Exercise 1: Unary Languages

1. Prove that if a unary language is \(\text{NP}\)-complete, then \(\text{P} = \text{NP}\).

   \textit{Hint: consider a reduction from } \text{SAT} \text{ 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 \(\text{P}\), 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 unsatisfiable. 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 \(\phi\):

   \[
   \text{Initialise hash table } H \\
   \text{Sat}(\phi) \\
   \quad \text{if } |t| = n \text{ then return 'yes' if } \phi(t) \text{ has no clauses, } \\
   \quad \quad \text{else return 'no'} \\
   \quad \text{Otherwise, if } R(\phi(t)) \in H, \text{ then return } H(R(\phi(t))) \\
   \quad \text{Otherwise, return 'yes' if either } \text{Sat}(\phi(t0)) \text{ or } \text{Sat}(\phi(t1)). \\
   \quad \text{return no otherwise} \\
   \text{In both case, set } H(R(\phi(t))) \text{ to the answer}
   \]

   There will be at most \(p(n)\) different possibilities values for the \(R(\phi(t))\) (\(U\) is unary), so there will be at most \(p(n)\) recursive calls 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 \(\text{P}\). Thus \(\text{SAT} \in \text{P}\), and \(\text{P} = \text{NP}\).

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-formedness 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 recognize \(L_{pad}\), 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, \(L \in \text{P}\), 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 intergers to \(k\)-bit intergers 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 \( \text{UNARY-PRIME} = \{1^n \mid n \text{ is a prime number}\} \) 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) \( ap^{-1} \equiv 1 \) \( [p] \), and

(b) for all \( q \) prime divisor of \( p - 1 \), \( a^\frac{p-1}{q} \neq 1[p] \)

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 \( i | n \)
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:

\[
X = \{x^0, x^1 \mid x \in V\}
\]

\[S = \{x^0 \mid x \in X \land \text{label}(x) = \bot\} \cup \{x^1 \mid x \in X \land \text{label}(x) = \top\}\]

\[x^i \ast y^j :=
\begin{cases}
  t^{i\land j} & \text{if } (x, t) \in E, (y, t) \in E, \text{label}(t) = \land \\
  t^{i\lor j} & \text{if } (x, t) \in E, (y, t) \in E, \text{label}(t) = \lor \\
  \text{undefined otherwise}
\end{cases}\]

Finally, we have:

\[v(x) = a \iff x^a \in \text{Closure}(S) (a \in \{0, 1\})\]
2. CKY is in polynomial time. For the hardness, we reduce from the previous problem. We are given \((X, S, x, \ast)\) and we define \(G = (V, T, S, P)\) and \(w \in T^*\) in the following way: \(w\) is the empty string, the set of variables \(V = X\), there is only one terminal symbol, \(T = \{a\}\), the initial variable is \(S = \{x\}\), and the set of production is:

\[
P := \{x \rightarrow yz : y \ast z = x\} \cup \{x \rightarrow \epsilon : x \in S\}
\]

We then have:

\[
x \in \text{Closure}(S) \iff \epsilon \text{ can be generated from } x \text{ in } G
\]