tim allec

Name, SID, Date ....

In Class Assignment 19: Strassen's Square Matrix Multiply Recursive

Benjamin Sanders, MS November 5, 2021

1 Introduction

You may work in groups of up to two or three students. Write all student names at the top of all pages for this assignment. Turn in all work to Blackboard on or before the deadline to receive credit.

You may use additional libraries and online resources, if you get them approved in writing, over email, from the instructor first. If you have received approval from the instructor, write the approved libraries and any references in the space below.

....
....
....
....

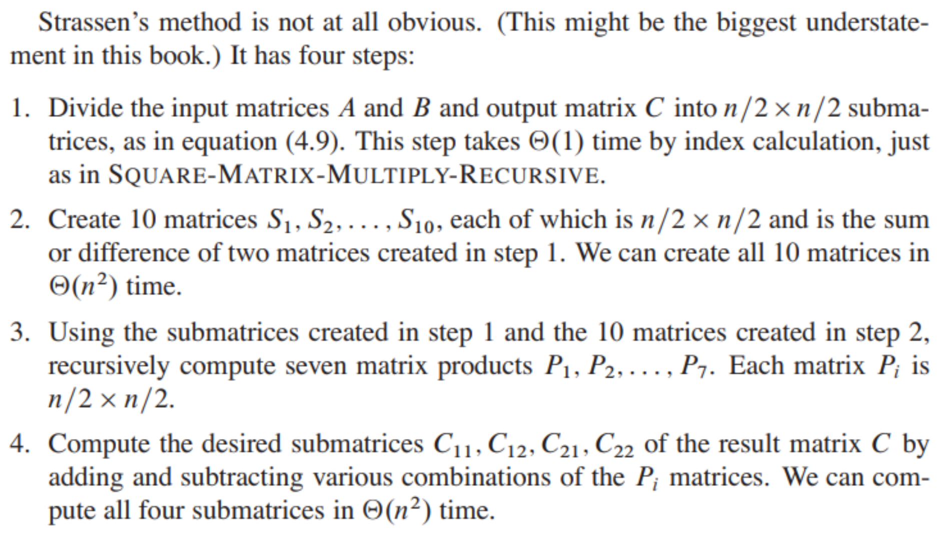
2 Assignment Description

2.1 Big Picture

Strassen's Square Matrix Multiply Recursive is a function in Linear Algebra, and is used in extensively large image processing systems, such as rendering the graphics for a feature film.

2.2 Algorithm Implementation

Implement the following algorithm in Java, using the Vector data structure for any 1-D array, 2-D array (a.k.a. matrix), or linear algebra purposes.



Strassen's method is not at all obvious. (This might be the biggest understatement in this book.) It has four steps:

1. Divide the input matrices A and B and output matrix C into $n/2 \times n/2$ submatrices, as in equation (4.9). This step takes $\Theta(1)$ time by index calculation, just as in SQUARE-MATRIX-MULTIPLY-RECURSIVE.
2. Create 10 matrices $S_1, S_2, \ldots, S_{10}$, each of which is $n/2 \times n/2$ and is the sum or difference of two matrices created in step 1. We can create all 10 matrices in $\Theta(n^2)$ time.
3. Using the submatrices created in step 1 and the 10 matrices created in step 2, recursively compute seven matrix products $P_1, P_2, \ldots, P_7$. Each matrix P_i is $n/2 \times n/2$.
4. Compute the desired submatrices $C_{11}, C_{12}, C_{21}, C_{22}$ of the result matrix C by adding and subtracting various combinations of the P_i matrices. We can compute all four submatrices in $\Theta(n^2)$ time.

Note that Step 1 is described in-detail in the previous ICA.

We now proceed to describe the details. In step 2, we create the following 10 matrices:

$$
\begin{aligned}
S_1 &= B_{12} - B_{22} ,\\
S_2 &= A_{11} + A_{12} ,\\
S_3 &= A_{21} + A_{22} ,\\
S_4 &= B_{21} - B_{11} ,\\
S_5 &= A_{11} + A_{22} ,\\
S_6 &= B_{11} + B_{22} ,\\
S_7 &= A_{12} - A_{22} ,\\
S_8 &= B_{21} + B_{22} ,\\
S_9 &= A_{11} - A_{21} ,\\
S_{10} &= B_{11} + B_{12} .
\end{aligned}
$$

Since we must add or subtract $n/2 \times n/2$ matrices 10 times, this step does indeed take $\Theta(n^2)$ time.

In step 3, we recursively multiply $n/2 \times n/2$ matrices seven times to compute the following $n/2 \times n/2$ matrices, each of which is the sum or difference of products of A and B submatrices:

$$
\begin{aligned}
P_1 &= A_{11} \cdot S_1 &&= A_{11} \cdot B_{12} - A_{11} \cdot B_{22} ,\\
P_2 &= S_2 \cdot B_{22} &&= A_{11} \cdot B_{22} + A_{12} \cdot B_{22} ,\\
P_3 &= S_3 \cdot B_{11} &&= A_{21} \cdot B_{11} + A_{22} \cdot B_{11} ,\\
P_4 &= A_{22} \cdot S_4 &&= A_{22} \cdot B_{21} - A_{22} \cdot B_{11} ,\\
P_5 &= S_5 \cdot S_6 &&= A_{11} \cdot B_{11} + A_{11} \cdot B_{22} + A_{22} \cdot B_{11} + A_{22} \cdot B_{22} ,\\
P_6 &= S_7 \cdot S_8 &&= A_{12} \cdot B_{21} + A_{12} \cdot B_{22} - A_{22} \cdot B_{21} - A_{22} \cdot B_{22} ,\\
P_7 &= S_9 \cdot S_{10} &&= A_{11} \cdot B_{11} + A_{11} \cdot B_{12} - A_{21} \cdot B_{11} - A_{21} \cdot B_{12} .
\end{aligned}
$$

Note that the only multiplications we need to perform are those in the middle column of the above equations. The right-hand column just shows what these products equal in terms of the original submatrices created in step 1.

Step 4 adds and subtracts the P_i matrices created in step 3 to construct the four $n/2 \times n/2$ submatrices of the product C. We start with

$$C_{11} = P_5 + P_4 - P_2 + P_6 .$$

Similarly, we set

$$C_{12} = P_1 + P_2 ,$$

Similarly, we set

$$C_{21} = P_3 + P_4 ,$$

Finally, we set

$$C_{22} = P_5 + P_1 - P_3 - P_7 ,$$

Name, SID, Date ....

2.3 Time Complexity Analysis

Where n is the number of data points in A and B together, analyze the time complexity of the given algorithm with respect to n. Write the result of your analysis in big-O notation, i.e. $O(n^2)$ in the space below.

O(n^2.8)

2.4 Space Complexity Analysis

Where n is the number of data points in A and B together, analyze the space complexity of the given algorithm with respect to n. Write the result of your analysis in big-O notation, i.e. $O(n \cdot log(n))$ in the space below.

O(n^2)

2.5 Optimize the Algorithm for a Purpose

Choose an optimization, either in time or in space, for the given algorithm. Write your intended big-O notation, i.e. $O(1)$, $O(n)$, etc., in the space below, and write N/A in the other space.

- Time Complexity: O(n^2.3)
- Space Complexity: O(n^2)

What application would benefit from the purpose of the above optimization? Why? Write two sentences to answer these questions in the space below.

Le Gall's improvement can significantly reduce memory consumption and improve cache efficiency.

2.6 New Algorithm Design ~~and Implementation~~

In the space below, design an algorithm that achieves the same purpose of the given algorithm, but includes the optimization you have specified above. Use pseudocode written in a style similar to the given algorithm, ~~and implement it in Java~~. You may use as many additional pages as necessary for this purpose.

3 What to Turn In

Turn in one PDF or Word document on Blackboard, containing the following items.

1. All pages scanned or photographed of the In Class Assignment completed document.
2. Any additional pages you used to complete the assignment.
3. All code created for the assignment, along with test cases.
4. One statement indicating which parts of your implementation(s) are working, and which parts are not.
5. Screenshots demonstrating the code working, if it is working.