1 Matrix Multiplication

Let $A = (a_{ij}), B = (b_{ij})$ be square, $n \times n$ matrices.

Then $C = A \cdot B = (c_{ij})$, where for all $i, j \in \{1, 2, \ldots, n\}$

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

This lead to a standard method of matrix multiplication, which takes time $\Theta(n^3)$ for multiplying two square $n \times n$ matrices.
For example, consider the 2×2 case:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

$$c_{11} = a_{11}b_{11} + a_{12}b_{21}$$

Here we have 4 additions and 8 multiplications.

Now, we consider Strassen's method for 2×2 matrix multiplication:

$$\begin{aligned}
m_1 &= (a_{11} + a_{22})(b_{11} + b_{22}) \\
m_2 &= (a_{12} - a_{22})(b_{21} + b_{22}) \\
m_3 &= (a_{11} - a_{21})(b_{11} + b_{12}) \\
m_4 &= (a_{11} + a_{12})b_{22} \\
m_5 &= (a_{21} + a_{22})b_{11} \\
m_6 &= a_{11}(b_{12} - b_{22}) \\
m_7 &= a_{22}(b_{21} - b_{11}) \\
c_{11} &= m_1 + m_2 - m_4 + m_7 \\
c_{12} &= m_4 + m_6 \\
c_{21} &= m_5 + m_7 \\
c_{22} &= m_1 - m_3 - m_5 - m_6
\end{aligned}$$

Here we have 18 additions and 7 multiplications! Why would we ever want to do this?
Suppose now that n is a power of 2.

Then the 2×2 methods above can be used to create recursive methods of multiplying $n \times n$ matrices together:

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

where A_{ij}, B_{ij}, C_{ij} are all $\frac{n}{2} \times \frac{n}{2}$ matrices.
If we apply the standard method recursively, the running times is captured by the following recurrence relation:

$$T(n) = 8T(n/2) + 4(n/2)^2$$

and if we analyze this we see that $T(n) = \Theta(n^3)$.
If we apply Strassen's method recursively, the running time is captured by this recurrence relation:

$$T(n) = 7T(n/2) + 18(n/2)^2$$

$$
\begin{aligned}
T(n) &= 7T(n/2) + 18(n/2)^2 \\
&= 7(7T(n/4) + 18(n/4)^2) + 18(n/2)^2 \\
&= 7(7(7T(n/8) + 18(n/8)^2 + 18(n/4)^2) + 18(n/2)^2 \\
&= 7^3T(n/2^3) + 7^2 \cdot 18(n/8)^2 + 7 \cdot 18(n/4)^2) + 18(n/2)^2 \\
&= 7^{k+1}T(n/2^{k+1}) + \sum_{i=0}^{k} 7^i \cdot 18(n/2^{i+1})^2 \\
&= 7^{k+1}T(n/2^{k+1}) + 18n^2 \sum_{i=0}^{k} \left(\frac{7}{16}\right)^i \\
&= 7^{\log_2(n)-1+1}T(n/2^{\log_2(n)-1+1}) + 18n^2 \sum_{i=0}^{\log_2(n)-1} \left(\frac{7}{16}\right)^i \\
&= 7^{\log_2(n)} + 18n^2 \sum_{i=0}^{\log_2(n)-1} \left(\frac{7}{16}\right)^i
\end{aligned}
$$

and if we follow a careful analysis, we will find that $T(n) = \Theta\left(7^{\log_2(n)}\right) \approx \Theta(n^{2.81})$, giving us a faster algorithm!

Strassen's method can be improved upon further. For example, there is Victor Pan's algorithm, which multiplies two 70×70 matrices with 143640 multiplications, and finding the running times relies on solving this recurrence relation:

$$T(n) = 143640T(n/70) + c(n/70)^2$$

Matrix multiplication is an important operation in computer science, so it is an active area of research and even small decreases in the exponent garner much attention from the computer scientists. Unless there have been recent results, the current fastest algorithm runs in time approximately $\Theta(n^{2.37})$, but the constants hidden by the asymptotic notation are huge! In practice, we do not work with huge matrices, and so the standard algorithm or Strassen's algorithm are likely still the most used algorithms for matrix multiplication.