EE 355 Lab2
Spring 2019 Nazarian 100 Points

Student ID: _____ Name: _____

Assigned: Monday Jan. 14

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 have learned in this course including list, circulation, etc. **in Python**.

Problem Description

Given two matrices A and B of size 8×8 each, find the matrix product and show the running time of your program. You can choose one algorithm to implement matrix multiplication, but you are not allowed to use any existed libraries of matrix multiplication, such as numpy, etc.

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 two pieces of information. (i) content of the first matrix separated by " ", a space character. (ii) content of the second array separated by a space character. Always, the size of the matrix is fixed as 8×8. Here is an example of "input.txt". For convenience, the example shows in case of 3x3 matrices. The first 3 lines are content of the first 3x3 matrix and the remaining 3 lines are content of the

second 3x3 matrix. Similarly, in case of 8×8 matrix, the first 8 lines are content of the first 8x8 matrix and the remaining 8 lines are content of the second 8x8 matrix.

In "input.txt" file:

---

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

Part 1: Calculate Matrix Multiplication

You can choose any **ONE** algorithm to implement matrix multiplication. Here, two methods are introduced for reference. The naive method is an easy one to implement with three FOR loops. The Strassen's method has lower time complexity than the naive one but needs more advanced programming skills, such as recursive function, divide and conquer algorithm, etc. Note that you are not required to implement Strassen's method, but if you want more practice, you can try Strassen's method.

1. Naive Method

You can 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 (Advanced)

The second choice is to implement Strassen's method for matrix multiplication. Recursive function is recommended to implement Strassen's 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, a 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 as following steps.

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, $i = 1 \ldots 7$, recursively

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

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 content of the result to a text files with name "**output.txt**", separating by a space character. Do **NOT** print in terminal but write the result to file. For example, the content of matrix product result for above example will be as follows:

In “output.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 fixed as 8 by 8.
- Do **NOT** use any library/package that can directly calculate matrix multiplication, such as numpy, etc.
- Some “input.txt” and correct “output_golden.txt” files are uploaded to Blackboard for your reference. You can use those to test your code, but there will be other inputs for grading your code.
- Note that you are **NOT** required to implement Strassen’s method, but if you want more practice, you can try Strassen’s method.

Part 2: Time Complexity Analysis

In this part, you will add codes to your code part 1 in order to calculate running time of your program. You do not need to analyze time complexity of your algorithm. But since time complexity is a useful tool for algorithm analysis, you can refer to the following section 1 for a brief introduction.

1. Time Complexity of Two Methods (Reference)

Time complexity of the Naive method is $O\left(n^3\right)$ 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 $O\left(n^3\right)$ 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. Thus, its time complexity can be written as $T(n) = 7T(n/2) + O(n^2)$, i.e. $O(n^{log_2^7}) = O(n^{2.8074})$.

2. Calculate running time of your code

There is a python package called "time" for time access and conversions. The function "time.time()" gives the value of current time in seconds. Then, the running time of a process can be calculated by the end time minus the start time. An example which prints the running time of printing "hello" is as follows:

```
import time
start = time.time()
print("hello")
end = time.time()
print(end - start)
```

3. Output Files

After obtaining the time, you should write the result to a text files with name "**output_time.txt**". For example, the output file may be as follows:

In "output_time.txt" file:

```
1.9128199821
```

4. Notes

- The running time you analyze should only consist of the time for running matrix multiplication part, but **NOT** include the time of importing packages, reading files or writing files.
- The unit of time is generally in second.
- There is no testing cases given to you in this part since the running time varies on different computers and systems.

Summary

- For both parts you have to follow the exact formats and syntaxes 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 “EE355_Lab2_<STUDENT-ID>.py” and running it will generate two output files, “output.txt” and “output_time.txt”. Do **NOT** submit your own output files.
- You can use the given shell script “script.sh” to check whether your matrix product result is the same with 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 python 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 “EE355_Lab2_<STUDENT-ID>.py” and “readme.txt” to Assignments in Blackboard. Please replace <STUDENT-ID> with your own 10 digit USC-ID, the name of the file would be something like: EE355_Lab2_1234567890.py