1. (a) A subset $S$ of vertices in an undirected graph $G$ is \textbf{half-independent} if each vertex in $S$ is adjacent to \textit{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.

\textbf{Solution:} We prove this problem NP-hard by reduction from the maximum independent set problem.

Let $G = (V, E)$ be an arbitrary undirected graph, and let $n = |V|$. We construct a graph $H$ by attaching a new vertex with degree 1 to every vertex in $G$. More formally, let $V' = \{v' \mid v \in V\}$ be a set of $n$ new vertices; we call $v'$ the \textit{clone} of $v$. Then the vertices of $H$ are $V \cup V'$ and the edges of $H$ are $E \cup \{vv' \mid v \in V\}$.

Now I claim, for any integer $k \geq 0$, that $G$ has an independent set of size $k$ if and only if $H$ has a half-independent set of size $n + k$.

\[
\implies\text{For any independent set } S \text{ in } G, \text{ the set } V' \cup S \text{ is a half-independent set in } H \text{ moreover, } |S \cup V'| = n + |S|.
\]

\[
\Leftarrow\text{Let } S \text{ be an arbitrary half-independent set in } H \text{ such that } |S| \geq n. \text{ We define a new half-independent set } S' \text{ as follows:}
\]

\[
S' = (S \cup V') \setminus \{v \in S \cap V \mid u \in S \cap V \text{ for some } uv \in E\}
\]

That is, we construct $S'$ by adding every clone to $S$, and then removing any non-clone in $S$ with a non-clone neighbor in $S$. If $S$ contains both an original vertex $v$ and its clone $v'$, then $S$ contains no other neighbor of $v$, so $S'$ also contains both $v$ and $v'$. Otherwise, $S$ contains at most one of $v$ and $v'$, and $S'$ contains $v'$ (and possibly $v$). It follows that $|S'| \geq |S|$.

The set $S' \setminus V'$ is an independent set in $G$. Moreover, $|S' \setminus V'| = |S'| - n \geq |S| - n$. Thus, if $|S| = n + k$, then $|S' \setminus V'| \geq k$. If necessary, we can discard vertices from $S' \setminus V'$ to get an independent set of size exactly $k$.

Thus, to compute the size of the largest independent set in $G$, we can compute the size of largest half-independent set in $H$ and subtract $n$. We can construct $H$ from $G$ by brute force in polynomial time. \hfill \blacksquare

\textbf{Rubric:} 5 points, standard reduction rubric (scaled)
(b) 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.

**Solution:** Again, we prove this problem NP-hard by reduction from the maximum independent set problem.

Let $G = (V, E)$ be an arbitrary undirected graph, and let $n = |V|$. We construct a graph $H$ by attaching a new clique of 373 vertices to every vertex in $G$. More formally, let $W = V \times \{1, 2, \ldots, 373\}$, and let us write $v_i$ to denote the pair $(v, i)$. Then $H = (V', E')$ where

$$V' = V \cup W$$

$$E' = E \cup \{vv_i \mid v \in V \text{ and } i \in \{1, 2, \ldots, 373\}\}$$

$$\cup \{v_i v_j \mid v \in V \text{ and } i, j \in \{1, 2, \ldots, 373\}\}$$

We call vertices in $V$ original vertices, and we call each vertex $v_i$ a clone of $v$.

The rest of the proof is nearly identical to part (a). For any integer $k \geq 0$, that $G$ has an independent set of size $k$ if and only if $H$ has a sort-of-independent set of size $373n + k$.

$\Rightarrow$ For any independent set $S$ in $G$, the set $S \cup W$ is a half-independent set in $H$; moreover, $|S \cup W| = 373n + |S|$.

$\Leftarrow$ Let $S$ be an arbitrary sort-of-independent set in $H$ such that $|S| \geq n$. We define a new sort-of-independent set $S'$ as follows:

$$S' = (S \cup W) \setminus \{v \in S \cap V \mid u \in S \cap V \text{ for some } uv \in E\}$$

That is, we construct $S'$ by adding every clone to $S$, and then removing any non-clone in $S$ with a non-clone neighbor in $S$. If $S$ contains both an original vertex $v$ and all its clones $v_i$, then $S$ contains no other neighbor of $v$, so $S'$ also contains both $v$ and all its clones. Otherwise, $S$ contains at most 373 vertices among $v$ and its clones, and $S'$ contains all 373 cones of $v$ (and possibly $v$ itself). It follows that $|S'| \geq |S|$.

The set $S' \setminus V'$ is an independent set in $G$. Moreover, $|S' \setminus V'| = |S'| - 373n \geq |S| - 373n$. Thus, if $|S| = 373n + k$, then $|S' \setminus V'| \geq k$. If necessary, we can discard vertices from $S' \setminus V'$ to get an independent set of size exactly $k$.

Thus, to compute the size of the largest independent set in $G$, we can compute the size of largest sort-of-independent set in $H$ and subtract $373n$. We can construct $H$ from $G$ by brute force in polynomial time. ■

**Rubric:** 5 points, standard reduction rubric (scaled)
2. Fix an alphabet $\Sigma = \{0, 1\}$. Prove that the following problems are NP-hard.

(a) Given a regular expression $R$ over the alphabet $\Sigma$, is $L(R) \neq \Sigma^*$?

**Solution:** We describe a polynomial-time reduction from 3Sat. Let $\Phi$ be an arbitrary 3CNF boolean formula. Let $n$ be the number of variables in $\Phi$ and let $k$ be the number of clauses. We construct a regular expression $R$ of length $O(nk)$ as follows.

Let $\Phi = C_1 \land C_2 \land \cdots \land C_k$, where each $C_k$ is a clause. For each clause $C_i$ and each variable $x_j$, we define a regular expression $r_{ij}$ as follows:

- If $C_i$ contains the variable $x_j$, then $r_{ij} = 0$.
- If $C_i$ contains the negated variable $\bar{x}_j$, then $r_{ij} = 1$.
- Otherwise, $r_{ij} = (0 + 1)$.

For each index $i$, let $R_i = r_{i1}r_{i2}\cdots r_{in}$. The regular expression $R_i$ encodes all assignments to the $n$ variables that do not satisfy $C_i$. For example, if $n = 8$, we would transform the clause $(x_3 + \bar{x}_7 + \bar{x}_4)$ into the regular expression $$(0 + 1)(0 + 1)01(0 + 1)1(0 + 1)$$

Let $R = R_1 + R_2 + \cdots + R_k$. The regular expression $R$ encodes all assignments to the $n$ variables that do not satisfy the formula $\Phi$. In particular, $\Phi$ is satisfiable if and only if $L(R) \neq (0 + 1)^n$.

Finally, let $R_<$ be a regular expression for the set of all strings of length smaller than $n$, and let $R_>$ be a regular expression for the set of all strings of length larger than $n$. For example:

$$R_\lt = (0 + 1 + \varepsilon)(0 + 1 + \varepsilon)\cdots (0 + 1 + \varepsilon)$$

$$R_\gt = (0 + 1)^n(0 + 1)(0 + 1)\cdots (0 + 1)$$

Then $\Phi$ is satisfiable if and only if $L(R_\lt + R + R_\gt) \neq (0 + 1)^n$. The final regular expression $R_\lt + R + R_\gt$ has length $O(n^2 + nk)$, and it can be constructed from $\Phi$ in $O(n^2 + nk)$ time by brute force.

---

**Solution:** We describe a polynomial-time reduction from 4COLOR. Let $G = (V, E)$ be an arbitrary graph; arbitrarily index the vertices as $V = \{1, 2, \ldots, n\}$. We construct a regular expression of length $O(n^3)$ whose language is not $\Sigma^*$ if and only if $G$ is 4-colorable, as follows.

Intuitively, we represent each possible 4-coloring of $G$ as a string of length $2n$, where each pair of bits represents the color of one vertex. For each edge $ij$, 

---
where without loss of generality $i < j$, let $R_{ij}$ be the regular expression

\[
(\emptyset + 1)^{2(i-1)} \emptyset (\emptyset + 1)^{2(j-i-1)} \emptyset \emptyset (\emptyset + 1)^{2(n-j)} + (\emptyset + 1)^{2(i-1)} \emptyset (\emptyset + 1)^{2(j-i-1)} \emptyset (\emptyset + 1)^{2(n-j)} + (\emptyset + 1)^{2(i-1)} \emptyset (\emptyset + 1)^{2(j-i-1)} \emptyset (\emptyset + 1)^{2(n-j)} + (\emptyset + 1)^{2(i-1)} \emptyset (\emptyset + 1)^{2(j-i-1)} \emptyset (\emptyset + 1)^{2(n-j)},
\]

where $A^k$ is shorthand for the concatenation of $k$ copies of $A$. $R_{ij}$ encodes the set of all 4-colorings of $G$ in which vertices $i$ and $j$ have the same color. Each expression $R_{ij}$ has length $O(n)$.

Now let $R$ be the sum of the expressions $R_{ij}$, over all edges $ij$. Then $L(R)$ is the set of all strings encoding bad 4-colorings of $G$. In particular, $G$ is 4-colorable if and only if $L(R) \neq (\emptyset + 1)^{2n}$.

Finally, let $R_<$ be a regular expression for the set of all strings of length smaller than $2n$, and let $R_>$ be a regular expression for the set of all strings of length larger than $2n$. For example,

\[
R_< = (\emptyset + 1 + \varepsilon)^{2n-1}
\]
\[
R_> = (\emptyset + 1)^*(\emptyset + 1)^{2n},
\]

where again $A^k$ is shorthand for the concatenation of $k$ copies of $S$.) Then $G$ is 4-colorable if and only if $L(R_< + R + R_>) \neq (\emptyset + 1)^*$. The final regular expression $R_< + R + R_>$ has length $O(n^3)$, and it can be constructed from $\Phi$ in $O(n^3)$ time by brute force.

\[\text{Rubric: 5 points: standard poly-time reduction rubric (scaled). These are not the only correct solutions.}\]

(b) Given an NFA $M$ over the alphabet $\Sigma$, is $L(M) \neq \Sigma^*$?

**Solution:** We can reduce from the problem in part (a) using Thompson’s algorithm, which converts any regular expressions into equivalent NFA in polynomial time.

\[\text{Rubric: 5 points: standard poly-time reduction rubric (scaled). Yes, this is enough for full credit.}\]
3. Let $\langle M \rangle$ denote the encoding of a Turing machine $M$ (or if you prefer, the Python source code for the executable code $M$). Recall that $x \cdot y$ denotes the concatenation of strings $x$ and $y$. Prove that the following language is undecidable.

$$\text{SELFSELFACCEPT} := \{ \langle M \rangle \mid M \text{ accepts the string } \langle M \rangle \cdot \langle M \rangle \}$$

Note that Rice’s theorem does not apply to this language.

**Solution (diagonalization):** Suppose to the contrary that there is a Turing machine $SSA$ that decides $\text{SELFSELFACCEPT}$. For any Turing machine $M$, we have

$$SSA \text{ accepts } \langle M \rangle \iff M \text{ accepts } \langle M \rangle \langle M \rangle.$$  

Let $\overline{SSA}$ be the Turing machine obtained from $SSA$ by swapping its accept and reject states. For any Turing machine $M$, we have

$$\overline{SSA} \text{ rejects } \langle M \rangle \iff M \text{ accepts } \langle M \rangle \langle M \rangle.$$ 

Finally, let $SSA^*$ be the Turing machine that deletes the second half of its input string and then passes control to $SSA$. For any Turing machine $M$, we have

$$SSA^* \text{ rejects } \langle M \rangle \langle M \rangle \iff M \text{ accepts } \langle M \rangle \langle M \rangle.$$  

In particular, if we set $M = SSA^*$, we have

$$SSA^* \text{ rejects } \langle SSA^* \rangle \langle SSA^* \rangle \iff SSA^* \text{ accepts } \langle SSA^* \rangle \langle SSA^* \rangle.$$ 

We have a contradiction; $SSA$ must not exist.  

**Rubric:** Standard diagonalization rubric.

**Solution (reduction from HALT):** For the sake of argument, suppose there is an algorithm $\text{DECIDESELFSELFACCEPT}$ that correctly decides the language $\text{SELFSELFACCEPT}$. Then we can solve the halting problem as follows:

$$\text{DECIDEHALT}(\langle M, w \rangle):$$  

Encode the following Turing machine $M'$:

$$M'(x):$$  

run $M$ on input $w$  

return True

return $\text{DECIDESELFSELFACCEPT}(\langle M' \rangle)$

We prove this reduction correct as follows:

$$\iff$$ Suppose $M$ halts on input $w$. Then $M'$ accepts every input string $x$. In particular, $M'$ accepts the string $\langle M \rangle \langle M \rangle$. So $\text{DECIDESELFSELFACCEPT}$ must accept the encoding $\langle M' \rangle$. We conclude that $\text{DECIDEHALT}$ correctly accepts the encoding $\langle M, w \rangle$. 


Suppose $M$ does not halt on input $w$. Then $M'$ diverges on every input string $x$. In particular, $M'$ does not accept the string $\langle M \rangle \langle M \rangle$. So $\text{DE} \text{CIDESELFSELFAC} \text{CEPT}$ must reject the encoding $\langle M' \rangle$. We conclude that $\text{DE} \text{CIDEHALT}$ correctly rejects the encoding $\langle M, w \rangle$.

In both cases, $\text{DE} \text{CIDEHALT}$ is correct. But that's impossible, because $\text{HALT}$ is undecidable. We conclude that the algorithm $\text{DE} \text{CIDESELFSELFAC} \text{CEPT}$ does not exist. ■

Rubric: Standard undecidability reduction rubric. This is not the only correct reduction.

<table>
<thead>
<tr>
<th>Standard rubrics for undecidability proofs. For problems out of 10 points:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Diagonalization:</strong></td>
</tr>
<tr>
<td>+ 4 for correct wrapper Turing machine</td>
</tr>
<tr>
<td>+ 6 for self-contradiction proof (= 3 for $\Leftarrow$ + 3 for $\Rightarrow$)</td>
</tr>
<tr>
<td><strong>Reduction:</strong></td>
</tr>
<tr>
<td>+ 4 for correct reduction</td>
</tr>
<tr>
<td>+ 3 for &quot;if&quot; proof</td>
</tr>
<tr>
<td>+ 3 for &quot;only if&quot; proof</td>
</tr>
<tr>
<td><strong>Rice’s Theorem:</strong></td>
</tr>
<tr>
<td>+ 4 for positive Turing machine</td>
</tr>
<tr>
<td>+ 4 for negative Turing machine</td>
</tr>
<tr>
<td>+ 2 for other details (including using the correct variant of Rice’s Theorem)</td>
</tr>
</tbody>
</table>