Substitution Method
1. Guess the asymptotic complexity.
2. Prove your guess using induction
   (a) Assume inductive hypothesis holds for \( k < n \)
   (b) Try to prove the general case for \( n \)

Note: You MUST prove the EXACT inequality, you CANNOT ignore lower order terms.
If the proof fails, strengthen the inductive hypothesis and try again.

Master Method:
Let \( a \geq 1, b \geq 1 \) be constants, let \( f(n) \) be an asymptotically positive function, and let \( T(n) \) be defined on the nonnegative integers by the recurrence; for
\[
T(n) = aT\left(\frac{n}{b}\right) + f(n)
\]
Then \( T(n) \) has the following asymptotic bounds:

Case 1: \[
\left\{ \begin{array}{l}
    \frac{n^{\log_b a}}{f(n)} = \Omega(n^\varepsilon) \quad \varepsilon > 0 \\
    \frac{f(n)}{n^{\log_b a}} = \Theta(\log n)
\end{array} \right. \implies T(n) = \Theta(n^{\log_b a})
\]

Case 2: \[
\left\{ \begin{array}{l}
    \frac{f(n)}{n^{\log_b a}} = \Theta(\log^k n) \\
    \frac{n^{\log_b a}}{f(n)} = \Omega(n^\varepsilon) \quad \varepsilon > 0
\end{array} \right. \implies T(n) = \Theta(n^{\log_b a} \log^{k+1} n)
\]

Case 3: \[
\left\{ \begin{array}{l}
    \frac{f(n)}{n^{\log_b a}} = \Omega(n^\varepsilon) \quad a f(n/b) \leq cf(n) \quad \text{for } c < 1 \text{ and } \varepsilon > 0
\end{array} \right. \implies T(n) = \Theta(f(n))
\]

You can use the following algorithms covered in the lectures/book as indicated below:
- **INSERTION-SORT**(*A*, *n*): Insertion sort on *A*[1 ... *n*]
- **MERGE-SORT**(*A*, *p*, *r*): Merge sort on *A*[*p* ... *r*]
- **BINARY-SEARCH**(*A*, *p*, *r*, *key*): Binary search for *key* on *A*[*p* ... *r*]
- **H-PARTITION**(*A*, *p*, *r*): Hoare’s partitioning on *A*[*p* ... *r*], returns the last position of the region \( \leq \text{pivot} \)
- **L-PARTITION**(*A*, *p*, *r*): Lomuto’s partitioning on *A*[*p* ... *r*], returns the last position of the region \( \leq \text{pivot} \)
- **QUICKSORT**(*A*, *p*, *r*): Quicksort on *A*[*p* ... *r*]
- **R-QUICKSORT**(*A*, *p*, *r*): Randomized quicksort on *A*[*p* ... *r*]
- **R-H-PARTITION**(*A*, *p*, *r*): Randomized Hoare’s partitioning on *A*[*p* ... *r*] with a randomly selected pivot, returns the last position of the region \( \leq \text{pivot} \)
- **R-L-PARTITION**(*A*, *p*, *r*): Randomized Lomuto’s partitioning on *A*[*p* ... *r*] with a randomly selected pivot, returns the last position of the region \( \leq \text{pivot} \)
- **R-SELECT**(*A*, *p*, *r*, *i*): Randomized select algorithm on array *A*[*p* ... *r*], returns the value of the *i*th smallest element
- **MM-SELECT**(*S*, *n*, *i*): Median of medians algorithm on set *S* containing *n* elements, returns the value of the *i*th smallest element
- **HEAP-EXTRACT-MAX**(*A*, *n*): Extract max on the heap of *n* elements stored in *A*[1 ... *n*
- **MAX-HEAPIFY**(*A*, *i*, *n*): Max heapify on the *i*th element of the heap stored in *A*[1 ... *n*]
- **BUILD-MAX-HEAP**(*A*, *n*): Builds heap in *A*[1 ... *n*] using the *n* elements given in *A*[1 ... *n*]
- **HEAPSORT**(*A*, *n*): Heapsort on *A*[1 ... *n*]
- **MAX-HEAP-INSERT**(*A*, *key*, *n*): Inserts *key* into the heap stored in *A*[1 ... *n*]
- **HEAP-INCREASE-KEY**(*A*, *i*, *key*): Increases the value of the *i*th element of the heap to *key*
- **COUNTING-SORT**(*A*, *B*, *n*, *k*): Counting sort on *A*[1 ... *n*] with integers in the range \{1,2, ..., *k*\}, returns the sorted array *B*[1 ... *n*]