Exercise 1: Unary Languages

1. Prove that if a unary language is $\text{NP}$-complete, then $P = \text{NP}$.
   
   
   Hint: consider a reduction from $\text{SAT}$ to this unary language and exhibit a polynomial time recursive algorithm for $\text{SAT}$

2. Prove that if every unary language in $\text{NP}$ is actually in $P$, then $\text{EXP} = \text{NEXP}$.

Solution:

1. Suppose we have a unary language $U$ $\text{NP}$-complete. We then have a reduction $R$ from $\text{SAT}$ to $U$. $R(\phi)$ is computed in polynomial time, so we have $p$ such that $|R(\phi)| \leq p(|\phi|)$. Basically, we can then use the self reducibility of $\text{SAT}$, but by cutting some recursions branching by using the fact that $R(\phi) = R(\psi)$ if and only if $\phi$ and $\psi$ are both satisfiable or both un-satisfiable. We will write $\phi(t)$ where $t \in \{0, 1\}^*$ to consider partial evaluation of $\phi$ where we substituted $x_i$ with the truth value of $t_i$.

   This yields the algorithm, where $n$ is the number of variables in $\phi$:

   Initialise hash table $H$

   $\text{Sat}(\phi)$

   if $|t| = n$ then return 'yes' if $\phi(t)$ has no clauses, 
   
   else return 'no'

   Otherwise, if $R(\phi(t)) \in H$, then return $H(R(\phi(t)))$

   Otherwise, return 'yes' if either $\text{Sat}(\phi(t0))$ or $\text{Sat}(\phi(t1))$.

   return no otherwise

   In both case, set $H(R(\phi(t)))$ to the answer

   There will be at most $p(n)$ different possibles values for the $R(\phi(t))$ ($U$ is unary), so there will be at most $p(n)$ recursive call of the functions. And in every recursive call, we make a computation of $R$ in time $p(n)$. So our algorithms runs in $O(p^2(n))$ which is in $P$. Thus $\text{SAT} \in P$, and $P = \text{NP}$.

2. For a language $L$ decided in time $T(n)$, we define $L_{pad} = \{1^{(x,10^{T(|x|)})}, x \in L\}$. Let $L \in \text{NEXP}$ recognized by $N$ in time $T(n)$ exponential. We build $N' \in \text{NP}$ which recognizes $L_{pad}$:

   — On input $1^m$, check the well-formdness to obtain $(x, 10^y) = m$

   — Simulate $N$ on $x$ for at most $y$ step

   — Either return the result of $N$, or reject in case of time out.

   $N'$ does recognizes $L_{pad}$, and it runs in polynomial times for the first step, and then $y$ step for the second, with $y$ being part of the input. Thus, $N' \in \text{NP}$. But then by assumption, $L \in P$, and we have $M$ a DTM which recognizes $L_{pad}$ in polynomial time. We thus simply construct $M'$ which is in exponential time, which given $x$ computes $1^{(x,10^{T(|x|)})}$ and then simulate $M$ with this input, and we are done.

Exercise 2: On the existence of one-way functions

A one-way function is a bijection $f$ from $k$-bit integers to $k$-bit integers such that $f$ is
computable in polynomial time, but \( f^{-1} \) is not. Prove that if there exists one-way functions, then
\[
A = \{(x, y) \mid f^{-1}(x) < y\} \in (\text{NP} \cap \text{coNP}) \setminus P
\]

Solution:
- \( A \in \text{NP} \) : guess a number \( c \), check that \( f(c) = x \), i.e. \( c = f^{-1}(x) \), and finally, that \( c < y \).
- \( A \in \text{coNP} \iff \{(x, y) \mid f^{-1}(x) \geq y\} \in \text{NP} \), which we solve as previously
- \( A \in P \Rightarrow f^{-1} \) computable in polynomial time

Exercise 3: Prime Numbers

1. Show that \( \text{UNARY-PRIME} = \{1^a \mid a \text{ is a prime number}\} \) is in \( P \).
2. Show that \( \text{PRIME} = \{p \mid p \text{ is a prime number encoded in binary}\} \) is in \( \text{coNP} \).
3. We want to prove that \( \text{PRIME} \) is in \( \text{NP} \). Use the following characterization of prime numbers to formulate a non-deterministic algorithm running in polynomial time.
   A number \( p \) is prime if and only if there exists \( a \in [2, p - 1] \) such that:
   (a) \( a^{p-1} \equiv 1 \pmod{p} \), and
   (b) for all \( q \) prime divisor of \( p - 1 \), \( a^{\frac{p-1}{q}} \not\equiv 1 \pmod{p} \)
   To prove that your algorithm runs in polynomial time, you can admit that all common arithmetical operations on \( \mathbb{Z}/p\mathbb{Z} \) can be performed in polynomial time.

Solution:
- 1. For every \( i < n \), we test if \( i \mid n \)
- 2. We guess the two factors
- 3. We guess the \( a \), and then make \( O(p) \) modulo exponentiation.

Exercise 4: Some \( P \)-complete problems

Show the following problems to be \( P \)-complete:
1. — INPUT : A set \( X \), a binary operator \( * \) defined on \( X \), a subset \( S \subseteq X \) and \( x \in X \)
   — QUESTION : Does \( x \) belongs to the closure of \( S \) with respect to \( * \)?
   Hint : for the hardness, reduce from Monotone Circuit Value
2. — INPUT : \( G \) a context-free grammar, and \( w \) a word
   — QUESTION : \( w \in L(G) \)?
   Hint : for the hardness, reduce from the previous problem

Solution:
1. The problem is in \( P \) as we can easily saturate until a fix point is reached. To show the hardness, we reduce Monotone Circuit Value, with gates with maximum two inputs. We are given a circuit \( C = (V, E, \text{label}) \), and we define:
   \[
   X = \{x^0, x^1 \mid x \in V\}
   \]
   \[
   S = \{x^0 \mid x \in X \land \text{label}(x) = \bot\} \cup \{x^1 \mid x \in X \land \text{label}(x) = \top\}
   \]
   \[
   x^i \cdot y^j := \begin{cases} 
   t^{i\land j} & \text{if } (x, t) \in E, (y, t) \in E, \text{label}(t) = \land \\
   t^{i\lor j} & \text{if } (x, t) \in E, (y, t) \in E, \text{label}(t) = \lor \\
   \text{undefined} & \text{otherwise}
   \end{cases}
   \]
   Finally, we have:
   \[
v(x) = a \iff x^a \in \text{Closure}(S)(a \in \{0, 1\})
   \]
Exercise 6: Complete problems for levels of \( \text{PH} \)

Show that the following problem is \( \Sigma^P_k \)-complete (under polynomial time reductions).

\( \Sigma_k \text{QBF} \): \( \text{INPUT} \): A quantified boolean formula \( \psi := \exists X_1 \forall X_2 \exists \ldots \exists Q_k X_k \phi(X_1, \ldots, X_k) \), where \( X_1, \ldots, X_k \) are \( k \) disjoint sets of variables, \( Q_k \) is the quantifier \( \forall \) if \( k \) is even, and the quantifier \( \exists \) if \( k \) is odd; \( \phi \) is a boolean formula over variables \( X_1 \cup \cdots \cup X_k \);

— \text{QUESTION} : is the input formula true?

Define a similar problem \( \Pi_k \text{QBF} \) such that \( \Pi_k \text{QBF} \) is \( \Pi^P_k \)-complete.
Exercise 7: Oracle machines

Let $O$ be a language. A Turing machine with oracle $O$ is a Turing machine with a special additional read/write tape, called the oracle tape, and three special states: $q_{\text{query}}, q_{\text{yes}}, q_{\text{no}}$. Whenever the machine enters the state $q_{\text{query}}$, with some word $w$ written on the oracle tape, it moves in one step to the state $q_{\text{yes}}$ or $q_{\text{no}}$ depending on whether $w \in O$.

We denote by $P^O$ (resp. $NP^O$) the class of languages decided in polynomial time by a deterministic (resp. non-deterministic) Turing machine with Oracle $O$. Given a complexity class $C$, we define $P^C = \bigcup_{O \in C} P^O$ (and similarly for $NP^C$).

1. Prove that for any $C$-complete language $L$, $P^C = P^L$ and $NP^C = NP^L$.
2. Show that for any language $L$, $P^L = P^L$ and $NP^L = NP^L$.
3. Prove that if $NP = \text{P}^{SAT}$ then $NP = \text{coNP}$.

Solution:

1. We do the proof for $NP$. Let $B \in NP^C$, we have $N$ a polynomial NTM for $B$ with an oracle $O$, $C \in C$. We also have a polynomial reduction $f$ such that : $x \in C \Leftrightarrow f(x) \in A$. We build $N'$ for $B$ with oracle $A$, by simulating $N$ and replacing a call $u \in C?$ with a call $f(u) \in A$?. $f$ is polynomial, so we are still in $NP$, which concludes the proof.
2. We simply have to swap the states $q_{\text{yes}}$ and $q_{\text{no}}$ in the computation.
3. $\text{P}^{SAT}$ is a deterministic class, so it is closed by complementation, so if $NP = \text{P}^{SAT}$, $\text{coNP} = NP$.
4. (Continued...)

Exercise 8: Collapse of PH

1. Prove that if $\Sigma_k^P = \Sigma_{k+1}^P$ for some $k \geq 0$ then $PH = \Sigma_k^P$. (Remark that this is implied by $P = NP$).
2. Show that if $\Sigma_k^P = \Pi_k^P$ for some $k$ then $PH = \Sigma_k^P$ (i.e. PH collapses).
3. Show that if $PH = \text{PSPACE}$ then PH collapses.
4. Do you think there is a polynomial time procedure to convert any QBF formula into a QBF formula with at most 10 variables?
Solution:

1. We assume that $\Sigma^P_k = \Sigma^P_{k+1}$ for some $k \geq 0$, we prove by induction that $\forall t \geq k, \Sigma^P_k = \Sigma^P_j$. For $j = i$, it is directly correct. For $j > i$, $\Sigma^P_j = \text{NP}^{\Sigma^P_{i+1}} = \text{NP}^{\Sigma^P_i}$ by induction, and thus $\Sigma^P_j = \Sigma^P_i$. By hypothesis, we then have $\Sigma^P_j = \Sigma^P_i$.

2. With the previous question, we just have to prove that $\Sigma^P_k = \Sigma^P_{k+1}$.

Let there be $A \in \Sigma^P_{k+1}$. A can be expressed as follows:

$$x \in A \iff \exists y_1 \in \{0, 1\}^{p(|x|)} \forall y_{k+1} \in \{0, 1\}^{p(|x|)} (x, y_1, \ldots, y_{k+1}) \in B$$

with $B \in \text{P}$. On input, $(x, y_1)$, decide if $\forall y_2 \in \{0, 1\}^{p(|x|)} \forall Q_{k+1} y_{k+1} \in \{0, 1\}^{p(|x|)} (x, y_1, \ldots, y_{k+1}) \in B$ is a problem in $\Pi^P_k = \Sigma^P_k$. We can thus rewrite it as, with $C \in \text{P}$:

$$\exists y_2 \in \{0, 1\}^{p(|x|)} \forall Q_{k+1} y_{k+1} \in \{0, 1\}^{p(|x|)} (x, y_1, \ldots, y_{k+1}) \in C$$

Finally:

$$x \in A \iff \exists y_1, y_2 \in \{0, 1\}^{p(|x|)} \forall Q_{k+1} y_{k+1} \in \{0, 1\}^{p(|x|)} (x, y_1, \ldots, y_{k+1}) \in B$$

with $B \in \text{P}$. And this is the expression of a problem in $\Sigma^P_k$. Finally, $\Sigma^P_k = \Sigma^P_{k+1}$.

3. If $\text{PH} = \text{PSPACE}$, then QBF is in $\Sigma^P_k$ for some $k$. But QBF is a complete problem for $\text{PSPACE}$, and thus $\text{PH}$. Let there be $B \in \text{PH}$, it can be reduced to $QBF \in \Sigma^P_k$, so $B \in \Sigma^P_k$, and $\text{PH} = \Sigma^P_k$.

4. It is unlikely that $\text{PH}$ collapses, and the statement would imply the previous question.

Exercise 9: Relativization

Show that there is an oracle $O$ such that $\text{P}^O = \text{NP}^O$.

Solution:

We have $\text{PSPACE} \subseteq \text{NP}^{\text{PSPACE}}$, we must show the converse. Let there be $N$ a polynomial NTM with oracle $A \in \text{PSPACE}$. We can simulate $N$ in $\text{PSPACE}$ on input $x$ by:

— enumerate all possible path of $N^A(x)$
— For each of them, compute the oracle calls
— accept if one of the path accepts.

Each path is in polynomial size, thus the enumeration is, and the oracle calls are $\text{PSPACE}$. We do have $\text{NP}^{\text{PSPACE}} \subseteq \text{PSPACE}$.