1. Define

\[ \text{MODEXP} = \{ (a, b, c, p) : a, b, c, p \text{ are binary integers such that } a^b \equiv c \pmod{p} \}. \]

Show that \( \text{MODEXP} \in \mathbf{P} \).

2. Show that \( \mathbf{P} \) is closed under the \( \ast \) operation. (Hint: Use dynamic programming!)

3. Show that \( \mathbf{NP} \) is closed under union and concatenation.

4. Show that if \( \mathbf{P} = \mathbf{NP} \), then a polynomial-time algorithm exists, that, given a 3SAT instance \( \phi \), actually produces a satisfying assignment for \( \phi \) if it is satisfiable.

5. (Extra credit, not so easy) Let \( f : \mathbb{N} \to \mathbb{N} \) be any function with \( f(n) = o(n \log n) \).

Show that \( \text{TIME}(f(n)) \) contains only regular languages.

(Hint: The key concept that aids showing the above is that of a crossing sequence. When a TM is run on an input, the crossing sequence at a given cell is the sequence of states that the machine enters at that cell as the computation progresses. Now, expand on the following two high-level ideas concerning crossing sequences. First, show that for a particular TM, if all crossing sequences on all inputs are of a fixed length \( \ell \) or less, one can simulate the TM by an NFA. Second, show that a TM with unbounded crossing sequence length cannot run in \( o(n \log n) \) time. For this, use a counting/pigeonhole argument to deduce repetition of crossing sequences at multiple places and then get a contradiction by “splicing” a minimal length input to a smaller string on which the TM has identical behavior.)