1

Introduction

A dramatic leap for signal processing occurred in the 1960s with the implementation of the fast Fourier transform, an algorithm that surprised the engineering community with its efficiency.¹ Is there a way to predict the existence of such fast unexpected algorithms? Can we determine when they do not exist? Complexity theory addresses these questions.

This book is concerned with the use of geometry toward these goals. I focus primarily on two central questions: the complexity of matrix multiplication and algebraic variants of the famous P versus NP problem. In the first case, a surprising algorithm exists, and it is conjectured that even better algorithms exist. In the second case, it is conjectured that no surprising algorithm exists.

In this chapter I introduce the main open questions discussed in this book, establish notation that is used throughout the book, and introduce fundamental geometric notions.

1.1 Matrix Multiplication

Much of scientific computation amounts to linear algebra, and the basic operation of linear algebra is matrix multiplication. All operations of linear algebra – solving systems of linear equations, computing determinants, etc. – use matrix multiplication.

1.1.1 The Standard Algorithm

The standard algorithm for multiplying matrices is row-column multiplication: let \( A, B \) be \( 2 \times 2 \) matrices

\[
A = \begin{pmatrix} a_1^1 & a_1^2 \\ a_2^1 & a_2^2 \end{pmatrix}, \quad B = \begin{pmatrix} b_1^1 & b_1^2 \\ b_2^1 & b_2^2 \end{pmatrix}.
\]

¹ To this day, it is not known if there is an even more efficient algorithm than the FFT. See [Val77, Lok08, KLPSMN09, GHIL16].
Introduction

Remark 1.1.1.1 While computer scientists generally keep all indices down (to distinguish from powers), I use the convention from differential geometry that in a matrix $X$, the entry in the $i$th row and $j$th column is labeled $x_{ij}$.

The usual algorithm to calculate the matrix product $C = AB$ is

\[
\begin{align*}
c_{11} &= a_1^1 b_1^1 + a_2^1 b_1^2, \\
c_{12} &= a_1^1 b_2^1 + a_2^1 b_2^2, \\
c_{21} &= a_1^2 b_1^1 + a_2^2 b_1^2, \\
c_{22} &= a_1^2 b_2^1 + a_2^2 b_2^2.
\end{align*}
\]

It requires 8 multiplications and 4 additions to execute, and applied to $n \times n$ matrices, it uses $n^3$ multiplications and $n^3 - n^2$ additions.

This algorithm has been around for about two centuries. In 1968, V. Strassen set out to prove the standard algorithm was optimal in the sense that no algorithm using fewer multiplications exists (personal communication). Since that might be difficult to prove, he set out to show it was true at least for $2 \times 2$ matrices – at least over $\mathbb{Z}_2$. His spectacular failure opened up a whole new area of research.

1.1.2 Strassen’s Algorithm for Multiplying $2 \times 2$ Matrices using 7 Scalar Multiplications [Str69]

Set

\[
\begin{align*}
I &= (a_1^1 + a_2^2)(b_1^1 + b_2^2), \\
II &= (a_1^2 + a_2^1)b_1^1, \\
III &= a_1^1(b_1^2 - b_2^2), \\
IV &= a_2^2(-b_1^1 + b_1^2), \\
V &= (a_1^1 + a_2^1)b_2^2, \\
VI &= (-a_1^1 + a_2^2)(b_1^1 + b_2^1), \\
VII &= (a_2^1 - a_2^2)(b_1^2 + b_2^2).
\end{align*}
\]

Exercise 1.1.2.1 (1) Show that if $C = AB$, then

\[
\begin{align*}
c_{11} &= I + IV - V + VII, \\
c_{12} &= II + IV, \\
c_{21} &= III + V, \\
c_{22} &= I + III - II + VI.
\end{align*}
\]
1.1 Matrix Multiplication

This raises questions:

1. Can one find an algorithm that uses just six multiplications?
2. Could Strassen’s algorithm have been predicted in advance?
3. Since it uses more additions, is it actually better in practice?
4. This algorithm was found by accident and looks ad hoc. Is there any way to make sense of it? For example, is there any way to see that it is correct other than a brute force calculation?
5. What about algorithms for \( n \times n \) matrices?

I address question 4 in §1.1.15 and the others below, with the last question first.

1.1.3 Fast Multiplication of \( n \times n \) Matrices

In Strassen’s algorithm, the entries of the matrices need not be scalars – they could themselves be matrices. Let \( A, B \) be \( 4 \times 4 \) matrices, and write

\[
A = \begin{pmatrix} a_1^1 & a_2^1 \\ a_1^2 & a_2^2 \end{pmatrix}, \quad B = \begin{pmatrix} b_1^1 & b_2^1 \\ b_1^2 & b_2^2 \end{pmatrix}.
\]

where \( a_j^i, b_j^i \) are \( 2 \times 2 \) matrices. One may apply Strassen’s algorithm to get the blocks of \( C = AB \) in terms of the blocks of \( A, B \) by performing seven multiplications of \( 2 \times 2 \) matrices. Since one can apply Strassen’s algorithm to each block, one can multiply \( 4 \times 4 \) matrices using \( 7^2 = 49 \) multiplications instead of the usual \( 4^3 = 64 \).

If \( A, B \) are \( 2^k \times 2^k \) matrices, one may multiply them using \( 7^k \) multiplications instead of the usual \( 8^k \). If \( n \) is not a power of 2, enlarge the matrices with blocks of zeros to obtain matrices whose size is a power of 2. Asymptotically, by recursion and block multiplication, one can multiply \( n \times n \) matrices using approximately \( n \log_2(7) \approx n^{2.81} \) multiplications. To see this, let \( n = 2^k \) and write \( 7^k = (2^k)^a \), so \( a = \log_2 7 \).

1.1.4 Regarding the Number of Additions

The number of additions in Strassen’s algorithm also grows like \( n^{2.81} \), so this algorithm is more efficient in practice when the matrices are large. For any efficient algorithm for matrix multiplication, the total complexity is governed by the number of multiplications; see [BCS97, Proposition 15.1]. This is fortuitous because there is a geometric object, tensor rank, discussed in §1.1.11, that counts the number of multiplications in an optimal algorithm (within a
factor of 2), and thus provides a geometric measure of the complexity of matrix multiplication.

Just how large a matrix must be in order to obtain a substantial savings with Strassen’s algorithm (a size of about 2,000 suffices) and other practical matters are addressed in [BB].

1.1.5 An Even Better Algorithm?

Regarding question (1) above, one cannot improve upon Strassen’s algorithm for $2 \times 2$ matrices. This was first shown in [Win71]. I will give a proof, using geometry and representation theory, of a stronger statement in §8.3.2. However, for $n > 2$, very little is known, as discussed below and in Chapters 2–5. What is known is that better algorithms than Strassen’s exist for $n \times n$ matrices when $n$ is large.

1.1.6 How to Predict in Advance?

The answer to question (2) is yes! In fact, it could have been predicted 100 years ago.

Had someone asked Terracini in 1913, he would have been able to predict the existence of something like Strassen’s algorithm from geometric considerations alone. Matrix multiplication is a bilinear map (see §1.1.9). Terracini would have been able to tell you, thanks to a simple parameter count (see §2.1.6), that even a general bilinear map $\mathbb{C}^4 \times \mathbb{C}^4 \to \mathbb{C}^4$ can be executed using seven multiplications, and thus, fixing any $\epsilon > 0$, one can perform any bilinear map $\mathbb{C}^4 \times \mathbb{C}^4 \to \mathbb{C}^4$ within an error of $\epsilon$ using seven multiplications.

1.1.7 Big/Little O, etc., Notation

For functions $f$, $g$ of a real variable (or integer) $x$,

\[
\begin{align*}
 f(x) &= O(g(x)) \text{ if there exists a constant } C > 0 \text{ and } x_0 \text{ such that } |f(x)| \leq C|g(x)| \text{ for all } x \geq x_0, \\
 f(x) &= o(g(x)) \text{ if } \lim_{x \to \infty} \frac{|f(x)|}{|g(x)|} = 0, \\
 f(x) &= \Omega(g(x)) \text{ if there exists a constant } C > 0 \text{ and } x_0 \text{ such that } C|f(x)| \geq |g(x)| \text{ for all } x \geq x_0, \\
 f(x) &= \omega(g(x)) \text{ if } \lim_{x \to \infty} \frac{|g(x)|}{|f(x)|} = 0, \text{ and} \\
 f(x) &= \Theta(g(x)) \text{ if } f(x) = O(g(x)) \text{ and } f(x) = \Omega(g(x)).
\end{align*}
\]

1.1.8 The Exponent of Matrix Multiplication

The following quantity is the standard measure of the complexity of matrix multiplication.
1.1 Matrix Multiplication

Definition 1.1.8.1 The exponent $\omega$ of matrix multiplication is

$$\omega := \inf \{ h \in \mathbb{R} \mid n \times n \text{ matrices can be multiplied using } O(n^h) \text{ arithmetic operations} \},$$

where $\inf$ denotes the infimum.

By Theorem 1.1.11.3, Strassen’s algorithm shows $\omega \leq \log_2(7) < 2.81$, and it is easy to prove $\omega \geq 2$. Determining $\omega$ is a central open problem in complexity theory. After Strassen’s work, it was shown in 1978 that $\omega \leq 2.7962$ [Pan78], then $\omega \leq 2.7799$ [Bin80] in 1979, then $\omega \leq 2.55$ [Sch81] in 1981, then $\omega \leq 2.48$ [Str87] in 1987, and then $\omega \leq 2.38$ [CW90] in 1989, which might have led people in 1990 to think a resolution was near. However, then nothing happened for more than 20 years, and the current “world record” of $\omega < 2.373$ [Wil, LG14, Sto] is not much of an improvement since 1990. These results are the topic of Chapter 3.

If one is interested in multiplying matrices of reasonable size, only the algorithms in [Pan78, Smi13] are known to beat Strassen’s. This “practical” exponent is discussed in Chapter 4.

The above work has led to the following astounding conjecture.

Conjecture 1.1.8.2 $\omega = 2$.

That is, it is conjectured that, asymptotically, it is nearly just as easy to multiply matrices as it is to add them!

Although I am unaware of anyone taking responsibility for the conjecture, most computer scientists I have discussed it with expect it to be true. (For example, it is possible to multiply $n$-bit integers in near-linear time $O(n \log(n))$, which is almost as efficient as adding them.)

I have no opinion on whether the conjecture should be true or false and thus discuss both upper and lower bounds for the complexity of matrix multiplication, focusing on the role of geometry. Chapters 2 and 5 are dedicated to lower bounds and Chapters 3 and 4 to upper bounds.

1.1.9 Matrix Multiplication as a Bilinear Map

I will use the notation

$$M_{(n,m,l)} : \mathbb{C}^{n \times m} \times \mathbb{C}^{m \times l} \rightarrow \mathbb{C}^{n \times l}$$

for matrix multiplication of an $n \times m$ matrix with an $m \times l$ matrix, and write $M_{(n)} = M_{(n,n,n)}$. 

Matrix multiplication is a bilinear map; that is, for all $X_j, X \in \mathbb{C}^{n \times m}, Y_j, Y \in \mathbb{C}^{m \times l}$, one can always change bases, i.e., multiply $X$ on the left by an invertible $a \times a$ matrix and on the right by an invertible $b \times b$ matrix, to obtain a matrix with some number of 1s along the diagonal and zeros elsewhere. The number of 1s appearing is called the rank of the matrix, and it is the rank of the linear map $X$. In other words, the only property that we can make in a matrix representing the map, and for each rank, we have a normal form. This is not surprising, because the dimension of the space of such linear maps is $n \times m \times l$. If we represent a bilinear map by a three-dimensional matrix, it may be thought of as taking two column vectors and returning a third column vector.

### 1.1.10 Ranks of Linear Maps

I use the notation $\mathbb{C}^a$ for the column vectors of height $a$ and $\mathbb{C}^{a*}$ for the row vectors.

**Definition 1.1.10.1** A linear map $f : \mathbb{C}^a \rightarrow \mathbb{C}^b$ has rank one if there exist $\alpha \in \mathbb{C}^{a*}$ and $\omega \in \mathbb{C}^b$ such that $f(\omega) = \alpha(\omega)w$. (In other words, every rank one matrix is the product of a row vector with a column vector.) In this case I write $f = \alpha \otimes \omega$. The rank of a linear map $h : \mathbb{C}^a \rightarrow \mathbb{C}^b$ is the smallest $r$ such that $h$ may be expressed as a sum of $r$ rank one linear maps.

Given an $a \times b$ matrix $X$, one can always change bases, i.e., multiply $X$ on the left by an invertible $a \times a$ matrix and on the right by an invertible $b \times b$ matrix, to obtain a matrix with some number of 1s along the diagonal and zeros elsewhere. The number of 1s appearing is called the rank of the matrix, and it is the rank of the linear map $X$. In other words, the only property of a linear map $\mathbb{C}^a \rightarrow \mathbb{C}^b$ that is invariant under changes of bases is its rank, and for each rank, we have a normal form. This is not surprising, because the dimension of the space of such linear maps is $ab$, we have $a^2$ parameters of changes of bases in $\mathbb{C}^a$ that we can make in a matrix representing the map, and $a^2 + b^2 > ab$.

### 1.1.11 Tensor Rank

For bilinear maps $\mathbb{C}^a \times \mathbb{C}^b \rightarrow \mathbb{C}^c$ we are not so lucky as with linear maps, as usually $abc > a^2 + b^2 + c^2$, i.e., there are fewer free parameters of changes of
bases than the number of parameters needed to describe the map. This indicates why the study of bilinear maps is vastly more complicated than the study of linear maps.

Nonetheless, there are properties of a bilinear map that will not change under a change of basis. The most important properties for complexity are tensor rank and tensor border rank. Tensor border rank is defined in §1.1.12. Tensor rank is a generalization of the rank of a linear map. Tensor rank is defined properly in §2.1.3. Informally, a bilinear map $T$ has tensor rank one if it can be computed with one multiplication. More precisely, $T$ has tensor rank one if, in some coordinate system, the multidimensional matrix representing it has exactly one nonzero entry. This may be expressed without coordinates:

**Definition 1.1.11.1** $T \in \mathbb{C}^{a\ast} \otimes \mathbb{C}^{b\ast} \otimes \mathbb{C}^{c\ast}$ has tensor rank one if there exist row vectors $\alpha \in \mathbb{C}^{a\ast}$, $\beta \in \mathbb{C}^{b\ast}$ and a column vector $w \in \mathbb{C}^{c\ast}$ such that $T(u, v) = \alpha(u) \beta(v) w$. $T$ has tensor rank $r$ if it can be written as the sum of $r$ rank one tensors but no fewer, in which case we write $R(T) = r$. Let $\hat{\sigma}^0_r$ denote the set of bilinear maps in $\mathbb{C}^{a\ast} \otimes \mathbb{C}^{b\ast} \otimes \mathbb{C}^{c\ast}$ of tensor rank at most $r$.

**Remark 1.1.11.2** The peculiar notation $\hat{\sigma}^0_r$ will be explained in §4.7.1. For now, to give an idea where it comes from, $\sigma_r = \sigma_r(\text{Seg}(\mathbb{P}^{a-1} \times \mathbb{P}^{b-1} \times \mathbb{P}^{c-1}))$ is standard notation in algebraic geometry for the $r$th secant variety of the Segre variety, which is the object we will study. The hatted object $\hat{\sigma}$ denotes its cone in affine space, and the 0 indicates the subset of this set consisting of tensors of rank at most $r$.

The following theorem shows that tensor rank is a legitimate measure of complexity.

**Theorem 1.1.11.3** (Strassen [Str69]; also see [BCS97, §15.1])

$$\omega = \inf \{ \tau \in \mathbb{R} \mid R(M_{(n)}) = O(n^\tau) \}.$$  

That is, $n \times n$ matrices may be multiplied using $O(n^{\omega+\epsilon})$ arithmetic operations if and only if the tensor rank of $M_{(n)}$ is $O(n^{\omega+\epsilon})$.

Our goal is thus to determine, for a given $r$, whether or not matrix multiplication lies in $\delta^0_r$.

### 1.1.12 How to Use Algebraic Geometry to Prove Lower Bounds for the Complexity of Matrix Multiplication

Algebraic geometry deals with the study of zero sets of polynomials. By a polynomial on the space of bilinear maps $\mathbb{C}^{a\ast} \otimes \mathbb{C}^{b\ast} \otimes \mathbb{C}^{c\ast}$, I mean a polynomial in the coefficients $t^{ijk}$, i.e., in $abc$ variables. In §1.1.14 I describe a plan to use...
Introduction

algebraic geometry to prove upper complexity bounds. A plan to use algebraic geometry for lower bounds is as follows.

**Plan to Show** $M_{(n,m,l)} \not\in \delta^0_r$ via Algebraic Geometry

- Find a polynomial $P$ on the space of bilinear maps $\mathbb{C}^{nm} \times \mathbb{C}^{ml} \to \mathbb{C}^{nl}$, such that $P(T) = 0$ for all $T \in \delta^0_r$.
- Show that $P(M_{(n,m,l)}) \neq 0$.

Chapters 2 and 5 discuss techniques for finding such polynomials, using algebraic geometry and *representation theory*, the study of symmetry in linear algebra.

### 1.1.13 Representation Theory

Representation theory is the systematic study of symmetry. We will primarily be concerned with properties of bilinear maps, tensors, polynomials, etc., that are invariant under changes of bases. Representation theory will facilitate the study of these properties. It has been essential for proving lower bounds for the complexity of $M_{(n)}$.

Let $V$ be a complex vector space of dimension $v$. (I reserve the notation $\mathbb{C}^v$ for the column vectors with their standard basis.) Let $GL(V)$ denote the group of invertible linear maps $V \to V$, and I write $GL_v$ for $GL(\mathbb{C}^v)$. If we have fixed a basis of $V$, this is the group of invertible $v \times v$ matrices. If $G$ is a group and $\mu : G \to GL(V)$ is a group homomorphism, we will say $G$ acts on $V$ and that $V$ is a $G$-module. The image of $\mu$ is called a representation of $G$.

For example, the *permutation group* on $n$ elements $\mathfrak{S}_n$ acts on $\mathbb{C}^n$ by

$$\sigma \left( \begin{array}{c} v_1 \\ \vdots \\ v_n \end{array} \right) = \left( \begin{array}{c} v_{\sigma^{-1}(1)} \\ \vdots \\ v_{\sigma^{-1}(n)} \end{array} \right),$$

where $\sigma \in \mathfrak{S}_n$ is a permutation. That is, the image of $\mathfrak{S}_n$ in $GL_n$ is the set of permutation matrices. (The inverse is used so that for a vector $\vec{v}$, $\sigma(\tau \vec{v}) = (\sigma \tau) \vec{v}$.)

A group action is *irreducible* if there does not exist a proper subspace $U \subset V$ such that $\mu(g)u \in U$ for all $u \in U$ and $g \in G$.

The action of $\mathfrak{S}_n$ on $\mathbb{C}^n$ is not irreducible since the line spanned by $e_1 + \cdots + e_n$ is preserved by $\mathfrak{S}_n$. Note that the subspace spanned by $e_1 - e_2, \ldots, e_1 - e_n$ is also preserved by $\mathfrak{S}_n$. Both these subspaces are irreducible $\mathfrak{S}_n$-modules.

The essential point is that the sets $X$, such as $X = \delta^0_r \subset \mathbb{C}^{abc}$, for which we want polynomials that are zero at the points of $X$ are invariant under the action of groups.
1.1 Matrix Multiplication

Definition 1.1.13.1 A set \( X \subset V \) is invariant under a group \( G \subset GL(V) \) if, for all \( x \in X \) and all \( g \in G \), \( g(x) \in X \). Let \( G_X \subset GL(V) \) denote the group preserving \( X \), the largest subgroup of \( GL(V) \) under which \( X \) is invariant.

When one says that an object has symmetry, it means the object is invariant under the action of a group.

In the case at hand, \( X = \hat{\sigma}_0^r \subset V = A \otimes B \otimes C \). Then \( \hat{\sigma}_0^r \) is invariant under the action of the subgroup \( GL(A) \times GL(B) \times GL(C) \) of \( GL(V) \), i.e., this subgroup lies in \( G_{\hat{\sigma}_0^r} \).

Recall that an ideal \( I \) is a vector subspace such that for all \( P \in I \) and \( Q \in R \), \( PQ \in I \).

Definition 1.1.13.2 For a set \( X \subset V \), we will say a polynomial \( P \) vanishes on \( X \) if \( P(x) = 0 \) for all \( x \in X \). The set of all polynomials vanishing on \( X \) forms an ideal in the space of polynomials on \( V \), called the ideal of \( X \) and denoted \( I(X) \).

If a polynomial \( P \) is in the ideal of \( X \), then the polynomial \( g(P) \) will also be in the ideal of \( X \) for all \( g \in G_X \). That is, the ideal of polynomials vanishing on \( X \) is a \( G_X \)-module.

The systematic exploitation of symmetry is used throughout this book: to study the ideals of varieties such as \( \hat{\sigma}_0^r \) via their irreducible components in Chapter 2, to find new decompositions of the matrix multiplication tensor in Chapter 4, to find normal forms, e.g., in order to prove the state of the art lower bound for the complexity of matrix multiplication in Chapter 5, and to define the only restricted model where an exponential separation of the permanent from the determinant is known in Chapter 7. Chapter 8 is dedicated to representation theory, and Chapters 9 and 10 approach problems in algebraic geometry using representation theory.

1.1.14 How to Use Algebraic Geometry to Prove upper Bounds for the Complexity of Matrix Multiplication

Based on the above discussion, one could try the following strategy.

Plan to show \( M_{\langle n,m,l \rangle} \in \hat{\sigma}_0^r \) with algebraic geometry.

- Find a set of polynomials \( \{P_j\} \) on the space of bilinear maps \( \mathbb{C}^{nm} \times \mathbb{C}^{ml} \to \mathbb{C}^{nl} \) such that \( T \in \hat{\sigma}_0^r \) if and only if \( P_j(T) = 0 \) for all \( j \).
- Show that \( P_j(M_{\langle n,m,l \rangle}) = 0 \) for all \( j \).

This plan has a problem: consider the set \( S = \{(w, z) \in \mathbb{C}^2 \mid z = 0, w \neq 0\} \), whose real picture looks like the \( z \)-axis with the origin removed:
Any polynomial $P \in I(S)$, i.e., any $P$ that evaluates to zero at all points of $S$, will also be zero at the origin:

**Exercise 1.1.14.1** (1!) Prove the above assertion.

Just as in this example, the zero set of the polynomials vanishing on $\hat{\sigma}^0_r$ is larger than $\hat{\sigma}^0_r$ when $r > 1$ (see §2.1.5), so one cannot certify membership in $\hat{\sigma}^0_r$ via polynomials, but rather its Zariski closure, which I now define.

**Definition 1.1.14.2** The Zariski closure of a set $S \subset V$, denoted $\overline{S}$, is the set of $u \in V$ such that $P(u) = 0$ for all $P \in I(S)$. A set $S$ is said to be Zariski closed or an algebraic variety if $S = \overline{S}$, i.e., $S$ is the common zero set of a collection of polynomials.

In the example above, $S = \{(w, z) \in \mathbb{C}^2 \mid z = 0\}$.

When $U = \mathbb{C}^a \otimes \mathbb{C}^b \otimes \mathbb{C}^c$, let $\hat{\sigma}_r := \overline{\hat{\sigma}^0_r}$ denote the Zariski closure of the set of bilinear maps of tensor rank at most $r$.

We will see that for almost all $a, b, c$, and $r$, $\hat{\sigma}^0_r \subsetneq \hat{\sigma}_r$. The problem with the above plan is that it would only show $M_{(a)} \in \hat{\sigma}_r$.

**Definition 1.1.14.3** $T \in \mathbb{C}^a \otimes \mathbb{C}^b \otimes \mathbb{C}^c$ has border rank $r$ if $T \in \hat{\sigma}_r$ and $T \not\in \hat{\sigma}_{r-1}$. In this case we write $R(T) = r$.

For the study of the exponent of matrix multiplication, we have good luck:

**Theorem 1.1.14.4** (Bini [Bin80], see §3.2)

$$\omega = \inf \{ \tau \in \mathbb{R} \mid R(M_{(n)}) = O(n^\tau) \}.$$  

That is, although we may have $R(M_{(a)}) < R(M_{(a)})$, they are not different enough to affect the exponent. In other words, as far as the exponent is concerned, the plan does not have a problem.

For $n = 2$, we will see that $R(M_{(2)}) = R(M_{(3)}) = 7$. It is expected that for $n > 2$, $R(M_{(n)}) < R(M_{(n)})$. For $n = 3$, we only know $16 \leq R(M_{(3)}) \leq 20$ and $19 \leq R(M_{(3)}) \leq 23$. In general, we know $R(M_{(n)}) \geq 3n^2 - o(n)$ (see §2.6), and $R(M_{(n)}) \geq 2n^2 - \lceil \log_2(n) \rceil - 1$ (see §5.4.5).

### 1.1.15 Symmetry and Algorithms

In this subsection I mention three uses of symmetry groups in the study of algorithms.

I first address the question raised in §1.1.2: can we make sense of Strassen’s algorithm (1.1.1)? Just as the set $\hat{\sigma}_r$ has a symmetry group, the point $M_{(l,m,n)}$