Big O notation

For each of the following, show that \( f(n) \in O(g(n)) \)

1. \( f(n) = 3n \quad g(n) = 5n^2 \)
   \[
   \begin{align*}
   5n^2 &= 5n^2 \\
   3n^2 &\leq 5n^2 \text{ for } n \geq 1 \\
   3n &\leq 5n^2 \text{ for } n \geq 1
   
   \text{Therefore, with } c=1 \text{ and } n_0 = 1, \quad f(n) \in O(g(n))
   \end{align*}
   \]

2. \( f(n) = 17 \quad g(n) = 32n + n \cdot \log(n) \)
   \[
   \begin{align*}
   32n + n\log n &= 32n + n\log n \\
   32n &\leq 32n + n\log n \text{ for } n \geq 1 \\
   32 &\leq 32n + n\log n \text{ for } n \geq 1 \\
   17 &\leq 32n + n\log n
   \end{align*}
   \]

3. \( f(n) = 121n^3 \quad g(n) = 11n^3 \)
   \[
   \begin{align*}
   121n^3 &= 11 \cdot 11n^3 \\
   121n^3 &\leq 11 \cdot 11n^3 \text{ for all } n \\
   \text{Therefore, with } c=11 \text{ and } n_0 = 1 \text{ and } f(n) \in O(g(n))
   \end{align*}
   \]
Asymptotic Analysis

For the following methods, determine asymptotic runtime in terms of n

1. void method1(int n, int sum){
   for(int i = 0; i<n*100;i++){
      for(int j = n; j > 0; j--){
         sum++;
      }
   }
   for(int k = 0; k<i; k++){
      sum++;
   }
}

\[ O(n^2) \]

\[ \sum_{i=0}^{n} 100i = n^2 \]
\[ \frac{100n(100n+1)}{2} = n^2 \]

2. void method2(int n, int sum){
   int j = n;
   while (j>2){
      sum++;
      j = j/2;
   }
}

\[ O(\log_2 n) \]
Comparing runtimes

Order the following functions from fastest to slowest in terms of asymptotic runtime. If there are multiple in the same bigO family, indicate this.

- \( n \times (n^2 \times \log(n) + n) \)
- \( n^2 \)
- \( 10000n^3 \)
- \( 2^n + 3 \)
- \( n^{\frac{1}{2}} + n + 128 \)
- \( n + \log(n) + \frac{n}{5} \)

\[
\begin{align*}
&n + \log(n) + \frac{n}{13} \geq \text{both } O(n) \\
&\sqrt{n} + n + 128 \\
&n^2 \\
&\sqrt{n^2} = 10000n^3 \\
&n \cdot (n^2 \cdot \log(n) + n) \\
&2^n + 3
\end{align*}
\]
Dictionaries

Provide pseudocode for inserting into a sorted array. What are the best and worst case runtimes for the method you proposed?

\[
given \text{ element } \leq \ \text{perform a binary search to find the correct location}\n\]

\[
\text{shift all other elements over}\n\]

**best case:** last element in list

because there is no shifting

only the binary search takes time

\[O(\log(n))\]

**worst case:** first element

all n elements must be shifted

\[O(n)\]