Suppose you are given an array \( A[1..n] \) of \( n \) integers and you seek an integer which is not in \( A \).

\( a \) Give a \( O(n) \) upper bound on the worst-case complexity of the problem.

\( b \) Give a \( \Omega(n) \) lower bound on the worst-case complexity of the problem.

**SOLUTIONS:**

\( a \) The following algorithm solves the problem in time in \( O(n) \).

\[
\begin{aligned}
\text{MAX} &\leftarrow -\infty \\
\text{for } i &\leftarrow 1 \text{ to } n \\
&\quad \text{if } A[i] > \text{MAX} \\
&\quad \quad \text{then } \text{MAX} \leftarrow A[i] \\
\text{return } \text{MAX}+1
\end{aligned}
\]

\( b \) Assume there is an algorithm to solve the problem which does not examine every element in \( A \). For input \( A[i]=i, \ 1 \leq i \leq n \), it returns some value, say \( x \). Let \( A[j] \) be some element of \( A \) which wasn’t examined. Construct array \( A^*[1..n] \) with

\[
A^*[i] = \begin{cases} 
  i, & \text{if } i \neq j \\
  x, & \text{if } i = j
\end{cases}
\]

Every question that the algorithm asks of \( A^* \) is the same it asked of \( A \), and it gets the same answer. So it must return that \( x \) is not in \( A^* \), which is wrong. This contradiction means that the assumption that the algorithm exists must be wrong.