Introduction to Algorithms **Note onStrassen Matrix Multiplication**
CS 4820, Spring 2026

In this note we discuss a divide and conquer algorithm for matrix multiplication. Given two $n \times n$ matrices A and B with entries a_{ij} and b_{ij} respectively, the product $C = AB$ is defined as having entries $c_{ij} = \sum_k a_{ik}b_{kj}$. The Naive method to compute this product is $O(n^3)$ time, as there are n^3 entries $a_{ik}b_{kj}$ that we need to compute. The question is: can we do this faster.

It is useful to note that adding matrices is faster. Given the two $n \times n$ matrices A and B, computing $D = A + B$ only takes $O(n^2)$ time.

Assume that n is even, than we can think of our matrices as each composed of 4 sub-matrices $A_{11}, A_{12}, A_{21}, A_{22}$, each of size $n/2 \times n/2$ only, as suggested below.

$$A = \begin{array}{|c|c|}\hline A_{11} & A_{12} \\ \hline A_{21} & A_{22} \\ \hline\end{array} \quad \text{and } B = \begin{array}{|c|c|}\hline B_{11} & B_{12} \\ \hline B_{21} & B_{22} \\ \hline\end{array}$$

Using this notation, we get that the matrix C can be written as

$$C = \begin{array}{|c|c|}\hline A_{11}B_{11} + A_{12}B_{21} & A_{12}B_{21} + A_{12}B_{22} \\ \hline A_{21}B_{11} + A_{22}B_{21} & A_{22}B_{21} + A_{22}B_{22} \\ \hline\end{array}$$

So a natural recursive algorithm would suggest to compute each of the 8 products of the smaller matrices to get the matrix C. This gives us the following recursion for the running time (assuming n even)

$$T(n) \leq 8T(n/2) + cn^2$$

for some constant c, where the cn^2 term is accounting for the 4 matrix additions we have to do, plus any data manipulations.

Assuming n is a power of 2, we can use this recursively, till we get down to 1 by 1 matrices, and get the following subproblems

level	number of subproblems	size of subproblems
0	1	$n \times n$
1	8	$n/2 \times n/2$
2	64	$n/4 \times n/4$
. . .		
i	8^i	$n/2^i \times n/2^i$
. . .		

The time we spend on matrix addition at level i is $8^i c(n/2^i)^2$. There will be $\log_2 n$ levels, so the total time is bounded by

$$\sum_{i=0}^{\log_2 n} 8^i \cdot c(\frac{n}{2^i})^2 = \sum_{i=0}^{\log_2 n} 8^i \cdot c\frac{n^2}{4^i} = cn^2 \sum_{i=0}^{\log_2 n} 2^i \leq 2cn^2 \cdot 2^{\log_2 n} = 2cn^3$$

To save over the $O(n^3)$ running time, we need to so fewer than 8 multiplication. Strassen's observation [1] was that this is possible:

$$\begin{array}{rcl}
M_1 & = & (A_{11}+A_{22})(B_{11}+B_{22}) \\
M_2 & = & (A_{21}+A_{22})B_{11} \\
M_3 & = & A_{11}(B_{12}-B_{22}) \\
M_4 & = & A_{22}(B_{21}-B_{11}) \\
M_5 & = & (A_{11}+A_{12})B_{22} \\
M_6 & = & (A_{21}-A_{11})(B_{11}+B_{12}) \\
M_7 & = & (A_{12}-A_{22})(B_{21}+B_{22})
\end{array}$$

Using the results of these 7 multiplications we can write

$$\begin{array}{rcl}
C_{11} & = & M_1+M_4-M_5+M_7 \\
C_{12} & = & M_3+M_5 \\
C_{21} & = & M_2+M_4 \\
C_{22} & = & M_1-M_2+M_3+M_6
\end{array}$$

This method gets us the product $C = AB$ with 7 multiplications with $n/2 \times n/2$ matrices and 10 matrix additions or subtractions to just set up the parts needed for the products and 8 more to get the final answer. A total of $O(n^2)$ time, though with 18 additions the constant in the $O(.)$ is getting rather big. We then get the following recurrence for the running time $T(n)$ for $n \times n$ matrix multiplication, assuming n is even:

$$T(n) \leq 7T(n/2) + cn^2$$

The number of levels is $\log_2 n$ again. Unrolling the recurrence we get the following bound on the running time, assuming n is a power of 2

$$T(n) \leq \sum_{i=0}^{\log_2 n} 7^i \cdot c(\frac{n}{2^i})^2 = cn^2 \sum_{i=0}^{\log_2 n} (\frac{7}{4})^i \leq cn^2(\frac{7}{4})^{\log_2 n} = c7^{\log_2 n} = cn^{\log_2 7} = O(n^{2.81})$$

There has been improvements in bounds for matrix multiplication since, the best know bound is $O(n^{2.373})$ (see [2, 3, 4] at this time, but decreasing the worse case does come with increasing the constant in $O(.)$ so at this point maybe Strassen is the most practical even for very big matrices. Another issue that may make other the classical methods better if the matrix is structured and sparse (as many 0 entries), possibly allowing much fewer multiplications.

References

[1] V. Strassen. "Gaussian Elimination is not Optimal". Numer. Math. 13 (4): 354–356, (1969).

[2] D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions, Proceedings of the nineteenth annual ACM symposium on Theory of computing, ACM New York, NY, USA, 1987, pp. 1–6.

[3] D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions, Journal of Symbolic Computation 9 (1990), no. 3, 251–280.

[4] Virginia Vassilevska Williams: Multiplying matrices faster than coppersmith-winograd. STOC 2012: 887-898.