Matrix Multiplication

1 General Method

The naive approach of matrix multiplication is to multiply every element of the rows of a matrix, say A, to the columns of another matrix, say B, and sum up the products and store them in another matrix, say C.

$$C[i][j] = \sum_{k=1}^{n} A[i][k].B[k][j] \tag{1}$$

1.1 Algorithm

Algorithm 1 Naive Matrix Multiplication

```
1:  procedure MATRIX-MULTIPLY(A, B)
2:      n ← number of rows of A
3:      Let C be a new n × n matrix
4:      for i = 1 to n − 1 do
5:          for j = 1 to n − 1 do
6:              C[i, j] ← 0
7:              for k = 1 to n − 1 do
8:                  C[i, j] ← C[i, j] + A[i, k] × B[k, j]
9:              end for
10:         end for
11:     end for
12:     return C
13: end procedure
```

1.2 Time Complexity

Here, multiplication and addition are executed once on each repetition in the innermost loop. So, the total number of multiplications and additions are,

$$M(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1 \tag{2}$$

Solving this will give us $T(n) = O(n^3)$, the total complexity incurred while calculating the matrix multiplication.

2 Divide and Conquer Method

This approach splits the matrices into smaller sub-matrices and recursively computes their products. Here, the matrix sizes are assumed to be taken in the order of 2^i to make the division easy. For the n x n matrices, split them into four $n/2$ x $n/2$ submatrices: Let us take two matrices, A and B, of size n x n,

$$A = \begin{bmatrix} A_0 & A_1 \\ A_2 & A_3 \end{bmatrix} \quad B = \begin{bmatrix} B_0 & B_1 \\ B_2 & B_3 \end{bmatrix}$$

Here each of $A_0, A_1, A_2...$ etc. are sub-matrices.

Now, for calculating their product, C=AB, where C_0, C_1, C_2, C_3 will be

$$C = \begin{bmatrix} A_0 * B_0 + A_1 * B_2 & A_0 * B_1 + A_1 * B_3 \\ A_2 * B_0 + A_3 * B_2 & A_2 * B_1 + A_3 * B_3 \end{bmatrix}$$

2.1 Algorithm

Algorithm 2 Divide and Conquer Matrix Multiplication

procedure RECURSIVE-MATRIX-MULTIPLY(A, B)
  $n \leftarrow$ number of rows of A
  if $n = 1$ **then**
    return $A[1,1] \times B[1,1]$
  end if
  Partition A and B into $n/2 \times n/2$ submatrices:
  A_0, A_1, A_2, A_3 and B_0, B_1, B_2, B_3
  $C_0 \leftarrow Recursive - Matrix - Multiply(A_0, B_0) + Recursive - Matrix - Multiply(A_1, B_2)$
  $C_1 \leftarrow Recursive - Matrix - Multiply(A_0, B_1) + Recursive - Matrix - Multiply(A_1, B_3)$
  $C_2 \leftarrow Recursive - Matrix - Multiply(A_2, B_0) + Recursive - Matrix - Multiply(A_3, B_2)$
  $C_3 \leftarrow Recursive - Matrix - Multiply(A_2, B_1) + Recursive - Matrix - Multiply(A_3, B_3)$
  Combine C_0, C_1, C_2, C_3 into C
  return C
end procedure

2.2 Time Complexity

So, in the above example, we need eight multiplications and four additions to get the final output. We know that adding two matrices takes $O(n^2)$ time complexity.

$$T(n) = 8T(n/2) + O(n^2) \tag{3}$$

Solving the above equation using the master's theorem will give us the final complexity of the divide-and-conquer methodology as $O(n^3)$.

From both the above algorithms, the time complexity has not improved. So, now we will look at the Strassen's algorithm.

3 Strassen's Algorithm

While the divide and conquer uses eight multiplications to get the final output, what Strassen's algorithm does is that it tries to lower the number of multiplications involved by giving formulas that involve seven multiplications instead of 8, thus reducing the complexity by a bit.
Let us take two matrices A and B of size 2 x 2

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Now, if we want to find the multiplication of these two matrices, we can do this by simply using four formulas, which are

$$C_{11} = A_{11} * B_{11} + A_{12} * B_{21} \tag{4}$$
$$C_{12} = A_{11} * B_{12} + A_{12} * B_{22} \tag{5}$$
$$C_{21} = A_{21} * B_{11} + A_{22} * B_{21} \tag{6}$$
$$C_{22} = A_{21} * B_{12} + A_{22} * B_{22} \tag{7}$$

Now, we can see that for the 2 x 2 matrix, there are eight multiplications and four additions. This will take linear time, but as the order of the matrix increases more than 2, it faces the problem we have seen in the divide and conquer methodology.

So to solve that, Strassen's Algorithm gives formulas that involve seven multiplications in place of eight.

Consider two 2×2 matrices A and B:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad B = \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

The standard multiplication of $C = A \cdot B$ involves 8 multiplications:

$$C = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

Strassen's algorithm reduces this to 7 multiplications by introducing intermediate products:

$$\begin{aligned} M_1 &= (a + d)(e + h), \\ M_2 &= (c + d)e, \\ M_3 &= a(f - h), \\ M_4 &= d(g - e), \\ M_5 &= (a + b)h, \\ M_6 &= (c - a)(e + f), \\ M_7 &= (b - d)(g + h). \end{aligned}$$

Using these, the resulting matrix C is computed as:

$$C = \begin{bmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 - M_2 + M_3 + M_6 \end{bmatrix}$$

Thus, by reusing results through these intermediate products, Strassen's algorithm reduces the number of multiplications from 8 to 7, at the expense of more additions and subtractions.

3.1 Algorithm

Algorithm 3 Strassen's Matrix Multiplication

procedure STRASSEN(A, B)
  $n \leftarrow$ number of rows of A
  if $n = 1$ **then**
    return $A[1,1] \times B[1,1]$
  end if
  Partition A and B into $n/2 \times n/2$ submatrices:
  $A_{11}, A_{12}, A_{21}, A_{22}$ and $B_{11}, B_{12}, B_{21}, B_{22}$
  Compute:
  $M_1 \leftarrow (A_{11} + A_{22})(B_{11} + B_{22})$
  $M_2 \leftarrow (A_{21} + A_{22})B_{11}$
  $M_3 \leftarrow A_{11}(B_{12} - B_{22})$
  $M_4 \leftarrow A_{22}(B_{21} - B_{11})$
  $M_5 \leftarrow (A_{11} + A_{12})B_{22}$
  $M_6 \leftarrow (A_{21} - A_{11})(B_{11} + B_{12})$
  $M_7 \leftarrow (A_{12} - A_{22})(B_{21} + B_{22})$
  Compute submatrices of C:
  $C_{11} \leftarrow M_1 + M_4 - M_5 + M_7$
  $C_{12} \leftarrow M_3 + M_5$
  $C_{21} \leftarrow M_2 + M_4$
  $C_{22} \leftarrow M_1 - M_2 + M_3 + M_6$
  Combine $C_{11}, C_{12}, C_{21}, C_{22}$ into C
  return C
end procedure

3.2 Time Complexity

As the number of multiplications has now changed to 7.
The time complexity now will be

$$\mathrm{T(n)} = \begin{cases} 1 & \text{if } n \leq 2 \\ 7T(n/2) + n^2 & \text{if } n > 0 \end{cases}$$

So, now the overall complexity comes to $O(n^{2.81})$.

There are also a few algorithms that have focused on lowering the complexity even more. Further research has used Strassen's algorithm as a base. Though the number of multiplications is still seven, I have made some other adjustments to make the complexity a little bit better.

4 Recent Advances

Algorithms like the Coppersmith-Winograd [2] and Alman-Williams [1] algorithms represent advanced approaches to matrix multiplication, using tensor representations and contraction optimization to minimize computational complexity. Coppersmith-Winograd focuses on expressing matrix multiplication as a tensor problem, applying recursive divide-and-conquer strategies to achieve a time complexity of $O(n^{2.376})$. Alman-Williams builds on this foundation, introducing laser methods and exploiting tensor symmetries to refine operations further, lowering the complexity to $O(n^{2.372})$. A detailed comparison of the number of arithmetic operations and input/output (I/O) complexity across different state-of-the-art matrix multiplication algorithms is summarized in Table 1.

The paper by Karstadt and Schwartz (2017) refines the Coppersmith-Winograd framework for matrix multiplication by optimizing tensor decompositions and balancing recursive strategies. Their approach reduces computational complexity slightly, achieving an improved theoretical upper bound for matrix multiplication. While primarily theoretical, their work demonstrates progress in lowering the complexity.

Table 1: Comparison of different state-of-the-art algorithms on matrix multiplication.

Year	Reference	Total Arithmetic Operations	Total I/O Complexity
1969	Strassen [5]	$7n^{\log_2 7} - 6n^2$	$6\left(\frac{\sqrt{3}n}{\sqrt{M}}\right)^{\log_2 7} \cdot M - 18n^2 + 3M$
1971	Winograd [6]	$6n^{\log_2 7} - 5n^2$	$5\left(\frac{\sqrt{3}n}{\sqrt{M}}\right)^{\log_2 7} \cdot M - 15n^2 + 3M$
2017	Karstadt, Schwartz [3]	$5n^{\log_2 7} - 4n^2 + 3n^2 \log_2 n$	$4\left(\frac{\sqrt{3}n}{\sqrt{M}}\right)^{\log_2 7} \cdot M - 12n^2 + 3n^2 \cdot \log_2\left(\frac{\sqrt{2}n}{\sqrt{M}}\right) + 5M$
2023	Schwartz, Vaknin [4]	$5n^{\log_2 7} - 4n^2 + 1.5n^2 \log_2 n$	$4\left(\frac{\sqrt{3}n}{\sqrt{M}}\right)^{\log_2 7} \cdot M - 12n^2 + 1.5n^2 \cdot \log_2\left(\frac{\sqrt{2}n}{\sqrt{M}}\right) + 5M$

References

[1] J. Alman and V. V. Williams. A refined laser method and faster matrix multiplication. *TheoretiCS*, Volume 3, Sept. 2024.

[2] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. STOC '87, page 1–6, New York, NY, USA, 1987. Association for Computing Machinery.

[3] E. Karstadt and O. Schwartz. Matrix multiplication, a little faster. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '17, page 101–110, New York, NY, USA, 2017. Association for Computing Machinery.

[4] O. Schwartz and N. Vaknin. Pebbling game and alternative basis for high performance matrix multiplication. *SIAM Journal on Scientific Computing*, 45(6):C277–C303, 2023.

[5] V. Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

[6] S. Winograd. On multiplication of 2× 2 matrices. *Linear algebra and its applications*, 1971.