MA398 Matrix Analysis and Algorithms: Exercise Sheet 5

1. Consider an algorithm that performs a sorting operation on an array of size n. This algorithm has a cost of $C(n) = n \log n$ operations. As $n \to \infty$, what can be inferred about the time complexity of this algorithm? Furthermore, if this sorting algorithm is to be run on a parallel computer architecture, what considerations might come into play in terms of the computational cost?

2. Consider a simple Python function that performs a multiplication operation on two matrices A and B of size $n \times n$. Write a Python function using numpy's built-in np.dot() function to perform this operation and a function to compute the cost of this algorithm. Assume each multiplication and addition operation counts as a single operation.

   In your computation, consider:

   - The cost of multiplying two elements (one operation)
   - The cost of adding two elements (one operation)
   - The nested loops required to perform the multiplication (consider the number of times the operation is repeated).

   Does your cost calculation match with the standard cost of matrix multiplication, which is $O(n^3)$? Can you explain why?

   For simplicity, you can assume that all matrices are square and of the same dimension, $n \times n$.

3. Implement Strassen's algorithm for matrix multiplication in Python. Your Python function should take two square matrices as input and return the product matrix as output.

   For the input, you may assume that the matrices are 2D numpy arrays of shape (n, n), where n is a power of 2. Also, assume that each element of these matrices is a complex number.

   After implementing the function, test it with two example matrices and verify the output by comparing it to the output from numpy's built-in matrix multiplication function (np.dot()).