(1) (Simple induction practice) Use induction to prove that \( \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \).

(2) Using the definition of big-O notation, show that \( 2n^2 + 25n \) is \( O(n^2) \) (that is, choose an appropriate \( c \) and \( n_0 \), and show that the definition of big-O is fulfilled).

(3) Is \( 3^n \) \( O(2^n) \)? Prove your answer.

(4) Is \( \log_2(n) \) \( O(\log_3 n) \)? Prove your answer.

(5) Is \( \log_2(n) \) \( \Theta(\log_3 n) \)? Prove your answer.

(6) Use induction to prove that \( \sum_{i=1}^{n} i \) is \( O(n^2) \).

(7) Is \( \sum_{i=1}^{n} i! \) \( O(n^n) \)? Prove your answer.

(8) I just invented a new version of binary search, where rather than using \( \text{mid} = (\text{low} + \text{high})/2 \), I choose \( \text{mid} = (\text{low} + \text{high}) / 4 \). Is the running time of the resulting algorithm still \( O(\log n) \)? Prove your answer.

(9) Consider the following algorithm.

Algorithm \texttt{DoIt} (array \( A \), \( \text{low} \), \( \text{high} \))

\begin{verbatim}
if (\text{low} == \text{high})
    done
\text{counter} = 0
for (\text{i} = \text{low} to \text{high} / 2)
    \text{counter} = \text{A}[\text{i}] + \text{counter}
if (\text{counter} == 27)
    print “foo!”
\text{DoIt} (\text{A}, (\text{low} + \text{high}) / 2)
\end{verbatim}

Use induction to prove that the running time of this algorithm is \( O(n) \)