CSE331 Introduction to Algorithm
Lecture 9: Matrix Multiplication

Antoine Vigneron
antoine@unist.ac.kr

Ulsan National Institute of Science and Technology

July 23, 2021

Introduction

- Today's topic: computing the product of two $n \times n$ matrices.
- Naive algorithm: time $\Theta(n^3)$.
- This lecture: *Strassen's algorithm*, which runs in $\Theta(n^{\log 7}) \simeq n^{2.81}$ time.
- It is a divide and conquer algorithm.
- It splits the problem recursively into 7 subproblems.
- The idea is similar with Karatsuba's algorithm.
- Best known algorithm: $\approx n^{2.37}$. Not covered in CSE331.
- Matrix multiplication algorithms are very important in numerical analysis, and also have applications to combinatorial algorithms.
- Reference: Section 4.2 of the textbook Introduction to Algorithms by Cormen, Leiserson, Rivest and Stein.

Matrix Multiplication

- In this lecture, we only consider $n \times n$ square matrices where n is a power of 2.

$$A = (a_{ij}) = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2j} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ a_{i1} & a_{i2} & \dots & a_{ij} & \dots & a_{in} \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nj} & \dots & a_{nn} \end{pmatrix}$$

Matrix Multiplication

- Let $A = (a_{ij}), B = (b_{ij})$ be $n \times n$ matrices.
- Then the product $C = A \cdot B$ is the $n \times n$ matrix $C = (c_{ij})$ such that

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}.$$

$$\begin{pmatrix} \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & c_{ij} & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \end{pmatrix} = \begin{pmatrix} \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ a_{i1} & a_{i2} & \dots & a_{in} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \end{pmatrix} \cdot \begin{pmatrix} \dots & \dots & b_{1j} & \dots \\ \dots & \dots & b_{2j} & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & b_{nj} & \dots \end{pmatrix}$$

- So c_{ij} is the dot product of the ith row of A with the jth column of B.

Naive Algorithm

Pseudocode

1: **procedure** NaiveSquareMatrixMultiplication(A, B)
2: $\quad n \leftarrow$ size of A
3: $\quad C \leftarrow$ new $n \times n$ matrix
4: $\quad$**for** $i \leftarrow 1, n$ **do**
5: $\quad\quad$**for** $j \leftarrow 1, n$ **do**
6: $\quad\quad\quad c_{ij} \leftarrow 0$
7: $\quad\quad\quad$**for** $k \leftarrow 1, n$ **do**
8: $\quad\quad\quad\quad c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$
9: $\quad$**return** C

- Running time?
- Dominated by lines 7 and 8, which are executed n^3 times.
- So it is $\Theta(n^3)$.
  - Remark: Here n is not the input size. The input size is $2n^2$.

Block Matrices

- Example:

$$A = \begin{pmatrix} 2 & 5 & 3 & 1 \\ 4 & 3 & 2 & 2 \\ 3 & 1 & 5 & 6 \\ 1 & 3 & 2 & 4 \end{pmatrix} = \left(\begin{array}{cc|cc} 2 & 5 & 3 & 1 \\ 4 & 3 & 2 & 2 \\ \hline 3 & 1 & 5 & 6 \\ 1 & 3 & 2 & 4 \end{array}\right) = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

where

$$A_{11} = \begin{pmatrix} 2 & 5 \\ 4 & 3 \end{pmatrix} \quad A_{12} = \begin{pmatrix} 3 & 1 \\ 2 & 2 \end{pmatrix}$$

$$A_{21} = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix} \quad A_{22} = \begin{pmatrix} 5 & 6 \\ 2 & 4 \end{pmatrix}$$

Block Matrices

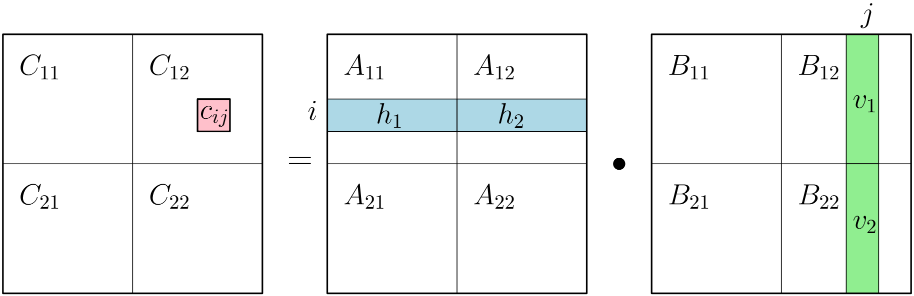
Definition

A *block matrix* is a matrix which is interpreted as having been broken into sections called *blocks* or *submatrices*.

- In this lecture, we will only consider blocks of size $(n/2) \times (n/2)$ from an $n \times n$ matrix:

$$M = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}$$

Block Matrix Multiplication



$$c_{ij} = h_1 \cdot v_1 + h_2 \cdot v_2$$
$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$

Block Matrix Multiplication

- Suppose $C = A \cdot B$ are $n \times n$ matrices.

$$C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$
$$= \begin{pmatrix} A_{11} \cdot B_{11} + A_{12} \cdot B_{21} & A_{11} \cdot B_{12} + A_{12} \cdot B_{22} \\ A_{21} \cdot B_{11} + A_{22} \cdot B_{21} & A_{21} \cdot B_{12} + A_{22} \cdot B_{22} \end{pmatrix}$$

- So the product of two block matrices can be done block by block, in the same way as we multiply two 2×2 matrices.
- I will not prove it formally, but the previous slide gives the intuition.

First Divide-and-Conquer Algorithm

Pseudocode

procedure $\textsc{Multiply}(A, B)$
    $n \leftarrow$ size of A
    $C \leftarrow$ new $n \times n$ matrix
    if $n = 1$ **then**
        $c_{11} \leftarrow a_{11} \cdot b_{11}$
        return C
    partition A, B, C into four $(n/2) \times (n/2)$ blocks each
    $C_{11} \leftarrow \mathrm{Multiply}(A_{11}, B_{11}) + \mathrm{Multiply}(A_{12}, B_{21})$
    $C_{12} \leftarrow \mathrm{Multiply}(A_{11}, B_{12}) + \mathrm{Multiply}(A_{12}, B_{22})$
    $C_{21} \leftarrow \mathrm{Multiply}(A_{21}, B_{11}) + \mathrm{Multiply}(A_{22}, B_{21})$
    $C_{22} \leftarrow \mathrm{Multiply}(A_{21}, B_{12}) + \mathrm{Multiply}(A_{22}, B_{22})$
    return C

First Divide-and-Conquer Algorithm: Analysis

- Ignoring recursive calls, lines 8–11 take $\Theta(n^2)$ time.
  - Consists of 4 matrix addition, which take $\Theta(n^2)$ time each.
- Line 7:
  - We have not explained in details how matrices are split.
  - We can create 12 new $(n/2) \times (n/2)$ arrays and copy the entries of A, B and C into them.
  - It takes time $\Theta(n^2)$.
  - The textbook mentions a better approach, but in any case it will not improve the time bound as lines 8–11 already take $\Theta(n^2)$.
- Line 12:
  - copy $C_{11}, C_{12}, C_{21}, C_{22}$ into C which takes $O(n^2)$ time.
- After counting the 8 recursive calls, we obtain the recurrence relation:

$$T(n) = 8T(n/2) + \Theta(n^2)$$

First Divide-and-Conquer Algorithm: Analysis

- We apply the Master theorem.
- $a = 8$, $b = 2$, and $f(n) = cn^2$ for some constant c.
- $n^{\log_b a} = n^3$.
- We are in Case 1: $f(n) = O(n^{2.99})$.
- So $T(n) = \Theta(n^3)$.
- It is not faster than the naive algorithm.
- Conclusion: divide and conquer does not always help.

Strassen's Algorithm

- In the analysis from previous slide, if $a = 7$ instead of $a = 8$, we get an improved running time $\Theta(n^{\log 7}) \simeq n^{2.81}$.
- $a = 8$ comes from the 8 recursive matrix multiplications.
- Matrix additions do not matter as they take $\Theta(n^2)$ time.
- Idea: Try a similar approach with only 7 recursive multiplications.

Strassen's Algorithm

Strassen's Algorithm

1. Divide input matrices A, B and output matrix C into $(n/2) \times (n/2)$ blocks.
2. Compute 7 $(n/2) \times (n/2)$ matrices $P_1, \ldots, P_7$ from the submatrices A_{ij}, B_{ij}, using only one recursive multiplication for each, as well as one or two additions or subtractions. (See Slide 16, left.)
3. Compute $C_{11}, C_{12}, C_{21}, C_{22}$ using only additions and subtractions. (See Slide 16, right.)

Strassen's Algorithm

Intermediate results

$$P_1 = A_{11} \cdot (B_{12} - B_{22})$$
$$P_2 = (A_{11} + A_{12}) \cdot B_{22}$$
$$P_3 = (A_{21} + A_{22}) \cdot B_{11}$$
$$P_4 = A_{22} \cdot (B_{21} - B_{11})$$
$$P_5 = (A_{11} + A_{22}) \cdot (B_{11} + B_{22})$$
$$P_6 = (A_{12} - A_{22}) \cdot (B_{21} + B_{22})$$
$$P_7 = (A_{11} - A_{21}) \cdot (B_{11} + B_{12})$$

Final result

$$C_{11} = P_5 + P_4 - P_2 + P_6$$
$$C_{12} = P_1 + P_2$$
$$C_{21} = P_3 + P_4$$
$$C_{22} = P_5 + P_1 - P_3 - P_7$$

Strassen's Algorithm: Correctness

- We need to verify that the algorithm is correct.
- So we need to prove that the expressions for C_{ij} on Slide 16 are consistent with Slide 10.
- We need to prove that

$$
\begin{aligned}
P_5 + P_4 - P_2 + P_6 &= A_{11}B_{11} + A_{12}B_{21} \\
P_1 + P_2 &= A_{11}B_{12} + A_{12}B_{22} \\
P_3 + P_4 &= A_{21}B_{11} + A_{22}B_{21} \\
P_5 + P_1 - P_3 - P_7 &= A_{21}B_{12} + A_{22}B_{22}
\end{aligned}
$$

Strassen's Algorithm: Correctness

- We only prove the first equation

$$P_5 + P_4 - P_2 + P_6 = A_{11}B_{11} + A_{12}B_{21}.$$

The other three proofs are analogous.

$$\begin{array}{rlllllll}
P_5 = & A_{11}B_{11} & +A_{11}B_{22} & +A_{22}B_{11} & +A_{22}B_{22} & & & \\
P_4 = & & & -A_{22}B_{11} & & +A_{22}B_{21} & & \\
-\ P_2 = & & -A_{11}B_{22} & & & & -A_{12}B_{22} & \\
P_6 = & & & & -A_{22}B_{22} & -A_{22}B_{21} & +A_{12}B_{22} & +A_{12}B_{21} \\
\hline
C_{11} = & A_{11}B_{11} & & & & & & +A_{12}B_{21}
\end{array}$$

Conclusion

- The running time of Strassen's algorithm is given by the recurrence relation
$$T(n) = 7T(n/2) + \Theta(n^2)$$
- By the master theorem, it yields $T(n) = \Theta(n^{\log 7}) \simeq n^{2.81}$.
- It improves on the naive approach which runs in $\Theta(n^3)$ time.
- Strassen's algorithm, published in 1969, was the first known algorithm to run in $o(n^3)$ time.
- The algorithm and its proof are simple, the difficult part is to come up with the matrices $P_1, \dots, P_7$.
- Apparently it is not known how Strassen found these matrices.
- It cannot be done with 6 matrices:
  On multiplications of 2×2 *matrices* by Winograd, 1971.
- Faster algorithms use a similar approach, but are much more complicated.