1. **Asymptotic comparisons.** Order the following functions from slowest-growing to fastest-growing. Some of them may be equal (i.e., may belong to the same $\Theta$ class).

\[
2n^2 \quad 65n^{4321} \quad \lg n \quad 1700n \quad 1.5^n \quad n^3 + 4n^2 + 9000 \quad n \quad \frac{1}{1000}n^3
\]

2. **Analyzing algorithms.** Suppose I wrote an expression, $T(n)$, for the runtime of my algorithm, and then I compared $T(n)$ to some other functions as follows:

Which of the following claims are true about the worst-case performance of my algorithm? (More than one claim may be true).

I. It's $\Theta(f(n))$.

II. It's $\Omega(h(n))$.

III. It's $O(g(n))$, provided $T(n)$ describes the runtime of the worst possible input.

IV. It's $O(g(n))$.

V. It's $\Theta(f(n))$, provided $T(n)$ describes the runtime of the worst possible input.

VI. It's $\Omega(f(n))$. 

![Graph showing runtime functions](image)
3. Writing recurrences. Here’s an algorithm that you may (or may not) have seen before:

\[
\text{BinarySearch}(A, x, i, j) \quad // \ A \text{ is an array of length } n
\]
\[
\begin{align*}
\text{if } j < i & \text{ return } false \\
\text{mid } & = (i+j)/2 \\
\text{if } A[\text{mid}] & = x \text{ return } true \\
\text{if } A[\text{mid}] & > x \text{ return BinarySearch}(A, x, i, \text{mid-1}) \\
\text{else } & \text{ return BinarySearch}(A, x, \text{mid+1}, j)
\end{align*}
\]

Write a recurrence for \( T(n) \), the runtime of BinarySearch. You do not need to solve your recurrence (but feel free to if you want the practice!)