Strassen's Matrix Multiplication Algorithm

- The standard method of matrix multiplication of two $n \times n$ matrices takes $O(n^3)$ operations.
- Strassen's algorithm is a *Divide-and-Conquer* algorithm that is asymptotically faster, i.e. $O(n^{\lg 7})$.
- The usual multiplication of two 2×2 matrices takes 8 multiplications and 4 additions. Strassen showed how two 2×2 matrices can be multiplied using only 7 multiplications and 18 additions.

Motivation

- For 2×2 matrices, there is no benefit in using the method.
- To see where this is of help, think about multiplication two $(2k) \times (2k)$ matrices.
- For this problem, the scalar multiplications and additions become matrix multiplications and additions.
- An addition of two matrices requires $O(k^2)$ time, a multiplication requires $O(k^3)$.
- Hence, multiplications are much more expensive and it makes sense to trade one multiplication operation for 18 additions.

Algorithm

Imagine that A and B are each partitioned into four square sub-matrices, each submatrix having dimensions $\frac{n}{2} \times \frac{n}{2}$.

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

, where

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

Strassen's algorithm

Strassen "observed" that:

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

, where

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

Complexity

- $T(n) = 7T(\frac{n}{2}) + cn^2$, where c is a fixed constant. The term cn^2 captures the time for the matrix additions and subtractions needed to compute $P_1, ..., P_7$ and $C_{11}, ..., C_{22}$.
- The solution works out to be:

$$T(n) = \Theta(n^{lg7}) = O(n^{2.81}).$$

- Currently, the best known algorithm was given by Coppersmith and Winograd and has time complexity $O(n^{2.376})$.