L. Assume that $A \leq_p A$ via function $f$. This means that also $\overline{A} \leq_p A$ via function $f$ (weA imagination). Let $B \in \text{NP}$ be arbitrary. Since $A$ is $\text{NP}$-complete, we have $B \leq_p A$, which implies $\overline{B} \leq_p \overline{A}$. By the transitivity of $\leq_p$, we have $B \leq_p A$. Since $A \in \text{NP}$, it follows that $\overline{B} \in \text{NP}$, that is $B \in \text{coNP}$. This is seen as in the proof of Theorem 7.31, we just need to replace $M$ there by an $\text{NP}$-machine. We have shown that $\text{NP} \subseteq \text{coNP}$.

For the converse inclusion note:

- $B \in \text{coNP}$ implies $\overline{B} \in \text{NP} \subseteq \text{coNP}$ implies $B \in \text{NP}$.

\[\text{shown above}\]
2. Problems in $P$ are believed not to be 
   NP-complete and $\text{PATH} \in P$ (Theorem 7.14) 
   (In fact $\text{PATH} \notin \text{NL}$, Example 8.19.)

b) Claim. If $\text{PATH}$ is not NP-complete, then $P \neq \text{NP}$.

   Proof. It is sufficient to show that $P = \text{NP}$ implies 
   $\text{PATH}$ is NP-complete.

   Choose $w_1 \in \text{PATH}$ and $w_2 \notin \text{PATH}$ 
   (this is possible since $\text{PATH} \neq \emptyset$, $\neq \Sigma^*$.)

   Let $A \in \text{NP}$ be arbitrary. Define mapping reduction 
   $f(a) = \begin{cases} 
   w_1 & \text{if } a \in A \\
   w_2 & \text{if } a \notin A 
   \end{cases}$

   $f$ mapping reduces $A$ to $\text{PATH}$. Since $A \in \text{NP} = P$, 
   $f$ can be computed in polynomial time.

   Thus, every $A \in \text{NP}$ reduces to $\text{PATH}$, and we have 
   shown that $\text{PATH}$ is $\text{NP}$-hard. It is known that 
   $\text{PATH} \in P$. 


3. Testing formula non-equivalence is in $NP$:
   - guess an assignment that makes one formula true and the other false and verify the guess.
   - Assumption $P=NP$ implies $NP = coNP$, and thus testing equivalence is in $NP$.

Now complement of MIN-FORMULA is in $NP$:
   - guess shorter equivalent formula in $NP$
   - verify that the guessed formula and original formula are equivalent.

Since $NP = coNP$, we have shown that MIN-FORMULA $\in NP$ and the assumption $P=NP$ implies MIN-FORMULA $\notin P$. 
Since PSPACE is closed under complementation, it is sufficient to give a polynomial space algorithm for $\overline{\text{EQ}_{\text{NFA}}}$ (complement of the language). The following nondeterministic algorithm uses a similar idea as the algorithm for $\overline{\text{ALL}_{\text{NFA}}}$ on p. 333.

Q = "On input $<A, B>$ where $A, B$ are NFAs

1. Place a marker on the start states of $A$ and $B$

2. Let $m$ be the sum of numbers of states of $A$ and $B$. 
Repeat $2^m$ times:

2(i) Nondeterministically select an input symbol and change the positions of each machine’s markers to simulate reading that symbol. The markers indicate all possible states the automaton can be in with input read so far.

3. Accept if Stage 2 finds some string that one NFA accepts and the other rejects. Otherwise reject.

The algorithm needs to mark some subset of each of the NFAs. Hence the algorithm runs in nondeterministic space $O(n^*)$ and by Savitch’s theorem $\text{EQ}_{\text{NFA}}$ is in $\text{SPACE}(n^2)$.

[(*) Also the counter (up to $2^m$) can be stored in linear space.]
Consider an input instance \( <M, w> \) when the NFA \( M \) has \( m \) states. An NTM \( N \), with \( <M, w> \) on the read-only input tape, can nondeterministically simulate the computation of \( M \) on \( w \) by keeping track on the work tape of the current state \( q_1 \) of \( M \) and the position in \( w \) where \( M \)'s computation is at.

When simulating on computation step \( s \), \( N \) increments the position by one, writes down the new state \( q_2 \) of \( M \) and erases the description of \( q_1 \).

At a given time, \( N \) needs to store on the work tape 2 states of \( M \) and a position in \( w \). Denote the length of the input \( n = |<M, w>| \). Now \( m \leq n \) and \( |w| \leq n \) and hence the names of the two states and a position in \( w \) can be stored in \( O(\log n) \) space.
6.1 a) \( \text{TIme}(n^3) \leq \text{TIme}(n^3 \cdot \log n) \) not known to be strict

With \( t_1(n) = n^3 \), \( t_2(n) = n^3 \log n \), \( t_3(n) \) is not \( o\left(\frac{t_2(n)}{\log t_2(n)}\right) \) and time hierarchy theorem does not apply.

b) \( \text{NSpace}(n^2) \leq \text{Space}(n^4) \leq \text{Space}(n^4 \cdot \log n) \)

Switch space hierarchy, \( n^4 \) is \( o(n^4 \cdot \log n) \)

c) \( \text{TIme}(3^n) \not\leq \text{TIme}(n^3 \cdot 3^n) \) strict inclusion

\[
t_3(n) = n^3 \cdot 3^n, \quad \frac{t_2(n)}{\log t_2(n)} = \frac{n^2 \cdot 3^n}{2n + (\log 3)n} \approx c \cdot n \cdot 3^n
\]

Thus \( 3^n = o\left(\frac{t_2(n)}{\log t_2(n)}\right) \) and time hierarchy theorem implies strict inclusion.
d) $\text{NSPACE}(\log n) \subseteq \text{SPACE}((\log n)^2) \subseteq \text{SPACE}(\sqrt{n})$

Switch space hierarchy

since $(\log n)^2$ is $o(\sqrt{n})$

e) $f(n)$ is $O(n^3 \cdot \log n)$

$n^3 \cdot \log n$ is $o\left(\frac{n^4}{\log n^4}\right)$ (because $n^3(\log n)^2$ is $o(n^4)$)

$\text{TIME}(A(n)) \leq \text{TIME}(n^4)$ by the time hierarchy theorem