Dr. Hasmik Gharibyan
CSC/CPE349

Project2: Matrix Multiplication Algorithms

PART 2 (*85% of the credit for Project2*): Divide-and-conquer algorithms

Deadlines: submit your files **electronically** by **midnight** (end of day) on **Friday, 04/23/21.**

Late submission: you can submit your work within 24 hours after deadline (penalty applies):
by 2 a.m. on 4/24/21 for **8-point penalty**; from 2:01 a.m. till *noon* on 4/24/21 for **21-point penalty**, from 12:01 p.m. till *midnight* (end of the day) on 4/24/21 for **42 point penalty.**

Programming Language and Environment: All programs should be written in ***Java***. You can use any Java program development environment you are comfortable with (any text editor, any IDE you like). However, you must make sure your programs can be compiled and executed from the command line on our departmental servers as the grading will be done in this environment.

DON'T CHANGE any class/method names, sequence of steps in the assignment. Failure to do so will hinder the grading process and will end in penalty.

Goals of the assignment:

1. Implement divide-and-conquer algorithms.
2. Implement two classic algorithms for matrix multiplication.
2. Handle intricate details of recursive matrix manipulation – identifying and working with segments of two-dimensional arrays, constructing two-dimensional arrays from smaller ones.

I. Design and implement a class named *MatrixProduct.*

The only content of *MartixProduct* class should be the implementation of the following 2 classic algorithms for matrix multiplication: the ***"simple" divide-and-conquer* algorithm** and the ***Strassen's* algorithm** (*see the lecture handout/slides for these algorithms*). Both algorithms take two square matrices A(n×n) and B(n×n) and return a product-matrix C(n×n); n is required to be a power of 2 (you need it to be divisible by 2 without remainder at each level of recursion).

Both algorithms are recursive so **for *each* algorithm** you need to have the following members:

- a *public* method that will get two ***square*** matrices A and B (both of the ***same size***) and will **initiate** the recursion. It will return the final product-matrix (of the same size as A and B). *The **signature** of this method for each algorithm (i.e. the name, the number and type of parameters, and the return-value type) is specified on the top of the next page*.*

  Important: The first thing to do in this method is the **validity check**, namely:
  1. Check if A and B are square matrices of the same size.
  2. Check if the matrix size (i.e. the n-value) is a power of 2.

  IF at least one requirement is **not satisfied**, throw ***IllegalArgumentException*** exception.

- a *private* method that carries the recursion (*see the lecture handout/slides*)

- as many *private* supporting methods as you need to do the work.

Note: the signature of *private* methods (name, type and number of parameters, return value's type) is not mandated, but to stay on track and to avoid confusion, follow the path of the lecture handout/slides. Do **NOT** seek help on the internet – all you need is in the handout.

* **The above-mentioned two *public* methods should have the following headers:**

public static int[][] matrixProduct_DAC(int[][] A, int[][] B)
//Compute and return the product of A, B matrices using "**simple" DAC algorithm** presented in class.

public static int[][] matrixProduct_Strassen(int[][] A, int[][] B)
//Compute and return the product of A, B matrixes using **Strassen's algorithm** presented in class.

Note: both algorithms are working with integer values.

II. Design and implement application class(es) to test your two algorithms.

You can make a single application class to test both algorithms, or you can create two separate classes – one for each algorithm. An application class will contain a *main* method very similar to the *main* method in your *Project2_part1* assignment.

Reminder: when testing the functionality of your algorithms, make sure the input matrices are square and of the same size, and that their size is a power of 2 (the two mentioned algorithms can only multiply matrices that satisfy these conditions).

ATTENTION!!!

Do NOT submit this application class (or classes) – it only serves as a testing tool for you.

The grader will run his own driver to test your algorithms.

Submitting your work:

Turn in the *MatrixProduct* source file electronically via "*handin*" procedure by the deadline (see the top of the first page of this document for the due date and time):

The account *id* is: ***hg-cpe103***
The *name* of the assignment is: ***Project2***
The *name* of the assignment for late submissions is: ***Project2_late.***

Important:

1. Your programs must **compile and run from command line** on our departmental servers. So, before submitting, make sure your programs compile and run on lab machines.
2. Do not create packages for your source code, do not zip, and do not use folders for your submission – all these complicate the automated grading process.
3. Each file must have a **comment-header** which should include both authors' **names** and **ids** (as in *id@calpoly.edu*), as well as the **date** and the **assignment name** (e.g. Project 2).
   Reminder: only one submission needs to be made for a pair/team.
4. The header of each file must **start at the first line** (no empty lines or *import* statements before it) and should be followed by at least one empty line before the first line of the code.