Problem 1.1. If $T_1 : \mathbb{N} \to \mathbb{N}$ and $T_2 : \mathbb{N} \to \mathbb{N}$ are such that $T_1 = O(T_2)$, then $\text{DTIME}(T_1) \subseteq \text{DTIME}(T_2)$.

Problem 1.2. Consider an alternate notion of Turing machines which can delete the current symbol as well as insert a symbol in their tapes, in addition to only overwriting. Define carefully the transition function and the computation of such machines. Argue that for every $f : \{0, 1\}^* \to 0, 1^*$ and function $T : \mathbb{N} \to \mathbb{N}$, if $f$ is computed by a delete- and insert-enabled TM in time $T(n)$, then it is computed by a “normal” TM in time at most $O(T(n)^2)$. (Assume for simplicity there is one working tape.)

Problem 1.3. Prove that the following languages are in P:

1. CONNECTED: The set of all connected graphs. That is, $G \in \text{CONNECTED}$ if there is a path between every two pair of vertices $u$ and $v$.

2. TRIANGLE: The set of all graphs that contain a “triangle”: vertices $u, v, w$ with edges $(u, v), (v, w), (w, u)$.

3. Let

   $$\text{MODEXP} = \{ \langle a, b, c, p \rangle \mid a, b, c, \text{ and } p \text{ are binary integers s.t. } a^b \equiv c \pmod{p} \}$$

(Note that the obvious algorithm does not run in polynomial time. Hint: Try it first where $b$ is a power of 2.)

You can give a short description or pseudocode for the algorithm. Do not give a Turing machine!

Problem 1.4. Show that P and NP are closed under union and intersection: given $L_1$ and $L_2$ in P (respectively, NP), the languages $L_1 \cup L_2$ and $L_1 \cap L_2$ are also in P (respectively, NP).

Problem 1.5. Show that P and NP are closed under concatenation: given $L_1$ and $L_2$ in P (respectively, NP), the language $L_1 \cdot L_2 = \{ w \mid \exists u, v : w = u \cdot v \text{ and } u \in L_1, v \in L_2 \}$ is also in P (respectively, NP).

Problem 1.5. Show the following languages are NP-complete:
1. \[ \text{HALFCLIQUE} = \{ G \mid G \text{ is an undirected graph having a clique of at least } n/2 \text{ nodes,} \]

where \( n \) is the number of nodes of \( G \).

2. \[ \text{LPATH} = \{ (G, s, t, k) \mid \text{graph } G \text{ contains a simple path from } s \text{ to } t \text{ of length at least } k \}. \]

3. Would your answer to (2) change if \( k \) is given in unary?

Remember to show two properties: the language belongs to NP and that it is NP-hard. You may take any language shown to be NP-hard in class or in Arora-Barak as a starting point for a reduction.

**Problem 1.6.** Let

\[ \text{CNF}_k = \{ \varphi \mid \varphi \text{ is a satisfiable cnf-formula} \]

where each clause has at most \( k \) variables\}.\]

Show that \( \text{CNF}_2 \) is in P and \( \text{CNF}_3 \) is NP-complete.

**Problem 1.7.** Show that if P = NP, then there is a polynomial time algorithm that takes a graph as input and finds a largest clique contained in that graph.

**Problem 1.8.** Look up an example of an NP-complete problem on the web that is not described in Arora-Barak’s textbook in Chapter 2 (not even in the exercises). State the problem you have found.