1. To show that $3n^3$ is $O(n^4)$ we must find constants $c > 0$ and $n_0 \geq 1$ such that
\[ 3n^3 \leq c \times n^4, \quad \forall n \geq n_0 \] (1)
We can simplify this inequality by dividing both sides of the inequality by $n^3$ (this can be done as $n > 0$) to get
\[ 3 \leq cn, \quad \forall n \geq n_0 \]
Now we can choose, for example, $c = 1$ and the above inequality becomes
\[ 3 \leq n, \quad \forall n \geq n_0. \]
Therefore, we can choose $n_0 = 3$. Since we have found constant value $c = 1$ and $n_0 = 3$ that make inequality (1) true then we have proven that $3n^3$ is $O(n^4)$.

2. We can use a proof by contradiction: Assume that $n^3 + n^2$ is $O(n^2)$ and derive a contradiction. If $n^3 + n^2$ is $O(n^2)$ then there are constants $c > 0$ and $n_0 \geq 1$ for which
\[ n^3 + n^2 \leq cn^2, \quad \forall n \geq n_0. \]
Dividing both sides by $n^2$, and moving the term 1 to the right, we get
\[ n \leq c - 1, \quad \text{for all } n \geq n_0, \]
which cannot hold, since $c$ is a constant but $n$ grows without bound. Therefore, $n^3 + n^2$ is not $O(n^2)$.

Alternative proof
To show that $n^3 + n^2$ is not $O(n^2)$ we have to prove that it is not possible to find constant values $c > 0$ and $n_0 \geq 1$ such that
\[ n^3 + n^2 \leq cn^2, \quad \forall n \geq n_0. \]
This is equivalent to show that for every constant values $c > 0$ and $n_0 \geq 1$, the following inequality is true
\[ n^3 + n^2 > cn^2, \quad \text{for at least one value } n \geq n_0. \] (2)
Divide both sides of the inequality by $n^2$ and move the term 1 to the right to get
\[ n > c - 1, \quad \text{for at least one value } n \geq n_0. \]
Since $n$ grows without bound, then regardless of the values for $c$ and $n_0$ (as long as they are constant) for all values $n > \max \{c-1, n_0\}$ the above inequality holds and therefore $n^3 + n^2$ is not $O(n^2)$.
3. To show that \( f(n) - g(n) \) is \( O(f(n)) \), we must find constant values \( c > 0 \) and \( n_0 \geq 1 \) such that

\[
f(n) - g(n) \leq cf(n), \quad \forall n \geq n_0
\]  

(3)

To find these values for \( c \) and \( n_0 \) we use the fact that \( f(n) = O(g(n)) \) and \( g(n) = O(f(n)) \), or in other words there are constant values \( c' > 0 \) and \( n_0' \geq 1 \) such that

\[
f(n) \leq c'g(n), \quad \forall n \geq n_0'.
\]  

(4)

and there are constant values \( c'' > 0 \) and \( n_0'' \geq 1 \) such that

\[
g(n) \leq c''f(n), \quad \forall n \geq n_0''.
\]  

(5)

Subtracting \( g(n) \) from both sides of inequality (4) we get

\[
f(n) - g(n) \leq c'g(n) - g(n) = (c' - 1)g(n), \quad \forall n \geq n_0'.
\]  

(6)

Using inequality (5) in the right hand side of (6) we get

\[
f(n) - g(n) \leq (c' - 1)c''f(n), \quad \forall n \geq \max\{n_0', n_0''\}
\]  

(7)

Note that inequality (6) holds for all \( n \geq n_0' \) and inequality (5) holds for all \( n \geq n_0'' \), so inequality (7) holds for all values \( n \) that satisfy \( n \geq n_0' \) and \( n \geq n_0'' \), namely all values \( n \geq \max\{n_0', n_0''\} \).

Hence, choosing \( c = (c' - 1)c'' \) and \( n_0 = \max\{n_0', n_0''\}, \) gives the desired result. Observe that \( (c' - 1)c'' \) and \( \max\{n_0', n_0''\} \) are constants.

4.i. Algorithm NoCommonValues(\( A, B, n \))

**In:** Arrays \( A \) and \( B \) storing each \( n \) different integer values

**Out:** True if no value in \( A \) is in \( B \); false otherwise

\[
\{ \\
\text{for } i \leftarrow 0 \text{ to } n - 1 \text{ do } \{ \\
\quad j \leftarrow 0 \\
\quad \text{while } (j < n) \text{ and } (B[j] \neq A[i]) \text{ do} \\
\quad \quad j \leftarrow j + 1 \\
\quad \quad \text{if } j < n \text{ then return false} \\
\} \\
\text{return true} \\
\}
\]

4.ii(a) The outside loop is repeated at most once for each value of \( i \) between 0 and \( n - 1 \). Hence, the outside loop is repeated at most \( n \) times. The inside while loop is repeated at most \( n \) times for each iteration of the outside loop (once for each value of \( j \) between 0 and \( n - 1 \)); therefore, the total number of iterations of the inside loop is at most \( n^2 \). Since the number of iterations of each loop is finite, the algorithm must terminate after a finite amount of time.
4.ii (b) The nested loops in the algorithm consider all pairs of values $A[i], B[j]$ such that $0 \leq i, j \leq n - 1$. Hence, if for any value $A[i]$ there is a value $B[j]$ such that $A[i] = B[j]$ the while loop will find it: If $A[i] = B[j]$ the second condition of the while loop is false so the loop terminates; note that then $j < n$ so the condition of the if statement is true and the algorithm correctly returns the value false.

On the other hand, if no value $A[i]$ is in $B$ the second condition of the while loop will never be false and so the condition of the if statement will never be true and hence the algorithm will correctly return the value true after the for loop ends.

4.iii. Worst case. The worst case for the algorithm is when $A$ and $B$ have no common values as in this case the condition of the if statement is always false and so the algorithm will not end early; furthermore, in this case the second condition of the while loop is never false causing the loops to perform the maximum possible number of iterations. Hence, if $A$ and $B$ have no common values, the algorithm will perform the maximum possible number of operations.

4.iv. Time complexity. We analyze the while loop first. In every iteration of this loop a constant number $c$ of operations is performed and in the worst case the loop is repeated $n$ times. Thus, the total number of operations performed by this loop is $cn$.

Outside the while loop, but inside the for loop, the algorithm performs a constant number $c'$ of operations (to update the value of $i$ and to set $j$ to 0). Therefore, each iteration of the for loop performs $c' + cn$ operations.

The for loop iterates once for each value of $i$ from 0 to $n - 1$, thus the total number of operations performed in this loop is

\[ \sum_{i=0}^{n-1} (cn + c') = cn^2 + c'n \]

Ignoring constant terms and since $n^2 > n$, we conclude that the time complexity of the algorithm is $O(n^2)$.