1. (a) 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.

**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 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 $n + k$.

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

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

**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

$$V' = V \cup W$$

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

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

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 373$n$. 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:

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

$$R_> = (0 + 1)^n(0 + 1)(0 + 1)\cdots (0 + 1)^{n+1}$$

Then $\Phi$ is satisfiable if and only if $L(R_<= R_+ R_>) \neq (0 + 1)^n$. The final regular expression $R_<= R_+ R_>$ 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

$$
(0 + 1)^{2(i−1)} \cdot 0\cdot (0 + 1)^{2(j−i−1)} \cdot 0\cdot (0 + 1)^{2(n−j)}
+ (0 + 1)^{2(i−1)} \cdot 1\cdot (0 + 1)^{2(j−i−1)} \cdot 0\cdot (0 + 1)^{2(n−j)}
+ (0 + 1)^{2(i−1)} \cdot 10\cdot (0 + 1)^{2(j−i−1)} \cdot 10\cdot (0 + 1)^{2(n−j)}
+ (0 + 1)^{2(i−1)} \cdot 11\cdot (0 + 1)^{2(j−i−1)} \cdot 11\cdot (0 + 1)^{2(n−j)},
$$

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 $L(R) \neq (0 + 1)^{2n}$.

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,

$$
R_< = (0 + 1 + \epsilon)^{2n−1},
R_> = (0 + 1)^*(0 + 1)^{2n+1},
$$

where again $A^k$ is shorthand for the concatenation of $k$ copies of $S$.) Then $G$ is 4-colorable if and only if $L(R_+ + R + R_>) \neq (0 + 1)^*$. 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.  

---

(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 in equivalent NFA in polynomial time.

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

\[
\text{SSA} \text{ accepts } \langle M \rangle \iff M \text{ accepts } \langle M \rangle \langle M \rangle.
\]

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

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

Finally, let \( \text{SSA}^* \) be the Turing machine that deletes the second half of its input string and then passes control to \( \overline{\text{SSA}} \). For any Turing machine \( M \), we have

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

In particular, if we set \( M = \text{SSA}^* \), we have

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

We have a contradiction; \( \text{SSA} \) must not exist.

\[
\text{Rubric: Standard diagonalization rubric.}
\]

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

\[
\text{DECI sesame HALT}(\langle M, w \rangle): \quad \text{Encode the following Turing machine } M':
\]

\[
M'(x): \quad \text{run } M \text{ on input } w \quad \text{return } \text{True}
\]

\[
\text{return DECI sesame SELFSELFACCEPT}(\langle M' \rangle)
\]

We prove this reduction correct as follows:

\[
\implies \text{Suppose } M \text{ halts on input } w. \text{ Then } M' \text{ accepts every input string } x. \text{ In particular, } M' \text{ accepts the string } \langle M \rangle \langle M \rangle. \text{ So } \text{DECI sesame SELFSELFACCEPT} \text{ must accept the encoding } \langle M' \rangle. \text{ We conclude that } \text{DECI sesame HALT} \text{ correctly accepts the encoding } \langle M, w \rangle.
\]
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 $\text{DECODESELFSELFACCEPT}$ must reject the encoding $\langle M' \rangle$. We conclude that $\text{DECEDEHALT}$ correctly rejects the encoding $\langle M, w \rangle$.

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

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

Standard rubrics for undecidability proofs. For problems out of 10 points:

**Diagonalization:**
- + 4 for correct wrapper Turing machine
- + 6 for self-contradiction proof (= 3 for $\Leftarrow$ + 3 for $\Rightarrow$)

**Reduction:**
- + 4 for correct reduction
- + 3 for "if" proof
- + 3 for "only if" proof

**Rice’s Theorem:**
- + 4 for positive Turing machine
- + 4 for negative Turing machine
- + 2 for other details (including using the correct variant of Rice’s Theorem)