1. Prove the following sum is $\Theta(n^2 \log n)$.

\[ \sum_{i=1}^{n} i \log i \leq \sum_{i=1}^{n} n \log n \leq n^2 \log n \leq \Omega(n^2 \log n) \]

(a) Prove $O(n^2 \log n)$

\[ \sum_{i=1}^{n} i \log i \leq \sum_{i=1}^{n} n \log n \leq n^2 \log n \leq \Omega(n^2 \log n) \]

(b) Prove $\Omega(n^2 \log n)$

\[ \sum_{i=1}^{\left\lfloor \frac{n}{2} \right\rfloor} i \log i \geq \sum_{i=1}^{\left\lfloor \frac{n}{2} \right\rfloor} i \log i \geq \sum_{i=1}^{\left\lfloor \frac{n}{2} \right\rfloor} \left\lfloor \frac{n}{2} \right\rfloor \log \left\lfloor \frac{n}{2} \right\rfloor \]

\[ \geq \left(\frac{n}{2}\right)^2 \log \frac{n}{2} \geq \left(\frac{n}{2}\right)^2 (\log n - 1) \]

\[ = \Omega(n^2 \log n) \]
2. (a) Prove by induction that the solution of the following recurrence equation is \( T(n) = An^2 + Bn + C \), and determine the constants \( A, B, C \). Assume \( n \) is a power of 2.

\[
T(n) = \begin{cases} 
4T(n/2) + n + 6, & n \geq 2 \\
2, & n = 1.
\end{cases}
\]

**Case:** \( n = 1 \):

\[
T(1) = 2 = A + B + C
\]

**Hypothesis:** \( n \geq 2 \), suppose

\[
T\left(\frac{n}{2}\right) = A\left(\frac{n}{2}\right)^2 + B\left(\frac{n}{2}\right) + C
\]

Then,

\[
T(n) = 4\left[A\frac{n^2}{4} + B\frac{n}{2} + C\right] + n + 6
\]

\[
= An^2 + (2B + 1)n + (4C + 6)
\]

\[
= An^2 + B \cdot n + C
\]

\[
\Rightarrow \quad T(n) = 5n^2 - n - 2
\]

(b) Use master theorem to find the exact solution of the following recurrence. Assume \( n \) is a power of 2.

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

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

\[
h = \log_2 8 = 3
\]

\[
\therefore \quad T(n) = An^3 + Bn
\]

\[
n = 1: \quad T(1) = 1 = A + B
\]

\[
n = 2: \quad T(2) = 8T(1) + 2 = 10 = 8A + 2B
\]

\[
\Rightarrow \quad A = 4/3
\]

\[
B = -1/3
\]

\[
\therefore \quad T(n) = \frac{4}{3}n^3 - \frac{1}{3}n
\]
3. Find the solution of the following recurrence equation by repeated substitution, assuming $n = 2^k$ for some integer $k$.

$$T(n) = \begin{cases} 4T(n/2) + n, & n \geq 2 \\ 1, & n = 1. \end{cases}$$

$$T(n) = n + 4 \left( \frac{m}{2} \right)$$

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

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

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

$$= n + 2 \cdot n + 4^3 T \left( \frac{m}{2^3} \right)$$

$$= n + 2 \cdot n + 4^k T \left( \frac{m}{2^k} \right)$$

$$= n \left( 1 + 2 + 4 + \cdots + 2^{k-1} \right) + 4^k T \left( \frac{m}{2^k} \right)$$

$$= n \left( \frac{2^k - 1}{2 - 1} \right) + 4^k T \left( \frac{m}{2^k} \right)$$

$$= n \left( \frac{2^k - 1}{n - 1} \right) + 4^k T \left( \frac{m}{2^k} \right)$$

$$= n \left( \frac{2^k - 1}{n - 1} \right) + n^2$$

$$= n^2 - n + n^2$$

$$T(n) = 2n^2 - n$$
4. Given a sorted array of $n$ elements. We want to determine if there is a pair of elements $A[i]$ and $A[j]$, $i \neq j$, such that $A[i] = -A[j]$. (We want a Boolean function which returns a TRUE/FALSE answer.)

(a) Outline how this problem may be solved by using the binary-search algorithm, $BS(A, left, right, key)$. (Do not give the code for BS. It is a given function, which you call.) Analyze the time complexity of this approach.

For $i = 0 \text{ to } n-2$
\begin{align*}
\{ & BS(A, i+1, n-1, -A[i]) \\
& \text{if found return (TRUE)} \\
& \text{else return (FALSE)}
\end{align*}

Complexity: $O(n \log n)$

(b) Describe a more efficient $O(n)$ algorithm to solve this problem. Give the pseudocode. Explain how the algorithm works, and provide a numerical illustration.

We do inward scan.

If $(A[\text{left}] + A[\text{right}] \geq 0)$
else if $(A[\text{left}] + A[\text{right}] < 0)$

\begin{align*}
\text{left} &\leftarrow \text{left} + 1 \\
\text{right} &\leftarrow \text{right} - 1
\end{align*}

\begin{align*}
\text{left} &\leftarrow 0 \\
\text{right} &\leftarrow n-1
\end{align*}

While ($\text{left} < \text{right}$)
\begin{align*}
\{ & \text{if } (A[\text{left}] + A[\text{right}] \geq 0) \\
& \text{then } \text{right} \leftarrow \text{right} - 1; \\
& \text{else } \text{left} \leftarrow \text{left} + 1; \\
& \text{else return (TRUE)}
\}
\end{align*}

\text{return (false)}
5. This problem deals with computing $x^n$, where $x$ is real and $n$ is an integer, by repeated multiplications.

(a) It is possible to perform this computation with $O(\log n)$ multiplications. Write an efficient recursive algorithm (pseudocode), $\text{POWER}(x, n)$, to compute $x^n$.

```
real \text{POWER} (real x, int n)
{
    if (n == 1) return x;
    T = \text{POWER} (x, \lfloor n/2 \rfloor);
    T = T \times T;
    if (n is odd) T = T \times x;
    return (T)
}
```

(b) Let $f(n)$ be the worst-case number of multiplications used by your algorithm. Write a recurrence for $f(n)$ and find the solution.

$$
\begin{align*}
f(n) &= \begin{cases} 
0 & n = 1 \\
f\left(\lfloor \frac{n}{2} \rfloor\right) + 2, & n \geq 2
\end{cases}
\end{align*}
$$

$$
f(n) = 2 \lfloor \log n \rfloor
$$

Easy to prove by induction.
6. Consider the algorithm BUILD-HEAP, which starts with an array A[1..n] of n random elements and builds a valid min-heap of n elements.

(a) The following algorithm is an inefficient way of building the heap, by a series of $n - 1$ insert operations. Analyze the time complexity of this algorithm.

for $k = 2$ to $n$
Insert $A[k]$ into the heap $A[1..k-1]$ and make a heap with one more element $A[1..k]$.

\[
\sum_{k=2}^{n} \log k = \Theta(n \log n)
\]

(b) Now give a more efficient recursive algorithm (pseudocode) for BUILD-HEAP based on divide-and-conquer approach, using PUSHDOWN.

Build (upw A[1..n])
\{ if ($2r < n$) return; // IT IS LEAF OR NIL
Build (A, 2r, n); // Build left subtree
Build (A, 2r+1, n); // Build right subtree
Pushdown (A, r, n); \}

(c) What is the time complexity of this efficient BUILD-HEAP? Use a recurrence equation for the analysis. Use master theorem as a guide to guess the solution form. Don't bother to prove the correctness of it.

Let $f(n) = \text{worst case number of key swaps.}$

\[
f(n) = \begin{cases} 2f\left(\frac{n}{2}\right) + 2 \log n, & n > 1 \\ \Theta, & n = 1 \end{cases}
\]

\[
f(n) = An + B \log n + C
\]
7. Starting with an empty AVL tree, insert the following sequence of elements into it. Show the result (which must be balanced) after each insertion.

50, 60, 70, 65, 80, 90, 62, 69

```
50
 / 
45 60
 / 
70 60
 / 
50 70
 / 
65 80
 / 
50 65
 / 
62 69
```