2021

Counting Conjugacy Classes of Elements of Finite Order in Compact Exceptional Groups

Qidong He

Counting Conjugacy Classes of Elements of Finite Order in Compact Exceptional Groups

Qidong He

A thesis submitted in partial fulfillment of the requirements for the degree of

Bachelor of Arts with Honors

Examined and approved on

_____

by the following examiners:

_____

Dr. Tamar Friedmann (advisor)

_____

Dr. Fernando Gouvêa (reader)

Department of Mathematics and Statistics

Colby College

Acknowledgments

In temporal order, I thank (a) my parents for bringing me into this world and making me the person I am today, (b) my middle school teacher Dr. Guoshuang Pan for supporting me through my earliest journey in mathematics, (c) my high school teacher Angela Benjamin for helping me develop an appreciation of physics and the applied sciences, (d) Dr. Ben Mathes and Dr. Scott Taylor for providing me with an accessible and passionate introduction to mathematical research, (e) Dr. Otto Bretscher, Dr. Evan Randles, and Dr. Nora Youngs for their rich supply of mathematical jokes and helping me grow in mathematical maturity, and (f) Dr. Robert Bluhm and Dr. Kelly Patton for demonstrating to me the prospects for a mathematically rigorous formulation of theoretical physics.

I would particularly like to thank my thesis advisor Dr. Tamar Friedmann for suggesting the topic of this thesis and providing me with invaluable advice pertaining to both research and the application process for graduate programs as well as my academic advisor Dr. Fernando Gouvêa for guiding me in almost all aspects of my college life.

It is due to all of you that I will soon be embarking on a new adventure to pursue graduate studies in the theory and applications of mathematical physics.

Contents

Chapter 1

Introduction

Let G be the compact real form of a simple Lie group. Then, by the classification theorem of simple Lie groups, G is isomorphic to one of

1. the compact classical groups, i.e., $\mathrm{SU}(n+1)$ for $n \geq 1$, $\mathrm{SO}(2n+1)$ for $n \geq 2$, $\mathrm{Sp}(n)$ for $n \geq 3$, and $\mathrm{SO}(2n)$ for $n \geq 4$, or

2. the five compact exceptional groups, i.e., G_2, F_4, E_6, E_7, and E_8.

In this thesis, we study the number of conjugacy classes of elements of finite order in a compact exceptional group G. Specifically, given integers $m, s \geq 1$, we are interested in determining the number of conjugacy classes in G whose elements (1) have order dividing m, or (2) have s distinct eigenvalues in addition to having order dividing m. To put this more concisely, we follow the notation of [5] and define, for $m \geq 1$,

$$E(G, m) = \{x \in G : \ x^m = 1\}.$$

By viewing G as a matrix Lie group via its standard (i.e., smallest-degree faithful) representation and treating the eigenvalues of the matrix corresponding to each $x \in G$ as the eigenvalues of x, we are able to define

$$E(G, m, s) = \{x \in E(G, m) : \ x \text{ has } s \text{ distinct eigenvalues}\}.^1$$

Finally, define

$$N(G, m) = \text{number of conjugacy classes of } G \text{ in } E(G, m),$$

$$N(G, m, s) = \text{number of conjugacy classes of } G \text{ in } E(G, m, s).$$

We note that the preceding definitions can be readily extended, for instance, to the compact classical groups. Here, we intentionally omit such extensions in the interests of clarity.

[1] For $G = G_2, F_4, E_7, E_8$, we can alternatively define $E(G, m, s)$ as the elements in $E(G, m)$ with s distinct conjugate pairs of eigenvalues. Our algorithm can be easily modified to compute $N(G, m, s)$ under the alternative definition by adjusting the s-function in Definition 4.2.3.

The quantities $N(G, m)$ and $N(G, m, s)$ have been studied for various simple Lie groups in several different contexts. In [3, 4], $N(G, m)$ is computed for all simple Lie groups using certain high-powered machinery in Lie theory. In [5, 6], the study of $N(G, m)$ and $N(G, m, s)$ is motivated by an explicit enumeration problem in string theory; the authors give a purely combinatorial method to compute these quantities for the compact classical groups. The purpose of this thesis is to provide a unified combinatorial method to compute both quantities in the case that G is a compact exceptional group.[2]

This thesis is organized as follows. In Chapter 2, we provide an overview of certain results from Lie theory, combinatorics, and linear algebra that will become relevant in the later chapters. In particular, we seek to provide clues as to how the results presented therein will be used later on, whenever and wherever appropriate. In Chapter 3, we use the smallest exceptional simple Lie group G_2 as a motivating example to introduce several ideas central to the formulation of our general method. Owing to the great tractability of G_2, we are able to provide a large portion of the computational details for $N(G_2, m)$ and $N(G_2, m, s)$ within a reasonable number of pages. In Chapter 4, we consolidate the results and ideas from the previous chapters to develop our general algorithm. The appendix includes the raw code for an implementation of our method in Sage. We give the full results for $N(G, m)$, where $G = F_4, E_6, E_7, E_8$, as well as partial results for $N(F_4, m, s)$ obtained using the program.[3] Finally, in Chapter 5, we give a brief overview of the sort of practical issues that may arise from implementing the algorithm and suggest ways to deal with said issues.

Throughout our writing, we assume that the reader is familiar with the basic concepts and results in Lie theory. Occasionally, we may provide the reader with refreshers on certain fundamental ideas, but we do so only when we believe that such discussions can fit nicely into our narrative. Furthermore, when the exact technical details are of secondary concern to our discussion, we may choose to omit a proof, to simplify the conditions of a theorem as requiring certain objects to be "nice," etc.

[2] As a side note for the attentive reader, we explain in Section 3.3 why the simple combinatorial treatment of the problem used in [5, 6] cannot be applied to the compact exceptional groups without any modification.

[3] While the program is capable of producing the full results in approximately 30 minutes on a personal laptop, the table that it produces is organized by m modulo 12252240, which precludes any effort to include the full table as a part of this thesis.

Chapter 2

Preliminaries

In this chapter, we present select definitions and results from Lie theory, combinatorics, and linear algebra that will be applied repeatedly in the later chapters.

2.1 Lie Theory

Generally speaking, it requires an incredible amount of hard work to explicitly describe a compact exceptional group. The main reason is that such endeavors frequently necessitate the use of certain algebraic structures built upon the non-associative octonion algebra $\mathbb{O}$ [9]. For this reason, studying the conjugacy classes in a compact exceptional group from the perspective of an explicit description will likely pose too great a logistical challenge, and it behooves us to seek out an alternative approach.

In this section, we observe that certain topological properties possessed by a compact exceptional group allow us to focus our attention entirely on one of its combinatorically well-behaved subgroups, known as a maximal torus. We also explain how we can straightforwardly obtain a desired matrix representation of a maximal torus from the implementation of the results of [8] in Sage. As noted in [9], every compact exceptional group is compact, connected, and simply connected. Hence, while the definitions and results that follow may be directed at generic Lie groups that satisfy certain properties, it should be understood that all of them apply, in particular, to the compact exceptional groups.

The definitions and propositions that follow are taken from [7].

Definition 2.1.1 (maximal torus)**.** Let K be a compact, connected matrix Lie group. A subgroup T of K is a *torus* if $T \cong \mathrm{U}(1)^k$ for some $k \in \mathbb{N}$. If T is a torus that is not properly contained in any other torus in K, then T is called a *maximal torus* of K.

The following proposition implies that every conjugacy class in a compact exceptional group intersects with a fixed maximal torus. Hence, there is a correspondence between the conjugacy classes in the group and the elements of a (fixed) maximal torus.

Proposition 2.1.2. *Let K be a compact, connected matrix Lie group, and $T \leq K$ a maximal torus. Then, every $y \in K$ can be written as*

$$y = xtx^{-1}$$

for some $x \in K$ and $t \in T$.

A natural question to ask is when do two elements in a maximal torus correspond to the same conjugacy class. The answer is that this happens if and only if the elements themselves are conjugate by an element of the finite group known as the Weyl group.

Definition 2.1.3 (Weyl group)**.** Let K be a compact, connected matrix Lie group, and $T \leq K$ a maximal torus. The *Weyl group* of K (with respect to T) is the quotient group

$$W = N(T)/T,$$

where $N(T) = \{x \in K : xTx^{-1} = T\}$ is the *normalizer* of T.

Proposition 2.1.4. *Let K be a compact, connected matrix Lie group, and $T \leq K$ a maximal torus. Then, the corresponding Weyl group W is a finite group and acts on T by conjugation (cf. Definition 2.2.1). Moreover, if $t \in T$ satisfies $t = xsx^{-1}$ for some $x \in K$ and $s \in T$, then there exists $w \in W$ such that $s = w \cdot t$.*

To recapitulate, in order to count the conjugacy classes in a compact exceptional group, it is not necessary to work with (a matrix representation of) the full group. Rather, it suffices to keep at our disposal (the corresponding matrix representations of) a maximal torus and its associated Weyl group. Accordingly, we redirect our attention to the task of obtaining said matrix representations. Our strategy is to exploit the construction of matrix representations of the exceptional Lie algebras in [8].

The next proposition tells us that, to obtain a matrix representation of a compact exceptional group, it suffices to exponentiate a matrix representation of its Lie algebra.

Proposition 2.1.5. *If K is a compact, connected matrix Lie group, then the exponential map for K is surjective.*

It is now straightforward to obtain a maximal torus T of K. By the following proposition, we need only exponentiate a maximal commutative subalgebra $\mathfrak{t}$ of the Lie algebra $\mathfrak{k}$ of K.

Proposition 2.1.6. *Let K be a compact, connected matrix Lie group with Lie algebra $\mathfrak{k}$. If T is a maximal torus of K, then the Lie algebra $\mathfrak{t}$ of T is a maximal commutative subalgebra of $\mathfrak{k}$. Conversely, if $\mathfrak{t}$ is a maximal commutative subalgebra of $\mathfrak{k}$, then the connected Lie subgroup T of K with Lie algebra $\mathfrak{t}$ is a maximal torus of K.*

In comparison, it requires significantly more theoretical preparation to describe rigorously and comprehensively how we can obtain a matrix representation of the Weyl group as a quotient subgroup of K from the Lie algebra $\mathfrak{k}$. Since this can easily distract us from our main objective, we present here only the essential points of the reasoning process. We refer the curious reader to [2, 7] for full details.

Since a compact exceptional group G is simply connected, the *complexification* of its Lie algebra, denoted by $\mathfrak{g}_{\mathbb{C}}$, is *semisimple*. The semisimple Lie algebra $\mathfrak{g}_{\mathbb{C}}$ is associated with a finite collection of vectors in Euclidean space that constitute its *root system*, denoted by R. R is associated with a group of symmetries that is, not coincidentally, also called the *Weyl group*. Here, the Weyl group is generated by reflections across the hyperplanes orthogonal

to the *fundamental roots*, which are a subset $\Delta \subseteq R$ that satisfies special properties. It turns out that the Weyl group in the context of the associated root system is isomorphic to the Weyl group of the original compact exceptional group. An explicit description of this group isomorphism can be found in Chapter 6 of [2].

We record in Table 2.1 the matrix representations of a maximal torus and the corresponding Weyl group for each of the five compact exceptional groups, using the notation of [8].

<table>
<tr><th>G</th><th>Δ</th><th>T</th><th>W</th></tr>
<tr><td>G_2</td><td>$\{A, B\}$</td><td rowspan="2">$\{\prod_{\alpha\in\Delta} \exp(2\pi i x_\alpha[\psi(e_\alpha), \psi(e_{-\alpha})]) \mid x_\alpha \in \mathbb{R}\}$</td><td rowspan="5">$\langle n_\alpha(1) \cdot T \mid \alpha \in \Delta\rangle$</td></tr>
<tr><td>F_4</td><td>$\{A, B, C, D\}$</td></tr>
<tr><td>E_6</td><td>$\{a, b, \ldots, f\}$</td><td rowspan="3">$\{\prod_{\alpha\in\Delta} \exp(2\pi i x_\alpha[\phi(e_\alpha), \phi(e_{-\alpha})]) \mid x_\alpha \in \mathbb{R}\}$</td></tr>
<tr><td>E_7</td><td>$\{a, b, \ldots, g\}$</td></tr>
<tr><td>E_8</td><td>$\{a, b, \ldots, h\}$</td></tr>
</table>

Table 2.1: Matrix representations of a maximal torus and the corresponding Weyl group in each compact exceptional group, recorded here using the notation of [8].

2.2 Combinatorics

As we have hinted at in Section 2.1, the combinatorics of conjugacy classes in a compact exceptional group is intimately connected to the conjugation action of the Weyl group on a maximal torus. In this section, we formally define a group action and present a famous group-theoretic result known as Burnside's Lemma, which provides important combinatorial information on a group action and will become the guiding theme of our algorithm. We end the section with a brief introduction to Möbius inversion on a poset.

Definition 2.2.1 (group action)**.** Let G be a group with the identity element e and X a set. A *(left) group action* of G on X is a function $\alpha : G \times X \to X$ satisfying

$$\alpha(e, x) = x$$

for all $x \in X$, and

$$\alpha(g, \alpha(h, x)) = \alpha(gh, x)$$

for all $g, h \in G$ and $x \in X$. If α is a group action of G on X, we say that G acts on X and abbreviate $\alpha(g, x)$ as $g \cdot x$ for simplicity.

A group action induces a partition of the set.

Definition 2.2.2 (orbit)**.** Let G be a group that acts on a set X. Given $x \in X$, the *orbit* of x under the group action of G is defined as

$$G \cdot x := \{g \cdot x : g \in G\}.$$

We denote by X/G the collection of orbits of X under the action of G.

Proposition 2.2.3. *If G acts on X, then the orbits of X under this group action constitute a partition of X.*

We can now formalize the correspondence between the conjugacy classes in a compact exceptional group G and the elements in a maximal torus T. Together, Propositions 2.1.2 and 2.1.4 indicate a one-to-one correspondence between the conjugacy classes in G and the orbits of T under the (conjugation) action of the Weyl group W. Since the order and eigenvalues of any element $x \in G$ are invariant under conjugation, we conclude that

1. $N(G, m)$ is the number of orbits of elements in T with order dividing m under the action of W, and
2. $N(G, m, s)$ is the number of orbits of elements in T with order dividing m and s distinct eigenvalues under the action of W.

The central tool for counting these orbits that we will use later on is Burnside's Lemma, which relates the number of orbits under a group action to the number of set elements that are unaffected by each element of the group.

Theorem 2.2.4 (Burnside's Lemma)**.** *Let G be a finite group that acts on a set X. Then, the number of orbits of X under the action of G is*

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |\mathrm{Fix}(g)| \,,$$

where $\mathrm{Fix}(g) = \{x \in X : g \cdot x = x\}$ *is the set of* fixed points *of X under the action of g.*

We can simplify the sum in Burnside's Lemma using the following result.

Proposition 2.2.5. *Let G be a finite group that acts on a set X. If $g, h \in G$ are conjugate, then $|\mathrm{Fix}(g)| = |\mathrm{Fix}(h)|$.*

Theorem 2.2.6 (Burnside's Lemma, reformulated)**.** *Let G be a finite group that acts on a set X. Denote by $\mathrm{Cl}(G)$ the collection of conjugacy classes in G. Then, the number of orbits of X under the action of G is*

$$|X/G| = \frac{1}{|G|} \sum_{c \in \mathrm{Cl}(G)} |c| \, |\mathrm{Fix}(g_c)| \,,$$

where $g_c \in c$ is a representative of the conjugacy class.

The utility of the reformulated Burnside's Lemma stems from the fact that while the Weyl group of a compact exceptional group may be forbiddingly large, the number of conjugacy classes inside the Weyl group is generally small. Hence, provided that we know the size and a representative of each conjugacy class in the Weyl group, the sum in Theorem 2.2.6 is much easier to evaluate than the one in Theorem 2.2.4. Fortunately, this information has been completely determined in [1] and translated into an accessible form in GAP 3, which we will use to our advantage.

Finally, we take a detour into the realm of posets (partially ordered sets). The reason for their introduction is roughly as follows. In order to compute $N(G, m, s)$, we will partition the set $E(T, m)$ by the "type" of repeats of eigenvalues that occur within each element. We will then impose a partial order on the subsets based on how "restrictive" the corresponding types are. This, in turn, will enable us to use the Möbius inversion to determine the number of elements of $E(T, m, s)$ "exclusive" to each type, which we can then use to formulate an expression for $N(G, m, s)$.

Definition 2.2.7 (poset)**.** A *partial order* on a set P is a binary order $\preceq$ on P such that, for all $x, y, z \in P$,

1. (reflexivity) $x \preceq x$;
2. (antisymmetry) if $x \preceq y$ and $y \preceq x$, then $x = y$; and
3. (transitivity) if $x \preceq y$ and $y \preceq z$, then $x \preceq z$.

When equipped with a partial order $\preceq$, the set P is called a *poset*.

Theorem 2.2.8 (Möbius inversion)**.** *Let $(P, \preceq)$ be a "nice" poset, and $f, g : P \to \mathbb{R}$. Then,*

$$g(t) = \sum_{s \preceq t} f(s) \text{ for all } t \in P$$

if and only if

$$f(t) = \sum_{s \preceq t} g(s)\mu(s, t) \text{ for all } t \in P.$$

Here, $\mu(s, t)$ is known as the Möbius function *of P and is computed by*

$$\mu(s, u) = \begin{cases} 1, & \text{if } s = u \\ -\sum_{s \preceq t \prec u} \mu(s, t), & \text{if } s \prec u \\ 0, & \text{else} \end{cases}. \tag{2.1}$$

2.3 Linear Algebra

In this section, we introduce two normal forms for rectangular matrices with integer entries, the Hermite normal form and the Smith normal form, which will function as useful computational devices in our algorithm. Both rely on the notion of unimodularity, which refers to the invertibility of a square integer matrix over $\mathbb{Z}$.

Given positive integers m, n, let $\mathrm{M}_m(\mathbb{Z})$ (resp. $\mathrm{M}_{m \times n}(\mathbb{Z})$) denote the set of $m \times m$ (resp. $m \times n$) matrices with integer entries.

Definition 2.3.1 (unimodularity)**.** A matrix $P \in \mathrm{M}_m(\mathbb{Z})$ is *unimodular* if $\det P = \pm 1$, or, equivalently, if P is invertible over $\mathbb{Z}$.

The main utility of the Hermite normal form in our context is for optimizing the determination of the types of repeats of eigenvalues that can occur in an element of $E(T, m, s)$ (cf. the discussion preceding Definition 2.2.7). It is characterized by the following theorem.

Theorem 2.3.2. *If $A \in \mathrm{M}_{m\times n}(\mathbb{Z})$, then there exists a unimodular matrix $P \in \mathrm{M}_n(\mathbb{Z})$ such that AP satisfies the following properties:*

1. *AP is lower triangular, and any column of zeros is located to the right of all the nonzero columns;*
2. *the pivot of any nonzero column of AP is positive and strictly below the pivot of any column to its left; and*
3. *any entry to the right of a pivot is zero, and any entry to the left of a pivot is nonnegative and strictly smaller than the pivot.*

Definition 2.3.3 (Hermite normal form)**.** Let $A \in \mathrm{M}_{m\times n}(\mathbb{Z})$. The *(column) Hermite normal form* of A, denoted by $\mathrm{HNF}(A)$, is the matrix AP in the statement of Theorem 2.3.2.

The optimization makes use of integer lattices, which we define below.

Definition 2.3.4 (lattice generated by a matrix)**.** Let $A \in \mathrm{M}_{m\times n}(\mathbb{Z})$. The *lattice* generated by A is the set

$$\Lambda(A) = \{A\mathbf{v} : \mathbf{v} \in \mathbb{Z}^n\} \subseteq \mathbb{Z}^m.$$

The Hermite normal form gives a necessary and sufficient condition for when two integer matrices, not necessarily of the same size, generate the same integer lattice.

Proposition 2.3.5. *Let $A \in \mathrm{M}_{m\times l}(\mathbb{Z})$ and $B \in \mathrm{M}_{n\times l}(\mathbb{Z})$. Then, $\Lambda(A) = \Lambda(B)$ if and only if $\mathrm{HNF}(A)$ and $\mathrm{HNF}(B)$ are or can be identified by adding or removing columns of zeros.*

Now, we introduce the Smith normal form, which is characterized by the following theorem.

Theorem 2.3.6. *If $A \in \mathrm{M}_{m\times n}(\mathbb{Z})$, then there exist unimodular matrices $P \in \mathrm{M}_m(\mathbb{Z})$ and $Q \in \mathrm{M}_n(\mathbb{Z})$ such that*

$$PAQ = \begin{bmatrix} d_1 & & & & \\ & \ddots & & & \\ & & d_r & & \\ & & & 0 & \\ & & & & \ddots \end{bmatrix},$$

where each $d_i \neq 0$ and $d_i \mid d_j$ whenever $i \leq j$. In particular, the d_i's are unique up to a sign.

Definition 2.3.7 (Smith normal form)**.** Let $A \in \mathrm{M}_{m\times n}(\mathbb{Z})$. The *Smith normal form* of A, denoted by $\mathrm{SNF}(A)$, is the matrix

$$\begin{bmatrix} d_1 & & & & \\ & \ddots & & & \\ & & d_r & & \\ & & & 0 & \\ & & & & \ddots \end{bmatrix}$$

associated to A in the statement of Theorem 2.3.6. Further, the nonzero integers d_i are called the *elementary divisors* of A.

The Smith normal form enables us to count certain objects inside the solution set of a system of linear equations without needing to solve the system at all. In our context, such objects may correspond to the fixed points in $E(T, m)$ under the action of an element of the Weyl group, or the elements in $E(T, m)$ that are characterized by having a particular type of repeats of eigenvalues.

Proposition 2.3.8. *Let $A \in \mathrm{M}_{n\times l}(\mathbb{Z})$. For any positive integer m, the map between the left kernel over $\frac{1}{m}\mathbb{Z}/\mathbb{Z}$ of A and that of $\mathrm{SNF}(A) = PAQ$ given by*

$$\begin{aligned} \phi : \operatorname{coker} A &\to \operatorname{coker} \mathrm{SNF}(A) \\ \mathbf{v} &\mapsto \mathbf{v}P^{-1} \end{aligned}$$

is a bijection.

Corollary 2.3.9. *Let $A \in \mathrm{M}_{n\times l}(\mathbb{Z})$. For any positive integer m, the size of the left kernel of A over $\frac{1}{m}\mathbb{Z}/\mathbb{Z}$ is given by*

$$|\operatorname{coker} A| = m^{l-r} \cdot \prod_{i=1}^{r} \gcd(d_i, m),$$

where $d_1, \ldots, d_r$ are the elementary divisors of A.

Chapter 3

A Gentle Example: G_2

In this chapter, we piece together the ideas introduced in Chapter 2 to compute $N(G, m)$ and $N(G, m, s)$ for the smallest compact exceptional group, G_2.

First, we obtain the matrix representations of a maximal torus T in G_2 and the generators of its corresponding Weyl group W. Referring to Table 2.1, we find that

$$T = \left\{ \begin{bmatrix} e^{2\pi i x_A} & & & & & & & \\ & e^{-2\pi i(x_A - x_B)} & & & & & & \\ & & e^{2\pi i(2x_A - x_B)} & & & & & \\ & & & 1 & & & & \\ & & & & e^{-2\pi i(2x_A - x_B)} & & & \\ & & & & & e^{2\pi i(x_A - x_B)} & & \\ & & & & & & e^{-2\pi i x_A} \end{bmatrix} \right\},$$

$$n_A(1) = \begin{bmatrix} & 1 & & & & & \\ -1 & & & & & & \\ & & & 1 & & & \\ & & -1 & & & & \\ & 1 & & & & & \\ & & & & & 1 & \\ & & & & -1 & & \end{bmatrix}, \text{ and } n_B(1) = \begin{bmatrix} 1 & & & & & & \\ & & 1 & & & & \\ & -1 & & & & & \\ & & & 1 & & & \\ & & & & & 1 & \\ & & & & -1 & & \\ & & & & & & 1 \end{bmatrix},$$

using the expressions for $\psi(e_{\pm\alpha})$ computed in [8], where $\alpha \in \{A, B\}$.

Remark 3.0.1. Note that an element

$$t(x_A, x_B) = \begin{bmatrix} e^{2\pi i x_A} & & & & & & & \\ & e^{-2\pi i(x_A - x_B)} & & & & & & \\ & & e^{2\pi i(2x_A - x_B)} & & & & & \\ & & & 1 & & & & \\ & & & & e^{-2\pi i(2x_A - x_B)} & & & \\ & & & & & e^{2\pi i(x_A - x_B)} & & \\ & & & & & & e^{-2\pi i x_A} \end{bmatrix} \in T$$

has order dividing m if and only if $(x_A, x_B) \in \left(\frac{1}{m}\mathbb{Z}\right)^2$. Since two pairs $(x_A, x_B), (x'_A, x'_B) \in$

$\left(\frac{1}{m}\mathbb{Z}\right)^2$ represent the same element of the maximal torus if and only if $(x_A - x'_A, x_B - x'_B) \in \mathbb{Z}^2$, we will henceforth require that $(x_A, x_B) \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2$.

3.1 The Computation of $N(G_2, m)$

By the discussion preceding Theorem 2.2.4, $N(G_2, m)$ is the number of orbits under the action of W of the set

$$\left\{ t(x_A, x_B) : (x_A, x_B) \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2 \right\}.$$

It then follows from Theorem 2.2.6 that

$$\boxed{N(G_2, m) = \frac{1}{|W|} \sum_{c \in \mathrm{Cl}(W)} |c| \, |\mathrm{Fix}(w_c)|,}$$

where $\mathrm{Cl}(W)$ is the collection of conjugacy classes in W, and $w_c \in c$ is a conjugacy class representative.

The results of [1] about the conjugacy classes in W can be accessed in GAP 3 using the following command:

```
gap> ChevieClassInfo(CoxeterGroup("G",2));
rec(
  classtext := [ [  ], [ 2 ], [ 1 ], [ 1, 2 ], [ 1, 2, 1, 2 ],
     [ 1, 2, 1, 2, 1, 2 ] ],
  classnames :=
   [ "A_0", "\\tilde A_1", "A_1", "G_2", "A_2", "A_1+\\tilde A_1" ],
  classparams := [ [ "A_0" ], [ "\\tilde A_1" ], [ "A_1" ], [ "G_2" ],
     [ "A_2" ], [ "A_1+\\tilde A_1" ] ],
  orders := [ 1, 2, 2, 6, 3, 2 ],
  classes := [ 1, 3, 3, 2, 2, 1 ] )
```

Of particular interest to us are `classes` and `classtext`, which indicate the size and a representative of each conjugacy class in W. See Table 3.1.[1]

It remains to determine $|\mathrm{Fix}(w_c)|$ for each conjugacy class representative w_c. Before we begin, we note a necessary and sufficient condition for $t(x_A, x_B)$ to be fixed by some $w \in W$.

Proposition 3.1.1. *Let $w \in W$. Acting on $t(x_A, x_B)$ by w results in a signed permutation of the exponents in its first three diagonal entries. Further, $t(x_A, x_B)$ is fixed by w if and only if $w \cdot t(x_A, x_B)$ and $t(x_A, x_B)$ agree in their first two diagonal entries.*

As an example, we consider $w_c = (n_B(1)n_A(1))^2 \cdot T$.

[1] We need to be cautious when using the generators $n_A(1)$ and $n_B(1)$ in conjunction with the information in `classtext`, since they rely on different labelings of the Dynkin diagram of $\mathfrak{g}_2$. For more details, see Remark 4.1.1.

$\lvert c\rvert$	Representative $w_c \in c$
1	$I \cdot T$
3	$n_A(1) \cdot T$
3	$n_B(1) \cdot T$
2	$n_B(1)n_A(1) \cdot T$
2	$(n_B(1)n_A(1))^2 \cdot T$
1	$(n_B(1)n_A(1))^3 \cdot T$

Table 3.1: The size and a representative of each conjugacy class c in the Weyl group of G_2.

Example 3.1.2 (computing $|\mathrm{Fix}((n_B(1)n_A(1))^2 \cdot T)|$)**.** Conjugating

$$t(x_A, x_B) = \begin{bmatrix} e^{2\pi i x_A} & & & & & & & \\ & e^{-2\pi i(x_A - x_B)} & & & & & & \\ & & e^{2\pi i(2x_A - x_B)} & & & & & \\ & & & 1 & & & & \\ & & & & e^{-2\pi i(2x_A - x_B)} & & & \\ & & & & & e^{2\pi i(x_A - x_B)} & & \\ & & & & & & e^{-2\pi i x_A} & \end{bmatrix}$$

by

$$w = (n_B(1)n_A(1))^2 = \begin{bmatrix} & & 1 & & & & \\ & & & & & & 1 \\ & 1 & & & & & \\ & & & 1 & & & \\ & & & & & -1 & \\ -1 & & & & & & \\ & & 1 & & & & \end{bmatrix} \in (n_B(1)n_A(1))^2 \cdot T$$

gives

$$\begin{aligned} &wt(x_A, x_B)w^{-1} \\ &= \begin{bmatrix} e^{-2\pi i(2x_A - x_B)} & & & & & & & \\ & e^{-2\pi i x_A} & & & & & & \\ & & e^{-2\pi i(x_A - x_B)} & & & & & \\ & & & 1 & & & & \\ & & & & e^{2\pi i(x_A - x_B)} & & & \\ & & & & & e^{2\pi i x_A} & & \\ & & & & & & e^{2\pi i(2x_A - x_B)} & \end{bmatrix}. \end{aligned}$$

By Proposition 3.1.1, $|\mathrm{Fix}(w)|$ is the number of solutions to the system of linear equations

$$\begin{cases} x_A = -(2x_A - x_B) \\ -(x_A - x_B) = -x_A \end{cases}$$

over $\frac{1}{m}\mathbb{Z}/\mathbb{Z}$. We can express the above system of equations using matrices as

$$\begin{bmatrix} x_A & x_B \end{bmatrix} \begin{bmatrix} 3 & 0 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}.$$

The Smith normal form of the coefficient matrix is

$$\mathrm{SNF}\left(\begin{bmatrix} 3 & 0 \\ -1 & 1 \end{bmatrix}\right) = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}.$$

Hence, it follows from Corollary 2.3.9 that

$$|\mathrm{Fix}((n_B(1)n_A(1))^2 \cdot T)| = \gcd(1,m)\gcd(3,m) = \begin{cases} 3 & m \equiv 0 \bmod 3 \\ 1 & m \not\equiv 0 \bmod 3 \end{cases}.$$

$w_c \in c$	Associated System	SNF	$\lvert\mathrm{Fix}(w_c)\rvert$
$I \cdot T$	$\begin{bmatrix} x_A & x_B \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	m^2
$n_A(1) \cdot T$	$\begin{bmatrix} x_A & x_B \end{bmatrix}\begin{bmatrix} 2 & 2 \\ -1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$	m
$n_B(1) \cdot T$	$\begin{bmatrix} x_A & x_B \end{bmatrix}\begin{bmatrix} 0 & 3 \\ 0 & -2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$	m
$n_B(1)n_A(1) \cdot T$	$\begin{bmatrix} x_A & x_B \end{bmatrix}\begin{bmatrix} 2 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	1
$(n_B(1)n_A(1))^2 \cdot T$	$\begin{bmatrix} x_A & x_B \end{bmatrix}\begin{bmatrix} 3 & 0 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$	$\begin{cases} 3 & m \equiv 0 \bmod 3 \\ 1 & m \not\equiv 0 \bmod 3 \end{cases}$
$(n_B(1)n_A(1))^3 \cdot T$	$\begin{bmatrix} x_A & x_B \end{bmatrix}\begin{bmatrix} 2 & 2 \\ 0 & -2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$	$\begin{cases} 4 & m \equiv 0 \bmod 2 \\ 1 & m \not\equiv 0 \bmod 2 \end{cases}$

Table 3.2: The number of fixed points in $\left\{t(x_A, x_B) : (x_A, x_B) \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2\right\}$ under the action of each conjugacy class $c \in \mathrm{Cl}(W)$ as represented by $w_c \in c$.

Repeating this for the other conjugacy classes in W yields Table 3.2. Finally, combining the results on $|c|$ and $|\mathrm{Fix}(w_c)|$ using the equation

$$N(G_2, m) = \frac{1}{|W|} \sum_{c \in \mathrm{Cl}(W)} |c| \, |\mathrm{Fix}(w_c)|,$$

we obtain the following theorem:

Theorem 3.1.3. *For any positive integer m,*

$m \bmod 6$	$N(G_2, m)$
0	$(m^2+6m+12)/12$
± 1	$(m^2+6m+5)/12$
± 2	$(m^2+6m+8)/12$
3	$(m^2+6m+9)/12$

3.2 The Computation of $N(G_2, m, s)$

We approach this more delicate enumeration problem in the same way as before. By Theorem 2.2.6 and the discussion preceding Theorem 2.2.4, we have

$$\boxed{N(G_2, m, s) = \frac{1}{|W|} \sum_{c \in \mathrm{Cl}(W)} |c| \, |\mathrm{Fix}(w_c)| \,,} \tag{3.1}$$

where $|\mathrm{Fix}(w_c)|$ is the number of fixed points of the set

$$\left\{ t(x_A, x_B) : (x_A, x_B) \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2, \; t(x_A, x_B) \text{ has } s \text{ distinct eigenvalues} \right\}$$

under the action of the conjugacy class representative $w_c \in c$.

Whereas the requirement that $(x_A, x_B) \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2$ is easy to account for, it is not quite clear at first glance as to how we can isolate such pairs (x_A, x_B) that give rise to s distinct eigenvalues in $t(x_A, x_B)$. Indeed, upon inspecting the set of eigenvalues of $t(x_A, x_B)$,

$$\left\{ 1, e^{\pm 2\pi i x_A}, e^{\pm 2\pi i (x_A - x_B)}, e^{\pm 2\pi i (2x_A - x_B)} \right\}, \tag{3.2}$$

we may observe that the eigenvalues of $t(x_A, x_B)$ are coupled in the sense that each of the parameters, x_A and x_B, affects more than one conjugate pair of eigenvalues. Moreover, we must pay special attention to when a pair (x_A, x_B) reduces one or more conjugate pairs to 1, from which no new eigenvalue arises (since 1 is already an eigenvalue of $t(x_A, x_B)$), or to -1, from which only one new eigenvalue arises (i.e., -1).

3.2.1 Determining the Number of distinct eigenvalues of $t(x_A, x_B)$

For clarity, define

$$\begin{aligned} P_1(x_A, x_B) &= x_A, \\ P_2(x_A, x_B) &= -(x_A - x_B), \text{ and} \\ P_3(x_A, x_B) &= 2x_A - x_B. \end{aligned}$$

We make the following observation.

Proposition 3.2.1. *A pair* $(x_A, x_B) \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2$ *leads to one or more repeats in* (3.2) *if and only if*

1. $P_i = \pm P_j$ *for some* $i \neq j$ *(whence* $e^{\pm 2\pi i P_i}$ *coincide with* $e^{\pm 2\pi i P_j}$*),*
2. $P_i = 0$ *for some* i *(whence* $e^{\pm 2\pi i P_i} = 1$*), or*
3. $2P_i = 0$ *but* $P_i \neq 0$ *for some* i *(i.e.,* $P_i = \frac{1}{2}$*, whence* $e^{\pm 2\pi i P_i} = -1$*).*

We translate this into the language of matrices and kernels as follows. Corresponding to each P_i, where $i = 1, 2, 3$, we define

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ \mathbf{v}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \text{ and } \mathbf{v}_3 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}.$$

By stacking together the distinct vectors (up to a sign) in the list

$$\{\mathbf{v}_1 \pm \mathbf{v}_2, \mathbf{v}_1 \pm \mathbf{v}_3, \mathbf{v}_2 \pm \mathbf{v}_3, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, 2\mathbf{v}_1, 2\mathbf{v}_2, 2\mathbf{v}_3\},$$

we form the matrix

$$P = \begin{bmatrix} 1 & 1 & 2 & 2 & 0 & 3 & 2 & 3 & 4 \\ 0 & -1 & -1 & 0 & 1 & -1 & -2 & -2 & -2 \end{bmatrix}.^2$$

Then, Proposition 3.2.1 implies the following.

Proposition 3.2.2. *A pair* $(x_A, x_B) \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2$ *leads to one or more repeats in* (3.2) *if and only if* (x_A, x_B) *is in the left kernel of some* $2 \times k$ *submatrix of* P*, where* $k \geq 1$*.*

Example 3.2.3. Consider $m = 3$ and the submatrix

$$Q = \begin{bmatrix} 0 & 3 & 3 \\ 1 & -1 & -2 \end{bmatrix}$$

of P. It can be verified that $\left(\frac{1}{3}, 0\right) \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2$ is in the left kernel of Q. Observe that

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \mathbf{v}_1 + \mathbf{v}_2, \ \begin{bmatrix} 3 \\ -1 \end{bmatrix} = \mathbf{v}_1 + \mathbf{v}_3, \text{ and } \begin{bmatrix} 3 \\ -2 \end{bmatrix} = -\mathbf{v}_2 + \mathbf{v}_3.$$

By the correspondence between $\mathbf{v}_i$ and P_i for each i, we have that

$$P_1\left(\frac{1}{3}, 0\right) = -P_2\left(\frac{1}{3}, 0\right), \ P_1\left(\frac{1}{3}, 0\right) = -P_3\left(\frac{1}{3}, 0\right), \text{ and } P_2\left(\frac{1}{3}, 0\right) = P_3\left(\frac{1}{3}, 0\right).$$

Thus, in (3.2) with $(x_A, x_B) = \left(\frac{1}{3}, 0\right)$,

$$e^{\pm 2\pi i x_A}, \ e^{\pm 2\pi i (x_A - x_B)}, \text{ and } e^{\pm 2\pi i (2x_A - x_B)}$$

[2]Since we are mostly interested in the left kernel of submatrices of P over $\frac{1}{m}\mathbb{Z}/\mathbb{Z}$, we are free to change the sign of any column of P as we see fit. Here, for aesthetic reasons, we require that the top nonzero element in each column of P be positive.

all coincide. It can also be verified that Q is the largest submatrix of P having $\left(\frac{1}{3}, 0\right)$ in its left kernel. Hence, we have fully captured the manner (in the sense of Proposition 3.2.1) by which the eigenvalues of $t\left(\frac{1}{3}, 0\right)$ repeat. We conclude that

$$\left\{1, e^{\pm 2\pi i x_A}\right\}$$

is the irredundant list of eigenvalues of $t\left(\frac{1}{3}, 0\right)$. Thus, $t\left(\frac{1}{3}, 0\right)$ has three distinct eigenvalues.

Our next step is to generalize the idea used in Example 3.2.3 to a method for determining the number of distinct eigenvalues of $t(x_A, x_B)$ based on the interaction between (x_A, x_B) and P. For convenience, we will treat $\left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2$ as the left kernel of the unique 2×0 submatrix of P.

Let S be a $2 \times k$ submatrix of P, where $0 \leq k \leq 9$, and suppose that $(x_A, x_B) \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2$ is in the left kernel of S. If (x_A, x_B) is not in the left kernel of any other submatrix of P properly containing S, then S completely characterizes the manner (in the sense of Proposition 3.2.1) by which the eigenvalues of $t(x_A, x_B)$ repeat. In this case, we define an equivalence relation $\equiv_S$ on the set

$$H = \left\{0, \frac{1}{2}, 1, 2, 3\right\}$$

by the rules:

1. $i \equiv_S j$ if $\mathbf{v}_i \pm \mathbf{v}_j$ is (up to a sign) a column vector of S;
2. $i \equiv_S 0$ if $\mathbf{v}_i$ is (up to a sign) a column vector of S; and
3. $i \equiv_S \frac{1}{2}$ if $2\mathbf{v}_i$ is (up to a sign) a column vector of S but $\mathbf{v}_i$ is not.

The number of distinct eigenvalues of $t(x_A, x_B)$ can then be expressed in terms of the number of equivalence classes of the set under $\equiv_S$.

Proposition 3.2.4. *Let S be a $2 \times k$ submatrix of P. Suppose that there exists $(x_A, x_B) \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2$ which is in the left kernel of S but not the left kernel of any other submatrix of P properly containing S. Then, the number of distinct eigenvalues of $t(x_A, x_B)$ is*

$$\begin{cases} 2\left|H/\equiv_S\right| - 3 & \text{if } \left|\left[\frac{1}{2}\right]\right| = 1 \\ 2\left|H/\equiv_S\right| - 2 & \text{else} \end{cases}. \tag{3.3}$$

We strengthen Proposition 3.2.4 as follows.

Definition 3.2.5. Let $\mathcal{S}$ be the set of $2 \times k$ submatrices of P, where $0 \leq k \leq 9$. We define a partial order $\preceq$ on $\mathcal{S}$ by $S_1 \preceq S_2$ if and only if S_2 is a submatrix of S_1.

Remark 3.2.6. By viewing each $S \in \mathcal{S}$ as a set of conditions imposed on the pairs in $\left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2$, by which (x_A, x_B) meets the conditions if and only if (x_A, x_B) is in the left kernel of S, the partial order $\preceq$ can be interpreted as "being more restrictive than."

Proposition 3.2.7. *For all* $(x_A, x_B) \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2$*, there exists a unique minimal element* $S \in \mathcal{S}$ *having* (x_A, x_B) *in its left kernel. In particular,* $t(x_A, x_B)$ *has the number of distinct eigenvalues given by Equation* (3.3)*.*

Remark 3.2.8. Since the definition of the equivalence relation $\equiv_S$ on H does not rely on whether S meets the requirement of Proposition 3.2.4 as such, it makes sense to extend it to all $S \in \mathcal{S}$. Then, Equation (3.3) defines a function $s : \mathcal{S} \to \mathbb{N}$ given by

$$s(S) = \begin{cases} 2\left|H/\equiv_S\right| - 3 & \text{if } \left|\left[\frac{1}{2}\right]\right| = 1 \\ 2\left|H/\equiv_S\right| - 2 & \text{else} \end{cases}.$$

Thus, Proposition 3.2.7 states the following: given $(x_A, x_B) \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2$, let $S \in \mathcal{S}$ be the minimal element having (x_A, x_B) in its left kernel; then, $t(x_A, x_B)$ has $s(S)$ distinct eigenvalues.

Example 3.2.9. Consider the matrix Q in Example 3.2.3. The equivalence classes in H under the equivalence relation $\equiv_Q$ are

$$\{0\}, \left\{\frac{1}{2}\right\}, \{1, 2, 3\},$$

so $s(Q) = 2 \cdot 3 - 3 = 3$. On the other hand, as we have seen in Example 3.2.3, when $m = 3$, $\left(\frac{1}{3}, 0\right) \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2$ is in the left kernel of Q but not that of any other submatrix of P properly containing Q, and $t\left(\frac{1}{3}, 0\right)$ has three distinct eigenvalues.

3.2.2 Determining when $t(x_A, x_B)$ is fixed by some $w \in W$

In relation to Equation (3.1), we have obtained a characterization of when $(x_A, x_B) \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2$ is such that $t(x_A, x_B)$ has s distinct eigenvalues. To reiterate, this occurs if and only if the minimal element $S \in \mathcal{S}$ associated to (x_A, x_B) satisfies $s(S) = s$. It remains to determine the number of such pairs that give rise to fixed points under the action of an element of W.

Here, we establish a correspondence between W and a certain subset of $\mathcal{S}$, whereby $t(x_A, x_B)$ is fixed by some $w \in W$ if and only if (x_A, x_B) lies in the left kernel of the element in $\mathcal{S}$ corresponding to w. Recall that, ignoring the common factor of $2\pi i$, the exponents of the first three diagonal entries of $t(x_A, x_B)$ are

$$(P_1, P_2, P_3).$$

Let σ be the signed permutation on $\{1, 2, 3\}$ that is associated to w by Proposition 3.1.1. For $i = 1, 2, 3$, we define

$$\sigma(P_i) = \begin{cases} P_{\sigma(i)} & \sigma(i) > 0 \\ -P_{-\sigma(i)} & \text{else} \end{cases}.$$

Thus, the exponents of the first three diagonal entries of $w \cdot t(x_A, x_B)$ are, once again ignoring the common factor of $2\pi i$,

$$(\sigma(P_1), \sigma(P_2), \sigma(P_3)).$$

Now, we consider when $t(x_A, x_B)$ and $w \cdot t(x_A, x_B)$ agree in their ith diagonal entries, for each $i = 1, 2$:

1. if $\sigma(i) = i$, then this happens no matter what;
2. if $\sigma(i) = -i$, then this happens if and only if $2P_i = 0$, or (x_A, x_B) is in the left kernel of $2\mathbf{v}_i$;
3. if $\sigma(i) = \pm j$, where $j \in \{1, 2, 3\}$ and $j \neq i$, then this happens if and only if $P_i \mp P_j = 0$, or (x_A, x_B) is in the left kernel of $\mathbf{v}_i \mp \mathbf{v}_j$.

By stacking together the distinct vectors obtained for each $i = 1, 2$, we obtain a matrix S_w (possibly of size 2×0) in $\mathcal{S}$ with the following property.

Proposition 3.2.10. *Let $w \in W$ and $S_w \in \mathcal{S}$ be defined as above. Then, $t(x_A, x_B)$ is fixed by w if and only if (x_A, x_B) is in the left kernel of S_w.*

Remark 3.2.11. The matrices S_w, where $w \in W$ is a conjugacy class representative of our choice, have almost been computed in Table 3.2 and used in Example 3.1.2. The only modification we have made here is that we are excluding any column of zeros or repeated columns that may arise from comparing $t(x_A, x_B)$ and $w \cdot t(x_A, x_B)$ in their first two diagonal entries. See Table 3.3.

$\lvert c\rvert$	$w_c \in c$	S_{w_c}
1	$I \cdot T$	$\emptyset$
3	$n_A(1) \cdot T$	$\begin{bmatrix} 2 \\ -1 \end{bmatrix}$
3	$n_B(1) \cdot T$	$\begin{bmatrix} 3 \\ -2 \end{bmatrix}$
2	$n_B(1)n_A(1) \cdot T$	$\begin{bmatrix} 2 & 1 \\ -1 & 0 \end{bmatrix}$
2	$(n_B(1)n_A(1))^2 \cdot T$	$\begin{bmatrix} 3 & 0 \\ -1 & 1 \end{bmatrix}$
1	$(n_B(1)n_A(1))^3 \cdot T$	$\begin{bmatrix} 2 & 2 \\ 0 & -2 \end{bmatrix}$

Table 3.3: The matrices S_{w_c}, where $w_c \in c \subseteq W$ is a conjugacy class representative of our choice, which we will use to detect when $t(x_A, x_B)$ is fixed by each w_c.

3.2.3 Counting elements in $E(T, m, s)$ fixed by some $w \in W$

Equipped with the means to detect when $t(x_A, x_B)$ has s distinct eigenvalues or is fixed by some $w \in W$, we are ready to study the number of fixed points in $E(T, m, s)$ under the action of each $w \in W$. The approach that we take involves repeated exploitations of the partial order $\preceq$ on $\mathcal{S}$.

For each $S \in \mathcal{S}$, let

$$G_m(S) = \left\{(x_A, x_B) \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2 : \ (x_A, x_B)S = \mathbf{0}^t\right\}, \text{ and}$$

$$F_m(S) = \left\{(x_A, x_B) \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^2 : \ (x_A, x_B)S = \mathbf{0}^t, \ (x_A, x_B)T \neq \mathbf{0}^t \text{ for all } T \prec S\right\},$$

and define, accordingly,

$$g_m(S) = |G_m(S)|, \text{ and } f_m(S) = |F_m(S)|.$$

It follows from Proposition 3.2.7 that we have the partition

$$E(T, m, s) = \bigcup_{s(S)=s} \{t(x_A, x_B) : \ (x_A, x_B) \in F_m(S)\}. \tag{3.4}$$

As we shall see, if $F_m(S) \neq \emptyset$, then each $w \in W$ fixes either all or none of the $t(x_A, x_B)$ for $(x_A, x_B) \in F_m(S)$. Hence, to determine the number of fixed points in $E(T, m, s)$ under the action of each $w \in W$, it makes practical sense for us to compute $f_m(S)$ in the event that $F_m(S) \neq \emptyset$. We introduce a necessary condition for $S \in \mathcal{S}$ to satisfy $F_m(S) \neq \emptyset$.

Proposition 3.2.12. *Let $S \in \mathcal{S}$. If there exists a column vector $\mathbf{v}$ of P that is in $\Lambda(S)$ but is not a column vector of S, then $f_m(S) = 0$.*

Proof. Suppose that such a $\mathbf{v}$ exists. Let S' be the smallest submatrix of P containing both S and $\mathbf{v}$. Since $\mathbf{v} \in \Lambda(S)$, $\mathbf{v}$ is a $\mathbb{Z}$-linear combination of the column vectors of S, and so the left kernels of S and S' over $\frac{1}{m}\mathbb{Z}/\mathbb{Z}$ coincide. Moreover, $S' \prec S$. It follows from the definition that $F_m(S) = \emptyset$, and so $f_m(S) = 0$. □

The contrapositive of Proposition 3.2.12 states that if $F_m(S) \neq \emptyset$, then every column vector of P that is in $\Lambda(S)$ is a column vector of S. Let $\mathcal{M}$ denote the subset of $\mathcal{S}$ that consists of such matrices in $\mathcal{S}$ that satisfy the conclusion of the contrapositive statement. Note that $\mathcal{M}$ comes with a partially ordered structure that is inherited from $\mathcal{S}$.

Observe that, for all $S \in \mathcal{S}$,

$$g_m(S) = \sum_{T \preceq S} f_m(T) = \sum_{\substack{T \preceq S \\ T \in \mathcal{M}}} f_m(T).$$

By restricting our attention to matrices in $\in \mathcal{M}$, the above identity reduces to

$$g_m(S) = \sum_{T \preceq S} f_m(T), \text{ for } S, T \in \mathcal{M}.$$

Then, it follows from Theorem 2.2.8 that

$$\boxed{f_m(S) = \sum_{T \preceq S} \mu_{\mathcal{M}}(T, S) g_m(T).} \tag{3.5}$$

With respect to computing $f_m(S)$, it remains to determine $g_m(T)$ for each $T \preceq S$ with $T \in \mathcal{M}$. However, by definition, $g_m(T)$ is the size of the left kernel of T over $\frac{1}{m}\mathbb{Z}/\mathbb{Z}$, and so it can be directly computed using Corollary 2.3.9. Thus, in conjunction with Corollary 2.3.9, Equation (3.5) provides us with a practical method for computing $f_m(S)$.

Now, we address the following claim, to which we have briefly alluded earlier.

Proposition 3.2.13. *If $F_m(S) \neq \emptyset$, then each $w \in W$ fixes either all or none of the $t(x_A, x_B)$ for $(x_A, x_B) \in F_m(S)$. In particular, w fixes all such elements if and only if $S \preceq S_w$.*

Proof. Suppose that $F_m(S) \neq \emptyset$, and fix $w \in W$. Suppose that for some $(x_A, x_B) \in F_m(S)$, $t(x_A, x_B)$ is fixed by w. Then, by Proposition 3.2.10, (x_A, x_B) is in the left kernel of $S_w \in \mathcal{S}$. However, S is the unique minimal element of $\mathcal{S}$ having (x_A, x_B) in its left kernel, so $S \preceq S_w$. Hence, the left kernel of S is contained inside that of S_w. By Proposition 3.2.10, this implies that w fixes all of the $t(x_A, x_B)$, where $(x_A, x_B) \in F_m(S)$. □

In consideration of Equation (3.4), the number of fixed points of $E(T, m, s)$ under the action of some $w \in W$ is the sum of the number of elements in each subset on the right-hand side that are fixed by w. In view of Proposition 3.2.13, we have

$$\boxed{|\mathrm{Fix}(w)| = \sum_{\substack{S \in \mathcal{M} \\ s(S)=s \\ S \preceq S_w}} f_m(S).} \tag{3.6}$$

3.2.4 Computing $N(G_2, m, s)$

Combining Equations (3.1) and (3.6), we obtain an expression for $N(G_2, m, s)$:

$$\boxed{N(G_2, m, s) = \frac{1}{|W|} \sum_{c \in \mathrm{Cl}(W)} |c| \sum_{\substack{S \in \mathcal{M} \\ s(S)=s \\ S \preceq S_{w_c}}} f_m(S),} \tag{3.7}$$

where $f_m(S)$ can be computed using Equation (3.5).

We introduce a straightforward optimization to using Equation (3.7), which is based on the following fact (see Chapter 5 for more details).

Fact 3.2.14. For all $S \in \mathcal{S}$, there exists a $2 \times k$ submatrix S', where $0 \leq k \leq 2$, such that $\Lambda(S) = \Lambda(S')$.

Let $\mathrm{col}(P)$ denote the set of column vectors of P. Given any $S \in \mathcal{M}$, the set of column vectors of S coincides with $\Lambda(S) \cap \mathrm{col}(P)$ by definition. Conversely, given any $S' \in \mathcal{S}$, the submatrix S of P with the set of column vectors $\Lambda(S') \cap \mathrm{col}(P)$ is necessarily in $\mathcal{M}$. Hence, $\mathcal{M}$ consists of the matrices in $\mathcal{S}$ having the set of column vectors $\Lambda(S) \cap \mathrm{col}(P)$ for some $S \in \mathcal{S}$. With Fact 3.2.14, we can deduce the following.

Proposition 3.2.15. *$\mathcal{M}$ consists of the matrices in $\mathcal{S}$ having the set of column vectors $\Lambda(S) \cap \mathrm{col}(P)$, where $S \in \mathcal{S}$ has no more than two columns.*

It turns out that there are 19 matrices in $\mathcal{M}$, which we list in Table 3.4 along with their s- and g_m-values. With this information, we are completely prepared to compute $N(G_2, m, s)$. As an example, consider $N(G_2, m, 2)$.

Example 3.2.16 (computing $N(G_2, m, 2)$)**.** For convenience, let

$$S_1 = \begin{bmatrix} 2 & 2 & 4 \\ 0 & -2 & -2 \end{bmatrix},$$

$S \in \mathcal{M}$	$s(S)$	$g_m(S)$
$\emptyset$	7	m^2
$\begin{bmatrix}2\\0\end{bmatrix}, \begin{bmatrix}2\\-2\end{bmatrix}, \begin{bmatrix}4\\-2\end{bmatrix}$	6	$\begin{cases}2m & m \equiv 0 \bmod 2\\ m & \text{else}\end{cases}$
$\begin{bmatrix}3\\-2\end{bmatrix}, \begin{bmatrix}3\\-1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}$	5	m
$\begin{bmatrix}0&4\\1&-2\end{bmatrix}, \begin{bmatrix}3&2\\-2&0\end{bmatrix}, \begin{bmatrix}3&2\\-1&-2\end{bmatrix}$	4	$\begin{cases}4 & m \equiv 0 \bmod 4\\ 1 & m \equiv 1,3 \bmod 4\\ 2 & \text{else}\end{cases}$
$\begin{bmatrix}1&2\\0&0\end{bmatrix}, \begin{bmatrix}1&2\\-1&-2\end{bmatrix}, \begin{bmatrix}2&4\\-1&-2\end{bmatrix}$	3	m
$\begin{bmatrix}3&3&0\\-2&-1&1\end{bmatrix}$	3	$\begin{cases}3 & m \equiv 0 \bmod 3\\ 1 & \text{else}\end{cases}$
$\begin{bmatrix}2&2&4\\0&-2&-2\end{bmatrix}$	2	$\begin{cases}4 & m \equiv 0 \bmod 2\\ 1 & \text{else}\end{cases}$
$\begin{bmatrix}2&0&2&2&4\\-1&1&0&-2&-2\end{bmatrix}, \begin{bmatrix}1&3&2&2&4\\0&-2&0&-2&-2\end{bmatrix}, \begin{bmatrix}1&3&2&2&4\\-1&-1&0&-2&-2\end{bmatrix}$	2	$\begin{cases}2 & m \equiv 0 \bmod 2\\ 1 & \text{else}\end{cases}$
$\begin{bmatrix}1&1&2&2&0&3&2&3&4\\0&-1&-1&0&1&-1&-2&-2&-2\end{bmatrix}$	1	1

Table 3.4: The 19 matrices in $\mathcal{M}$.

$$S_2 = \begin{bmatrix}2&0&2&2&4\\-1&1&0&-2&-2\end{bmatrix},$$
$$S_3 = \begin{bmatrix}1&3&2&2&4\\0&-2&0&-2&-2\end{bmatrix},$$
$$S_4 = \begin{bmatrix}1&3&2&2&4\\-1&-1&0&-2&-2\end{bmatrix}, \text{ and}$$
$$S_5 = \begin{bmatrix}1&1&2&2&0&3&2&3&4\\0&-1&-1&0&1&-1&-2&-2&-2\end{bmatrix} = P.$$

We begin by computing the Möbius coefficients using Equation (2.1):

$$\mu(S_5, S_4) = \mu(S_5, S_3) = \mu(S_5, S_2) = \mu(S_4, S_1) = \mu(S_3, S_1) = \mu(S_2, S_1) = -1, \text{ and}$$
$$\mu(S_5, S_1) = -\left[\mu(S_5, S_5) + \mu(S_5, S_4) + \mu(S_5, S_3) + \mu(S_5, S_2)\right] = 2.$$

Then, we can compute $f_m(S_i)$ for $i = 1, \ldots, 4$ using Equation (3.5):

$$f_m(S_1) = \sum_{i=1}^{5} \mu(S_i, S_1) g_m(S_i) = \begin{cases} 4 & m \equiv 0 \bmod 2 \\ 1 & \text{else} \end{cases} - 3 \cdot \begin{cases} 2 & m \equiv 0 \bmod 2 \\ 1 & \text{else} \end{cases} + 2 = 0, \text{ and}$$

$$f_m(S_2) = \mu(S_5, S_2) g_m(S_5) + \mu(S_2, S_2) g_m(S_2) = \begin{cases} 1 & m \equiv 0 \bmod 2 \\ 0 & \text{else} \end{cases} = f_m(S_3) = f_m(S_4).$$

Since $f_m(S_1) = 0$, it can be neglected for the purpose of applying Equation (3.7). To check for fixed points, we observe that, by referring to Table 3.3,

1. $S_2, S_3, S_4 \preceq S_{I \cdot T}, S_{(n_B(1) n_A(1))^3 \cdot T}$,
2. $S_2 \preceq S_{n_A(1) \cdot T}$, and
3. $S_3 \preceq S_{n_B(1) \cdot T}$.

Hence,

$$\begin{aligned} N(G_2, m, 2) &= \frac{1}{12} \left[(1+1) \cdot (f_m(S_2) + f_m(S_3) + f_m(S_4)) + 3 \cdot f_m(S_2) + 3 \cdot f_m(S_3) \right] \\ &= \begin{cases} 1 & m \equiv 0 \bmod 2 \\ 0 & \text{else} \end{cases}. \end{aligned}$$

Repeating this for the other possible values of s, we obtain the following theorem.

Theorem 3.2.17. *For any positive integer m and $1 \leq s \leq 7$,*

$N(G_2, m, s)$	s						
$m \bmod 12$	1	2	3	4	5	6	7
0	1	1	$m/2$	1	$(m-6)/2$	$(m-4)/4$	$m(m-9)/12 + 2$
$1, 5, 7, 11$	1	0	$(m-1)/2$	0	$(m-1)/2$	0	$(m-1)(m-5)/12$
$2, 10$	1	1	$(m-2)/2$	0	$(m-2)/2$	$(m-2)/4$	$(m-2)(m-7)/12$
$3, 9$	1	0	$(m+1)/2$	0	$(m-3)/2$	0	$(m-3)^2/12$
$4, 8$	1	1	$(m-2)/2$	1	$(m-4)/2$	$(m-4)/4$	$(m-4)(m-5)/12$
6	1	1	$m/2$	0	$(m-4)/2$	$(m-2)/4$	$(m-3)(m-6)/12$

3.3 A Brief Comparison with $\mathrm{Sp}(n)$

Before delving into the description of the general algorithm, we take a brief moment to discuss the exact reason why the simple combinatorial approach taken by [5, 6] to study the

quantities $N(G,m)$ and $N(G,m,s)$ for the compact classical groups is not directly applicable to the compact exceptional groups.

We use $\mathrm{Sp}(n) = \mathrm{Sp}(n;\mathbb{C}) \cap \mathrm{U}(2n)$ as an instructive example. Since $\mathrm{Sp}(n)$ is compact and connected, the discussion in Section 2.1 transfers without difficult. It is known that, under a suitable basis, $\mathrm{Sp}(n)$ has a maximal torus T consisting of elements of the form

$$t(x_1,\ldots,x_n) = \begin{bmatrix} e^{2\pi i x_1} & & & & & & \\ & \ddots & & & & & \\ & & e^{2\pi i x_n} & & & & \\ & & & e^{-2\pi i x_n} & & & \\ & & & & \ddots & & \\ & & & & & e^{-2\pi i x_1} \end{bmatrix},$$

where $x_1,\ldots,x_n \in \mathbb{R}$. Further, the Weyl group of $\mathrm{Sp}(n)$ is isomorphic to the signed permutation group on $\{1,\ldots,n\}$. With respect to T, the Weyl group induces all possible signed permutations of the exponents of the first n diagonal entries of $t(x_1,\ldots,x_n)$, whereby its last n diagonal entries are permuted accordingly. More concretely, if we write

$$t(\mathbf{x}) = t(x_1,\ldots,x_n),$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix},$$

then for any $n \times n$ signed permutation matrix A, $t(A\mathbf{x})$ is conjugate to $t(\mathbf{x})$ in $\mathrm{Sp}(n)$. Thus, we conclude that every conjugacy class in $E(\mathrm{Sp}(n),m)$ has a unique representative of the form

$$\begin{bmatrix} e^{2\pi i x_1} & & & & & & \\ & \ddots & & & & & \\ & & e^{2\pi i x_n} & & & & \\ & & & e^{-2\pi i x_n} & & & \\ & & & & \ddots & & \\ & & & & & e^{-2\pi i x_1} \end{bmatrix},$$

which satisfies that each $x_i \in J = \left\{0, \frac{1}{m}, \ldots, \frac{1}{m}\left\lfloor \frac{m}{2} \right\rfloor\right\}$, and $x_1 \leq \cdots \leq x_n$. Hence, the conjugacy classes in $E(\mathrm{Sp}(n),m)$ are in one-to-one correspondence with the non-decreasing sequences of length n

$$(x_1,\ldots,x_n)$$

having entries in J. The combinatorial properties of such sequences are quite well-understood, and there are a number of elementary methods to count these sequences. Thus, we have established that the problem of computing $N(\mathrm{Sp}(n),m)$ is equivalent to a well-studied enumeration problem in combinatorics.

In regard to $N(\mathrm{Sp}(n),m,s)$, [5, 6] have defined this quantity as the number of conjugacy classes in $E(\mathrm{Sp}(n),m)$ whose elements have s distinct *conjugate pairs* of eigenvalues.

In view of the discussion in the previous paragraph, it suffices to keep track of the exact number of elements of J that appear in a particular sequence $(x_1, \ldots, x_n)$. Considering that $t(x_1, \ldots, x_n)$ has the list of eigenvalues

$$\left\{e^{\pm 2\pi i x_1}, \ldots, e^{\pm 2\pi i x_n}\right\},$$

it has s distinct conjugate pairs of eigenvalues if and only if exactly s distinct elements of J appear in the sequence $(x_1, \ldots, x_n)$. The enumeration problem of these sequences is also relatively straightforward to handle.

With respect to the other three classes of compact classical groups, their Weyl groups are generally large enough to contain the symmetric group, which allows one to simplify the computation of $N(G, m)$ for these groups considerably. Oftentimes, one can reduce it to the enumeration of certain non-decreasing sequences with entries in a similar set to J, which in turn facilitates the computation of $N(G, m, s)$. Even when this reduction proves to be too naïve, one can use the Weyl group to introduce a convenient canonical form in T and work with representatives in this canonical form, which generally simplifies the problem to a great extent.

It is now quite clear as to why the simple combinatorial methods applicable to the compact classical groups cannot be directly extended to the compact exceptional groups, which include G_2. For a compact exceptional group, its Weyl group is generally too small to contain the full (signed) permutation group that would make possible the introduction of a canonical form to its maximal torus. Furthermore, the parametrized entries in a typical element of its maximal torus are generally too coupled for us to keep track of the number of distinct eigenvalues that appear in a realistic manner via as simple a method as for $\mathrm{Sp}(n)$.

Chapter 4

The General Method

In this chapter, we summarize and generalize the method used in Chapter 3 to all compact exceptional groups. Since almost everything transfer with only slight modification to the general case, we will be somewhat cavalier about motivating or justifying any definition, proposition, or result that we present here.

Let G be a compact exceptional group with rank r. Using Table 2.1, we can obtain the matrix representations of a maximal torus T in G and the generators of its corresponding Weyl group W. Let

$$t(\mathbf{x}) = t(x_{\delta_1}, \ldots, x_{\delta_r})$$

denote the typical element of T, where $(\delta_1, \ldots, \delta_r)$ are the fundamental roots of $\mathfrak{g}_{\mathbb{C}}$ listed in Table 2.1. In a suitable basis, $t(\mathbf{x})$ takes the form

1. $\mathrm{diag}(\exp(2\pi i P_1), \ldots, \exp(2\pi i P_t), 1, 1, \exp(-2\pi i P_{\varphi(t)}), \ldots, \exp(-2\pi i P_{\varphi(1)}))$, where $\varphi \in S_t$, if $G = G_2, F_4$, for which $t = 3, 12$, respectively;

2. $\mathrm{diag}(\exp(2\pi i P_1), \ldots, \exp(2\pi i P_t))$, where $t = 27$, if $G = E_6$;

3. $\mathrm{diag}(\exp(2\pi i P_1), \ldots, \exp(2\pi i P_t), \exp(-2\pi i P_{\varphi(t)}), \ldots, \exp(-2\pi i P_{\varphi(1)}))$, where $t = 28$ and $\varphi \in S_t$, if $G = E_7$; or

4. $\mathrm{diag}(\exp(2\pi i P_1), \ldots, \exp(2\pi i P_t), \underbrace{1, \ldots, 1}_{8}, \exp(-2\pi i P_{\varphi(t)}), \ldots, \exp(-2\pi i P_{\varphi(1)}))$, where $t = 120$ and $\varphi \in S_t$, if $G = E_8$.

Here, for each G, $P_1, \ldots, P_t$ are distinct, nonzero, $\mathbb{Z}$-linear combinations of $x_{\delta_1}, \ldots, x_{\delta_r}$. A generalization of Remark 3.0.1 remains true.

Remark 4.0.1. For a compact exceptional group G with rank r, the element $t(\mathbf{x}) \in T$ has order dividing m if and only if $\mathbf{x} \in \left(\frac{1}{m}\mathbb{Z}\right)^r$. Moreover, $t(\mathbf{x}) = t(\mathbf{x}')$ if and only if $\mathbf{x} - \mathbf{x}' \in \mathbb{Z}^r$. Hence, we shall require that $\mathbf{x} \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^r$.

To each P_i, we associate a vector $\mathbf{v}_i \in \mathbb{Z}^r$ that is the coefficient vector of P_i with respect to the ordered basis $(x_{\delta_1}, \ldots, x_{\delta_r})$ (cf. the discussion following Proposition 3.2.1).

4.1 The Computation of $N(G, m)$

As before, $N(G, m)$ is the number of orbits under the action of W of the set

$$\left\{ t(\mathbf{x}) : \mathbf{x} \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z} \right)^r \right\}$$

by the discussion preceding Theorem 2.2.4. In view of Theorem 2.2.6, it suffices to determine $|\mathrm{Fix}(w_c)|$, the number of fixed points of the above set under the action of a representative $w_c \in c \subseteq W$, for each conjugacy class $c \in \mathrm{Cl}(W)$.

We use the results of [1], which can be accessed in GAP 3 (cf. Section 3.1), to compute a representative for each conjugacy class in W.

Remark 4.1.1. The matrix generators of W computed as per [8] rely on a different labeling of the Dynkin diagram of $\mathfrak{g}_{\mathbb{C}}$ than the conjugacy class representatives given by GAP 3. For instance, for G_2, [8] uses the labeling

for its Dynkin diagram, whereas the labeling used by GAP 3 is

,

which can be checked with the following command:

```
gap> PrintDiagram(ReflectionType(CartanMat("G",2)));
G2 1 >>> 2
```

Hence, the correspondence between the labelings of the fundamental roots of $\mathfrak{g}_2$ in [8] and GAP 3 is

$$A \leftrightarrow 2, \text{ and } B \leftrightarrow 1.$$

The upshot is that we must account for the different labelings of the fundamental roots used by [8] and GAP 3 to ensure that we obtain correct results.

It turns out that an analogue of Proposition 3.1.1 still applies in the general case, which provides a necessary and sufficient condition for $t(\mathbf{x})$ to be fixed by some $w \in W$.

Proposition 4.1.2. *Let $w \in W$. Acting on $t(\mathbf{x})$ by w results in a signed permutation of the exponents in its first t diagonal entries. Further, there exist r indices $\{i_1, \ldots, i_r\} \subseteq \{1, \ldots, t\}$ such that, for any $w \in W$, $t(\mathbf{x})$ is fixed by w if and only if $w \cdot t(\mathbf{x})$ and $t(\mathbf{x})$ agree in their i_jth diagonal entries for each $j = 1, \ldots, r$.*

Proof. Computationally verified. □

The procedure for computing each $|\mathrm{Fix}(w_c)|$ is then completely analogous to what we have done in Example 3.1.2; see Algorithm 1.

Algorithm 1 An algorithm for computing the number of fixed points in $E(T, m)$ under the action of a conjugacy class representative w_c.

Require: a conjugacy class representative w_c

1: compute its matrix representation
2: compute $w_c \cdot t(\mathbf{x}) = w_c t(\mathbf{x}) w_c^{-1}$
3: by equating the i_jth entries of $w_c \cdot t(\mathbf{x})$ and $t(\mathbf{x})$ for each $j = 1, \ldots, r$, write down a system of linear equations in r variables: $x_1, \ldots, x_r$
4: express the system in terms of the left kernel of an $r \times r$ coefficient matrix
5: compute the size of the left kernel using Corollary 2.3.9, which will be $|\mathrm{Fix}(w_c)|$
6: **return** $|\mathrm{Fix}(w_c)|$

With the size of each conjugacy class in W that is displayed under `classes` in GAP 3, we apply Theorem 2.2.6 to compute

$$N(G, m) = \frac{1}{|W|} \sum_{c \in \mathrm{Cl}(W)} |c| \, |\mathrm{Fix}(w_c)| \, .$$

Thus, we obtain the following theorems.

Theorem 4.1.3. *For any positive integer m,*

m mod 12	$N(F_4, m)$
0	$(m^4 + 24m^3 + 208m^2 + 768m + 1152)/1152$
$\pm 1, \pm 5$	$(m^4 + 24m^3 + 190m^2 + 552m + 385)/1152$
± 2	$(m^4 + 24m^3 + 208m^2 + 768m + 880)/1152$
± 3	$(m^4 + 24m^3 + 190m^2 + 552m + 513)/1152$
± 4	$(m^4 + 24m^3 + 208m^2 + 768m + 1024)/1152$
6	$(m^4 + 24m^3 + 208m^2 + 768m + 1008)/1152$

Theorem 4.1.4. *For any positive integer m,*

m mod 6	$N(E_6, m)$
0	$(m^6 + 36m^5 + 510m^4 + 3600m^3 + 14184m^2 + 35424m + 51840)/51840$
± 1	$(m^6 + 36m^5 + 510m^4 + 3600m^3 + 13089m^2 + 22284m + 12320)/51840$
± 2	$(m^6 + 36m^5 + 510m^4 + 3600m^3 + 13224m^2 + 23904m + 16640)/51840$
3	$(m^6 + 36m^5 + 510m^4 + 3600m^3 + 14049m^2 + 33804m + 38880)/51840$

Theorem 4.1.5. *For any positive integer m,*

m mod 12	$N(E_7, m)$
0	$(m^7 + 63m^6 + 1617m^5 + 22050m^4 + 175224m^3 + 830592m^2 + 2239488m + 2903040)/2903040$
$\pm 1, \pm 5$	$(m^7 + 63m^6 + 1617m^5 + 21735m^4 + 162939m^3 + 663957m^2 + 1286963m + 765765)/2903040$
± 2	$(m^7 + 63m^6 + 1617m^5 + 22050m^4 + 175224m^3 + 830592m^2 + 2176208m + 2126880)/2903040$
± 3	$(m^7 + 63m^6 + 1617m^5 + 21735m^4 + 162939m^3 + 663957m^2 + 1304883m + 927045)/2903040$
± 4	$(m^7 + 63m^6 + 1617m^5 + 22050m^4 + 175224m^3 + 830592m^2 + 2221568m + 2580480)/2903040$
6	$(m^7 + 63m^6 + 1617m^5 + 22050m^4 + 175224m^3 + 830592m^2 + 2194128m + 2449440)/2903040$

Theorem 4.1.6. *For any positive integer m,*

m mod 60	$N(E_8, m)$
0	$(m^8+120m^7+6020m^6+163800m^5+2626008m^4+25260480m^3+142577280m^2+445824000m+696729600)/696729600$
$\pm1, \pm7, \pm11, \pm13, \pm17, \pm19, \pm23, \pm29$	$(m^8+120m^7+6020m^6+163800m^5+2616558m^4+24693480m^3+130085780m^2+323507400m+215656441)/696729600$
$\pm2, \pm14, \pm22, \pm26$	$(m^8+120m^7+6020m^6+163800m^5+2626008m^4+25260480m^3+141860480m^2+418876800m+435250816)/696729600$
$\pm3, \pm9, \pm21, \pm27$	$(m^8+120m^7+6020m^6+163800m^5+2616558m^4+24693480m^3+130802580m^2+345011400m+348264441)/696729600$
$\pm4, \pm8, \pm16, \pm28$	$(m^8+120m^7+6020m^6+163800m^5+2626008m^4+25260480m^3+141860480m^2+424320000m+516898816)/696729600$
$\pm5, \pm25$	$(m^8+120m^7+6020m^6+163800m^5+2616558m^4+24693480m^3+130085780m^2+323507400m+243525625)/696729600$
$\pm6, \pm18$	$(m^8+120m^7+6020m^6+163800m^5+2626008m^4+25260480m^3+142577280m^2+440380800m+587212416)/696729600$
±10	$(m^8+120m^7+6020m^6+163800m^5+2626008m^4+25260480m^3+141860480m^2+418876800m+463120000)/696729600$
$\pm12, \pm24$	$(m^8+120m^7+6020m^6+163800m^5+2626008m^4+25260480m^3+142577280m^2+445824000m+668860416)/696729600$
±15	$(m^8+120m^7+6020m^6+163800m^5+2616558m^4+24693480m^3+130802580m^2+345011400m+376133625)/696729600$
±20	$(m^8+120m^7+6020m^6+163800m^5+2626008m^4+25260480m^3+141860480m^2+424320000m+544768000)/696729600$
30	$(m^8+120m^7+6020m^6+163800m^5+2626008m^4+25260480m^3+142577280m^2+440380800m+615081600)/696729600$

4.2 The Computation of $N(G, m, s)$

We follow essentially the same four-step approach as in Section 3.2 to tackle this refined enumeration problem. The main difference here is that, due to the variation in the form of $t(\mathbf{x})$ among the compact exceptional groups, such objects as the equivalence relation $\equiv_S$ and the induced s-function may assume a slightly different form than before.

In view of Theorem 2.2.6 and the discussion preceding Theorem 2.2.4, we need to determine the number of fixed points of the set

$$\left\{t(\mathbf{x}) : \mathbf{x} \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^r,\ t(\mathbf{x}) \text{ has } s \text{ distinct eigenvalues}\right\}$$

under the action of each conjugacy class representative $w_c \in c \in \mathrm{Cl}(W)$. Before we get started, we note that the (possibly redundant) set of eigenvalues of $t(\mathbf{x})$ is

$$\begin{cases} \left\{1, e^{\pm 2\pi i P_1}, \ldots, e^{\pm 2\pi i P_t}\right\}, \text{ if } G = G_2, F_4, E_8; \\ \left\{e^{2\pi i P_1}, \ldots, e^{2\pi i P_t}\right\}, \text{ if } G = E_6; \text{ or} \\ \left\{e^{\pm 2\pi i P_1}, \ldots, e^{\pm 2\pi i P_t}\right\}, \text{ if } G = E_7. \end{cases} \tag{4.1}$$

4.2.1 Determining the number of distinct eigenvalues of $t(\mathbf{x})$

First, we develop a systematic method for determining the number of distinct eigenvalues of $t(\mathbf{x})$ for all $\mathbf{x} \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^r$. We begin by making a similar observation as in Proposition 3.2.1 regarding when repeats can occur in (4.1).

Proposition 4.2.1. *An r-tuple $\mathbf{x} \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^r$ leads to one or more repeats in (4.1) if and only if*

1. *$P_i = \pm P_j$ for some $i \neq j$,*
2. *$P_i = 0$ for some i, or*
3. *$2P_i = 0$ but $P_i \neq 0$ for some i,*

for $G = G_2, F_4, E_7, E_8$; or

1. *$P_i = P_j$ for some $i \neq j$,*

for $G = E_6$.

To allow this to be translated into the language of matrices and kernels, we form a matrix P by stacking together the distinct vectors (up to a sign) in the list

$$\{\mathbf{v}_1 \pm \mathbf{v}_2, \mathbf{v}_1 \pm \mathbf{v}_3, \ldots, \mathbf{v}_{t-1} \pm \mathbf{v}_t, \mathbf{v}_1, \ldots, \mathbf{v}_t, 2\mathbf{v}_1, \ldots, 2\mathbf{v}_t\}.$$

The conditions in Proposition 4.2.1 can then be rephrased concisely as follows.[1]

Proposition 4.2.2. *An r-tuple $\mathbf{x} \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^r$ leads to one or more repeats in (4.1) if and only if, for $G = G_2, F_4, E_7, E_8$, $\mathbf{x}$ is in the left kernel of some $r \times k$ submatrix of P, where $k \geq 1$, or, for $G = E_6$, $\mathbf{x}$ is in the left kernel of some $r \times k$ submatrix of P, where $k \geq 1$, that contains a column vector of the form $\mathbf{v}_i - \mathbf{v}_j$, for some $i \neq j$.*

We adopt a completely analogous strategy as before to determine the number of distinct eigenvalues of $t(\mathbf{x})$ using P. Once again, for convenience, we treat $\left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^r$ as the left kernel of the unique $r \times 0$ submatrix of P.

[1] While the repeats of eigenvalues for $t(\mathbf{x}) \in E_6$ can be accounted for with only the vectors in the set $\{\mathbf{v}_1 - \mathbf{v}_2, \mathbf{v}_1 - \mathbf{v}_3, \ldots, \mathbf{v}_{t-1} - \mathbf{v}_t\}$, doing so will not afford sufficient information for us to determine whether $t(\mathbf{x})$ is fixed by some $w \in W$.

Definition 4.2.3. Let $\mathcal{S}$ be the set of all (possibly empty) submatrices of P with r rows. Define on $\mathcal{S}$ a partial order $\preceq$ by $S_1 \preceq S_2$ if and only if S_2 is a submatrix of S_1. Given $S \in \mathcal{S}$, define on

$$H = \left\{0, \frac{1}{2}, 1, \ldots, t\right\}$$

an equivalence relation $\equiv_S$ by the rules:

1. $i \equiv_S j$ if $\mathbf{v}_i \pm \mathbf{v}_j$ is (up to a sign) a column vector of S,
2. $i \equiv_S 0$ if $\mathbf{v}_i$ is (up to a sign) a column vector of S, and
3. $i \equiv_S \frac{1}{2}$ if $2\mathbf{v}_i$ is (up to a sign) a column vector of S but $\mathbf{v}_i$ is not,

for $G = G_2, F_4, E_7, E_8$; and define on

$$H = \{1, \ldots, t\}$$

an equivalence relation $\equiv_S$ by the rule:

1. $i \equiv_S j$ if $\mathbf{v}_i - \mathbf{v}_j$ is (up to a sign) a column vector of S,

for $G = E_6$. Finally, define a function $s : \mathcal{S} \to \mathbb{N}$ by

$$s(S) = \begin{cases} 2\left|H/\equiv_S\right| - 3 & \text{if } \left|\left[\frac{1}{2}\right]\right| = 1 \\ 2\left|H/\equiv_S\right| - 2 & \text{else} \end{cases},$$

for $G = G_2, F_4, E_8$;

$$s(S) = \left|H/\equiv_S\right|,$$

for $G = E_6$; and

$$s(S) = \begin{cases} 2\left|H/\equiv_S\right| - 4 & \text{if } |[0]| = 1 \text{ and } \left|\left[\frac{1}{2}\right]\right| = 1 \\ 2\left|H/\equiv_S\right| - 2 & \text{if } |[0]| > 1 \text{ and } \left|\left[\frac{1}{2}\right]\right| > 1 \\ 2\left|H/\equiv_S\right| - 3 & \text{else} \end{cases},$$

for $G = E_7$.

With the above definitions, we can characterize when $\mathbf{x} \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^r$ is such that $t(\mathbf{x})$ has s distinct eigenvalues in the same manner as in Propositions 3.2.4 and 3.2.7.

Proposition 4.2.4. *Let $S \in \mathcal{S}$. If $\mathbf{x} \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^r$ is in the left kernel of S but not that of any $T \in \mathcal{S}$ with $T \prec S$, then $t(\mathbf{x})$ has $s(S)$ distinct eigenvalues.*

Proposition 4.2.5. *For all $\mathbf{x} \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^r$, there exists a unique minimal element $S \in \mathcal{S}$ having $\mathbf{x}$ in its left kernel. In particular, $t(\mathbf{x})$ has $s(S)$ distinct eigenvalues.*

4.2.2 Determining when $t(\mathbf{x})$ is fixed by some $w \in W$

Now that we have a method of computing the number of distinct eigenvalues of any $t(\mathbf{x})$, we proceed to derive a necessary and sufficient condition for any $t(\mathbf{x})$ to be fixed by some $w \in W$, which will be completely analogous to Proposition 3.2.10.

Definition 4.2.6. Given $w \in W$, let σ be the signed permutation on $[t] = \{1, \ldots, t\}$ associated to w by Proposition 4.1.2. We associate to w the (possibly empty) matrix $S_w \in \mathcal{S}$, obtained by stacking together the distinct vectors (up to a sign) found via the following procedure, for $i = i_1, \ldots, i_r$:

1. if $\sigma(i) = i$, do nothing;
2. if $\sigma(i) = -i$, stack $2\mathbf{v}_i$;
3. if $\sigma(i) = \pm j$, where $j \in [t]$ and $j \neq i$, stack $\mathbf{v}_i \mp \mathbf{v}_j$.

Through Proposition 4.1.2, we obtain the desired necessary and sufficient condition.

Proposition 4.2.7. *Let $w \in W$. Then, $t(\mathbf{x})$ is fixed by w if and only if $\mathbf{x}$ is in the left kernel of S_w.*

4.2.3 Counting elements in $E(T, m, s)$ fixed by some $w \in W$

The penultimate step is to compute the number of fixed points in $E(T, m, s)$ under the action of each $w \in W$.

Similar to before, for each $S \in \mathcal{S}$, let

$$G_m(S) = \left\{ \mathbf{x} \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^r : \ \mathbf{x}S = \mathbf{0}^t \right\}, \text{ and}$$
$$F_m(S) = \left\{ \mathbf{x} \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z}\right)^r : \ \mathbf{x}S = \mathbf{0}^t, \ \mathbf{x}T \neq \mathbf{0}^t \text{ for all } T \prec S \right\},$$

and define, accordingly,

$$g_m(S) = |G_m(S)|, \text{ and } f_m(S) = |F_m(S)|.$$

It follows from Proposition 4.2.5 that we have the partition

$$E(T, m, s) = \bigcup_{s(S)=s} \{t(\mathbf{x}) : \ \mathbf{x} \in F_m(S)\}. \tag{4.2}$$

We are particularly interested in computing the quantity $f_m(S)$ for the same reason as before. The following analogue of Proposition 3.2.12 tells us that we only need to do this for each S in a particular subset of $\mathcal{S}$.

Proposition 4.2.8. *Let $S \in \mathcal{S}$. If there exists a column vector $\mathbf{v}$ of P that is in $\Lambda(S)$ but is not a column vector of S, then $f_m(S) = 0$.*

We are now ready to define this subset of $\mathcal{S}$.

Definition 4.2.9. Let $\mathcal{M} \subseteq \mathcal{S}$ consist of the matrices S such that every column vector of P that is in $\Lambda(S)$ is a column vector of S. We impose on $\mathcal{M}$ the partially ordered structure that it inherits from $\mathcal{S}$.

To compute $f_m(S)$ for some $S \in \mathcal{M}$, observe that, by restricting to matrices in $\mathcal{M}$, we have the identity

$$g_m(S) = \sum_{T \preceq S} f_m(T).$$

By Theorem 2.2.8, we get

$$\boxed{f_m(S) = \sum_{T \preceq S} \mu_{\mathcal{M}}(T, S) g_m(T),} \tag{4.3}$$

where, once again, each $g_m(T)$ can be computed straightforwardly using Corollary 2.3.9. To see how the determination of $f_m(S)$ for each $S \in \mathcal{M}$ serves our objective, we note that the following analogue of Proposition 3.2.13 still applies in the general setting.

Proposition 4.2.10. *If $F_m(S) \neq \emptyset$, then each $w \in W$ fixes either all or none of the $t(\mathbf{x})$ for $\mathbf{x} \in F_m(S)$. In particular, w fixes all such elements if and only if $S \preceq S_w$.*

The following formula is a consequence of Equation (4.2) and Proposition 4.2.10:

$$\boxed{|\mathrm{Fix}(w)| = \sum_{\substack{S \in \mathcal{M} \\ s(S) = s \\ S \preceq S_w}} f_m(S),} \tag{4.4}$$

where the set of fixed points is with respect to

$$\left\{ t(\mathbf{x}) : \mathbf{x} \in \left(\frac{1}{m}\mathbb{Z}/\mathbb{Z} \right)^r,\ t(\mathbf{x}) \text{ has } s \text{ distinct eigenvalues} \right\}.$$

4.2.4 Computing $N(G, m, s)$

Combining Theorem 2.2.6 and Equation (4.4), we obtain our final expression for $N(G, m, s)$:

$$\boxed{N(G, m, s) = \frac{1}{|W|} \sum_{c \in \mathrm{Cl}(W)} |c| \sum_{\substack{S \in \mathcal{M} \\ s(S) = s \\ S \preceq S_{w_c}}} f_m(S).} \tag{4.5}$$

The main hurdle with applying Equation (4.5) originates from the fact that it is not entirely clear whether or not there exists an efficient method to generate the elements of $\mathcal{M}$. Nonetheless, by the same reasoning that leads to Proposition 3.2.15, we have the following characterization of the elements of $\mathcal{M}$.

Proposition 4.2.11. *$\mathcal{M}$ consists of the matrices in $\mathcal{S}$ having the set of column vectors $\Lambda(S) \cap \mathrm{col}(P)$, where $S \in \mathcal{S}$.*

An analogue of Fact 3.2.14 applies in the case that $G = F_4$, which can also be computationally verified; see Chapter 5 for more details.

Fact 4.2.12. If $G = F_4$, then for all $S \in \mathcal{S}$, there exists a $4 \times k$ submatrix S', where $0 \leq k \leq 4$, such that $\Lambda(S) = \Lambda(S')$.

Hence, we may use Proposition 4.2.11 to generate the matrices in $\mathcal{M}$ for F_4 by restricting to such matrices $S \in \mathcal{S}$ that have no more than four columns. Thus, we find a total of 22075 matrices in $\mathcal{M}$. The full table for $N(F_4, m, s)$ can then be determined using Equation (4.5) with the assistance of a computer, which turns out to have dimension 12252240×25. We record its first 24 rows and 8 columns in Table 4.1; a copy of the Sage program that we have used to generate these formulas can be found in the appendix.

$1152 \cdot N(F_4, m, s)$	s							
m mod 12252240	1	2	3	4	5	6	7	8
0	1152	2304	$576m + 1152$	4608	$1152m - 3456$	$864m + 2304$	$96m^2 + 288m - 6912$	$864m + 1152$
1	1152	0	$576m - 576$	0	$1152m - 1152$	0	$96m^2 + 576m - 672$	0
2	1152	2304	$576m - 1152$	0	$1152m - 2304$	$864m - 1728$	$96m^2 + 288m - 960$	$864m - 1728$
3	1152	0	$576m + 1728$	0	$1152m - 3456$	0	$96m^2 + 576m - 2592$	0
4	1152	2304	$576m - 1152$	4608	$1152m - 4608$	$864m - 3456$	$96m^2 + 288m - 2688$	$864m - 3456$
5	1152	0	$576m - 576$	0	$1152m + 2304$	0	$96m^2 + 576m - 5280$	0
6	1152	2304	$576m + 1152$	0	$1152m - 4608$	$864m + 4032$	$96m^2 + 288m - 5184$	$864m - 5184$
7	1152	0	$576m - 576$	0	$1152m - 1152$	0	$96m^2 + 576m + 3936$	0
8	1152	2304	$576m - 1152$	4608	$1152m - 4608$	$864m - 3456$	$96m^2 + 288m - 2688$	$864m + 4608$
9	1152	0	$576m + 1728$	0	$1152m - 3456$	0	$96m^2 + 576m - 2592$	0
10	1152	2304	$576m - 1152$	0	$1152m + 1152$	$864m - 1728$	$96m^2 + 288m - 5568$	$864m - 1728$
11	1152	0	$576m - 576$	0	$1152m - 1152$	0	$96m^2 + 576m - 672$	0
12	1152	2304	$576m + 1152$	4608	$1152m - 6912$	$864m + 2304$	$96m^2 + 288m - 6912$	$864m - 6912$
13	1152	0	$576m - 576$	0	$1152m - 1152$	0	$96m^2 + 576m - 672$	0
14	1152	2304	$576m - 1152$	0	$1152m - 2304$	$864m - 1728$	$96m^2 + 288m + 3648$	$864m - 1728$
15	1152	0	$576m + 1728$	0	$1152m$	0	$96m^2 + 576m - 7200$	0
16	1152	2304	$576m - 1152$	4608	$1152m - 4608$	$864m - 3456$	$96m^2 + 288m - 2688$	$864m + 4608$
17	1152	0	$576m - 576$	0	$1152m - 1152$	0	$96m^2 + 576m - 672$	0
18	1152	2304	$576m + 1152$	0	$1152m - 4608$	$864m + 4032$	$96m^2 + 288m - 5184$	$864m - 5184$
19	1152	0	$576m - 576$	0	$1152m - 1152$	0	$96m^2 + 576m - 672$	0
20	1152	2304	$576m - 1152$	4608	$1152m - 1152$	$864m - 3456$	$96m^2 + 288m - 7296$	$864m - 3456$
21	1152	0	$576m + 1728$	0	$1152m - 3456$	0	$96m^2 + 576m + 2016$	0
22	1152	2304	$576m - 1152$	0	$1152m - 2304$	$864m - 1728$	$96m^2 + 288m - 960$	$864m - 1728$
23	1152	0	$576m - 576$	0	$1152m - 1152$	0	$96m^2 + 576m - 672$	0

Table 4.1: The first 24 rows and 8 columns of the full table of $N(F_4, m, s)$.

Chapter 5

Future Work

We have successfully computed $N(G, m)$ for all compact exceptional groups via a unified, combinatorial approach. Moreover, we have designed a general algorithm for computing $N(G, m, s)$, which we have applied to G_2 and F_4. As noted at the end of Subsection 4.2.4, the main difficulty vis-à-vis the application of the algorithm is due to the lack of a unified, efficient method to determine $\mathcal{M}$. In this chapter, we discuss two potential approaches to dealing with or circumventing this problem.

5.1 A Brute-Force Approach via HNFs

For our purposes, it might not be necessary to develop a clever method for generating the matrices in $\mathcal{M}$. Instead, inspired by the simplifications in Facts 3.2.14 and 4.2.12, we might seek for the smallest positive integer p such that every lattice $\Lambda(S)$, where $S \in \mathcal{S}$, can be generated by some matrix in $\mathcal{S}$ with no more than p columns. The method that we suggest relies on the following easy proposition.

Proposition 5.1.1. *Let $P \in \mathrm{M}_{r\times l}(\mathbb{Z})$. If every lattice generated by a submatrix of P with r rows and no more than $p+1$ columns can be generated by a submatrix with r rows and no more than p columns, then every lattice generated by a submatrix of P with r rows can be generated by a submatrix with r rows and no more than p columns.*

Recall from Proposition 2.3.5 that the Hermite normal form allows us to check whether two matrices generate the same integer lattice. Hence, we can use the brute-force method described in Algorithm 2 to find the minimum p to be used for generating all the matrices in $\mathcal{M}$.

Algorithm 2 A brute-force algorithm for determining the minimum integer p such that every lattice generated by a submatrix of P with r rows can be generated by a submatrix of P with r rows and no more than p columns.

Require: the matrix P

for $p \geq 1$ **do**
  compute the set H_p of distinct HNFs of the submatrices of P with r rows and no more than p columns
  compute the set H_{p+1} of distinct HNFs of the submatrices of P with r rows and no more than $p+1$ columns
  if $H_p = H_{p+1}$ up to adding or removing columns of zeros **then**
    return p
  else
    $p = p + 1$
  end if
end for

This brute-force approach has an obvious drawback. Namely, with larger compact exceptional groups, the size of the corresponding matrix P grows rapidly, which renders any endeavor involving the inspection of all $r \times p$ submatrices of P computationally costly, if not downright unrealistic.

5.2 A Subtle Approach via Root Vectors

Since the matrices in $\mathcal{M}$ are made up of the column vectors of P, which are generated by the vectors $\mathbf{v}_i$, it makes sense for us to investigate the origin of these vectors.

Through a careful study of how [8] constructs the matrix representations of the compact exceptional groups used herein, we obtain after some algebra the concrete description of the vectors $\mathbf{v}_i$ shown in Table 5.1. There, for each compact exceptional group in Column 1, the

E_8	$\mathfrak{e}_8$	$A\,[\alpha]_\Delta$
E_7	$\mathfrak{e}_8$	$\begin{bmatrix} I_7 & \mathbf{0} \end{bmatrix} A\,[\alpha]_\Delta$
E_6	$\mathfrak{e}_7$	$\begin{bmatrix} I_6 & \mathbf{0} \end{bmatrix} A\,[\alpha]_\Delta$
F_4	$\mathfrak{e}_7$	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} A\,[\alpha]_\Delta$
G_2	$\mathfrak{d}_5$	$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} A\,[\alpha]_\Delta$

Table 5.1: A concrete description of the vectors $\mathbf{v}_i$.

vectors $\mathbf{v}_i$ may be found using the formula in Column 3, where A is the Cartan matrix of the Lie algebra in Column 2, and the $\alpha \in X$ are given in [8] for $G \neq E_8$ and are precisely the root vectors of $\mathfrak{e}_8$ for $G = E_8$.

It is a curious fact that the set of vectors $\mathbf{v}_i$ for G_2 (resp. F_4), of which there are 12 (resp. 24), may be transformed onto the set of vectors in the A_2 (resp. D_4) root system. This trend breaks for E_6, for which there is no root system with the same number of vectors, and E_7, for which A_7 is the only root system with the correct number of vectors but there does not exist a linear transformation mapping the $\mathbf{v}_i$ onto the root vectors in A_7. However, in view of Table 5.1 and considering that the Cartan matrix for $\mathfrak{e}_8$ is invertible over $\mathbb{R}$, we know that the set of vectors $\mathbf{v}_i$ for E_8 can indeed be transformed onto the vectors in the E_8 root system. This correspondence, or the lack thereof, between the vectors $\mathbf{v}_i$ for the compact exceptional groups and the vectors in certain root systems might suggest that the simplifications in Facts 3.2.14 and 4.2.12 generalize to E_8 but not to E_6 or E_7.

Appendix

We record below our Sage-based code for computing $N(F_4, m, s)$. At the time when we wrote this program, we were referring to the matrices in $\mathcal{M}$ as the "maximal subsystems" and the vectors $\mathbf{v}_i$ as the "seed vectors." The seed vectors used in the code can be transformed into the vectors $\mathbf{v}_i$ described in the thesis via a 4×4 unimodular matrix, so the results produced by the code are indeed the table of $N(F_4, m, s)$ that we are seeking. Further, here, the vectors $\mathbf{v}_i$ are treated as row vectors as opposed to column vectors. The program takes approximately 30 minutes to run on a personal laptop.

```
from sage.all import *
import itertools


class F4ms:
    SEED_VECTOR = [vector(ZZ, [1, 0, 0, 0]), vector(ZZ, [0, 1, 0, 0]),
                   vector(ZZ, [0, 0, 1, 0]),
                   vector(ZZ, [0, 0, 0, 1]), vector(ZZ, [1, 0, 0, -1]),
                   vector(ZZ, [0, 1, 0, -1]),
                   vector(ZZ, [0, 0, 1, 1]), vector(ZZ, [0, -1, 1, 1]),
                   vector(ZZ, [1, 1, 0, -1]),
                   vector(ZZ, [-1, 0, 1, 1]), vector(ZZ, [1, 1, -1, -2]),
                   vector(ZZ, [-1, -1, 1, 1])]

    RANK = 4

    MAXIMAL_SUBSYSTEM = list()

    S_REFERENCE = list()

    S = list()

    ELEMENTARY_REFLECTION = (
    matrix(ZZ, [[0, -1, 0, 0], [-1, 0, 0, 0], [0, 0, 1, 0], [-1, -1, 0, 1]]),
    matrix(ZZ, [[1, 0, 0, 0], [0, 0, -1, 0], [0, -1, 0, 0], [0, 0, 0, 1]]),
    matrix(ZZ, [[-1, 0, 0, 0], [0, -1, 0, 0], [0, 0, 1, 0], [0, 0, -1, -1]]),
    matrix(ZZ, [[0, 0, 0, -1], [-1, -1, 0, 1], [1, 0, -1, -1], [-1, 0, 0, 0]]))
```

```
WEYL_GROUP = set()

WEYL_GROUP_CONJUGACY_CLASS = list()

P = matrix(ZZ, 0, RANK)

INCIDENCE_MATRIX = Matrix()

MOBIUS = Matrix()

ROUGH_N = list()

@staticmethod
def normalize(vec):
    for i in range(len(vec)):
        if vec[i] > 0:
            return vec
        elif vec[i] < 0:
            return -vec
    return vec

@classmethod
def generate_maximal_system(cls):
    hermite_form = set()  # Initialized as a set to take advantage of O(1)
    #  time complexity of 'in' operator
    pl = list()
    pl.extend(cls.SEED_VECTOR)
    # Build P, the matrix corresponding to the largest maximal subsystem
    for i in range(len(cls.SEED_VECTOR)):
        cls.P = cls.P.stack(cls.SEED_VECTOR[i])
    for i in range(len(cls.SEED_VECTOR)):
        if pl[i] * 2 not in pl and pl[i] * (-2) not in pl:
            cls.P = cls.P.stack(pl[i] * 2)
            pl.append(pl[i] * 2)
        for j in range(i + 1, len(cls.SEED_VECTOR)):
            if pl[i] + pl[j] not in pl and (pl[i] + pl[j]) * (-1) not in pl:
                cls.P = cls.P.stack(pl[i] + pl[j])
                pl.append(pl[i] + pl[j])
            if pl[i] - pl[j] not in pl and (pl[i] - pl[j]) * (-1) not in pl:
                cls.P = cls.P.stack(pl[i] - pl[j])
                pl.append(pl[i] - pl[j])
    # Find all maximal subsystems
    empty_sys = matrix(ZZ, cls.RANK, cls.RANK)
    empty_sys.set_immutable()
```

```
    cls.MAXIMAL_SUBSYSTEM.append(cls.extract_maximal_subsystem(empty_sys))
    for n in range(1, cls.RANK + 1):
        for ind in itertools.combinations(range(cls.P.nrows()), n):
            h = cls.P[list(ind)].hermite_form().stack(
                matrix(ZZ, cls.RANK + 1 - n, cls.RANK))
            h.set_immutable()
            if h not in hermite_form:
                hermite_form.add(h)
                cls.MAXIMAL_SUBSYSTEM.append(
                    cls.extract_maximal_subsystem(cls.P[list(ind)]))
    hermite_form.clear()

@classmethod
def extract_maximal_subsystem(cls, mtx):
    if mtx == matrix(ZZ, cls.RANK, cls.RANK):
        ms = set()
        ms.add(-1)
        return cls.MaximalSubsystem(ms)
    else:
        hr = mtx.stack(matrix(ZZ, 1, cls.RANK)).hermite_form()
        ms = set()
        for i in range(cls.P.nrows()):
            if mtx.stack(cls.P[[i]]).hermite_form() == hr:
                ms.add(i)
        return cls.MaximalSubsystem(ms)

@classmethod
def generate_s_reference(cls):
    rl = cls.P.rows()
    for i in range(cls.P.nrows()):
        cls.S_REFERENCE.append(set())
    # -2 corresponds to 1/2, -1 corresponds to 0
    for i in range(len(cls.SEED_VECTOR)):
        v1 = cls.SEED_VECTOR[i]
        cls.S_REFERENCE[rl.index(v1)].add((-1, i))
        cls.S_REFERENCE[rl.index(2 * v1)].add((-2, i))
        for j in range(i + 1, len(cls.SEED_VECTOR)):
            v2 = cls.SEED_VECTOR[j]
            if v1 + v2 in rl:
                cls.S_REFERENCE[rl.index(v1 + v2)].add((i, j))
            else:
                cls.S_REFERENCE[rl.index(-v1 - v2)].add((i, j))
            if v1 - v2 in rl:
                cls.S_REFERENCE[rl.index(v1 - v2)].add((i, j))
            else:
```

```
                cls.S_REFERENCE[rl.index(v2 - v1)].add((i, j))

    @staticmethod
    def index_of(lst, obj):
        for i in range(len(lst)):
            if obj in lst[i]:
                return i
        return -1

    @classmethod
    def generate_weyl_group(cls):
        cls.generate_weyl_group_by_recursion(
            MatrixSpace(ZZ, cls.RANK).identity_matrix())

    @classmethod
    def generate_weyl_group_conjugacy_class(cls):
        weyl_group = list(cls.WEYL_GROUP)
        accounted_for = [False] * len(cls.WEYL_GROUP)
        for i in range(len(weyl_group)):
            if not accounted_for[i]:
                size = 0
                for weyl_element in weyl_group:
                    j = weyl_group.index(
                        weyl_element * weyl_group[i] * weyl_element.inverse())
                    if not accounted_for[j]:
                        size += 1
                        accounted_for[j] = True
                cls.WEYL_GROUP_CONJUGACY_CLASS.append((weyl_group[i], size))

    @classmethod
    def generate_weyl_group_by_recursion(cls, elem):
        m = copy(elem)
        m.set_immutable()
        if m not in cls.WEYL_GROUP:
            cls.WEYL_GROUP.add(m)
            for er in cls.ELEMENTARY_REFLECTION:
                cls.generate_weyl_group_by_recursion(er * elem)

    @classmethod
    def generate_s(cls):
        for ms in cls.MAXIMAL_SUBSYSTEM:
            cls.S.append(ms.s())

    @classmethod
    def generate_incidence_matrix(cls):
```

```
    print('Incidence␣matrix␣is␣being␣generated.')
    cls.INCIDENCE_MATRIX = MatrixSpace(ZZ, len(cls.MAXIMAL_SUBSYSTEM),
                                       sparse=True)()
    for i in range(len(cls.MAXIMAL_SUBSYSTEM)):
        sys1 = cls.MAXIMAL_SUBSYSTEM[i]
        for j in range(i + 1, len(cls.MAXIMAL_SUBSYSTEM)):
            sys2 = cls.MAXIMAL_SUBSYSTEM[j]
            c = sys1.compare_to(sys2)
            if c <= 0:
                cls.INCIDENCE_MATRIX[i, j] = 1
            elif c == 1:
                cls.INCIDENCE_MATRIX[j, i] = 1
        if i % 1000 == 1:
            print(str(i) + '␣out␣of␣' + str(
                len(cls.MAXIMAL_SUBSYSTEM)) + '␣generated...')

@classmethod
def generate_mobius_coefficient(cls):
    cls.MOBIUS = MatrixSpace(ZZ, len(cls.MAXIMAL_SUBSYSTEM), sparse=True)()
    nonzero_position = cls.INCIDENCE_MATRIX.nonzero_positions()
    for coord in nonzero_position:
        cls.MOBIUS[coord[0], coord[1]] = sys.maxsize
    for i in range(len(cls.MAXIMAL_SUBSYSTEM)):
        cls.MOBIUS[i, i] = 1
    progress = 0
    for coord in nonzero_position:
        s = coord[0]
        u = coord[1]
        if cls.MOBIUS[s, u] == sys.maxsize:
            cls.MOBIUS[s, u] = cls.generate_mobius_coefficient_by_recursion(
                s, u)
        progress += 1
        if progress % 10000 == 1:
            print(str(progress) + '␣out␣of␣' + str(
                len(nonzero_position)) + '␣generated...')

@classmethod
def save_data(cls):
    cls.generate_weyl_group()
    cls.generate_weyl_group_conjugacy_class()
    del cls.WEYL_GROUP
    save(cls.WEYL_GROUP_CONJUGACY_CLASS, 'F4WGCC')
    del cls.WEYL_GROUP_CONJUGACY_CLASS
    cls.generate_maximal_system()
    save(cls.P, 'F4P')
```

```
    save(cls.MAXIMAL_SUBSYSTEM, 'F4MSS')
    cls.generate_s_reference()
    cls.generate_s()
    del cls.S_REFERENCE
    save(cls.S, 'F4SVAL')
    del cls.S
    cls.generate_rough_n()
    save(cls.ROUGH_N, 'F4RN')
    cls.generate_incidence_matrix()
    cls.generate_mobius_coefficient()
    del cls.INCIDENCE_MATRIX
    save(cls.MOBIUS, 'F4MC')
    del cls.MOBIUS
    del cls.MAXIMAL_SUBSYSTEM

@classmethod
def load_data(cls):
    cls.WEYL_GROUP_CONJUGACY_CLASS = load('F4WGCC')
    cls.S = load('F4SVAL')
    cls.P = load('F4P')
    cls.MAXIMAL_SUBSYSTEM = load('F4MSS')
    cls.ROUGH_N = load('F4RN')
    cls.MOBIUS = load('F4MC')

@classmethod
def generate_mobius_coefficient_by_recursion(cls, s, u):
    if cls.MOBIUS[s, u] != sys.maxsize:
        return cls.MOBIUS[s, u]
    elif s == u:
        cls.MOBIUS[s, u] = 1
        return 1
    elif cls.INCIDENCE_MATRIX[s, u] == 0:
        cls.MOBIUS[s, u] = 0
        return 0
    else:
        mu = -cls.generate_mobius_coefficient_by_recursion(s, s)
        nonzero_position_in_row = cls.INCIDENCE_MATRIX.nonzero_positions_in_row(
            s)
        for t in nonzero_position_in_row:
            if cls.INCIDENCE_MATRIX[t, u] == 1:
                mu -= cls.generate_mobius_coefficient_by_recursion(s, t)
        cls.MOBIUS[s, u] = mu
        return mu

@classmethod
```

```
def generate_rough_n(cls):
    for ms in cls.MAXIMAL_SUBSYSTEM:
        sc = cls.SolutionCount()
        sc.build_from_subsystem(ms)
        cls.ROUGH_N.append(sc)

@classmethod
def consolidate(cls):
    weyl_system = list()
    for info_pair in cls.WEYL_GROUP_CONJUGACY_CLASS:
        weyl_system.append((cls.extract_maximal_subsystem(
            info_pair[0] - MatrixSpace(ZZ, cls.RANK).identity_matrix()),
                        info_pair[1]))
    for s_val in range(1, max(cls.S) + 1):
        progress = 0
        s_val_column = list()
        for t in range(len(cls.MAXIMAL_SUBSYSTEM)):
            if cls.S[t] == s_val:
                s_val_column.append(t)
        multiplier_of_t = dict()
        constituent_entry = list()
        for t in s_val_column:
            t_sys = cls.MAXIMAL_SUBSYSTEM[t]
            multiplier = 0
            for weyl_pair in weyl_system:
                if t_sys.compare_to(weyl_pair[0]) <= 0:
                    multiplier += weyl_pair[1]
            multiplier_of_t[t] = multiplier
            nonzero_position = cls.MOBIUS.nonzero_positions_in_column(t)
            for s in nonzero_position:
                constituent_entry.append((s, t))
        constituent_entry.sort(key=lambda x: cls.ROUGH_N[x[0]].marker)
        constituent_count = list()
        current_marker = -1
        for e in constituent_entry:
            if cls.ROUGH_N[e[0]].marker == current_marker:
                if lcm(len(constituent_count[-1].solution_count),
                       len(cls.ROUGH_N[e[0]].solution_count)) == len(
                        constituent_count[-1].solution_count):
                    constituent_count[-1] = constituent_count[
                        -1].add_times_self(cls.ROUGH_N[e[0]],
                                           cls.MOBIUS[e[0], e[1]] *
                                           multiplier_of_t[e[1]])
                else:
                    constituent_count[-1] = constituent_count[-1].add_times(
```

```
                    cls.ROUGH_N[e[0]],
                    cls.MOBIUS[e[0], e[1]] *
                    multiplier_of_t[e[1]])
            else:
                current_marker = cls.ROUGH_N[e[0]].marker
                constituent_count.append(cls.SolutionCount())
                constituent_count[-1] = constituent_count[-1].add_times(
                    cls.ROUGH_N[e[0]],
                    cls.MOBIUS[e[0], e[1]] * multiplier_of_t[
                        e[1]])
            progress += 1
            if progress % 2000 == 1:
                print(str(progress) + ' out of ' + str(
                    len(constituent_entry)) + ' generated...')
        row_length = 1
        for cc in constituent_count:
            row_length = lcm(row_length, len(cc.solution_count))
        f = open('F4Table Row ' + str(s_val) + '_Unexpanded.txt', 'w')
        _ = f.write('[')
        for i in range(row_length):
            entry = cls.Polynomial()
            for cc in constituent_count:
                entry.add_self(
                    cc.solution_count[i % len(cc.solution_count)])
            _ = f.write(str(entry))
            if i == row_length - 1:
                _ = f.write(']\n')
            else:
                _ = f.write(', ')
        f.close()
        print('Row with s-value ' + str(s_val) + ' has been computed.')

class MaximalSubsystem:

    def ___init___(self, index_set):
        self.subsystem = index_set

    def to_matrix(self):
        if len(self.subsystem) == 1 and -1 in self.subsystem:
            return matrix(ZZ, F4ms.RANK, F4ms.RANK)
        else:
            mtx = matrix(ZZ, 0, F4ms.RANK)
            for i in self.subsystem:
                mtx = mtx.stack(F4ms.P[[i]])
            return mtx
```

```
def s(self):
    if len(self.subsystem) == 1 and -1 in self.subsystem:
        return 1 + 2 * len(F4ms.SEED_VECTOR)
    equiv_class = list()  # List of equivalence classes to be counted
    # for determining s
    rl = F4ms.P.rows()  # List of row vectors of P
    subsystem = self.to_matrix()
    # Begin by identifying all instances of zeros
    zero_par = list()
    equiv_class.append(zero_par)
    for row in subsystem:
        for pair in F4ms.S_REFERENCE[rl.index(row)]:
            if pair[0] == -1:
                equiv_class[0].append(pair[1])
    for row in subsystem:
        for pair in F4ms.S_REFERENCE[rl.index(row)]:
            if pair[0] >= 0:
                i = F4ms.index_of(equiv_class, pair[0])
                j = F4ms.index_of(equiv_class, pair[1])
                if i < 0 and j < 0:
                    new_par = list()
                    new_par.append(pair[0])
                    new_par.append(pair[1])
                    equiv_class.append(new_par)
                elif i < 0:
                    equiv_class[j].append(pair[0])
                elif j < 0:
                    equiv_class[i].append(pair[1])
                elif i != j:  # pair[0] and pair[1] have been
                    # placed into different equivalence classes
                    if i == 0:
                        equiv_class[0].extend(equiv_class[j])
                        del equiv_class[j]
                    elif j == 0:
                        equiv_class[0].extend(equiv_class[i])
                        del equiv_class[i]
                    else:
                        equiv_class[i].extend(equiv_class[j])
                        del equiv_class[j]
    del equiv_class[1:]
    # Now that we have identified all instances of zeros,
    # we identify all instances of 1/2
    half_par = list()
    equiv_class.append(half_par)
```

```
for row in subsystem:
    for pair in F4ms.S_REFERENCE[rl.index(row)]:
        if pair[0] == -2 and pair[1] not in equiv_class[0]:
            equiv_class[1].append(pair[1])
for row in subsystem:
    for pair in F4ms.S_REFERENCE[rl.index(row)]:
        if pair[0] >= 0 and pair[0] not in equiv_class[0] and pair[
            1] not in equiv_class[0]:
            i = F4ms.index_of(equiv_class, pair[0])
            j = F4ms.index_of(equiv_class, pair[1])
            if i < 0 and j < 0:
                new_par = list()
                new_par.append(pair[0])
                new_par.append(pair[1])
                equiv_class.append(new_par)
            elif i < 0:
                equiv_class[j].append(pair[0])
            elif j < 0:
                equiv_class[i].append(pair[1])
            elif i != j:  # pair[0] and pair[1] have been placed
                # into different equivalence classes
                if i == 1:
                    equiv_class[1].extend(equiv_class[j])
                    del equiv_class[j]
                elif j == 1:
                    equiv_class[1].extend(equiv_class[i])
                    del equiv_class[i]
                else:
                    equiv_class[i].extend(equiv_class[j])
                    del equiv_class[j]
del equiv_class[2:]
# Build the other equivalence classes, if any
for row in subsystem:
    for pair in F4ms.S_REFERENCE[rl.index(row)]:
        if pair[0] >= 0:
            i = F4ms.index_of(equiv_class, pair[0])
            j = F4ms.index_of(equiv_class, pair[1])
            if i != 0 and i != 1 and j != 0 and j != 1:
            # Neither pair[0] or pair[1] leads to 0 or 1/2
                if i < 0 and j < 0:
                    new_par = list()
                    new_par.append(pair[0])
                    new_par.append(pair[1])
                    equiv_class.append(new_par)
                elif i < 0:
```

```
                    equiv_class[j].append(pair[0])
                elif j < 0:
                    equiv_class[i].append(pair[1])
                elif i != j:  # pair[0] and pair[1] have been
                    # placed into different equivalence classes
                    equiv_class[i].extend(equiv_class[j])
                    del equiv_class[j]
    # Account for singletons that do not appear in any
    # hitherto constructed equivalence classes
    for i in range(len(F4ms.SEED_VECTOR)):
        if F4ms.index_of(equiv_class, i) == -1:
            sig = list()
            sig.append(i)
            equiv_class.append(sig)
    # Compute s
    s = 1
    if len(equiv_class[1]) > 0:
        s += 1
    s += 2 * (len(equiv_class) - 2)
    return s

# Returns -1 if lower, 0 if equal, 1 if higher, and 10 if incomparable
def compare_to(self, other):
    if self.subsystem == other.subsystem:
        return 0
    elif self.subsystem > other.subsystem:
        return -1
    elif self.subsystem < other.subsystem:
        return 1
    else:
        if -1 in other.subsystem:
            if len(other.subsystem) > 1:
                raise Exception(
                    "Nonempty␣subsystem␣containing␣a␣row␣of␣zero:" + str(
                        other))
            else:
                return -1
        elif -1 in self.subsystem:
            if len(self.subsystem) > 1:
                raise Exception(
                    "Nonempty␣subsystem␣containing␣a␣row␣of␣zero:" + str(
                        self))
            else:
                return 1
        else:
```

```
            return 10

    def __str__(self):
        return self.to_matrix().str()

    def __repr__(self):
        return str(self)

class SolutionCount:

    def __init__(self):
        self.solution_count = list()
        self.solution_count.append(F4ms.Polynomial.zero_polynomial())
        self.marker = 1  # largest prime divisor of len(self.solution_count)

    def build_from_subsystem(self, subsys):
        sysmtx = subsys.to_matrix()
        sysmtx = sysmtx.stack(
            matrix(ZZ, max(F4ms.RANK - sysmtx.nrows(), 0), F4ms.RANK))
        edlst = sysmtx.elementary_divisors()[:F4ms.RANK]
        if sysmtx == matrix(ZZ, F4ms.RANK, F4ms.RANK):
            self.solution_count = [F4ms.Polynomial.m_power(edlst.count(0))]
            return
        max_ed = max(edlst)
        self.solution_count = list()
        for i in range(max_ed):
            self.solution_count.append(
                F4ms.Polynomial.m_power(edlst.count(0)))
        if len(self.solution_count) > 1:
            self.marker = list(factor(len(self.solution_count)))[-1][0]
        for m_mod_max_ed in range(max_ed):
            effective_m_mod_max_ed = m_mod_max_ed
            if m_mod_max_ed == 0:
                effective_m_mod_max_ed = max_ed
            multiplier = 1;
            for ed in edlst:
                if ed != 0:
                    sub_multiplier = 0
                    for k in range(effective_m_mod_max_ed):
                        if ed * k % effective_m_mod_max_ed == 0:
                            sub_multiplier += 1
                    multiplier *= sub_multiplier
            self.solution_count[m_mod_max_ed].times_self(multiplier)

    def add_times(self, sc, c):
```

```
    r = F4ms.SolutionCount()
    lst = [0] * lcm(len(self.solution_count), len(sc.solution_count))
    for mmod in range(len(lst)):
        lst[mmod] = self.solution_count[
            mmod % len(self.solution_count)].add_times(
            sc.solution_count[
                mmod % len(
                    sc.solution_count)],
            c)
    r.solution_count = lst
    r.marker = max(self.marker, sc.marker)
    return r

def add_times_self(self, sc, c):
    for mmod in range(len(self.solution_count)):
        self.solution_count[mmod].add_times_self(
            sc.solution_count[mmod % len(sc.solution_count)], c)
    return self

def expand(self, mod):
    r = F4ms.SolutionCount()
    lst = self.solution_count * (mod // len(self.solution_count))
    r.solution_count = lst
    return r

def __str__(self):
    return str(self.solution_count)

def __repr__(self):
    return str(self)

def __lt__(self, other):
    return self.marker < other.marker or (
            self.marker == other.marker and len(
        self.solution_count) < len(other.solution_count))

def __le__(self, other):
    return self.marker < other.marker or (
            self.marker == other.marker and len(
        self.solution_count) <= len(other.solution_count))

def __eq__(self, other):
    return self.marker == other.marker and len(
        self.solution_count) == len(
        other.solution_count) and self.solution_count == other.solution_count
```

```
    def __ne__(self, other):
        return not self.__eq__(other)

    def __ge__(self, other):
        return self.marker > other.marker or (
                self.marker == other.marker and len(
            self.solution_count) >= len(other.solution_count))

    def __gt__(self, other):
        return self.marker > other.marker or (
                self.marker == other.marker and len(
            self.solution_count) > len(other.solution_count))

class Polynomial:

    def __init__(self):
        self.polynomial = [0] * (F4ms.RANK + 1)

    @classmethod
    def m_power(cls, power):
        p = F4ms.Polynomial()
        p.polynomial[power] = 1
        return p

    @classmethod
    def zero_polynomial(cls):
        return F4ms.Polynomial()

    def add(self, other):
        r = F4ms.Polynomial()
        for i in range(len(self.polynomial)):
            r.polynomial[i] = self.polynomial[i] + other.polynomial[i]
        return r

    def add_self(self, other):
        for i in range(len(self.polynomial)):
            self.polynomial[i] += other.polynomial[i]

    def times(self, c):
        r = F4ms.Polynomial()
        for i in range(len(self.polynomial)):
            r.polynomial[i] = c * self.polynomial[i]
        return r
```

```
def times_self(self, c):
    for i in range(len(self.polynomial)):
        self.polynomial[i] = c * self.polynomial[i]

def add_times(self, other, c):
    r = F4ms.Polynomial()
    for i in range(len(self.polynomial)):
        r.polynomial[i] = self.polynomial[i] + c * other.polynomial[i]
    return r

def add_times_self(self, other, c):
    for i in range(len(self.polynomial)):
        self.polynomial[i] += c * other.polynomial[i]

def ___str___(self):
    s = ""
    for i in reversed(range(len(self.polynomial))):
        if self.polynomial[i] != 0:
            if i == 0:
                if self.polynomial[i] > 0:
                    s = s + '+' + str(self.polynomial[i])
                else:
                    s = s + str(self.polynomial[i])
            elif i == 1:
                if self.polynomial[i] == 1:
                    s = s + '+m'
                elif self.polynomial[i] == -1:
                    s = s + '-m'
                elif self.polynomial[i] > 0:
                    s = s + '+' + str(self.polynomial[i]) + '*m'
                else:
                    s = s + str(self.polynomial[i]) + '*m'
            else:
                if self.polynomial[i] == 1:
                    s = s + '+m^' + str(i)
                elif self.polynomial[i] == -1:
                    s = s + '-m^' + str(i)
                elif self.polynomial[i] > 0:
                    s = s + '+' + str(self.polynomial[i]) + '*m^' + str(
                        i)
                else:
                    s = s + str(self.polynomial[i]) + '*m^' + str(i)
    if len(s) == 0:
        return '0'
    elif s[0] == '+':
```

```
            return s[1:]
        else:
            return s

    def ___repr___(self):
        return str(self)


def main():
    F4ms.save_data()
    F4ms.load_data()
    F4ms.consolidate()


main()
```

Bibliography

[1] R. W. Carter. Conjugacy classes in the weyl group. *Compositio Mathematica*, 25(1):1–59, 1972.

[2] R.W. Carter. *Simple Groups of Lie Type*. Wiley Classics Library. Wiley, 1989.

[3] Dragomir Ž. Djoković. On Conjugacy Classes of Elements of Finite Order in Compact or Complex Semisimple Lie Groups. *Proceedings of the American Mathematical Society*, 80(1):181–184, 1980.

[4] Dragomir Ž. Djoković. On conjugacy classes of elements of finite order in complex semisimple lie groups. *Journal of Pure and Applied Algebra*, 35:1 – 13, 1985.

[5] Tamar Friedmann and Richard Stanley. Counting Conjugacy Classes of Elements of Finite Order in Lie Groups. *European Journal of Combinatorics*, 36, 11 2013.

[6] Tamar Friedmann and Richard P. Stanley. The string landscape: On formulas for counting vacua. *Nuclear Physics B*, 869(1):74–88, Apr 2013.

[7] Brian C. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*. Graduate Texts in Mathematics. Springer New York, 2004.

[8] R.B Howlett, L.J Rylands, and D.E Taylor. Matrix Generators for Exceptional Groups of Lie Type. *Journal of Symbolic Computation*, 31(4):429 – 445, 2001.

[9] Ichiro Yokota. Exceptional Lie groups. *arXiv: Differential Geometry*, 2009.