A Novel Approach to Computing Magic Squares Using Group Theory

Nathan Keough
nathan.keough@my.maryvillecollege.edu

November 21, 2023

Contents

Abstract

This paper introduces a novel algorithm for computing magic squares, exploiting group theory concepts such as permutation representation, group operations, and group actions to encode symmetries. By defining the group operation as composition, the set as a subset of the group of magic squares in a specific order, we may systematically explore the permutations of the group and extrapolate information about the magic squares to generate new magic squares not in the originating set. This vastly reduces computation times for enumerating the solutions to magic squares, while also encoding the symmetries in a manner that is easier to analyze programmatically. Overall, this study reveals the profound connection between magic squares and group theory, offering promising avenues for symmetry-driven algorithms and applications in combinatorial mathematics.

Chapter 0

Introduction

2	9	4
7	5	3
6	1	8

Figure 1: Magic square.

The secrets behind magic squares have eluded mathematicians for millennia. The magic square problem itself is believed to have originated in China around 2200 BCE with the introduction of what are now known as "Lo-Shu" magic squares. These magic squares are defined as 3×3 grids with the numbers 1 through 9 arranged such that the sums of all the rows, columns, and diagonals are the same. Note in Fig. 1 that the rows, columns, and diagonals each add to 15.

This property is what makes a square "magic," lending a connotation of power that is respected by many ancient cultures. In many ways, magic squares have held a mythical reputation in culture due to their unique properties. As mathematics evolved, so did the study of magic squares worldwide, later appearing in India, The Middle East, and Latin Europe. Each rendition of the magic square problem brings with it new questions in

mathematics. The number of solutions to the Lo-Shu magic square problem is well known. However, The number of solutions to other kinds of squares remains a great mystery, especially as the size of the square grows.

There is only one unique solution for the Lo-Shu Square, pictured in Fig. 1. There are extensions to the magic square problem, with variants having larger side lengths (order), different "magic" requirements, or different kinds of element values. For order four, there are 880 unique solutions. For order five, there are $275,305,224$ unique solutions. However, the number of exact unique solutions for order six remains unknown. The number of unique solutions for each order exhibits a kind of exponential growth, and their values have been of interest to mathematicians and hobbyists since the puzzle's inception.

Today, magic squares hold less of a mythical and powerful cultural reputation. Mathematics, at this point in history, has been able to deduce new and creative uses for magic squares in various different fields, especially physics. In 2021, scientists found a connection between electrostatic potentials and magic squares of the 5^{th} and 6^{th} order [CITE]. Similarly, magic squares of the 6^{th} order have been associated with the Weak Force in physics [CITE]. Outside of physics, there are also uses for magic squares in image encryption, specifically, using large squares as chaos maps in chaos-based encryption schemes [CITE]. For these reasons and many others, the continual study of magic squares carries with it the possibility of new breakthroughs in applied mathematics.

Although there are many kinds of magic squares, we will only interest ourselves in "normal" magic squares, which have two main requirements: (1) The square must be filled with the integers from 1 to n^2 inclusive, where n is the order of the magic square, and (2) the sums of the main columns, rows, and diagonals must equal the same integer. This integer S is called the "magic sum" and is calculated by $nS = \frac{n^2(n^2+1)}{2}$ with the right-hand side equating to the sum of the first n^2 positive integers, and the left-hand side equal to the magic sum multiplied by a factor n, the order of the square. Dividing by this factor on both sides satisfies this equation for the magic sum. The number of rows, columns, and

diagonals is equal to $2n + 2$ with n being the order of the square. This is considered the number of "constraint vectors" of a magic square.

2	9	4
7	5	3
6	1	8

$\equiv$

6	7	2
1	5	9
8	3	4

Figure 2: The magic squares are equivalent.

We are also only interested in studying "unique" magic squares. By this, we mean magic squares that are unique up to rotations and reflections. An example of this is taking a magic square A and rotating it 90° or reflecting it about the y-axis any number of times. The resulting square will always be considered equivalent to A. There may also be times when we refer to a "positionally distinct" magic square. In this case, reflections or rotations of a magic square A, results in a "different" square B. Neither of these definitions affect the validity of magic squares; they only affect how we count them. Fig. 2 accurately represents two magic squares that we would consider to be the "same" square.

To help us compute magic squares, we have implemented a custom Computational Algebra System (CAS) written in the Rust Programming Language. This system allows us to construct and manipulate square data to find magic squares. Once we have magic squares, we may do additional processing on them to learn more about their structures. The system is geared for high performance and thus implements some of the best known strategies for working with magic square structures, both mathematically and programmatically.

0.1 Permutations

The main mechanism that we use for encoding a magic square and its properties comes from Group and Number Theory. Specifically, we will study the permutations of magic squares. In the 1770s, Joseph Louis Lagrange studied permutations of the roots of polynomial equations. This led to Galois theory founded by Évariste Galois, which describes what is or is not possible with respect to solving polynomial equations by radicals [CITE]. In modern mathematics, there are many similar situations where studying permutations can help us understand a problem.

Roughly speaking, magic squares may be represented as permutations of n^2 objects. By the definition of a normal magic square, these are the positive integers (mod n^2). The dimension need not matter since the coordinates of elements in a two-dimensional, row-major grid square may be mapped to a one-dimensional sequence, which is a bijection. The square, laid out as a one-dimensional sequence, represents the sequence of integers as a permutation. There are total of $n^2!$ permutations of a square, that is n^2 factorial permutations, n being the order of the square. The permutations of squares represent the different possible configurations of integers in the square, many of them meeting the requirements to be considered magic squares.

We believe that by studying different permutations of squares, including those that are magic, we may be able to describe various kinds of symmetry related to magic squares of specific orders. Treating individual squares as permutations and vice-versa allows us to make use of the properties relating to permutations in general, meaning we can perform unique transformations or actions on them using methods originating from group theory. Additionally, we may define other mathematical properties that allow us to better describe magic squares and their associated symmetries that go beyond simple row and column transposition.

0.2 Enumeration

In our investigation into the inner structures and symmetries of magic squares, we may find it useful to enumerate the magic squares, i.e. exhaustively listing and analyzing every magic square of a specific order. By hand, this is futile, but with high performance programming, we can do this very easily for certain orders and with certain algorithms. Listing magic squares in this way, as we will see, is useful and allows us to generalize certain properties of magic squares, potentially also for higher orders.

Recall that the number of permutations of n objects is $n!$. We can actually exploit this fact to implement an element of ordering for magic squares. There exists a natural ordering from S_n, the group of all permutations on n elements, to $\mathbb{Z}^*_{n!-1}$ in the factor-adic number system, (*) meaning non-negative. This number system, also known as the factorial number system, is a mixed radix adapted for combinatorial systems. In this system, we can express the permutations of n objects in lexicographical order, that is naturally from 0 to $n! - 1$ as a bijection. Using the factor-adic number system's properties we can treat magic squares (or any grid square) as a permutation, and map that permutation uniquely to an integer. More generally, this concept is called a Lehmer Encoding of a permutation on n integers [CITE]. This not only simplifies our intuition of what a unique magic square looks like, but also improves the performance of a magic square computation in some cases, specifically, the cost of copying integers over whole arrays and storing magic squares in memory.

It should be noted that the indices of magic squares in their ordered set do not follow an easily identifiable pattern. Perhaps there is a pattern, but we have no way of identifying it based on our current assumptions of magic squares. In any case, analyzing the frequency of magic squares in their ordered set has not so far proved to be helpful. Exhaustive enumeration of magic squares is presumed to be an NP-Hard problem, making it suitable for certain cryptographic applications. However, it is at least NP; the NP-completeness and classification of the magic square problem is formally unknown. The

complexity of predicting magic squares is unknown, yet, there do exist applications in artificial intelligence for predicting and classifying magic squares [CITE].

Chapter 1

Permutation Groups

1.1 Introduction

The set of magic square solutions falls under the definition of a permutation group. Generally, there are many kinds of permutation groups. For example, Sn is considered a permutation group, but as a whole, it is typically considered a representation of the symmetric group.

Definition 1.1.1

Let S be a set with n elements. Let $\Gamma(S)$ represent the set of permutations of S. Let $(\Gamma(S), \circ)$ be the algebraic structure such that $\circ$ denotes the composition of mappings. Then, $(\Gamma(S), \circ)$ is the symmetric group on S, also known as S_n.

Additionally, a permutation group may be considered a subgroup of the symmetric group on S, a set of n integers.

Theorem 1.1.1

A Permutation Group is a Subgroup of The Symmetric Group

Proof: Recall that $(\Gamma(S), \circ)$ is the symmetric group on S of n elements (S_n).
Let $(H, \circ)$ be a set of permutations of S forming a group under $\circ$. Following the definition of a subgroup, $(H, \circ)$ is a subgroup of $(\Gamma(S), \circ)$,
$\therefore$ a permutation group is a subgroup of S_n. □

A set of magic squares may also be considered a permutation group. The group operation, in this case, is multiplication. This is different from the typical arithmetic operation. Rather, it could be called "composition" since the behavior of multiplication on a permutation resembles function composition. We may think of permutations as functions that bijectively map a set to itself. The product of two functions, $\pi \cdot \sigma$ is the function mapping any element x in the set to $\pi(\sigma(x))$. Generally, the operation is defined for left multiplication, but since these are permutation representations (and to simplify expressions in our code) we will use right multiplication. In other words "π is permuted by σ" or "π is acted on by σ".

It follows that a permutation composed with a permutation results in another permutation. Notably, we can define a function g to be the group operation of composition, and X to be a permutation group. Then $g : X \to X$ denotes that g is a function mapping any element in X back to another element in X which is a group endomorphism. Any composition of permutations results in a permutation.

Theorem 1.1.2

Composite of a Permutation is a Permutation

Proof: Let π, σ be permutations of the set S, and $\pi \circ \sigma$ denotes the composite of the permutations. Recall that a permutation of S is a bijection, and that composition of bijections is a bijection. Their domain and codomains are coincident,
$\therefore \pi \circ \sigma$ must also be a permutation of S. □

Generally, we may study how magic squares behave under multiplication. Magic squares, being permutations in S_{n^2}, exhibit group properties. However, we may not describe a set of magic squares as a group due to the absence of the identity permutation or an identity magic square, inverses, and closure. Rather, we may treat a set K of magic squares as a set of generators for some group. With these generators, we may force closure of the permutation group under composition, thus enabling further group exploration. It is important to note, now, that forcing closure does not always preserve the magic property

of the permutations, as in the original set K - this group simply contains the elements of K, which is a set of known magic squares. The same definition can be applied for the set containing all magic squares of order n. This set, which we call M_n, denotes a permutation group closed under composition, in which all permutations representing magic squares of order n are members.

Due to the properties imposed on M_n, we can treat it as a permutation group, though we generally don't know much about this group, aside from its properties. Taking a more computational route normally involves already knowing some information about the group's generators or internal structure. With these, far more can be learned and analysis of the group then is trivially dependent on computation time. For magic squares, this is not the case since not much is known about how the internal structures and generating subsets generalize for larger order squares. The route we are left with is to use iterative brute force methods to find the generators of M_n so that we may perform further analyses.

Chapter 2

Group Actions

2.1 Introduction

A rational way to understand a complicated or obscure group is to let it act on something. The concept of a group action in the context of group theory defines a set of functions that act on a specific group. These functions may also be considered transformations of the group. The functions applied to elements in the group produce permutations of the group and define the kind of transformations that exist between elements of the group.

Definition 2.1.1

Let G be a group and let S be a set. Define an action on S by G to be a homomorphism $\varphi : G \to Sym\,(S)$

A simple example would be the set of transformations of a Rubix Cube. The set of transformations or actions of a Rubix Cube may be described by the possible moves a player may take to change the cube i.e. rotating specific segments in different directions on various axes. These group actions serve as a basis for the type of configurations we can produce on a Rubix Cube. More generally, they describe the kinds of permutations that can exist in a group, given a set S of permutations and group G that acts on it via

composition.

2.2 Dihedral Groups

We apply the concept of a group action heavily in our code. Specifically, the idea of a group acting on a set defines how we know if a magic square is unique or not. Although the actual square data is flat, we may imagine the magic square in its normal grid square arrangement. We define a unique magic square as a magic square unique up to rotations and reflections. This means that the action of rotating, reflecting, or any combination of rotations and reflections will not result in a new magic square being enumerated. It should be noted that these transformations are magic-preserving since they do not alter the distances between consecutive elements in the grid. Only one magic square from the set of its rotations and reflections is counted; the rest are considered the "same" square.

The set of structure-preserving transformations of a square (or any regular polygon) actually has its own name: the dihedral group denoted D_n. A regular polygon has $2n$ symmetries - n rotational and n reflection symmetries. We may also call it D_{2n} due to this fact. In any case, D_8 is the group of rigid symmetries of a square. This dihedral group encodes all of the orientations of a unique magic square. However, magic squares contain more than four elements, thus the rotations and reflections of magic squares are merely congruent to D_8.

	1	r	r^2	r^3	s	sr	sr^2	sr^3
1	1	r	r^2	r^3	s	sr	sr^2	sr^3
r	r	r^2	r^3	1	sr	sr^2	sr^3	s
r^2	r^2	r^3	1	r	sr^2	sr^3	s	sr
r^3	r^3	1	r	r^2	sr^3	s	sr	sr^2
s	s	sr^3	sr^2	sr	1	r^3	r^2	r
sr	sr	s	sr^3	sr^2	r	1	r^3	r^2
sr^2	sr^2	sr	s	sr^3	r^2	r	1	r^3
sr^3	sr^3	sr^2	sr	s	r^3	r^2	r	1

Figure 2.1: Cayley Table of D_8

Computationally, we can select a unique element from the square's dihedral group consistently by selecting the first element from the Cayley table of transformations. Alternatively, we can generate the full set of the dihedral group. This table, shown in Fig. 2.1, represents the transformations of the dihedral group D generated by $< r, s | r^4 = s^2 = (sr)^2 = 1 >$. We treat the table like a set and filter another set containing various arbitrary magic squares such that the resulting set only contains unique elements. This is useful for brute-force algorithms where we may not have total control over element uniqueness in a set. The same applies for certain constructive algorithms, but still, we ensure that we do not duplicate magic squares in our sets, resulting in better analysis and more efficient runtimes.

2.3 Group Actions on Magic Squares

In the same manner that the dihedral group applied to the permutation of a square results in the same square, the dihedral group applied to a magic square always results in a magic square. This property represents a symmetry of a magic square. Specifically, it represents the rotational and reflective symmetry of an arbitrary magic square. Based on this, the application of group actions on sets presents itself as a potential medium for representing group symmetry in the magic square permutation group.

If we assume that the full set of magic squares is known, the group actions of the set would be the full set of transformations between magic squares in the set. One thing we are interested in is if these actions can tell us something new about the set that we didn't know before. Are there additional magic-preserving sets of permutations, like those from the dihedral group? How many symmetries exist for magic squares of a specific order? The answers to these questions may reveal new patterns, lead to more efficient methods, or allow us to use smaller search-spaces for larger orders.

Computationally, we can produce the transformation between two magic squares, π and σ, by factoring out the permutation α that acts on π to produce σ. The factorization of permutations is a well known problem and is computed by inverting part of the original composition:

$$\alpha = \pi^{-1}\sigma \qquad \text{By definition of function inverse, composition} \qquad (2.1)$$

The full set of transformations can be computed by taking the Cartesian product defined as:

$$M_n \times M_n = \left(\mu \in \mathcal{P}\left(\mathcal{P}\left(M_n\right)\right) | \exists \pi, \sigma \in M_n : \mu = (\pi, \sigma)\right) \qquad \mathcal{P} \text{ denotes power set} \qquad (2.2)$$

and modifying the set such that:

$$\beta = \{f(\mu) = \alpha | \mu \in M_n \times M_n\} \tag{2.3}$$

We will refer to this modified product as which represents the set of transformations or group actions on M_n.

Typically, when faced with actually computing magic squares, we only have access to subgroups of M_n. We denote an arbitrary subgroup of M_n as K_n, more specifically, $K_n \leq M_n$. Many methods for computing magic squares exist. As such, various distinct subgroups are possible. Regardless of the methods used, the group actions still may reveal information about M_n despite its incompleteness. We believe that additional magic squares can be extrapolated based on the existing knowledge gained from analysis of its group actions.

Chapter 3

Solving Magic Squares

3.1 Introduction

Ideally, we would like to have enough information about M_n so that by computing one magic square, we can generate M_n. This generally is the case for order three, considering there is only one solution. However, this is not very interesting. For order four, there are 880 unique solutions, so we should expect to have to compute more than one magic square to generate the full set of solutions in M_4. In practice this seems logical, considering there isn't much that two arbitrary magic squares have in common besides their group properties. We will have to dig deeper in order to find patterns amongst elements in M_n.

It should also be noted that the elements of are typically never magic. The idea is that given a group action α in β, and a magic square π from K_n, then $\pi\alpha$ may potentially be a new magic square, not already in K_n. Not all elements of β produce magic squares on elements of K_n. Discovering the space of solutions given an action is something we wish to explore. One reason for this is that certain magic squares may exist in distinct cosets that have a distinct permutation or permutations that act on it. These cosets may provide some hints about the symmetries on M_n.

3.2 Current Method

Solving magic squares and producing the set K_n is itself a different problem. Programmatically, we use Rust in a multithreaded environment to squeeze out as much performance as we can. We also utilize message passing on a single process, specifically, a Multiple Producer, Single Consumer (MPSC) Queue to produce results on all threads available to the system. This method is widely used in parallel systems that enable data communication primitives. In Rust, we are able to do this very easily.

These results are then collected into a set-like data structure. We will use a `BTreeSet` (Balanced Binary Search Tree) to store our magic squares for the fast lookup, insertion, and concurrency potential. For our initial proof of concept, we computed the first 440 magic squares of order four. This computation took about 40 minutes on 16 threads, each running at 4.2 GHz. Of these 440 magic squares, 239 are unique. Based on the time taken to compute the initial set, we anticipate that a full brute-force computation of M_n would require about 10.67 hours.

3.3 New Method

We begin by reading the set of 440 cached magic squares into a set. We then filter the set to reduce duplication, hence the resulting set only contains unique squares. The resulting unique set, in our case, contains 239 elements and we will consider this set as K_4. Up to this point, we have only defined K_4 as a set containing 239 magic squares in relative order. It should be noted that these are not necessarily the first 239 unique magic squares in order, but we will eventually explore this difference in our analysis.

Once K_4 is computed, we may then start producing the actions of K_4. We compute the set of actions by generating the Cartesian Product $K_4 \times K_4$. The resulting set contains every possible unique ordered pair of elements in K_4. The size of this set is $|K_4|^2$. For each pair $(\pi, \sigma)\,|a, c \in K_4$, compute $\pi^{-1}\sigma$. The set after this operation contains every action on

K_4. Filtered for uniqueness, this set contains 22,490 elements which we call β.

Now, compute the Cartesian Product $K_4 \times \beta$. The resulting set of size $|K_n| \cdot |\beta|$ contains elements $(s, b) | s \in K_n, b \in \beta$. For each element in K_4, compute the dihedral group of the magic square s as the set D, then element-wise compute $db = m$ for each element $d \in D$. For each element m in Db, if m is magic, then collect the element into a set K'; if not, then do nothing with m. Finally, we filter unique elements from K'. The result is an extension of K_4 which we call $Ext(K_4)$ where $|Ext(K_4)| \geq |K_4|$. For this particular example with order four magic squares and the specific pre-computed K_4, $|Ext(K_4)| = 880$ and $Ext(K_4) = M_4$.

Chapter 4

Analysis

4.1 Introduction

The fact that $|K_4| \leq 880, |Ext\,(K_4)| = 880$ not only tells us that we've found a great computational performance improvement, it also hints to us that there are possibly more symmetries than just the 8 induced by the action of the dihedral group. These additional symmetries may be key to producing more efficient algorithms as well as for making profound generalizations on larger order magic square symmetry.

Overall, it will be very important for us to not only make statements based on experimental data, but to back up our claims with logical reasoning. In any case, we will typically begin with experimental data. The algorithm outlined in the previous section enables a greater range in computational capability, however, it also opens a door to a whole new set of questioning regarding the structure of the solution space for M_4. One of the things we would like to do with this information is to understand why we are able to exploit magic square symmetry in this way so that we may apply the same method for larger orders.

4.2 Computational Analysis

The result of $Ext\,(K_4)$ with $|K_4| = 239$ (in relative order) computes in 71 seconds. This is an amazing computational performance improvement over the remaining 10 hours for brute-force enumeration. The code is parallel across threads, thus we may conclude that the implementation we've chosen is adequate. Of course, the sizes of the sets we work with are still rather large. Only now, we may start to consider that storage is our most limiting factor. As $|K_n|$ approaches $|M_n|$, the amount of memory required to continue working with the sets grows extremely fast. In one case, we used M_n as our K_n and the program ran out of memory. This is suboptimal, considering we would like to be able to compute larger orders with this algorithm.

The simplest solution to this is additional parallelization, although there also exists the possibility for set partitioning. The combination of these solutions lands the problem in the realm of High-Performance Computing (HPC) for proper multi-process computation using extremely high CPU counts, super-scale storage solutions, and vast quantities of working memory. Given our current time and budget constraints, we see partitioning as the most viable solution to cutting down on the memory/storage requirement at the cost of program runtime.

The amortized runtime analysis of the algorithm is difficult considering the amount of abstraction used in the body of the code. Given that we compute $\beta = K_n \times K_n$ and $\beta \times K_n$, we may say the algorithm runs somewhere around $O\,(n^3) + C$. This beats the performance of iterative brute-force which is $O\,(n!) + C$. With this in mind, the set reduction incurred by looking at group actions in this way is likely vast. The largest possible set with $n = 239$ is $\approx 1.36519 \times 10^7$ whereas $16! \approx 2.09227 \times 10^{13}$.

4.3 Generating Sets

While the maximum set size difference is great for $n = 239$, it is entirely dependant on our choice of Kn. In a way, the choice of Kn is completely arbitrary. We will choose a different set now, also calling it K_4, however, we will select the first 440 magic squares from M_4. This set contains 232 unique magic squares, produces a set of 20,210 actions, but only generates 871 unique magic squares from $Ext\,(K_4)$. This is interesting since it is the first choice of K_4 that actually fails to compute M_4. What if we chose a different set using a different sampling method?

One of our hypotheses was that certain types of magic squares have different kinds of influence on the output of $Ext\,(K_4)$, and we thought it could be related to the order of the permutations in the set. Specifically, the order of a permutation is the `lcm` of all its cycle lengths and represents the smallest integer $n \ni p^n = e$. Given our assumption, we selected four elements from each conjugacy class and ran the resulting set through $Ext\,(K_4)$. The set was small, only 104 elements, yet still produced M_4. Following this, elements were iteratively were iteratively removed from the set via backtracking if M_4 could still be computed without them. The resulting set contained only 50 elements and $Ext\,(K_4)$ still resulted in M_4. This discovery was extremely valuable since we could now compute M_4 from 50 magic squares alone with a total run time of about 1 second.

Initially, we were not entirely sure what the smallest set K_n that produces M_n represented, assuming one existed. Our main theory was that the elements in the minimal set are representatives of the cosets of M_n, where each element encodes with it some type of group symmetry. If this is the case, then the minimal set possibly could contain fewer than 50 elements, dividing the order of the group (due to Lagrange's theorem). Attempts at Monte Carlo simulation to find smaller sets were not successful due to time constraints, but did suggest that the size of the minimal set is very close to 50.

However, revisiting our definitions tells us that K_n is a set of generators for a group containing the elements of M_n. This implies that the smallest set K_n that produces M_n is

considered a "minimal set S that generates G" which we use interchangeably with "generating subset of minimal cardinality." For finite, solvable groups, the cardinality k of the minimally generating subset is at least $\lceil \log_p |G| \rceil$ (where p is the smallest prime dividing G), which is a byproduct of Lagrange's Theorem. Using the "Groups, Algorithms, and Programming" system (GAP), an open-source programming model for computational discrete algebra [CITE], we found that the group containing all magic squares of order four has an order of 10,461,394,944,000, which is equal to the order of the group generated by our small K_4. This may indicate that these groups are identical. With $k = 2$, $|G|$=10,461,394,944,000, we found that the minimal cardinality of the subset that generates G is 44 (rounded up to the nearest integer).

In other words, the group containing all magic squares of order four requires that our initial generating set contain at least 44 magic squares as generators, assuming our group is finite and solvable. This is very close to the size of our small set ($|K_n| = 50$). Finding a group's minimally generating subset, whose length is the rank of the group, would allow us to more accurately describe the group's structure, relationships, and dependencies among elements of the group. The process for actually describing a generating set with minimal cardinality is still quite difficult, even for computational approaches. In general, it's easiest to use methods that don't require a total enumeration of the group. Also known as the Rank Problem, the concept of computing a generating subset of a group with minimal cardinality is one of the most well known and challenging problems in algorithmic group theory.

One other hypothesis we had is that different sampling methods or random distributions may allow us to build more interesting sets to use as K_n. The relative distance between elements in the set of 50 magic squares appears random - the set contains no discernible pattern. Our code implements some functionality that enables random sampling from M_n using various distributions. The main goal of testing different distributions is to explore the relative variety generated by using different sampling

methods. Of course, the next easiest method would be to build K_n from the various constructive algorithms that exist for magic squares. It would be interesting to see what kind of variety and sampling distributions of M_n we can achieve by using certain algorithms to produce small sets of solutions as well as what effect this has on the output of $Ext\,(K_n)$.

4.4 Magic-Preserving Group

The action of the dihedral group on any magic square will always result in a magic square. This is true for all magic squares, since the actions of rotating and reflecting a square are magic-preserving. In a similar manner, other kinds of transformations exist that when applied to a magic square, also produce a magic square. This group, in the context of group actions on sets and element transformations, reveals a high degree of symmetry and can be used to further improve the computational performance of enumerating magic squares.

One of the things that we can do with our code is collect the transformations of a set. For example, given a set of permutations, we can collect the actions between elements in the set. For the group action magic square solver, we collect the actions of K_4 into β. Then, when K_4 is being conditionally extended, we insert the actions that are not magic-preserving into a rejection set R. In other words, if $\pi\alpha$ is not magic, for $\pi \in K_n, \alpha \in \beta$, then α is rejected - it cannot be a member of the magic-preserving group. Once $Ext\,(K_4)$ has completed, $MP = \beta \setminus R$ is the magic-preserving group.

To our surprise, $|MP| = 4$ (excluding rotations and reflections), implying three new magic-preserving symmetries in addition to the identity element. These permutations are:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	3	2	4
9	11	10	12
5	7	6	8
13	15	14	16

7	5	8	6
15	13	16	14
3	1	4	2
11	9	12	10

7	8	5	6
3	4	1	2
15	16	13	14
11	12	9	10

Figure 4.1: Members of the magic-preserving group.

It is proven that $\forall \pi \in M_4, \alpha \in MP, \pi\alpha \in M_4$. One question that arises from the knowledge of this set is on whether or not $\pi\alpha$ should still be considered uniquely as π, since this is how we treat the actions of the dihedral group on π.

Chapter 5

Conclusions

5.1 Introduction

While confident in the performance of our new algorithm for solving magic squares, we have yet to fully understand the implications of using magic squares in a group-theoretic context. The application of a group structure to the set of magic squares of a specific order makes the intuition for computations simpler. However, it also adds a significant amount of complexity and increases the number of assumptions we use to compute them.

5.2 Alternative Methods

Compared to alternative methods for computing magic squares, we recognized very few approaches that use a similar permutation group representation. Some of our initial assumptions and applications of group theory almost transform the grid-square construction into a different kind of problem, so we find it somewhat difficult to make direct comparisons to more standard models of the magic square problem. In the few examples we found of solving magic squares using group theory, we found that researchers tended to apply general linear group structures over permutation groups. While this is a

completely different approach, bringing with it other kinds of questions and results, this is a perfectly valid way to go about solving magic squares as the information in general linear groups is more geared towards grid/matrix problems, and thus contains a more intuitive structure. Standard models, however, are typically very well defined and date back to some of the earliest research into the magic square puzzle. Oftentimes these standard models use methods on magic squares that do not encapsulate the full solution set, rather, they typically aim to produce magic squares in the most efficient/intuitive way regardless of the specific value of a solution. This can be useful still, in cases where one only needs a solution, not necessarily information about all solutions.

5.3 Generalizations

For almost all of our examples, the algorithm we used to generate the complete set of magic squares was just made for order four. One generalization that we've considered and would like to see, is if this algorithm can successfully be applied to magic squares with orders greater than four. The code for this algorithm was designed to be order-agnostic meaning that, in theory, we could very easily just use a different order of magic square and the same processes and computations would still be applied to potentially generate the full set of solutions. Of course, this assumes that solutions of magic squares of different orders can be computed using the same transformations as order four.

We tested this on magic squares of order five but due to the sheer size of the solution space for brute force calculations, we were unable to obtain a large enough initial generating set to produce new terms. Our hypothesis is that with additional known magic squares of order five, we may be able to generate additional solutions using our method. The same could potentially be said for magic squares of order six and above, if proven to be true for smaller orders.

Additionally, we are curious if this algorithm still yields solutions for other kinds of

magic squares. For example, does removing the constraint on the diagonals affect the way permutations interact with one another in a group structure? What about for pandiagonal magic squares? Generalizing our algorithm for other kinds of magic squares could introduce more or less complexity depending on the assumptions that other types of magic squares make. They could also yield wildly different results based on their own group structure analysis.

5.4 Open Questions

Aside from the questions induced by thinking about potential generalizations of our algorithm, there are still some things we don't know but would find useful in our discovery. Many questions that we've had depend on the individual assumptions we've made up to that point. Eliminating or potentially proving our assumptions will almost certainly improve our results.

Does random sampling from a given distribution improve the relative variety of our generators over normal methods? Given the difficulty of finding a minimally generating set, we would like to know to what degree randomness could affect our ability to create smaller or "stronger" generating sets.

To what degree might "supercomputing" improve our results? This sort of problem could be geared more for high-performance or massively distributed systems. Our code itself is actually thread-agnostic in some cases which could enable increased performance and throughput of computations.

Are there better ways to represent magic squares as algebraic objects? Our paper suffers from formality. Since developing the algorithm in code was always a priority

over formalization of the underlying mathematics, our terminology and definitions might inadvertently be limiting our ability to perform efficient analysis.

Are our results unique? To our knowledge, our method introduces something new in the way of solving magic squares, however we are not completely confident in its uniqueness according to group theory. We believe with moderate confidence that our method uses results from theorems that haven't necessarily been formalized in this paper.

Bibliography

[1] Albert Einstein. "Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]". In: *Annalen der Physik* 322.10 (1905), pp. 891–921. DOI: http://dx.doi.org/10.1002/andp.19053221004.