

**Small Base Groups, Large Base Groups  
and the Case of Giants**

Jonathan Cohen

Submitted in partial fulfilment of the requirements for the degree of Bachelor of Computer  
and Mathematical Sciences with Honours.

ABSTRACT. Let  $G$  be a finite permutation group acting on a set  $\Gamma$ . A *base* for  $G$  is a finite sequence of elements of  $\Gamma$  whose pointwise stabiliser in  $G$  is trivial. Most (families of) finite permutation groups admit a base that grows very slowly as the degree of the group increases. Such groups are known as *small base* and very efficient algorithms exist for dealing with them. However, some families of permutation groups, such as the symmetric groups, do not admit a small base. Dealing with these so-called *large base* groups is a fascinating area of current research.

This thesis explores two closely interrelated strands of modern group theory. Initially, the focus is on identifying the large base primitive permutation groups, which can be achieved by making use of two landmark results in finite group theory: The Classification of Finite Simple Groups and the O’Nan-Scott Theorem for primitive permutation groups.

Focus then shifts to algorithmic aspects of large base groups, in particular to the family known as the *giants*. We cover details such as recognition of large base Galois groups, generation of random elements of finite groups and give details of the very new paradigm of algorithms for *black box groups*. We conclude with an investigation into the *constructive recognition* problem for large base black box groups.

2000 Mathematics Subject Classification: 20B05, 20B10, 20B15, 20B40.

## Contents

Acknowledgements	v
Chapter 1. Introduction	1
Chapter 2. Groups and Computation	3
1. Groups	3
2. Permutation Groups	4
3. Computation and Complexity	7
4. Monte Carlo and Las Vegas Algorithms	9
Chapter 3. Bases for Permutation Groups	13
1. Introduction	13
2. The Symmetric Group on Partitions	16
3. Elementary Results	20
4. Classification Results	24
Chapter 4. Recognising the Giants	31
1. Recognition Via Action	31
2. Black Box Groups	35
3. Random Elements of Finite Groups	39
4. Towards Constructive Recognition of Black Box Giants	43
5. Roots of Unity in the Symmetric Group	46
Chapter 5. Conclusion and Open Problems	53
Bibliography	54

## Acknowledgements

They say that gratitude is the most exquisite form of courtesy. Well, at least Jacques Maritain is quoted as having said that. I am grateful for the support and encouragement of many people, who have contributed to my “mathematical growth” this year. My supervisors Alice Niemeyer and John Bamberg put up with me for an entire year, certainly no mean feat. Their constant enthusiasm, willingness to let me explore whatever paths seemed interesting to me and ability to shoot down my more hair-brained ideas was greatly appreciated. I extend my thanks to Cheryl Praeger for organising funding for me to present some of this work at the 48th Annual Meeting of the Australian Mathematical Society and to Maska Law for proof reading this dissertation. Of course, I cannot forget the contribution of my parents, who convinced me (much to my dismay at the time) to take Ad. Maths all those years ago.

This year would have been far duller were it not for the encouragement to procrastinate provided by the inhabitants of Room 1.19 and 1.20: Alan, Ash, Betsy, Gemma, Geoff, Jared, Matt, Neil, Rhiannon and Stephen. Through good times (fantasy football victory) and bad (“The Comp” defeat), it’s been a blast.

Any split infinitives, dangling participles or symbol clashes that remain are, of course, no one’s fault but my own.

## CHAPTER 1

### Introduction

*“The proof uses various tools from analysis, number theory, probability theory and algebra; their participation in different portions seems to be characteristic for this theory.”*

– P. Erdős and P. Turán “On Some Problems of a Statistical Group Theory, I” [29].

Jurij Vega was a Slovenian mathematician who is best remembered for his book of logarithms, first published in 1794. It is over 600 pages in length and mainly contains the results of his hand calculations. These days, most people would consider such a publication an immense waste of time; the project of a madman. However, Vega’s tables served a useful purpose for hundreds of years, with nobody questioning the value of his work. So, what has changed in the last few decades? The answer is, of course, the invention of the digital computer.

Computers play an ever-increasing rôle in modern mathematics, allowing mathematicians to explore vast new areas whose very conception would have been unthinkable a mere few decades ago. Their entry into permutation group theory was heralded by the landmark papers of Charles Sims [55, 56]. For many years after the publication of these papers, the techniques presented therein were successively honed and refined until they ran blindingly fast on almost all classes of permutation groups. However, one class of permutation groups did not respond very well to the Sims-like methods. This class is known as the *large base permutation groups*.

For a long time, computational group theorists focused their efforts on developing algorithms for the so-called *small base* groups and considered computation with large base groups to be more or less intractable. This viewpoint began to change when László Babai and Endre Szemerédi demonstrated the power of randomised algorithm when applied to problems in group theory in [2].

Computational group theorists soon realised that randomised algorithms could be applied to computing with large base groups. In essence, they realised that if they could develop a fast algorithm for determining whether a given group is large base, then they could bypass most of the difficulties associated with using the Sims-like methods on these groups.

In Chapter 2, we introduce the theory of permutation groups, as well as the theory of randomised algorithms. This chapter should be used as indicated at its commencement. We then set about the task of recognising the large base groups. Of course, before we can even think about developing algorithms for recognising large base groups, we need to know a more

concrete description of these groups.

Chapter 3 explores the theory behind the problem of identifying which groups are large base, and which are small. It utilises a mixture of combinatorial and group-theoretic techniques in determining precisely what the large base primitive permutation groups are.

Armed with the knowledge from Chapter 3, we then set about developing algorithms for certain families of large base groups in Chapter 4 known as the giants. The algorithms are, for the most part, randomised. As such, their development and analysis incorporates a fascinating interplay between tools and techniques from probability theory, combinatorics, number theory, computational complexity and algebra.

**Contributions.** Chapter 3: All of the theorems have been recast and reordered so as to form a logical (rather than chronological) sequence and the proofs have been reworked and expanded from their original journal versions. Section 2 presents the author's own work and ideas and forms the basis for [18].

Chapter 4: Section 2.1 is the author's own work and ideas. Section 5 is the author's own work, though it is heavily based on the ideas of others.

**Conventions.** As each problem or tool is introduced in this thesis, we place it in its correct historical context, covering many results that contribute to the intrinsic interest of the problems and give valuable insight. Naturally, we do not have the space to cover the proofs of all of these results and so, where necessary, we have attempted to provide a reference to where they were first proven. A proof of a result is included if it is of central importance to our study, or if it gives great insight into the problem concerned.

The symbol  $\square$  denotes the end of a proof. If it appears at the end of the statement of a lemma or theorem, then this means that the proof is omitted but that a reference to a proof is given. We use “log” to denote the logarithm with base 2 and “ln” to denote the natural logarithm.

## CHAPTER 2

# Groups and Computation

This chapter collects together various topics that are required later in the dissertation. It is not expected that the reader will be able to remember all of the terms and results covered in this chapter for the duration of the dissertation. Therefore, it is recommended to read only those sections which are required for a chapter; and to do so immediately before reading the relevant chapter. We have tried to aid the reader in making an appropriate selection of topics from this chapter by explicitly mentioning the requirements at the beginning of each chapter. The reader should then feel free to refer back to this chapter whenever a definition or result slips her/his mind.

### 1. Groups

It is not our intention to give a complete introduction to group theory, as we assume that the reader has attended a basic first course in the subject. The reader who is unsure of the material in this section should consult an introductory text, such as [25]. In particular, we assume that the reader is familiar with the following terms and standard results: group, subgroup (denoted  $H \leq G$ ), abelian group, normal subgroup, quotient group, simple group, coset, Lagrange's Theorem, First Isomorphism Theorem.

**1.1. Morphisms.** If  $G$  and  $H$  are groups, then a map  $\varphi : G \rightarrow H$  is called a *homomorphism* if  $\varphi(ab) = \varphi(a)\varphi(b)$  for all  $a, b \in G$ . A bijective homomorphism is called an *isomorphism* and an isomorphism from a group to itself is called an *automorphism*. Two groups  $G$  and  $H$  are called *isomorphic*, denoted  $G \cong H$ , if there is an isomorphism between them. If  $\varphi : G \rightarrow H$  is a homomorphism, then the *kernel* of  $\varphi$ , denoted  $\ker(\varphi)$ , is the set  $\{g \in G : \varphi(g) = 1\}$ . A subgroup  $H$  of  $G$  is called *normal*, denoted  $H \trianglelefteq G$ , if it is the kernel of a homomorphism  $\varphi : G \rightarrow \hat{G}$  for some group  $\hat{G}$  or, equivalently, if  $gHg^{-1} = H$  for all  $g \in G$ .

**1.2. Symmetric and Alternating Groups.** If  $\Gamma$  is a finite set, then a *permutation* of  $\Gamma$  is a bijection from  $\Gamma$  to itself. The *symmetric group on  $\Gamma$* , denoted  $\text{Sym}(\Gamma)$ , is the group of all permutations of  $\Gamma$  under function composition. The notation  $S_n$  is equivalent to  $\text{Sym}(\{1, 2, \dots, n\})$  and it is easy to see that  $|S_n| = n!$ . A *cycle* is a string of integers which represents the element of  $S_n$  that cyclically permutes these integers. The cycle  $c := (a_1, a_2, \dots, a_m)$  is the element of  $S_n$  that sends  $a_i$  to  $a_{i+1}$  for  $1 \leq i < m$  and sends  $a_m$  to  $a_1$ . The set of points  $\{a_1, a_2, \dots, a_m\}$  is called the *support* of  $c$ . Two cycles are called disjoint if their supports are disjoint. It is easy to see that each element of  $S_n$  can be written as a product of disjoint cycles. A cycle of length two is called a *transposition*.

Each element of  $S_n$  can be written as either an odd number of transpositions, or an even number but not both (though, there may be more than one way in which to do this). If an element  $g \in S_n$  is a product of  $n$  transpositions, then the *sign* of  $g$  is  $(-1)^n$ . We define the map  $\varphi : S_n \rightarrow \{1, -1\}$ , which takes each element to its sign. It is easy to see that this mapping is a homomorphism and its kernel consists of all elements of  $S_n$  that can be written as a product of an even number of transpositions. This kernel is called the *alternating group on  $n$  points*, denoted  $A_n$ . In a similar manner, one can define the alternating group on an arbitrary set  $\Gamma$ , denoted  $\text{Alt}(\Gamma)$ .

## 2. Permutation Groups

The classical reference for permutation group theory is Wielandt [61], which remains a valuable resource today. More recent accounts can be found in the books by Dixon and Mortimer [24] and Cameron [16].

**2.1. Actions.** Given a group  $G$  and a finite set  $\Gamma$ , an *action* of  $G$  on  $\Gamma$  is a homomorphism  $\varphi : G \rightarrow \text{Sym}(\Gamma)$ . A *permutation group*,  $H$ , is the image of such a homomorphism and its *degree*, denoted  $\text{deg}(H)$ , is the size of  $\Gamma$ . If  $\alpha \in \Gamma$  and  $g \in G$ , then we use  $\alpha^g$  to denote  $(\varphi g)(\alpha)$  and call  $\alpha^g$  the *image* of  $\alpha$  under  $g$ . Moreover, we use  $\alpha^G$  to denote the set  $\{\alpha^g : g \in G\}$ , which is called the *orbit of  $\alpha$  under  $G$* . Similarly, if  $\Delta \subseteq \Gamma$ , then  $\Delta^g$  is the set  $\{\alpha^g : \alpha \in \Delta\}$ .

If  $\Delta \subseteq \Gamma$ , then the *pointwise stabiliser* is defined to be  $G_{(\Delta)} := \{g \in G : \alpha^g = \alpha \text{ for all } \alpha \in \Delta\}$ . The pointwise stabiliser of a sequence of points  $(\alpha_1, \alpha_2, \dots, \alpha_k)$  in  $G$  is defined by

$$G_{(\alpha_1, \alpha_2, \dots, \alpha_k)} = (G_{(\alpha_1)})_{(\alpha_2, \dots, \alpha_k)}$$

(that is, stabilise each point in turn). If  $\Delta \subseteq \Gamma$ , then the *setwise stabiliser* of  $\Delta$  in  $G$  is the subgroup,  $G_\Delta$ , of  $G$  consisting of the elements  $\{g \in G : \Delta^g = \Delta\}$ . The following standard result may be seen as a permutation group version of Lagrange's Theorem.

**THEOREM 2.1 (Orbit-Stabiliser Theorem).** *Let  $G$  be a permutation group acting on a set  $\Gamma$ . Then, for each  $\alpha \in \Gamma$ , we have that  $|G| = |\alpha^G| \cdot |G_{(\alpha)}|$ .*

**PROOF.** We omit the proof. See, e.g., [24, Theorem 1.4A]. □

**2.2. Transitivity.** A permutation group  $G$  acting on a set  $\Gamma$  is called *transitive* if it has precisely one orbit on  $\Gamma$ . This is equivalent to requiring that for each  $\alpha_1, \alpha_2 \in \Gamma$ , there is some  $g \in G$  such that  $\alpha_1^g = \alpha_2$ . If  $G$  is not transitive, then one easily sees that it induces a partition of  $\Gamma$  into disjoint subsets  $\Gamma_1, \Gamma_2, \dots, \Gamma_k$ , where  $G$  induces a transitive group  $G^{\Gamma_i}$  on each  $\Gamma_i$ . Each  $\Gamma_i$  corresponds to a different orbit of  $G$  and  $G^{\Gamma_i}$  is called a *transitive constituent* of  $G$ . Every intransitive permutation group is a “subcartesian product” of its transitive constituents (we do not provide details of this construction, which may be found in [16]). However, in general, a group cannot be represented in a unique way as a subcartesian product.

A permutation group  $G$  acting on a set  $\Gamma$  is said to be  *$k$ -fold transitive* if it acts transitively on the set of ordered  $k$ -tuples of distinct elements of  $\Gamma$ , where the action is given componentwise

by  $(\alpha_1, \alpha_2, \dots, \alpha_k)^g = (\alpha_1^g, \alpha_2^g, \dots, \alpha_k^g)$ , for each  $g \in G$ . An equivalent requirement is that  $G$  is  $k$ -fold transitive if it is transitive on  $\Gamma$  and  $G_{(\alpha)}$  is  $(k - 1)$ -fold transitive on  $\Gamma \setminus \{\alpha\}$  for any  $\alpha \in \Gamma$ . We often call a 2-fold transitive group *doubly transitive*.

### Examples.

(1) It is easy to see that  $S_n$  is  $n$ -fold transitive on  $\{1, 2, \dots, n\}$  by using the definition in terms of  $n$ -tuples.

(2)  $A_n$  is  $(n - 2)$ -fold transitive on  $\{1, 2, \dots, n\}$ . In order to see this, suppose that we have stabilised fewer than  $n - 2$  points of  $\Gamma$ . Then, for each  $\alpha_1, \alpha_2, \alpha_3 \in \Gamma$ , the 3-cycle  $(\alpha_1, \alpha_2, \alpha_3)$  is in  $A_n$ . That is,  $A_n$  acts transitively on the remaining points. However, if we stabilise  $n - 2$  points, then no element moves the remaining 2 points as there are no transpositions in  $A_n$ .

**2.3. (Semi)regularity.** A permutation group  $G$  on a set  $\Gamma$  is said to be *semiregular* if  $G_{(\alpha)} = \{1\}$  for any  $\alpha \in \Gamma$  and is said to be *regular* if it is semiregular and transitive. It can be shown that, if  $G$  is regular on  $\Gamma$ , then  $\Gamma$  is in bijective correspondence with the set of right cosets in  $G$  of the trivial subgroup, each member of which is a 1-element set. Therefore, we can identify the coset space with  $G$ , on which  $G$  acts by right multiplication:  $x^g = xg$ . This is called the *right regular representation of  $G$* .

**2.4. Primitivity.** Let  $G$  be a permutation group acting transitively on a set  $\Gamma$ . A nonempty subset  $\Delta$  of  $\Gamma$  is called a *block* if, for all  $g \in G$ , either  $\Delta^g = \Delta$  or  $\Delta \cap \Delta^g = \emptyset$ . It is easy to see that any group acting transitively has  $\Gamma$  and the singletons  $\{\{\alpha\} : \alpha \in \Gamma\}$  as blocks. We call these blocks *trivial* and any other block *nontrivial*. A transitive group  $G$  acting on a set  $\Gamma$  is said to be *primitive* if it has no nontrivial blocks on  $\Gamma$ . If there are nontrivial blocks, then we say that  $G$  is *imprimitive* and call the blocks *blocks of imprimitivity*. There is a relationship between primitivity and multiple transitivity, given by the following result.

**THEOREM 2.2.** *A doubly transitive group is primitive.*

**PROOF.** Let  $G$  be a doubly transitive group on  $\Gamma$  and suppose, without loss of generality, that  $\Gamma = \{1, 2, \dots, n\}$ , with  $n \geq 3$  and let  $\Delta$  be a block containing 1 and 2 and not containing  $n$ . Now,  $G_{(1)}$  acts transitively on  $\Gamma \setminus \{1\}$  so, in particular, there is some  $g \in G_{(1)}$  such that  $2^g = n$ . But then  $\Delta^g \neq \Delta$  and  $\Delta \cap \Delta^g \neq \emptyset$ , contradicting the choice of  $\Delta$ .  $\square$

Unfortunately, the converse of the last theorem is not true. For example, let  $G = \langle (1, 2, 3, 4, 5) \rangle$ . Then,  $G$  acts primitively and regularly, so the stabiliser of any point is trivial (and, in particular, not transitive). This leads us to define a *simply primitive* group to be a group that is primitive but not doubly transitive.

**2.5. Wreath Products.** Informally, wreath products may be thought of as a way of “gluing together permutation groups via their actions”. We make this more precise as follows. Let  $C$  be a group and  $D$  a permutation group acting on a set  $\Delta$ . Let  $C^\Delta$  denote the *direct product* of  $|\Delta|$  copies of  $C$ . The *wreath product*<sup>1</sup>  $C \wr D$  is the group  $G$  constructed from  $C^\Delta$  and  $D$  in the following way. The underlying set of  $G$  is the set consisting of the ordered pairs of elements

<sup>1</sup>Some authors denote this by  $C \wr D$ .

from  $C^\Delta$  and  $D$  (that is,  $G = C^\Delta \times D$ ). We write elements of  $G$  as  $(c, d)$ , where  $c \in C^\Delta$  and  $d \in D$  and define a binary operation on  $G$  as follows:

$$(c, d) \cdot (c', d') = (c(c')^{d^{-1}}, dd'),$$

for all  $(c, d)$  and  $(c', d')$  in  $C^\Delta \times D$ ; where  $(c')^{d^{-1}}$  is the element of  $C^\Delta$  obtained by permuting the coordinates of  $c'$  under the permutation  $d^{-1}$ . One then checks that  $G$  is indeed a group under this binary operation.

We call  $C^\Delta$  the *bottom group* of the wreath product<sup>2</sup> and  $D$  the *top group* of the wreath product. One may identify  $C^\Delta$  with the set of functions  $f : \Delta \rightarrow C$ . Then, we have that  $d^{-1}fd(\delta) = f(\delta^{d^{-1}})$  for  $d \in D$  and  $\delta \in \Delta$ . Suppose that  $C$  is a permutation group acting on a set  $\Gamma$ . Then, there are two different actions of  $C \wr D$  that we make use of in this thesis.

(1) *The imprimitive action on  $\Gamma \times \Delta$ .* The bottom group acts on the first coordinate by the rule  $(\gamma, \delta)^f = (\gamma f(\delta), \delta)$  and the top group acts naturally on the second coordinate. As one may have surmised from the name, this action is imprimitive if  $|\Gamma|$  and  $|\Delta|$  are both greater than 1, because the elements of  $\bigcup_{\delta \in \Delta} \Gamma \times \{\delta\}$  are nontrivial blocks of the wreath product on  $\Gamma \times \Delta$ . The following result is of fundamental importance, though we omit the proof, which may be found in, for example, [24, Theorem 2.6A].

**THEOREM 2.3 (Universal Embedding Theorem).** *Let  $G$  act transitively but imprimitively on  $\Lambda$ . Let  $\Gamma$  be a block of imprimitivity and let  $\Delta$  be the set  $\{\Gamma^g : g \in G\}$  of translates of  $\Gamma$ . Furthermore let  $C = (G_\Gamma)^\Gamma$  be the group induced on  $\Gamma$  by its setwise stabiliser  $G_\Gamma$  and let  $D = G^\Delta$  be the group induced on  $\Delta$  by  $G$ . Then, there is a bijection between  $\Lambda$  and  $\Gamma \times \Delta$  which embeds  $G$  into the wreath product  $C \wr D$ , where the wreath product is in its imprimitive action.  $\square$*

Therefore, if  $G$  is imprimitive, it is “built up” from smaller groups. If  $\Lambda$  is finite, then we may continue this process until we obtain primitive components for  $G$ . However, much like what happened with decomposing a non-transitive group into its transitive constituents, we “lose information” about  $G$  in the process. In particular, making different choices when carrying out the decomposition may result in different primitive components (see Cameron [14]).

(2) *The product action of  $C \wr D$  on the set  $\Gamma^\Delta$  of functions from  $\Delta$  to  $\Gamma$ ,* where the bottom group acts coordinatewise and the top group permutes the coordinates. More formally, for  $\phi \in \Gamma^\Delta$  and  $f \in C^\Delta$ , we set  $(\phi f)(\delta) = \phi(\delta)f(\delta)$  and for  $\phi \in \Gamma^\Delta$  and  $d \in D$ , we set  $(\phi d)(\delta) = \phi(\delta k^{-1})$ .

**THEOREM 2.4.** *Suppose that  $D$  is transitive on  $\Delta$  and  $C$  is primitive but not regular on  $\Gamma$ . Then, the product action of  $C \wr D$  on  $\Gamma^\Delta$  is primitive.*

**PROOF.** See [24, Lemma 2.7A].  $\square$

<sup>2</sup>It is more standard to call this the “base” group, but we refrain from doing so in order to prevent confusion with the main object of study in this thesis.

In particular, we have that the wreath products  $A_m \wr S_n$  and  $S_m \wr S_n$  are primitive. These groups will reappear throughout the following chapter.

### 3. Computation and Complexity

What is computation? For that matter, what is a computer? Such questions are likely to be met with looks of disbelief in this era of personal computers. The answer, however, is far from trivial.

In attempting to provide a complete answer, one would need to provide a rigorous definition of what it means *to compute*; and what, indeed, a computational problem is. In particular, is every computational problem solvable, in finite time, by some sort of computation? David Hilbert, one of the most influential mathematicians of the late nineteenth and early twentieth centuries, asked for such a characterisation, though he couched his question in the language of logic. His *Entscheidungsproblem*, or “decision problem”, simply asked whether or not there is a procedure which, when presented with a formula of first order logic, determines whether or not it is a theorem of first order logic in *finite time*. This question led the English mathematician Alan Turing to define a formal notion of computation, which has come to be known as a “Turing Machine” in his honour. Details of this construction, and indeed of most of this section, may be found in Papadimitriou [47]. The answer to Hilbert’s question, as proven by Turing, is a resounding no.

The requirement that a computation run in *finite* time is essential for Hilbert’s problem. In particular, first order logic is “recursively enumerable”, which may be taken to mean that there is an algorithm that lists every theorem of first order logic, if it is allowed to run for an arbitrary length of time. The word “algorithm” requires some explanation, though the majority of modern scientists would at least have an intuitive feel for what it means.

We do not provide a formal definition of an algorithm here, a development of such a definition could easily form the content of an entire thesis. Before providing an informal definition, it is incumbent upon us to provide a definition of a computational problem, which we do below.

By an *alphabet*, we mean a finite or countably infinite set of symbols, which we denote by  $\mathcal{A}$ . A *string* of length  $k$  over some alphabet  $\mathcal{A}$  is an element of the cartesian product  $\mathcal{A}^k$ . A *language*,  $\mathcal{L}$ , over  $\mathcal{A}$  is a subset of the set of all strings over  $\mathcal{A}$  (of any length). We define a *computational problem* to be a binary relation on  $\mathcal{L}$ ; that is, a subset of  $\mathcal{L} \times \mathcal{L}$ . If  $\mathcal{R}$  is a computational problem, then we write  $\mathcal{R}(x, y)$  to mean  $(x, y) \in \mathcal{R}$  and refer to  $x$  as the *input string* and to  $y$  as the *output string*.

Let  $\mathcal{R}$  be a computational problem. If the output of  $\mathcal{R}$  has only two possibilities (“yes” or “no”), then we refer to  $\mathcal{R}$  as a *decision problem*, otherwise we refer to  $\mathcal{R}$  as a *function problem*. If  $\mathcal{R}$  is a decision problem over the alphabet  $\mathcal{A}$ , then we refer to the set of all strings  $x$  over  $\mathcal{A}$

such that  $\mathcal{R}(x, \text{“yes”})$  as the *valid language for*  $\mathcal{R}$ .

A *deterministic* algorithm for a computational problem  $\mathcal{R}$  is an *explicit, finite* sequence of instructions which transforms the string  $x$  into the string  $y$ , whenever  $\mathcal{R}(x, y)$ . Moreover, we require that the algorithm produces the same output whenever it is run on a specific input.

In contrast, a *nondeterministic* algorithm may make arbitrary (but finite in number) choices as it works and, in general, produces different outputs when run on the same input. We say that a nondeterministic algorithm  $\Lambda$  is correct for a computational problem  $\mathcal{R}$  if, whenever  $x$  and  $y$  are strings such that  $\mathcal{R}(x, y)$ , *some* possible computation of  $\Lambda$  on  $x$  results in an output of  $y$ .

We are only concerned with algorithms that run in finite time. However, merely knowing that an algorithm runs in finite time is too coarse. Fortunately, there are natural ways to measure the amount of time that an algorithm takes to run.

We denote by  $\mathbb{N}$  the set of non-negative integers and by  $\mathbb{R}$  the set of real numbers. Let  $f$  and  $g$  be two functions from  $\mathbb{N}$  to  $\mathbb{R}$ . We write  $f(n) = O(g(n))$  (pronounced “ $f(n)$  is big-oh of  $g(n)$ ”) if there are positive integers  $c$  and  $n_0$  such that, for all  $n \geq n_0$ , the inequality  $f(n) \leq c \cdot g(n)$  holds. Informally,  $f(n) = O(g(n))$  means that  $f(n)$  grows as fast as  $g(n)$  or slower. We write  $f(n) = \Omega(g(n))$  (pronounced “ $f(n)$  is big-omega of  $g(n)$ ”) if the opposite happens, that is, if  $g(n) = O(f(n))$ . Informally, this means that  $f(n)$  grows as fast as  $g(n)$ , or faster. Finally,  $f(n) = \Theta(g(n))$  (pronounced “ $f(n)$  is big-theta of  $g(n)$ ”) means that  $f(n) = O(g(n))$  and  $f(n) = \Omega(g(n))$ . Informally, this means that  $f(n)$  grows at precisely the same rate as  $g(n)$ .

Clearly, if  $p(n)$  is a polynomial of degree  $d$ , then  $p(n) = O(n^d)$ . Moreover, if  $c > 1$  and  $p(n)$  is any polynomial, then  $p(n) = O(c^n)$ , but  $p(n) \neq \Omega(c^n)$ . In other words, any polynomial grows strictly slower than any exponential. Similarly,  $\log^k(n) = O(n)$  for any power  $k$ , but  $\log^k(n) \neq \Omega(n)$ . That is, any power of  $\log(n)$  grows strictly slower than  $n$ .

If an algorithm is presented with an input string of length  $n$ , then one can find functions  $f : \mathbb{N} \rightarrow \mathbb{R}$  and  $g : \mathbb{N} \rightarrow \mathbb{R}$  such that the algorithm has a running time of  $O(f(n))$  and a space requirement of  $O(g(n))$  and similarly for the other measures of complexity. The time requirement for a nondeterministic algorithm is defined to be the time required for an accepting computation (that is, one which returns the correct output) and *not* the requirement for all possible computations; similarly for the space requirement.

The class of all computational problems that admit a polynomial time *deterministic* algorithm is denoted by  $\mathbf{P}$  and the class of all computational problems which admit a polynomial time *nondeterministic* algorithm is denoted by  $\mathbf{NP}$ . Upon a little reflection, one may convince oneself that  $\mathbf{P} \subseteq \mathbf{NP}$ . Whether or not this inclusion is strict is one of the most important open

problems in mathematics, though it is unlikely that  $\mathbf{P} = \mathbf{NP}$ . One reason for this, is that  $\mathbf{P}$  is a *useful* complexity class, in that if we possess a polynomial time deterministic algorithm for a problem, then we can solve it relatively efficiently. However, if all we have is a polynomial time nondeterministic algorithm, then we have no way of knowing how many times we need to run the computation on an input until we are sure that the output is correct. Later in this chapter, we introduce several stronger models of nondeterminism, which allow one to define a nondeterministic algorithm relative to a given problem in a sensible manner.

**3.1. Repeated Squaring.** We now illustrate how complexity measures are used in practice by way of an example. Suppose that we have natural numbers  $a$  and  $n$  and that we wish to compute  $a^n$ . The naïve way in which to do this is to just compute  $\prod_{i=1}^n a$ . It is easy to see that this method has a time complexity of  $\Theta(n)$ . With a bit of reflection, one can see that we can, in fact, do better than this. Let  $(b_k, b_{k-1}, \dots, b_1)$  be the binary representation of  $n$ . Then, we have that:

$$a^n = a^{(b_k 2^{k-1} + b_{k-1} 2^{k-2} + \dots + b_1)} = a^{b_k 2^{k-1}} a^{b_{k-1} 2^{k-2}} \dots a^{b_1}.$$

We extend this observation into an efficient method for exponentiation, presented in Algorithm 1.

---

**Algorithm 1:** Repeated Squaring

---

**Input:** Natural numbers  $a$  and  $n$

**Output:**  $a^n$

**begin**

    Set  $k := \log(n)$

    Set  $b := [b_1, b_2, \dots, b_k]$ , the binary representation of  $n$

    Compute  $\Gamma := [a, a^2, \dots, a^{2^{k-1}}]$

    Initiate  $x := 1$

**for**  $i := 1$  **to**  $k$  **do**

**if**  $b[i] = 1$  **then**

$x := x * \Gamma[i]$

**return**  $x$

**end**

---

Algorithm 1 is known as *repeated squaring*, as each term in  $\Gamma$  is the square of the previous term. Noting that  $\Gamma$  has  $\log(n)$  entries and that we multiply at most  $\log(n)$  elements of it together, we obtain that the running time of the repeated squaring algorithm is  $O(\log(n) + \log(n)) = O(\log(n))$ . As the sequence of values of  $\{\log(n)\}_{n \in \mathbb{N}}$  grows far slower than  $\{n\}_{n \in \mathbb{N}}$ , the repeated squaring algorithm is far more efficient than the naïve method. Finally, noting that we did not actually use the fact that  $a$  is a natural number, we have established:

LEMMA 3.1. *Given an object,  $x$ , for which multiplication by itself makes sense and a natural number  $n$ , one can compute  $x^n$  in  $O(\log(n))$  time.  $\square$*

## 4. Monte Carlo and Las Vegas Algorithms

In this section, we present two related models of nondeterministic computation, which allow us to apply nondeterministic algorithms to practical problems. The basic idea is to define

a very simple discrete probability measure on the set of choices presented at a nondeterministic state of the computation. In other words, we assign a nonnegative probability of selection to each choice in such a way that the probabilities sum to one. In fact, there is no loss of generality if we allow a nondeterministic algorithm to have only two choices at each state (one can number the choices and make a sequence of “coin flips” in order to decide the values in the binary representation of the choice number).

This is simplest model of *randomised computation* and it is not a very useful one. Suppose that we have some decision problem  $\mathcal{R}$  that we wish to solve with a randomised algorithm. In the current situation, the only requirement is that, for any input string in the valid language of  $\mathcal{R}$ , *some* sequence of coin flips results in the algorithm returning “yes”. In other words, we don’t have any bounds on the probability of a false positive or a false negative<sup>3</sup>, so any response is effectively useless.

A possible improvement is to add the requirement that, for any input string in the valid language of  $\mathcal{R}$ , more than half of the possible sequences of choices halt with “yes”. If we enforce the additional requirement that the algorithm has polynomial time complexity, then the class of all decision problems admitting such an algorithm is known as **PP**, for “probabilistically polynomial”. While one might hope that this would give us a more restrictive complexity class than **NP**, this is unfortunately not the case.

**THEOREM 4.1.**  $\mathbf{NP} \subseteq \mathbf{PP}$ .

**PROOF.** See [47, Theorem 11.3]. □

Furthermore, it turns out that **PP** is not a practically useful model of randomisation. Suppose that you are given a coin and told that it is biased in such a way that it comes up on one side with probability  $1/2 + \epsilon$  and on the other side with probability  $1/2 - \epsilon$ , but *you are not told which side has which probability*. Your task, then, is to use a sequence of coin flips in order to determine which side has the greater probability.

Being slightly more formal about the problem, let  $X_1, X_2, \dots, X_n$  be independent random variables taking the values 1 and 0 with probabilities  $p$  and  $1 - p$  respectively (these are our “coin flips”). Let  $X = \sum_{i=1}^n X_i$ . This is known as a *binomial random variable*. A result in probability theory, known as the Chernoff Bound (see [47, Page 258]), states that the probability that a binomial random variable deviates from its expectation decreases exponentially with the deviation (that is, with  $\epsilon$ ). In other words, we can detect a bias of  $\epsilon$  in the coin by taking the majority of approximately  $\frac{1}{\epsilon^2}$  experiments. In **PP**, however, this bias can be as low as  $2^{-p(n)}$ , where  $p(n)$  is a polynomial in the length,  $n$ , of the input string. Therefore, we would need to repeat the algorithm an exponential number of times in order to be reasonably confident that we have achieved the correct answer.

---

<sup>3</sup>A false positive is a reply of “yes” when the answer is “no” and vice versa for false negative.

We, therefore, strengthen the requirements on a randomised algorithm, and define our first “useful” model of randomised computation.

**DEFINITION 4.2 (Monte Carlo Algorithm).** *Let  $R$  be a decision problem and  $L$  be its corresponding valid language. Let  $\Lambda$  be a nondeterministic algorithm such that at each step of the algorithm, there are exactly two choices for the next state and that the number of steps in each computation on an input of length  $n$  is bounded by a polynomial,  $p(n)$  in  $n$ . Suppose, moreover, that for all  $x \in L$ , at least half of the  $2^{p(|x|)}$  possible computations of  $\Lambda$  on  $x$  halt with output “yes” and for  $x \notin L$  then all of the possible computations of  $\Lambda$  on  $x$  halt with output “no”. We then call  $\Lambda$  a Monte Carlo Algorithm.*

As we have defined it, a Monte Carlo algorithm produces no false positive answers, because for any string not in the valid language, rejection is unanimous. Moreover, since each step of the computation involves a random choice between two alternatives, each with probability  $1/2$ , each possible computation is equiprobable, with probability  $2^{-p(|x|)}$ . Therefore, the probability of a false negative is at most one half. However, the selection of  $1/2$  for the probability of acceptance was arbitrary. In order to see this, suppose that the probability was only  $\epsilon < 1/2$ . By repeating the algorithm  $k$  times, we ensure that the probability of a false negative is at most  $(1 - \epsilon)^k$ , which is at most  $1/2$  for large enough  $k$ .

The above discussion presents what we call a “Monte Carlo Yes” algorithm (that is, if it outputs “yes”, then it is guaranteed correct). By swapping “no” and “yes” above and changing “false negative” to “false positive”, it is possible to define a “Monte Carlo No” algorithm.

**DEFINITION 4.3 (**RP** and **coRP**).** *The class of all decision problems possessing a Monte Carlo Yes algorithm is denoted by **RP** (for Random Polynomial time) and the class of all decision problems possessing a Monte Carlo No algorithm by **coRP** (for the complemented<sup>4</sup> decision problems from **RP**).*

Note that there is an asymmetry in these classes, as one has no false positives and the other no false negatives. Therefore, it is worthwhile to consider the intersection **ZPP** := **RP**  $\cap$  **coRP** where **ZPP** stands for “**Z**ero **P**robability of error in **P**olynomial time”. A quick check of the definitions shows that **P**  $\subseteq$  **ZPP**. This leads to our second “useful” model of randomised computation.

**DEFINITION 4.4 (Las Vegas Algorithm).** *An algorithm for a decision problem is called Las Vegas if it is both Monte Carlo Yes and Monte Carlo No.*

In other words, **ZPP** is the class of all decision problems which possess a Las Vegas Algorithm. Roughly speaking, a Las Vegas algorithm always returns a correct answer, *however*, there is some probability (defined similarly as above), that the algorithm returns no output at all. Another (more practical) way in which to obtain a Las Vegas algorithm is to add a deterministic procedure to check the output of a Monte Carlo algorithm. Note, this does not mean that merely suppressing an output of “no” from a Monte Carlo Yes algorithm yields a

<sup>4</sup>That is, where “yes” and “no” are swapped around.

Las Vegas algorithm, *such an algorithm is still only Monte Carlo* (from the point of view of a user, if the algorithm does not return anything, then it is the same as if it returned “no”).

Some authors refer to *any* randomised algorithm as a Monte Carlo algorithm and call what we have defined a “one-sided Monte Carlo algorithm” (for example, Babai [6]). However, as we have seen, weaker models of randomisation are not very “useful”, so we only consider these “one-sided” algorithms in practice. Our presentation of these concepts has been more formal than is customary in computational group theory. The reason for this is that some authors coming from a group theory background appear to often confuse the different complexity classes, including using such terms as “one-sided Las Vegas algorithm” [13].

Although all of our definitions in this section have been in terms of decision problems, the notions extend naturally to function problems. In order to see this, we model the case that the algorithm produces an output as returning “yes”. One can then consider the construction of the actual output to be a deterministic procedure. Thus, in particular, it makes sense to speak about a “Monte Carlo algorithm for constructing  $X$ ”.

## CHAPTER 3

# Bases for Permutation Groups

This chapter requires Sections 1 and 2 of Chapter 2.

### 1. Introduction

Given a permutation group  $G$ , there are many basic questions that one might want to be able to answer quickly and efficiently. For example, finding the order  $|G|$  and testing to see whether a given permutation lies in  $G$ . The naïve way in which to do this would be to just store all the elements of  $G$ , thereby rendering the problems trivial. However, this is very inefficient. For example, for  $S_8$  in its natural action, this scheme would require us to store 40320 elements. We note that  $S_8$  acts on only 8 points so if we could somehow find a way of encoding a group in terms of its action, then we may be able to improve the efficiency of these procedures considerably. This is essentially what Charles Sims accomplished in [55, 56].

Recall the basic result that a linear transformation on a vector space  $V$  is uniquely determined by its image on a set of basis vectors for  $V$ . This gives us a good insight for reasoning about linear transformations. As we shall see, the following construct allows us to do something similar for permutation groups.

**DEFINITION 1.1 (Base).** *Let  $G$  be a permutation group acting on a set  $\Gamma$ . A base for  $G$  is a nonempty ordered subset  $\Sigma \subseteq \Gamma$  such that the pointwise stabiliser  $G_{(\Sigma)}$  is trivial.*

For the rest of this section,  $G$  denotes a permutation group acting on the set  $\Gamma = \{1, 2, \dots, n\}$ . Suppose that  $\mathcal{B} = [\alpha_1, \alpha_2, \dots, \alpha_k]$  is a base for  $G$ . Then,  $\mathcal{B}$  induces a chain of subgroups of  $G$ :

$$(1) \quad G = G^{(1)} \geq G^{(2)} \geq \dots \geq G^{(k)} \geq G^{(k+1)} = \{1\},$$

where for each  $i$  with  $1 \leq i \leq k + 1$ , we have that  $G^{(i)}$  is the pointwise stabiliser of the first  $i - 1$  elements of  $\mathcal{B}$ . The sequence (1) is known as a *stabiliser chain* for  $G$  relative to  $\mathcal{B}$ . We call a base *irredundant* if all of the inclusions in (1) are proper. Unfortunately, it is not true that the length of an irredundant base is an invariant for a group  $G$ . For example, if  $G = \langle (1, 2)(3, 4, 5, 6) \rangle$ , then the point stabiliser  $G_{(1)}$  is  $\langle (3, 5)(4, 6) \rangle$  but  $G_{(3)}$  is trivial; hence,  $[3]$  and  $[1, 3]$  are both irredundant bases for  $G$ .

Now, by the Orbit-Stabiliser theorem<sup>1</sup>, we have that

$$|G| = |\alpha_1^{G^{(1)}}| \cdot |G^{(2)}| = |\alpha_1^{G^{(1)}}| \cdot |\alpha_2^{G^{(2)}}| \cdot |G^{(3)}| = \cdots = \prod_{i=1}^k |\alpha_i^{G^{(i)}}|.$$

The Orbit-Stabiliser theorem also implies that the points in  $\alpha_i^{G^{(i)}}$  are in one-to-one correspondence with the cosets of  $G^{(i+1)}$  in  $G^{(i)}$ . Specifically, the point  $\alpha_i^g$  corresponds to the coset  $G^{(i)}g$ . Let  $R_i$  be a set of coset representatives, for each  $i$ . Then, there is a unique  $r_1 \in R_1$  such that  $g = G^{(2)}r_1$ . Noting that  $gr_1^{-1} \in G^{(2)}$  and continuing by induction, we can write  $g$  uniquely as a product  $r_1r_2 \dots r_k$  where each  $r_i \in R_i$ . Conversely, we clearly have that each such product is a member of  $G$ . We have, therefore, established the following lemma.

LEMMA 1.2. *Each element of a permutation group is uniquely determined by its action on a base.* □

Using bases and “strong generating sets” (which we do not require for this thesis), Sims developed algorithms for solving many problems related to permutation groups efficiently [55, 56]. For example, using these structures, one can represent  $S_8$  by at most 56 elements, a vast improvement on the naïve approach. These methods, though heavily updated, still form the foundation for many modern-day algorithms for computing with permutation groups (see [54]).

We do not consider strong generating sets again in this thesis and now shift our focus onto bases themselves.

**Examples** We construct bases for some well known groups.

(1) *Mathieu Group:*

$$G = \langle (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), (3, 7, 11, 8)(4, 10, 5, 6) \rangle.$$

This is commonly known as the Mathieu Group  $M_{11}$  acting on 11 points and is of interest because it is the smallest sporadic simple group (i.e., it does not belong to any infinite family of simple groups). One may verify by routine calculations that  $\mathcal{B} = [1, 2, 3, 4]$  forms a base for  $G$ , as

$$G^{(2)} = \langle (3, 7, 11, 8)(4, 10, 5, 6), (2, 6)(4, 9)(5, 11)(8, 10) \rangle$$

$$G^{(3)} = \langle (3, 7, 11, 8)(4, 10, 5, 6), (4, 7, 10, 6)(5, 9, 8, 11) \rangle$$

$$G^{(4)} = \langle (4, 7, 10, 6)(5, 9, 8, 11), (4, 8, 10, 5)(6, 11, 7, 9) \rangle$$

from whence  $G^{(5)} = 1$ .

(2) *Semiregular groups:* In this case, stabilising any point immediately yields the trivial group and, so, we may take a base to be  $\mathcal{B} = [1]$ .

(3) *Dihedral group on  $n$  points:* Recall that the dihedral group on  $n$  points,  $D_n$ , is the group of rotations and reflections which leave a regular polygon on  $n$  vertices invariant. A

<sup>1</sup>See the preliminaries chapter for a statement of this.

dihedral group can be generated by two elements, namely, rotation through an angle of  $2\pi/n$  radians and a reflection about a line passing through opposite vertices if  $n$  is even and a vertex and the opposite edge if  $n$  is odd. So, if we stabilise one point, we can no longer rotate the polygon but we may reflect it. Stabilising another point (not opposite the first stabilised point, if  $n$  is even) removes the ability to reflect and so,  $\mathcal{B} = [1, 2]$  is a base.

- (4) *Symmetric group acting on an  $n$ -element set:* As  $S_n$  is  $n$ -fold transitive, we clearly have that a base is  $\mathcal{B} = [1, 2, \dots, n-1]$ .
- (5) *Alternating group acting on an  $n$ -element set:* As  $A_n$  is  $(n-2)$ -fold transitive and contains no transpositions, we have that  $\mathcal{B} = [1, 2, \dots, n-2]$  is a base.

Each of the above examples, except for the first, gives us a base for an infinite family of groups. Furthermore, the lengths of the bases in the last two examples are dependent on the degree of the group involved. We are now able to define the main object of study in this thesis.

**DEFINITION 1.3 (Small/Large Base).** *Let  $\mathcal{G}$  be an infinite family of permutation groups. Then,  $\mathcal{G}$  is called small base if there are positive constants  $b$  and  $c$  such that each  $G \in \mathcal{G}$  admits a base of size  $b \log^c(\deg(G))$ . The family  $\mathcal{G}$  is called large base if it is not small base.*

We often abuse the definition and refer to individual groups as being small/large base where it is clear what infinite family they belong to. In the above examples, we have seen that  $S_n$  and  $A_n$  are large base, whereas  $D_n$  is small base. It should be reasonably clear that the requirement that the family be infinite is necessary, as we would be able to adjust the constants for any finite family of permutation groups so as to make them fit the definition of “small base” otherwise.

Clearly, any irredundant base can be enlarged by adding redundant points and we saw earlier that even the length of an irredundant base is not invariant. We reserve the notation  $b(G)$  to denote the size of a minimal base for  $G$ . The next result follows easily from Lemma 1.2.

**LEMMA 1.4.** *Let  $G$  be a permutation group of degree  $n \geq 2$  and  $b(G)$  be the minimal base size for  $G$ . Then  $2^{b(G)} \leq |G| \leq n^{b(G)}$ .  $\square$*

Historically, small base groups were seen as those that could be handled computationally. This viewpoint was so prevalent, that it led the authors in [7] to comment: “From a practical point of view, small base groups are the only groups with which one can effectively compute in the case of ‘very large’  $n$ ”. In the following chapter, we introduce some recent progress at computing with large base groups. However, before we can do that, we need to know a little more about the structure of permutation groups.

Our main goal in this chapter is to partition all finite permutation groups according to base size. That is, we wish to be able to say precisely which families are small base and which are large base. Lemma 1.4 suggests that this question is related to determining the order of a permutation group, as  $b(G) \leq \log(|G|)$ . To this end, much of our focus will be on bounding the orders of permutation groups.

The above problem remains open for general permutation groups. However, we saw in Chapter 2 that an intransitive permutation group is a subcartesian product of its transitive constituents and that a transitive but imprimitive group can be decomposed into primitive components. Therefore, our main focus throughout this chapter is on primitive groups, for which there are powerful tools that can be used to resolve the problem.

We use elementary methods in Section 3 to bound the orders of families of groups. Precisely what we mean by “elementary” will be made clear later in this chapter. While we can get quite far by these techniques, we will see that much deeper methods are required in order to complete the story.

Section 4 provides an overview of some of the deepest results of finite group theory before utilising them for our problem. The results of this chapter require substantially deeper properties of finite groups and the proofs are, correspondingly, more technical than the elementary results. This culminates in the main result of this chapter, Theorem 4.11, which finishes the story for primitive permutation groups.

Before dealing with more general results, we investigate the minimal base size of a certain action of the symmetric group. The importance of the obtained bound is apparent in light of Theorem 4.11, where this particular action of the symmetric group is conspicuous by its absence.

## 2. The Symmetric Group on Partitions

A *partition* of a set  $\Gamma$  is a collection of nonempty pairwise-disjoint subsets of  $\Gamma$  whose union is all of  $\Gamma$ . The sets which constitute a partition of a set are usually referred to as *blocks*. This is in agreement with the notion of a “block of imprimitivity” when one considers the action of a group. Specifically, let  $a$  and  $b$  be positive integers and let  $\Gamma = \{1, 2, \dots, ab\}$ . Let  $\mathbb{P}$  denote the collection of all partitions of  $\Gamma$  into  $a$  blocks of size  $b$ . The symmetric group  $S_{ab}$  induces a natural action  $\varphi : S_{ab} \rightarrow \text{Sym}(\mathbb{P})$  such that  $G := \varphi(S_{ab})$  acts primitively on  $\mathbb{P}$ .

Now, let  $\mathcal{P} \in \mathbb{P}$  and consider the pointwise stabiliser  $G_{(\mathcal{P})}$ . What does this group look like? Since it has to fix  $\mathcal{P}$  as a partition, for each set  $\Delta \in \mathcal{P}$  and each  $g \in \varphi^{-1}(G_{(\mathcal{P})})$ , either  $\Delta^g = \Delta$ , or  $\Delta \cap \Delta^g = \emptyset$ . That is, the blocks of  $\mathcal{P}$  form blocks of imprimitivity for  $\varphi^{-1}(G_{(\mathcal{P})})$ . Abstractly, it is not hard to see that  $G_{(\mathcal{P})}$  is the wreath product  $S_b \wr S_a$  in its imprimitive action of degree  $ab$ .

Our goal in this section is to provide effective bounds on the minimal base size  $b(G)$ . Liebeck [39] previously showed that  $b(G) < (a-1)(b-1) + 2$ . We provide a far sharper bound, by making use of a new approach to the problem. The work in this section is the author’s own and forms the basis for [18].

We introduce a different representation for a partition  $\mathcal{P}$ . Specifically, we create a  $b \times a$  matrix  $M$  whose columns correspond to blocks of  $\mathcal{P}$ , while each block of  $\mathcal{P}$  corresponds to a column of  $M$ . Clearly, there is more than one way in which to do this, as partitions are unordered, whereas matrices are ordered. However, it is equally easy to see that a matrix representation of a partition uniquely determines a partition. Therefore, it makes sense to speak about a group element  $g \in G$  “fixing”  $M$ , meaning that  $g$  fixes the underlying partition of  $M$ . We shall have need to speak about various pieces of  $M$  and so introduce some notation.  $M(i, j)$  denotes the point appearing in row  $i$  and column  $j$  of  $M$ . If  $g \in G$ , then we use  $M(i, j)^g$  to denote the image of  $M(i, j)$  under  $g$ . Furthermore,  $\text{Row}(M, i)$  denotes the set of points corresponding to the  $i^{\text{th}}$  row of  $M$  and  $\text{Col}(M, j)$  denotes the set of points corresponding to the  $j^{\text{th}}$  column of  $M$ . For  $g \in G$ , the notation  $\text{Row}(M, i)^g$ ,  $\text{Col}(M, j)^g$  and  $M^g$  are defined in a similar way to  $M(i, j)^g$ .

Before beginning our analysis of  $b(G)$ , we introduce three ways of constructing a new (different) partition from an existing one. With this in mind, let  $M$  be the matrix representation of a partition of  $\Gamma$  into  $a$  parts of size  $b$ . The constructions are as follows:

- (i) “Transpose”. This is only defined when  $a = b$ . The transpose of  $M$ , denoted  $M^t$ , is obtained by making a copy of  $M$  and then interchanging the elements corresponding to  $M(i, j)$  and  $M(j, i)$  for each  $i, j \in \{1, 2, \dots, b\}$ . Intuition: this is the standard matrix transpose.
- (ii) “Flip”. This is only defined when  $a = b$ . The flip of  $M$ , denoted  $M^f$ , is obtained by making a copy of  $M$  and then interchanging the elements corresponding to  $M(i, i)$  and  $M(i, 1)$  for each  $i \in \{1, 2, \dots, b\}$ . Intuition: interchange the first column of  $M$  with the main diagonal.
- (iii) “Cycle  $i$ ”. This is defined for all  $a$  and  $b$ . The cycle  $i$  of  $M$ , denoted  $M^{c_i}$ , is obtained by making a copy of  $M$  and then sending the point corresponding to  $M(i, j)$  to  $M(i, j+1)$ , for each  $j \in \{1, 2, \dots, a\}$ , where  $b+1$  is taken to be 1. Intuition: cycle the  $i^{\text{th}}$  row of  $M$  once.

These constructions are illustrated in Figure 1 for the case where  $a = b = 3$ .

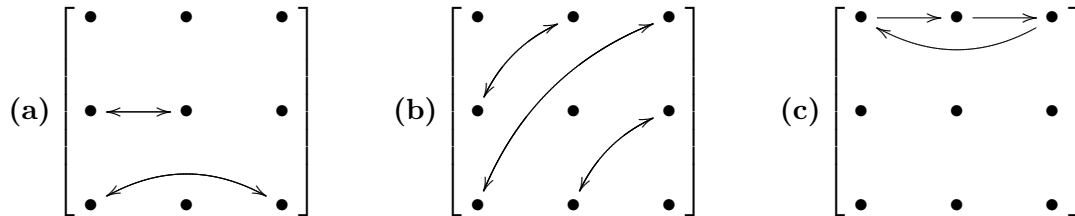


FIGURE 1. The basic operations when  $a = b = 3$ . Double headed arrows represent transpositions: (a) “Flip” (b) “Transpose” (c) “Cycle 1”.

Initially, we concentrate on the case where  $a = b$ , as the analysis for the other cases will proceed largely by iterating the construction used in this case. Note that it is possible to carry out two (or more) of the constructions in succession, in order to obtain one new partition. For

instance,  $M^{tf}$  corresponds to first transposing  $M$  and then flipping  $M^t$ .

LEMMA 2.1. *Let  $G$  be the group induced by the action of  $S_{n^2}$  on the collection of all partitions of  $\{1, 2, \dots, n^2\}$  into  $n$  parts of size  $n$ , where  $n$  is at least 3. Let  $M$  be a matrix representation of such a partition. Then only the trivial permutation both moves no columns of  $M$  and fixes each entry of  $[M, M^t, M^{tf}]$ .*

PROOF. The proof is by ‘‘point chasing’’. If  $g \in S_{n^2}$  is nontrivial, then we must have  $M(i, j)^g = M(k, j)$  for some  $i \neq k$ . But then  $M^t(j, i)^g = M^t(j, k)$ , so we must have that  $\text{Col}(M^t, i)^g = \text{Col}(M^t, k)$ .

If  $i \neq 1$ , then  $M^{tf}(i, i)^g = M^t(i, 1)^g$ , and  $M^t(i, 1)^g$  is not in column  $k$  of  $M^t$ . However, for  $\ell \neq i$ , we have that  $M^{tf}(\ell, i)^g = M^t(\ell, i)^g$ , and  $M^t(\ell, i)^g$  is in column  $k$  of  $M^t$ . So,  $g$  does not fix  $M^{tf}$ . If  $i = 1$ , then it is easy to see that  $g$  does not fix  $M^{tf}$  by considering the point that gets sent to  $M^t(k, k)$ . So, if  $g$  fixes  $[M, M^t, M^{tf}]$  and does not move any columns of  $M$ , then  $g$  is trivial.  $\square$

Before continuing, we need a new construction. If  $M$  is a partition of  $\{1, \dots, ab\}$  into  $a$  blocks of size  $b$ , then the partition  $M^{c1s}$  is obtained by making a copy of  $M^{c1}$  and interchanging the elements corresponding to  $M^{c1}(2, 1)$  and  $M^{c1}(2, 3)$ .

LEMMA 2.2. *Let  $M$  be a matrix representation of a partition of  $\{1, 2, \dots, ab\}$  into  $a$  blocks of size  $b$ , where  $a$  and  $b$  are both at least 3.*

- (i) *If  $g \in S_{ab}$  fixes  $[M, M^{c1}]$ , then it fixes  $\text{Row}(M, 1)$  setwise.*
- (ii) *Moreover, no permutation that moves columns of  $M$  fixes  $[M, M^{c1s}]$ .*

PROOF. Part (i). Let  $g$  be as in the statement of the lemma and suppose that it does not fix  $\text{Row}(M, 1)$  setwise. It is now easy to perform a point chase to show that  $g$  does not fix the partitions. There are two cases to consider, according to whether a point from  $\text{Row}(M, 1)$  is sent to a point in the same column or a different column. We only consider the first case, the second being largely similar.

Suppose that  $M(1, i)^g = M(j, i)$ , where  $j \neq 1$ . This, in particular, means that  $\text{Col}(M, i)$  is not moved by  $g$ . Then,  $M^{c1}(1, i+1)^g = M^{c1}(j, i)$ , where  $n+1$  is taken to be 1. Since  $g$  fixes the underlying partition of  $M^{c1}$ , we must have that  $\text{Col}(M^{c1}, i+1)^g = \text{Col}(M^{c1}, i)$ . In particular, there are some  $\ell, t \neq 1$  such that  $M^{c1}(t, i+1)^g = M^{c1}(\ell, i)$ . This implies that  $M(t, i+1)^g = M(\ell, i)$ . Since  $g$  fixes  $M$ , we must have that  $\text{Col}(M, i+1)^g = \text{Col}(M, i)$ . But this is impossible, since  $g$  does not move  $\text{Col}(M, i)$ . So, we have established that if  $g \in S_{ab}$  fixes  $[M, M^{c1}]$  then it fixes  $\text{Row}(M, 1)$  setwise.

Part (ii). Suppose that  $M(1, i)^g = M(1, j)$ . Then,  $M^{c1}(1, i+1)^g = M^{c1}(1, j+1)$ . Since  $g$  fixes  $M^{c1}$ , we have that  $\text{Col}(M^{c1}, i+1)^g = \text{Col}(M^{c1}, j+1)$ , where we again take  $n+1$  to be 1. So, we must have that  $\text{Col}(M, i+1)^g = \text{Col}(M, j+1)$ . Continuing in this manner, we see that  $g$  has to ‘‘cyclically permute’’ the columns of  $M$ . That is,  $\text{Col}(M, i+k)^g = \text{Col}(M, j+k)$ , where

a column number  $\ell > n$  is taken to be  $\ell - \lfloor \ell/n \rfloor$ . Note that  $\text{Row}(M^{c_1}, 1) = \text{Row}(M^{c_1^s}, 1)$ . It is then easy to see that the same must be true for any permutation fixing  $[M, M^{c_1^s}]$  by considering the points in  $M^{c_1^s}$  other than those corresponding to  $M^{c_1^s}(2, 1)$  and  $M^{c_1^s}(2, 3)$ . However, by subsequently considering these points, we see that no such permutation fixes  $[M, M^{c_1^s}]$ .  $\square$

The following result is a consequence of combining the previous two lemmas.

**LEMMA 2.3.** *Let  $G$  be the group induced by the action of  $S_{n^2}$  on the collection of all partitions of  $\{1, 2, \dots, n^2\}$  into  $n$  blocks of size  $n$  and let  $M$  denote the matrix representation of such a partition. Then, only the trivial permutation fixes  $\mathcal{B} = [M, M^t, M^{tf}, M^{c_1^s}]$ . That is,  $b(G) \leq 4$ .  $\square$*

It is now possible to make the jump to the case where  $M$  is non-square. By now, the reader should have realised that the difficulty in the results is not so much the proofs themselves, as coming up with suitable partitions. Thus, before we continue, we cover some new constructions that are used in the proof of the main result.

Firstly, the previous results were in terms of square matrices, so we need a way in which to access special square matrices. Let  $G$  once more denote the collection of partitions of  $\{1, 2, \dots, ab\}$  into  $a$  blocks of size  $b$ .

Suppose, firstly, that  $a > b$  and that  $b$  divides  $a$ . Then, we can “slice” up  $M$  into  $[S_1, S_2, \dots, S_k]$  where  $k = a/b$  and each  $S_i$  is a  $b \times b$  matrix corresponding to columns  $(i-1)b+1$  through  $ib$  of  $M$ . We then define  $M^t = [S_1^t, S_2^t, \dots, S_k^t]$  and  $M^{tf} = [S_1^{tf}, S_2^{tf}, \dots, S_k^{tf}]$ . If  $b$  does not divide  $a$ , then we can slice up  $M$  into  $[S_1, S_2, \dots, S_k, R]$  where  $k = \lfloor a/b \rfloor$ , each  $S_i$  is as before and  $R$  consists of columns  $b\lfloor a/b \rfloor + 1$  through  $a$  of  $M$  (that is, the rectangular matrix that does now fit into any of the squares). The partitions  $M^t$  and  $M^{tf}$  are defined in terms of the  $S_i$  in a similar manner to before, with the  $R$  being left unchanged.

If  $a < b$ , then we can split up  $M$  into  $[S_1, S_2, \dots, S_k, R]$  where  $k = \lfloor b/a \rfloor$ , each  $S_i$  corresponds to rows  $(i-1)a+1$  through  $ia$  of  $M$  and  $R$  consists of the rows that do not fit into any of the squares. We define  $M^t$  and  $M^{tf}$  in terms of the  $S_i$ , in the same manner as above.

With these definitions in hand, we can now prove the main result.

**THEOREM 2.4.** *Let  $G$  be the group induced by the action of  $S_{ab}$  on the collection of partitions of  $\{1, 2, \dots, ab\}$  into  $a$  blocks of size  $b$ , where  $a$  and  $b$  are both at least 3.*

- If  $a \geq b$ , then  $b(G) \leq 5$ .
- If  $a < b$ , then  $b(G) \leq \left\lceil \frac{b}{a(a-1)} \right\rceil + 4$ .

**PROOF.** Let  $M$  be a partition of  $\{1, 2, \dots, ab\}$  into  $a$  parts of size  $b$ . If  $a = b$ , then the theorem reduces to Lemma 2.3.

Suppose that  $a > b$  and that  $b$  divides  $a$ . Note that Lemma 2.1 carries over to this more general setting via our “square” construction and that Lemma 2.2 did not require that  $a = b$ .

Then, it follows from these two results that  $\mathcal{B} = [M, M^t, M^{tf}, M^{c_1s}]$  forms a base for  $G$ .

If  $b$  does not divide  $a$ , that is,  $M = [S_1, S_2, \dots, S_k, R]$ , then any  $g \in S_{ab}$  that fixes  $\mathcal{B}$  can still move elements within a column of  $R$ . In order to prevent this, we pick some square  $S_i$  and create a new partition  $M^r$  by taking a copy of  $M$  and interchanging the elements in  $\text{Col}(R, j)$  and  $\text{Row}(S_i, j)$  (that is, swap each column of  $R$  with a different row of  $S_i$ ). Then, we clearly have that  $[M, M^t, M^{tf}, M^{c_1s}, M^r]$  forms a base for  $G$  and we have proven the first part of the theorem.

We now consider the case where  $b > a$ , which is somewhat more delicate. Recall that any permutation that only moves elements within columns of  $M$  fixes  $M$ . Since  $b$  can be arbitrarily large, it follows from the pigeonhole principle that no constant number of partitions will form a base for  $G$  for all  $b > a$ .

Suppose, firstly, that  $a$  divides  $b$ . Form a new partition,  $M^{(1)}$ , by taking a copy of  $M = [S_1, S_2, \dots, S_k]$  and carrying out the following procedure. For each  $S_i$ , where  $i \in \{1, 2, \dots, a-1\}$ , apply the cycle construction  $i$  times to each row of  $S_i$  (halting the construction if there are no remaining squares in  $M$ ). That is, apply the permutation  $(1, 2, \dots, a)^i$  to the column indices of each row of  $S_i$ . Create  $M^{(2)}$  by cycling the next sequence of  $(a-1)$  squares and so on in this manner, in order to create  $M^{(1)}, M^{(2)}, \dots, M^{(\ell)}$ , where  $\ell = \frac{b}{a(a-1)}$ .

We claim that, in this case,  $\mathcal{B} = [M, M^t, M^{tf}, M^{(1)}, \dots, M^{(\ell)}]$  forms a base for  $G$ . It is not hard to see that a nontrivial permutation  $g \in S_{ab}$  that fixes  $\mathcal{B}$ , cannot move columns of  $M$ . Therefore, the only case to consider is when  $g$  moves points within a column of  $M$ . So, suppose that  $M(i, j)^g = M(k, j)$ , where  $i \neq k$ . Lemma 2.1 implies that  $i$  and  $k$  have to correspond to rows appearing in different squares of  $M$ . However, it is easy to see that there is some  $M^{(p)}$  such that row  $i$  gets cycled a different amount of times from row  $k$ , so that  $g$  does not fix  $M^{(p)}$ .

If  $a$  does not divide  $b$  then, in a similar manner to the  $a > b$  case, we can create an extra partition in order to prevent points in the rectangular matrix that does not fit into any square from being moved by a permutation that fixes all of the  $\left\lceil \frac{b}{a(a-1)} \right\rceil + 4$  partitions. This completes the proof of the theorem.  $\square$

### 3. Elementary Results

Bounding the orders of primitive permutation groups in terms of their degrees was a major focus of nineteenth century group theorists. The most striking result of this period is the following.

**THEOREM 3.1** (Bochert [11]). *Let  $G$  be a primitive permutation group acting on a set  $\Gamma$  where  $|\Gamma| = n$ . If  $G$  does not contain  $\text{Alt}(\Gamma)$ , then  $\lfloor (n+1)/2 \rfloor! \leq |\text{Sym}(\Gamma) : G|$ .*

**PROOF.** We use without proof the well known fact that if a primitive group acting on a set  $\Lambda$  contains a 3-cycle, then it contains  $\text{Alt}(\Lambda)$  (see, e.g., [61, Theorem 13.3]).

Our basic strategy is to argue the contrapositive and show that, if the stated bound does not hold, then we can construct a 3-cycle and, therefore,  $G$  would contain  $\text{Alt}(\Gamma)$ , contradicting our hypothesis.

Now, if we can find  $a, b \in G$  such that  $a$  and  $b$  have precisely one point  $\alpha$  in common when written in disjoint cycle form, then an easy calculation shows that the commutator  $[a, b] = a^{-1}b^{-1}ab$  is a 3-cycle. So, assume that the stated bound does not hold. We now proceed to construct elements having precisely one point in common.

Let  $\Delta \subseteq \Gamma$  be as large as possible with  $\text{Sym}(\Delta) \cap G = 1$  and set  $k = |\Delta|$  (this is always satisfied for  $k = 1$ ). We wish to show that  $k \geq n/2$ , so let us suppose to the contrary that  $k < n/2$ . Then,  $|\Gamma \setminus \Delta| = n - k > k$ , so  $\text{Sym}(\Gamma \setminus \Delta) \cap G \neq 1$ . Let  $g \in \text{Sym}(\Gamma \setminus \Delta) \cap G$  be such that  $g$  moves some  $\alpha \in \Gamma \setminus \Delta$  but no element of  $\Delta$ . We then have that  $\text{Sym}(\Delta \cup \{\alpha\}) \cap G \neq 1$ , so choose some  $h \neq 1$  from  $\text{Sym}(\Delta \cup \{\alpha\}) \cap G$ . If  $h$  fixed  $\alpha$ , then  $h$  would be in  $\text{Sym}(\Delta) \cap G = 1$ , which is impossible. Therefore,  $h$  moves  $\alpha$  but no other point of  $\Gamma \setminus \Delta$ , so there is precisely one point in common in the cycle decomposition of  $g$  and  $h$ . So,  $[g, h]$  is a 3-cycle and  $G$  contains  $\text{Alt}(\Gamma)$ , contradicting our hypothesis. We, therefore, have that  $k \geq n/2$ .

If  $u, v \in \text{Sym}(\Delta)$  belong to the same coset of  $G$ , then  $uv^{-1} \in \text{Sym}(\Delta) \cap G = 1$ , that is  $u = v$ . So, each element of  $\text{Sym}(\Delta)$  determines at most one coset of  $G$  in  $\text{Sym}(\Gamma)$ . Therefore, we have that  $|\text{Sym}(\Gamma) : G| \geq |\text{Sym}(\Delta)| = k!$ . Because  $k \geq n/2$  and  $k$  is an integer, we have that  $k \geq \lfloor (n+1)/2 \rfloor$  and we are done.  $\square$

We shall need the following consequence later in this section.

**COROLLARY 3.2.** *For  $n \geq 23$ , let  $G$  be a permutation group acting on a set  $\Gamma$  such that  $|\Gamma| = n$  and*

$$|S_n : G| < \frac{1}{2} \binom{n}{\lfloor n/2 \rfloor}.$$

*Then, there is a subset  $\Delta \subseteq \Gamma$  with  $|\Delta| > \lfloor n/2 \rfloor$  such that the setwise stabiliser  $G_\Delta$  induces a group containing  $\text{Alt}(\Delta)$  on  $\Delta$ .*

**PROOF.** This is almost immediate from the proof of Theorem 3.1. See [38] for the full result.  $\square$

While the argument in the proof of Bochert's Theorem is elegant in that it requires no special group theoretic properties, the bound it provides is not substantially better than the trivial one of  $|G| \leq n!$ . However, we shall use it frequently in deriving far better bounds throughout this chapter.

For the rest of this section,  $G$  denotes an arbitrary primitive permutation group.

Bochert's Theorem gives us that  $|G| \leq \exp(\frac{n}{2}(\log(n) + \epsilon))$ , for some  $\epsilon > 0$ . This was vastly improved by Helmut Wielandt in the case where  $G$  is simply primitive by using substantially different techniques. Recall that a Sylow  $p$ -subgroup of a group  $G$  is a subgroup of order  $p^e$

where  $p$  is prime and  $e$  is the largest integer such that  $p^e$  divides  $|G|$ . These are named after Ludwig Sylow who proved, amongst other things, that Sylow  $p$ -subgroups always exist [48]. By bounding the exponent of Sylow  $p$ -subgroups, Wielandt proved the following theorem.

**THEOREM 3.3** (Wielandt [62]). *Let  $G$  be a simply primitive group of degree  $n$ . If  $G$  does not contain  $A_n$ , then  $|G| < 24^n$ .*  $\square$

Cheryl Praeger and Jan Saxl subsequently showed that not only is the restriction that  $G$  be simply primitive not required for the strategy to work, but that it in fact yields a stronger bound if one is more careful and patient.

**THEOREM 3.4** (Praeger and Saxl [49]). *Let  $G$  be a primitive group of degree  $n$ . If  $G$  does not contain  $A_n$  then  $|G| < 4^n$ .*  $\square$

Shortly after Theorem 3.4 was published, László Babai was able to obtain asymptotically sharper bounds in the case where  $G$  is simply primitive [3]. His paper marks the first direct use of bases in order to obtain bounds on the orders of primitive groups. The techniques are combinatorial and elementary but sufficiently complex that it would take us too far astray to cover the details. However, we illustrate a result using similar techniques later in this section.

**THEOREM 3.5** (Babai [3]). *If  $G$  is a simply primitive group of degree  $n$ , then the minimal base size for  $G$  is less than  $4\sqrt{n}\log(n)$ .*  $\square$

Combining Theorem 3.5 with Lemma 1.4 immediately yields the following result.

**THEOREM 3.6.** *If  $G$  is a simply primitive group of degree  $n$ , then  $|G| \leq n^{4\sqrt{n}\log(n)}$ .*  $\square$

This last result is close to best possible. In order to see this, let  $n = m(m-1)/2$  and let  $G$  be the symmetric group  $S_n$  in its action on 2-element subsets of  $\{1, 2, \dots, m\}$ . Then,  $|G| = m! > (m/e)^m$ , so  $|G| > n^{c\sqrt{n}}$  for some constant  $c$  (see [16, Page 113]).

Babai subsequently proved a similar result for doubly transitive groups by using similar but more intricate techniques.

**THEOREM 3.7** (Babai [4]). *Let  $G$  be a doubly transitive group of degree  $n$ . If  $G$  does not contain  $A_n$  then  $|G| < \exp \exp(1.18\sqrt{\log(n)})$  for  $n \geq n_0$ , where  $n_0 \approx 5 \cdot 10^5$ .*  $\square$

Combining Theorem 3.7 with Lemma 1.4, we obtain that the minimal base size for a doubly transitive group is less than  $2^a\sqrt{\log(n)}$  for some constant  $a$ .

While the results we have covered provide us with nontrivial bounds on the order of a permutation group, we have, as of yet, not gained any results that help us in our goal of partitioning permutation groups according to base size. Indeed, Babai notes in [4] that “...it seems that, at best, a substantial refinement of our methods are (sic) required in order to achieve an improved asymptotic bound such as  $\exp(\log^c(n))$ ”.

This is precisely what László Pyber managed to achieve in [51]. What is striking about his results is that, not only do they provide sharper bounds than those in [4], but the proofs are

substantially simpler. However, the approach and techniques are still based on those of [3, 4]. In particular, we have the following definition.

**DEFINITION 3.8 (Splitting Set).** *Let  $G$  be a permutation group acting on a set  $\Gamma$ . Let  $(\alpha, \beta)$  be a pair of elements from  $\Gamma$ . We say that  $\Sigma \subseteq \Gamma$  splits  $(\alpha, \beta)$  if  $\alpha$  and  $\beta$  do not belong to the same orbit of the pointwise stabiliser  $G_{(\Sigma)}$ .*

With this terminology, a base is a set which splits all  $\binom{n}{2}$  pairs of points. We shall require the following result, whose proof we omit.

**LEMMA 3.9 (Babai [4]).** *Let  $G$  be a doubly transitive group of degree  $n$ . If there is an  $m$ -element set which splits at least  $n(n-1)/4$  pairs of points, then  $G$  has a minimal base size of at most  $2m \log(n)$ .  $\square$*

Let  $G$  be a permutation group acting on a set  $\Gamma$ . We say that  $\Sigma \subseteq \Gamma$  is a *halving set* if the pointwise stabiliser  $G_{(\Sigma)}$  has no orbits of size greater than  $n/2$ . It is easy to see that halving sets split more than  $n(n-1)/4$  pairs of points and so satisfy the requirements of Lemma 3.9. The heart of Pyber's proof is the following:

**LEMMA 3.10 (Pyber [51]).** *Let  $G$  be a permutation group of degree  $n$  acting on a set  $\Gamma$ . Let  $k \geq 12$  and suppose that  $\Gamma$  has no subset  $\Delta$  of size  $|\Delta| \geq k$ , such that  $G_{\Delta}$  induces a group containing  $\text{Alt}(\Delta)$  on  $\Delta$ . Then there is a halving set  $\Sigma$  of size  $|\Sigma| \leq 4k$ .*

**PROOF.** Let  $\Sigma = [\alpha_1, \alpha_2, \dots, \alpha_m]$  be such that, for each  $i$  with  $1 \leq i \leq m$ , we have that  $\alpha_i$  is an element of the longest orbit of the pointwise stabiliser  $G^{(i-1)}$  of  $\{\alpha_1, \alpha_2, \dots, \alpha_{i-1}\}$  in  $G$ .

*Claim:* If  $m \geq 2k$ , then  $G^{(m)}$  has no orbits of size greater than  $2n/3$ .

Assume the claim and suppose that  $G^{(m)}$  has an orbit  $\Gamma_0$  of length greater than  $n/2$ . If  $m = 2k$ , then  $n/2 < |\Gamma_0| \leq 2n/3$ . By adding at most  $m$  more points to  $\Sigma$  and applying the claim to the  $G^{(m)}$ -action on  $\Gamma_0$ , we obtain that the longest orbit of the pointwise stabiliser  $G_{(\Sigma)}$  has size at most  $\frac{2}{3}(2n/3) = 4n/9 < n/2$ . Therefore,  $\Sigma$  is a halving set and  $|\Sigma| \leq 4k$ .

We now proceed to prove the claim. Assume that  $m \geq 2k$  and that, contrary to the claim,  $G^{(m)}$  has an orbit of length greater than  $2n/3$ . Then, the longest orbit of each  $G^{(i)}$  must have length at least  $2n/3$ . As  $|G^{(i)} : G^{(i+1)}|$  is equal to the length of the longest orbit of  $G^{(i)}$ , we have that  $|G^{(i)} : G^{(i+1)}| \geq 2n/3$  and, so,  $|G : G^{(m)}| \geq (2n/3)^m$ .

Now, let  $\hat{G}$  denote the group induced by the action of  $G$  on the ordered  $m$ -element subsets of  $\Gamma$ . As there are  $m!$  orderings of each of the  $\binom{n}{m}$  unordered subsets of  $\Gamma$ , we see that  $\hat{G}$  has degree  $\binom{n}{m}m!$ . By the Orbit-Stabiliser theorem, the orbit of the ordered set  $\{\alpha_1, \alpha_2, \dots, \alpha_m\}$  has length  $|\hat{G} : \hat{G}^{(m)}|$ . So, there are at least  $(2n/3)^m / \binom{n}{m}$  ordered  $m$ -sets of this orbit corresponding to some unordered  $m$ -element subset  $\Delta_0 \subseteq \Gamma$ .

It follows, then, that the setwise stabiliser  $G_{\Delta_0}$  induces a permutation group  $H$  of order at least  $(2n/3)^m / \binom{n}{m} \geq m! / (3/2)^m$  on  $\Delta_0$ . Since  $m \geq 2k \geq 24$ , we see that  $|\text{Sym}(\Delta_0) : H| \leq$

$(3/2)^m < \frac{1}{2} \binom{m}{\lfloor m/2 \rfloor}$ . So, by Corollary 3.2, there is a subset  $\Delta \subseteq \Delta_0$  of size at least  $k$  such that the setwise stabiliser  $G_\Delta$  induces a group containing  $\text{Alt}(\Delta)$  on  $\Delta$ , contradicting our hypothesis. This completes the proof of the claim and of the lemma.  $\square$

Essentially, the proof of Lemma 3.10 follows a very similar pattern to that of Theorem 3.1. That is, assume that the bound does not hold and show that the resulting sets are above the critical size required in order to contain an alternating group. Before proceeding further, we need another classical result due to Bochert.

**THEOREM 3.11** (Bochert [12]). *If  $n \geq 26$ , then every doubly transitive group of degree  $n$  moves at least  $n/4$  points.*  $\square$

We say that a group  $G$  *involves* a group  $M$  if there exist subgroups  $N \trianglelefteq H \leq G$  such that  $H/N \cong M$ . Babai points out in [4] that a combination of Theorem 3.11 with the main result of Wielandt's doctoral thesis (see [4] for a statement) immediately yields the following lemma.

**LEMMA 3.12.** *Let  $G$  be a doubly transitive group of degree  $n$  not containing  $A_n$ . If  $G$  involves  $A_k$ , then  $k < 4 \log(n)$ .*  $\square$

We now have all the tools necessary to prove Pyber's main result:

**THEOREM 3.13** (Pyber [51]). *Let  $G$  be a doubly transitive group of degree  $n$  not containing  $A_n$ . Then, the minimum base size  $b(G)$ , is at most  $c \log^2(n)$  for some constant  $c > 0$ .*

**PROOF.** Let  $G$  be a doubly transitive group of degree  $n$ . Then, by Lemma 3.12,  $G$  involves no  $A_i$  for  $i \geq k = \lceil 4 \log(n) \rceil$  and, so,  $G$  satisfies the hypotheses of Lemma 3.10 with  $k = \lceil 4 \log(n) \rceil$ . Therefore,  $G$  has a halving set of size at most  $4 \lceil 4 \log(n) \rceil$  and so, by Lemma 3.9,  $b(G) \leq 8 \lceil 4 \log(n) \rceil \log(n)$ .  $\square$

Without resorting to any overly technical tools, we have been able to establish that all doubly transitive groups, except the symmetric and alternating groups, are small base. This reduces our problem to the case where the group is simply primitive, where the sorts of methods we have been using are unable to obtain more for us. What we need is a more detailed knowledge of the structure of finite groups, in order to be able to distinguish those simply primitive groups that are small base. We cover this in the following section.

## 4. Classification Results

In the last section, we obtained reasonable bounds on the orders of primitive groups by merely considering the elements of the various groups. An alternative technique, and one which has yielded very powerful results in finite group theory, is to study a group in terms of certain types of its subgroups.

Recall that a group  $G$  is called *simple* if its only normal subgroups are the trivial group and  $G$  itself. A *minimal normal subgroup* of a group  $G$  is a nontrivial normal subgroup of  $G$  which does not properly contain any other nontrivial normal subgroup of  $G$ . Although this does not mean that a minimal normal subgroup is simple, the following is true.

PROPOSITION 4.1. *Every minimal normal subgroup of a group is a direct product of isomorphic simple groups.*

PROOF. See [24, Theorem 4.3A]. □

In general, however, a group does not have a unique minimal normal subgroup. We, therefore, make the following definition.

DEFINITION 4.2 (Socle). *Let  $G$  be a group. The socle of  $G$ , denoted  $\text{Soc}(G)$ , is the subgroup generated by the nontrivial minimal normal subgroups of  $G$ .*

In general, the socle of a group can have quite elaborate structure. However, for primitive groups, the structure is more familiar.

THEOREM 4.3. *The socle of a primitive group is a direct product of isomorphic simple groups.*

PROOF. See [24, Corollary 4.3B]. □

Therefore, if we wish to understand primitive groups in terms of their socles, we first need to understand simple groups. In particular, if we could obtain a list of all the finite simple groups, then that would greatly aid us in our task.

The *classification of finite simple groups* (CFSG) was a most remarkable collaborative mathematical project. It involved scores of mathematicians from dozens of countries working on various aspects of the problem for over three decades. The end result of this work is the following.

THEOREM 4.4 (CFSG). *A nonabelian finite simple group is either alternating, of Lie type or one of 26 “sporadic” groups which do not naturally fit into any infinite family.*

The “groups of Lie type” may be thought of as special sorts of matrix groups, which together form 16 infinite families of simple groups. Four of these families are collectively dubbed the “classical groups”. As we shall not require groups of Lie type in this thesis, we omit further details from our discussion and proofs, providing references where necessary.

It is estimated that the length of the final proof of the classification theorem is somewhere between 10,000 and 15,000 pages spread across journal articles, theses, monographs and manuscripts with the bulk of this work being spread across three decades from the 1950’s through to the late 1980’s. There is currently a revisionist project underway to simplify and unify the proof. There have been several volumes published, the first of which appears as [31] and provides a high level overview of the strategy for the proof. It is hoped that the revisionist effort will produce a more manageable proof of between 3,000 and 4,000 pages.

Many open problems were quickly resolved as a consequence of the classification theorem. While these results are certainly proven, they have the drawback that their proofs are not transparent. This is in contrast to the results of the last section, which rely only on the intrinsic properties of the groups involved. It is for this reason that we termed those results

“elementary”. We shall take care to mention explicitly the use of CFSG throughout the rest of this chapter.

One of the first people to utilise CFSG was Peter Cameron [14]. In fact, the proof of CFSG was not yet completed at that stage, however, the version stated in Cameron’s paper is the same as what we now know to be correct. Utilising CFSG, Cameron was able to determine all the doubly transitive groups explicitly. Using the information in [14], one can draw the same conclusions as in the last section, namely that all doubly transitive groups except the symmetric and alternating groups are small base. A complete proof of this statement via the classification theorem would take many thousands of pages, whereas the elementary version takes only a handful of pages!

It is easily seen that any subgroup of an abelian group is normal, so an abelian simple group  $G$  has as subgroups only the trivial group and  $G$  itself. As all cyclic groups are abelian, it follows from Lagrange’s Theorem that the only simple cyclic groups are the cyclic groups of prime degree, denoted  $\mathbb{Z}_p$  for an arbitrary prime  $p$ . In fact, the only abelian simple groups are the cyclic groups of prime degree. A group is said to be an *elementary abelian  $p$ -group* if it is isomorphic to  $\mathbb{Z}_p \times \mathbb{Z}_p \times \cdots \times \mathbb{Z}_p$  for some prime  $p$ . Since we know from Theorem 4.3 that the socle of a primitive group is a direct product of isomorphic simple groups, we can roughly split primitive groups up into those whose socle is elementary abelian and those whose socle is  $T^m$  for some nonabelian simple group  $T$ .

At the 1979 Santa Cruz Conference on Finite Groups, M. O’Nan and L.L. Scott independently announced a theorem which splits primitive groups up rather more finely [53]. Only a sketch proof of the theorem was given there and, in fact, it contained an error. A corrected and complete proof was given by Liebeck et. al. in [40]<sup>2</sup>. The theorem is still known as the O’Nan-Scott Theorem, however, and we give a statement of it below. The original statement contains more details than we present, as we only present those aspects that are necessary for our discussion.

**THEOREM 4.5 (O’Nan-Scott).** *Let  $G$  be a finite primitive group and  $H := \text{Soc}(G) \cong T^m$  for some simple group  $T$ .*

*If the socle,  $H$ , is regular, then one of the following holds.*

- (i) *(Regular abelian type)  $H$  is a regular elementary abelian  $p$ -group for some prime  $p$ . In this case,  $\deg(G) = p^m = |H|$ .*
- (ii) *(Regular nonabelian type)  $H$  is regular with  $\deg(G) = |T|^m$ .*

*If the socle,  $H$ , is not regular, then  $H$  is nonabelian and one of the following holds.*

- (iii) *(Almost simple type)  $m = 1$  and  $G$  is isomorphic to a subgroup of  $\text{Aut}(T)$ .*
- (iv) *(Diagonal type)  $m \geq 2$  and  $G$  is a group of “diagonal type” with  $\deg(G) = |T|^{m-1}$ .*
- (v) *(Product type)  $m \geq 2$  and  $G$  is isomorphic to a subgroup of the wreath product  $U \wr \text{Sym}(m/d)$*

<sup>2</sup>Their proof makes slight use of CFSG for one of the cases, though this is not required for the version of the theorem that we present.

in the product action for some proper divisor  $d$  of  $m$  and some primitive group  $U$  with  $\text{Soc}(U) \cong T^d$ . So,  $\deg(G) = \deg(U)^{m/d}$ .  $\square$

We shall not deal directly with groups of “diagonal type” and, therefore, omit a discussion of these. Intuitively, one would expect a group with a regular socle to be small base and this is indeed the case.

LEMMA 4.6. *Let  $G$  be a primitive permutation group of degree  $n$  acting on the set  $\Gamma$  and  $H = \text{Soc}(G) \cong T^m$  for some simple group  $T$ . Suppose that  $H$  acts regularly on  $\Gamma$ . Then the minimal base size,  $b(G)$ , of  $G$  is at most  $2 \log(n) + 1$ .*

PROOF. Our proof is an expanded version of the discussion given in [39]. As the pointwise stabiliser  $H_{(\alpha)}$  is trivial for any  $\alpha \in \Gamma$ , the only element of  $G$  that stabilises a generating set  $\{1, \alpha_1, \alpha_2, \dots, \alpha_k\}$  of  $H$  is the identity. From CFSG, we know that any finite simple group can be generated by two elements, so we may take  $k \leq 2m$ . Now, whether  $T$  is abelian or not, we have that  $n = |T|^m$ , so  $m \leq \log(n)$ . We, therefore, have that  $b(G) \leq 2m + 1 \leq 2 \log(n) + 1$ .  $\square$

Lemma 4.6 relies on the structure of the stabiliser of a point in a group with a regular socle. In order to investigate the cases of Theorem 4.5 further, it may be a good idea to look at the stabiliser of a point in a primitive group. The following is a standard result.

LEMMA 4.7. *Let  $G$  be a permutation group acting on the set  $\Gamma$ , with  $|\Gamma| > 1$ . Then,  $G$  is primitive if and only if for each  $\alpha \in \Gamma$ , the pointwise stabiliser  $G_{(\alpha)}$  is a maximal subgroup of  $G$ .*

PROOF. We give a sketch proof. See, e.g., [24, Theorem 1.5A] for a more rigorous discussion. If  $H$  is a transitive permutation group on a set  $\Sigma$ , and  $\sigma \in \Sigma$ , then there is a bijection between all the subgroups of  $H$  containing the pointwise stabiliser,  $H_{(\sigma)}$ , of  $\sigma$  in  $H$  and the blocks of  $H$ . This bijection is merely the map which sends each such subgroup  $J$  to the orbit of  $\sigma$  under  $J$ . So, if  $\gamma \in \Gamma$ , then  $J = G_{(\gamma)}$  corresponds to the singleton  $\{\gamma\}$  and  $J = G$  corresponds to the whole of  $\Gamma$  and we are done.  $\square$

The symmetric and alternating groups in their natural actions are the “largest” primitive groups of any given degree. Theorem 4.5 can then be stated in terms of what the possibilities for the maximal subgroups of a symmetric or alternating group are [16]. We only give the statement for the symmetric group, the corresponding statement for the alternating group is obtained by simply taking the intersection of each case with the alternating group.

THEOREM 4.8 (O’Nan-Scott). *Let  $G$  be the symmetric group of degree  $n$ . Then, a maximal subgroup of  $G$  is one of the following:*

- (1) *Intransitive:  $S_k \times S_\ell$  with  $n = k + \ell$ .*
- (2) *Transitive imprimitive:  $S_k \wr S_\ell$  (imprimitive action),  $n = k\ell$ .*
- (3) *Primitive:  $S_k \wr S_\ell$  (product action),  $n = k^\ell$  and  $k \neq 2$ .*
- (4) *Regular abelian:  $n = p^d$  for some prime  $p$ .*
- (5) *Diagonal:  $n = |T|^{k-1}$  for some nonabelian simple group  $T$  and some  $k \in \mathbb{Z}^+$ .*
- (6) *Almost simple:  $T \lesssim G \leq \text{Aut}(T)$  for a nonabelian simple group  $T$ .*  $\square$

Lemma 4.6 handles two of the cases of Theorem 4.5. For the groups having a simple socle (case (iii) of Theorem 4.5), we have the following result, originally due to Liebeck [39].

**THEOREM 4.9.** *Let  $G$  be a primitive group of degree  $n$  acting on the set  $\Gamma$ . If the socle,  $T$ , of  $G$  is simple, then one of the following holds:*

- (1)  $T$  is  $A_n$  acting on  $k$ -element subsets of  $\Gamma$
- (2)  $T$  is  $A_n$  acting on partitions of  $\Gamma$  into parts of equal size.
- (3)  $T$  is a classical simple group acting on a certain sort of subspace.
- (4)  $|G| < n^9$ .

**PROOF.** We make heavy use of CFSG for this proof, by going through all of the possibilities for a simple group.

Suppose that  $T$  is  $A_m$  and that  $m > 6$ . In this case,  $G$  is either  $S_m$  or  $A_m$  (as  $\text{Aut}(A_m) = S_m$  for all  $m > 6$ ). Let  $\alpha \in \Gamma$  and  $H = G_{(\alpha)}$ . Then,  $G$  is a maximal subgroup of either  $S_m$  or  $A_m$  and we may use Theorem 4.8, or the alternating group version of it. We only deal with the symmetric case here, the alternating case being largely similar.

If  $H$  is intransitive or transitive and imprimitive, then case (1) or (2) holds. If  $H$  is imprimitive on  $\{1, 2, \dots, m\}$ , so that  $H = S_k \wr S_l$  with  $kl = m$ , then Bochert's Theorem 3.1 gives us that  $n = |S_m : H| \geq [(m+1)/2]!$ , which yields that  $|S_m| = m! \leq n^3$  and we are in case (4).

Suppose that  $T$  is of Lie type, then the discussion given in [39] establishes that the four classes of classical groups are in case (3) and the rest are in case (4). We omit further details for this case.

Finally, if  $T$  is sporadic, then one can use ad-hoc calculations based on the information in [19] to verify the remainder of the theorem.  $\square$

Cases (1), (2) and (3) of Theorem 4.9 are the only candidates for large base primitive groups with a simple socle. However, the results in Section 2 establish that case (2) is small base. Liebeck establishes in [39] that if  $G$  is as in case (3) of Theorem 4.9, then  $b(G) < 9 \log(n)$ . Therefore, the only primitive groups with a simple socle that can possibly be large base are the groups whose socle corresponds to the action of  $A_n$  on  $k$ -element subsets of an  $n$  element set. We turn our attention to the case of a general primitive group.

**THEOREM 4.10.** *Let  $G$  be a primitive group of degree  $n$  and  $b(G)$  denote the minimal base size of  $G$ . Then, one of the following holds:*

- (1)  $G$  is a subgroup of  $S_m \wr S_r$  containing  $(A_m)^r$ , where the action of  $S_m$  is on  $k$ -element subsets of  $\Gamma = \{1, 2, \dots, m\}$  and the wreath product has the product action.
- (2)  $b(G) < 9 \log(n)$ .

**PROOF.** Let  $N = \text{Soc}(G) \cong T^r$  for some simple group  $T$ . By the preceding discussions, we only have two cases of Theorem 4.5 left to consider:

- (i) *product action*:  $G \leq G_0 \wr S_r$  in the product action of degree  $n = n_0^r$ , where  $G_0$  acts primitively on  $\Gamma_0$  and has degree  $n_0$  and  $\text{Soc}(G_0) = T$ ;
- (ii) *diagonal action*.

First consider case (i). If  $b(G_0) \geq 9 \log(n)$  then, by Theorem 4.9,  $G_0$  is  $A_m$  or  $S_m$  acting on  $k$ -element subsets of  $\Gamma$  and part (1) holds. So, assume that  $b(G_0) < 9 \log(n)$ . Then, we have that  $b(G) < b(G_0) + r < 9 \log(n_0) + r - 1 \leq 9 \log(n)$ .

If  $G$  is as in case (ii), then [39] establishes via CFSG that  $b(G) < 9 \log(n)$ . Finally, combining Theorem 4.9 and Lemma 1.4 completes the proof of the theorem.  $\square$

We now have all the results necessary to prove the main result of this chapter.

**THEOREM 4.11.** *Let  $G$  be a primitive group. Then  $G$  is large base if and only if  $G$  is a subgroup of  $S_m \wr S_r$  containing  $(A_m)^r$ , where the action of  $S_m$  is on  $k$ -element subsets of  $\Gamma = \{1, 2, \dots, m\}$  and the wreath product has the product action of degree  $\binom{m}{k}^r$ .*

**PROOF.** By Theorem 4.10, it suffices to show that the group induced by the action of  $A_m$  on  $k$ -element subsets of  $\Gamma$  is large base.

We show that if we fix  $k$  and vary  $m$ , then we get a large base family. Indeed, let  $H_j$  denote the group induced by the action of  $A_m$  on  $j$ -element subsets of  $\Gamma$ , with  $m > j$ . Then, we use the extravagantly bad bound of  $n = \deg(H_j) \leq m^j$  to obtain that  $\log(n) \leq j \log(m)$ . Now,  $H_j$  acts as an alternating group on the  $j$ -element subsets of  $\Gamma$ , so has a base of size  $m - c$  for some constant  $c$ . As this grows faster than any integer power of  $\log(m)$ , we are done.  $\square$

This brings us to an end of our discussion of bases for primitive groups. By using the full force of the classification of finite simple groups, we have been able to go much further than in the last section, at the cost of having non-transparent proofs.

While the results of both this section and the last leave no doubt as to precisely what the large base primitive groups are, one may be left wondering what is true of permutation groups in general. There is strong evidence to suggest that a version of Theorem 4.11 holds for transitive groups and we briefly outline some of the motivation. The first extension is immediately established by the following result.

**THEOREM 4.12.** *A transitive group of prime degree is primitive.*

**PROOF.** It is well known (see, e.g., [61, Theorem 8.2]) that the length of each block of a transitive group divides the degree of the group. So, in our case, each block is trivial.  $\square$

Now, if  $G$  is a primitive group then every nontrivial normal subgroup of  $G$  is transitive (see, e.g., [61, Theorem 8.8]). However, the converse is not true, for example, consider nonabelian simple groups acting regularly [50]. A group  $G$  acting on a set  $\Gamma$  is said to be *quasiprimitive* if every nontrivial normal subgroup of  $G$  is transitive on  $\Gamma$ . Cheryl Praeger and Aner Shalev have recently proven the following result.

**THEOREM 4.13** (Praeger and Shalev [50]). *All the results of this section that we have established on base size hold for the class of quasiprimitive groups.*  $\square$

John Bamberg has further widened the scope of the results.

**THEOREM 4.14** (Bamberg [8]). *All the results of this section that we have established on base size hold for groups with a transitive minimal normal subgroup.*  $\square$

Now that we have identified the class of large base (primitive) groups, we turn our attention to computing with them. The rest of this thesis is devoted to discussing various aspects of computing with symmetric and alternating groups in their natural actions.

## Recognising the Giants

In the previous chapter, we have seen how, on the one hand, large base groups are considered to be computationally difficult whereas, on the other hand, they form a relatively small subclass of the class of permutation groups. Moreover, since they mainly appear as various actions of symmetric (or alternating) groups, they are relatively well understood. Therefore, if we can recognise whether an arbitrary group is large base, we can compute a base and strong generating set far faster than by using generic algorithms.

Our aim in this chapter is to develop methods for computing with the symmetric and alternating groups in their natural actions. As we saw in the last chapter, both of these groups are certainly large base. We collectively call the alternating and symmetric groups in their natural action the *giants*<sup>1</sup>, a term which is justified by the results of the last chapter.

Sections 1, 2.1, 2.2 and 2.5 of Chapter 2, which were used in the last chapter, as well as Sections 3 and 4 of Chapter 2 are required for this chapter.

### 1. Recognition Via Action

Let  $G$  be a permutation group acting on a finite set  $\Gamma$ . Recall that we say  $G$  is  $m$ -fold transitive if it acts transitively on the set of ordered  $m$ -tuples of distinct elements of  $\Gamma$  or, equivalently, if the pointwise stabiliser in  $G$  of any  $m - 1$  distinct elements of  $\Gamma$  acts transitively. A natural question that one may ask is “What (families of) permutation groups are  $m$ -fold transitive for a given  $m$ ?”. This question runs surprisingly deep and, although we have a fairly good knowledge of the answer to this question, almost all of the proofs in the literature require the use of The Classification of Finite Simple Groups (CFSG).

The results, however, are very strong and lead to immediate practical applications. Clearly, any group that is  $m$ -fold transitive is  $(m - 1)$ -fold transitive. As it turns out, groups with a high degree of transitivity are quite rare.

**THEOREM 1.1** (Cameron [14]). *The only 6-fold transitive permutation groups are the giants.*

□

The proof of this theorem relies on CFSG and gives a possible technique for recognising giants, provided that we can easily test the degree of transitivity of a permutation group. A classical result due to Jordan [35] (see also [61, Theorem 13.9]) makes matters somewhat easier:

---

<sup>1</sup>A term originally due to Babai et al. [1].

**THEOREM 1.2** (Jordan [35]). *Let  $G$  be a primitive permutation group of degree  $n$  and let  $p$  be a prime less than  $n - 2$ . If  $G$  contains a  $p$ -cycle, then  $G$  is a giant.*  $\square$

We can relax the requirement on  $G$  in Theorem 1.2 if we strengthen the requirement on the prime  $p$ .

**COROLLARY 1.3.** *Let  $G$  be a transitive permutation group of degree  $n$  and let  $p$  be a prime satisfying  $n/2 < p < n - 2$ . If  $G$  contains an element which has a cycle of length  $p$ , then  $G$  is a giant.*

**PROOF.** Our proof is based on the one given in [54, Corollary 10.2.2]. Let  $G$  be a transitive permutation group of degree  $n$  and let  $g \in G$  contain a cycle of length  $p$ , where  $n/2 < p < n - 2$ . Now, because  $p > n/2$ , the lengths of the remaining cycles in  $g$  must divide  $(n - p)!$ . Since  $p$  does not divide  $(n - p)!$ , we have that  $g^{(n-p)!}$  is a  $p$ -cycle. Moreover,  $G$  has to be primitive, since a transitive imprimitive group of degree  $n$  must be isomorphic to a subgroup of  $S_m \wr S_k$  for some  $k, m \in [2, n/2]$ , where the wreath product is endowed with the imprimitive action<sup>2</sup>. Thus, Theorem 1.2 applies and we are done.  $\square$

J. Davenport and G. Smith [23] proved a similar result to Corollary 1.3, however, they were only able to conclude that  $G$  is at least 4-fold transitive and not that it is necessarily a giant. This leads to the result stated above if we use CFSG: It is known that the only 4-transitive permutation groups are the giants and four of the sporadic finite simple groups [15]. Using ad-hoc analysis, Davenport and Smith were then able to reach much the same conclusion as Corollary 1.3 by eliminating these four sporadic groups. However, as we have seen, it is certainly not necessary to resort to CFSG.

Before we can give a rigorous Monte Carlo Yes algorithm for recognising whether a permutation group is a giant, we first need to estimate the frequency of certain elements. The following result is adapted from [54].

**LEMMA 1.4.** (a) *Let  $q$  be an integer such that  $n/2 < q < n - 2$ . The proportion of elements of a giant of degree  $n$  containing a cycle of length  $q$  is  $1/q$ .*

(b) *The proportion of elements of the giants which contain a cycle of length  $p$  for some prime  $p$  in the range  $n/2 < p < n - 2$  is asymptotically  $\ln(2)/\ln(n)$ .*

**PROOF.** (a) There are  $\binom{n}{q}$  possibilities for the support of a  $q$ -cycle in  $S_n$ . Within each cycle, there are  $(q - 1)!$  ways of arranging the points and, finally, there are  $(n - q)!$  permutations on the remaining points. So, we have

$$\binom{n}{q} (q - 1)! (n - q)! = \frac{n! (q - 1)! (n - q)!}{(n - q)! q!} = \frac{n!}{q}$$

elements in total, since there are no elements with two cycles of length  $q$ , as  $q > n/2$ . Since there are  $n!$  elements in  $S_n$ , we are done. Similarly, there are  $\binom{n}{q} (q - 1)! (n - q)! / 2$  elements of  $A_n$  which have a cycle of length  $q$ , as  $q < n - 2$ . As there are  $n!/2$  elements in  $A_n$ , we again

<sup>2</sup>See Chapter 2.

have the desired result.

(b) Clearly, no element of a giant can have cycles of length  $p_1$  and  $p_2$ , where  $p_1$  and  $p_2$  are primes bigger than  $n/2$ . So, the proportion of elements of the giants which contain a cycle of length  $p$  for some prime  $p$  in the range  $n/2 < p < n - 2$ , is the sum of the proportions of each such prime. By [32, Theorem 427],  $\sum_{p < n} 1/p \approx \ln \ln(n)$ , where the sum is over all primes less than  $n$ . So, by part (a), the desired quantity is:

$$\sum_{n/2 < p < n-2} \frac{1}{p} \approx \ln \ln(n-2) - \ln \ln(n/2) = \ln \frac{\ln(n-2)}{\ln(n) - \ln(2)} \approx \ln \left( 1 + \frac{\ln(2)}{\ln(n)} \right) \approx \frac{\ln(2)}{\ln(n)}.$$

□

Davenport and Smith [23] obtained much the same result as Lemma 1.4. We are now in a position to give a Monte Carlo Yes algorithm for recognising giants, which is presented in Algorithm 2. The correctness of the algorithm is clear from the preceding discussion. If it returns “No”, then with probability asymptotically  $1 - e^{-c}$ , the input group is not a giant. A couple of steps need to be justified, namely, picking a random element of  $G$  and testing to see whether a cycle length is prime. We cover methods for generating random elements of a group later in this chapter and, as for primality testing, there are very fast Monte Carlo methods that are used in practice (and which are deterministic if the Extended Riemann Hypothesis<sup>3</sup> is true), see [41, 52] for details. In practice, this algorithm is provably deterministic for the range of values we are interested in (at the time of creating GAP 4 [30], this was the case for  $n < 10^{13}$ ), so does not affect the correctness of Algorithm 2.

---

**Algorithm 2:** Non-Constructive Giant Recognition

---

**Input:** A permutation group  $G$  of degree  $n$  and a positive real  $c$ .

**Output:** “Yes” or “No”

**begin**

$i := 0$

**repeat**

        Let  $g$  be a random element of  $G$ .

$i := i + 1$

**if**  $g$  contains a cycle of length  $p$ , where  $p$  is prime and  $n/2 < p < n - 2$  **then**

**return** “Yes”

**until**  $i > c \ln(n) / \ln(2)$

**return** “No”

**end**

---

**1.1. Application: Computing Galois Groups of Polynomials.** Permutation groups originally arose through the work of the French mathematician Évariste Galois, who was concerned with the roots of polynomials over the rational field. The Galois group of a polynomial  $f$  with rational<sup>4</sup> coefficients, denoted  $\text{Gal}(f)$ , measures the “symmetry in the roots

<sup>3</sup>The first quadratic nonresidue mod  $p$  of a number is always less than  $2 \ln^2(p)$ .

<sup>4</sup>The Galois Group can be defined relative to any field, however, we shall not need this.

of  $f^n$ . We refrain from giving a formal definition of this group, as we shall not actually require it. Suffice to say that a lot of useful information related to  $f$  is encoded in  $\text{Gal}(f)$ . The group  $\text{Gal}(f)$  is a permutation group, which acts on the set of roots of  $f$ . See [26] for a historical account of the theory, including a translation of Galois' original manuscript. In general, computing the Galois group of a polynomial is computationally expensive, however, we show that the techniques we have developed can be used to solve an important special case.

Before we can proceed further, we need to recall some basic algebraic definitions. A *ring* is a triple  $(R, +, \cdot)$ , where  $(R, +)$  is an abelian group,  $\cdot$  is an associative binary operation called "multiplication" and the usual left and right distributive laws hold for multiplication (we usually abbreviate this by  $R$  when the details of the addition and multiplication are either clear or not important). We denote by  $0$  the identity element in  $(R, +)$ . A *field* is a ring for which  $(R \setminus \{0\}, \cdot)$  is an abelian group. A field is called finite if it consists of finitely many elements. Galois proved that a finite field of order  $q$  exists if and only if  $q$  is a prime power and, moreover, any two finite fields of the same order are isomorphic (see, e.g., [26] for a proof). We denote the finite field of order  $q$ , where  $q$  is a prime power, by  $\mathbb{F}_q$ .

If  $R$  is a ring, then we denote by  $R[x]$  the set of all polynomials in  $x$  with coefficients from  $R$  (this is easily seen to be a ring under the usual addition and multiplication of polynomials). The *degree* of a polynomial is the highest exponent of  $x$  appearing in it. If  $f \in R[x]$ , then we say that  $f$  is a polynomial *over*  $R$ . A polynomial  $f \in R[x]$  is said to be irreducible if it has no factorisation into non-constant polynomials of lower degree over  $R$  (for example,  $x^2 + 1$  is irreducible over  $\mathbb{Q}$  and  $\mathbb{R}$ , but factors into  $(x - i)(x + i)$  when considered as an element of  $\mathbb{C}[x]$ ). *Reduction modulo*  $p$ , for some prime  $p$ , is the map that takes  $f \in \mathbb{Z}[x]$  to  $\bar{f} \in \mathbb{F}_p[x]$  by replacing each coefficient of  $f$  with its remainder upon division by  $p$ . In general, a polynomial which is irreducible over  $\mathbb{Z}$  need not be irreducible when reduced modulo  $p$ . For example,  $x^2 + 1$  is irreducible over  $\mathbb{Z}$  but reducing modulo 2 gives  $x^2 + 1 = x^2 + 2x + 1 = (x + 1)^2$  in  $\mathbb{F}_2[x]$ .

Factoring polynomials over an arbitrary field is notoriously difficult, however, in the case where the field is finite, E. Berlekamp created an efficient algorithm for factorisation [10], which has been optimised to run very quickly and is a standard feature of many algebraic packages.

A beautiful theorem of analytic number theory, due to Nikolai Čebotarev [58], allows us to use reduction modulo  $p$  in order to access the cycle structure of random elements of  $\text{Gal}(f)$ , thus opening the door to use Algorithm 2. The basic idea is to take an irreducible polynomial  $f$  with integer coefficients and reduce it modulo a random prime,  $p$ , that satisfies certain conditions. One then factors  $\bar{f}$  over  $\mathbb{F}_p$  into  $\bar{f}_1 \bar{f}_2 \dots \bar{f}_k$  where each  $\bar{f}_i$ , with  $1 \leq i \leq k$ , is irreducible over  $\mathbb{F}_p$ . Letting  $d_i$  be the degree of  $\bar{f}_i$ , we then have that  $(d_1, d_2, \dots, d_k)$  is the cycle structure of a random element of  $\text{Gal}(f)$ . This is stated more formally as follows.

**THEOREM 1.5** (Čebotarev Density Theorem [58]). *Let  $f \in \mathbb{Q}[x]$  be an irreducible polynomial with integer coefficients and let  $p$  be a prime that divides neither the leading coefficient of  $f$ , nor the discriminant<sup>5</sup> of  $f$ . Then, the distribution of the degrees of the irreducible factors of  $f$  modulo  $p$  corresponds to the cycle structure of the action of  $\text{Gal}(f)$  on the roots of  $f$ . If  $P > 0$  and the frequency of these distributions is averaged over all primes  $p \leq P$ , then this converges to the frequency of cycle structures in  $\text{Gal}(f)$  as  $P \rightarrow \infty$ .  $\square$*

Davenport and Smith [23] show that testing whether a Galois group is transitive is easy. Following the above procedure, one can then apply Algorithm 2 to quickly test whether the Galois group of a random polynomial is giant. This is useful, because van der Waerden [59] proved that asymptotically almost all polynomials of degree  $n$  (that is, a proportion tending to 1 as  $n \rightarrow \infty$ ) have a giant Galois group. Empirical tests on this procedure were carried out in [23]. Due to its efficiency, it is often used as an initial stage in the computation of a Galois group, in order to rule out the giants before performing more expensive computations [34].

## 2. Black Box Groups

From a computational point of view, we have a better understanding of permutation groups than just about any other class of groups. Of course, there are other classes that we would like to be able to compute with efficiently.

Recall that the *general linear group*,  $\text{GL}(d, q)$ , is the group of all  $d \times d$  invertible matrices under matrix multiplication, whose entries are elements of the finite field of order  $q$ . A *matrix group* is a subgroup of  $\text{GL}(d, q)$ . Matrix groups are of great importance, not least of all because the vast majority of finite simple groups are defined as matrix groups. However, matrix groups are far more difficult to compute with than permutation groups. For example, the membership problem, namely “given a matrix  $g$ , is  $g \in G \leq \text{GL}(d, q)$ ?”, is extremely expensive computationally. In the special case where  $d = 1$  and  $q$  is prime, it is equivalent to the *discrete logarithm problem modulo  $q$*  [2]: Suppose that  $b$  is a generator of the cyclic group  $\mathbb{Z}_p^*$ , which is the group  $\{1, \dots, p - 1\}$  together with multiplication modulo some prime  $p$ . Then, given  $a \in \mathbb{Z}_p^*$ , the discrete logarithm problem modulo  $p$  is to find  $n$  such that  $b^n \equiv a \pmod{p}$ . The exact computational complexity of the discrete logarithm problem is unknown, although it is widely believed to be intractable for large randomly selected primes and forms the basis of many modern cryptographic systems, for example ElGamal [27].

For this reason, given a matrix group  $G$ , until recently most computer algebra packages would first construct a permutation representation of  $G$ , that is a group  $H \leq S_n$  such that  $G \cong H$ . One can then compute with  $H$  in order to answer questions about  $G$ . This solution is undesirable for at least two reasons.

Firstly, suppose that we have some matrix group  $G$ . In general,  $|G| = q^{\Theta(d^2)}$ , while elements of  $G$  can be represented as strings of length  $\Theta(d^2 \log(q))$ . However, generally, the

<sup>5</sup>We omit the details of this number, see [26] for details.

smallest permutation representation of  $G$  has degree  $q^{\Theta(d)}$ , which represents an exponential increase in the size of group elements when moving from the matrix group to its permutation representation. One can construct smaller representations in certain cases [21], however, this is not possible in general.

Secondly, in the case where the permutation representation,  $H$ , of  $G$  is large base, we do not gain a significant advantage unless we *a priori* know that  $H$  is large base.

Groups can also be represented by their multiplication tables, as factor groups of certain infinite groups<sup>6</sup> or as automorphism groups of combinatorial structures. Thus, if we are to effectively recognise whether a group is large base, we ought to be able to do this irrespective of the manner in which the group is represented.

The notion of a *black box group* was popularised in [2]. It may be likened to the abstract definition of a group, in that it captures the *computational* properties of a group without making assumptions about its structure, whereas an abstract group captures the algebraic properties in this manner.

Black box groups are important for at least two reasons. Firstly, they provide a uniform method by which problems in group theory can be encoded in the traditional language of computational complexity theory. This leads to interesting results about certain group theoretic problems, which have a very different flavour to traditional group theory. Secondly, if one writes an algorithm for a black box group, then this will work for *any* group, thus negating the need to provide a different implementation for each representation of the group (it can be argued that one can do this without thinking about black box groups by being careful about what operations are used, however, black box groups provide a useful way of thinking which ensures that one does not fall into the trap of using a property that is unique to a particular representation).

Let  $S$  be a set. Recall that  $\sim \subseteq S \times S$  is called an *equivalence relation* if it is symmetric, reflexive and transitive. If  $s, t \in S$ , then  $s \sim t$  means  $(s, t) \in \sim$ . If  $s \in S$ , then the *equivalence class* of  $s$  is  $[s] := \{t \in S : t \sim s\}$  and  $S/\sim := \{[s] : s \in S\}$ .

Informally, a black box group,  $\mathcal{G}$ , is a set of strings of uniform length over some alphabet (which is almost always finite and usually equal to  $\{0, 1\}$ ) with a distinguished element  $e$ , which plays the role of the identity, together with three oracles:

- (1) *Equality* ( $\doteq$ ): given  $a, b \in \mathcal{G}$ , return “Yes” if  $a \doteq b$  and “No” otherwise.
- (2) *Multiplication* ( $\otimes$ ): Given  $a, b \in \mathcal{G}$ , return  $a \otimes b$ .
- (3) *Inversion* ( $\natural$ ): Given  $a \in \mathcal{G}$ , return  $a^\natural$ , where  $a \otimes a^\natural \doteq e \doteq a^\natural \otimes a$ .

---

<sup>6</sup>I.e., as a presentation. That is, a quotient of a free group by the normal closure of certain relations.

It is also common to require that the elements of the underlying group are uniquely encoded (that is, the underlying equivalence relation is trivial and  $\doteq$  reduces to equality of strings). Note that, when designing algorithms for black box groups, *we do not know how the oracles work*. For example, we can multiply two elements together by using the multiplication oracle, but have no information on how this multiplication is performed. This is what leads to the name “black box” group. We give a formal definition below.

**DEFINITION 2.1 (Black Box Group).** *Given a tuple  $(\mathcal{A}, n, S, \doteq, \otimes, \natural)$ , where  $\mathcal{A}$  is an alphabet,  $n$  is a positive integer,  $S \subseteq \mathcal{A}^n$  and  $\doteq$  is an equivalence relation on  $S$ , we say that the quadruple  $(S/\doteq, \doteq, \otimes, \natural)$  is a black box group if  $\otimes$  is an associative binary operation on  $S/\doteq$  and  $\natural$  is a unary operation on  $S/\doteq$  such that for each  $a \in S/\doteq$ , we have  $a \otimes \natural(a) \doteq e \doteq \natural(a) \otimes a$  for some distinguished element  $e$  of  $\mathcal{A}^n$ , satisfying  $a \otimes e \doteq a$  for all  $a \in S/\doteq$ .*

It is immediate from the definition that the order of a black box group is at most  $|\mathcal{A}|^n$ . Membership in a black box group is equivalent to membership in  $S/\doteq$  and we write  $a^\natural$  to mean  $\natural(a)$ . At the moment, the definition of a black box group is very abstract. In the following section, we construct an explicit example of an infinite family of black box groups.

**2.1. Constructing Black Box Representations for Finite Abelian Groups.** It should be reasonably clear from the definition that a black box group is itself an abstract group and that, dually, most groups can be seen, *a fortiori*, as a black box group<sup>7</sup>. The question then arises as to whether we can define a set of strings, together with three oracles (inversion, multiplication, equality), that is *a fortiori* isomorphic to an abstract group. This section is the author’s own work and fills a gap in the literature, which does not contain nontrivial constructions of black box groups. We carry out such a construction for the class of finite abelian groups, inspired by the following standard result.

**THEOREM 2.2 (Fundamental Theorem of Finite Abelian Groups).** *Every finite abelian group is isomorphic to a direct sum of cyclic groups and, moreover, the factors in the sum are unique up to reordering.*

**PROOF.** As it is tangential to our discussion, we omit details. See [25, Section 6.1] for a group-theoretic proof.  $\square$

Thus, we may think of finite abelian groups in terms of their decomposition into cyclic groups. We take as our alphabet the natural numbers,  $\mathbb{N}$ , and proceed to construct a language with “enough” elements to be able to represent any product of cyclic groups. In this section, we bend Definition 2.1 slightly to allow the strings to be countably long, as opposed to having finite length. This is done solely to make the exposition clearer and the number of elements of the resulting black box group will always be finite. With this in mind, let  $\mathcal{C}_i = \{0, 1, \dots, i-1\}$  and form the infinite product  $\mathcal{F}_i = \mathcal{C}_i \times \mathcal{C}_i \times \dots$  for each  $i \in \{2, 3, \dots\}$ . We then take our language to be  $\mathcal{L} = \mathcal{F}_2 \times \mathcal{F}_3 \times \dots$ . This construction is illustrated in Figure 1.

<sup>7</sup>The exception being some finitely presented groups, which may fail to have a solvable word problem, which in turn negates the possibility of an equality oracle.



as above, then the resulting black box group is isomorphic to

$$\bigoplus_{i=1}^k \bigoplus_{j=1}^{pe_i} \mathbb{Z}_{e_i},$$

where  $\oplus$  denotes the direct sum operation. □

**2.2. Generating Tuples.** Of course, it would be tedious and inefficient to equate all but the smallest black box groups with the set of strings representing their elements. For this reason, one usually takes a subset of strings which, together with the oracles, generate the remaining elements of the group. Such a collection is known as a *generating tuple*.

**DEFINITION 2.4 (Generating Tuple).** A generating tuple for a black box group  $\mathcal{G}$  is a subset of strings of  $\mathcal{G}$  which, together with the equality, multiplication and inversion oracles of  $\mathcal{G}$ , generate the remainder of the elements of  $\mathcal{G}$ .

Henceforth, “black box group” means a black box group that is presented as a generating tuple together with oracles. Obtaining lower bounds on the order of black box groups is extremely difficult. In particular, we have the following result.

**THEOREM 2.5 (Babai [5]).** No polynomial time Monte Carlo algorithm can determine the logarithm of the order of a black box group of order  $n$  within a factor of  $\sqrt{n}$ . □

If we are to take as input only a generating tuple together with group oracles, then we require a method for accessing the remaining group elements.

**DEFINITION 2.6 (Straight Line Program).** Let  $\mathcal{G}$  be a black box group with generating tuple  $X$  and let  $g \in \mathcal{G}$ . A straight line program for  $g$  is a sequence  $(s_1, s_2, \dots, s_m)$  of elements of  $\mathcal{G}$  such that  $s_m \doteq g$  and for each  $i$ , with  $1 \leq i \leq m$ , one of the following holds:

- (1)  $s_i$  is a member of  $X$ , or
- (2)  $s_i = s_j^{\natural}$  for some  $j < i$ , or
- (3)  $s_i = s_j \otimes s_k$  for some  $k, j < i$ .

While it is clear that straight line programs reaching an arbitrary element of a black box group exist, we still require effective bounds on the length of such a program. A result of Babai and Szemerédi gives an effective bound, though it does not give a procedure for actually constructing the program.

**LEMMA 2.7 (Babai and Szemerédi [2], Babai [5]).** Given a set  $S$  of generators of the finite group  $G$ , there exists a straight line program of length at most  $\log^2(|G|)$ , which reaches a sequence of elements  $h_1, h_2, \dots, h_t$  such that every element of  $G$  can be represented as a product  $x^{-1}y$ , where  $x, y$  belong to the set  $\{h_1^{c_1} h_2^{c_2} \dots h_t^{c_t} : c_j \in \{0, 1\}\}$  and  $t \leq \log(|G|)$ . □

### 3. Random Elements of Finite Groups

Generating random integers is a classical area of computer science, for which many advanced algorithms already exist. As it is beyond the scope of this dissertation, we do not discuss them here and take their existence for granted. The reader is referred to [36, Chapter

3] for a detailed account.

In contrast, generating random elements of finite groups is a relatively recent area of research, with many problems still open. The black box representation of groups has proven to be particularly fruitful in this situation, as it allows one to design algorithms that are agnostic to the structure of the input group. Throughout this section, we take “group” to mean “black box group”. For the remainder of this chapter, we do not use special symbols for the operations within a black box group, in order to aid readability.

Algorithms for generating uniformly random elements of a group are not currently known, so one usually settles for a slightly weaker notion of randomness.

**DEFINITION 3.1** ( $\epsilon$ -uniform). *A probability distribution over a set  $S$  is called  $\epsilon$ -uniform if each element is selected with probability  $p$ , where  $(1-\epsilon)/|S| < p < (1+\epsilon)/|S|$  for some  $0 < \epsilon < 1$ .*

As we are working in the context of black box groups, we effectively only have access to a generating tuple. Some definitions are in order before we can deal with random generation in this context. Recall that a graph,  $\Gamma$ , is a pair  $(V, E)$  where  $V$  is a finite set and  $E$  is a subset of  $\{\{v_1, v_2\} : v_1, v_2 \in V \text{ and } v_1 \neq v_2\}$ . The elements of the set  $V$  are called the *vertices* of the graph and the elements of the set  $E$  are called the *edges* of the graph. A *neighbour* of a vertex  $v \in V$  is another vertex  $v'$  such that  $\{v, v'\} \in E$  and two vertices are *adjacent* if they are neighbours. The *degree* of a vertex is the number of its neighbours.

**DEFINITION 3.2** (Cayley Graph). *Let  $G$  be a group with generating set  $S$ . The Cayley Graph  $\text{Cay}(G, S)$  is the graph which has  $G$  for its vertex set, with two vertices  $g, h \in G$  being adjacent if and only if  $gs = h$  for some  $s \in S \cup S^{-1}$ , where  $S^{-1} := \{s^{-1} : s \in S\}$ .*

If  $\Gamma = (V, E)$  is a graph, then an *automorphism* of  $\Gamma$  is a bijection  $\varphi : V \rightarrow V$  such that  $\{v_1, v_2\} \in E$  if and only if  $\{\varphi(v_1), \varphi(v_2)\} \in E$ . The set of all automorphisms of  $\Gamma$  forms a group under composition, called the *automorphism group* of  $\Gamma$ , and denoted by  $\text{Aut}(\Gamma)$ . The group  $\text{Aut}(\Gamma)$  is called *vertex transitive* if it acts transitively on  $V$ . Informally, this means that each vertex “looks the same”. In particular, each vertex has the same degree.

Since the group  $G$  acts on its Cayley graph  $\text{Cay}(G, S)$  by right translations  $\rho_g : x \mapsto xg$  for  $x, g \in G$ , it follows that all Cayley graphs are vertex transitive.

**DEFINITION 3.3** (Random Walk). *Let  $G$  be a group with generating set  $S$  and let  $\mathcal{C} = \text{Cay}(G, S)$  be its Cayley graph, where each vertex has degree  $k$ . A random walk of length  $m$  on  $G$  is a sequence of points  $(s_1, s_2, \dots, s_m)$  of  $\mathcal{C}$ , where  $s_1 = 1$  and each  $s_{i+1}$  is a randomly selected neighbour of  $s_i$  in  $\mathcal{C}$ , picked with uniform probability  $1/k$ .*

Since Cayley graphs are vertex transitive, we have not lost any generality by making all random walks on them start from 1. We now have enough tools in order to describe the approach taken by Babai [5] for generating random group elements. The basic strategy is to create a Monte Carlo algorithm, which makes Theorem 2.7 effective in the following sense.

Let  $G$  be a group and let  $N$  be an upper bound on its order. Given a generating set  $S$  of  $G$ , our goal is to create another generating set  $S'$  such that  $|S'| = |S| + c_1 \log(N)$  and every element of  $G$  can be represented as a product of at most  $c_2 \log(N)$  elements of  $S'$  and their inverses, for some real constants  $c_1$  and  $c_2$ . The creation of the set  $S'$  is achieved by Algorithm 3.

---

**Algorithm 3:** Create  $S'$ 


---

**Data:** Generating set  $S$  for a group  $G$  and an upper bound  $N$  on  $|G|$

**Result:** Generating set  $S'$  such that each element of  $G$  can be represented by a product of  $c_2 \log(N)$  elements of  $S'$  and their inverses.

**begin**

$S' := S$

**for**  $i := 1$  **to**  $c_3 \log(N)$  **do**

$R := \emptyset$

$\ell := \lceil 2052i^2 |S'| \ln(2N) \rceil$

**for**  $j := 1$  **to**  $c_1/c_3$  **do**

            Select a random integer  $\tau \in \{2i + 1, \dots, \ell\}$

            Make a random walk of length  $\tau$  in the Cayley Graph  $C(G, S')$

            Add the element reached to  $R$

$S' := S' \cup R$

**return**  $S'$

**end**

---

**THEOREM 3.4** (Babai [5]). *Algorithm 3 costs  $O(\log^5(N))$  group operations. Moreover, upon successful termination of Algorithm 3, for any  $\epsilon > 0$ , we are able to generate  $\epsilon$ -uniformly distributed random elements of  $G$  at a cost of  $O(\log^4(N) + \log^3(N) \log(1/\epsilon))$  group operations per random element.*  $\square$

At the moment, the cost of generating random elements is almost as high as the cost of setting up the set  $S'$ . We now set about lowering this cost considerably. If  $g_1, g_2, \dots, g_k$  are elements of a group, a *subproduct* of this sequence is an element of the form  $g_1^{c_1} g_2^{c_2} \dots g_k^{c_k}$ , where each  $c_i \in \{0, 1\}$ . A sequence of points  $h_1, h_2, \dots, h_k$  of a group  $G$  is called a sequence of  $\epsilon$ -uniform Erdős-Rényi generators if each element of  $G$  is represented in  $(2^k/|G|)(1 \pm \epsilon)$  as a subproduct of  $h_1 h_2 \dots h_k$ . In other words, by picking random subproducts, one obtains  $\epsilon$ -uniform group members. The name comes from the following famous result.

**THEOREM 3.5** (Erdős and Rényi [28]). *Let  $G$  be a group and  $k$  be an integer such that  $k \geq 2 \log(|G|) + 2 \log(1/\epsilon) + \log(1/\delta)$  for some  $\epsilon > 0$  and  $\delta > 0$ . Then, a sequence of  $k$  random elements of  $G$  is a sequence of  $\epsilon$ -uniform Erdős-Rényi generators for  $G$  with probability at least  $1 - \delta$ .*  $\square$

It is easy to see that if we had a set of Erdős-Rényi generators, then we could generate  $\epsilon$ -uniform elements of  $G$  with about  $\log(N)$  group operations - just flip a coin for each Erdős-Rényi generator in order to decide its exponent. However, one must also compensate for the value of  $\epsilon$  and the true complexity of producing a random element, as Babai [5] shows, is  $O(\log(N) \log(1/\epsilon))$  group operations. Our only task now is to construct such a set of generators.

**THEOREM 3.6** (Babai [5]). *There is a Monte Carlo algorithm which, with probability at least  $1 - \delta$ , produces a sequence of  $\epsilon$ -uniform Erdős-Rényi generators. The cost of the algorithm is  $O(\log^5(N) + \log^3(N)(\log(1/\epsilon) + \log^2(1/\delta)))$  group operations.*  $\square$

A preprint of Cooperman [20] provides an algorithm which considerably lowers the preprocessing time. His analysis is far more technical and involved than Babai's, so we omit details.

**THEOREM 3.7** (Cooperman [20]). *Let  $G$  be a group with generating set  $S$  and let  $N$  be an upper bound on the order of  $G$ . One can construct a data structure for computing  $\epsilon$ -uniform elements of  $G$  in  $O(\log^2(N) + |S| \log(N))$  group operations. One can then construct an  $\epsilon$ -uniform element of  $G$  in  $O(\log(N) \log(1/\epsilon))$  group operations.*  $\square$

At the moment, Theorem 3.7 is the best known theoretical result, from the point of view of the time required to set up the data structure. Although Cooperman's algorithm is more efficient than Babai's, in the sense that its preprocessing time is far lower, it may still be impractical. For example, suppose that  $G = \text{GL}(d, q)$ . Since  $\log(|G|) \leq d^2 \log(q)$ , the cost of producing random elements of  $G$  may be prohibitively high for large  $d$  (i.e., in the low hundreds). This is an important case, as it arises in the on-going international "computational matrix group project", see [37].

Both Babai and Cooperman's algorithms are similar in nature, in that they apply a top-down approach to generating the random element. That is, start with a set of elements and return a random subproduct of them. Heuristically, bottom-up procedures run much faster in practice, with the downside that rigorous theoretical analysis is very difficult with current techniques. The most widely used algorithm is the so-called *product replacement* algorithm, introduced in [17]. It is described in Algorithm 4.

---

**Algorithm 4:** Product Replacement

---

**Input:** List  $X$  of length  $k$  containing elements of  $G$

**Output:** An element of  $G$

**begin**

    Select random  $i, j \in \{1, 2, \dots, k\}$ , with  $i \neq j$

    Select random  $\ell \in \{0, 1\}$

**if**  $\ell = 1$  **then**

$X[i] := X[i]X[j]$

**else**

$X[i] := X[j]X[i]$

**return**  $X[i]$

**end**

---

The way in which the product replacement algorithm is used in practice, is to initialise the list  $X$  to contain elements of a generating set for  $G$  and to first run Algorithm 4 a certain number of times and discard the element returned (this is the "preprocessing" stage). Subsequent calls to the algorithm then produce an element that is likely to be random (note the contrast to

Babai and Cooperman's algorithms, where one can actually prove properties of the distribution of the elements returned).

Experiments described in [17], which test the product replacement algorithm on practical problems, show that the elements returned approach a uniform probability after a preprocessing stage of about 100 iterations of the algorithm. Due to the practical importance of the algorithm, there has been considerable interest in trying to prove stronger theoretical properties. Early results are summarised in the paper by Pak [46] and a stronger result, also due to Pak, appears in [45].

#### 4. Towards Constructive Recognition of Black Box Giants

Our goal in this section is to investigate the *constructive recognition problem* for giants. Specifically, given a black box group  $G$ , we wish to be able to test whether it is isomorphic to a given large base group  $H$  quickly and, if it is, to construct an isomorphism  $\varphi : G \rightarrow H$ , such that  $\varphi$  can be used to rewrite elements of  $G$  into elements of  $H$ . Our focus throughout will be on the case where  $H = S_n$ , the case where  $H = A_n$  being a technical, rather than conceptual, modification. As a further simplification, we make the assumption that  $n$  is already known. The case where  $n$  is not known in advance has not been adequately dealt with in the literature yet, so most papers make this assumption as well.

Given a black box group  $G$ , it is more or less hopeless to wish for a *deterministic* algorithm for solving the constructive recognition problem, as we cannot even determine the order of  $G$  in Monte Carlo polynomial time (see Theorem 2.5)! Thus, our focus will be on using randomisation to solve the problem, an approach that is justified by the last section.

Recalling the basic non-constructive recognition algorithm covered earlier in this chapter (Algorithm 2), a possible strategy is to generate random elements of  $G$  and hope that they satisfy some property such that, firstly, they prove that  $G \cong S_n$  and, secondly, we can rewrite elements of  $G$  in terms of the elements that we have found. The book of Coxeter and Moser [22] proves that if we can find nontrivial elements  $r, s \in G$  such that

$$(3) \quad r^n = s^2 = (rs)^{(n-1)} = [s, r^j]^2 = 1 \text{ for } 2 \leq j \leq n/2,$$

where  $[a, b] = a^{-1}b^{-1}ab$  denotes the *commutator* of  $a$  and  $b$ , then  $G$  is isomorphic to  $S_n$ . So, our first task is to identify elements satisfying the relations. This is done in the following lemma.

LEMMA 4.1. *For  $n \geq 5$ , an  $n$ -cycle,  $r$ , and a transposition,  $s$ , moving two consecutive points of  $r$  satisfy the relations in equation 3.*

PROOF. The proof is a straightforward calculation. Suppose that  $r = (a_1, a_2, \dots, a_n)$  and  $s = (a_1, a_2)$ . The relations  $r^n = s^2 = 1$  and  $r^2 \neq 1$  are clear. Since  $rs = (a_1)(a_2, \dots, a_n)$ , it follows that  $(rs)^{n-1} = 1$ . Now,  $[s, r^j] = s^{-1}(r^{-1})^j s r^j = s s^{r^j} = (a_1, a_2)(a_1^{r^j}, a_2^{r^j})$ . We clearly have that  $r^j$  maps  $a_1$  to  $a_{j+1}$  and  $a_2$  to  $a_{j+2}$ . Moreover, since  $2 \leq j \leq n/2$ , it follows that

$j + 1 \geq 3$  and  $j + 2 \leq n/2 + 2 \leq n$ . Therefore,  $\{a_1, a_2\} \cap \{a_1^{r^j}, a_2^{r^j}\} = \emptyset$ , which implies that  $[s, r^j]^2 = 1$  and we are done.  $\square$

One's first thought may be to select a certain number of random elements of  $G$  until either an  $n$ -cycle and a transposition moving two consecutive points are found or else, with a suitably small probability of error, the group is not isomorphic to  $S_n$ . In the black box setting, however, it is not quite as straightforward as that. Since we do not have access to the cycle structure of elements, it is impossible to tell directly whether a given element is a transposition or an  $n$ -cycle. Moreover, we cannot even test the order of a group element efficiently, as the best we can do is keep multiplying the element by itself until we get the identity. As a further complication, the proportion of transpositions in  $S_n$  is  $\binom{n}{2} \cdot (n!)^{-1} = (2(n-2)!)^{-1}$ , which is far too small to be effective for large  $n$ . We outline two different approaches to overcoming these problems.

The first published work that aims to solve the constructive giant recognition problem is a paper by Sergey Bratus and Igor Pak [13]. Their paper begins by simplifying the problem further by making the assumption that the black box group comes equipped with an *order oracle*, which can quickly determine the order of a group element. This is a rather undesirable extra condition to impose, as  $S_n$  can have elements of order as large as  $\exp(\sqrt{n \ln(n)})$  (see [29]). Furthermore, they subdivide the problem according to the parity of  $n$ . We outline only the case where  $n$  is even.

The first task to address is finding an  $n$ -cycle. In order to do this, Bratus and Pak first search for an element  $y \in G$  such that the order of  $y$  is  $p_1 p_2$  for distinct primes  $p_1$  and  $p_2$  such that  $p_1 + p_2 = n$ . This is where the second complication arises, as one has no guarantee that such primes exist. It is a very old number theoretic conjecture, going back to Christian Goldbach in 1742, that any even integer  $n \geq 2$  can be expressed as the sum of two primes. Although much progress has been made in number theory since it was first conjectured, the problem remains open. The analysis in [13] relies on a slightly stronger conjecture, the precise nature of which was first proposed by G.H. Hardy and J.E. Littlewood in [33].

**CONJECTURE 4.2** (Extended Goldbach Conjecture). *For every  $n \geq 20$ , there exist primes  $p_1 > p_2 > 3$  such that  $p_1 + p_2 = n$ . Moreover, the number of such pairs is greater than  $\frac{n}{3 \ln^2(n)}$ .*

Assuming we have such a  $y \in G$  of order  $p_1 p_2$  where  $p_1 + p_2 = n$ , it is clear that  $y_1 := y^{p_2}$  and  $y_2 := y^{p_1}$  are cycles of length  $p_1$  and  $p_2$  respectively. By making similar use of the order oracle, we can find a transposition  $a$  that interchanges a point in  $y_1$  with a point in  $y_2$  (details of how to do this may be found in [13]). It is then easy to see that the element  $b := y_1 a y_2$  is an  $n$ -cycle in  $G$  and we have succeeded in finding the elements that we wanted.

How much does this procedure cost? Let  $\rho$  denote the time required to compute an  $\epsilon$ -random element of  $G$ , let  $v$  denote the time required to find the order of an arbitrary element of  $G$  and let  $\mu$  denote the time required to perform a multiplication or inversion in  $G$ .

**THEOREM 4.3** (Bratus and Pak [13]). *Suppose that  $n \geq 20$  and it is known that the Extended Goldbach Conjecture holds. Then, for  $\epsilon > 0$ , there is a Las Vegas algorithm which, if successful, produces a data structure that can then be used to compute the image in  $S_n$  of an arbitrary element of  $G$  under some fixed isomorphism  $\varphi : G \rightarrow S_n$ . If unsuccessful, the probability of no result being returned is  $\epsilon$ . The algorithm has a time complexity of  $O([\rho+v+\mu] \log(1/\epsilon)n \log^2(n))$  and the time complexity of computing the image of an element is  $O(\mu n \log(n))$ . The space complexity is  $O(n)$ .  $\square$*

Unfortunately, Bratus and Pak note that “...the constant implied by the  $O(\cdot)$  notation is expected to be very large...”. This, coupled with the reliance on an order oracle, makes the algorithm undesirable even in the event that the Extended Goldbach Conjecture is proven to be true. Moreover, their algorithm requires  $O(n)$  space, which is more of a problem in practice, as one quickly runs out of memory when running the algorithm on moderately sized examples. For example, on a PIII 933 Mhz with 512Mb of RAM, the implementation in Magma runs out of memory when run on  $S_{1200}$ , presented as a permutation group<sup>9</sup>.

The approach taken by the authors in [9] is more direct. Recall that, if  $x$  is a permutation, then  $x^m = 1$  if and only if the order of  $x$  divides  $m$ . The main insight of [9] is that “most” elements of order dividing  $m$  are in fact an  $m$ -cycle. One can use repeated squaring (see the Chapter 2) in order to (relatively) quickly determine whether an element has order dividing  $m$  in  $O(\log(m))$  multiplications.

In order to find a transposition, we set  $f := n - 2$  if  $n$  is odd and  $f := n - 3$  if  $n$  is even. The basic strategy is to search for an element,  $g$ , of order dividing  $2f$  but not  $f$ . Then, [9] proves that if  $n > 48$ , we have that the conditional probability that  $g$  contains a transposition and an  $f$ -cycle in its cycle decomposition tends to 1 and is greater than  $1/3$ . If  $g$  has this cycle decomposition, then we can easily obtain a transposition by computing  $g^f$ . A similar argument to Lemma 1.4 shows that the proportion of elements in  $S_n$  that have the desired cycle type is  $1/(2f)$ . Determining the proportion of elements of order dividing  $2f$  is more difficult and we do not cover it here, though we cover a related problem later in this chapter.

Suppose, for the moment, that we have found an  $n$ -cycle,  $a$ , and a transposition  $b$ . The problem now is to find a transposition which moves consecutive points of  $a$ . It is easy to see that if  $b = (b_1, b_2)$  and  $x$  is an arbitrary permutation, then the conjugate of  $b$  by  $x$  is  $x^{-1}bx = ((b_1)^x, (b_2)^x)$ . That is, the conjugate of a transposition is a transposition. So, we can conjugate  $b$  by random elements in order to find one that moves consecutive elements of  $a$  (this is essentially the same strategy as Bratus and Pak, although they were looking for a transposition which moved points between their prime cycles). All that remains is for us to find a way to test whether a transposition moves consecutive points of an  $n$ -cycle.

**LEMMA 4.4.** *If  $x \in S_n$  is an  $n$ -cycle and  $z \in S_n$  is a transposition, then  $z$  moves two consecutive points of  $x$  if and only if  $[z, z^x] \neq 1$ .*

<sup>9</sup>Thanks to Maska Law for details of this experimentation, conducted in September 2004.

PROOF. We may assume, throughout the proof, that  $x = (1, 2, \dots, n)$ . Let  $z = (i, j)$ , so that  $z^x = (i + 1, j + 1)$ , where we take  $n + 1$  to be 1. Suppose, firstly, that  $[z, z^x] \neq 1$ . This happens precisely when  $z$  does not commute with its image under  $x$ , that is, when  $\{i, j\} \cap \{i + 1, j + 1\} \neq \emptyset$ . In this case, we either have  $i \in \{i + 1, j + 1\}$  or  $j \in \{i + 1, j + 1\}$ . If  $i \in \{i + 1, j + 1\}$ , then we must have  $i = j + 1$ , so that  $z = (i, i - 1)$ . Similarly, if  $j \in \{i + 1, j + 1\}$ , then  $z = (i, i + 1)$ . In either case, we have that  $z$  moves two consecutive points of  $x$ .

Suppose now, without loss of generality, that  $z = (i, i + 1)$  (where we take  $n + 1 = 1$ ). Then,  $[z, z^x] = (i, i + 1)(i + 1, i + 2)(i, i + 1)(i + 1, i + 2) = (i, i + 1, i + 2) \neq 1$ .  $\square$

Our aim now is to find an  $n$ -cycle in  $G$ . In order to do this effectively, we estimate the conditional probability that an element of order dividing  $n$  in  $S_n$  is an  $n$ -cycle. We deal with this in the following section.

### 5. Roots of Unity in the Symmetric Group

Our goal in this section is to provide effective bounds on the number of solutions to the equation  $x^n = 1$  in the symmetric group  $S_n$ . In other words, we wish to estimate the proportion of elements that have order dividing  $n$ . In particular, we show that there is a high conditional probability that an element of order dividing  $n$  is an  $n$ -cycle. Clearly, the number of  $n$ -cycles is  $(n - 1)!/n! = 1/n$ . Counting the remaining elements requires somewhat more work.

The earliest relevant result goes back to Frobenius in the 1895 *Berliner Sitzungsberichte*.

**THEOREM 5.1 (Frobenius).** *Let  $G$  be a finite group. If  $m$  divides the order of  $G$ , then the number of solutions to  $x^m = 1$  in  $G$  is a multiple of  $m$ .*  $\square$

Let  $f(m, n)$  denote the number of solutions to the equation  $x^m = 1$  in  $S_n$ . Moser and Wyman [42] prove that, for  $p$  a prime,

$$f(p, n) \approx \frac{1}{\sqrt{p}} n^{n(1-1/p)} \exp(-n(1-1/p) + n^{1/p}),$$

utilising an argument that relies heavily on complex analysis. Now, for  $p$  a prime,  $f(p, p) = (p - 1)! + 1$ , since the only elements of order dividing  $p$  are the  $p$ -cycles and the identity. Together with Frobenius' Theorem, this gives another proof of Wilson's Theorem from elementary number theory, which states that  $(p - 1)! \equiv -1 \pmod{p}$ . Moser and Wyman further prove that, for  $d < p$ , the identity  $f(d, p) \equiv 1 \pmod{p}$  holds, which may be seen as a generalisation of Wilson's Theorem.

Three decades later, Wilf [63] used an intricate complex analysis argument to find an asymptotic formula for  $f(m, n)$  when  $m$  is fixed and  $n \rightarrow \infty$ . Unfortunately, the formula does not provide an *effective* bound on  $f(m, n)$  as it still uses a sum over divisors of  $m$ .

A very different approach was taken by Warlimont [60], who provided effective asymptotic bounds for  $f(n, n)$ . His proof is entirely elementary and combinatorial in nature. Unfortunately,

this paper seems to be relatively unknown amongst English-speaking mathematicians. We focus our attention on the case where  $m = n$ . Our discussion below uses ideas from [9], as well as ideas due to Alice Niemeyer and Cheryl Praeger, who extend the method to general  $m$  in [43].

Our first observation is that a permutation  $x \in S_n$  satisfying  $x^n = 1$  can only have cycles of length dividing  $n$  in its disjoint cycle decomposition. As a very rough first approximation, we begin by considering the function

$$(4) \quad \tau(n) = \sum_{d|n} 1,$$

which counts the number of divisors of  $n$ . Our discussion of this function is, unless otherwise stated, based on Hardy and Wright [32] and Tenenbaum [57].

Recall the Fundamental Theorem of Arithmetic, which states that any positive integer can be written as a product of primes in a unique way. As a matter of convenience, throughout this section we reserve the symbol  $p$ , possibly with subscripts and/or superscripts, to refer to a prime number. We write  $p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$  for the decomposition of a number into a product of primes, where  $p_1, p_2, \dots, p_k$  are distinct primes. Our first task is to turn Equation (4) into something with which it is easier to work.

LEMMA 5.2. *If  $n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$ , then*

$$\tau(n) = \prod_{i=1}^k (\alpha_i + 1).$$

Moreover,  $\tau$  is a multiplicative function, that is,  $\tau(nm) = \tau(n)\tau(m)$  if  $\gcd(n, m) = 1$ .

PROOF. The first part follows immediately from the fact that each divisor of  $n$  is a sub-product of  $p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$ . That  $\tau$  is multiplicative follows at once, since if  $n$  and  $m$  are coprime, then they share no common factors in their prime decompositions.  $\square$

If  $n \geq 2$ , then, using Lemma 5.2, one can see that  $\tau(n) = 2$  if and only if  $n$  is itself a prime. In other words, we have that, as  $n \rightarrow \infty$ , the lower limit of  $\tau(n)$  is 2. Unfortunately, this is of little help to us, as we require sharp *upper* bounds on the order of  $\tau$ . This requires some more careful analysis.

THEOREM 5.3.  $\tau(n) = O(n^\delta)$  for all  $\delta > 0$ .

PROOF. Suppose that  $n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$ . Then, by Lemma 5.2,

$$(5) \quad \frac{\tau(n)}{n^\delta} = \prod_{i=1}^k \frac{\alpha_i + 1}{p_i^{\alpha_i \delta}}.$$

We write  $p$  for an arbitrary  $p_i$  appearing in (5) and write  $\alpha$  for  $\alpha_i$ . If  $p \geq 2^{1/\delta}$ , then  $p^\delta \geq 2$ , so

$$\frac{\alpha + 1}{p^{\alpha \delta}} \leq \frac{\alpha + 1}{2^\alpha} \leq 1.$$

Now, since  $\alpha\delta \ln(2) \leq e^{\alpha\delta \ln(2)} = 2^{\alpha\delta} \leq p^{\alpha\delta}$ , we have

$$(\alpha + 1)/p^{\alpha\delta} \leq 1 + \alpha/p^{\alpha\delta} \leq 1 + 1/(\delta \ln(2)) \leq \exp(1/(\delta \ln(2))).$$

Using this estimate for those  $p < 2^{1/\delta}$ , we obtain

$$\frac{\tau(n)}{n^\delta} \leq \prod_{p \leq 2^{1/\delta}} \exp\left(\frac{1}{\delta \ln(2)}\right) < \exp\left(\frac{2^{1/\delta}}{\delta \ln(2)}\right) = O(1)$$

and we are done.  $\square$

COROLLARY 5.4. *For all  $n > 0$ , the bound  $\tau(n) \leq 3.53n^{1/3}$  holds.*

PROOF. See [44, Page 395].  $\square$

We now turn our attention to bounding the number of solutions to  $x^n = 1$  in  $S_n$ . Our argument is elementary and relatively straightforward, though one must take care not to get lost in the details. The basic strategy is as follows. First, let  $X(n)$  be the set consisting of all those permutations  $g$  in  $S_n$  satisfying  $g^n = 1$ . Denote by  $P(n)$  the proportion of those elements in  $S_n$ , that is,  $P(n) = |X(n)|/n!$ . We subdivide  $X(n)$  into three disjoint sets

$$\begin{aligned} X_1(n) &= \{g \in X(n) : 1, 2, 3 \text{ lie in the same } g\text{-cycle}\}, \\ X_2(n) &= \{g \in X(n) : 1, 2, 3 \text{ lie in precisely two } g\text{-cycles}\} \text{ and} \\ X_3(n) &= \{g \in X(n) : 1, 2, 3 \text{ lie in three different } g\text{-cycles}\}. \end{aligned}$$

By a “ $g$ -cycle”, we mean one of the disjoint cycles of  $g$ . For example, if  $g = (1, 2)(3, 4)$ , then the  $g$ -cycles are  $(1, 2)$  and  $(3, 4)$ . Let  $P_i(n)$  be the proportion of each  $X_i(n)$  in  $S_n$ , for  $i \in \{1, 2, 3\}$ . That is,  $P_i(n) = |X_i(n)|/n!$ . Then, since the  $X_i$  are pairwise disjoint, we have that

$$P(n) = P_1(n) + P_2(n) + P_3(n).$$

We then proceed to estimate the size of each  $P_i$ . This approach proves to be quite fruitful and leads to effective bounds. Our first task is to obtain recursive formulae for each of the  $P_i$ . Henceforth, we adopt the convention that  $P(0) = 1$ . Note the trivial observation that, for all  $i \in \{1, 2, 3\}$ , we have  $|X_i(n)|n! = P_i(n)$ .

LEMMA 5.5. *For every positive integer  $n$ , the following equalities hold.*

$$(6) \quad P_1(n) = \frac{(n-3)!}{n!} \sum (d-1)(d-2)P(n-d),$$

where the sum is over all divisors  $d$  of  $n$  such that  $3 \leq d \leq n$ .

$$(7) \quad P_2(n) = \frac{3(n-3)!}{n!} \sum \sum (d_2-1)P(n-d_1-d_2),$$

where the double sum is over all divisors  $d_1$  and  $d_2$  of  $n$  such that  $d_1 + d_2 \leq n$  and  $2 \leq d_2$ .

$$(8) \quad P_3(n) = \frac{(n-3)!}{n!} \sum \sum \sum P(n-d_1-d_2-d_3),$$

where the triple sum is over all divisors  $d_1, d_2, d_3$  of  $n$  such that  $d_1 + d_2 + d_3 \leq n$ .

PROOF. In all three cases, we count the number of permutations for fixed  $d_i$  and then sum over the valid range. Recall that the *support* of a permutation is the set of points that it

moves. Throughout the proof,  $d$  denotes a divisor of  $n$  and distinct subscripts denote distinct divisors. We leave some of the routine calculations to the reader.

$P_1(n)$ : For  $g \in X_1(n)$ , all of the points  $1, 2, 3$  are contained in the same  $g$ -cycle, say  $C$ , of length  $d$ , for some divisor  $d$  of  $n$  satisfying  $3 \leq d \leq n$ . Now, the remainder of the support of  $C$  can be chosen in  $\binom{n-3}{d-3}$  ways and there are  $(d-1)!$  ways in which  $C$  can be obtained from this set. Since the remaining components of  $g$  can be picked in  $(n-d)!P(n-d)$  different ways, we have that  $|X_1(n)| = \binom{n-3}{d-3}(d-1)!(n-d)!P(n-d) = (n-3)!(d-1)(d-2)P(n-d)$ . Summing over the valid  $d$ , we obtain the stated result.

$P_2(n)$ : For  $g \in X_2(n)$ , we must have one of the points  $\{1, 2, 3\}$  in some  $g$ -cycle, say  $C_1$ , of length  $d_1$  and the other two lying in another  $g$ -cycle, say  $C_2$ , of length  $d_2$ , with  $d_2 \geq 2$ . The remainder of the support of  $C_1$  can be chosen in  $\binom{n-3}{d_1-1}$  different ways, and there are  $(d_1-1)!$  ways in which  $C_1$  can be obtained from this set. Similarly, there are  $\binom{n-d_1-2}{d_2-2}(d_2-1)!$  ways to pick  $C_2$ . Since the rest of  $g$  can be picked in  $(n-d_1-d_2)!P(n-d_1-d_2)$  ways, and noting that there are three ways in which to select the point from  $\{1, 2, 3\}$  in  $C_1$ , we have  $|X_2(n)| = 3(n-3)!(d_2-1)P(n-d_1-d_2)$ . Summing over the valid  $d_1$  and  $d_2$ , we obtain the stated result.

$P_3(n)$ : For  $g \in X_3(n)$ , each of  $1, 2$  and  $3$  must lie in different  $g$ -cycles, say  $C_1, C_2$  and  $C_3$ , of lengths  $d_1, d_2$  and  $d_3$  respectively, with  $d_1 + d_2 + d_3 \leq n$ . Similar arguments to the previous two cases give that the number of possibilities for  $C_1$  is  $\binom{n-3}{d_1-1}(d_1-1)!$ , for  $C_2$  is  $\binom{n-d_1-2}{d_2-1}(d_2-1)!$  and for  $C_3$  is  $\binom{n-d_1-d_2-1}{d_3-1}(d_3-1)!$ . Since there are  $(n-d_1-d_2-d_3)!P(n-d_1-d_2-d_3)$  possibilities for the remainder of  $g$ , we have that  $|X_3(n)| = (n-3)!P(n-d_1-d_2-d_3)$ . Summing over the valid  $d_1, d_2$  and  $d_3$ , we obtain the desired result.  $\square$

We now set about utilising Lemma 5.5 in order to obtain an effective bound on  $P(n)$ . We go about this in two stages. In the first stage, we take  $P(m) = 1$  for all  $m < n$ , in order to obtain a first approximation to  $P(n)$ . In the second stage, we use the bound obtained in stage 1 towards obtaining a tighter bound.

LEMMA 5.6 (Round 1). *If  $n \geq 160$ , then  $P(n) < \frac{6}{n-1}$ .*

PROOF. Throughout this proof, we take  $P(m) = 1$  for all  $m < n$ . We remove the  $n$ -cycles and count those separately at the end. That is, we may assume that any divisor of  $n$  is at most  $n/2$ .

$P_1(n)$ : By Lemma 5.5,  $P_1(n) \leq \frac{(n-3)!}{n!} \sum (d-1)(d-1)$ , where the sum is over all divisors  $d$  of  $n$  such that  $d \geq 3$ . Since  $d = n/t$  for some  $t > 0$ , we have  $3 \leq n/t \leq n/2$ , which implies that

$2 \leq t \leq n/3$ . Therefore, we have

$$\begin{aligned} P_1(n) &\leq \frac{(n-3)!}{n!} \int_2^{n/3} \left(\frac{n}{t} - 1\right) \left(\frac{n}{t} - 2\right) dt \\ &\leq \frac{(n-3)!}{n!} \int_2^{n/3} \frac{n^2}{t^2} dt = \frac{n(n-6)(n-3)!}{2n!} = \frac{(n-6)}{2(n-1)(n-2)} < \frac{1}{2(n-1)}. \end{aligned}$$

$P_2(n)$ : By Lemma 5.5, we have  $P_2(n) \leq \frac{3(n-3)!}{n!} \sum_{d_1|n} \sum_{d_2|n} (d_2 - 1)$ , where  $d_1 + d_2 \leq n$  and  $d_2 \geq 2$ . We then have

$$P_2(n) \leq \frac{3(n-2)}{2n(n-1)(n-2)} \sum_{d_1|n} \sum_{d_2|n} 1 = \frac{3(\tau(n))^2}{2n(n-1)}.$$

By Corollary 5.4, we may take  $\tau(n) \leq 3.53n^{1/3}$ . Taking  $n \geq 160$ , we then obtain that  $P_2(n) < 3.5/(n-1)$ .

$P_3(n)$ : By similar arguments, we have

$$P_3(n) \leq \frac{(n-3)!}{n!} \sum_{d_1|n} \sum_{d_2|n} \sum_{d_3|n} 1 \leq \frac{(n-3)! (\tau(n))^3}{n!} < \frac{44}{(n-1)(n-2)},$$

where we take  $\tau(n) \leq 3.53n^{1/3}$ . Taking  $n \geq 160$ , we have that  $P_3(n) < 0.28/(n-1)$ . Combining the obtained bounds and adding back the  $n$ -cycles, we see that

$$(9) \quad P(n) = P_1(n) + P_2(n) + P_3(n) < \frac{1}{n} + \frac{1}{2(n-1)} + \frac{3.5}{(n-1)} + \frac{0.28}{n-1} < \frac{6}{(n-1)}.$$

□

The following lemma will be needed in the course of the second round of analysis of  $P(n)$ .

LEMMA 5.7. *The only triples of integers  $(a, b, c)$ , with  $a, b$  and  $c$  all at least 2 and  $1/a + 1/b + 1/c = 1$  are  $(2, 3, 6), (2, 4, 4)$  and  $(3, 3, 3)$ , together with the nontrivial permutations of these triples.*

PROOF. Without loss of generality, we may take  $a \leq b \leq c$ . Then, the only possibilities for  $a$  are 2 and 3. If  $a = 2$ , then  $1/b + 1/c = 1/2$  and one easily sees that the only way this could happen is if  $b = 3$  and  $c = 6$ , or if  $b = c = 4$ . If  $a = 3$ , then  $1/b + 1/c = 2/3$  and one easily sees that the only way for this to happen is if  $b = c = 3$ . □

We now perform the second round of analysis on  $P(n)$ .

THEOREM 5.8 (Round 2). *If  $n \geq 250$ , then*

$$P(n) < \frac{1}{n} + \frac{10.6}{n^{2/3}(n-1)} + \frac{107}{(n-1)(n-6)} + \frac{3}{2n(n-1)} + \frac{4.5}{(n-1)(n-2)}.$$

PROOF. As in the first round, we remove the  $n$ -cycles at the start and add them back at the end. In other words, we only consider those divisors  $d$  of  $n$  such that  $d \leq n/2$ . Throughout, we use the bound from Lemma 5.6 to bound  $P(m)$  for  $m \leq n$ , and denote this by  $L(m)$ .

$P_1(n)$ : From Lemmas 5.6 and 5.5 and Corollary 5.4, taking  $\tau(n) \leq 3.53n^{1/3}$ , we have

$$\begin{aligned} P_1(n) &= \frac{(n-3)!}{n!} \sum_{d|n} (d-1)(d-2)P(n-d) \leq \frac{(n-2)(n-4)L(n/2)}{4n(n-1)(n-2)} \sum_{d|n} 1 \\ &\leq \frac{12(n-2)(n-4)\tau(n)}{4n(n-1)(n-2)(n-4)} \leq \frac{10.6}{n^{2/3}(n-1)}. \end{aligned}$$

$P_2(n)$ : From Lemma 5.5, we have

$$P_2(n) = \frac{3(n-3)!}{n!} \sum_{d_1|n} \sum_{d_2|n} (d_2-1)P(n-d_1-d_2),$$

where  $d_1 + d_2 \leq n$  and  $d_2 \geq 2$ . If  $d_1 = d_2 = n/2$ , then we easily obtain that this contributes  $3/(2n(n-1))$  to  $P_2(n)$ . Removing this element, we have that one of  $d_1$  or  $d_2$  is less than  $n/2$ . Then, arguing in the usual manner, we have

$$P_2(n) \leq \frac{3(n-3)!}{n!} \sum_{d_1|n} \sum_{d_2|n} L(n/6)(d_2-1) \leq \frac{3(36(n-2))}{2n(n-1)(n-2)(n-6)} \sum_{d_1|n} \sum_{d_2|n} 1 = \frac{54(\tau(n))^2}{n(n-1)(n-6)}.$$

By Corollary 5.4, we may take  $\tau(n) \leq 3.53n^{1/3}$ , whereupon we obtain, after assuming that  $n \geq 250$ , that  $P_2(n) < 107/(n-1)(n-6)$ . Adding back those elements that occur when  $d_1 = d_2 = n/2$ , we obtain the final bound

$$P_2(n) \leq \frac{107}{(n-1)(n-6)} + \frac{3}{2n(n-1)}.$$

$P_3(n)$ : By Lemma 5.5, we have

$$P_3(n) \leq \frac{(n-3)!}{n!} \sum_{d_1|n} \sum_{d_2|n} \sum_{d_3|n} P(n-d_1-d_2-d_3),$$

where  $d_1 + d_2 + d_3 \leq n$ . By Lemma 5.7, there are only ten possible values for  $(d_1, d_2, d_3)$ , such that  $d_1 + d_2 + d_3 = n$ . Moreover, we may take  $d_i \leq n/4$  for the remaining elements, where  $i \in \{1, 2, 3\}$ . We then calculate

$$\begin{aligned} P_3(n) &\leq \frac{10(n-3)!}{n!} + \frac{(n-3)!}{n!} \sum_{d_1|n} \sum_{d_2|n} \sum_{d_3|n} L(n/4) \\ &\leq \frac{10}{n(n-1)(n-2)} + \frac{24(\tau(n))^3}{n(n-1)(n-2)(n-4)}. \end{aligned}$$

Once more taking  $\tau(n) \leq 3.53n^{1/3}$  and  $n \geq 250$ , we obtain that  $P_3(n) < 4.5/(n-1)(n-2)$ .

Since  $P(n) = P_1(n) + P_2(n) + P_3(n)$ , combining the obtained bounds and adding back the  $n$ -cycles yields the stated result.  $\square$

**COROLLARY 5.9.** *If  $n > 281$ , then the conditional probability that an element,  $g$ , of the symmetric group of degree  $n$  satisfying  $g^n = 1$  is an  $n$ -cycle is at least 0.6. For  $24 < n \leq 281$ , this probability is at least 0.5.*

**PROOF.** The result for  $n > 281$  follows by a numerical approximation using Theorem 5.8. For  $24 < n \leq 281$ , the stated bound follows by explicit computation of the probability, which was carried out in [9].  $\square$

This last corollary is a significant strengthening of the result in [9], which established a probability of at least  $1/180$  for all  $n$ .

For each  $\epsilon > 0$ , we can compute a constant  $N(\epsilon)$  such that the probability that an  $n$ -cycle is found within  $N(\epsilon)$  random group elements is at least  $1/2$ . The authors in [9] carry out a similar analysis for finding transpositions, searching for elements of order dividing  $2(n-2)$  if  $n$  is odd or  $2(n-3)$  if  $n$  is even. Once elements of each order are found, a deterministic procedure constructs the isomorphism and checks that the elements are indeed an  $n$ -cycle and a transposition, respectively.

This brings us to an end of our discussion of algorithms for computing with giants. By making use of a variety of tools from number theory, combinatorics and probability theory, we have been able to illustrate state-of-the-art algorithms for both non-constructive and constructive recognition of giants.

## CHAPTER 5

### Conclusion and Open Problems

*All human knowledge thus begins with intuitions,  
proceeds thence to concepts, and ends with ideas.*

– Immanuel Kant

We have sampled a large variety of modern group theory during the course of our discussions. In Chapter 3, we saw how the classification of finite simple groups links in with the O’Nan-Scott Theorem to allow one to investigate primitive permutation groups in quite some detail, which is a widely used technique in modern algebraic combinatorics. We also covered combinatorial methods, which use very few properties of the groups involved in order to derive very strong results, seemingly by magic. Apart from providing the first comprehensive overview of the various tools and techniques, we obtained a significant strengthening of an old result from the literature (Section 2), which forms the basis for [18]. Moreover, all of the proofs of that chapter were significantly reworked by the author, in such a way that they are easier to follow for those unfamiliar with the relevant research literature.

Given the success of the combinatorial methods in identifying the large base doubly transitive groups, we leave the following open problem, in the hope that it stimulates the creation of new and interesting techniques.

OPEN PROBLEM 1. *Identify the large base primitive groups without utilising the classification of finite simple groups.*

Of course, by using deeper properties of finite groups, more is possible.

OPEN PROBLEM 2. *Identify the large base transitive groups.*

Chapter 4 covered a large variety of modern computational group theory. Besides providing a coherent overview of the different areas and how they interact, we provided a new encoding of an infinite family of black box groups (Section 2.1) and gave a new rigorous analysis of the proportion of  $n$ -cycles amongst the elements of order dividing  $n$  in  $S_n$  (Section 5).

## Bibliography

- [1] L. Babai, E. Luks, and A. Seress. Permutation groups in NC. In *Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 409–420. ACM Press, 1987.
- [2] L. Babai and E. Szemerédi. On the complexity of matrix group problems I. In *25th annual Symposium on Foundations of Computer Science, October 24–26, 1984, Singer Island, Florida*, pages 229–240, 1984.
- [3] László Babai. On the order of uniprimitive permutation groups. *Ann. of Math. (2)*, 113(3):553–568, 1981.
- [4] László Babai. On the order of doubly transitive permutation groups. *Invent. Math.*, 65(3):473–484, 1981/82.
- [5] László Babai. Local expansion of vertex-transitive graphs and random generation in finite groups. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 164–174. ACM Press, 1991.
- [6] László Babai. Randomization in group algorithms: conceptual questions. In *Groups and computation, II (New Brunswick, NJ, 1995)*, volume 28 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 1–17. Amer. Math. Soc., Providence, RI, 1997.
- [7] László Babai, Gene Cooperman, Larry Finkelstein, and Ákos Seress. Nearly linear time algorithms for permutation groups with a small base. In *Proceedings of the 1991 international symposium on Symbolic and algebraic computation*, pages 200–209. ACM Press, 1991.
- [8] John Bamberg. Bounds and quotient actions of innately transitive groups. to appear in *J. Aust. Math. Soc.*
- [9] Robert Beals, Charles R. Leedham-Green, Alice C. Niemeyer, Cheryl E. Praeger, and Ákos Seress. A black-box group algorithm for recognizing finite symmetric and alternating groups. I. *Trans. Amer. Math. Soc.*, 355(5):2097–2113 (electronic), 2003.
- [10] E. R. Berlekamp. Factoring polynomials over large finite fields. *Math. Comp.*, 24:713–735, 1970.
- [11] Alfred Bochert. Ueber die zahl der verschiedenen werthe, die eine function gegebener buchstaben durch vertauschung derselben erlangen kann. *Math. Ann.*, 33:584–590, 1889.
- [12] Alfred Bochert. Ueber die classe der transitiven substitutionengruppen. *Math. Ann.*, 40:584–590, 1892.
- [13] Sergey Bratus and Igor Pak. Fast constructive recognition of a black box group isomorphic to  $S_n$  or  $A_n$  using Goldbach’s conjecture. *J. Symbolic Comput.*, 29(1):33–57, 2000.
- [14] Peter J. Cameron. Finite permutation groups and finite simple groups. *Bull. London Math. Soc.*, 13(1):1–22, 1981.
- [15] Peter J. Cameron. Some multiply transitive permutation groups. In *Coding theory, design theory, group theory (Burlington, VT, 1990)*, Wiley-Intersci. Publ., pages 1–11. Wiley, New York, 1993.
- [16] Peter J. Cameron. *Permutation groups*, volume 45 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 1999.
- [17] Frank Celler, Charles R. Leedham-Green, Scott H. Murray, Alice C. Niemeyer, and E. A. O’Brien. Generating random elements of a finite group. *Comm. Algebra*, 23(13):4931–4948, 1995.
- [18] Jonathan Cohen. Bounds on the minimal base size for the symmetric group on partitions. In Preparation.
- [19] J. H. Conway, R. T. Curtis, S. P. Norton, R. A. Parker, and R. A. Wilson. *Atlas of finite groups*. Oxford University Press, Eynsham, 1985. Maximal subgroups and ordinary characters for simple groups, With computational assistance from J. G. Thackray.
- [20] Gene Cooperman. Towards a practical, theoretically sound algorithm for random generation in finite groups. Preprint available at <http://arxiv.org/abs/math.PR/0205203>.

- [21] Gene Cooperman, Larry Finkelstein, Michael Tselman, and Bryant York. Constructing permutation representations for matrix groups. *J. Symbolic Comput.*, 24(3-4):471–488, 1997. Computational algebra and number theory (London, 1993).
- [22] H. S. M. Coxeter and W. O. J. Moser. *Generators and relations for discrete groups*. Springer-Verlag, Berlin, 1957.
- [23] J. H. Davenport and G. C. Smith. Fast recognition of alternating and symmetric Galois groups. *J. Pure Appl. Algebra*, 153(1):17–25, 2000.
- [24] John D. Dixon and Brian Mortimer. *Permutation groups*, volume 163 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1996.
- [25] David S. Dummit and Richard M. Foote. *Abstract algebra*. Prentice Hall Inc., Englewood Cliffs, NJ, 1991.
- [26] Harold M. Edwards. *Galois theory*, volume 101 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1984.
- [27] Taher ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.
- [28] P. Erdős and A. Rényi. Probabilistic methods in group theory. *J. Analyse Math.*, 14:127–138, 1965.
- [29] P. Erdős and P. Turán. On some problems of a statistical group-theory. I. *Z. Wahrscheinlichkeitstheorie und Verw. Gebiete*, 4:175–186 (1965), 1965. Reprinted in Volume 3 of the collected papers of Paul Turán.
- [30] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.4*, 2004. (<http://www.gap-system.org>).
- [31] Daniel Gorenstein, Richard Lyons, and Ronald Solomon. *The classification of the finite simple groups*, volume 40 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 1994.
- [32] G. H. Hardy and E. M. Wright. *An introduction to the theory of numbers*. Oxford, at the Clarendon Press, 1954. 3rd ed.
- [33] G.H. Hardy and J.E. Littlewood. Some problems of "Partitio Numerorum"; III: on the expression of a number as a sum of primes. *Acta Math.*, 44:1–70, 1922.
- [34] Alexander Hulpke. Techniques for the computation of Galois groups. In *Algorithmic algebra and number theory (Heidelberg, 1997)*, pages 65–77. Springer, Berlin, 1999.
- [35] C. Jordan. Sur la limite du degré des groupes primitifs qui contiennent une substitution donnée. *Bull. Soc. Math. France*, 1873.
- [36] Donald E. Knuth. *The art of computer programming. Vol. 2*. Addison-Wesley Publishing Co., Reading, Mass., second edition, 1981. Seminumerical algorithms, Addison-Wesley Series in Computer Science and Information Processing.
- [37] Charles R. Leedham-Green. The computational matrix group project. In *Groups and computation, III (Columbus, OH, 1999)*, volume 8 of *Ohio State Univ. Math. Res. Inst. Publ.*, pages 229–247. de Gruyter, Berlin, 2001.
- [38] Martin W. Liebeck. On graphs whose full automorphism group is an alternating group or a finite classical group. *Proc. London Math. Soc. (3)*, 47(2):337–362, 1983.
- [39] Martin W. Liebeck. On minimal degrees and base sizes of primitive permutation groups. *Arch. Math. (Basel)*, 43(1):11–15, 1984.
- [40] Martin W. Liebeck, Cheryl E. Praeger, and Jan Saxl. On the O’Nan-Scott theorem for finite primitive permutation groups. *J. Austral. Math. Soc. Ser. A*, 44(3):389–396, 1988.
- [41] Gary L. Miller. Riemann’s hypothesis and tests for primality. *J. Comput. System Sci.*, 13(3):300–317, 1976. Working papers presented at the ACM-SIGACT Symposium on the Theory of Computing (Albuquerque, N.M., 1975).
- [42] Leo Moser and Max Wyman. On solutions of  $x^d = 1$  in symmetric groups. *Canad. J. Math.*, 7:159–168, 1955.
- [43] Alice Niemeyer and Cheryl Praeger. On permutations whose order divides a given quantity. In preparation.
- [44] Ivan Niven, Herbert S. Zuckerman, and Hugh L. Montgomery. *An introduction to the theory of numbers*. John Wiley & Sons Inc., New York, fifth edition, 1991.

- [45] Igor Pak. The product replacement algorithm is polynomial. In *41st Annual Symposium on Foundations of Computer Science (Redondo Beach, CA, 2000)*, pages 476–485. IEEE Comput. Soc. Press, Los Alamitos, CA, 2000.
- [46] Igor Pak. What do we know about the product replacement algorithm? In *Groups and computation, III (Columbus, OH, 1999)*, volume 8 of *Ohio State Univ. Math. Res. Inst. Publ.*, pages 301–347. de Gruyter, Berlin, 2001.
- [47] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley Publishing Company, Reading, MA, 1994.
- [48] Peter Ludwig Sylow. Théorèmes sur les groupes de Substitutions. *Mathematische Annalen*, 5:584 – 594, 1872.
- [49] Cheryl E. Praeger and Jan Saxl. On the orders of primitive permutation groups. *Bull. London Math. Soc.*, 12(4):303–307, 1980.
- [50] Cheryl E. Praeger and Aner Shalev. Bounds on finite quasiprimitive permutation groups. *J. Aust. Math. Soc.*, 71(2):243–258, 2001. Special issue on group theory.
- [51] L. Pyber. On the orders of doubly transitive permutation groups, elementary estimates. *J. Combin. Theory Ser. A*, 62(2):361–366, 1993.
- [52] Michael O. Rabin. Probabilistic algorithm for testing primality. *J. Number Theory*, 12(1):128–138, 1980.
- [53] Leonard L. Scott. Representations in characteristic  $p$ . In *The Santa Cruz Conference on Finite Groups (Univ. California, Santa Cruz, Calif., 1979)*, volume 37 of *Proc. Sympos. Pure Math.*, pages 319–331. Amer. Math. Soc., Providence, R.I., 1980.
- [54] Ákos Seress. *Permutation group algorithms*, volume 152 of *Cambridge Tracts in Mathematics*. Cambridge University Press, Cambridge, 2003.
- [55] Charles C. Sims. Computational methods in the study of permutation groups. In *Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967)*, pages 169–183. Pergamon, Oxford, 1970.
- [56] Charles C. Sims. Computation with permutation groups. In *Proceedings of the second ACM symposium on Symbolic and algebraic manipulation*, pages 23–28. ACM Press, 1971.
- [57] Gérald Tenenbaum. *Introduction to analytic and probabilistic number theory*, volume 46 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 1995. Translated from the second French edition (1995) by C. B. Thomas.
- [58] Nikolaj Tschebotareff. Die Bestimmung der Dichtigkeit einer Menge von Primzahlen welche zu einer gegebenen Substitutionklasse gehören. *Mathematische Annalen*, 95:191 – 228, 1929.
- [59] B.L. van der Waerden. Die Seltenheit der Gleichung mit Affekt. *Mathematische Annalen*, 109:13 – 16, 1934.
- [60] Richard Warlimont. Über die Anzahl der Lösungen von  $x^n = 1$  in der symmetrischen Gruppe  $S_n$ . *Arch. Math. (Basel)*, 30(6):591–594, 1978.
- [61] Helmut Wielandt. *Finite permutation groups*. Translated from the German by R. Bercov. Academic Press, New York, 1964.
- [62] Helmut Wielandt. Permutation groups through invariant relations and invariant functions. In *Lecture Notes, Ohio State University*, 1969.
- [63] Herbert S. Wilf. The asymptotics of  $e^{P(z)}$  and the number of elements of each order in  $S_n$ . *Bull. Amer. Math. Soc. (N.S.)*, 15(2):228–232, 1986.