Matrix multiplication:

- With the direct method, it is possible to multiply two matrices of size $n \times n$ with exactly $n^3$ scalar multiplications and $n^2(n - 1)$ scalar additions.

- With the Strassen method, it is possible to multiply two matrices of size $2 \times 2$ with exactly 7 scalar multiplications and 18 scalar additions.

- The complexity for multiplying two matrices of size $n \times n$ using the Strassen algorithm is

  $$\Theta \left(n^{\log_2 7}\right) = \Theta \left(n^{2.807354922...}\right) = O \left(n^{2.81}\right)$$
1. Use the recursive Strassen method to compute the **exact** number of scalar multiplications and scalar additions (subtractions are considered as additions) to multiply two matrices of size $4 \times 4$, two matrices of size $8 \times 8$, and two matrices of size $16 \times 16$.

**Solution for matrices of size $4 \times 4$:**

- 7 multiplications of two $2 \times 2$ matrices are required to multiply two $4 \times 4$ matrices. Each multiplication of two $2 \times 2$ matrix requires 7 scalar multiplications. All together,
  \[ 49 = 7 \cdot 7 = 7^2 \]
  scalar multiplications are required to multiply two $4 \times 4$ matrices.

- 7 multiplications of two $2 \times 2$ matrices and 18 additions of two $2 \times 2$ matrices are required to multiply two $4 \times 4$ matrices. Each multiplication of two $2 \times 2$ matrices requires 18 scalar additions and each addition of two $2 \times 2$ matrices requires 4 scalar additions. All together,
  \[ 198 = 7 \cdot 18 + 18 \cdot 4 \]
  scalar additions are required to multiply two $4 \times 4$ matrices.

**Solution for matrices of size $8 \times 8$:**

- 7 multiplications of two $4 \times 4$ matrices are required to multiply two $8 \times 8$ matrices. Each multiplication of two $4 \times 4$ matrix requires $7^2$ scalar multiplications. All together,
  \[ 343 = 7 \cdot 7^2 = 7^3 \]
  scalar multiplications are required to multiply two $8 \times 8$ matrices.

- 7 multiplications of two $4 \times 4$ matrices and 18 additions of two $4 \times 4$ matrices are required to multiply two $8 \times 8$ matrices. Each multiplication of two $4 \times 4$ matrices requires 198 scalar additions and each addition of two $4 \times 4$ matrices requires 16 scalar additions. All together,
  \[ 1674 = 7 \cdot 198 + 18 \cdot 16 \]
  scalar additions are required to multiply two $8 \times 8$ matrices.

**Solution for matrices of size $16 \times 16$:**

- 7 multiplications of two $8 \times 8$ matrices are required to multiply two $16 \times 16$ matrices. Each multiplication of two $8 \times 8$ matrix requires $7^3$ scalar multiplications. All together,
  \[ 2401 = 7 \cdot 7^3 = 7^4 \]
  scalar multiplications are required to multiply two $16 \times 16$ matrices.

- 7 multiplications of two $8 \times 8$ matrices and 18 additions of two $8 \times 8$ matrices are required to multiply two $16 \times 16$ matrices. Each multiplication of two $8 \times 8$ matrices requires 1674 scalar additions and each addition of two $8 \times 8$ matrices requires 64 scalar additions. All together,
  \[ 12870 = 7 \cdot 1674 + 18 \cdot 64 \]
  scalar additions are required to multiply two $16 \times 16$ matrices.
2. Let $M(n)$ be the number of multiplications used by the Strassen algorithm for $n = 2^k$ ($k \geq 1$). What is the recursive formula for $M(n)$? What is the exact solution for $M(n)$? Note that in this problem the number of additions is ignored.

**Solution:** The recursive call involves no additional multiplications beyond those performed by the 7 multiplications of the smaller matrices. Trivially, only one multiplication is performed when multiplying two $1 \times 1$ matrices. As a result, the recursive formula is,

\[
M(1) = 1 \\
M(n) = 7M\left(\frac{n}{2}\right)
\]

**Proposition:** The solution to the above recursion is

\[
T(n) = 7^{\log_2(n)} = n^{\log_2(7)}
\]

**Proof by induction:** For $n = 1$, it follows that $M(1) = 1$ and $7^{\log_2(1)} = 7^0 = 1$.

For $n \geq 1$, assume that $M(n) = 7^{\log_2(n)}$ and prove that $M(2n) = 7^{\log_2(2n)}$.

\[
M(2n) = 7M(n) \\
= 7 \cdot 7^{\log_2(n)} \\
= 7^{\log_2(n)+1} \\
= 7^{\log_2(2n)}
\]

**Top down proof:** Apply the recursion until $n = 1$,

\[
M(n) = 7 \cdot M\left(\frac{n}{2}\right) \\
= 7 \cdot 7 \cdot M\left(\frac{n}{4}\right) \\
= 7 \cdot 7 \cdot 7 \cdot M\left(\frac{n}{8}\right) \\
\vdots \\
= 7^i \cdot M\left(\frac{n}{2^i}\right) \\
\vdots \\
= 7^{\log_2(n)} \cdot M\left(\frac{n}{2^{\log_2(n)}}\right) \\
= 7^{\log_2(n)} \cdot M\left(\frac{n}{n}\right) \\
= 7^{\log_2(n)} \cdot M(1) \\
= 7^{\log_2(n)} \cdot 1 \\
= 7^{\log_2(n)}
\]

**Remark:** The above line “$= 7^i \cdot M\left(\frac{n}{2^i}\right)$” needs a proof by induction. However, this method helps getting the guess that allows a direct proof by induction.
3. Let \( A(n) \) be the number of additions used by the Strassen algorithm for \( n = 2^k \) \((k \geq 1)\). What is the recursive formula for \( A(n) \)? What is the exact solution to \( A(n) \)? Note that in this problem the number of multiplications is ignored.

**Solution:** The recursive call involves additions that are performed by the 7 multiplications of the smaller matrices and \( 18 \cdot \frac{n^2}{2} \) scalar additions due to the 18 additions of \((n/2) \times (n/2)\) matrices. Trivially, no additions are needed when multiplying two 1 \times 1 matrices. As a result, the recursive formula is,

\[
A(1) = 0 \\
A(n) = 7A(n/2) + 18(n^2/4)
\]

**Proposition:** The solution to the above recursion is

\[
A(n) = 6 \cdot 7^{\log_2(n)} - 6 \cdot 4^{\log_2(n)} = 6 \cdot n^{\log_2(7)} - 6 \cdot n^2
\]

**Proof by induction:** for \( n = 1 \), it follows that \( A(1) = 0 \) and \( 6 \cdot 7^{\log_2(1)} - 6 \cdot 1^2 = 0 \).

For \( n \geq 1 \), assume that \( A(n) = 6 \cdot 7^{\log_2(n)} - 6 \cdot n^2 \) and prove that \( A(2n) = 6 \cdot 7^{\log_2(2n)} - 6 \cdot (2n)^2 \).

\[
A(2n) = 7A(n) + 18 \cdot ((2n)^2/4) \\
= 7(6 \cdot 7^{\log_2(n)} - 6 \cdot n^2) + 18 \cdot n^2 \\
= 6 \cdot 7^{\log_2(n)+1} - 42 \cdot n^2 + 18 \cdot n^2 \\
= 6 \cdot 7^{\log_2(n)+1} - 24 \cdot n^2 \\
= 6 \cdot 7^{\log_2(2n)} - 6 \cdot (2n)^2
\]

**Top down proof:** Apply the recursion until \( n = 1 \),

\[
A(n) = 7 \cdot A(n/2) + 18(n^2/4) \\
= 7 \cdot 7 \cdot A(n/4) + 7 \cdot 18(n^2/16) + 18(n^2/4) \\
= 7 \cdot 7 \cdot 7 \cdot A(n/8) + 7 \cdot 7 \cdot 18(n^2/64) + 7 \cdot 18(n^2/16) + 18(n^2/4) \\
\vdots \\
= 7^i \cdot A(n/(2^i)) + 7^{i-1} \cdot 18(n^2/4^i) + \cdots + 7 \cdot 18(n^2/16) + 18(n^2/4) \\
= 7^i \cdot A(n/(2^i)) + ((7/4)^{i-1} + \cdots + (7/4)^1 + (7/4)^0) \cdot (18/4) \cdot n^2 \\
= 7^i \cdot A(n/(2^i)) + ((7/4)^{i-1} - (7/4 - 1)) \cdot (18/4) \cdot n^2 \\
= 7^i \cdot A(n/(2^i)) \cdot (4/3)(18/4) \cdot n^2 \\
= 7^i \cdot A(n/(2^i)) \cdot (7/4)^{i-1} \cdot 6 \cdot n^2 \\
\vdots \\
= 7^{\log_2(n)} \cdot A(n/(2^{\log_2(n)})) + (7/4)^{\log_2(n)-1}) \cdot 6 \cdot n^2 \\
= 7^{\log_2(n)} \cdot A(1) + (7^{\log_2(n)} / 4^{\log_2(n)}) \cdot 6 \cdot n^2 - 6 \cdot n^2 \\
= 0 + (7^{\log_2(n)} / n^{\log_2(4)}) \cdot 6 \cdot n^2 - 6 \cdot n^2 \\
= (7^{\log_2(n)} / n^2) \cdot 6 \cdot n^2 - 6 \cdot n^2 \\
= 6 \cdot 7^{\log_2(n)} - 6 \cdot n^2 \\
= 6 \cdot n^{\log_2(7)} - 6 \cdot n^2
\]

**Remark:** The identity \( 1 + q + q^2 + \cdots + q^{i-1} = (q^i - 1)/(q - 1) \) was used for \( q = 7/4 \). Other algebraic justifications for the equalities are omitted.
4. Assume that \( n \) is a power of 3. Assume that there is a way to multiply two matrices of size \( 3 \times 3 \) with 25 scalar multiplications and 50 scalar additions. What is the complexity of a recursive algorithm that is based on this method?

**Solution:**

The following is the recursive formula for the complexity of this algorithm,

\[
\begin{align*}
T(1) &= 1 \\
T(n) &= 25T(n/3) + \Theta(n^2)
\end{align*}
\]

By the master theorem, the solution to this recursion is

\[
T(n) = \Theta \left( n^{\log_3(25)} \right) = O \left( n^{2.93} \right) = \Omega \left( n^{2.9299} \right)
\]

since \( \log_3(25) = 2.929947 \ldots \)

5. What should be the number of scalar multiplications in multiplying two matrices of size \( 3 \times 3 \) in order to get a recursive way to multiply two matrices of size \( n \times n \) with a better complexity than the original Strassen algorithm?

**Solution:**

Denote by \( s \) the number of scalar multiplications in multiplying two matrices of size \( 3 \times 3 \). Then the following is the recursive formula for the complexity of this algorithm,

\[
\begin{align*}
T(1) &= 1 \\
T(n) &= sT(n/3) + \Theta(n^2)
\end{align*}
\]

By the master theorem, the solution to this recursion is

\[
T(n) = \Theta \left( n^{\log_3(s)} \right)
\]

Since \( \log_3(22) > 2.81 > \log_2(7) \) and \( \log_3(21) < 2.78 < \log_2(7) \), it follows that in order to get a better complexity than the original Strassen algorithm \( s \) must be at most 21.

6. Assume that \( n \) is a power of 70. There exists a way to multiply two matrices of size \( 70 \times 70 \) with 143640 scalar multiplications and \( \alpha \) scalar additions for some constant \( \alpha \). What is the complexity of a recursive algorithm that is based on this method? Write the recursive formula and use the master theorem to solve it.

**Solution:**

The following is the recursive formula for the complexity of this algorithm,

\[
\begin{align*}
T(1) &= 1 \\
T(n) &= 143640T(n/70) + \Theta(n^2)
\end{align*}
\]

By the master theorem, the solution to this recursion is

\[
T(n) = \Theta \left( n^{\log_{70}(143640)} \right) = O \left( n^{2.796} \right) = \Omega \left( n^{2.795} \right)
\]

since \( \log_{70}(143640) = 2.795122689 \ldots \)