Problem 1

In this problem, we will derive the Tree Theorem together. Let’s start with a divide-and-conquer recurrence relation of the form:

\[ f(n) = a \cdot f\left(\frac{n}{b}\right) + c \cdot n^d \]

where \( n = b^k \) for some \( k \in \mathbb{Z}^+ \), \( a \geq 1 \), \( b > 1 \), \( b \in \mathbb{Z} \), \( c > 0 \), \( d \geq 0 \) and \( a, c, d \in \mathbb{R} \). Recall that \( a \) refers to the number of subproblems created on each recursive step, \( b \) is the fraction by which the problem size decreases on each recursive function call, and \( n^d \) characterizes the amount of work done (in the recursive step) separate from the recursive function call.

Consider drawing a stack diagram for the recursive function calls. This will look like a tree, in which each internal vertex has \( a \) children. Let \( k \) represent the number of levels we have traversed down the tree.

(a) How many vertices are there at level \( k \)?

(b) What is the size of the problem at level \( k \)?

(c) How much work is done within a single vertex of the tree at level \( k \)?

(d) How much work is done at level \( k \)?

(e) At what level will the original array of length \( n \) have been broken up into arrays of length 1?

(f) How much total work is done?

(g) Simplify your expression for part (d) for the case when \( n \) gets really really big. Do this for three cases: (i) \( a < b^d \), (ii) \( a = b^d \) and (iii) \( a > b^d \).

Problem 2

Recall how to multiply two matrices \( A \) and \( B \) - in the problem, we’ll write out matrix multiplication as an algorithm! We’ll restrict our attention to square matrices of size \( n \times n \) where \( n = 2^k \) for some nonnegative integer \( k \). A naive version of the algorithm can be written as follows:

```plaintext
matmul(A, B)

input: n \times n matrices A and B
output: n \times n matrix C = AB.
1  C ← 0
2  for i = 1 → n
3     for j = 1 → n
4       for k = 1 → n
5          c_{i,j} = c_{i,j} + a_{i,k} * b_{k,j}
```

(a) Give a big-oh estimate for the number additions and multiplications performed by the `matmul` algorithm?
(b) Consider a divide-and-conquer (DAC) method for multiplying two matrices. We will represent the matrix multiplication as a multiplication of $2 \times 2$ block matrices, in which each block is an $n/2 \times n/2$ matrix. These blocks are denoted by $a$, $b$, $c$, $d$, $e$, $f$, $g$, $h$, $s$, $t$, $u$, $v$ below. The matrix multiplication then becomes:

$$
\begin{bmatrix}
  r & s \\
  t & u
\end{bmatrix} =
\begin{bmatrix}
  a & b \\
  c & d
\end{bmatrix}
\begin{bmatrix}
  e & f \\
  g & h
\end{bmatrix} =
\begin{bmatrix}
  (ae + bg) & (af + bh) \\
  (ce + dg) & (cf + dh)
\end{bmatrix}
$$

The DAC approach thus consists of computing $(ae + bg)$, $(af + bh)$, $(ce + dg)$ and $(cf + dh)$ recursively. How many $n/2 \times n/2$ matrix multiplications are performed on each recursive function call? How many additions are performed? Develop a recurrence relation to describe the number of operations (counting only additions and multiplications) for the DAC algorithm. Use the Tree Method to express the number of operations of the DAC `matmul` algorithm in big-oh notation.

(c) The DAC algorithm of part (b) can be improved by reducing the number of block multiplications performed in the recursive step. This is know as Strassen’s algorithm, which computes the $n \times n$ matrix multiplication during the recursive step as follows:

$$
\begin{align*}
P_1 &= a \cdot (f - h) \\
P_2 &= (a + b) \cdot h \\
P_3 &= (c + d) \cdot e \\
P_4 &= d \cdot (g - e) \\
P_5 &= (a + d) \cdot (e + h) \\
P_6 &= (b - d) \cdot (g + h) \\
P_7 &= (a - c) \cdot (e + f)
\end{align*}
$$

then

$$
\begin{align*}
r &= P_5 + P_4 - P_2 + P_6 \\
s &= P_1 + P_2 \\
t &= P_3 + P_4 \\
u &= P_5 + P_1 - P_3 - P_7
\end{align*}
$$

How many multiplications are performed in the recursive step? How many additions/subtractions? Develop a recurrence relation to describe the number of operations for Strassen’s algorithm, and use the Tree Method to bound the number of operations using big-oh notation.

(d) Verify that Strassen’s algorithm computes the correct matrix-matrix multiplication on each recursive step.