Fast matrix multiplication

Yuval Filmus

May 3, 2017

1 Problem statement

How fast can we multiply two square matrices? To make this problem precise, we need to fix a computational model. The model that we use is the *unit cost arithmetic model*. In this model, a program is a sequence of instructions, each of which is of one of the following forms:

1. Load input: $t_n \leftarrow x_i$, where x_i is one of the inputs.
2. Load constant: $t_n \leftarrow c$, where $c \in \mathbb{R}$.
3. Arithmetic operation: $t_n \leftarrow t_i \circ t_j$, where $i, j < n$, and $\circ \in \{+, -, \cdot, /\}$.

In all cases, n is the number of the instruction.

We will say that an arithmetic program *computes the product of two $n \times n$ matrices* if:

1. Its inputs are x_{ij}, y_{ij} for $1 \leq i, j \leq n$.
2. For each $1 \leq i, k \leq n$ there is a variable t_m whose value is always equal to
$$\sum_{j=1}^{n} x_{ij} y_{jk}.$$
3. The program never divides by zero.

The *cost* of multiplying two $n \times n$ matrices is the minimum length of an arithmetic program that computes the product of two $n \times n$ matrices.

Here is an example. The following program computes the product of two 1×1 matrices:

$$\begin{aligned} t_1 &\leftarrow x_{11} \\ t_2 &\leftarrow y_{11} \\ t_3 &\leftarrow t_1 \cdot t_2 \end{aligned}$$

The variable t_3 contains $x_{11}y_{11}$, and so satisfies the second requirement above for $(i, k) = (1, 1)$. This shows that the cost of multiplying two 1×1 matrices is at most 3.

Usually we will not spell out all steps of an arithmetic program, and use simple shortcuts, such as the ones indicated by the following program for multiplying two 2×2 matrices:

$$\begin{aligned} z_{11} &\leftarrow x_{11}y_{11} + x_{12}y_{21} \\ z_{12} &\leftarrow x_{11}y_{12} + x_{12}y_{22} \\ z_{21} &\leftarrow x_{21}y_{11} + x_{22}y_{21} \\ z_{22} &\leftarrow x_{21}y_{12} + x_{22}y_{22} \end{aligned}$$

When spelled out, this program uses 20 instructions: 8 load instructions, 8 product instructions, and 4 addition instructions.

Along these lines we can write a program that multiplies two $n \times n$ matrices using $2n^2$ load instructions, n^3 product instructions, and $n^2(n-1)$ addition instructions, for a total of $\Theta(n^3)$ instructions. Can we do any better?

1.1 Uniformity

The usual algorithm for matrix multiplication is often written using three nested *for* loops:

```
for (i = 0; i < n; i++)
  for (j = 0; j < n; j++)
    for (k = 0; k < n; k++)
      Z[i][j] += X[i][k] * Y[k][j]
```

(Assuming Z is initialized to zero.) This is an example of a *uniform* algorithm: intuitively, the same algorithm is used for all values of n. In contrast, the model described above is *non-uniform*: for each n there might be a completely different algorithm. This might be an issue since in practice we don't want to write a different algorithm for each n. Moreover, what good is an algorithm for multiplying $n \times n$ matrices if it is difficult to *find*?

This suggests considering the uniform complexity of matrix multiplication, in which the program is not given for each n separately, but rather there is a "uniform way" to describe the programs for all different n. We won't define this concept precisely here. It turns out that in the case of matrix multiplication, uniformity doesn't affect the theoretical complexity, as we indicate below.

2 Karatsuba multiplication

It turns out that we can multiply $n \times n$ matrices faster than $O(n^3)$, using a sophisticated divide-and-conquer approach. A simpler instance of this approach is the Karatsuba multiplication algorithm for multiplying two n-bit numbers.

Integer multiplication algorithms are usually analyzed using the *bit complexity* model (although Fürer advocates using a "word complexity" model instead). This model is similar to the arithmetic model that we described above, with a crucial difference: all our variables are now *bits* rather than *real numbers*. The basic operations allowed are unary NOT and binary AND, OR, and XOR. (The basic inventory of operations is not so important as long as there is a finite number of them — changing the inventory changes the complexity measure by at most a constant factor.)

The multiplication algorithm we all learned in school can be described as follows:

1. Input: $x = x_{n-1} \dots x_0$, $y = y_{n-1} \dots y_0$.
2. For $0 \leq i \leq n-1$, compute $Y_i = x_i \cdot y \cdot 2^i$.
3. Add the numbers $Y_0, \dots, Y_{n-1}$.

Computing each Y_i uses n AND operations, so computing all Y_i takes n^2 operations. Adding two numbers takes $O(n)$ operations, and altogether the numbers $Y_0, \dots, Y_{n-1}$ can be summed in $O(n^2)$ operations. The overall complexity is $O(n^2)$.

We can try to improve on this using a divide-and-conquer approach. Assume for simplicity that n is even, and split x into two bit-strings, $x_h = x_{n-1} \dots x_{n/2}$ and $x_\ell = x_{n/2-1} \dots x_0$. As numbers, $x = 2^{n/2} x_h + x_\ell$. Split y similarly into $y = 2^{n/2} y_h + y_\ell$. A simple divide-and-conquer approach for computing xy uses the formula

$$xy = 2^n x_h y_h + 2^{n/2}(x_h y_\ell + x_\ell y_h) + x_\ell y_\ell.$$

How do we use this formula in an algorithm? We compute the four products $x_h y_h, x_h y_\ell, x_\ell y_h, x_\ell y_\ell$ recursively, and then put them together using several addition operations. If we denote by $T(n)$ the number of operations that this algorithm uses, then $T(n)$ satisfies the recurrence

$$T(n) = 4T(n/2) + \Theta(n),$$

whose disappointing solution is $T(n) = \Theta(n^2)$. This algorithm is thus no better than the trivial one.

Karatsuba's algorithm improves on this by using a smarter formula. It first computes recursively the following *three* smaller products:

$$m_1 = x_h y_h, m_2 = x_\ell y_\ell, m_3 = (x_h + x_\ell)(y_h + y_\ell).$$

(The numbers $x_h + x_\ell$ and $y_h + y_\ell$ are in fact $n/2+1$ bits long, but this doesn't make any difference.) Given these numbers, it calculates

$$xy = 2^n m_1 + 2^{n/2}(m_3 - m_1 - m_2) + m_3.$$

To check that this formula is valid, it suffices to verify that

$$m_3 - m_1 - m_2 = x_h y_\ell + x_\ell y_h.$$

The number of operations $T(n)$ used by this algorithm satisfies the recurrence

$$T(n) = 3T(n/2) + \Theta(n),$$

whose solution is $T(n) = \Theta(n^{3/2})$, a significant improvement! Similar ideas will enable us to improve on the trivial matrix multiplication algorithm.

Karatsuba's algorithm has been improved by Toom and Cook using a similar approach, and then by Schönhage and Strassen who gave an FFT-based $O(n \log n \log\log n)$ algorithm. The asymptotically fastest algorithm currently known is Fürer's algorithm, whose running time is $O(n \log n 2^{\log^* n})$, where $\log^* n$ is the number of times you have to apply log until the number gets below 1. Some people conjecture that integer multiplication requires $\Omega(n \log n)$ operations in the bit complexity model, though no lower bound is known.

The bit complexity of integer multiplication is sometimes denoted $M(n)$. It is known to control the complexity of integer division, which can be done in $O(M(n))$ bit operations.

3 Strassen's algorithm

Strassen tried to prove that the $\Theta(n^3)$ algorithm is optimal, and ended up improving it, using a smart way to multiply two 2×2 matrices:

$$\begin{aligned}
m_1 &\leftarrow (x_{11} + x_{22})(y_{11} + y_{22}) \\
m_2 &\leftarrow (x_{21} + x_{22})y_{11} \\
m_3 &\leftarrow x_{11}(y_{12} - y_{22}) \\
m_4 &\leftarrow x_{22}(y_{21} - y_{11}) \\
m_5 &\leftarrow (x_{11} + x_{12})y_{22} \\
m_6 &\leftarrow (x_{21} - x_{11})(y_{11} + y_{12}) \\
m_7 &\leftarrow (x_{12} - x_{22})(y_{21} + y_{22}) \\
z_{11} &\leftarrow m_1 + m_4 - m_5 + m_7 \\
z_{12} &\leftarrow m_3 + m_5 \\
z_{21} &\leftarrow m_2 + m_4 \\
z_{22} &\leftarrow m_1 - m_2 + m_3 + m_6
\end{aligned}$$

A short calculation, which we skip, shows that this algorithm indeed computes the product of two 2×2 matrices. (It is an edifying exercise to verify this using a computer algebra system.)

An important property of Strassen's algorithm is that it never relies on commutativity of the entries of the matrices. That is, it never uses $x_{ij}y_{k\ell} = y_{k\ell}x_{ij}$. This guarantees that it is a valid algorithm for multiplying two 2×2 matrices over arbitrary rings, and is important for using it recursively.

How do we use Strassen's algorithm to multiply larger matrices? Suppose for simplicity that we want to multiply two $n \times n$ matrices, when n is a power of 2. We can think of an $n \times n$ matrix as a 2×2 matrix whose entries are $(n/2) \times (n/2)$ matrices. Multiplying these 2×2 block matrices is the same as multiplying the original matrices. In more detail, consider the following equation:

$$\begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix} \begin{pmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{pmatrix} = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}.$$

The entries of X_{11} are x_{ij} for $1 \leq i, j \leq n/2$. The entries of X_{12} are x_{ij} for $1 \leq i \leq n/2$ and $n/2+1 \leq j \leq n$, and so on. Consider the (i, j)th entry of Z_{11}:

$$\begin{aligned}
(Z_{11})_{ij} &= (X_{11}Y_{11})_{ij} + (X_{12}Y_{21})_{ij} \\
&= \sum_{k=1}^{n/2} (X_{11})_{ik}(Y_{11})_{kj} + (X_{12})_{ik}(Y_{21})_{kj} \\
&= \sum_{k=1}^{n/2} x_{ik}y_{kj} + x_{i(n/2+k)}y_{(n/2+k)j} \\
&= \sum_{k=1}^{n} x_{ik}y_{kj} \\
&= z_{ij}.
\end{aligned}$$

A similar calculation works for the other three blocks. In other words, if we multiply these block matrices, then the product matrix, when "flattened", will contain the product of the original (non-block) matrices.

This suggests that we can multiply two $n \times n$ matrices by using Strassen's original algorithm to multiply $\begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix}$ and $\begin{pmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{pmatrix}$. Strassen's algorithm contains addition and multiplication instructions. Addition instructions we can handle by adding the matrices, at a cost of $(n/2)^2$ per addition. Multiplication instructions we handle by calling Strassen's algorithm recursively. Since there are only a constant number of addition and subtraction instructions, the arithmetic complexity $T(n)$ of Strassen's algorithm satisfies the recurrence

$$T(n) = 7T(n/2) + \Theta(n^2),$$

whose solution is $T(n) = \Theta(n^{\log_2 7})$, where $\log_2 7 \approx 2.80 < 3$.

What if n is not a power of 2? In that case, we can pad the original matrices to the next power of 2 $N \leq 2n$, to obtain an algorithm whose arithmetic complexity is $\Theta(N^{\log_2 7}) = O(N^{\log_2 7})$. (In practice it might be better to pad the matrices so that n is always even, and then pad them again as needed during the recursion.)

3.1 The exponent of matrix multiplication

Strassen's algorithm shows that we can improve on the trivial $\Theta(n^3)$ algorithm. But by how much? To make this question precise, let us define ω, the exponent of matrix multiplication.

Let T consist of all values x such that two $n \times n$ matrices can be multiplied using $O(n^x)$ arithmetic operations. We define ω as the infimum of T. As a result, for every $\epsilon > 0$, two $n \times n$ matrices can be multiplied using $O(n^{\omega+\epsilon})$ arithmetic operations, where the hidden constant can depend on ϵ.

Why define ω in this roundabout way? Why not define it as the minimum value such that two $n \times n$ matrices can be multiplied using $O(n^\omega)$ arithmetic operations? The reason is that it is suspected that $\omega = 2$ but $2 \notin T$. That is, it is conjectured that two $n \times n$ matrices can be multiplied using $O(n^{2+\epsilon})$ arithmetic operations for every $\epsilon > 0$, but that there is no $O(n^2)$ algorithm. (In fact, Ran Raz proved a lower bound of $\Omega(n^2 \log n)$ under some mild restrictions.)

Often in papers one sees a running time quoted as $O(n^\omega)$, say. This is, however, formally wrong, since ω is an infimum rather than a minimum. Such running times should be interpreted as $O(n^{\omega+\epsilon})$ for all $\epsilon > 0$.

The exponent ω is important since many other operations on matrices have the same exponent, for example matrix inversion and solving linear equations.

Another quantity of interest in the field is α, the exponent of rectangular matrix multiplication. It is the supremum of the values $x < 1$ such that an $n \times n^x$ matrix can be multiplied by an $n^x \times n$ matrix in time $O(n^{2+o(1)})$. It is known that $\alpha > 0$ (and there are some concrete algorithms showing that), and the conjecture $\omega = 2$ is equivalent to the conjecture $\alpha = 1$.

Strassen's algorithm is considered on the verge of being practical (some claim it is practical, some claim it is not). Most developments following Strassen's algorithm are of purely theoretical interest. The reader interested in practical matrix multiplication should take a look at the 2016 survey of Dumas and Pan, *Fast Matrix Multiplication and Symbolic Computation.*

3.2 Strassen's algorithm as tensor decomposition

We have described Strassen's algorithm (in its base case) as a sequence of instructions. A different way of describing it is via the following identity:

$$\begin{aligned}\sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2} x_{ij}y_{jk}z_{ik} = {} & (x_{11}+x_{22})(y_{11}+y_{22})(z_{11}+z_{22}) \\ & + (x_{21}+x_{22})y_{11}(z_{21}-z_{22}) \\ & + x_{11}(y_{12}-y_{22})(z_{12}+z_{22}) \\ & + x_{22}(y_{21}-y_{11})(z_{11}+z_{21}) \\ & + (x_{11}+x_{12})y_{22}(z_{12}-z_{11}) \\ & + (x_{21}-x_{11})(y_{11}+y_{12})z_{22} \\ & + (x_{12}-x_{22})(y_{21}+y_{22})z_{11}.\end{aligned}$$

How did we generate this identity? In the left-hand side, the coefficient of the formal variable z_{ik} is the value that z_{ik} should have at the end of the algorithm, that is, $\sum_{j=1}^{2} x_{ij}y_{jk}$. In the right-hand side, the coefficient of the formal variable z_{ik} is the value that z_{ik} has at the end of the algorithm. For example, $z_{11} = m_1 + m_4 - m_5 + m_7$, and accordingly, z_{11} appears in lines $1, 4, 5, 7$ (with a negative sign on line 5), in which m_1, m_4, m_5, m_7 are multiplied by some linear combination of the z_{ik}'s.

Later on we will show that this identity can be summarized as

$$R(\langle 2,2,2\rangle) \leq 7.$$

Here $\langle 2,2,2\rangle$ corresponds to the task of multiplying two 2×2 matrices (the numbers $2,2,2$ correspond to the dimensions of the matrices involved), and 7 corresponds to the number of lines in the decomposition.

4 Bilinear notation

Our task in this section is to explain what the notation $R(\langle 2,2,2\rangle) \leq 7$ means. We will start with a different way of viewing matrices.

Suppose that A is an $n \times m$ matrix with entries a_{ij}, and consider the expression

$$x^T A y = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix} A \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} x_i y_j.$$

This expression is called a *bilinear function* (in the same way that $\sum_i a_i x_i$, corresponding to a vector, is a linear function), but we will think of it as an expression (in fact, a polynomial) in the formal variables $x_1, \ldots, x_n, y_1, \ldots, y_m$.[1]

We will particularly be interested in the notion of *rank* from the point of view of this new notation. A matrix A has *row rank* 1 if there exists a row vector v such that all rows are multiples of v. If the ith row is $\beta_i v$, then $a_{ij} = \beta_i v_j$, and so

$$x^T A y = \sum_{i=1}^{n} \sum_{j=1}^{m} x_i y_j \beta_i v_j = \sum_{i=1}^{n} \sum_{j=1}^{m} \beta_i x_i v_j y_j = \left(\sum_{i=1}^{n} \beta_i x_i \right) \left(\sum_{j=1}^{m} v_j y_j \right).$$

You can think of this equation as a *formal* equation, or as an equation of formal expressions. It expresses the fact that A is the outer product of the column vector β and the row vector v.[2]

A matrix A has *row rank* r if there are r row vectors $v_1, \ldots, v_r$ such that all rows are linear combinations of $v_1, \ldots, v_r$, say the ith row is $\sum_{\ell=1}^{r} \beta_{i\ell} v_\ell$. In that case

$$\begin{aligned} x^T A y &= \sum_{i=1}^{n} \sum_{j=1}^{m} x_i y_j \sum_{\ell=1}^{r} \beta_{i\ell} (v_\ell)_j \\ &= \sum_{\ell=1}^{r} \sum_{i=1}^{n} \sum_{j=1}^{m} \beta_{i\ell} x_i (v_\ell)_j y_j \\ &= \sum_{\ell=1}^{r} \left(\sum_{i=1}^{n} \beta_{i\ell} x_i \right) \left(\sum_{j=1}^{m} (v_\ell)_j y_j \right). \end{aligned}$$

In other words, A is a sum of r rank one matrices.

This shows that the row rank of a matrix A is the minimum number of rank one matrices which sum to A. Since the definition of rank one matrix is symmetric in rows and columns (that is, it remains the same if we switch rows and columns), we obtain a proof of the non-trivial fact that the row rank of a matrix equals its column matrix. Indeed, both are equal to the *tensor rank* of the matrix, which is the minimal number of outer products which sum to the matrix.

4.1 3D tensors

We can do exactly the same in more than two dimensions. We will only be interested in three-dimensional tensors. These are three-dimensional arrays, which correspond to trilinear functions in the same way that two-dimensional arrays (matrices) correspond to bilinear functions (and one-dimensional arrays, that is, vectors, correspond to linear functions). We will identify $a \times b \times c$ tensors with the corresponding trilinear function

$$\sum_{i=1}^{a} \sum_{j=1}^{b} \sum_{k=1}^{c} x_i y_j z_k t_{ijk}.$$

[1] The sophisticated reader can notice that A is a linear transformation from some vector sapce V with basis $x_1, \ldots, x_n$ to some other vector space W with basis $y_1, \ldots, y_m$, and $x_i \otimes y_j$ is a basis for the tensor product $V \otimes W$, and this expression is just the expansion of $A \in V \otimes W$ in this basis.

[2] If you think of A as belonging to the tensor product $V \otimes W$, then this shows that $A = (\sum_{i=1}^{n} \beta_i x_i) \otimes (\sum_{j=1}^{m} v_j y_j)$.

A rank one tensor (the analog of an outer product) is a tensor in which $t_{ijk} = \alpha_i \beta_j \gamma_k$. The corresponding trilinear function is

$$\left(\sum_{i=1}^{a} \alpha_i x_i\right)\left(\sum_{j=1}^{b} \beta_j y_j\right)\left(\sum_{k=1}^{c} \gamma_k z_k\right).$$

The *rank* of a tensor T, denoted $R(T)$, is the minimal number of rank one tensors which sum up to T. In contrast to matrix rank, which can be calculated efficiently using Gaussian elimination, tensor rank is NP-hard to compute (as shown by Håstad).

We will be interested in particular tensors called *matrix multiplication tensors*. The $\langle n, m, p\rangle$ tensor is an $nm \times mp \times np$ tensor whose rows are indexed by pairs of indices from $[n] \times [m]$, whose columns are indexed by $[m] \times [p]$, and whose depths are indexed by $[n] \times [p]$. The $(i_r, j_r), (j_c, k_c), (i_d, k_d)$-entry is 1 if $i_r = i_d$, $j_r = j_c$, and $k_c = k_d$, and 0 otherwise. The corresponding trilinear function is

$$\langle n, m, p\rangle = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p} x_{ij} y_{jk} z_{ik}.$$

In what way does this correspond to matrix multiplication? Suppose that we multiply an $n \times m$ matrix X whose entries are x_{ij} by an $m \times p$ matrix Y whose entries are y_{ij}, obtaining an $n \times p$ matrix W whose entries are w_{ik}. Then

$$w_{ik} = \sum_{j=1}^{m} x_{ij} y_{jk},$$

and so

$$\langle n, m, p\rangle = \sum_{i=1}^{n}\sum_{k=1}^{p} z_{ik} \sum_{j=1}^{m} x_{ij} y_{jk} = \sum_{i=1}^{n}\sum_{k=1}^{p} z_{ik} w_{ik}.$$

In other words, the coefficient of z_{ik} in $\langle n, m, p\rangle$ is the bilinear function w_{ik} which is the (j, k)th entry of the matrix XY. We can thus think of $\langle n, m, p\rangle$ as describing the task of multiplying an $n \times m$ matrix by an $m \times p$ matrix.

Another interpretation of $\langle n, m, p\rangle$ is

$$\langle n, m, p\rangle = \mathrm{Tr}(XYZ^T),$$

where again X is an $n \times m$ matrix with entries x_{ij}, Y is an $m \times p$ matrix with entries y_{jk}, and Z is an $n \times p$ matrix with entries z_{ik}. (In terms of $W = XY$, this is $\mathrm{Tr}(WZ^T)$, which is the inner product of XY and Z.)

The rank of $\langle n, n, n\rangle$ is intimately connected to the complexity of matrix multiplication, as we show in detail below.

At this point we have explained all pieces of notation in the statement $R(\langle 2, 2, 2\rangle) \leq 7$; in fact, it can be shown that $R(\langle 2, 2, 2\rangle) = 7$. We have seen that the algorithm corresponding to the bound $R(\langle 2, 2, 2\rangle) \leq 7$ implies that $\omega \leq \log_2 7$, and this can be generalized.

4.2 Algorithms from tensor rank

Given a bound on the tensor rank of $\langle n, n, n\rangle$, can we construct an algorithm for multiplying arbitrary square matrices?

Theorem 1. *If $R(\langle n, n, n\rangle) \leq r$ then $\omega \leq \log_n r$. In other words,*

$$n^\omega \leq R(\langle n, n, n\rangle).$$

Proof. The proof is very similar to the analysis of Strassen's algorithm. The first step is an algorithm for multiplying two $n \times n$ matrices using r "non-commutative" multiplications (in the literature, these are known as *essential* multiplications, in contrast to multiplication by real constants).

Suppose that the decomposition is

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p} x_{ij}y_{jk}z_{ik} = \sum_{\ell=1}^{r}\left(\sum_{ij} a_{\ell ij}x_{ij}\right)\left(\sum_{jk} b_{\ell jk}y_{jk}\right)\left(\sum_{ik} c_{\ell ik}z_{ik}\right). \tag{1}$$

Our algorithm proceeds as follows, on input X, Y:

1. Compute $s_\ell = \sum_{ij} a_{\ell ij}x_{ij}$ for $1 \le \ell \le r$.
2. Compute $t_\ell = \sum_{jk} b_{\ell jk}y_{jk}$ for $1 \le \ell \le r$.
3. Compute $m_\ell = s_\ell t_\ell$ for $1 \le \ell \le r$.
4. Compute $\hat{z}_{ik} = \sum_{\ell=1}^{r} m_\ell c_{\ell ik}$ for all i, k.

We claim that $\hat{z}_{ik}$ is the (i,k)th entry of $Z = XY$. Indeed, notice that

$$\hat{z}_{ik} = \sum_{\ell=1}^{r}\left(\sum_{ij} a_{\ell ij}x_{ij}\right)\left(\sum_{jk} b_{\ell jk}y_{jk}\right)c_{\ell ik}.$$

After a bit of algebra, this shows that

$$\sum_{i=1}^{n}\sum_{k=1}^{p} \hat{z}_{ik}z_{ik} = \sum_{\ell=1}^{r}\left(\sum_{ij} a_{\ell ij}x_{ij}\right)\left(\sum_{jk} b_{\ell jk}y_{jk}\right)\left(\sum_{ik} c_{\ell ik}z_{ik}\right).$$

On the other hand, by (1), the right-hand side is equal to

$$\sum_{i=1}^{n}\sum_{k=1}^{p}\left(\sum_{j=1}^{m} x_{ij}y_{jk}\right)z_{ik}.$$

Comparing coefficients, we see that

$$\hat{z}_{ik} = \sum_{j=1}^{m} x_{ij}y_{jk} = (XY)_{ik}.$$

This shows that the algorithm correctly computes the product of two $n \times n$ matrices.

Just as we did for Strassen's algorithm, we can apply this algorithm recursively for multiplying larger square matrices whose dimensions are a power of n. Given two $N \times N$ matrices (where N is a power of n), we think of them as $n \times n$ block matrices, whose entries are themselves $(N/n) \times (N/n)$ matrices. We can run the algorithm described above on these block matrices. Computing $s_1, \ldots, s_r, t_1, \ldots, t_r$ uses $O(N^2)$ operations. We compute each m_ℓ recursively. Finally, given $m_1, \ldots, m_r$, we can compute each $\hat{z}_{ik}$ using $O(N^2)$ operations. Since there are only a constant number of output entries, the overall complexity is $O(N^2)$ plus r products of $(N/n) \times (N/n)$ matrices. The number of operations $T(N)$ satisfies the recurrence

$$T(N) = rT(N/n) + \Theta(N^2),$$

whose solution (assuming[3] $r > n^2$) is $T(N) = \Theta(N^{\log_n r})$.

As in the case of Strassen's algorithm, if N is not a power of n then we can pad it to the next power of n, which is at most a factor of n larger. Since n is a constant, the resulting algorithm uses $\Theta(N^{\log_n r})$ arithmetic operations for *all* N. □

[3]It is not too hard to show that $r \ge n^2$ always, essentially since we have to compute n^2 "independent" entries. It turns out that in fact $r > n^2$ always.

As a corollary, we get that ω is at most the *tensor rank exponent* c, which is the infimum of the values x such that $R(\langle n,n,n\rangle) = O(n^x)$. Indeed, for every $\epsilon > 0$ there are constants n_0, C such that for $n \geq n_0$, $R(\langle n,n,n\rangle) \leq Cn^{c+\epsilon}$. Theorem 1 shows that

$$\omega \leq \log_n(Cn^{c+\epsilon}) = c + \epsilon + \frac{\log C}{\log n}.$$

Taking the limit $n \to \infty$, we see that $\omega \leq c + \epsilon$. Since this holds for all $\epsilon > 0$, we see that $\omega \leq c$.

Surprisingly, the converse also holds, and in fact $c = \omega$. We prove this interesting result in the next section.

5 Bilinear algorithms

Given an arbitrary algorithm for multiplying two $n \times n$ matrices, we will show how to convert it to an algorithm whose form is very similar to Strassen's. In particular, we will be able to read from this algorithm a bound on the rank of $\langle n,n,n\rangle$.

The main observation that facilitates this conversion is that the entries of the product matrix are a *quadratic* polynomial in the entries of the input matrices. This allows us to maintain only the "degree 2 prefix" of all values calculated in the program, and brings us almost all the way to the required normal form.

5.1 Decomposition into homogeneous parts

Let P be a multivariate polynomial. We can write P as a linear combination of monomials. The *(total) degree* of a monomial m is the sum of the degrees of the variables appearing in it. For example, $\deg(x_{12}y_{23}^2) = 3$ (you can also think of it as the number of variables which are multiplied together, counted with repetition). A polynomial is *homogeneous* if all monomials appearing in it have the same degree. For example, $x_{12} + y_{12}$ is homogeneous, but $x_{12} + y_{12}^2$ is not.

Every polynomial P can be written as a sum of homogeneous polynomials, one of each degree. This is done by taking all monomials of degree d appearing in P, together with their coefficients, into a new polynomial $P^{=d}$. If we do this for all degrees, then P is the sum of all $P^{=d}$. For example, if $P = 1 + x_{12} + 3y_{22} + x_{13}y_{22} - 4x_{12}^2$ then

$$P^{=0} = 1,\ P^{=1} = x_{12} + 3y_{22},\ P^{=2} = x_{13}y_{22} - 4x_{12}^2.$$

The degree of P, denoted $\deg P$, is the maximal degree of a monomial in P. We thus have

$$P = \sum_{d=0}^{\deg P} P^{=d}.$$

The main observation behind our transformation is that if z_{ik} are the entries of the product matrix, then z_{ik} is homogeneous of degree 2 as a function of the entries of the matrices being multiplied. In fact, z_{ik} is a *bilinear* function of the x_{ij} and y_{ij}, which says that the degree 2 monomials are all products of an x_{ij} and a $y_{k\ell}$. We will use this stronger property in the sequel.

Our transformation is embedded in the following theorem, which we only describe for programs without division. Handling division is not much more difficult, and we leave it as a nice exercise.

Theorem 2. *Let Π be a (valid) program for multiplying two $n \times n$ matrices without division. Replace each instruction with the following mini-program, to comprise a new program Π':*

- *Replace $t_p \leftarrow x_{ij}$ with*

$$\begin{aligned} t_{p,0} &\leftarrow 0 \\ t_{p,1} &\leftarrow x_{ij} \\ t_{p,2} &\leftarrow 0 \end{aligned}$$

- *Replace* $t_p \leftarrow y_{ij}$ *with*

$$\begin{aligned} t_{p,0} &\leftarrow 0 \\ t_{p,1} &\leftarrow y_{ij} \\ t_{p,2} &\leftarrow 0 \end{aligned}$$

- *Replace* $t_p \leftarrow r$ *(for* $r \in \mathbb{R}$*) with*

$$\begin{aligned} t_{p,0} &\leftarrow r \\ t_{p,1} &\leftarrow 0 \\ t_{p,2} &\leftarrow 0 \end{aligned}$$

- *Replace* $t_p \leftarrow t_q + t_r$ *with*

$$\begin{aligned} t_{p,0} &\leftarrow t_{q,0} + t_{r,0} \\ t_{p,1} &\leftarrow t_{q,1} + t_{r,1} \\ t_{p,2} &\leftarrow t_{q,2} + t_{r,2} \end{aligned}$$

- *Replace* $t_p \leftarrow t_q - t_r$ *with*

$$\begin{aligned} t_{p,0} &\leftarrow t_{q,0} - t_{r,0} \\ t_{p,1} &\leftarrow t_{q,1} - t_{r,1} \\ t_{p,2} &\leftarrow t_{q,2} - t_{r,2} \end{aligned}$$

- *Replace* $t_p \leftarrow t_q t_r$ *with*

$$\begin{aligned} t_{p,0} &\leftarrow t_{q,0} t_{r,0} \\ t_{p,1} &\leftarrow t_{q,0} t_{r,1} + t_{q,1} t_{r,0} \\ t_{p,2} &\leftarrow t_{q,0} t_{r,2} + t_{q,1} t_{r,1} + t_{q,2} t_{r,0} \end{aligned}$$

Then Π' *also computes the product of two* $n \times n$ *matrices, and moreover* $t_{p,d} = t_p^{=d}$ *for all* p *and* $d \in \{0,1,2\}$.

Proof. If we show that $t_{p,d} = t_p^{=d}$, then it follows that Π' also computes the product of two $n \times n$ matrices. Indeed, for every $1 \le i,k \le n$ there is p such that in Π it holds that $t_p = \sum_{j=1}^n x_{ij} y_{jk}$. Since $t_p = t_p^{=2}$, in Π' it holds that $t_{p,2} = \sum_{j=1}^n x_{ij} y_{jk}$.

We prove that $t_{p,d} = t_p^{=d}$ by induction. This is easy to check for the base cases $t_p \leftarrow x_{ij}$, $t_p \leftarrow y_{ij}$, and $t_p \leftarrow r$. Suppose now that $t_{q,d} = t_q^{=d}$ and $t_{r,d} = t_r^{=d}$ holds for some q, r and $d \in \{0,1,2\}$. The result for $t_p \leftarrow t_q \pm t_r$ is not difficult to verify directly (since $(P+Q)^{=d} = P^{=d} + Q^{=d}$), and the result for $t_p \leftarrow t_q \cdot t_r$ follows from the formula $\deg(m_1 m_2) = (\deg m_1)(\deg m_2)$ for the degree of the product of two monomials m_1, m_2. □

We encourage the reader to fill in the various details we left out in the proof of the preceding theorem.

Theorem 2 is helpful since the expressions computed by $t_{p,0}, t_{p,1}, t_{p,2}$ have a particular form, as indicated by the following theorem.

Theorem 3. *Let* Π' *be the program described in Theorem 2. Let* $p_1, \ldots, p_m$ *be all the product instructions in* Π*, say* $t_{p_i} \leftarrow t_{q_i} \cdot t_{r_i}$.

For each original instruction t_p*:*

1. $t_{p,0}$ *is a constant (doesn't depend on the* x_{ij} *or* y_{ij}*).*

2. $t_{p,1}$ *is a linear combination of the* x_{ij} *and* y_{ij}*.*

3. $t_{p,2}$ *is a linear combination of products* $t_{q_i,1}t_{r_i,1}$*.*

Proof. The proof is again by induction. We have to consider the various types of instructions. If the pth instruction is $t_p \leftarrow x_{ij}$, then $t_{p,0} = t_{p,2} = 0$ and $t_{p,1}$ is the linear combination x_{ij}. The instruction $t_p \leftarrow y_{ij}$ is handled similarly. If the pth instruction is $t_p \leftarrow r$, then $t_{p,0} = r$ is constant, and $t_{p,1} = t_{p,2} = 0$.

Suppose now that the pth instruction is $t_p \leftarrow t_q + t_r$. First, $t_{p,0} = t_{q,0} + t_{r,0}$ is the sum of two constants, and so also a constant. Second, $t_{p,1} = t_{q,1} + t_{r,1}$ is the sum of two linear combinations of x_{ij}, y_{ij}, and so also such a linear combination. Third, $t_{p,2} = t_{q,2} + t_{r,2}$ is the sum of two linear combinations of products $t_{q_i,1}t_{r_i,1}$, and so also such a linear combination. The instruction $t_p \leftarrow t_q - t_r$ is handled similarly.

The most interesting case is $t_p \leftarrow t_q t_r$. First, $t_{p,0} = t_{q,0}t_{r,0}$ is the product of two constants, and so also a constant. Second, $t_{p,1} = t_{q,0}t_{r,1} + t_{q,1}t_{r,0}$. Since $t_{r,1}$ is a linear combination of x_{ij}, y_{ij} and $t_{q,0}$ is constant, the product $t_{q,0}t_{r,1}$ is also such a linear combination. The same can be said about the other term, and so $t_{p,1}$ is a linear combination of x_{ij}, y_{ij}.

Finally, $t_{p,2} = t_{q,0}t_{r,2} + t_{q,2}t_{r,0} + t_{q,1}t_{r,1}$. The first two terms are linear combinations of products $t_{q_i,1}t_{q_i,1}$ multiplied by a constant, and so also such linear combinations. The final terms is itself a product $t_{q_i,1}t_{r_i,1}$ (for i such that $p = p_i$), and so $t_{p,2}$ is a linear combination of products $t_{q_i,1}t_{r_i,1}$. □

This allows us to rewrite Π' in a new equivalent form.

Theorem 4. *In the notation of Theorem 3, consider the following program* Π''*:*

1. *For* $1 \le s \le m$*, compute* $t_{q_s,1} \leftarrow \sum_{ij} \alpha_{q_s ij} x_{ij} + \sum_{jk} \beta_{q_s jk} y_{jk}$ *and* $t_{r_s,1} \leftarrow \sum_{ij} \alpha_{r_s ij} x_{ij} + \sum_{jk} \beta_{r_s jk} y_{jk}$*, the linear combinations promised by Theorem 3.*

2. *For* $1 \le s \le m$*, compute* $m_s \leftarrow t_{q_s,1}t_{r_s,1}$*.*

3. *For each* $1 \le i,k \le n$*, let* $t_{p_{ik}}$ *be the instruction computing* $\sum_{j=1}^{n} x_{ij}y_{jk}$*. For each* i,k*, compute* $t_{p_{ik},2} \leftarrow \sum_s \gamma_{sik} m_s$*, the linear combination promised by Theorem 3.*

Then the program Π'' *computes the product of two* $n \times n$ *matrices.*

Proof. The new program is valid since $t_{p_{ik},2} = t_{p_{ik}} = \sum_{j=1}^{n} x_{ij}y_{jk}$ for each i,k. □

This new program is very similar to Strassen's algorithm as described above. The only difference is that in Strassen's algorithm, each m_s was a product of a linear combination of the x_{ij} and a linear combination of the y_{ij}, whereas here $t_{q_s,1}$ and $t_{r_s,1}$ can mix both. However, this is not difficult to fix.

5.2 Decomposition into xy parts

Let P be a homogeneous polynomial in the x_{ij} and y_{ij}. Just as we had earlier partitioned P according to the total degree, we can partition P according to the decomposition of each monomial — how many variables of type x_{ij} and how many of type y_{ij} are involved. We denote the part which contains monomials which are products of a variables of type x_{ij} and b variables of type y_{ij} by $P^{=x^a y^b}$. For example, if $P = x_{12} + 3y_{23} - y_{14}$ then

$$P^{=x} = x_{12},\ P^{=y} = 3y_{23} - y_{14}.$$

Similarly, if $P = x_{12}^2 + x_{13}y_{15} - y_{25}y_{17}$ then

$$P^{=xx} = x_{12}^2,\ P^{=xy} = x_{13}y_{15},\ P^{=yy} = -y_{25}y_{17}.$$

Earlier we observed that $z_{ik} = z_{ik}^{=2}$. We can now say even more: $z_{ik} = z_{ik}^{=xy}$. This suggests a way of massaging Π'''.

Theorem 5. *In the notation of Theorem 4, consider the following program* Π'''*:*

1. *Compute* $t_{q_s,1,x} \leftarrow \sum_{ij} \alpha_{q_s ij} x_{ij}$ *and* $t_{r_s,1,x} \leftarrow \sum_{ij} \alpha_{r_s ij} x_{ij}$ *for each* s.
2. *Compute* $t_{q_s,1,y} \leftarrow \sum_{jk} \beta_{q_s jk} y_{jk}$ *and* $t_{r_s,1,y} \leftarrow \sum_{jk} \beta_{r_s jk} y_{jk}$ *for each* s.
3. *Compute* $m_{s,xy} \leftarrow t_{q_s,1,x} t_{r_s,1,y} + t_{r_s,1,x} t_{q_s,1,y}$ *for each* s.
4. *For each* $1 \le i,k \le n$, *compute* $t_{p_{ik},2,xy} \leftarrow \sum_s \gamma_{sik} m_s$.

Then the program Π''' *computes the product of two* $n \times n$ *matrices.*

Proof. Note that $t_{q_s,1,x} = t^{=x}_{q_s,1}$ and $t_{q_s,1,y} = t^{=y}_{q_s,1}$, and similarly for r_s. Thus $t_{q_s,1} = t_{q_s,1,x} + t_{q_s,1,y}$ (which is also easy to see directly). It is then not hard to check that $m_{s,xy} = m_s^{=xy}$, and so that $t_{p_{ik},2,xy} = t^{=xy}_{p,2}$ for each i,k. Since $t_{p_{ik},2} = \sum_j x_{ij} y_{jk} = t^{=xy}_{p_{ik},2}$, we see that Π''' is indeed valid. □

The final stroke is to observe that Theorem 5 gives a rank $2m$ decomposition of $\langle n,n,n \rangle$.

Theorem 6. *Let* Π *be a program that computes the product of two* $n \times n$ *matrices, without division, and contains* m *multiplication steps. Then* $R(\langle n,n,n \rangle) \le 2m$.

Proof. The validity of Π'''' is expressed by the following identity, for all $1 \le i,k \le n$:

$$\sum_{j=1}^{n} x_{ij} y_{jk} = t_{p_{ik},2,xy} = \sum_{s=1}^{m} \gamma_{sik} \left(\sum_{ij} \alpha_{q_s ij} x_{ij} \right) \left(\sum_{jk} \beta_{r_s jk} y_{jk} \right) + \gamma_{sik} \left(\sum_{ij} \alpha_{r_s ij} x_{ij} \right) \left(\sum_{jk} \beta_{q_s jk} y_{jk} \right).$$

Multiplying this by z_{ik} and summing over all i,k, we obtain

$$\langle n,n,n \rangle = \sum_{s=1}^{m} \left(\sum_{ij} \alpha_{q_s ij} x_{ij} \right) \left(\sum_{jk} \beta_{r_s jk} y_{jk} \right) \left(\sum_{ik} \gamma_{sik} z_{ik} \right) + \sum_{s=1}^{m} \left(\sum_{ij} \alpha_{r_s ij} x_{ij} \right) \left(\sum_{jk} \beta_{q_s jk} y_{jk} \right) \left(\sum_{ik} \gamma_{sik} z_{ik} \right),$$

which shows that $R(\langle n,n,n \rangle) \le 2m$. □

As a corollary, we can deduce that $c \le \omega$, thus completing the proof that $c = \omega$. Indeed, for every $\epsilon > 0$ there are constants n_0, C such that for $n \ge n_0$, there exists a program Π_n that multiplies two $n \times n$ matrices using at most $Cn^{\omega+\epsilon}$ operations. Theorem 6 shows that $R(\langle n,n,n \rangle) \le 2Cn^{\omega+\epsilon} = O(n^{\omega+\epsilon})$, and so $c \le \omega + \epsilon$. Since this holds for all $\epsilon > 0$, we conclude that $c \le \omega$.

5.3 Note on uniformity

Earlier we discussed the issue of uniformity. The exponent ω is defined so that for every $\epsilon > 0$ and for large enough n, there is an arithmetic program that multiplies two $n \times n$ matrices using $O(n^{\omega+\epsilon})$ arithmetic operations. Can we obtain such a uniform program?

Indeed we can! For every $\epsilon > 0$, there are arithmetic programs that multiply two $n \times n$ matrices using $O(n^{\omega+\epsilon/2})$ operations. For large enough n, this will be at most $n^{\omega+\epsilon}/2$ operations. Theorem 6 shows that $R(\langle n,n,n \rangle) \le n^{\omega+\epsilon}$ for large enough n. Fix such n, and use Theorem 1 to deduce a uniform algorithm which uses $O(n^{\omega+\epsilon})$ arithmetic operations.

From a theoretical perspective, if we know ω, then for each $\epsilon > 0$ we can find n large enough such that $R(\langle n,n,n \rangle) \le n^{\omega+\epsilon}$, together with such a decomposition, in constant time. Therefore the uniform complexity of matrix multiplication (say in the word RAM model) is $O(n^{\omega+\epsilon} M)$ for all $\epsilon > 0$, where M is the cost of multiplying two matrix entries.

6 Operations on tensors

Our next stop will require us to generalize some classical operations on matrices: transposition, Kronecker product (a generalization of the outer product), and direct sum (creating a block diagonal matrix).

6.1 Shuffle

Tensor shuffle (this is a non-standard name!) is the operation that corresponds to transposition of matrices. As usual, we start with the case of matrices. Let A be an $n \times m$ matrix whose corresponding bilinear form is

$$\sum_{i=1}^{n}\sum_{j=1}^{m} x_i y_j a_{ij}.$$

The bilinear form corresponding to the transposed matrix A^T is

$$\sum_{j=1}^{m}\sum_{i=1}^{n} x_j y_i a_{ij}.$$

This form is obtained from the preceding one simply by switching the xs and the ys, which corresponds exactly to switching rows and columns.

There are several corresponding operations on 3D tensors. In fact, any of the 5 different ways of permuting the three dimensions is a kind of "shuffle". The most important for us will be *rotation*, in which rows become depths, columns become rows, and depths become rows. We will denote this operation in the same way as matrix transpose (this is non-standard notation). That is, given an $a \times b \times c$ tensor

$$W = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} x_i y_j z_k w_{ijk},$$

the rotated tensor W^T is the $b \times c \times a$ tensor

$$W^T = \sum_{j=1}^{b}\sum_{k=1}^{c}\sum_{i=1}^{a} x_j y_k z_i w_{ijk}.$$

It is not difficult to see that rotation (in fact, any shuffle) preserves tensor rank, since the rotation of a rank one tensor is another rank one tensor:

$$\left(\sum_i \alpha_i x_i\right)\left(\sum_j \beta_j y_j\right)\left(\sum_k \gamma_k z_k\right)^T = \left(\sum_j \beta_j x_j\right)\left(\sum_k \gamma_k y_k\right)\left(\sum_i \alpha_i z_i\right).$$

Particularly interesting is the effect of rotation on matrix multiplication tensors:

$$\begin{aligned}\langle n,m,p\rangle^T &= \left(\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p} x_{ij}y_{jk}z_{ik}\right)^T \\ &= \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p} x_{jk}y_{ik}z_{ij} \\ &\approx \langle m,p,n\rangle.\end{aligned}$$

Here $\approx$ means that the two tensors are the same up to renaming of rows, columns, and depths. All we have to do is to switch the indices of the y variables, that is, replace y_{ik} with y_{ki}. (We could avoid this mishap by using $\sum_{ijk} x_{ij}y_{jk}z_{ki}$ instead of $\sum_{ijk} x_{ij}y_{jk}z_{ik}$.)

Renaming doesn't affect the rank, and we conclude that $R(\langle n,m,p\rangle) = R(\langle m,p,n\rangle)$. Another rotation shows that $R(\langle n,m,p\rangle) = R(\langle p,n,m\rangle)$.

If instead of rotating we transpose the rows and the columns, then we get

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p} x_{jk}y_{ij}z_{ik} \approx \langle p,m,n\rangle.$$

This time we have to switch the indices of both the x and the y variables. As before, we get that $R(\langle n,m,p\rangle) = R(\langle p,m,n\rangle)$. Continuing in this way, we see that the rank of $\langle n,m,p\rangle$ doesn't depend on the order of the three dimensions n,m,p.

Let us pause to consider an example, the tensor $\langle 1,n,1\rangle$:

$$\langle 1,n,1\rangle = \sum_{j=1}^{n} x_{1j}y_{j1}z_{11}.$$

This tensor describes the task of computing the inner product of two vectors of length n. In contrast, its rotations are

$$\langle n,1,1\rangle = \sum_{i=1}^{n} x_{i1}y_{11}z_{1i},$$
$$\langle 1,1,n\rangle = \sum_{k=1}^{n} x_{11}y_{1k}z_{1k},$$

and they describe the task of multiplying a vector of length n by a scalar. Surprisingly, these three different tasks have the same difficulty, as measured by rank. (In fact, it is not too difficult to show that $R(\langle 1,n,1\rangle) = n$.)

6.2 Tensor product

Our next operation is somewhat less standard, and known in the matrix world as *Kronecker product*. However, in some sense we have already seen it, since it corresponds to the divide-and-conquer approach.

We start by describing this operation for matrices. Let A be an $n_A \times m_A$ matrix, and let B be an $n_B \times m_B$ matrix. Their Kronecker product $A \otimes B$ is an $n_A n_B \times m_A m_B$ matrix whose rows and columns are indexed by pairs of indices, and whose entries are given by the formula

$$(A \otimes B)_{(i_A,i_B),(j_A,j_B)} = a_{i_A,j_A} b_{i_B,j_B}.$$

This formula makes it clear that Kronecker product distributes over addition: $A \otimes (B+C) = A \otimes B + A \otimes C$. Notice also that if A is a column vector and B is a row vector, then $A \otimes B$ is just their product, given by $(A \otimes B)_{ij} = (AB)_{ij} = A_i B_j$. In this case the tensor product is known as an *outer product*.

As an example of the Kronecker product, if we index the rows and columns in the order $(1,1),(1,2),(2,1),(2,2)$, then

$$\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 \cdot \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} & -1 \cdot \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \\ 1 \cdot \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} & -1 \cdot \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}.$$

(These matrices are known as *Hadamard matrices*, and describe the *Walsh transform*, which is the Fourier transform in the group $\mathbb{Z}_2^n$.)

In terms of bilinear forms, the bilinear form corresponding to $A \otimes B$ is

$$\sum_{i_A=1}^{n_A} \sum_{i_B=1}^{n_B} \sum_{j_A=1}^{m_A} \sum_{j_B=1}^{m_B} x_{i_A i_B} y_{j_A j_B} a_{i_A j_A} b_{i_B j_B}.$$

We can obtain this expression by multiplying the bilinear forms corresponding to A and B, and replacing $x_{i_A} x_{i_B}$ with $x_{i_A i_B}$ and $y_{j_A} y_{j_B}$ with $y_{j_A j_B}$.

We can do the same with 3D tensors. If P is an $a_P \times b_P \times c_P$ tensor and Q is an $a_Q \times b_Q \times c_Q$ tensor then $T = P \otimes Q$ is an $a_P a_Q \times b_P b_Q \times c_P c_Q$ tensor whose entries are $T(i_P i_Q, j_P j_Q, k_P k_Q) = P(i_P, j_P, k_P) Q(i_Q, j_Q, k_Q)$.

As in the case of matrices, we can obtain the trilinear form of $P \otimes Q$ from those of P and Q by replacing $x_{i_P} x_{i_Q}$ by $x_{i_P i_Q}$, and similarly for the y and z variables.

We will be particularly interested in the tensor product of two matrix multiplication tensors. Using the recipe above, we find that

$$\begin{aligned}
\langle n_1, m_1, p_i \rangle \otimes \langle n_2, m_2, p_2 \rangle &= \left(\sum_{i_1=1}^{n_1} \sum_{j_1=1}^{m_1} \sum_{k_1=1}^{p_1} x_{i_1 j_1} y_{j_1 k_1} z_{i_1 k_1} \right) \otimes \left(\sum_{i_2=1}^{n_2} \sum_{j_2=1}^{m_2} \sum_{k_2=1}^{p_2} x_{i_2 j_2} y_{j_2 k_2} z_{i_2 k_2} \right) \\
&= \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \sum_{j_1=1}^{m_1} \sum_{j_2=1}^{m_2} \sum_{k_1=1}^{p_1} \sum_{k_2=1}^{p_2} x_{(i_1,i_2),(j_1,j_2)} y_{(j_1,j_2),(k_1,k_2)} z_{(i_1,i_2),(k_1,k_2)} \\
&\approx \sum_{i=1}^{n_1 n_2} \sum_{j=1}^{m_1 m_2} \sum_{k=1}^{p_1 p_2} x_{i,j} y_{j,k} z_{i,k} \\
&= \langle n_1 n_2, m_1 m_2, p_1 p_2 \rangle.
\end{aligned}$$

In the third step we renamed (i_1, i_2) to $n_2(i_1-1)+i_2$, (j_1, j_2) to $m_2(j_1-1)+j_2$, and (k_1, k_2) to $p_2(k_1-1)+k_2$, and have surprisingly obtained a single matrix multiplication tensor. This should not be too surprising, however, since this is exactly how our recursive divide-and-conquer approach works: when we think of an $n \times n$ matrix as a 2×2 matrix with $(n/2) \times (n/2)$ blocks and use this to multiply $n \times n$ matrices, we are implicitly using the formula $\langle n, n, n \rangle = \langle 2, 2, 2 \rangle \otimes \langle n/2, n/2, n/2 \rangle$.

In the future, we will sometimes use the useful notation $T^{\otimes n}$ for the nth tensor power of T, which is obtained by computing the tensor product of n copies of T.

What is the influence of tensor product on rank? Consider again two tensors P, Q, and their rank decompositions

$$\begin{aligned}
P &= \sum_{s_P=1}^{R(P)} \left(\sum_{i_P} \alpha^P_{s_P i_P} x_{i_P} \right) \left(\sum_{j_P} \beta^P_{s_P j_P} y_{j_P} \right) \left(\sum_{k_P} \gamma^P_{s_P k_P} z_{k_P} \right), \\
Q &= \sum_{s_Q=1}^{R(Q)} \left(\sum_{i_Q} \alpha^Q_{s_Q i_Q} x_{i_Q} \right) \left(\sum_{j_Q} \beta^Q_{s_Q j_Q} y_{j_Q} \right) \left(\sum_{k_Q} \gamma^Q_{s_Q k_Q} z_{k_Q} \right).
\end{aligned}$$

Since tensor product distributes over addition (just like the Kronecker product of matrices),

$$P \otimes Q = \sum_{s_P=1}^{R(P)} \sum_{s_Q=1}^{R(Q)} \left(\sum_{i_P, i_Q} \alpha^P_{s_P i_P} \alpha^Q_{s_Q i_Q} x_{i_P i_Q} \right) \left(\sum_{j_P, j_Q} \beta^P_{s_P j_P} \beta^Q_{s_Q j_Q} y_{j_P j_Q} \right) \left(\sum_{k_P, k_Q} \gamma^P_{s_P k_P} \gamma^Q_{s_Q k_Q} z_{k_P k_Q} \right).$$

This shows that $R(P \otimes Q) \leq R(P) R(Q)$.

Combining rotation with tensor product, we obtain an important corollary, that allows us to obtain bounds on ω from bounds on the rank of $\langle n, m, p \rangle$ for *any* n, m, p.

Theorem 7. *If $R(\langle n, m, p \rangle) \leq r$ then $\omega \leq \log_{\sqrt[3]{nmp}} r$. In other words,*

$$(nmp)^{\omega/3} \leq r.$$

Proof. Observe that

$$\begin{aligned}
R(\langle nmp, nmp, nmp \rangle) = R(\langle n, m, p \rangle \otimes \langle m, p, n \rangle \otimes \langle p, n, m \rangle) \leq R(\langle n, m, p \rangle) R(\langle m, p, n \rangle) R(\langle p, n, m \rangle) = \\
R(\langle n, m, p \rangle)^3 = r^3.
\end{aligned}$$

Theorem 1 therefore implies that

$$\omega \leq \log_{nmp}(r^3) = \log_{\sqrt[3]{(nmp)}} r.$$

□

6.3 ω as a limit

Another important corollary shows that ω can be obtained as a limit.

Theorem 8. *The limit* $\lim_{n\to\infty} \log_n R(\langle n,n,n\rangle)$ *exists, and is equal to* ω.

Proof. Let $c_k = \log_{2^k} R(\langle 2^k, 2^k, 2^k\rangle) = \frac{1}{k}\log_2 R(\langle 2^k, 2^k, 2^k\rangle)$. Then

$$c_{a+b} = \frac{1}{a+b}\log_2 R(\langle 2^{a+b}, 2^{a+b}, 2^{a+b}\rangle) \leq \frac{1}{a+b}\log_2[R(\langle 2^a, 2^a, 2^a\rangle)R(\langle 2^b, 2^b, 2^b\rangle)] = \frac{a}{a+b}c_a + \frac{b}{a+b}c_b.$$

In particular, choosing $a = b$ we see that $c_{2a} \leq c_a$. The sequence c_{2^t} is monotone decreasing and bounded from below (since $c_k \geq 0$ for all k), and so approaches a limit c^*.

Our next step is showing that $\lim_{n\to\infty} c_n = c^*$. First note that the inequality $c_{a+b} \leq \frac{a}{a+b}c_a + \frac{b}{a+b}c_b$ easily generalizes to larger sums: if $a = \sum_{i=1}^m a_i$ then

$$c_a \leq \sum_{i=1}^m \frac{a_i}{a} c_{a_i}.$$

Now let k be an arbitrary index. Every $n \geq k$ can be written as the sum of $m = \lfloor n/k \rfloor$ copies of k and $a_{m+1} = n \bmod k$, and so

$$c_n \leq \sum_{i=1}^m \frac{k}{n}c_k + \frac{r}{n}c_r = \left(1 - \frac{r}{n}\right)c_k + \frac{r}{n}c_r \leq c_k + \frac{k}{n}\max_{0\leq s<k} c_s.$$

Let $\epsilon > 0$ be given. For $n \geq \max_{0\leq s<k} c_s k/\epsilon$, the second term is bounded by ϵ, and so for large enough n, $c_n \leq c_k + \epsilon$. For every $\epsilon > 0$ there exists t such that $c_{2^t} \leq c^* + \epsilon$, and so for large enough n, $c_n \leq c^* + 2\epsilon$. We conclude that $\limsup_{n\to\infty} c_n \leq c^*$. On the other hand, if $c_k < c^*$ for any $k > 0$ then $\limsup_{n\to\infty} c_n \leq c_k < c^*$, contradicting the fact that $\lim_{t\to\infty} c_{2^t} = c^*$. We conclude that $c^* = \lim_{k\to\infty} c_k$.

The final step is handling $\log_n R(\langle n,n,n\rangle)$ for arbitrary n. For that we will need the observation that $R(\langle n,n,n\rangle)$ is non-decreasing in n. Indeed, consider $n_1 < n_2$, and let $r = R(\langle n_2, n_2, n_2\rangle)$. Then there is a decomposition

$$\sum_{i=1}^{n_2}\sum_{j=1}^{n_2}\sum_{k=1}^{n_2} x_{ij}y_{jk}z_{ik} = \sum_{s=1}^{r}\left(\sum_{i=1}^{n_2}\sum_{j=1}^{n_2}\alpha_{sij}x_{ij}\right)\left(\sum_{j=1}^{n_2}\sum_{k=1}^{n_2}\beta_{sjk}y_{jk}\right)\left(\sum_{i=1}^{n_2}\sum_{k=1}^{n_2}\gamma_{sik}z_{ik}\right).$$

Take this identity and substitute zero for x_{ij}, y_{jk}, z_{ik} whenever $i > n_1$, $j > n_1$, or $k > n_1$, to obtain

$$\sum_{i=1}^{n_1}\sum_{j=1}^{n_1}\sum_{k=1}^{n_1} x_{ij}y_{jk}z_{ik} = \sum_{s=1}^{r}\left(\sum_{i=1}^{n_1}\sum_{j=1}^{n_1}\alpha_{sij}x_{ij}\right)\left(\sum_{j=1}^{n_1}\sum_{k=1}^{n_1}\beta_{sjk}y_{jk}\right)\left(\sum_{i=1}^{n_1}\sum_{k=1}^{n_1}\gamma_{sik}z_{ik}\right).$$

The left-hand side is just $\langle n_1, n_1, n_1\rangle$, and so this gives a rank r decomposition of $\langle n_1, n_1, n_1\rangle$, showing that $R(\langle n_1, n_1, n_1\rangle) \leq r = R(\langle n_2, n_2, n_2\rangle)$.

Every number n can be written as $2^k\alpha$ for $1 \leq \alpha < 2$. Monotonicity of $R(\langle n,n,n\rangle)$ implies that

$$R(\langle 2^k, 2^k, 2^k\rangle) \leq R(\langle n,n,n\rangle) \leq R(\langle 2^{k+1}, 2^{k+1}, 2^{k+1}\rangle).$$

Taking the logarithm with respect to n, we obtain

$$c_k \log_n(2^k) \leq \log_n R(\langle n,n,n\rangle) \leq c_{k+1}\log_n(2^{k+1}).$$

Substituting $n = 2^k\alpha$, we get $\log_n(2^k) = \log_n(n/\alpha) = 1 - \log_n\alpha$, and similarly $\log_n(2^{k+1}) = \log_n[(2/\alpha)n] = 1 + \log_n(2/\alpha)$, and so $1 \leq \alpha \leq 2$ implies the bounds

$$\left(1 - \frac{\log 2}{\log n}\right)c_k \leq \log_n R(\langle n,n,n\rangle) \leq \left(1 + \frac{\log 2}{\log n}\right)c_{k+1}.$$

As $n \to \infty$, both sides of the inequality tend to c^*, and we conclude that $\log_n R(\langle n,n,n\rangle) \longrightarrow \omega$. □

6.4 Direct sum

The final operation we consider is the direct sum. Given an $n_A \times m_A$ matrix A and an $n_B \times m_B$ matrix B, their direct sum $A \oplus B$ is the $(n_A + n_B) \times (m_A + m_B)$ matrix whose matrix form is

$$\begin{pmatrix} A & 0_{n_A \times m_B} \\ 0_{n_B \times m_A} & B \end{pmatrix}.$$

The corresponding bilinear form is

$$\sum_{i_A=1}^{n_A} \sum_{j_A=1}^{m_A} x_{i_A} y_{j_A} a_{i_A j_A} + \sum_{i_B=1}^{n_B} \sum_{j_B=1}^{m_B} x_{n_A+i_B} y_{m_A+j_B} b_{i_B j_B}.$$

Notice that up to renaming of the xs and ys, this is just the sum of the bilinear forms corresponding to A and B.

There is also a simple interpretation in terms of vector spaces. If we think of A as a linear transformation from V_A to W_A and of B as a linear transformation from V_B to W_B, then $A \oplus B$ is the linear transformation from $V_A \oplus V_B$ to $W_A \oplus W_B$ given by $(A \oplus B)(v_A, v_B) = (Av_A, Bv_B)$.

The direct sum of two tensors P, Q is defined analogously. Suppose that P is $a_P \times b_P \times c_P$ and that Q is $a_Q \times b_Q \times c_Q$. Their direct sum $P \oplus Q$ is the $(a_P + a_Q) \times (b_P + b_Q) \times (c_P + c_Q)$ tensor given by

$$\sum_{i_P=1}^{a_P} \sum_{j_P=1}^{b_P} \sum_{k_P=1}^{c_P} x_{i_P} y_{j_P} z_{k_P} p_{i_P j_P k_P} + \sum_{i_Q=1}^{a_Q} \sum_{j_Q=1}^{b_Q} \sum_{k_Q=1}^{c_Q} x_{a_P+i_Q} y_{b_P+j_Q} z_{c_P+k_Q} q_{i_Q j_Q k_Q}.$$

Again, up to renaming this is just the sum of the trilinear forms corresponding to P and Q. For this reason, if we sum a decomposition of P into the sum of $R(P)$ rank one tensors and a decomposition of Q into the sum of $R(Q)$ rank one tensors (suitably renamed), then we get a decomposition of $P \oplus Q$ into the sum of $R(P) + R(Q)$ rank one tensors. In other words, $R(P \oplus Q) \leq R(P) + R(Q)$. (Try to think whether equality holds — it certainly does for matrices.)

Direct sums figure prominently in the asymptotic sum inequality.

7 Border rank

Suppose that A_t is a sequence of $n \times m$ matrices of rank r which converge[4] to a matrix A. Then I claim that A also has rank at most r. (Certainly A can have smaller rank. For example, if I is the identity matrix then $I/t \longrightarrow 0$.) Indeed, suppose for simplicity that $n = m$ (otherwise add zero rows or columns, as needed). The co-rank of an $n \times n$ matrix M, which is $n - R(M)$, is the number of eigenvalues of M which are equal to zero. If the eigenvalues of M (with multiplicity) are $\lambda_1, \ldots, \lambda_n$ then the characteristic polynomial of M is equal to $C(M) = \det(xI - M) = \prod_{i=1}^{n}(x - \lambda_i)$. Thus $C(M)$ is a multiple of $x^{n-R(M)}$, and so the $n - R(M)$ least significant coefficients of $C(M)$ vanish (these are the coefficients of $x^0, \ldots, x^{n-R(M)-1}$). Back to our sequence, this shows that the $n - r$ least significant coefficients of $C(A_t)$ vanish. Since the characteristic polynomial is a continuous function of the entries, $C(A_t) \longrightarrow C(A)$, and so the $n - r$ least significant coefficients of $C(A)$ vanish, showing that the co-rank of A is at least $n - r$, and so its rank is at most R.

Surprisingly, the corresponding property fails for 3D tensors. Consider the tensor

$$T = x_1 y_1 z_2 + x_1 y_2 z_1 + x_2 y_1 z_1,$$

and the following sequence of tensors converging to T entrywise:

$$T_n = x_1 y_1 z_2 + x_1 y_2 z_1 + x_2 y_1 z_1 + \frac{1}{n}(x_1 y_2 z_2 + x_2 y_1 z_2 + x_2 y_2 z_1) + \frac{1}{n^2} x_2 y_2 z_2.$$

[4]The notion of convergence here is entrywise convergence. The same notion of convergence is obtained if we use convergence in Frobenius norm ($\|M\| = \sqrt{\mathrm{Tr}(M^2)}$), or indeed all other common matrix norms.

Clearly $R(T) \leq 3$. In fact, $R(T) = 3$. To see that, suppose that

$$x_1y_1z_2 + x_1y_2z_1 + x_2y_1z_1 = \sum_{i=1}^{2}(\alpha_{i1}x_1 + \alpha_{i2}x_2)(\beta_{i1}y_1 + \beta_{i2}y_2)(\gamma_{i1}z_1 + \gamma_{i2}z_2).$$

Since z_2 appears in the left-hand side, it must also appear in the right-hand side. In other words, $\gamma_{i2} \neq 0$ for some i. We can assume that $\gamma_{22} \neq 0$. Substitute $z_2 = -(\gamma_{21}/\gamma_{22})z_1$, which zeroes out the factor $(\gamma_{21}z_1 + \gamma_{22}z_2)$, to get the equation

$$-\frac{\gamma_{21}}{\gamma_{22}}x_1y_1z_1 + x_1y_2z_1 + x_2y_1z_1 = (\alpha_{11}x_1 + \alpha_{12}x_2)(\beta_{11}y_1 + \beta_{12}y_2)((\gamma_{11} - \tfrac{\gamma_{12}\gamma_{21}}{\gamma_{22}})z_1).$$

Since y_2 appears in the left-hand side, necessarily $\beta_{12} \neq 0$. If we substitute $y_2 = -(\beta_{11}/\beta_{12})y_1$ then the right-hand side zeroes out, whereas the left-hand side doesn't (the coefficient of $x_2y_1z_1$ is still 1), and we reach a contradiction.

We have seen that $R(T) = 3$. In contrast, $R(T_n) = 2$, since we can write

$$T_n = n\left(x_1 + \frac{x_2}{n}\right)\left(y_1 + \frac{y_2}{n}\right)\left(z_1 + \frac{z_2}{n}\right) - nx_1y_1z_1.$$

(One can show that $R(T_n) > 1$ along the same lines as above.) The reason that the argument for matrices doesn't work is apparent — the coefficients here do not converge.

If we set $\epsilon = 1/n$, then we have the exact identity

$$(x_1+\epsilon x_2)(y_1+\epsilon y_2)(z_1+\epsilon z_2) - x_1y_1z_1 = \epsilon(x_1y_1z_2+x_1y_2z_1+x_2y_1z_1) + \epsilon^2(x_1y_2z_2+x_2y_1z_2+x_2y_2z_1) + \epsilon^3 x_2y_2z_2.$$

We can write the same identity in the following approximate form:

$$(x_1 + \epsilon x_2)(y_1 + \epsilon y_2)(z_1 + \epsilon z_2) - x_1y_1z_1 = \epsilon T + O(\epsilon^2).$$

Here $O(\epsilon^2)$ hides terms which are products of ϵ^2 and higher powers of ϵ. If we think of ϵ as tending to zero, then this is just the usual big O notation.

The *border rank* of a tensor T, denoted $\underline{R}(T)$, is the minimal rank r such that there is a sequence of tensors T_n of rank at most r converging to T. For example, if T is the tensor considered above, then $R(T) = 3$ whereas $\underline{R}(T) = 2$ (one can show that $\underline{R}(T) > 1$). It is a highly non-trivial fact[5] that if $\underline{R}(T) \leq r$ then this can be witnessed by a decomposition of the form

$$\sum_{s=1}^{r}\left(\sum_i \alpha_{si}x_i\right)\left(\sum_j \beta_{sj}y_j\right)\left(\sum_k \gamma_{sk}z_k\right) = \epsilon^d T + O(\epsilon^{d+1})$$

for some integer $d \geq 0$, where the coefficients $\alpha_{si}, \beta_{sj}, \gamma_{sk}$ are *polynomials in* ϵ. We gave such a decomposition above for our example tensor T.

In the original paper of Bini, Capovani, Lotti, and Romani which introduced border rank, the idea was that by choosing some small $\epsilon > 0$, one can calculate the matrix multiplication approximately. However, it is also possible to eliminate the approximate aspect, and prove the analog of Theorem 7, with border rank replacing rank. We start with the case $n = m = p$.

Theorem 9. *If $\underline{R}(\langle n, n, n\rangle) \leq r$ then $\omega \leq \log_n r$. In other words,*

$$n^\omega \leq \underline{R}(\langle n, n, n\rangle).$$

[5]For the curious, this holds since the Zariski closure of the space of rank r tensors is the same as its closure under the usual topology. Apparently this is elementary algebraic geometry, but unfortunately such arguments are out of scope for the author.

Proof. Our starting point is the rank r decomposition

$$\sum_{s=1}^{r}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_{sij}x_{ij}\right)\left(\sum_{j=1}^{n}\sum_{k=1}^{n}\beta_{sjk}y_{jk}\right)\left(\sum_{i=1}^{n}\sum_{k=1}^{n}\gamma_{sik}z_{ik}\right)=\epsilon^{d}\langle n,n,n\rangle+O(\epsilon^{d+1}).$$

Raising both sides to the tth tensor power, on the left-hand side we have the sum of r^t rank one tensors (with ϵ), and on the right-hand side we have $\epsilon^{td}\langle n^t,n^t,n^t\rangle+O(\epsilon^{td+1})$:

$$\sum_{s=1}^{r^t}\left(\sum_{i=1}^{n^t}\sum_{j=1}^{n^t}\hat{\alpha}_{sij}x_{ij}\right)\left(\sum_{j=1}^{n^t}\sum_{k=1}^{n^t}\hat{\beta}_{sjk}y_{jk}\right)\left(\sum_{i=1}^{n^t}\sum_{k=1}^{n^t}\hat{\gamma}_{sik}z_{ik}\right)=\epsilon^{td}\langle n^t,n^t,n^t\rangle+O(\epsilon^{td+1}).$$

Each $\hat{\alpha}_{sij},\hat{\beta}_{sjk},\hat{\gamma}_{sik}$ is a polynomial in ϵ, say $\hat{\alpha}_{sij}=\sum_{e=0}^{E}\hat{\alpha}_{sije}\epsilon^{e}$, where $\hat{\alpha}_{sije}\in\mathbb{R}$, and E is an upper bound on the degree of all these polynomials. Substituting this, we get

$$\sum_{s=1}^{r^t}\left(\sum_{i=1}^{n^t}\sum_{j=1}^{n^t}\sum_{e_x=0}^{E}\hat{\alpha}_{sije_x}\epsilon^{e_x}x_{ij}\right)\left(\sum_{j=1}^{n^t}\sum_{k=1}^{n^t}\sum_{e_y=0}^{E}\hat{\beta}_{sjke_y}\epsilon^{e_y}y_{jk}\right)\left(\sum_{i=1}^{n^t}\sum_{k=1}^{n^t}\sum_{e_z=0}^{E}\hat{\gamma}_{ske_z}\epsilon^{e_z}z_{ik}\right)=\epsilon^{td}\langle n^t,n^t,n^t\rangle+O(\epsilon^{td+1}).$$

Comparing coefficients of ϵ^{td}, we see that

$$\langle n^t,n^t,n^t\rangle=\sum_{s=1}^{r^t}\sum_{\substack{e_x,e_y,e_z:\\ e_x+e_y+e_z=td}}\left(\sum_{i=1}^{n^t}\sum_{j=1}^{n^t}\hat{\alpha}_{sije_x}x_{ij}\right)\left(\sum_{j=1}^{n^t}\sum_{k=1}^{n_t}\hat{\beta}_{sjke_y}y_{jk}\right)\left(\sum_{i=1}^{n^t}\sum_{k=1}^{n^t}\hat{\gamma}_{ske_z}z_{ik}\right).$$

This shows that

$$R(\langle n^t,n^t,n^t\rangle)\le r^t|\{0\le e_x,e_y,e_z\le E:e_x+e_y+e_z=td\}|=\binom{td+2}{2}r^t\le Ct^2r^t,$$

for some constant $C>0$ depending only on d. Applying Theorem 1, we see that

$$\omega\le\log_{n^t}R(\langle n^t,n^t,n^t\rangle)\le\log_{n^t}r^t+\frac{\log Ct^2}{\log n^t}=\log_n r+O\left(\frac{\log t}{t}\right).$$

Taking the limit $t\to\infty$, we obtain $\omega\le\log_n r$. □

For the general case, we need to generalize all our work about tensor rotation and product to border rank. Since the proofs are very similar, we won't detail them, but only state the results:

- $\underline{R}(\langle n,m,p\rangle)=\underline{R}(\langle m,p,n\rangle)=\underline{R}(\langle p,n,m\rangle)$.
- $\underline{R}(P\otimes Q)\le\underline{R}(P)\underline{R}(Q)$.
- $\underline{R}(P\oplus Q)\le\underline{R}(P)+\underline{R}(Q)$.

As a result, we can conclude the following analog of Theorem 7.

Theorem 10. *If $\underline{R}(\langle n,m,p\rangle)\le r$ then $\omega\le\log_{\sqrt[3]{nmp}}r$. In other words,*

$$(nmp)^{\omega/3}\le\underline{R}(\langle n,m,p\rangle).$$

The proof is the same as the proof of Theorem 7, with Theorem 1 replaced with Theorem 9.

Bini, Capovani, Romani, and Lotti gave an identity implying that $\underline{R}(\langle 3,2,2\rangle)\le 10$, and as a result concluded that $\omega\le\log_{\sqrt[3]{12}}10\approx 2.78$, improving on Strassen's bound. The identity is not too complicated, but there is no reason to state it here.

Finally, let us mention that the asymptotic growth rate of $\underline{R}(\langle n,n,n\rangle)$ is the same as that of $R(\langle n,n,n\rangle)$.

Theorem 11. *The limit* $\lim_{n\to\infty} \log_n \underline{R}(\langle n,n,n\rangle)$ *exists, and is equal to* ω.

Proof. Since $\underline{R}(\langle n,n,n\rangle) \leq R(\langle n,n,n\rangle)$, Theorem 8 implies that $\limsup_{n\to\infty} \log_n \underline{R}(\langle n,n,n\rangle) \leq \omega$. In contrast, Theorem 9 shows that $\log_n \underline{R}(\langle n,n,n\rangle) \geq \omega$, and so $\lim_{n\to\infty} \log_n \underline{R}(\langle n,n,n\rangle) = \omega$. □

Finally, let us comment that $\underline{R}(\langle 2,2,2\rangle) = 7$, so that Strassen's identity is optimal even for border rank.

8 Asymptotic sum inequality

In this section we extend Theorem 10 to direct sums of tensors. The simplest case is when we have several different copies of the same tensor. We will denote the direct sum of ℓ copies of a tensor T by ℓT. As an example,

$$\ell\langle n,n,n\rangle = \sum_{s=1}^{\ell}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} x_{sij}y_{sjk}z_{sik}.$$

In this expression, we used the arbitrary indexing scheme $x_{sij}, y_{sjk}, z_{sik}$. In general, we will consider two tensors to be the same if they only differ in the names of rows, columns, and depths (earlier we used $\approx$ to express this relation).

Intuitively, if we can perform ℓ independent multiplications of pairs of $n \times n$ matrices using r essential multiplications, then the amortized cost of multiplying a single pair of $n \times n$ matrices is r/ℓ, and so we expect that Theorem 1 be generalized to

$$n^\omega \leq \frac{R(\ell\langle n,n,n\rangle)}{\ell}.$$

The actual proof is somewhat more subtle. The main part is the following claim.

Theorem 12. *Let* $n, N \geq 1$ *be arbitrary sizes. Then*

$$\underline{R}(\langle nN,nN,nN\rangle) \leq \left\lceil \frac{\underline{R}(\langle N,N,N\rangle)}{\ell} \right\rceil \underline{R}(\ell\langle n,n,n\rangle).$$

Proof. To simplify the proof, we first consider rank instead of border rank. Afterwards we will indicate how to generalize the proof to border rank.

Let $r = R(\langle N,N,N\rangle)$. Then $\langle N,N,N\rangle$ can be written as

$$\langle N,N,N\rangle = \sum_{s=1}^{r}\left(\sum_{IJ}\alpha_{sIJ}x_{IJ}\right)\left(\sum_{JK}\beta_{sJK}y_{JK}\right)\left(\sum_{IK}\gamma_{sIK}z_{IK}\right).$$

Let $r' = R(\ell\langle n,n,n\rangle)$. As we have seen above, this allows us to write

$$\sum_{s=1}^{\ell}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} x_{sij}y_{sjk}z_{sik} = \sum_{t=1}^{r'}\left(\sum_{sij}\hat{\alpha}_{tsij}x_{sij}\right)\left(\sum_{sjk}\hat{\beta}_{tsjk}y_{sjk}\right)\left(\sum_{sik}\hat{\gamma}_{tsik}z_{sik}\right). \tag{2}$$

We have seen above that $\langle nN,nN,nN\rangle = \langle N,N,N\rangle \otimes \langle n,n,n\rangle$, and so

$$\langle nN,nN,nN\rangle = \sum_{s=1}^{r}\left(\sum_{IJ}\alpha_{sIJ}x_{IJ}\right)\left(\sum_{JK}\beta_{sJK}y_{JK}\right)\left(\sum_{IK}\gamma_{sIK}z_{IK}\right) \otimes \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} x_{ij}y_{jk}z_{ik}. \tag{3}$$

The idea now is to use (2) to express the first ℓ terms as a sum of r' rank one tensors. We do this by making the following substitutions in (2):

$$x_{sij} = \sum_{IJ}\alpha_{sIJ}x_{IJ} \otimes x_{ij},\ y_{sjk} = \sum_{JK}\beta_{sJK}y_{JK} \otimes y_{jk},\ z_{sik} = \sum_{IK}\gamma_{sIK}z_{IK} \otimes z_{ik}.$$

(Recall that $x_{IJ} \otimes x_{ij}$ is just another name for x_{IiJj}, and so on for the columns and depths.) The left-hand side of (2) now becomes

$$\sum_{s=1}^{\ell} \left(\sum_{IJ} \alpha_{sIJ} x_{IJ} \right) \left(\sum_{JK} \beta_{sJK} y_{JK} \right) \left(\sum_{IK} \gamma_{sIK} z_{IK} \right) \otimes \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} x_{ij} y_{jk} z_{ik},$$

and the right-hand side becomes

$$\sum_{t=1}^{r'} \left(\sum_{sijIJ} \hat{\alpha}_{tsij} \alpha_{sIJ} x_{IJ} \otimes x_{ij} \right) \left(\sum_{sjkJK} \hat{\beta}_{tsjk} \beta_{sJK} y_{JK} \otimes y_{jk} \right) \left(\sum_{sikIK} \hat{\gamma}_{tsik} \gamma_{sIK} z_{IK} \otimes z_{ik} \right).$$

In other words, we have expressed the sum of the first ℓ terms of (3) as a sum of r' rank one tensors. In the same way we can express any other sum of ℓ terms of (3) as a sum of r' rank one tensors. We can also express in the same way the sum of fewer that ℓ terms, by substituting zeroes for the corresponding $\alpha_{sIJ}, \beta_{sJK}, \gamma_{sIK}$. In this way we can partition the terms in (3) into $\lceil r/\ell \rceil$ sets of at most ℓ terms, and so express $\langle nN, nN, nN \rangle$ as the sum of $\lceil r/\ell \rceil r'$ rank one tensors. This completes the proof for rank.

We now briefly indicate the necessary changes in the case of border rank. The expansion of $\langle N, N, N \rangle$ is now

$$\epsilon^D \langle N, N, N \rangle + O(\epsilon^{D+1}) = \sum_{s=1}^{r} \left(\sum_{IJ} \alpha_{sIJ} x_{IJ} \right) \left(\sum_{JK} \beta_{sJK} y_{JK} \right) \left(\sum_{IK} \gamma_{sIK} z_{IK} \right)$$

for some $D \geq 0$, and (2) becomes

$$\epsilon^d \sum_{s=1}^{\ell} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} x_{sij} y_{sjk} z_{sik} + O(\epsilon^{d+1}) = \sum_{t=1}^{r'} \left(\sum_{sij} \hat{\alpha}_{tsij} x_{sij} \right) \left(\sum_{sjk} \hat{\beta}_{tsjk} y_{sjk} \right) \left(\sum_{sik} \hat{\gamma}_{tsik} z_{sik} \right) \tag{4}$$

for some $d \geq 0$. We write (3) as

$$\epsilon^D \langle nN, nN, nN \rangle + O(\epsilon^{D+1}) = \sum_{s=1}^{r} \left(\sum_{IJ} \alpha_{sIJ} x_{IJ} \right) \left(\sum_{JK} \beta_{sJK} y_{JK} \right) \left(\sum_{IK} \gamma_{sIK} z_{IK} \right) \otimes \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} x_{ij} y_{jk} z_{ik}. \tag{5}$$

We claim that moreover,

$$\epsilon^{d+D} \langle nN, nN, nN \rangle + O(\epsilon^{d+D+1}) = \sum_{s=1}^{r} \left(\sum_{IJ} \alpha_{sIJ} x_{IJ} \right) \left(\sum_{JK} \beta_{sJK} y_{JK} \right) \left(\sum_{IK} \gamma_{sIK} z_{IK} \right) \otimes \left(\epsilon^d \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} x_{ij} y_{jk} z_{ik} + O(\epsilon^{d+1}) \right), \tag{6}$$

where the $O(\epsilon^{d+1})$ term matches the term appearing in (4). To obtain this from (5), first multiply both sides by ϵ^d, and then add the $O(\epsilon^{d+1})$ term. Since $\sum_{s=1}^{r} \left(\sum_{IJ} \alpha_{sIJ} x_{IJ} \right) \left(\sum_{JK} \beta_{sJK} y_{JK} \right) \left(\sum_{IK} \gamma_{sIK} z_{IK} \right)$ is a multiple of ϵ^D, this adds an error term of $O(\epsilon^{d+D+1})$ to the left-hand side.

Just as before, we can express sums of at most ℓ terms in the right-hand side of (6) as sums of r' rank one tensors using (4), and we can complete the proof in exactly the same way. □

The idea now is to use the fact that $\underline{R}(\langle N, N, N \rangle)$ grows like N^ω.

Lemma 13. *If $a, b > 0$ and $t \geq 1$ is an integer then*

$$(a+b)^{1/t} \leq a^{1/t} \left(1 + \frac{b}{ta} \right).$$

Proof. The binomial theorem shows that $(1 + b/(ta))^t \geq 1 + b/a$, and so $(1 + b/a)^{1/t} \leq 1 + b/(ta)$. We get the stated inequality by multiplying both sides by $a^{1/t}$. □

Theorem 14. *If $\underline{R}(\ell\langle n, n, n\rangle) \leq r$ then $\omega \leq \log_n(r/\ell)$. In other words,*

$$\ell n^\omega \leq \underline{R}(\ell\langle n, n, n\rangle).$$

Proof. We can assume that $\ell \geq 2$, since otherwise the result follows from Theorem 9.

Let $\epsilon > 0$ be given. Theorem 8 shows that there exists N_ϵ such that when $N \geq N_\epsilon$ we have $R(\langle N, N, N\rangle) \leq N^{\omega+\epsilon}$ (this also follows from Theorem 6, which shows that $R(\langle N, N, N\rangle) = O(N^{\omega+\epsilon/2})$, and in particular $R(\langle N, N, N\rangle) \leq N^{\omega+\epsilon}$ for large enough N). For every t we have $\underline{R}(\ell^t\langle n^t, n^t, n^t\rangle) \leq r^t$, and so Theorem 12 implies that

$$\underline{R}(\langle n^t N, n^t N, n^t N\rangle) \leq \left\lceil \frac{N^{\omega+\epsilon}}{\ell^t} \right\rceil r^t.$$

Theorem 9 then implies that

$$(n^t N)^\omega \leq \left\lceil \frac{N^{\omega+\epsilon}}{\ell^t} \right\rceil r^t < \left(\frac{r}{\ell}\right)^t (N^{\omega+\epsilon} + \ell^t).$$

Taking tth roots and applying Lemma 13, we see that

$$(nN^{1/t})^\omega < \frac{r}{\ell} N^{(\omega+\epsilon)/t} \left(1 + \frac{\ell^t}{tN^{\omega+\epsilon}}\right).$$

Cancelling $N^{\omega/t}$ on both sides and using $\omega + \epsilon \geq 1$, we obtain

$$n^\omega < \frac{r}{\ell} N^{\epsilon/t} \left(1 + \frac{\ell^t}{tN}\right).$$

All of this holds when $N \geq N_\epsilon$. Hence for $t \geq \log_\ell N_\epsilon$, we can choose $N = \ell^t$ to obtain

$$n^\omega \leq \frac{r}{\ell} \ell^\epsilon \left(1 + \frac{1}{t}\right).$$

Taking the limit $t \to \infty$, we get

$$n^\omega \leq \frac{r}{\ell} \ell^\epsilon.$$

Taking the limit $\epsilon \to \infty$, we conclude $n^\omega \leq r/\ell$, and so $\ell n^\omega \leq r$. □

As in several preceding cases, we can also handle $\langle n, m, p\rangle$ for general n, m, p.

Theorem 15. *If $\underline{R}(\ell\langle n, m, p\rangle) \leq r$ then $\omega \leq \log_{\sqrt[3]{nmp}}(r/\ell)$. In other words,*

$$\ell(nmp)^{\omega/3} \leq \underline{R}(\ell\langle n, m, p\rangle).$$

Proof. The proof is very similar to all previous cases. We use the identity $\underline{R}(\ell\langle n, m, p\rangle) = \underline{R}(\ell\langle m, p, n\rangle) = \underline{R}(\ell\langle p, n, m\rangle)$ together with the identity $\ell_1 T_1 \otimes \ell_2 T_2 = \ell_1\ell_2(T_1 \otimes T_2)$ to obtain

$$\underline{R}(\ell^3\langle nmp, nmp, nmp\rangle) \leq \underline{R}(\ell\langle n, m, p\rangle)\underline{R}(\ell\langle m, p, n\rangle)\underline{R}(\ell\langle p, n, m\rangle) \leq r^3.$$

Theorem 14 now shows that

$$\ell^3(nmp)^\omega \leq \underline{R}(\ell\langle n, m, p\rangle)^3.$$

We obtain the theorem by taking the cube root. □

We are finally ready to prove the asymptotic sum inequality, also known as Schönhage's τ theorem. Suppose that we have a bound on the border rank of $\bigoplus_{s=1}^{q} \ell_s\langle n_s, m_s, p_s\rangle$. How can we use that to bound ω? The asymptotic sum inequality provides the answer.

Theorem 16 (Asymptotic sum inequality)**.** *Let $T = \bigoplus_{s=1}^{q} \ell_s\langle n_s, m_s, p_s\rangle$, suppose that $\underline{R}(T) \leq r$, and suppose that τ is the solution of*

$$\sum_{s=1}^{q} \ell_s (n_s m_s p_s)^{\tau} = r.$$

Then $\omega \leq 3\tau$.

Notice how this generalizes Theorem 15. Before proving the asymptotic sum inequality, we define the notation of *value* of a tensor which is the direct sum of matrix multiplication tensors:

$$V_\tau\left(\bigoplus_{s=1}^{q} \ell_s\langle n_s, m_s, p_s\rangle\right) = \sum_{s=1}^{q} \ell_s (n_s m_s p_s)^{\tau}.$$

Notice that this is just the left-hand side of the equation in the asymptotic sum inequality. Also, Theorem 15 states that

$$V_{\omega/3}(\ell\langle n, m, p\rangle) \leq \underline{R}(\ell\langle n, m, p\rangle).$$

It is easy to check that the value is additive and multiplication:

$$V_\tau(T_1 \oplus T_2) = V_\tau(T_1) + V_\tau(T_2),\ V_\tau(T_1 \otimes T_2) = V_\tau(T_1)V_\tau(T_2).$$

Proof of Theorem 16. The idea in a nutshell is to take a high tensor power of the upper bound $\underline{R}(T) \leq r$, isolate one of the summands, and then apply Theorem 15. We choose the summand which gives us the best bound on ω. Figuring out which bound we get in this way is the hard part of the proof.

Since border rank satisfies $\underline{R}(T_1 \otimes T_2) \leq \underline{R}(T_1)\underline{R}(T_2)$, we see that $\underline{R}(T^{\otimes N}) \leq r^N$. The tensor $T^{\otimes N}$ is a direct sum of many matrix multiplication tensors:

$$T^{\otimes N} = \sum_{\substack{N_1,\ldots,N_q:\\ N_1+\cdots+N_q=N}} \ell_1^{N_1} \cdots \ell_q^{N_q} \langle n_1^{N_1} \cdots n_q^{N_q}, m_1^{N_1} \cdots m_q^{N_q}, p_1^{N_1} \cdots p_q^{N_q}\rangle.$$

Let us denote the summand corresponding to $N_1, \ldots, N_q$ by $T_{N_1,\ldots,N_q}$. The number of different summands is

$$\binom{N+q-1}{q-1} \leq CN^{q-1}$$

for some constant C (depending on q).

Since the value is multiplicative, we have $V_{\omega/3}(T^{\otimes N}) = V_{\omega/3}(T)^N$. Since it is additive,

$$\sum_{\substack{N_1,\ldots,N_q:\\ N_1+\cdots+N_q=N}} V_{\omega/3}(T_{N_1,\ldots,N_q}) = V_{\omega/3}(T)^N.$$

Since there are at most CN^{q-1} summands, one of them, say the one corresponding to $\nu_1, \ldots, \nu_q$, satisfies

$$V_{\omega/3}(T_{\nu_1,\ldots,\nu_q}) \geq \frac{V_{\omega/3}(T)^N}{CN^{q-1}}.$$

Substituting zero for all variables other than those appearing in $T_{\nu_1,\ldots,\nu_q}$, we see that $\underline{R}(T_{\nu_1,\ldots,\nu_q}) \leq r^N$. Hence Theorem 15 shows that

$$V_{\omega/3}(T)^N \leq CN^{q-1} V_{\omega/3}(T_{\nu_1,\ldots,\nu_q}) \leq CN^{q-1} \underline{R}(T_{\nu_1,\ldots,\nu_q}) \leq CN^{q-1} r^N.$$

Taking the Nth root, we obtain

$$V_{\omega/3}(T) \leq \sqrt[N]{CN^{q-1}}\, r.$$

Since $\sqrt[N]{CN^{q-1}} \longrightarrow 1$, taking the limit $N \to \infty$, we obtain $V_{\omega/3}(T) \leq r$. □

Schönhage showed that $\underline{R}(\langle 4,1,4\rangle \oplus \langle 1,9,1\rangle) \leq 17$. This is surprising, since $\underline{R}(\langle 4,1,4\rangle) = 16$. With one more essential product, we are suddenly able to also compute an additional inner product, which by itself would require rank 9. Applying the asymptotic sum inequality, we obtain $(16)^{\omega/3} + 9^{\omega/3} \leq 17$, or $\omega \leq 2.55$.

Coppersmith and Winograd showed that identities can always be improved in this way. They showed that if an upper bound $\underline{R}(T) \leq r$ gives some upper bound on ω via an application of the asymptotic sum inequality, then there in an "improved" upper bound on the border rank of $T \oplus \langle 1, N, 1\rangle$, for an appropriate N, which results in a better bound on ω via the asymptotic sum inequality. In other words, a single application of the asymptotic sum inequality cannot result in an optimal bound on ω.

Strassen developed the *laser method*, which gives an analog of the asymptotic sum inequality for *non-disjoint* tensors, in some cases. His method uses infinitely many applications of the asymptotic sum inequality, and so in principle could result in an optimal bound on ω. This method is used to prove the best bound on ω currently known, $\omega < 2.376$, due to Coppersmith and Winograd, and slightly improved by Stothers, by Vassilevska-Williams, and by Le Gall. See also the work of Ambainis, Filmus and Le Gall.

A completely different method for proving upper bounds on ω, using group theory, has been developed by Cohn and Umans. This method hasn't yielded any improved bounds so far.

9 Coppersmith–Winograd algorithm

The asymptotic sum inequality allows us to obtain a bound on ω given a bound on the border rank of a disjoint sum of matrix multiplication tensors. In this section, we briefly discuss the laser method and its application by Coppersmith and Winograd, a method which in some cases allows us to handle *non-disjoint* sums.

We will focus on the two tensors analyzed by Coppersmith and Winograd:

$$\begin{aligned} T_{simp} &= \sum_{i=1}^{q} x_0^{[0]} y_i^{[1]} z_i^{[1]} + \sum_{i=1}^{q} x_i^{[1]} y_0^{[0]} z_i^{[1]} + \sum_{i=1}^{q} x_i^{[1]} y_i^{[1]} z_0^{[0]}, \\ T_{CW} &= T_{simp} + x_{q+1}^{[2]} y_0^{[0]} z_0^{[0]} + x_0^{[0]} y_{q+1}^{[2]} z_0^{[0]} + x_0^{[0]} y_0^{[0]} z_{q+1}^{[2]}. \end{aligned}$$

Here q is an arbitrary non-negative integer. We will explain the superscripts in a moment. The tensor T_{simp} is $(q+1) \times (q+1) \times (q+1)$, and the tensor T_{CW} is $(q+2) \times (q+2) \times (q+2)$. Coppersmith and Winograd showed that both tensors have border rank $q+2$.

The rows, columns, and depths of T_{simp} are each partitioned into two sets, according to their subscript. For example, the rows are partitioned into the sets

$$X^{[0]} = \{x_0\},\ X^{[1]} = \{x_1, \ldots, x_q\}.$$

Similarly, the rows, columns, and depths of T_{CW} are partitioned into three sets:

$$X^{[0]} = \{x_0\},\ X^{[1]} = \{x_1, \ldots, x_q\},\ X^{[2]} = \{x_{q+1}\}.$$

We can express these tensors succinctly as follows:

$$\begin{aligned} T_{simp} &= \langle 1,1,q\rangle^{[0,1,1]} + \langle q,1,1\rangle^{[1,0,1]} + \langle 1,q,1\rangle^{[1,1,0]}, \\ T_{CW} &= \langle 1,1,q\rangle^{[0,1,1]} + \langle q,1,1\rangle^{[1,0,1]} + \langle 1,q,1\rangle^{[1,1,0]} + \langle 1,1,1\rangle^{[2,0,0]} + \langle 1,1,1\rangle^{[0,2,0]} + \langle 1,1,1\rangle^{[0,0,2]}. \end{aligned}$$

For example, $\langle 1,1,q\rangle^{[0,1,1]}$ means a tensor of the form $\langle 1,1,q\rangle$ in which the rows belong to part 0, the columns to part 1, and the depths to part 1. Notice that in each constituent the three part numbers sum to 2. This will become crucial later.

The sums in T_{simp}, T_{CW} are not direct sums, since they share variables. For this reason we cannot apply the asymptotic sum inequality directly. Instead, the idea of the laser method is to take a high tensor power, to zero out variables in such a way that the resulting tensor becomes a disjoint sum, and then to apply the asymptotic sum inequality.

In more detail, considering the tensor T_{simp}, we raise $\underline{R}(T_{simp}) \leq q+2$ to the Nth tensor power, obtaining $\underline{R}(T_{simp}^{\otimes N}) \leq (q+2)^N$. We can express $T_{simp}^{\otimes N}$ as a non-disjoint sum of 3^N tensors, for example

$$\begin{aligned} T_{simp}^{\otimes 2} = {} & \langle 1,1,q^2\rangle^{[00,11,11]} + \langle q,1,q\rangle^{[01,10,11]} + \langle 1,q,q\rangle^{[01,11,10]} \\ & + \langle q,1,q\rangle^{[10,01,11]} + \langle q^2,1,1\rangle^{[11,00,11]} + \langle q,q,1\rangle^{[11,01,10]} \\ & + \langle 1,q,q\rangle^{[10,11,01]} + \langle q,q,1\rangle^{[11,10,01]} + \langle 1,q^2,1\rangle^{[11,11,00]}. \end{aligned}$$

The indices correspond to the natural partition of the rows, columns, and depths of $T_{simp}^{\otimes 2}$ into four parts:

$$X^{[00]} = \{x_{00}\},\ X^{[01]} = \{x_{01},\ldots,x_{0q}\},\ X^{[10]} = \{x_{10},\ldots,x_{q0}\},\ X^{[11]} = \{x_{11},\ldots,x_{1q},\ldots,x_{q1},\ldots,x_{qq}\}.$$

The next step is to substitute zeroes for some of the variables, in such a way that the resulting tensor is a disjoint sum of matrix multiplication tensors. We substitute zeroes for entire parts. Continuing our example, if we zero out all variables in $X^{[00]}, X^{[01]}, Y^{[00]}, Y^{[11]}, Z^{[00]}, Z^{[10]}$ then we are left with

$$\langle q,1,q\rangle^{[10,01,11]} + \langle q,q,1\rangle^{[11,10,01]}.$$

Since the rows, the columns, and the depths are disjoint (we can tell since the indices of the row parts, column parts, and depth parts don't repeat), this is actually equivalent to the direct sum $\langle q,1,q\rangle \oplus \langle q,q,1\rangle$. Since $\underline{R}(T_{simp}^{\otimes 2}) \leq (1+2)^2$ and substitution doesn't increase the border rank (since a rank one tensor stays a rank one tensor under substitution), the asymptotic sum inequality implies that

$$2(q^2)^{\omega/3} \leq (q+2)^2.$$

Choosing $q = 22$, we obtain the bound $\omega < 2.7481$.

We can do the same with a higher tensor power: starting with $\underline{R}(T_{simp}^{\otimes N}) \leq (q+2)^N$, we zero out groups of variables so that whatever remains is a sum of tensors on *disjoint* variables, and then we apply the asymptotic sum inequality. If after zeroing out the variables in the best possible way there are $C(N)$ remaining tensors, then the asymptotic sum inequality gives the bound

$$C(N)(q^N)^{\omega/3} \leq (q+2)^N.$$

This is because all the tensors are of the form $\langle q^a, q^b, q^c\rangle$ where $a+b+c = N$. Taking Nth roots,

$$C(N)^{1/N} q^{\omega/3} \leq q+2.$$

The goal now is to determine the limit of $C(N)^{1/N}$ for the best zeroing strategy.

How large can we expect $C(N)^{1/N}$ to be? After zeroing out, all the row parts must be different, and so $C(N) \leq 2^N$. This shows that $\limsup C(N)^{1/N} \leq 2$. We can improve this naive bound using the *method of types*. The *type* of a constituent $\langle q^a, q^b, q^c\rangle$ of $T_{simp}^{\otimes N}$ is (a,b,c). Note that a is the number of factors $\langle q,1,1\rangle^{[1,0,1]}$ contributing to the constituent, b is the number of factors $\langle 1,q,1\rangle^{[1,1,0]}$, and c is the number of vectors $\langle 1,1,q\rangle^{[0,1,1]}$. The total number of types is $|\{(a,b,c) : a+b+c = N\}| = \binom{N+2}{2}$.

Consider now a specific type (a,b,c). After zeroing out, all row parts must be different. Each row part in a constituent of type (a,b,c) contains c zeroes and $a+b$ ones, and so after zeroing out, the number of surviving constituents of type (a,b,c), which we denote $C_{a,b,c}(N)$, satisfies $C_{a,b,c}(N) \leq \binom{N}{c}$. By considering also the column parts and the depth parts, we obtain (assuming for simplicity that N is divisible by 3)

$$C_{a,b,c}(N) \leq \min\left\{\binom{N}{a}\binom{N}{b}\binom{N}{c}\right\} \leq \binom{N}{N/3},$$

since $\min(a,b,c) \leq N/3$. A classical estimate[6] states that $\binom{N}{pN} \leq 2^{h(p)N}$, where $h(p) = -p\log_2 p - (1-$

[6] You can get this estimate using Stirling's approximation. A nicer approach is using information theory. Let $X = (X_1,\ldots,X_n)$ be a random variable over $\{0,1\}^n$ chosen uniformly at random from all vectors of sum pn, so that $H(X) = \log_2\binom{n}{pn}$. On the other hand, $H(X) \leq \sum_{i=1}^n H(X_i)$. Since X_i is binary and $\Pr[X_i = 1] = p$, we have $H(X_i) = h(\Pr[X_i = 1]) = h(p)$, and so $\log_2\binom{n}{pn} \leq h(p)n$. In the same way one can show that for $p \leq 1/2$ we have $\sum_{k=0}^{pn}\binom{n}{k} \leq 2^{h(p)n}$.

$p)\log_2(1-p)$ is the binary entropy function. Therefore

$$C(N) \leq \sum_{\substack{a,b,c: \\ a+b+c=N}} C_{a,b,c}(N) \leq \binom{N+2}{2}\binom{N}{N/3} \leq O(N^2)2^{h(1/3)N}.$$

This shows that

$$\limsup_{N\to\infty} C(N)^{1/N} = \limsup_{N\to\infty} O(N^2)^{1/N}2^{h(1/3)} \leq 2^{h(1/3)} = \frac{3}{2^{2/3}}.$$

Amazingly, it is in fact the case that

$$\lim_{N\to\infty} C(N)^{1/N} = \frac{3}{2^{2/3}}.$$

In other words, there is a construction matching the simple upper bound given by the method of types! The construction is quite non-trivial, and is beyond the scope of these notes. For this construction to work, it is essential that the sum of the row, column and depth indices for all three summands in T_{simp} is the same. The end result is that for every q, we get the bound

$$\frac{3}{2^{2/3}}q^{\omega/3} \leq q+2.$$

In particular, when $q=8$ we get $\omega < 2.404$.

The same machinery can be applied to T_{CW}. There is a slight complication, in that whereas all tensors in $T_{simp}^{\otimes N}$ had the same *volume* (the volume of the tensor $\langle n,m,p\rangle$ is nmp), tensors in $T_{CW}^{\otimes N}$ have different volumes. However, this can be taken into account, and the end result is the bound $\omega < 2.388$.

Coppersmith and Winograd obtained the improved bound $\omega < 2.376$ by considering a "merged" version of $T_{CW}^{\otimes 2}$, a technique which is beyond the scope of these notes. Stothers obtained the improved bound $\omega < 2.373$ by considering a merged version of $T_{CW}^{\otimes 4}$. Merged versions of higher powers were considered by Vassilevska-Williams and by Le Gall, who obtained slightly improved bounds (the bounds differ only on the fourth digit after the point or beyond). Ambainis, Filmus and Le Gall showed that considering merged versions of higher powers of T_{CW} could only improve the bounds on ω by a little bit.

We close these notes by mentioning two other directions of research. The first is *rectangular matrix multiplication*, in which the goal is to obtain lower bounds on α. The second is the *group-theoretic approach*, due to Cohn and Umans, which is a completely different approach to the theory of fast matrix multiplication. Both of these are beyond the scope of these notes.