We write \( f(n) \in O(g(n)) \) (aloud, this is “\( f(n) \) is in big-Oh of \( g(n) \)”) to mean that the function \( f \) is eventually bounded by some multiple of \( |g(n)| \). More precisely,

\[
f(n) \in O(g(n)) \iff |f(n)| \leq c \cdot |g(n)|, \text{ for all } n > n_0,
\]

for some constants \( c > 0 \) and \( n_0 \). That is, \( O(g(n)) \) is the set of functions that “grow no more quickly than” \( |g(n)| \) does as \( n \) gets sufficiently large. Somewhat confusingly, \( f(n) \) here does not mean “the result of applying \( f \) to \( n \),” as it usually does. Rather, it is to be interpreted as the body of a function whose parameter is \( n \). Thus, we often write things like \( O(n^2) \) to mean “the set of all functions that grow no more quickly than the square of their argument.”

1. Suppose \( T_1(n) \in O(f(n)) \) and \( T_2(n) \in O(f(n)) \). Answer whether the following are true or false and give justification. A justification for being true is a simple mathematical proof, while you can prove something to be false by giving a specific counterexample. You should use the above definition of big-Oh in both cases.

   (a) \( T_1(n) + T_2(n) \in O(f(n)) \)

   (b) \( T_1(n) - T_2(n) \in O(f(n)) \)

   (c) \( T_1(n)/T_2(n) \in O(1) \)

   (d) \( T_1(n) \in O(T_2(n)) \)
2. By doubling the size of an array used to store an ArrayList, we pay constant amortized time for each add operation. Suppose that allocating a ArrayList containing \( M \) elements takes \( M/2 + 10 \) time units, copying \( M \) elements from one ArrayList to another takes \( M \) time units, and a push takes 1 time unit plus the amount of time (if any) required to increase the size of the ArrayList.

(a) If the size of the ArrayList increases by 100 (that is, 100 more elements, not 100 times as many), how long will \( N \) add operations take?

(b) If the size of the ArrayList doubles each time the stack fills up, how long with \( N \) add operations take?

(c) If the size of the ArrayList increases by a factor of 1.5 each time, how long will \( N \) adds take?

3. Extra Credit: Big Omega gives the lower bound for a function. More precisely, \( f(n) \in \Omega(g(n)) \) means that for all \( n > n_0 \), \( |f(n)| \geq c|g(n)| \) for \( n > n_0 \), for some constants \( c > 0 \) and \( M \). That is, \( \Omega(g(n)) \) is the set of all functions that “grow at least as fast as” \( g \) beyond some point.

Show \( n! \in \Omega(2^n) \).