COMP 301 Analysis of Algorithms, Fall 2021
Instructor: Zafer Aydın
Lab Assignment 5

## Introduction

In this lab you will implement algorithms for matrix multiplication. Submit your answers to the questions below in a text file (e.g. Word document). Name your file in name_surname.docx format. Submit your solution document and Java codes as a compressed folder (.zip, .rar) in name_surname format to Canvas.

You can use the code templates in `matrix.java` in this lab.

## Problem Statement

Given two matrices $A$ and $B$ each with size $n \times n$ compute $C = A.B$, which is the product of $A$ and $B$.

## Assignment

1. (a) Implement the standard matrix multplication algorithm given below in Java. You can use 2D arrays to represent matrices. You can define array $C$ outside of your method and pass it as input to that method. You can use the code template in `matrix.java`.

```
SQUARE-MATRIX-MULTIPLY(A, B)
n = A.rows
let C be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        c_ij = 0
        for k = 1 to n
            c_ij = c_ij + a_ik · b_kj
return C
```

(b) Verify that your method works correctly for $A = \begin{bmatrix} 1 & 3 \\ 7 & 5 \end{bmatrix}$, $B = \begin{bmatrix} 6 & 8 \\ 4 & 2 \end{bmatrix}$, where $C$ should be $= \begin{bmatrix} 18 & 14 \\ 62 & 66 \end{bmatrix}$.

(c) Randomly generate $A$ and $B$ such that their sizes are 16 by 16 and elements are random integers from 0 to 99. Compue $C = A.B$ using the method you implemented in part (a). Report the time it takes to compute $C$.

(d) Repeat part (c) this time for arrays of size 64 by 64. Report the time it takes to compute $C = A.B$. How much did the time increase as compared to part (c)?

2. (a) Implement the recursive version of matrix multiplication algorithm given below that uses the divide and conquer strategy. You can define array $C$ outside of your method and pass it as input to that method. Use indexing to partitioning the matrices into four sub-blocks instead of defining new arrays each time. You can use the code template in `matrix.java`.

```
SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)
n = A.rows
let C be a new n × n matrix
if n == 1
    c11 = a11 · b11
else partition A, B, and C as in equations (4.9)
    C11 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B11)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B21)
    C12 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B12)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B22)
    C21 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B11)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B21)
    C22 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B12)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B22)
return C
```

(b) Verify that your method works correctly for $A = \begin{bmatrix} 1 & 3 \\ 7 & 5 \end{bmatrix}$, $B = \begin{bmatrix} 6 & 8 \\ 4 & 2 \end{bmatrix}$, where $C$ should be $= \begin{bmatrix} 18 & 14 \\ 62 & 66 \end{bmatrix}$.

(c) Use the same random matrix as in question 1(c) and compute $C = A.B$ using the method implemented in question 2(a). Report the time it takes to compute $C$.

(d) Repeat part (c) this time for arrays of size 64 by 64. Report the time it takes to compute $C = A.B$. How much did the time increase as compared to part (c)?