Ziqi Tan Homework 1

CS 530 Advanced Algorithm

U88387934 ziqi1756@bu.edu

1. LU Matrices

   (i) What is the determinant of a lower triangular matrix? Briefly justify your answer.

   Answer: it is the multiplication of all the diagonal elements of the lower triangular matrix. Briefly prove: we use the following formular to get the determinant of a matrix.

$$det(A) = \begin{cases} a_{11} & n = 1 \\ \sum_{j=1}^{n} a_{ij}(-1)^{i+j} A_{ij} & n > 1 \end{cases} \tag{1}$$

   where A_{ij} is the cofactor, the determinant of a matrix which is cut down from matrix A by removing its ith row and jth column.
   When $i = 1$, formular (1) will become:

$$det(A) = \begin{cases} a_{11} & n = 1 \\ \sum_{j=1}^{n} a_{1j}(-1)^{1+j} A_{1j} & n > 1 \end{cases} \tag{2}$$

   When we figure out the determinant A_{1j}, we will find that the first row of A_{1j} has only one non-zero element at the first column. Therefore,

$$det(A) = a_{11}a_{22}a_{33}...a_{nn} \tag{3}$$

   (ii) HelloWorld!

   (iii) Is it possible for a singular lower triangular matrix to have a inverse? Here either give an example showing this statement is true or prove that no singular lower triangular matrix has an inverse.

   Answer: a singular matrix does not have a inverse. In a singular matrix, the determinant is zero. For instance, A is an 2*2 matrix.

$$A = \begin{bmatrix} 1 & 0 \\ 3 & 0 \end{bmatrix} \tag{4}$$

$$A^{-1} = \frac{1}{det(A)} \begin{bmatrix} 0 & -3 \\ 0 & 1 \end{bmatrix} \tag{5}$$

   In (2), we find there is no inverse of A, because det(A) is zero.

(iv) Assume you are given a matrix M and a decomposition M=LU with L and U as above. Explain how to use this decomposition to compute M's deverminant. The computation shoud be use your knowledge of L and U.

Answer: According to the appendix D of the textbook, for any square matrices A and B, we have $det(AB) = det(A)det(B)$. Therefore, we will have the following formula:

$$det(M) = det(LU) = det(L)det(U) \tag{6}$$

According to formula(6) and my answer in 1.(i), det(M) is the multiplication of the diagnal elements of both L and U.

2. LU decompostion

(i) Prove that for n equal to 2 or 3 there is a non-singular square (n by n) matrix which has no LU docomposition with L unit lower triangular and U upper triangular. (In fact, this is true for any integer >= 2.)

Answer: If one cofactor of a matrix is zero, this matirx has no LU decomposition with L unit lower triangular and U upper triangular.

For instance, a matrix M with zeros in every position on the diagonal. The textbook use this fomulas (28.8 in Chapter 28 of textbook) to recursively calculate the LU decomposition:

$$\begin{aligned} A = \left[\begin{array}{c|ccc} a_{11} & a_{12} & \cdots & a_{1n} \\ \hline a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & a_{2n} \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{array}\right] &= \begin{bmatrix} a_{11} & \boldsymbol{\omega^T} \\ \boldsymbol{v} & A' \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ \boldsymbol{v}/a_{11} & I_{n-1} \end{bmatrix} \begin{bmatrix} a_{11} & \boldsymbol{\omega^T} \\ 0 & A' - \boldsymbol{v\omega^T}/a_{11} \end{bmatrix} \end{aligned} \tag{7}$$

Equation (7) shows that if the diagonal elements $a_{11} = 0$ or $A' - \boldsymbol{v\omega^T}/a_{11} = 0$, this method doesn't work, because it divides by 0. Therefore, there is non-singular square matrix having no LU decomposition with L unit lower triangular and U upper triangular.

(ii) Show that the inverse of a permutation matrix P is also a permutation matrix. (You can do this by explicitly difining what P^{-1} when P is given.)

Answer: a permutation matrix is a square binary matrix, each row or column of which only contains one 1, according to appendix D in textbook.

A permutation matrix P and its transpose can be written as follows:

$$P = \begin{bmatrix} r_1 \\ r_2 \\ \dots \\ r_i \\ \dots \\ r_n \end{bmatrix}, P^T = \begin{bmatrix} c_1 \\ c_2 \\ \dots \\ c_j \\ \dots \\ c_n \end{bmatrix} \tag{8}$$

where $r_i (i = 1, 2, ..., n)$ is a unit row vector and $c_j (j = 1, 2, ..., n)$ is a unit column vector. Now we can easily get the following fomula.

$$r_i c_j = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases} \tag{9}$$

This formula means:

$$PP^T = I \tag{10}$$

which means $P^{-1} = P^T$.

According the definition of permutaion matrix, the transpose of a permutaion matrix is also a permutation matrix. Therefore, P^{-1} is also a permutation matrix.

(iii) Give an example of a singular 2 by 2 matrix A and of a singular 3 by 3 matrix B which have a LU decomposition. Both A and B should be non-zero matrices and you should write the L and the U for both of them.

Answer: Here is an example of A and of B. They are singular matrices. $det(A) = 0$ and $det(B) = 0$.

$$A = \begin{bmatrix} 4 & 4 \\ 2 & 2 \end{bmatrix} = LU = \begin{bmatrix} 1 & 0 \\ 0.5 & 1 \end{bmatrix} \begin{bmatrix} 4 & 4 \\ 0 & 0 \end{bmatrix} \tag{11}$$

$$B = \begin{bmatrix} 4 & 4 & 4 \\ 2 & 2 & 4 \\ 3 & 3 & 4 \end{bmatrix} = LU = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0.75 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 4 & 4 \\ 0 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix} \tag{12}$$

3. Strassen's method
Assume you are handed Strassen's algorithm which multiplies two 2 by 2 matrices and uses 7 integer multiplications and 18 integer additions to do this.

(i) Show how to use this algorithm iteratively (using divide and conquer) to multiply two 4 by 4 matrices. Calculate exactly how many integer multiplications and integer additions are done? You should show your work and should compare the answer you get to the number of multiplications and adds which would be done if you turn the 4 by 4 algorithm into block multiplication of 2 by 2 matrices.

Answer: 49 multiplications and 198 additions.

Analysis: Refer to Strassen's medhod in chapter 4 of the textbook.
Let A and B be the input 4 by 4 matrices and C be the 4 by 4 output matrix.

Step 1: Divide the matrices A, B and C into 2 by 2 matrices. We don't make any copy the matrices but use the index calculation.

Step 2: Create 10 matrices $S_1, S_2, ..., S_{10}$, each of which is 2 by 2 and is the sum or difference of two matrices created in step 1.
In this step, we will have $10 \times (2 \times 2) = 40$ additions/subtractions operations.

Step 3: Using the submatrices created in step 1 and the 10 matrices created in step 2, recursively compute seven matrix products $P_1, P_2, ..., P_7$. Each matrix P_i is 2 by 2.
This step is in fact doing 7 times 2 by 2 matrcices multiplications by Strassen's method with 7 multiplications and 18 additions operations. Thus, $7 \times 7 = 49$ multiplications and $7 \times 18 = 126$ additions in all.

Step 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.
In this step, there are 8 times of adding/subtracting two 2 by 2 matrices. Thus, $8 \times 4 = 32$ additions in all.

To sum up, multiplying two 4 by 4 matrices by Strassens method will take 49 multiplications and $40 + 126 + 32 = 198$ additions in all.

(ii) In Section 4.2 of the textbook it is shown that Strassen's method uses only $O(n^{log_7})$ multiplications to multiply 2 n by n when n is an even power of 2. (You need not prove this.)
Using this result describe an algorithm how you can get the same $O(n^{log_7})$ matrix multiplication result even if you start with n by n matrices with n not an even power of 2.

I figure out two possible solutions.

Solution 1: When doing $n/2$ divide and conquer, we need to make sure $n/2$ is an even number for the next divide and conquer. Therefore, when we find $n/2$ is an odd number, we can do that by expanding the matrix with one more row of zeros and one more column of zeros .

However, when we have two 50 by 50 input matrices, this solution needs three times of expansion.
$50/2 = 25$, 25 by 25 matrix to 26 by 26 matrix.
$26/2 = 13$, 13 by 13 matrix to 14 by 14 matrix.
$14/2 = 7$, 7 by 7 matrix to 8 by 8 matrix.
8 is a power of 2.

Solution2: Before the strassen's algorithm, we could expand the input matrices with enough row and column of zeros until the size of the matrices is a power of 2.
In this solution, we could expand it for one time and make sure in the following recursions $n/2$ is an even number, which will save a lot of runing time compared to the previous solution. On the other hand, when n is big, it has to expand with many zeros to reach a power of 2, which indeed takes a lot of memory space.
For instance, for a 130 by 130 matrix, it should be expanded to a 256 by 256 matrix. ($2^7 = 128$, $2^8 = 256$)

Why we can still do it in $O(n^{log_7})$ time?
For solution 2, in the worst case the number of multiplications will almost double and the time complexity will be $O((2n)^{log_7})$. It doesn't change the exponential log_7. Thus, we can still do it in $O(n^{log_7})$ time.