Question 1

a) Write an algorithm in pseudocode for computing the $k$’th power of a square matrix of dimension $n \times n$. The runtime of your algorithm should be $o(kn^3)$. Perform an exact analysis and express the time complexity in big O notation.

Solution:
The naive algorithm for calculating $k$’th power of a matrix is to perform $k - 1$ matrix multiplications. The standard matrix multiplication algorithm for two square matrices of dimension $n$ takes $O(n^3)$ time. Hence, we should find another way for computing the $k$’th power since the naive approach does not yield an $o(kn^3)$ algorithm.

We will use the method of exponentiation by squaring to reduce the number of matrix multiplications.

<table>
<thead>
<tr>
<th>Function: MatrixPower($A, k$)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> $A$ ($n \times n$ dimensional matrix)</td>
</tr>
<tr>
<td>$k$ (a positive integer)</td>
</tr>
<tr>
<td><strong>Output:</strong> $M$ ($k$’th power of $A$)</td>
</tr>
<tr>
<td>if $k=1$ then</td>
</tr>
<tr>
<td>$M \leftarrow A$</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>if $n \mod 2 = 0$ then</td>
</tr>
<tr>
<td>$M \leftarrow$ MatrixPower($A \cdot A, n/2$)</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>$M \leftarrow A \cdot$ MatrixPower($A \cdot A, (n-1)/2$)</td>
</tr>
<tr>
<td>end if</td>
</tr>
<tr>
<td>end if</td>
</tr>
<tr>
<td>return $M$</td>
</tr>
<tr>
<td>end MatrixPower</td>
</tr>
</tbody>
</table>
Matrix multiplication algorithm is the classical $O(n^3)$ algorithm. Let us analyze the complexity of the MatrixPower algorithm. We can choose matrix multiplication as the basic operation.

Since the algorithm is recursive, we will write a recursion to find its complexity. Let $f(k, n)$ denote the complexity of MatrixPower and $t(n)$ denote the complexity of matrix multiplication.

Best case occurs when $k = 2^n$.

\[
\begin{align*}
  f(1, n) &= 0 \\
  f(k, n) &= f(k/2, n) + t(n) \\
            &= f(k/4, n) + 2t(n) \\
            &= f(k/8, n) + 3t(n) \\
            &\quad \vdots \\
            &= f(1, n) + \log_2 k \cdot t(n) \\
            &\in O(\log k \cdot n^3)
\end{align*}
\]

Worst case occurs when $k = 2^n - 1$

\[
\begin{align*}
  f(1, n) &= 0 \\
  f(k, n) &= f(k/2, n) + 2t(n) \\
            &= f(k/4, n) + 4t(n) \\
            &= f(k/8, n) + 6t(n) \\
            &\quad \vdots \\
            &= f(1, n) + 2\log_2 k \cdot t(n) \\
            &\in O(\log k \cdot n^3)
\end{align*}
\]

We conclude that $f(k, n) \in O(\log k \cdot n^3)$. Note that $f(k, n) \in o(k \cdot n^3)$.

b) Let $A$ and $B$ be integer square matrices of dimension $(n + 2) \times (n + 2)$ which have the following form:

\[
\begin{bmatrix}
  1 & a & c \\
  0 & I_n & b \\
  0 & 0 & 1
\end{bmatrix}
\]

where $a$ is an $n$ dimensional row vector, $b$ is an $n$ dimensional column vector, $c \in \mathbb{Z}$, and $I_n$ is the identity matrix of dimension $n$. An example for dimension 7 can be given as follows:

\[
\begin{bmatrix}
  1 & 3 & 4 & 0 & 1 & -2 & 5 \\
  0 & 1 & 0 & 0 & 0 & 0 & -3 \\
  0 & 0 & 1 & 0 & 0 & 0 & 5 \\
  0 & 0 & 0 & 1 & 0 & 0 & 1 \\
  0 & 0 & 0 & 0 & 1 & 0 & 4 \\
  0 & 0 & 0 & 0 & 0 & 1 & 3 \\
  0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]
Describe a method for multiplying $A$ and $B$ which requires $O(n)$ time.

**Solution:** When two matrices of the given form are multiplied, the resulting matrix has the following form.

$$
\begin{bmatrix}
1 & a_1 & c_1 \\
0 & I_n & b_1 \\
0 & 0 & 1
\end{bmatrix} \cdot
\begin{bmatrix}
1 & a_2 & c_2 \\
0 & I_n & b_2 \\
0 & 0 & 1
\end{bmatrix} =
\begin{bmatrix}
1 & a_1 + a_2 & c_1 + a_1 \cdot b_2 + c_2 \\
0 & I_n & b_1 + b_2 \\
0 & 0 & 1
\end{bmatrix}
$$

Therefore it is enough to compute $a_1 + a_2$, $a_1 \cdot b_2 + c_2$ and $b_1 + b_2$.

Adding two vectors of dimension $n$ requires $n$ additions. Dot product of two vectors requires $n$ multiplications and $n-1$ additions. Adding three scalars requires 2 additions. In total, we need to perform $3n + 1$ basic operations which requires $O(n)$ time.

**Question 2**

Consider the given function $f(n)$ and determine whether the following cases are true or false. Justify your answers formally.

$$f(n) = n^2 \log n + n^3 \sum_{i=1}^{n} \frac{1}{i} + n^3 \sum_{i=0}^{n} \frac{1}{2^i}$$

a) $f(n) \in O(n^4)$

b) $f(n) \in \theta(n^4)$

c) $f(n) \in \Omega(n^3 \log n)$

d) $f(n) \in o(n^4 \log n)$

**Solution:** Let us analyze the given function $f(n)$.

$\sum_{i=1}^{n} \frac{1}{i}$ is the harmonic series and evaluates to $\sim \log n$.

$\sum_{i=0}^{n} \frac{1}{2^i}$ is the geometric series and evaluates to $\frac{1-(1/2)^{n+1}}{1-(1/2)} = 2 - (1/2)^n$.

Therefore, $f(n) = n^2 \log n + n^3 \log n + n^3 (2 - (1/2)^n)$

a) $f(n) \in O(n^4)$ **True**

$$\lim_{n \to \infty} \frac{f(n)}{n^4} = 0$$

By Ratio Limit Theorem, $f(n) \in o(g(n)) \Rightarrow f(n) \in O(g(n))$
b) \( f(n) \in \Theta(n^4) \) **False**

\[
\lim_{n \to \infty} \frac{f(n)}{n^4} = 0 \text{ implies that } f(n) \notin \Theta(n^4) \text{ by Ratio Limit Theorem.}
\]

c) \( f(n) \in \Omega(n^3 \log n) \) **True**

Let \( n_0 = 1 \) and \( c = 1 \). \( cn^3 \log n \leq f(n) \) for all \( n \geq n_0 \). The assertion is true by the definition of \( \Omega \).

d) \( f(n) \in o(n^4 \log n) \) **True**

\[
\lim_{n \to \infty} \frac{f(n)}{n^4 \log n} = 0 \text{ implies that } f(n) \in o(n^4 \log n) \text{ by Ratio Limit Theorem.}
\]