EE 599 Lab2
Spring 2019 Nazarian 100 Points

Student ID: _____ Name: _____

Assigned: Wednesday Jan. 16

Due: Wednesday Jan. 23 at 11:59pm

Late submissions will be accepted for only a few hours after the deadline. Submissions between 12am and 1am (Jan. 24th) : 3% penalty, between 1-2am: 9%, 2-3am: 18%, and 3-4am: 30%. No submissions accepted after 4am.

Notes:

This assignment based on individual work. No collaboration is allowed (Discussing the technical details of each problem with other students before submitting your work, copying from students of current or previous semesters is not permitted in our lab and HW assignments). You may post your questions on the discussion forums and/or use the office hours. We may pick some students in random to demonstrate their design and simulations. Please refer to the syllabus and the first lecture regarding the USC and our policies, including the penalties for any violation. If you have any doubts about what is allowed or prohibited in this course, please contact the instructor.

What You Will Practice

In this lab you will going to practice a few programming skills you learn in this course including array, circulation, recursive function, divide and conquer algorithm **in C**.

Problem Description

Given two matrices A and B of size $n \times n$ each, find the matrix product with two different methods, naive method and Strassen's method. And analyze time complexity of two methods.

Input Files

You will be given a few text files named "input.txt" one at a time, i.e. your code will be run a few times and each time a new "input.txt" file will be given. This file contains three pieces of information. (i) size of a 2D matrix/array (ii) content of the first matrix separated by " ", a space character. (iii) content of the second array separated by a space character. Always, the first line in the given text file will be the size of the matrix. Here is an example of "input.txt". The first line is the size of a 3x3 matrix. The next 3 lines are content of the first 3x3 matrix and the remaining lines are content of the second 3x3 matrix.

In “input.txt” file:

3
2 5 6
3 4 7
0 2 5
1 4 6
3 2 1
9 7 8

Part 1: Calculate Matrix Multiplication (80%)

1. Naive Method (30%)

In this part, you should write a function to calculate matrix multiplication by definition. If **A** is an $n \times m$ matrix and **B** is an $m \times p$ matrix, here assuming $n = m = p$,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mp} \end{pmatrix}$$

the matrix product **C=AB** is defined to be the $n \times p$ matrix

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{np} \end{pmatrix}$$

such that

$$c_{ij} = a_{i1}b_{1j} + \cdots + a_{im}b_{mj} = \sum_{k=1}^{m} a_{ik}b_{kj},$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, p$

2. Strassen’s Method (50%)

In this part, you should write functions to implement Strassen’s method for matrix multiplication and you **MUST** use recursive function to implement it. Since simple Divide and

Conquer algorithm to multiply two square matrices has same time complexity with the naive method, Strassen's method reduces the number of recursive calls in the simple Divide and Conquer method and thus makes it faster than the naive method.

Similar to simple divide and conquer method, the square matrix is divided into 4 submatrices of size n/2 x n/2. If the matrices **A**, **B** are not of type $2^n \times 2^n$, fill the missing rows and columns with zeros. Then the multiplication results are calculated recursively as follows.

1) partition **A**, **B**, **C** into equally sized block matrices, **C=AB**

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix}$$

2) calculate the defined 7 new matrices M_i , $i = 1...7$

$$\mathbf{M}_1 := (\mathbf{A}_{1,1} + \mathbf{A}_{2,2})(\mathbf{B}_{1,1} + \mathbf{B}_{2,2})$$
$$\mathbf{M}_2 := (\mathbf{A}_{2,1} + \mathbf{A}_{2,2})\mathbf{B}_{1,1}$$
$$\mathbf{M}_3 := \mathbf{A}_{1,1}(\mathbf{B}_{1,2} - \mathbf{B}_{2,2})$$
$$\mathbf{M}_4 := \mathbf{A}_{2,2}(\mathbf{B}_{2,1} - \mathbf{B}_{1,1})$$
$$\mathbf{M}_5 := (\mathbf{A}_{1,1} + \mathbf{A}_{1,2})\mathbf{B}_{2,2}$$
$$\mathbf{M}_6 := (\mathbf{A}_{2,1} - \mathbf{A}_{1,1})(\mathbf{B}_{1,1} + \mathbf{B}_{1,2})$$
$$\mathbf{M}_7 := (\mathbf{A}_{1,2} - \mathbf{A}_{2,2})(\mathbf{B}_{2,1} + \mathbf{B}_{2,2})$$

3) calculate the target matrix **C**

$$\mathbf{C}_{1,1} = \mathbf{M}_1 + \mathbf{M}_4 - \mathbf{M}_5 + \mathbf{M}_7$$
$$\mathbf{C}_{1,2} = \mathbf{M}_3 + \mathbf{M}_5$$
$$\mathbf{C}_{2,1} = \mathbf{M}_2 + \mathbf{M}_4$$
$$\mathbf{C}_{2,2} = \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3 + \mathbf{M}_6$$

3. Output Files

After obtaining matrix product result, you should write your results to **TWO** text files with name "**output_p1_m1.txt**" for naive method and "**output_p1_m2.txt**" for Strassen's method, where numbers should be separated by a space character. You should write your result to files instead of print in terminal. For example, the content of matrix product result for above example will be as follows:

In "output_p1_m1.txt" file:

```
71 60 65
78 69 78
51 39 42
```

In "output_p1_m2.txt" file:

```
71 60 65
78 69 78
51 39 42
```

4. Notes

- You can assume that all numbers are integers (positive/negative).
- The given matrix always be in square shape, i.e. its size will be 1 by 1, 2 by 2 or 3 by 3, etc., but not 3 by 5.
- The size of matrices will be at most 512.
- Consider the corner cases, such as 1 by 1 matrix.
- you **MUST** use recursive functions to implement Strassen's method.
- In Strassen's method part, if two input matrices are not of type $2^n \times 2^n$, fill the missing rows and columns with zeros.
- For part 1, your code should be written in only **ONE** file named "**EE599_Lab2_<STUDENT-ID>.c**" which should include at least two functions to implement two methods and one main function. Please replace <STUDENT-ID> with your own 10 digit USC-ID, the name of the file would be something like: EE599_Lab2_1234567890.c
- Some "input.txt" and correct "output_p1_golden.txt" files are uploaded in the website of the course for your reference. You can use those for testing your code, but there will be other inputs for grading your code.
- You may write some simple functions for printing a row in an array or the whole array, and call it in each line of your code that you want to see the content of the array. This will help you in debugging your code. A simple example is as follows:

```
void print_1D(int M[], int size)
{
        for (int i=0;i<size;i++)
                printf("%d ", M[i]);
        printf("\n");
}
```

Part 2: Time Complexity Analysis (20%)

In this part, you should compare the running time of your programs for two methods.

1. Time Complexity of Two Methods:

Time complexity of the Naive method is $O(n^3)$ since there will be 3 FOR loops in your codes.

For the simple Divide and Conquer method, two matrices are divided into 4 submatrices of the half size $n/2 \times n/2$ and by definition of block matrix multiplication, there will be 8

multiplications for matrices of size $n/2$ and 4 additions. Thus, the time complexity of the simple Divide and Conquer method can be written as $T(n) = 8T(n/2) + O(n^2)$, where $T(n/2)$ is the total time of calculating multiplications for matrices of size $n/2 \times n/2$ including the time of recursive calls for its submatrices and $O(n^2)$ is the time of two matrices addition. From Master's Theorem, time complexity of simple Divide and Conquer method is $O(n^3)$ which is same as naive method.

Strassen's method reduces the number of multiplications from 8 to 7 and thus reduces the time of recursive calls. Then, its time complexity can be written as $T(n) = 7T(n/2) + O(n^2)$, i.e. $O(n^{\log_2^7}) = O(n^{2.8074})$.

Although Strassen's method is theoretically faster than the naive one comparing their time complexity (big O complexity), but we also need to consider other things such as memory when coding. You may found the naive method is always faster when you calculate running time. This is because extra memory-allocating in each recursive call and creating extra activation records for recursion are time consuming, which beats the gain from having fewer arithmetic operations.

2. Calculate running time of your code

There is a C library called <time.h> for manipulating dates and time information. Type "time_t" defines the current absolute time and the function "clock()" gives the value of current time. Then, the running time of a process can be given by the end absolute time minus the start absolute time. Please time as an integer. An example is as follows:

```
#include<time.h>
time_t start, end;
start = clock();
[running function]
end = clock();
running_time = end - start;
```

3. Output Files

After obtaining the time, you should write two integer running time results into **ONE** text files with name "output_p2.txt", separating two numbers by a space character. For example, the output file will be as follows:

In "output_p2.txt" file:

```
300 280
```

4. Notes

- The running time you analyze and output in the file should only consist of the time of running method function, but **NOT** include the time of reading files or writing files.
- In general, the unit of time is millisecond when using clock() function. Please make sure your output time is an integer, i.e. not 300.100100 but 300.
- For part 2, your code should be written in the same file of part 1, which is "**EE599_Lab2_<STUDENT-ID>.c**".
- There is a sample time result given to you. You may get different results with the same input since the running time varies on different computers and systems. However, this part will be graded identically on one computer.
- You are encouraged to simplify your code, memory usage, etc. as much as possible.

Summary

- You are **NOT** allowed to use any implemented libraries of matrix multiplication. You can only use standard libraries <stdio>, <stdlib>, <malloc>, <time>. For other unsure libraries, please ask in discussion forum first before you use it.
- For both parts, you have to follow the exact formats and syntax for generating the required results. Even adding an extra white line in the beginning of your output file or an extra space, etc., will result in losing the whole score of this lab.
- There should be only one code file "**EE599_Lab2_<STUDENT-ID>.c**", where includes at least two functions, main, etc., and running it will generate three output files.
- You can use the given shell script "script.sh" to check whether your matrix product result is same as the golden answer, which is similar to what you do in Lab1.
- You should also write a README file, named "readme.txt". A README file typically contains information of author's name and email, a brief program summary, references and instructions. Besides, include any information that you think the course staff, especially the grader should know while grading your assignment: references, any non-working part, any concerns, etc.
    1) Any non-working part should be clearly stated.
    2) The citations should be done carefully and clearly, e.g.: "to write my code, lines 27 to 65, I used the Diijkstra's shortest path algorithm c++ code from the following website: www.SampleWebsite.com/ ..."
    3) The Readme file content of labs cannot be hand-written and should only be typed.

Submission

Submit your code file "EE599_Lab2_<STUDENT-ID>.c" and "readme.txt" to Blackboard-Assignments. Please replace <STUDENT-ID> with your own 10 digit USC-ID, the name of the file would be something like: EE599_Lab2_1234567890.c