1. Prove the following polynomial is $\Theta(n^3)$.

\[ P(n) = 2n^3 - 5n^2 + 10n - 20 \]

(a) Prove $O(n^3)$:

\[
\begin{align*}
P(n) &= 2n^3 - 5n^2 + 10n - 20 \\
&\leq 2n^3 + 10n \\
&\leq 2n^3 + 10n \cdot \left(\frac{n}{10}\right)^2, \quad (\frac{n}{10}) \geq 1 \\
&\leq 2n^3 + 0.1n^3 \\
&\leq 2.1n^3, \quad n \geq 10
\end{align*}
\]

(b) Prove $\Omega(n^3)$:

\[
\begin{align*}
P(n) &\geq 2n^3 - 5n^2 - 20, \quad n \geq 0 \\
\geq 2n^3 - 5n^2 \left(\frac{n}{10}\right) - 20 \left(\frac{n}{10}\right)^3, \quad \frac{n}{10} \geq 1 \\
\geq 2n^3 - 0.5n^3 - 0.02n^3 \\
\geq 1.48n^3, \quad n \geq 10
\end{align*}
\]
2. Find the exact number of times (in terms of $n$) the innermost statement ($X = X + 1$) is executed in the following code. That is, find the final value of $X$. Then express the total running time in terms of $O(\cdot)$.

\[
X = 0;
\text{for } i = 1 \text{ to } 2n - 1
\quad \text{for } j = i \text{ to } 5n - i
\quad X = X + 1;
\]

\[
X = \sum_{i=1}^{2n-1} \sum_{j=i}^{5n-i} (1)
\]

\[
= \sum_{i=1}^{2n-1} (5n-2i+1)
\]

\[
= (2n-1) \frac{(5n-1) + (5n+1-2(2n-1))}{2}
\]

\[
= \frac{2n-1}{2} \cdot \left[ (5n-1) + (n+3) \right]
\]

\[
= (2n-1)(6n+2)/2
\]

\[
= (2n-1)(3n+1)
\]

\[
X = 6 \, n^2 - n - 1
\]

This is $O(n^2)$.

In fact, $\Theta(n^2)$. 

3. Consider the following divide-and-conquer algorithm (recursive function). Parameter \( i \) is the starting index of the array, and \( n \) is the number of elements. The initial call is \( \text{COMPUTE}(A, 0, n) \).

\[
\text{int COMPUTE (int A[], int i, int n) }
\]
\[
\text{\{ if } (n == 1) \text{ return } A[i];
\]
\[
n1 = \lfloor n/2 \rfloor; \quad \text{//Length of first half of array}
\]
\[
n2 = n - n1; \quad \text{//Length of second half of array}
\]
\[
C1 = \text{COMPUTE} (A, i, n1);
\]
\[
C2 = \text{COMPUTE} (A, i + n1, n2);
\]
\[
\text{return } (C1 \times C2)
\]

(a) Figure out what the function does. (What does it compute?) Explain briefly.

\[\text{Finds product of all elements.}\]
\[\text{That is, } A[0] \times A[1] \times \cdots \times A[n-1]\]
\[\text{This may be proved by induction}\]

(b) Let \( f(n) \) be the number of times the arithmetic operation \( C1 \times C2 \) is performed by this algorithm. Assume that \( n \) is a power of 2. Write a recurrence for \( f(n) \). Find the solution of the recurrence by repeated substitution.

\[
f(n) = \begin{cases} 2 \times f\left( \frac{n}{2} \right) + 1, & n \geq 2 \\ 0, & n = 1 \end{cases}
\]

\[f(n) = 1 + 2 \times f\left( \frac{n}{2} \right) = 1 + 2 \times \left( 1 + 2 \times f\left( \frac{n}{4} \right) \right)\]
\[= 1 + 2 + 4 \times f\left( \frac{n}{4} \right)\]
\[\frac{\sum_{k=0}^{\infty} 2^k}{\sum_{k=0}^{\infty} 2^k} = \frac{2^{k-1}}{2^k} = \frac{n-1}{n-1}
\]

(c) Now consider the general case where \( n \) is any integer. Write a recurrence for \( f(n) \). Guess the solution and prove it correct by induction.

\[
f(n) = \begin{cases} f\left( \frac{n}{2} \right) + f\left( n - \left\lfloor \frac{n}{2} \right\rfloor \right) + 1, & n \geq 2 \\ 0, & n = 1 \end{cases}
\]

\[f(n) = n-1 \quad \text{CLAIM}\]

\[\text{Proof: For } n = 1, \quad f(1) = 0 \quad \text{from rec}\]
\[f(1) = 1 - 1 = 0 \quad \text{from sol}\]
\[\text{So, sol is correct.}\]

For any \( n \geq 2 \), suppose \( f(m) = m-1, \quad \forall m < n \).

Then \( f(n) = \left\lceil \frac{n}{2} \right\rceil - 1 + \left\lceil \frac{n}{2} \right\rceil - 1 + 1 = n-1.\]
4. (a) Use Master Theorem to obtain the solution form for the following recurrence. Then find the exact solution. (Assume \( n \) is a power of 2.)

\[ a=8, \quad b=2, \quad \alpha = 1 \]

\[ T(n) = \begin{cases} 8T(n/2) + n, & n \geq 2 \\ 1, & n = 1 \end{cases} \]

\[ h = \log_b a = \log_2 8 = 3 \quad \Rightarrow \quad T(n) = A \cdot n^h + B \cdot n \]

So:
\[ T(n) = A \cdot n^3 + B \cdot n \]

To find \( A, B \):
\[ T(1) = 1 = A + B \]
\[ T(2) = 8T(1) + 2 = 8 + 2 = 10 = A \cdot 2^3 + B \cdot 2 \]

Solving:
\[ \begin{cases} A + B = 1 \\ 8A + 2B = 10 \end{cases} \]

\[ A = 4/3, \quad B = -1/3 \]

\[ T(n) = \frac{4}{3} n^3 - \frac{1}{3} n \]

(b) Use repeated substitution to find the solution of the following recurrence. (Assume \( n \) is a power of 2.)

\[ T(n) = \begin{cases} 8T(n/2) + n^2, & n \geq 2 \\ 1, & n = 1 \end{cases} \]

\[ T(n) = n + 8T(\frac{m}{2}) = n + 8 \left[ \left( \frac{m}{2} \right)^2 + 8T\left( \frac{m}{4} \right) \right] \]

\[ = n + \frac{8}{2^2} n^2 + 8^2 T\left( \frac{m}{4} \right) \]

\[ = n + \left( \frac{8}{2^2} \right) n^2 + 8^2 \left( \left( \frac{m}{4} \right)^2 + 8T\left( \frac{m}{8} \right) \right) \]

\[ = n^2 + \left( \frac{8}{4} \right) n^2 + \left( \frac{8}{4} \right)^2 n^2 + 8^3 T\left( \frac{m}{8} \right) \]

\[ = \ldots + 8^3 \left[ \left( \frac{m}{8} \right)^2 + 8T\left( \frac{m}{16} \right) \right] \]

\[ = n^2 + 2n^2 + 2^2 n^2 + 2^3 n^2 + \ldots + 2^k n^2 + 8^k T\left( \frac{m}{2^k} \right) \]

\[ T(1) = 1 \]

\[ = n^2 \left( 1 + 2 + 4 + 8 + \ldots + 2^{k-1} \right) + \left( 2^3 \right)^k \]

\[ = n^2 \left( 2^k - 1 \right) + 2^3 n^4 \]

\[ = n^2 (m-1) + n^3 = \boxed{2m^3 - n^2} \]
5. We have four sorted lists, each with \( n/4 \) elements. (Elements are real-valued.) We want to merge these lists into a single sorted list of \( n \) elements.

(a) First consider the following naive approach.

- Merge the first and second list into a sorted list of \( 2n/4 \) elements,
- Merge the result with the third list to get a sorted list of \( 3n/4 \) elements,
- Merge the result with the fourth list.

Analyze the worst-case number of key comparisons. (Find the exact worst-case number, not order of it.)

\[
\text{Total} = \left( \frac{2n}{4} - 1 \right) + \left( \frac{3n}{4} - 1 \right) + \left( \frac{4n}{4} - 1 \right)
\]

\[
= \frac{9n}{4} - 3
\]

\[
= 2.25n - 3
\]

(b) Describe a more efficient algorithm for this problem based on a divide-and-conquer technique. Use a diagram to help explain your algorithm. Analyze the worst-case number of key comparisons. (Again, find the exact worst-case number, not order of it.)

\[
\text{Total} = 2n - 3
\]