Complexity of Algorithms

1. Show that if \( f(n) \) is \( \Theta(h_1(n)) \) and \( g(n) \) is \( \Theta(h_2(n)) \), then \( f(n) * g(n) \) is \( \Theta(h_1(n) * h_2(n)) \).

   To show that \( f(n) * g(n) \) is \( \Theta(h_1(n) * h_2(n)) \) we need to show:
   1. \( f(n) * g(n) = O(h_1(n) * h_2(n)) \)
   2. \( f(n) * g(n) = \Omega(h_1(n) * h_2(n)) \)

   1.1. To show that \( f(n) * g(n) = O(h_1(n) * h_2(n)) \) we need to show that
   \[ |f(n) * g(n)| \leq C1 * |h_1(n) * h_2(n)| \]

   Since it is given that \( f(n) \) is \( \Theta(h_1(n)) \), we know from the definition of \( \Theta \) that \( f(n) = O(h_1(n)) \), so by the definition of big-O we have:
   \[ |f(n)| \leq C1 * |h_1(n)| \]
   for \( n \geq k_1 \), and some \( C1 > 0 \)

   Similarly, we find that
   \[ |g(n)| \leq C2 * |h_2(n)| \]
   for \( n \geq k_2 \), and some \( C2 > 0 \)

   We can combine the two inequalities by multiplying the two sides, but this can be done only when \( n \geq \max(k_1, k_2) \) to ensure that both inequalities are true.

   \[ |f(n) * g(n)| \leq C1 * |h_1(n)| * C2 * |h_2(n)| \]
   for \( n \geq \max(k_1, k_2) \)

   Since \( |a| * |b| = |a * b| \) we get:
   \[ |f(n) * g(n)| \leq C1 * C2 * |h_1(n) * h_2(n)| \]

   Letting \( C3 = C1 * C2 \) we get:
   \[ |f(n) * g(n)| \leq C3 * |h_1(n) * h_2(n)| \]
   for \( n \geq \max(k_1, k_2) \)

   This shows that \( f(n) * g(n) = O(h_1(n) * h_2(n)) \) with \( C3 = C1 * C2 > 0 \)

   1.2. To show that \( f(n) * g(n) = \Omega(h_1(n) * h_2(n)) \) repeat the above argument, but reverse the inequalities, use big-Omega and use different subscripts (i.e. \( C4, C5, C6, k_4, k_5, k_6 \))

2. Show that if \( f(n) \) is \( \Theta(h_1(n)) \) and \( g(n) \) is \( \Theta(h_2(n)) \), then \( f(n) + g(n) \) is \( \Theta(h(n), h_2(n)) \).

   To show that \( f(n) + g(n) \) is \( \Theta(h(n), h_2(n)) \) we need to show:
   1. \( f(n) + g(n) = O(max(h_1(n), h_2(n))) \)
   2. \( f(n) + g(n) = \Omega(max(h_1(n), h_2(n))) \)

   2.1. To show that \( f(n) + g(n) = O(max(h_1(n), h_2(n))) \) we need to show that
   \[ |f(n) + g(n)| \leq C * max(h_1(n), h_2(n)) \]

   Since it is given that \( f(n) \) is \( \Theta(h_1(n)) \), we know from the definition of \( \Theta \) that \( f(n) = O(h_1(n)) \), so by the definition of big-O we have:
   \[ |f(n)| \leq C1 * |h_1(n)| \]
   for \( n \geq k_1 \), and some \( C1 > 0 \)

   Similarly, we find that
   \[ |g(n)| \leq C2 * |h_2(n)| \]
   for \( n \geq k_2 \), and some \( C2 > 0 \)

   We can combine the two inequalities by adding the two sides, but this can be done only when \( n \geq \max(k_1, k_2) \) to ensure that both inequalities are true.

   \[ |f(n) + g(n)| \leq C1 * |h_1(n)| + C2 * |h_2(n)| \]
   for \( n \geq \max(k_1, k_2) \)

   By the triangle inequality we have \( |a| + |b| = |a + b| \) and thus:

   \[ |f(n) + g(n)| \leq |f(n)| + |g(n)| \leq C1 * |h_1(n)| + C2 * |h_2(n)| \]

   Letting \( h_3(n) = max(h_1(n), h_2(n)) \) we get:

   \[ |f(n) + g(n)| \leq C1 * |h_3(n)| + C2 * |h_3(n)| \]

   (each term is bounded by a bigger term overall)

   \[ |f(n) + g(n)| \leq C1 * |h_1(n)| + C2 * |h_2(n)| \]

   \[ |f(n) + g(n)| \leq (C1+C2) * |h_3(n)| \]

   \[ |f(n) + g(n)| \leq C3 * |h_3(n)| \]

   \[ |f(n) + g(n)| \leq C3 * max(h_1(n), h_2(n)) \]
   where \( C3 = C1 + C2 > 0 \), \( n \geq \max(k_1, k_2) \)

   This shows that \( f(n) * g(n) = O(h_1(n) * h_2(n)) \) with \( C3 = C1 + C2 > 0 \)

   2.2. To show that \( f(n) * g(n) = G(h_1(n) * h_2(n)) \) repeat the above argument, but reverse the inequalities, use big-Omega and use different subscripts (i.e. \( C4, C5, C6, k_4, k_5, k_6 \))

3. Application

   The second theorem is useful in analyzing the complexity of an algorithm that is implemented in terms of other algorithms.

   \[
   \text{def someAlgorithm(numbers):}
   \]

   1. find the max \( f1(n) \) is \( \Theta(n) \)
   2. add max to each number \( f2(n) \) is \( \Theta(n) \)
   3. sort the new numbers \( f3(n) \) is \( \Theta(n^2) \)

   The running time of the algorithm is \( f1(n) + f2(n) + f3(n) \) which by the second theorem is \( \Theta(max(n, n, n^2)) \) or just \( \Theta(n^2) \).