1. To show that \( 4/n \) is \( O(1) \) we must find constants \( c > 0 \) and \( n_0 \geq 1 \) such that

\[
\frac{4}{n} \leq c \times 1, \quad \forall n \geq n_0
\]  

(1)

We can simplify this inequality by moving \( n \) to the right-hand side to get

\[
4 \leq cn, \quad \forall n \geq n_0
\]

Now we can choose, for example, \( c = 1 \) and the above inequality becomes

\[
4 \leq n, \quad \forall n \geq n_0.
\]

Therefore, we can choose \( n_0 = 4 \). Since we have found constant value \( c = 1 \) and \( n_0 = 4 \) that make inequality (1) true then we have proven that \( 4/n \) is \( O(1) \).

2. To show that \( 2n \) is not \( O(1/n) \) we have to prove that it is not possible to find constant values \( c > 0 \) and \( n_0 \geq 1 \) such that

\[
2n \leq c/n, \quad \forall n \geq n_0.
\]

This is equivalent to show that for every constant values \( c > 0 \) and \( n_0 \geq 1 \), the following inequality is true

\[
2n > c/n, \quad \text{for at least one value } n \geq n_0.
\]  

(2)

Multiply both sides of the inequality by \( n \) and then divide them by 2 to get

\[
n^2 > c/2, \quad \text{for at least one value } n \geq n_0.
\]

Since \( \lim_{n \to \infty} n^2 = \infty \), or in other words \( n^2 \) grows without bound, then regardless of the values for \( c \) and \( n_0 \) (as long as they are constant) for all values \( n > \max\{\sqrt{c/2}, n_0\} \) the above inequality holds and therefore \( 2n \) is not \( O(1/n) \).

Alternative proof

We can also use a proof by contradiction: Assume that \( 2n \) is \( O(1/n) \) and derive a contradiction. If \( 2n \) is \( O(1/n) \) then there are constants \( c > 0 \) and \( n_0 \geq 1 \) for which

\[
2n \leq c/n, \quad \forall n \geq n_0.
\]

Multiplying both sides by \( n \), and then dividing them by 2, we get

\[
n^2 \leq c/2, \quad \text{for all } n \geq n_0,
\]

which cannot hold, since \( n^2 \) grows without bound, so for all values \( n > \max\{\sqrt{c/2}, n_0\} \) the above inequality does not hold regardless of the values for \( c \) and \( n_0 \). Therefore, \( 2n \) is not \( O(1/n) \).
3. To show that \( f(n) + g(n) \) is \( O(h(n)) \), we must find constant values \( c > 0 \) and \( n_0 \geq 1 \) such that

\[
f(n) + g(n) \leq ch(n), \quad \forall n \geq n_0
\]  

(3)

To find these values for \( c \) and \( n_0 \) we use the fact that \( f(n) \) is \( O(g(n)) \) and \( g(n) \) is \( O(h(n)) \). This means that 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

\[
g(n) \leq c''h(n), \quad \forall n \geq n''_0.
\]  

(5)

Using inequality (5) in the right hand side of inequality (4) we get

\[
f(n) \leq c'c''h(n), \quad \forall n \geq \max\{n'_0, n''_0\}.
\]  

(6)

Note that inequality (4) holds for all \( n \geq n'_0 \) and inequality (5) holds for all \( n \geq n''_0 \), so inequality (6) holds for all values \( n \) that satisfy \( n \geq n'_0 \) and \( n \geq n''_0 \).

Adding inequalities (5) and (6) we get

\[
f(n) + g(n) \leq c'c''h(n) + c''h(n) = (c'c'' + c'')h(n), \quad \forall n \geq \max\{n'_0, n''_0\}
\]

Note that this inequality holds only for those values of \( n \) for which both inequalities (4) and (6) hold, namely all values \( n \geq \max\{n'_0, n''_0\} \).

Hence, choosing \( c = c'c'' + c'' \) and \( n_0 = \max\{n'_0, n''_0\} \), gives the desired result. Observe that \( c'c'' + c'' \) and \( \max\{n'_0, n''_0\} \) are constant.

4. **Algorithm** AllDifferent(A, n)

   **In:** Array \( A \) storing \( n \) integer values

   **Out:** True if all values in \( A \) are different; false otherwise

   |
   |   for \( i \leftarrow 0 \) to \( n - 2 \) do |
   |       for \( j \leftarrow i + 1 \) to \( n - 1 \) do |
   |           if \( A[j] = A[i] \) then return false |
   |
   | return true |
   |

   (4a) The outside loop is repeated at most once for each value of \( i \) between 0 and \( n - 2 \). Hence, the outside loop is repeated at most \( n - 1 \) times. The inside for loop is repeated at most \( n - 1 - (i + 1) + 1 = n - i - 1 \) times for each iteration of the outside loop. Therefore, the total number of iterations of the inside loop is at most

\[
\sum_{i=0}^{n-2} (n - i - 1) = \sum_{i=0}^{n-2} n - \sum_{i=0}^{n-2} i - \sum_{i=0}^{n-2} 1 = n(n - 1) - \frac{(n - 1)(n - 2)}{2} - (n - 1) = \frac{n^2 - n}{2}
\]

This is a finite number, so the algorithm must terminate after a finite amount of time.
The for loops in the algorithm consider all pairs $A[i], A[j]$ such that $0 \leq i < j \leq n - 1$, or in other words the algorithm considers all pairs of values $A[i], A[j]$ in $A$ where $i \neq j$.

Since the algorithm considers all pairs $A[i], A[j]$ where $i \neq j$, then if a pair exists such that $A[i] = A[j]$ the algorithm must find it and so the condition of the if statement in the third line will be true for such a pair and the algorithm will correctly return the value $\text{false}$. If no pair exists such that $A[i] = A[j]$ and $i \neq j$ then the condition of the if statement of line 3 will always be false, so the algorithm will correctly return the value $\text{true}$.

- **Worst case.** The worst case for the algorithm is when all values in $A$ are different, as in this case the condition of the if statement is always false and so the algorithm will not end early causing the for loops to perform the maximum possible number of iterations; hence the algorithm performs in this case the maximum possible number of operations.

- **Time complexity.** We analyze the inner-most for loop first. In every iteration of this loop a constant number $c'$ of operations is performed and the loop is repeated $n - i - 1$ times. Thus, the total number of operations performed by this loop is $c(n - i - 1)$.

As for the first for loop, in every iteration it performs a constant number $c'$ of operations to update the value of $i$ plus the $c(n - 1 - i)$ operations of the second for. The first loop iterates once for each value of $i$ from 0 to $n - 2$, thus the total number of operations performed in this loop is

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

$$
= (n - 1)(c' - c) + c(n - 1)n - \frac{(n - 1)(n - 2)c}{2}
$$

$$
= (c - \frac{c}{2})n^2 + (c' - c - c + \frac{3}{2}c)n - (c' - c + c)
$$

$$
= \frac{c}{2}n^2 + (c' - \frac{c}{2})n - c' \text{ is } O(n^2)
$$