidable then $A$ is decidable.
- [ ] Yes  [ ] No  If $A$ is decidable then $B$ is decidable.
- [ ] Yes  [ ] No  If $B$ is NP-hard then $A$ is NP-hard.
- [ ] Yes  [ ] No  If $A$ has no polynomial-time algorithm then neither does $B$. 
8. Suppose there is a \textbf{polynomial-time} reduction from problem A to problem B. Which of the following statements \textit{must} be true?

\begin{tabular}{|c|c|}
\hline
Yes & No \\
\hline
\end{tabular}

\begin{enumerate}
\item Problem B is NP-hard.
\item A polynomial-time algorithm for B can be used to solve A in polynomial time.
\item If B has no polynomial-time algorithm then neither does A.
\item If A is NP-hard and B has a polynomial-time algorithm then P = NP.
\item If B is NP-hard then A is NP-hard.
\item If B is undecidable then A is undecidable.
\end{enumerate}

9. Consider the following pair of languages:

\begin{itemize}
\item \textsc{HamPath} := \{ G \mid G \text{ is an undirected graph with a Hamiltonian path}\}
\item \textsc{Connected} := \{ G \mid G \text{ is a connected undirected graph}\}
\end{itemize}

(For concreteness, assume that in both of these languages, graphs are represented by their adjacency matrices.) Which of the following \textit{must} be true, assuming P \neq NP?

\begin{tabular}{|c|c|}
\hline
Yes & No \\
\hline
\end{tabular}

\begin{enumerate}
\item \textsc{Connected} \in NP
\item \textsc{HamPath} \in NP
\item \textsc{HamPath} is decidable.
\item There is no polynomial-time reduction from \textsc{HamPath} to \textsc{Connected}.
\item There is no polynomial-time reduction from \textsc{Connected} to \textsc{HamPath}.
\end{enumerate}
10. Consider the following pair of languages:

- \( \text{DirHamPath} := \{ G \mid G \text{ is a directed graph with a Hamiltonian path} \} \)
- \( \text{Acyclic} := \{ G \mid G \text{ is a directed acyclic graph} \} \)

(For concreteness, assume that in both of these languages, graphs are represented by their
adjacency matrices.) Which of the following must be true, assuming \( P \neq NP \)?

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Acyclic ( \in ) NP</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Acyclic ( \cap ) DirHamPath ( \in ) P</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DirHamPath is decidable.</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>There is a polynomial-time reduction from DirHamPath to Acyclic.</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>There is a polynomial-time reduction from Acyclic to DirHamPath.</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Suppose we want to prove that the following language is undecidable.

**ALWAYSHALTS** := \{ \langle M \rangle \mid M \text{ halts on every input string} \}

Rocket J. Squirrel suggests a reduction from the standard halting language

**HALTONINPUT** := \{ \langle M, w \rangle \mid M \text{ halts on inputs } w \}.

Specifically, given a Turing machine **DECIDEALWAYSHALTS** that decides **ALWAYSHALTS**, Rocky claims that the following Turing machine **DECIDEHALT** decides **HALTONINPUT**.

**DECIDEHALT**(\langle M, w \rangle):
Encode the following Turing machine \( M' \):

\[
BULLWINKLE(x):
\]
if \( M \) accepts \( w \), reject
if \( M \) rejects \( w \), accept

return **DECIDEALWAYSHALTS**(\langle BULLWINKLE \rangle)

Which of the following statements is true for all inputs \langle M \rangle \#w?

<table>
<thead>
<tr>
<th></th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>If ( M ) accepts ( w ), then ( M' ) halts on every input string.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>If ( M ) rejects ( w ), then ( M' ) halts on every input string.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>If ( M ) diverges on ( w ), then ( M' ) halts on every input string.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>If ( M ) accepts ( w ), then <strong>DECIDEALWAYSHALTS</strong> accepts \langle BULLWINKLE \rangle.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>If ( M ) rejects ( w ), then <strong>DECIDEHALT</strong> rejects \langle M, w \rangle.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>If ( M ) diverges on ( w ), then <strong>DECIDEALWAYSHALTS</strong> diverges on \langle BULLWINKLE \rangle.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td><strong>DECIDEHALT</strong> decides <strong>HALTONINPUT</strong>. (That is, Rocky’s reduction is correct.)</td>
<td></td>
</tr>
</tbody>
</table>
12. Suppose we want to prove that the following language is undecidable.

\[ \text{Muggle} := \{ \langle M \rangle \mid M \text{ accepts SCIENCE but rejects MAGIC} \} \]

Professor Potter, your instructor in Defense Against Models of Computation and Other Dark Arts, suggests a reduction from the standard halting language

\[ \text{HALTOnInput} := \{ \langle M, w \rangle \mid M \text{ halts on inputs } w \}. \]

Specifically, suppose there is a Turing machine \text{DetectoMuggletum} that decides \text{Muggle}. Professor Potter claims that the following algorithm decides \text{HALTOnInput}.

\[
\text{DecideHalt}(\langle M, w \rangle):
\]

\[
\text{Encode the following Turing machine:}
\]

\[
\text{RubberDuck}(x):
\]

\[
\text{run } M \text{ on input } w
\]

\[
\text{if } x = \text{MAGIC}
\]

\[
\text{return FALSE}
\]

\[
\text{else}
\]

\[
\text{return TRUE}
\]

\[
\text{return DetectoMuggletum(\langle \text{RubberDuck} \rangle)}
\]

Which of the following statements is true for all inputs \langle M \rangle \#w?

- If \( M \) accepts \( w \), then \text{RubberDuck} accepts \text{SCIENCE}.
- If \( M \) accepts \( w \), then \text{RubberDuck} accepts \text{CHOCOLATE}.
- If \( M \) rejects \( w \), then \text{RubberDuck} rejects \text{MAGIC}.
- If \( M \) rejects \( w \), then \text{RubberDuck} halts on every input string.
- If \( M \) diverges on \( w \), then \text{RubberDuck} rejects every input string.
- If \( M \) accepts \( w \), then \text{DetectoMuggletum} accepts \langle \text{RubberDuck} \rangle.
- If \( M \) rejects \( w \), then \text{DecideHalt} rejects \langle \langle M \rangle, w \rangle.
- If \( M \) diverges on \( w \), then \text{DecideHalt} rejects \langle \langle M \rangle, w \rangle.
- \text{DecideHalt} decides the language \text{HALTOnInput}. (That is, Professor Potter’s reduction is actually correct.)
- \text{DecideHalt} actually runs (or simulates) \text{RubberDuck}.
- \text{Muggle} is decidable.
NP-hardness

1. A boolean formula is in *disjunctive normal form* (or DNF) if it consists of a *disjunction* (Or) or several *terms*, each of which is the conjunction (And) of one or more literals. For example, the formula

\[(\overline{x} \land y \land \overline{z}) \lor (y \land z) \lor (x \land \overline{y} \land \overline{z})\]

is in disjunctive normal form. DNF-Sat asks, given a boolean formula in disjunctive normal form, whether that formula is satisfiable.

(a) Describe a polynomial-time algorithm to solve DNF-Sat.

(b) What is the error in the following argument that P=NP?

_Suppose we are given a boolean formula in conjunctive normal form with at most three literals per clause, and we want to know if it is satisfiable. We can use the distributive law to construct an equivalent formula in disjunctive normal form. For example,

\[(x \lor y \lor \overline{z}) \land (\overline{x} \lor \overline{y}) \iff (x \land \overline{y}) \lor (y \land \overline{x}) \lor (\overline{z} \land \overline{y})\]

Now we can use the algorithm from part (a) to determine, in polynomial time, whether the resulting DNF formula is satisfiable. We have just solved 3Sat in polynomial time. Since 3Sat is NP-hard, we must conclude that P=NP!_

2. A *relaxed 3-coloring* of a graph \(G\) assigns each vertex of \(G\) one of three colors (for example, red, green, and blue), such that at most one edge in \(G\) has both endpoints the same color.

(a) Give an example of a graph that has a relaxed 3-coloring, but does not have a proper 3-coloring (where every edge has endpoints of different colors).

(b) **Prove** that it is NP-hard to determine whether a given graph has a relaxed 3-coloring.

3. An *ultra-Hamiltonian cycle* in \(G\) is a closed walk \(C\) that visits every vertex of \(G\) exactly once, except for at most one vertex that \(C\) visits more than once.

(a) Give an example of a graph that contains a ultra-Hamiltonian cycle, but does not contain a Hamiltonian cycle (which visits every vertex exactly once).

(b) **Prove** that it is NP-hard to determine whether a given graph contains a ultra-Hamiltonian cycle.

4. An *infra-Hamiltonian cycle* in \(G\) is a closed walk \(C\) that visits every vertex of \(G\) exactly once, except for at most one vertex that \(C\) does not visit at all.

(a) Give an example of a graph that contains a infra-Hamiltonian cycle, but does not contain a Hamiltonian cycle (which visits every vertex exactly once).

(b) **Prove** that it is NP-hard to determine whether a given graph contains a infra-Hamiltonian cycle.

5. A *quasi-satisfying assignment* for a 3CNF boolean formula \(\Phi\) is an assignment of truth values to the variables such that at most one clause in \(\Phi\) does not contain a true literal. **Prove** that it is NP-hard to determine whether a given 3CNF boolean formula has a quasi-satisfying assignment.
6. A subset $S$ of vertices in an undirected graph $G$ is **half-independent** if each vertex in $S$ is adjacent to *at most one* other vertex in $S$. Prove that finding the size of the largest half-independent set of vertices in a given undirected graph is NP-hard.

7. A subset $S$ of vertices in an undirected graph $G$ is **sort-of-independent** if if each vertex in $S$ is adjacent to *at most 374* other vertices in $S$. Prove that finding the size of the largest sort-of-independent set of vertices in a given undirected graph is NP-hard.

8. A subset $S$ of vertices in an undirected graph $G$ is **almost independent** if at most *374* edges in $G$ have both endpoints in $S$. Prove that finding the size of the largest almost-independent set of vertices in a given undirected graph is NP-hard.

9. A **kite** of size $k$ is a complete graph (clique) on $k$ vertices plus a tail of $k$ vertices. Prove that it is NP-hard to determine, if given an undirected graph $G$ and integer $k$, whether $G$ contains a kite of size $k$ as a subgraph.

![A kite of size 4.](image-url)

10. Let $G$ be an undirected graph with weighted edges. A **heavy Hamiltonian cycle** is a cycle $C$ that passes through each vertex of $G$ exactly once, such that the total weight of the edges in $C$ is more than half of the total weight of all edges in $G$. Prove that deciding whether a graph has a heavy Hamiltonian cycle is NP-hard.

![A heavy Hamiltonian cycle. The cycle has total weight 34; the graph has total weight 67.](image-url)

11. An undirected graph is an **eight-graph** if it has an odd number of nodes, say $2n - 1$, and consists of two cycles $C_1$ and $C_2$ on $n$ vertices each and $C_1$ and $C_2$ share exactly one vertex. Prove that is NP-hard to determine, if given an undirected graph $G$ and an integer $k$, whether $G$ has a subgraph which is an eight-graph on $2k - 1$ nodes.

![An eight-graph on 7 vertices.](image-url)
12. (a) A tonian path in a graph $G$ is a path that goes through at least half of the vertices of $G$. Show that determining whether a graph has a tonian path is NP-hard.

(b) A tonian cycle in a graph $G$ is a cycle that goes through at least half of the vertices of $G$. Show that determining whether a graph has a tonian cycle is NP-hard. [Hint: Use part (a). Or not.]

13. Prove that the following variants of SAT is NP-hard. [Hint: Describe reductions from $3$SAT.]

(a) Given a boolean formula $\Phi$ in conjunctive normal form, where each variable appears in at most three clauses, determine whether $F$ has a satisfying assignment. [Hint: First consider the variant where each variable appears in at most five clauses.]

(b) Given a boolean formula $\Phi$ in conjunctive normal form and given one satisfying assignment for $\Phi$, determine whether $\Phi$ has at least one other satisfying assignment.

14. Jerry Springer and Maury Povich have decided not to compete with each other over scheduling guests during the next talk-show season. There is only one set of Weird People who either host would consider having on their show. The hosts want to divide the Weird People into two (disjoint) groups: those to appear on Jerry’s show, and those to appear on Maury’s show. (Neither wants to “recycle” a guest that appeared on the other’s show.)

Both Jerry and Maury have preferences about which Weird People they are particularly interested in. For example, Jerry wants to be sure to get at least one person who fits the category “had extra-terrestrial affair”. Thus, on his list of preferences, he writes “$w_1$ or $w_3$ or $w_{45}$”, since weird people numbered 1, 3, and 45 are the only ones who fit that description. Jerry has other preferences as well, so he lists those also. Similarly, Maury might like to guarantee that his show includes at least one guest who confesses to “really enjoying eating toothpaste”. Each potential guest may fall into any number of different categories, such as the person who enjoys eating toothpaste more than the extra-terrestrial affair they had.

Jerry and Maury each prepare a list reflecting all of their preferences. Each list contains a collection of statements of the form “($w_i$ or $w_j$ or $w_k$)”. Your task is to prove that it is NP-hard to find an assignment of weird guests to the two shows that satisfies all of Jerry’s preferences and all of Maury’s preferences.

(a) The problem $\text{NoMixedClauses3Sat}$ is the special case of $3$Sat where the input formula cannot contain a clause with both a negated variable and a non-negated variable. Prove that $\text{NoMixedClauses3Sat}$ is NP-hard. [Hint: Reduce from the standard $3$Sat problem.]

(b) Describe a polynomial-time reduction from $\text{NoMixedClauses3Sat}$ to $3$Sat.

15. Prove that the following problem (which we call MATCH) is NP-hard. The input is a finite set $S$ of strings, all of the same length $n$, over the alphabet $\{0, 1, 2\}$. The problem is to determine whether there is a string $w \in \{0, 1\}^n$ such that for every string $s \in S$, the strings $s$ and $w$ have the same symbol in at least one position.

For example, given the set $S = \{01220, 21110, 21120, 00211, 11101\}$, the correct output is $\text{True}$, because the string $w = 01001$ matches the first three strings of $S$ in the second position, and matches the last two strings of $S$ in the last position. On the other hand, given the set $S = \{00, 11, 01, 10\}$, the correct output is $\text{False}$.

[Hint: Describe a reduction from SAT (or $3$Sat)]
16. To celebrate the end of the semester, Professor Jarling wants to treat himself to an ice-cream cone, at the Polynomial House of Flavors. For a fixed price, he can build a cone with as many scoops as he'd like. Because he has good balance (and because we want this problem to work out), assume that he can balance any number of scoops on top of the cone without it tipping over. He plans to eat the ice cream one scoop at a time, from top to bottom, and doesn’t want more than one scoop of any flavor.

However, he realizes that eating a scoop of bubblegum ice cream immediately after the scoop of potatoes-and-gravy ice cream would be unpalatable; these two flavors clearly should not be placed next to each other in the stack. He has other similar constraints; certain pairs of flavors cannot be adjacent in the stack.

He’d like to get as much ice cream as he can for the one fee by building the tallest cone possible that meets his flavor-incompatibility constraints. Prove that this problem is NP-hard.

17. Prove that the following problems are NP-hard.

(a) Given an undirected graph $G$, does $G$ contain a simple path that visits all but 17 vertices?
(b) Given an undirected graph $G$, does $G$ have a spanning tree in which every node has degree at most 23?
(c) Given an undirected graph $G$, does $G$ have a spanning tree with at most 42 leaves?

18. Prove that the following problems are NP-hard.

(a) Given an undirected graph $G$, is it possible to color the vertices of $G$ with three different colors, so that at most 31337 edges have both endpoints the same color?
(b) Given an undirected graph $G$, is it possible to color the vertices of $G$ with three different colors, so that each vertex has at most 8675309 neighbors with the same color?

19. At the end of the semester, the instructors need to solve the following ExamDesign problem. They have a list of problems, and they know for each problem which students will really enjoy that problem. They need to choose a subset of problems for the exam such that for each student in the class, the exam includes at least one question that student will really enjoy. On the other hand, they do not want to spend the entire summer grading an exam with dozens of questions, so the exam must also contain as few questions as possible. Prove that the ExamDesign problem is NP-hard.

20. Which of the following results would resolve the P vs. NP question? Justify each answer with a short sentence or two.

(a) The construction of a polynomial time algorithm for some problem in NP.
(b) A polynomial-time reduction from 3Sat to the language $\{0^n1^n \mid n \geq 0\}$.
(c) A polynomial-time reduction from $\{0^n1^n \mid n \geq 0\}$ to 3Sat.
(d) A polynomial-time reduction from 3Color to MinVertexCover.
(e) The construction of a nondeterministic Turing machine that cannot be simulated by any deterministic Turing machine with the same running time.
Some useful NP-hard problems. You are welcome to use any of these in your own NP-hardness proofs, except of course for the specific problem you are trying to prove NP-hard.

**CircuitSat:** Given a boolean circuit, are there any input values that make the circuit output True?

**3Sat:** Given a boolean formula in conjunctive normal form, with exactly three distinct literals per clause, does the formula have a satisfying assignment?

**MaxIndependentSet:** Given an undirected graph $G$, what is the size of the largest subset of vertices in $G$ that have no edges among them?

**MaxClique:** Given an undirected graph $G$, what is the size of the largest complete subgraph of $G$?

**MinVertexCover:** Given an undirected graph $G$, what is the size of the smallest subset of vertices that touch every edge in $G$?

**MinSetCover:** Given a collection of subsets $S_1, S_2, \ldots, S_m$ of a set $S$, what is the size of the smallest subcollection whose union is $S$?

**3Color:** Given an undirected graph $G$, can its vertices be colored with three colors, so that every edge touches vertices with two different colors?

**HamiltonianPath:** Given graph $G$ (either directed or undirected), is there a path in $G$ that visits every vertex exactly once?

**HamiltonianCycle:** Given a graph $G$ (either directed or undirected), is there a cycle in $G$ that visits every vertex exactly once?

**TravelingSalesman:** Given a graph $G$ (either directed or undirected) with weighted edges, what is the minimum total weight of any Hamiltonian path/cycle in $G$?

**LongestPath:** Given a graph $G$ (either directed or undirected, possibly with weighted edges), what is the length of the longest simple path in $G$?
Turing Machines and Undecidability

For each of the following languages, either sketch an algorithm to decide that language or prove that the language is undecidable, using a diagonalization argument, a reduction argument, closure properties, or some combination of the above. Recall that \( w^R \) denotes the reversal of string \( w \).

1. \( \emptyset \)
2. \( \{ \theta^n 1^n 2^n \mid n \geq 0 \} \)
3. \( \{ A \in \{ 0, 1 \}^{n \times n} \mid n \geq 0 \text{ and } A \text{ is the adjacency matrix of a dag with } n \text{ vertices} \} \)
4. \( \{ A \in \{ 0, 1 \}^{n \times n} \mid n \geq 0 \text{ and } A \text{ is the adjacency matrix of a 3-colorable graph with } n \text{ vertices} \} \)
5. \( \{ \langle M \rangle \mid M \text{ accepts } \langle M \rangle^R \} \)
6. \( \{ \langle M \rangle \mid M \text{ accepts } \langle M \rangle^R \} \cap \{ \langle M \rangle \mid M \text{ rejects } \langle M \rangle^R \} \)
7. \( \{ \langle M, w \rangle \mid M \text{ accepts } w w^R \} \)
8. \( \{ \langle M \rangle \mid M \text{ accepts at least one palindrome} \} \)
9. \( \Sigma^* \setminus \{ \langle M \rangle \mid M \text{ accepts at least one palindrome} \} \)
10. \( \{ \langle M \rangle \mid M \text{ rejects at least one palindrome} \} \)
11. \( \{ \langle M \rangle \mid M \text{ accepts exactly one string of length } \ell, \text{ for each integer } \ell \geq 0 \} \)
12. \( \{ \langle M \rangle \mid L(M) \text{ has an infinite fooling set} \} \)

**Some useful undecidable problems.** You are welcome to use any of these in your own undecidability proofs, except of course for the specific problem you are trying to prove undecidable.

\[
L_u := \{ \langle M, w \rangle \mid M \text{ accepts on } w \}
\]
\[
\text{Halt} := \{ \langle M \rangle \mid M \text{ halts on blank input} \}
\]
\[
\text{HaltOnInput} := \{ \langle M, w \rangle \mid M \text{ halts on } w \}
\]