The following are a set of reduction problems in approximate order of difficulty.

1. Consider the two problems:

   - FACTOR: Given a number $n = pq$, the product of two distinct primes p and q, find p and q.
   - EULERTOTIENT: Given a number $n = pq$, where p and q are unknown primes, find $\phi(n) = (p-1)(q-1)$, the number of numbers less than n relatively prime to n.

   Reduce FACTOR to EULERTOTIENT.

   Solution:

   Assume we have a black-box algorithm EULERTOTIENT(n) that returns $\phi(n) = (p-1)(q-1)$. To solve FACTOR(n), we find EULERTOTIENT(n) $= \phi(n) = (p-1)(q-1) = n+1-(p+q)$. We then solve the system of equations:

   $$\begin{aligned} p+q &= n+1-\phi(n) \\ p-q &= \sqrt{p^2-2pq+q^2} = \sqrt{(p+q)^2-4n} \end{aligned}$$

   to find p and q.

2. Consider the following two problems:

   - FEEDBACKNODESET: Given a directed graph $G = (V, E)$, find a set of vertices $U \subseteq V$ of minimal size such that if U is removed from V, G has no cycles.
   - VERTEXCOVER: Given an undirected graph $G = (V, E)$, find a minimum set of vertices $U \subseteq V$ such that for all $(u, v) \in E$, either $u \in U$ or $v \in U$ or both.

   Reduce VERTEXCOVER to FEEDBACKNODESET.

   Solution: Given an undirected graph $G = (V, E)$ and an algorithm for solving FEEDBACKNODESET, create a graph $G' = (V', E')$ with $V' = V$. For each undirected edge $(u, v) \in E$, create directed edges (u', v') and (v', u') in E. Return the unprimed versions of the vertices returned by FEEDBACKNODESET(G').

3. Consider the following problem:

   - SETCOVER: Given a set of sets $S = \{\{a_{11}, a_{12}, ..., a_{1n_1}\}, \{a_{21}, ..., a_{2n_2}\}, ..., \{a_{m1}, ..., a_{mn_m}\}\}$ where the a_{ij}'s may not necessarily be distinct, find the minimum number of sets, $A_1, ..., A_k$, necessary such that every element in any set in S is in at least one of the A_i.

   Reduce VERTEXCOVER to SETCOVER.

   Solution:

   Assume we have a black-box algorithm SETCOVER that solves the SETCOVER problem. Given a graph $G = (V, E)$, for each vertex $v_i \in V$, create a set $F_i \subseteq E$ of edges that have v_i as an endpoint. Run SETCOVER on the F_i to get the covering $F_{i_1}, ..., F_{i_m}$. Return the corresponding vertices $v_{i_1}, ..., v_{i_m}$.

4. Consider the following two problems:

   - PARTITION: Given a set n of non-negative integers $\{a_1, ..., a_n\}$, decide if there is there a subset $P \subseteq [1, n]$ such that $\sum_{i \in P} a_i = \sum_{i \notin P} a_i$.
   - KNAPSACK: Given a set $S = \{a_1, ..., a_n\}$ of non-negative integers, and an integer K, decide if there is there a subset $P \subseteq S$ such that $\sum_{a_i \in P} = K$.

Reduce KNAPSACK to PARTITION.

Solution: Assume we have a black-box algorithm PARTITION that can solve the PARTITION problem. Given an instance of KNAPSACK, with integers $S = a_1, ..., a_n$, let $H = \frac{1}{2}\sum_{i=1}^{n} a_i$. Add integers $a_{n+1} = 2H + 2K$ and $a_{n+2} = 4H$ to the set and return PARTITION$(a_1, ..., a_{n+2})$. The reduction is clearly polynomial. To show it works, we must show there exists $P \subseteq [1, n+2]$ with $\sum_{i\in P} a_i = \sum_{i\notin P} a_i$ if and only if there is some $Q \subseteq S$ such that $\sum_{a_i\in Q} = K$.

We first show that if there exists P, there exists Q. Firstly note that if $a_{n+1} \in P$, $a_{n+2} \notin P$ since their sum is greater than the sum of the remaining integers. WLOG assume $a_{n+2} \in P$. Then

$$\begin{aligned} & 4H + \sum_{a_i\in P\setminus a_{n+2}} a_i = 2H + 2K + \sum_{a_i\in S\setminus P} \\ \Rightarrow \quad & 4H + 2\sum_{a_i\in P\setminus a_{n+2}} a_i = 4H + 2K \\ \Rightarrow \quad & \sum_{a_i\in P\setminus a_{n+2}} = K \end{aligned}$$

Now assume there exists Q. Then

$$\sum_{a_i\in Q} a_i + 4H = 4H + K = 2H + K + \sum_{a_i} a_i = 2H + 2K \sum_{a_i\notin Q} a_i$$

so there also exists $P = Q \cup 4H$.

5. Consider the two problems:

   - CNFSATISFIABILITY: Given a set of boolean variables (i.e. variables that can only be TRUE or FALSE), $\{v_1, ..., v_n\}$ and the boolean operators $\wedge$ (AND), $\vee$ (OR), $\neg$ (NOT), a boolean expression written in conjunctive normal form (CNF) has the form $(x_{11} \vee x_{12} \vee ...x_{1m_1}) \wedge (x_{21} \vee ... \vee x_{2m_2}) \wedge ... \wedge (x_{k1} \vee ... \vee x_{km_k})$ where each x_{ij} is either a boolean variable or the negation of a boolean variable. Given a boolean expression written in CNF, determine if there is some assignment of TRUE/FALSE to each of the boolean variables $v_1, ..., v_n$ such that the expression evaluates to TRUE. Assume that $m_i \geq 3$ for all i.
   - 3-SAT: A boolean expression written in 3-CNF form has the form $(x_{11} \vee x_{12} \vee x_{13}) \wedge ... \wedge (x_{k1} \vee x_{k2} \vee x_{k3})$ where each clause has exactly 3 literals. Given a boolean expression written in 3-CNF form, determine if there is some assignment to each of the boolean variables such that the expression evaluates to TRUE.

   Reduce CNFSATISFIABILITY to 3-SAT.

   Solution:

   Assume we have a black-box algorithm for 3-SAT. We show we can use this algorithm to solve an instance of CNFSATISFIABILITY. Consider an instance of CNFSATISFIABILITY $f = (x_{1,1} \vee ... \vee x_{1,m_1}) \wedge ... \wedge (x_{n,1} \vee ... \vee x_{n,m_n})$. Let there be $L = \sum_{i=1}^{n} m_n$ literals. We introduce $L - 3n$ dummy variables $\lambda_{1,1}, ..., \lambda_{1,m_1-3}, ..., \lambda_{n,m_n-3}$ and create 3-SAT instance f' by rewriting each clause c_i as:

   $$\begin{aligned} c_i' \quad = \quad & (x_{i,1} \vee x_{i,2} \vee \neg\lambda_{i,1}) \wedge (\lambda_{i,1} \vee x_{i,3} \vee \neg\lambda_{i,2}) \wedge (\lambda_{i,2} \vee x_{i,4} \vee \neg\lambda_{i,3}) \wedge ... \\ & \wedge(\lambda_{i,m_i-4} \vee x_{i,m_i-2} \vee \neg\lambda_{i,m_i-3}) \wedge (\lambda_{i,m_i-3} \vee x_{i,m_i-1} \vee x_{i,m_i}) \end{aligned}$$

   We return 3-SAT(f').

   The reduction is polynomial since L is polynomial in the number of literals in the CNFSATISFIABILITY instance. To show it works, we show that f' has a satisfying solution if and only if f has a satisfying solution.

Firstly, assume f has a satisfying solution, $X_{1,1}, ..., X_{1,m_1}, ..., X_{n,m_n}$ where $X_{i,j}$ is the value of literal $x_{i,j}$. For example, if $x_{i,j} = \neg v_k$ and v_k has value TRUE in the satisfying assignment then $X_{i,j}$ =FALSE. Now consider applying that clause to c'_i. At least one of the $X_{i,j}$ must be TRUE or the solution would not be satisfying. Let $X_{i,k}$ be this TRUE literal. We can satisfy c'_i by setting $\lambda_{j \leq k-2}$ = FALSE and $\lambda_{j>k-2}$ = TRUE.

Now assume f' has a satisfying solution, $X_{1,1}, ..., X_{1,m_1}, ..., X_{n,m_n}, \Lambda_{1,1}, ..., \Lambda_{n,m_n-3}$. Consider clause c_i in f. We show that $X_{i,1}, ..., X_{i,m_i}$ satisfies c_i. Clearly if $\Lambda_{i,1}$ = TRUE or Λ_{i,m_i-3} = FALSE, c_i must be satisfied since one of $X_{i,1}$, $X_{i,2}$, X_{i,m_i-1}, or X_{i,m_i} is TRUE. Therefore, assume both $\Lambda_{i,1}$ = FALSE and Λ_{i,m_i-3} = TRUE. Then by construction there must be some $(\lambda_{i,j-2} \vee x_{i,j} \vee \neg\lambda_{i,j-1})$ in c'_i such that $\Lambda_{i,j-2}$ = FALSE and $\Lambda_{i,j-1}$ = TRUE. Since c'_i is satisfied, we must have $X_{i,j}$ = TRUE and c_i will also be satisfied.