King Fahd University of Petroleum and Minerals
Information and Computer Science Department

ICS 353: Design and Analysis of Algorithms

Programming Assignment
(Due Saturday December 8, 2018 at midnight)

The purpose of this assignment is to practically investigate the efficiency of Strassen's algorithm to multiply two square "long integer" random matrices of size $2^n \times 2^n$. The input values are random integers in the range [-100,100]. In order to do that, you are asked to carry out the following tasks:

1- (15 points) provide the following implementations of matrix multiplication using Java:
   a. The classical iterative version of the matrix multiplication algorithm.
   b. The classical divide and conquer recursive version of the matrix multiplication algorithm.
   c. The classical Strassen's divide and conquer recursive algorithm for matrix multiplication (with base case value of $n = 1$.)
   d. Strassen's divide and conquer recursive algorithm for matrix multiplication with base case value of $n > 1$. In the base case, you just use the classical iterative version of matrix multiplication.

   Note that your implementations should produce the output matrix (to make sure that your implementation is correct) and the time it took to produce the results. Note that input and output should be given in the form of text files and the parameter *n* and the filenames should be provided as arguments to the program. This means that you are not supposed to hardcode them. For example, a command like: `matrixMultiply 3 input.txt output.txt` would take as input two $2^3 \times 2^3$ matrices stored in File `input.txt` and will produce the execution time followed by the resulting product matrix in File `output.txt`.

   The input text file should contain the two input matrices, row by row. An example input file is shown at the end of this assignment.

2- (15 points) Run the four algorithms on different sizes of matrices (from small to very large) and for various base cases for the Strassen's algorithm. Make sure that you carry out all your experimentations using the same "environment" (machine, compiler, etc.)
3- (60 points) Write a report (word document or pdf-generated file from an editor) that contains the following information
   a. (5 points) How to compile and run the code, with any implementation details worth of mentioning.
   b. (15 points) Documentation of all experiments carried out in the form of a comparative table.
   c. (25 points) Analysis of the results present in the table. If there are any unexpected results, please highlight them and give possible justification for them.
   d. (10 points) Analysis of the results for different base values for Strassen's algorithm.

e. (5 points) Your conclusions of when to use each algorithm and the best Strassen's base case.
f. Who did what and the approximate percentage of total work done in this assignment for each member?

4- (10 points) Present your findings in a 7-minute presentation. Your presentation should contain the following:
   a. How many experiments did you run and a description of your testing environment?
   b. Comparative table of the results of all your experimentations. Highlight any "interesting" results, and summarize your analysis.
   c. Which base case values did you use for Strassen's algorithm?
   d. Summary of your conclusions.

   Note that the presentations will be done in 3 separate sessions during the last week of classes. Each session should be attended by groups included in that session, only.

5- Your code should be under one directory, your input files used in the experimentation (other than the ones provided) should be under Directory "input", and your report and presentation documents should be under Directory "Documents". Zip the three directories as a single file and submit to blackboard.

6- Note that you should do your implementations from scratch and without using any internet resources. Your answers will be verified by safeAssign and any copying of code among the groups or from the internet will result in a zero grade.

Sample input file containing two input matrices of size $(2^3 \times 2^3)$ entered row by row is shown below.

73	-32	-4	11	37	95	87	-42
32	61	-2	-43	-11	19	-1	-87
-61	-60	-44	-26	69	-9	-65	-63
-51	81	-99	76	21	31	57	93
74	-42	-43	-59	-27	39	93	-33
12	-59	84	-9	-60	-76	-53	35
88	3	50	73	-49	1	-50	-31
71	-39	58	-65	-77	24	51	78
58	46	52	96	-27	-23	-65	-65
-63	-53	100	61	20	-8	-61	33
7	76	-40	-72	36	-26	-85	-63
6	87	82	-99	27	16	90	-8
70	100	-23	-9	-80	61	-61	25
-30	-90	-84	10	-12	-30	-46	-31
60	71	-97	-46	-38	70	-23	-12
-3	19	82	24	-75	-61	-95	79