THEORETICAL AND COMPUTATIONAL METHODS FOR LATTICE POINT ENUMERATION IN INSIDE-OUT POLYTOPES

A thesis presented to the faculty of
San Francisco State University
In partial fulfillment of
The requirements for
The degree

Master of Arts
In
Mathematics

by

Andrew William van Herick

San Francisco, California

August, 2007

CERTIFICATION OF APPROVAL

I certify that I have read *Theoretical and Computational Methods for Lattice Point Enumeration in Inside-out Polytopes* by Andrew William van Herick, and that in my opinion this work meets the criteria for approving a thesis submitted in partial fulfillment of the requirements for the degree: Master of Arts in Mathematics at San Francisco State University.

Matthias Beck
Assistant Professor of Mathematics

Federico Ardila
Assistant Professor of Mathematics

Joseph Gubeladze
Associate Professor of Mathematics

THEORETICAL AND COMPUTATIONAL METHODS FOR LATTICE POINT ENUMERATION IN INSIDE-OUT POLYTOPES

Andrew William van Herick
San Francisco State University
2007

The theory of inside-out polytopes was developed to count the number of lattice points within a polytope and outside a set of excluded hyperplanes. One theoretical approach uses mixed integer programming. Another sums the Ehrhart rational generating functions over each region of the inside-out polytope. We give a polynomial time algorithm for computing the region descriptions when the dimension of the polyhedron is fixed. This yields a polynomial time algorithm for computing the lattice points count in rational semi-open polytopes of fixed dimension. We discuss a C++ implementation of this algorithm, along with various optimizations, and compute a number of new examples, including nowhere-zero flow polynomials for all connected bridgeless graphs with at most six vertices, as well as the number of four by four magic squares with distinct entries.

I certify that the Abstract is a correct representation of the content of this thesis.

______________________________ ______________

Chair, Thesis Committee Date

ACKNOWLEDGEMENTS

Thanks are in order to those who have helped along the way: to Matthias Beck, for your insights, patience and willingness to work at a new parent's pace; to Federico Ardila and Joseph Gubeladze for taking the time to read and comment on the text; to Serkan Hosten, for your insights into linear programming; to the Center for Computing in Life Sciences, in particular Dragutin Petkovic and Mike Wong, for support with and use of the high speed cluster; and most importantly, to my wife, Kristy, and son, Connor, for your emotional and financial support, and for putting up with the demands on my time and energy while working on 'da tesis'.

TABLE OF CONTENTS

Chapter 1

Introduction

Many counting problems reduce to finding the number of lattice points within a polytope that lie outside a union of hyperplanes. Some notable examples include weak magic and semi-magic squares [6] and possibly-zero k-flows on graphs [8]. The theory of inside-out polytopes [7] developed around an interest in proving results for problems involving the lattice points of a polytope that are not contained a union of hyperplanes. This work focuses on methods for actually computing such lattice point counts.

In 1994 Alexander Barvinok introduced the first algorithm [2] for counting the number of lattice points in a polytope known to run in polynomial time, provided that the dimension of the polytope is fixed a priori. Barvinok's publication spurred the development of software implementations such as LattE [11] and Barvinok [21], now available with general public licenses. Though [7] directly implies methods for using Barvinok's algorithm to count lattice points in a polytope minus an arrangement of hyperplanes (see Theorem 3.1 and Equation 4.2), as far as we are aware, no software implementations yet exist to solve such problems.

The algorithms implied by Theorem 3.1 and Equation 4.2 of [7] provide solutions to the problem of counting lattice points in an *open* polytope minus a set of hyperplanes. By themselves the implied algorithms cannot solve the problem of counting lattice points in a *closed* polytope minus a set of hyperplanes, nor can they deal with problems where certain faces of the closed polytope are omitted from consideration. One generalization of closed and open polytopes is the semi-open polytope. This is a closed polytope minus some of its faces. In this work, we use Equation 4.2 of [7] to offer a polynomial-time algorithm for computing the number of lattice-points in any semi-open polytope of fixed dimension that lie outside a union of hyperplanes.

Much of the work performed in the course of this research project went into trying to find a relationship between inside-out polytopes and the results of [4] using the techniques from Chapter 4. This effort was full of many interesting turns, but on the whole, largely unsuccessful at producing a cohesive body of results. Chapter 4 represents the portion of this research that pertains directly to computing generating functions for inside-out polytopes.

Part of this research involves a software implementation of the algorithm for open polytopes implied by Equation 4.2 called `ioehrhart`. As many of the functions written in support of this software are valuable in their own right, we have included them in the form of a callable C++ library called the Inside-out Polyhedral Library (IOP). Using a variation of `ioehrhart` designed for parallel processing over a network, we have computed generating functions that count the number of 4×4 magic and semi-magic squares with distinct entries [6]. For magic squares we have generated counts both as a function of magic sum, and as a function of upper bound on the matrix entries. Our results for semi-magic squares provide counts only as a function of upper bound on the entries. These results can be found among the examples given in Chapter 6 and on the compact disk accompanying this work. In some sense, inside-out polyhedra represent natural generalizations of both polyhedra and hyperplane arrangements. We hope that the IOP library may serve as a future framework in C++ for operations involving polyhedra and/or hyperplane arrangements.

Chapter 2

Notation and Background Material

2.1 Definitions/Notation

We represent column vectors in $\mathbb{R}^d$ using bold faced symbols and describe their components by subscripting the corresponding plain-faced symbol, so that z_i denotes the i^{th} component of $\mathbf{z}$. A bold faced $\mathbf{0}$ or $\mathbf{1}$ refers to the zero vector and to the vector whose entries are all 1's, respectively. For all vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ we say that $\mathbf{x} \leq \mathbf{y}$ if and only if $x_i \leq y_i$ for all $i = 1, \ldots, d$.

By $(a_{ij}) \in \mathbb{R}^{m \times d}$ we refer to an $m \times d$ matrix whose entry in the i^{th} row and j^{th} column is a_{ij}. For $A \in \mathbb{R}^{m \times d}$ the notation $A^{\intercal}$ denotes the transpose matrix of A.

Hence $\mathbf{a}^{\intercal}$ is the row vector associated with $\mathbf{a}$, and $\mathbf{a}^{\intercal}\mathbf{z} = \sum_{i=1}^{d} a_i z_i$ is the standard Euclidean inner product of $\mathbf{a}$ with $\mathbf{z}$.

A subset E of $\mathbb{R}^d$ is *convex* if for all $\mathbf{x}, \mathbf{y} \in E$, the line segment

$$\{\lambda \mathbf{x} + (1-\lambda)\,\mathbf{y} : 0 \leq \lambda \leq 1\}$$

is also contained in E. The *convex hull* of E, denoted $\operatorname{conv}(E)$, is intersection of all convex sets containing E. Given points $\mathbf{v}, \mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^d$ we say $\mathbf{v}$ is a *convex combination* of $\mathbf{v}_1, \ldots, \mathbf{v}_n$ if there exist $\alpha_1, \ldots, \alpha_n \geq 0$ with $\sum_{i=1}^{n} \alpha_i = 1$ such that $\mathbf{v} = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i$. If $V \subseteq \mathbb{R}^d$ is finite, then $\operatorname{conv}(V)$ is the set of all convex combinations of vectors in V.

An *affine combination* of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ is linear combination

$$\sum_{i=1}^{n} \alpha_i \mathbf{v}_i$$

such $\sum_{i=1}^{n} \alpha_i = 1$. An *affine subspace* is the translation of a linear subspace, i.e. of the form $\mathbf{v} + U$ for some linear subspace U of $\mathbb{R}^d$. For a convex set C we define the *affine hull* of C (denoted $\operatorname{aff}(C)$) to be the smallest affine subspace containing C. The dimension of an affine subspace is the dimension of its associated linear subspace.

For $\mathbf{a} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, we define the notations

$$H(\mathbf{a}, b) := \left\{\mathbf{x} \in \mathbb{R}^d : \mathbf{a}^{\intercal}\mathbf{x} = b\right\} \text{ and } K(\mathbf{a}, b) := \left\{\mathbf{x} \in \mathbb{R}^d : \mathbf{a}^{\intercal}\mathbf{x} \leq b\right\}.$$

For $A \in \mathbb{R}^{m \times d}$ and $\mathbf{b} \in \mathbb{R}^m$ we define

$$P(A, \mathbf{b}) := \left\{x \in \mathbb{R}^d : A\mathbf{x} \leq \mathbf{b}\right\}.$$

A *hyperplane* is any set of the form $\left\{\mathbf{x} \in \mathbb{R}^d : \mathbf{a}^\intercal \mathbf{x} = b\right\}$. This includes both the empty set and $\mathbb{R}^d$ as hyperplanes, which we refer to as the *empty* and *degenerate hyperplanes*, respectively. A *half-space* is any set of the form $\left\{\mathbf{x} \in \mathbb{R}^d : \mathbf{a}^\intercal \mathbf{x} < b\right\}$ or $\left\{\mathbf{x} \in \mathbb{R}^d : \mathbf{a}^\intercal \mathbf{x} \leq b\right\}$. Sets of this form are *open* and *closed half-spaces*, respectively. For a subset X of $\mathbb{R}^d$ we say a hyperplane H *supports* X (or is a *supporting hyperplane* of X), if $H \cap X \neq \emptyset$ and X is contained entirely in one of the two closed half-spaces defined by H. We note that

$$P(A, \mathbf{b}) = \bigcap_{i=1}^{m} K(\mathbf{a}_i, b_i)$$

where the $\mathbf{a}_i^\intercal$ are the rows of A.

By *lattice* we mean any discrete subgroup of $\mathbb{R}^d$. In particular the term *integer lattice* refers to $\mathbb{Z}^d$. A *lattice basis* is any set of linearly independent vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ where every member of the lattice can be expressed as $\sum_{i=1}^k \alpha_i \mathbf{v}_i$ for some $\alpha_1, \ldots, \alpha_k \in \mathbb{Z}$.

2.2 Computational Analysis

If f and g are two real valued functions defined on the positive integers we say that f is $\mathcal{O}(g)$ if there exists $\alpha \in \mathbb{R}_{>0}$ such that $|f(n)| \leq \alpha |g(n)|$ for all $n \geq 0$. Throughout our analysis we assume a binary encoding and refer to the number of bits required to encode a problem description as the *size* or *length of the input*. The *runtime* or *time complexity function* $t(l)$ of an algorithm is the largest number of elementary steps required to solve a problem with input length l. We a say that an algorithm A *runs in* $\mathcal{O}(f)$ *time* if its time complexity function is $\mathcal{O}(f)$. We say that a problem A *runs in polynomial time* if there exist polynomial functions p and q such that A runs in $\mathcal{O}(p(l))$ time, where l is the input length, and all intermediate data can be stored in $\mathcal{O}(q(l))$ bits.

2.3 Polyhedra and Polytopes

A *polyhedron* is a finite intersection of closed half-spaces. A *polytope* is the convex hull of a finite set of points in $\mathbb{R}^d$. A fundamental result in the theory of polytopes says that the set of polytopes is precisely the set of bounded polyhedra. (See [22, Theorem 1.1] and [9, Theorem 9.2] for two fundamentally different proof developments.) Hence for any polytope P we have two alternate ways to describe P, either as $\operatorname{conv}(\mathbf{v}_1, \ldots, \mathbf{v}_n)$ for some $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^d$, or as $P\left(A, \mathbf{b}\right)$ for some $A \in \mathbb{R}^{d \times m}$ and $\mathbf{b} \in \mathbb{R}^m$. Depending on the problem at hand, one form often proves more convenient than the other. Conversion between the two description methods is non-trivial in general; hence algorithms involving polytopes specify the particular form of both the input and/or the output. Where the description method is relevant, we use the terms *H-description* (*half-space description*) and *V-description* (*vertex description*) to refer to the half-space and point-set forms, respectively.

The *dimension of a polyhedron* (or, more generally, of any convex set) refers to the dimension of its affine hull. The term *d-polyhedron* (respectively, *d-polytope*) refers to a d-dimensional polyhedron (polytope).

Given a polyhedron P, the *faces of P* are precisely the sets of the form

$$P \cap H\left(\mathbf{a}, b\right)$$

where $\mathbf{a}^\intercal \mathbf{x} \leq b$ holds for all $\mathbf{x} \in P$. Since our definition of hyperplanes includes the degenerate and the empty hyperplane, our definition of faces includes both P and the empty set among the faces of P. Faces of polyhedra are again polyhedra, and likewise with polytopes. For any polyhedron an intersection of faces is again a face. The term *k-face* refers to a face of dimension k. The proper faces of P are the nonempty k-faces of P with $k < \dim\left(P\right)$. For a d-dimensional polyhedron the terms *facets*, *edges*, and *vertices* to refer to the $(d-1)$-, 1-, and 0-faces, respectively. In the case where P is a polytope and the vertices of P have rational (respectively, integer) coordinates, we call P a *rational* (*integral*) *polytope.* If P is rational we define the *denominator of* P as the least common multiple of the denominators taken from the coordinates of the vertices of of P. Rational polytopes have half-space descriptions given by *rational hyperplanes*, those hyperplanes whose defining equations have integer coefficients.

If $S \subseteq \mathbb{R}^d$ is the intersection of open and/or closed half-spaces, we call S a *semi-open polyhedron* and, if bounded, a *semi-open polytope.* If $S \neq \emptyset$ is a semi-open polyhedron, its topological closure $\overline{S}$ is a polyhedron with $S = \overline{S} \setminus \bigcup \mathcal{F}$ for some subset $\mathcal{F}$ of the proper faces of $\overline{S}$. A similar statements holds for polytopes. We call S a *rational semi-open polytope* if $\overline{S}$ is a rational polytope.

One valuable construction in the theory of polytopes is the idea of coning over a polytope. If $\mathbf{v}_1, \ldots, \mathbf{v}_k$ represent the vertices of a polytope $P \subseteq \mathbb{R}^d$, then for $i = 1, \ldots, k$ we set $\mathbf{w}_i = (\mathbf{v}_i^\intercal, 1)^\intercal$ and define

$$\operatorname{cone}(P) := \left\{ \sum_{i=1}^{k} \alpha_i \mathbf{w}_i : \alpha_i \geq 0 \ \forall i \right\}.$$

This construction proves especially useful to theories of lattice point enumeration in that

$$\lambda P = \left\{ \mathbf{x} \in \mathbb{R}^d : (\mathbf{x}^\intercal, \lambda)^\intercal \in \operatorname{cone}(P) \right\}$$

for all $\lambda \in \mathbb{R}^+$.

Polyhedra represent closed sets in $\mathbb{R}^d$. Only full-dimensional polyhedra have non-empty topological interiors. Hence many statements, which are valid for the interior of a fully-dimensioned polyhedron, fail for isomorphic polyhedra of less than full dimension. One would like such theorems to work for polyhedra which are isomorphic, regardless of dimension of the ambient space in which they live. This can be accomplished by considering the relative interior of the polyhedron with respect to its affine hull. For a convex set C, we define the *interior of* C (denoted C°) as the relative interior of C taken with respect to $\operatorname{aff}(C)$. The relative interior of a point is simply the point.

2.4 Set Dilation

For a subset X of $\mathbb{R}^d$ and a real number $t > 0$, the *dilation* of X by t is the set $tX := \{tx : x \in X\}$. For all hyperplanes we have $tH(\mathbf{a}, b) = H(\mathbf{a}, tb)$, with similar statements holding for half-spaces. We can think of the dilation of a polyhedron as a uniform dilation of the defining half-spaces so that $tP(A, \mathbf{b}) = P(A, t\mathbf{b})$.

2.5 Basic Ehrhart Theory

Classic theories of lattice point enumeration for polytopes concern themselves with counting the number of points in the intersection of a discrete lattice (for our purposes $\mathbb{Z}^d$) with a polytope P. We can ask what happens to this number if we dilate P by a positive integer factor t. An equivalent, and often useful, reformulation of the question asks what happens if we contract the integer lattice by the same factor t, leaving P fixed. In other words, what can we say about the function $L_P : \mathbb{Z}_{>0} \to \mathbb{Z}$ given by

$$L_P(t) := \#\left(\mathbb{Z}^d \cap tP\right) = \#\left(t^{-1}\mathbb{Z}^d \cap P\right)? \tag{2.1}$$

Is there a compact form for expressing L_P, and if so, how can we compute it? In this context it is useful to extend (2.1) to any bounded subset X of $\mathbb{R}^d$; we define $L_X : \mathbb{Z}_{>0} \to \mathbb{Z}$ by

$$L_X(t) := \#\left(\mathbb{Z}^d \cap tX\right) = \#\left(t^{-1}\mathbb{Z}^d \cap X\right).$$

The development of theories concerning the behavior of L_P and L_{P° for irrational polytopes remains for the most part an open problem. For rational polytopes, the question of whether L_P and L_{P° have closed-form expressions finds its answer in a remarkable theorem originally due to Eugene Ehrhart. Here we say that a function

f is a *quasipolynomial* if there exists $n \in \mathbb{Z}_{>0}$ and polynomials $f_0, \dots, f_{n-1}$ such that

$$f(t) = f_i(t) \ \text{ if } t \equiv i \pmod{n}.$$

The *degree* of f is the maximum degree of the polynomials $f_0, \dots, f_{n-1}$. In the case where n is minimal, we call n the *period* of f.

Theorem 1 (Ehrhart's Theorem for Rational Polytopes) *If P is a rational d-polytope, then $L_P(t)$ is a degree d quasipolynomial in t. Its period divides the denominator of P.*

Proof. Beck and Robins develop a thorough proof of this theorem in [5, Chapter 3]. Here we emphasize the key points in the buildup that prove useful in explaining the computational methods used later on. For this we define the *Ehrhart series* of P,

$$\mathrm{Ehr}_P(z) := 1 + \sum_{t \geq 1} L_P(t) z^t. \tag{2.2}$$

The following result comes verbatim from [5, Lemma 3.24]:

Lemma 2 *If*

$$\sum_{t \geq 0} f(t) z^t = \frac{g(z)}{h(z)},$$

then f is a quasipolynomial of degree d with period dividing p if and only if g and h are polynomials such that $\deg(g) < \deg(h)$, all roots of h are p^{th} roots of unity of *multiplicity at most* $d+1$*, and there is a root of multiplicity equal to* $d+1$ *(all of this assuming that* g/h *has been reduced to lowest terms.)*

The remaining proof of Theorem 1 is devoted to showing the existence of a polynomial g such that

$$\operatorname{Ehr}_P(z) = \frac{g(z)}{(1-z^p)^{d+1}}. \tag{2.3}$$

and $\deg(g) < p(d+1)$. Theorem 1 then follows from Lemma 2. ■

This represents a generalization of Ehrhart's earlier theorem for integral polytopes. In this case the denominator of P is 1 and $L_P(t)$ is a polynomial in t of degree d. We will also use the following result:

Proposition 3 *If* $P \subseteq \mathbb{R}^d$ *is a rational* d*-polytope and* $f_0, \ldots, f_{p-1}$ *are the polynomials defining* $L_P(t)$*, then* $L_P(0) = f_0(0) = 1$ *and every* f_i *is of degree* d *with leading term equal to the volume of* P*.*

Proof. Exercises 3.27 and 3.29 of [5]. ■

We call the rational function given in equation (2.3) the *Ehrhart rational generating function* of P. Since $\operatorname{Ehr}_P(z)$ is just the Taylor series expansion at 0 of the Ehrhart generating function, we can compute the first $p(d+1)$ terms of $\operatorname{Ehr}_P(z)$ and use polynomial interpolation to recover the polynomial coefficients of $L_P(t)$. For polytopes with large denominators this recovery can become an intractable task.

Hence the rational generating function $\mathrm{Ehr}_P(z)$ often represents a computationally more efficient form for encoding $L_P(t)$.

The counting functions L_P and L_{P° have a deep relationship to one another, as revealed by the next theorem. Together Theorems 1 and 4 imply that $L_{P^\circ}(t)$ is also a quasipolynomial function in t satisfying the conditions of Theorem 1.

Theorem 4 (Ehrhart Macdonald reciprocity) *If P is a rational d-polytope, then as quasi-polynomials*

$$L_P(-t) = (-1)^d L_{P^\circ}(t).$$

Proof. Establishing a proof of this relationship occupies the better part of Chapter 4 from [5]. Most relevant to our work here is the *Ehrhart series for the interior of a polytope,* defined by

$$\mathrm{Ehr}_{P^\circ}(z) := \sum_{t\geq 1} L_{P^\circ}(t)\, z^t,$$

as well as the following theorem:

Theorem 5 *If P is a rational d-polytope, then evaluating the Ehrhart rational generating function of P at $\frac{1}{z}$ yields*

$$Ehr_P\left(\frac{1}{z}\right) = (-1)^{d+1} Ehr_{P^\circ}(z).$$

■

Proof of Theorem 5. [5, Theorem 4.4]. ■

2.6 Barvinok's Algorithm

In many instances we can determine the counting functions L_P and L_{P° by insight into the structural properties of the particular polytope. For example, if C is the d-dimensional unit cube, determining that $L_C(t) = (t+1)^d$ is a simple exercise. (See [5, Chapter 2] for a number of less trivial examples.) However, for many polytopes the only practical way to derive L_P or L_{P° is via the computer.

In 1994 [2] Alexander Barvinok introduced an algorithm to compute Ehr_P in polynomial time, provided the dimension of the polytope is considered fixed. For polytope $P \subseteq \mathbb{R}^d$ this algorithm determines a multivariate rational generating function of the form

$$F(\mathbf{z}) = \sum_{m \in \mathbb{Z}^{d+1} \cap \operatorname{cone}(P)} \mathbf{z}^{\mathbf{m}}$$

(where $\mathbf{z}^{\mathbf{m}} := z_1^{m_1} \cdots z_{d+1}^{m_{d+1}}$), so that $F\left((1, \ldots, 1, z)^\intercal\right)$ evaluates to the rational generating function $\mathrm{Ehr}_P(z)$. Combined with the previous results of Ehrhart theory, this form lends itself well to computing a sum of counting functions taken from a large number of polytopes. The implementations of our algorithms rely heavily on the instance of Barvinok's Algorithm found in the c-library Barvinok [21], which among other things computes the Ehrhart rational generating function $\mathrm{Ehr}_P(z)$.

2.7 Hyperplane Arrangements and Inside-Out Polytopes

A *hyperplane arrangement* (or *arrangement*) is any finite collection of hyperplanes contained in an ambient real vector space. Normally, without qualification we take the ambient space to be $\mathbb{R}^d$; however it is convenient to consider arrangements in linear subspaces of $\mathbb{R}^d$. If H is a hyperplane and U an linear subspace of $\mathbb{R}^d$, the set $H \cap U$ is again a hyperplane in U. Extending this notion, a hyperplane arrangement in $\mathbb{R}^d$ induces an arrangement in any linear subspace of $\mathbb{R}^d$. If $U \subseteq \mathbb{R}^d$ is a subspace, we call the set $\{H \cap U : H \in \mathcal{H}\}$ *the arrangement induced by* $\mathcal{H}$ *in* U.

A hyperplane arrangement $\mathcal{H}$ partitions $\mathbb{R}^d$ into a set of regions, which are the connected components of $\mathbb{R}^d \setminus \bigcup \mathcal{H}$. A hyperplane arrangement induces a set of regions in any convex set. If $C \subseteq \mathbb{R}^d$ is convex we say a subset of $\mathbb{R}^d$ is an *open region* of $(C, \mathcal{H})$ if it is a connected component of $C^\circ \setminus \bigcup \mathcal{H}$. A *closed region* of $(C, \mathcal{H})$ is the topological closure of an open region. We refer to the *regions of* $(C, \mathcal{H})$ (or *regions*$(C, \mathcal{H})$) as the set of closed regions. If P is a d-polyhedron, then every region of $(P, \mathcal{H})$ is again a d-polyhedron, and likewise with polytopes. If P is a rational polytope and $\mathcal{H}$ a set of rational hyperplanes, then every region of $(P, \mathcal{H})$ is again a rational polytope.

An *inside-out polytope* of *dimension* d is a d-polytope P together with a hyperplane arrangement $\mathcal{H}$. The vertices of $(P, \mathcal{H})$ are the vertices taken over all of the regions of $(P, \mathcal{H})$. We call $(P, \mathcal{H})$ *rational* in the case where P is a rational polytope and $\mathcal{H}$ is a set of rational hyperplanes.

If C is a convex set, the *multiplicity of* $\mathbf{x} \in \mathbb{R}^d$ *with respect to* $(C, \mathcal{H})$ (denoted $m_{C,\mathcal{H}}(\mathbf{x})$) is the number of closed regions of $(C, \mathcal{H})$ containing $\mathbf{x}$. For any semi-open polytope S we define the *open and closed Ehrhart quasipolynomials* as

$$L^{\circ}_{S,\mathcal{H}}(t) := \#\left(t^{-1}\mathbb{Z}^d \cap (S \backslash \mathcal{H})\right) \text{ and } L_{S,\mathcal{H}}(t) := \sum_{\mathbf{x} \in t^{-1}\mathbb{Z}^d} m_{S,\mathcal{H}}(\mathbf{x}),$$

respectively. In the case where P is a polytope and $R_1, \ldots, R_k$ are the closed regions of $(P, \mathcal{H})$ we have

$$L^{\circ}_{P^\circ,\mathcal{H}}(t) = \sum L_{R_i^\circ}(t) \text{ and } L_{P,\mathcal{H}}(t) = \sum L_{R_i}(t). \tag{2.4}$$

As a consequence of Theorems 1 and 4 above, $L^{\circ}_{P^\circ,\mathcal{H}}$ and $L_{P,\mathcal{H}}$ are quasipolynomials in t satisfying

$$L_{P,\mathcal{H}}(-t) = (-1)^{\dim P} L^{\circ}_{P^\circ,\mathcal{H}}(t).$$

Chapter 3

Region Enumeration Method

Let P be a polytope and $\mathcal{H}$ a hyperplane arrangement. The alternative definition of $L_{P,\mathcal{H}}(t)$ given by (2.4) suggests an obvious algorithm for computing $L_{P,\mathcal{H}}(t)$; enumerate the regions of $(P,\mathcal{H})$ and compute the sum $\sum_{i=1}^{k} L_{R_i}(t)$ directly. At first glance one might doubt that this naive approach could represent a good computational solution. For an arrangement of n hyperplanes, we have 2^n possible candidates for the regions of $(P,\mathcal{H})$. (Every region of an inside-out polytope is the intersection of the polytope with one of two half-spaces taken from each hyperplane in the arrangement.) However, not all of these candidates represent regions. If the dimension of P is fixed, we first show that the number remains bounded by a polynomial in $\#\mathcal{H}$.

Using this, we show how to enumerate the regions of $(P, \mathcal{H})$ smartly to obtain a rational generating function for $L_{P,\mathcal{H}}$ in polynomial time whenever $\dim(P)$ is considered fixed.

In order to accomplish the first task, we need the following result:

Proposition 6 *Let $\mathcal{H}$ be a hyperplane arrangement, let H be a hyperplane, and let $\mathcal{J}$ be the arrangement induced by $\mathcal{H}$ in H. Then*

$$\#\mathit{regions}\left(\mathcal{H} \cup \{H\}\right) = \#\mathit{regions}\left(\mathcal{J}\right) + \#\mathit{regions}\left(\mathcal{H}\right).$$

Proof. By Proposition 11

$$\operatorname{regions}\left(\mathcal{H} \cup \{H\}\right) = \bigcup_{R \in \operatorname{regions}(\mathcal{H})} \operatorname{regions}\left((R, \{H\})\right).$$

For every open region R of $\mathcal{H}$, the set $R^\circ \cap H$ is an open region of $\mathcal{J}$ if and only if H is transverse to R°, i.e., if and only if H splits R° into 2 open regions of $\mathcal{H} \cup \{H\}$. The result now follows by a straight-forward counting argument. ■

With this result in hand we can now prove our first assertion, namely for any convex set C of fixed dimension d the number of regions of $(C, \mathcal{H})$ is bounded by a polynomial in $\#\mathcal{H}$.

Proposition 7 *Let $\mathcal{H}$ be an arrangement of n hyperplanes in an affine d-subspace U*

of $\mathbb{R}^m$. The number of regions of $\mathcal{H}$ is at most

$$\sum_{i=0}^{d}\binom{n}{i}.$$

Proof. Let $d = \dim(U)$. We use induction on n. The base case $n = 0$ is trivial. Suppose n is a non-negative integer, and let $H_0, \ldots, H_n$ be the hyperplanes of $\mathcal{H}$. Now $\{H_1, \ldots, H_n\}$ induces an arrangement in H_0 of k hyperplanes for some $k \leq n$. By inductive hypothesis and the above proposition

$$\#\text{regions}\left(\{H_0, \ldots, H_n\}\right) \leq \sum_{i=0}^{d}\binom{n}{i} + \sum_{i=0}^{d-1}\binom{k}{i} \leq \sum_{i=0}^{d}\binom{n}{i} + \sum_{i=0}^{d-1}\binom{n}{i}.$$

A straightforward application of the identity $\binom{n+1}{i} = \binom{n}{i} + \binom{n}{i-1}$ shows

$$\sum_{i=0}^{d}\binom{n}{i} + \sum_{i=0}^{d-1}\binom{n}{i} = \sum_{i=0}^{d}\binom{n+1}{i}.$$

■

Corollary 8 *Let C be a d-dimensional convex set and let $\mathcal{H}$ be an arrangement of n hyperplanes. Then the number of regions of $(C, \mathcal{H})$ is bounded by*

$$\sum_{i=0}^{d}\binom{n}{i}.$$

Proof. The number of regions of $(C, \mathcal{H})$ is bounded by the number of regions of the arrangement induced by $\mathcal{H}$ in $\text{aff}(C)$. ■

Let $R_1, \ldots, R_k$ again be the closed regions of $(P, \mathcal{H})$. If the dimension of P is fixed, the preceding corollary ensures the sum $\sum_{i=1}^{k} L_{R_i}(t)$ remains manageable, even for a large number hyperplanes. We'll use this result to our advantage in enumerating the regions of $(P, \mathcal{H})$.

The following observation will prove important for our algorithm:

Proposition 9 *If P is a d-polyhedron and $\mathcal{H} = \{H(\mathbf{a}_1, b_1), \ldots, H(\mathbf{a}_n, b_n)\}$ with no hyperplane containing P, then R is a region of $(P, \mathcal{H})$ if and only if $\dim R = d$ and*

$$R = P \cap \bigcap_{i=1}^{n} K(s_i \mathbf{a}_i, s_i b_i)$$

for some $s_1, \ldots, s_n \in \{-1, 1\}$.

Proof. For any $s_1, \ldots, s_n \in \{-1, 1\}$, the set $P \cap \bigcap_{i=1}^{n} K(s_i \mathbf{a}_i, s_i b_i)$ is the closure of $P^\circ \cap \bigcap_{i=1}^{n} \left\{\mathbf{x} \in \mathbb{R}^d : s_i \mathbf{a}_i \mathbf{x} \le s_i b_i\right\}$. Apply the definition of closed region using the observation that $\dim(Q) = \dim(Q^\circ)$ for any polyhedron Q. ■

In particular we make use of the special case where $\#\mathcal{H} = 1$ to compute the regions of $(P, \mathcal{H})$.

Corollary 10 *Let P be a d-polyhedron and $H(\mathbf{a}, b)$ be a hyperplane. Then*

$$\textit{regions}(P, H(\mathbf{a}, b)) = \left\{R \subseteq \mathbb{R}^d : R = K(\pm\mathbf{a}, \pm b) \cap P, \dim(R) = d\right\}. \qquad (3.1)$$

Proposition 11 *Let P be a d-polyhedron, and $\{H_i\}$ be a sequence of hyperplanes. If $\mathcal{R}_n :=$ regions$((P, \{H_1, \ldots, H_n\}))$, then*

$$\mathcal{R}_n = \bigcup_{R \in \mathcal{R}_{n-1}} \textit{regions}\left((R, \{H_n\})\right). \tag{3.2}$$

Proof. Apply Proposition 9. ■

In the equation (3.2) above we would like a method for determining the set regions$(R, \{H_n\})$. We accomplish this using the next result. Here a hyperplane H is *transverse* to a polyhedron P if $H \cap P^\circ \neq \emptyset$.

Proposition 12 *Let P be a polyhedron, let $\mathcal{H}$ be a hyperplane arrangement. If $H(\mathbf{a}, b)$ is a hyperplane and R is a region of $(P, \mathcal{H})$, the following statements are equivalent:*

1. *H is transverse to R.*

2. $\min\{\mathbf{ax} : \mathbf{x} \in R\} < b < \max\{\mathbf{ax} : \mathbf{x} \in R\}$

3. *$R \cap K(\mathbf{a}, b)$ and $R \cap K(-\mathbf{a}, -b)$ are both closed regions of $(P, \mathcal{H} \cup \{H\})$.*

4. $\dim(R \cap K(\mathbf{a}, b)) = \dim(R \cap K(-\mathbf{a}, -b))$

5. *#regions$(R, \{H\}) = 2$.*

6. *R is not a region of $(P, \mathcal{H} \cup \{H\})$* ■

The following algorithm describes a method for computing the regions of an inside-out polyhedron. At each iteration the new hyperplane effectively cuts all of the regions from the previous iteration that it intersects into two new regions.

Algorithm 13 *Slicing Hyperplane Algorithm*

Input: A polyhedron $P = P(A, \mathbf{b})$ $\left(A \in \mathbb{R}^{m \times d}\right)$ and an arrangement

$$\mathcal{H} = \{H(\mathbf{c}_1, d_1), \ldots, H(\mathbf{c}_n, d_n)\}.$$

Output: A set $\{(-S_1, \mathbf{t}_1), \ldots, (-S_k, \mathbf{t}_k)\}$ of $m + n \times d + 1$ matrices, such that

$$\{P(S_i, \mathbf{t}_i)\} = \text{regions}((P, \mathcal{H}))$$

and $P(S_i, \mathbf{t}_i) \neq P(S_j, \mathbf{t}_j)$ when $i \neq j$.

```
1. set $\mathcal{R} := \{P\}$
2. **for** $i := 1$ to $n$
3.     set $\mathcal{T} := \mathcal{R}$
4.     set $\mathcal{R} := \emptyset$
5.     **for each** $T$ in $\mathcal{T}$
6.         **if** $H(\mathbf{c}_i, d_i)$ is transverse to $T$ **then**
```

```
                set $\mathcal{R} := \mathcal{R} \cup \{T \cap K(\mathbf{c}_i, d_i)\} \cup \{T \cap K(-\mathbf{c}_i, -d_i)\}$
            else
                set $\mathcal{R} := \mathcal{R} \cup T$
return $\mathcal{R}$
```

Proof of algorithm. For $i = 1, \ldots, n$ let $H_i := K(\mathbf{c}_i, d_i)$. Let

$$\mathcal{R}_i := \text{regions}\left(P, \{H_1, \ldots, H_i\}\right).$$

At the beginning of i^{th} iteration of the outer "for loop" $\mathcal{T}$ corresponds the set $\mathcal{R}_{i-1}$. The "for each" loop makes use of Corollary 10 and Proposition 9 above to generate $\mathcal{R}_i$. ■

The time-complexity function of Algorithm 13 depends both on the number of iterations performed and on the time used at each iteration. The fact that this algorithm uses $\mathcal{H}$-descriptions to encode polyhedra means that computing the intersection $T \cap K(\pm\mathbf{c}, \pm d)$ amounts to the adjoining a row to a matrix. If $T = P(S, \mathbf{t})$, then

$$T \cap K(\pm\mathbf{c}, \pm d) = P\begin{pmatrix} S & \mathbf{t} \\ \pm\mathbf{c}^{\mathsf{T}}, & \mp d \end{pmatrix}.$$

Other than the looping structure itself, the only computationally expensive task this Algorithm 13 performs, is testing whether or not a hyperplane is transverse to a polyhedron. By Proposition 12 this amounts to solving two linear programming problems. Such problems are known to be solvable in polynomial time. (See, for example, [17, Chapters 13-15]).

The following proposition deals with Algorithm 13's looping structure.

Proposition 14 *Let P be a d-polyhedron and let $\mathcal{H} = \{H_1, \ldots, H_n\}$ be an arrangement of hyperplanes. The total number of transversality tests performed by Algorithm 13 is less than or equal to*

$$\sum_{j=1}^{d+1} \binom{n}{j}.$$

Proof. Let N be the number of tests performed. For $k = 0, \ldots, n$ let

$$\mathcal{R}_k := \text{regions}\left(P, \{H_1, \ldots, H_k\}\right).$$

Algorithm 13 computes $\mathcal{R}_k$ by performing a transversality test for each member of $\mathcal{R}_{k-1}$. Hence

$$N = \sum_{k=1}^{n} \#\mathcal{R}_{k-1}. \tag{3.3}$$

Corollary 3 implies

$$N \leq \sum_{k=1}^{n} \sum_{j=0}^{d} \binom{k-1}{j} = \sum_{j=0}^{d} \sum_{k=0}^{n-1} \binom{k}{j}.$$

A direct application of the identity $\binom{s+1}{r+1} = \sum_{k=0}^{s} \binom{k}{r}$ shows that

$$N \leq \sum_{j=0}^{d} \binom{n}{j+1} = \sum_{j=1}^{d+1} \binom{n}{j}.$$

■

This gives worst case upper bound on the number of expensive steps required by Algorithm 13 as a function of $\#\mathcal{H}$. Hyperplane arrangements often have considerable fewer regions than the upper bound given in Corollary 3. We might also ask how well this algorithm performs with respect to the number of regions of $(P, \mathcal{H})$. In other words, how much "extra work" does algorithm Algorithm 13 perform?

Proposition 15 *Let P be a d-polyhedron, let $\mathcal{H} = \{H_1, \dots, H_n\}$ be an arrangement of hyperplanes and $r := \#regions(P, \mathcal{H})$. The total number of transversality tests performed by Algorithm 13 is less than or equal to nr.*

Proof. Define N and $\mathcal{R}_k$ as above. Observe that for all $k = 0, \dots, n$ we have $\#\mathcal{R}_k \leq r$. Now apply (3.3) . ■

Proposition 16 *If P is a polyhedron of fixed dimension and $\mathcal{H}$ a hyperplane arrangement, Algorithm 13 computes regions$(P, \mathcal{H})$in polynomial time.*

Proof. Fix $d := \dim(P)$ and let l be the bit length of a binary encoding of $(P, \mathcal{H})$ as a $m + n \times r$ matrix M, where $n = \#\mathcal{H}$ and m represents the number of half-spaces defining P. Each polyhedron Q considered by the Sweep Hyperplane Algorithm (SHA) may be given as the intersection of $m + n$ inequalities which are either encoded as a row of zeros ($\mathbf{0}^\top \mathbf{x} \leq 0$), or which differ from the rows of M by at most a sign. Hence the encoding length l_Q of Q, for of every polyhedron considered by SHA, is $\mathcal{O}(l)$. This implies we can compute the dimension of every polyhedron considered by SHA in $\mathcal{O}(p(l))$ time for some polynomial p [17, Theorem 13.4], so that by Proposition 14 the runtime of SHA is

$$\mathcal{O}\left(p(l)\sum_{j=1}^{d+1}\binom{n}{j}\right).$$

The encoding size of every polyhedron output by SHA is likewise $\mathcal{O}(l)$, so by Corollary 8 the size of SHA's output is

$$\mathcal{O}\left(l\sum_{i=0}^{d}\binom{n}{i}\right).$$

Obviously $n \leq m+n \leq l$ implying output size is $\mathcal{O}\left(l\sum_{i=0}^{d}\binom{l}{i}\right)$ and runtime is $\mathcal{O}\left(p(l)\sum_{j=1}^{d+1}\binom{l}{j}\right)$. Hence Algorithm 13 runs in polynomial time for polytopes of fixed dimension. ■

With one exception we now have everything we need to compute a generating function for $L_{P,\mathcal{H}}(t)$. Additionally we require the definitions of the *open* and *closed rational generating functions* of an inside-out polytope,

$$\mathrm{Ehr}^{\circ}_{P^{\circ},\mathcal{H}}(z) := \sum_{t\geq 1} L^{\circ}_{P^{\circ},\mathcal{H}}(t)\, z^t \text{ and } \mathrm{Ehr}_{P,\mathcal{H}}(z) := \sum_{t\geq 0} L_{P,\mathcal{H}}(t)\, z^t,$$

respectively. If $R_1,\ldots,R_k$ again represent the closed regions of $(P,\mathcal{H})$, by straight forwardly applying Ehrhart theory to the definition of $L_{P,\mathcal{H}}$ and $L^{\circ}_{P^{\circ},\mathcal{H}}$ we see that

$\mathrm{Ehr}^{\circ}_{P^{\circ},\mathcal{H}}$ and $\mathrm{Ehr}_{P,\mathcal{H}}$ have representations as rational functions, which satisfy the reciprocity relation,

$$\begin{aligned} \mathrm{Ehr}_{P,\mathcal{H}}\left(\frac{1}{z}\right) &= \sum_{i=1}^{k} \mathrm{Ehr}_{R_i}\left(\frac{1}{z}\right) = \sum_{i=1}^{k} (-1)^{\dim R_i + 1} \mathrm{Ehr}_{R_i^{\circ}}(z) \\ &= (-1)^{\dim P} \mathrm{Ehr}^{\circ}_{P^{\circ},\mathcal{H}}(z)\,. \end{aligned}$$

This allows us to use the rational functions produced by Barvinok's Algorithm as efficient data structures for encoding the sum $L_{P,\mathcal{H}}(t)$.

Algorithm 17 *Computing* $\mathit{Ehr}_{P,\mathcal{H}}(t)$ *by direct enumeration of regions*

Input: An H-polytope P and a hyperplane arrangement $\mathcal{H}$.

Output: The rational function $\mathrm{Ehr}_{P,\mathcal{H}}(t)$.

```
1. set $\mathcal{R} :=$ regions($(P, \mathcal{H})$) (using Hyperplane Sweep Algorithm)
2. set $F := 0$
3. for each $R$ in $\mathcal{R}$
4.     set $F := F+$ $\mathrm{Ehr}_R(z)$ (using Barvinok's Algorithm)
5. return $F$
```

Theorem 18 *For polytopes of fixed dimension, Algorithm 17 computes* $\mathit{Ehr}_{P,\mathcal{H}}(z)$ *in polynomial time.*

Proof. Let l be the bit length of an encoding of $(P, \mathcal{H})$ as an $m + n \times r$ matrix M. Let $\mathcal{R}$ denote the closed regions of $(P, \mathcal{H})$. By Proposition 16 it suffices to show that the output size of Algorithm 17 is bounded by a polynomial in l and that the time T required to compute $\sum_{R \in \mathcal{R}} \mathrm{Ehr}_R(z)$, using descriptions generated by the Sweep Hyperplane Algorithm, is $\mathcal{O}(p(l))$ for some polynomial l.

Choose a polynomial $b(x)$ such that Barvinok's algorithm computes $\mathrm{Ehr}_P(z)$ in $\mathcal{O}(b(x))$ time for every d-polytope P with a binary H-description encoding of length x. [2] Fix $R \in \mathcal{R}$ and let l_R be the bit length of the description of R generated by the Sweep Hyperplane Algorithm. We obtained such a description by the negation of at most n rows of M, so it follows that l_R is $\mathcal{O}(l)$. By Corollary 8 the sum $\sum_{R \in \mathcal{R}} \mathrm{Ehr}_R(z)$ contains at most $\sum_{i=0}^{d} \binom{l}{i}$ rational functions, each bounded in length by the time required to compute $\mathrm{Ehr}_R(z)$. In particular both the output size S and runtime T of Algorithm 17 is $\mathcal{O}\left(b(l) \sum_{i=0}^{d} \binom{n}{i}\right)$. Obviously $n \leq l$, which implies both S and T are $\mathcal{O}\left(b(l) \sum_{i=0}^{d} \binom{l}{i}\right)$. Hence for constant d, Algorithm 17 runs in polynomial time. ■

The examples computed later in this work as well as in [7], [8], and [6], all rely on the fact that we can obtain an answer to a given problem from the function $L^{\circ}_{P^{\circ}, \mathcal{H}}$. Our computational solutions to these problems use the fact that the we can model the exclusion of forbidden values from a solution set using the *interior* of some polytope minus a union of hyperplanes. In principle one would expect that some problems with forbidden constraints could be modeled similarly using closed polytopes or even semi-open polytopes, but without an apparent solution using the relative interior.

In the language of inside-out polytopes we would like to know how to compute $L^{\circ}_{S,\mathcal{H}}$ when S is any rational semi-open polytope and $\mathcal{H}$ a rational arrangement. Using the fact that every polytope can be expressed as the disjoint union of its relatively open faces, one approach is to sum the functions $L^{\circ}_{F^{\circ},\mathcal{H}}$ over the relatively open faces F of $\overline{S}$ contained in S.

Theorem 19 *Let* $S = \{\mathbf{x} : A'\mathbf{x} \leq \mathbf{b}', A''\mathbf{x} < \mathbf{b}''\}$ *be a semi-open polytope of fixed dimension and let* $\mathcal{H}$ *be a hyperplane arrangement. If*

$$Ehr^{\circ}_{(S,\mathcal{H})}(z) := \sum_{t \geq 0} L^{\circ}_{S,\mathcal{H}}(t)\, z^{t},$$

then there is a polynomial time algorithm for computing $Ehr^{\circ}_{(S,\mathcal{H})}(z)$.

Proof. Determining whether S is empty can be accomplished in polynomial time using linear programming, so we may assume that

$$\overline{S} = \{\mathbf{x} : A'\mathbf{x} \leq \mathbf{b}', A''\mathbf{x} \leq \mathbf{b}''\}.$$

For a closed polyhedron of fixed dimension, descriptions of the faces of the polyhedron are well known to be computable in polynomial time. Deciding whether an open face of $\overline{S}$ is contained in S amounts to finding a relative interior point of the face. As this can be accomplished using linear programming, we can determine which open faces of $\overline{S}$ are contained in S in polynomial time by a solving a set of linear programming problems, the size of which is equal to the number of faces of $\overline{S}$. Observing that

$$S \backslash \cup \mathcal{H} = \bigcup_{\substack{F \in \text{faces}\left(\overline{S}\right) \\ F^{\circ} \subseteq S}} \left(F^{\circ} \backslash \cup \mathcal{H}\right),$$

we apply Algorithm 17 to compute the sum

$$\text{Ehr}^{\circ}_{(S,\mathcal{H})}\left(z\right) = \sum_{\substack{F \in \text{faces}\left(\overline{S}\right) \\ F^{\circ} \subseteq S}} \text{Ehr}^{\circ}_{(F^{\circ},\mathcal{H})}\left(z\right).$$

■

Chapter 4

Mixed Integer Programming

Another approach to computing $L^{\circ}_{S,\mathcal{H}}$, when S is any rational semi-open polytope and $\mathcal{H}$ a rational arrangement, involves finding a sequence of polytopes P_t such that

$$\#\mathbb{Z}^d \cap P_t = L^{\circ}_{S,\mathcal{H}}(t)\,.$$

This approach finds motivation in a technique taken from mixed integer programming used to reduce a problems involving polyhedral unions to problems involving a single polyhedron. As the construction is somewhat simpler, we first show how to apply the technique to find $L_{P,\mathcal{H}}(t)$.

In some linear programming problems one wishes to model a disjunctive statement like

$$\mathbf{a}^{\intercal}\mathbf{x} \leq b \text{ or } \mathbf{c}^{\intercal}\mathbf{x} \leq d \tag{4.1}$$

with a statement involving the use of "and" instead of "or". If $\mathbf{a}^{\intercal}\mathbf{x}$ and $\mathbf{c}^{\intercal}\mathbf{x}$ are bounded above (say by M) for all $\mathbf{x}$ under consideration, one can accomplish this by introducing a dummy variable y and using the equivalent statement,

$$\mathbf{a}^{\intercal}\mathbf{x} \leq (M-b)\,y + b \text{ and } \mathbf{c}^{\intercal}\mathbf{x} \leq (d-M)\,y + M \text{ and } y \in \{0,1\}\,. \tag{4.2}$$

Depending on whether y is 1 or 0, one inequality becomes a redundant statement about the upper bound M, while the other becomes an inequality from (4.1). This technique allows one to apply many algorithms that only work for intersections of polyhedra, to unions of polyhedra, which may be neither convex nor connected.

Because of our interest in counting lattice points, this technique is well suited to inside out polytopes. For a polytope $P = P\,(A, \mathbf{b})$ and hyperplane arrangement

$$\mathcal{H} = \{H\,(\mathbf{c}_1, d_1)\,, \ldots, H\,(\mathbf{c}_n, d_n)\}\,,$$

we have

$$P^{\circ} \backslash \bigcup \mathcal{H} = P^{\circ} \cap \bigcap_{i=1}^{n} \left(\left\{x \in \mathbb{R}^d : \mathbf{ax} < b\right\} \cup \left\{x \in \mathbb{R}^d : -\mathbf{ax} < -b\right\}\right),$$

so that the problem of computing $\#\left(Z^d \cap P^{\circ} \backslash \bigcup \mathcal{H}\right)$ can be restated in terms of finding the number of points

$$\mathbf{x} \in \mathbb{Z}^d \cap P^\circ : \mathbf{c}_i^\intercal \mathbf{x} < d_i \text{ or } -\mathbf{c}_i^\intercal \mathbf{x} < -d_i, \text{ for all } i = 1, \ldots, n. \tag{4.3}$$

Since P is by definition bounded, using linear programming we can always compute $\lambda = \max\left\{\left|\mathbf{c}_i^\intercal \mathbf{x}\right| : \mathbf{x} \in P, i = 1, \ldots, n\right\}$ (see, for example, [17]). For each $i = 1, \ldots, n$ we introduce variables y_i so that the following statement is equivalent to (4.3):

$$\begin{gathered} \mathbf{x} \in \mathbb{Z}^d \cap P^\circ \text{ and } \exists \mathbf{y} \in \{0,1\}^d : \forall i = 1, \ldots n, \\ \mathbf{c}_i^\intercal \mathbf{x} + (d_i - \lambda)\, y_i < d_i \text{ and } -\mathbf{c}_i^\intercal \mathbf{x} + (d_i + \lambda)\, y_i < \lambda. \end{gathered} \tag{4.4}$$

By [22, Lemma 2.8] a polytope is full dimensional if and only if a description of its interior can be given by a system of strict linear inequalities. This motivates restating $y_i \in \mathbb{Z} \cap \{0,1\}$ as, $-1 < y_i < 2$ and $y_i \in \mathbb{Z}$, so that we may encode (4.4) in matrix form as

$$\begin{pmatrix} A & 0 \\ C & (D - \lambda I_n) \\ -C & (D + \lambda I_n) \\ 0 & -I_n \\ 0 & I_n \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} < \begin{pmatrix} \mathbf{b} \\ \mathbf{d} \\ \lambda \mathbf{1} \\ \mathbf{1} \\ 2 \cdot \mathbf{1} \end{pmatrix} \text{ and } \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \in \mathbb{Z}^{d+n}.$$

where $C = \begin{pmatrix} \mathbf{c}_1 & \cdots & \mathbf{c}_n \end{pmatrix}^\intercal$ and $D = \begin{pmatrix} d_1 \mathbf{e}_1 & \cdots & d_n \mathbf{e}_n \end{pmatrix}$. If P is full dimensional we can count $\#\left(Z^d \cap P \backslash \bigcup \mathcal{H}\right)$ by instead finding $\#\left(Z^{d+n} \cap P\left(S, \mathbf{t}\right)^\circ\right)$ where S is the matrix, and $\mathbf{t}$ is the vector, given in (4.7).

Proposition 20 *Let $P = P(A, \mathbf{b}) \subseteq \mathbb{R}^d$ be a d-polytope and*

$$\mathcal{H} = \{H(\mathbf{c}_1, d_1), \ldots, H(\mathbf{c}_n, d_n)\}$$

be an arrangement of non-degenerate hyperplanes. If and S and $\mathbf{t}$ are of the form given by (4.7), *then* $\#\left(\mathbb{Z}^d \cap P^\circ \backslash \mathcal{H}\right) = \#\left(\mathbb{Z}^d \cap P(S, \mathbf{t})^\circ\right)$.

Proof. Let

$$Q := P(S, \mathbf{t}) = \left\{ \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} : \begin{pmatrix} A & 0 \\ C & (D - \lambda I_n) \\ -C & (D + \lambda I_n) \\ 0 & -I_n \\ 0 & I_n \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \leq \begin{pmatrix} \mathbf{b} \\ \mathbf{d} \\ \lambda \mathbf{1} \\ \mathbf{1} \\ 2 \cdot \mathbf{1} \end{pmatrix} \right\}$$

We'll show that $\varphi : \mathbb{Z}^d \cap P^\circ \backslash \mathcal{H} \to \mathbb{Z}^d \cap Q^\circ$, $\binom{\mathbf{x}}{\mathbf{y}} \longmapsto \mathbf{x}$ is a bijection. By Proposition

21 Q is full dimensional, so the following statements are equivalent

$$\begin{aligned}
\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} &\in \mathbb{Z}^{d+n}\cap Q^{\circ}\\
&\Leftrightarrow \begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix}\in\mathbb{Z}^{d+n}, \begin{pmatrix} A & 0\\ C & (D-\lambda I_n)\\ -C & (D+\lambda I_n)\\ 0 & -I_n\\ 0 & I_n\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} < \begin{pmatrix}\mathbf{b}\\ \mathbf{d}\\ \lambda\mathbf{1}\\ \mathbf{1}\\ 2\cdot\mathbf{1}\end{pmatrix}\\
&\Leftrightarrow \begin{matrix}\mathbf{x}\in\mathbb{Z}^d, \mathbf{y}\in\mathbb{Z}^n, A\mathbf{x}<\mathbf{b}, \mathbf{0}\le\mathbf{y}\le\mathbf{1},\\ C\mathbf{x}+(D-\lambda I_n)\,\mathbf{y}<\mathbf{d}, -C\mathbf{x}+(D+\lambda I_n)\,\mathbf{y}<\lambda\mathbf{1}\end{matrix}\\
&\Leftrightarrow \begin{matrix}\mathbf{x}\in\mathbb{Z}^d\cap P^{\circ}, \mathbf{y}\in\{0,1\}^n, \forall i=1,\dots,n:\\ \mathbf{c}_i\mathbf{x}+(d_i-\lambda)\,y_i<d_i, -\mathbf{c}_i\mathbf{x}+(d_i+\lambda)\,y_i<\lambda\end{matrix} \qquad (4.5)
\end{aligned}$$

The following are also equivalent

$$\begin{aligned}
\mathbf{x} &\in \mathbb{Z}^d\cap P^{\circ}\backslash\mathcal{H}\\
&\Leftrightarrow \mathbf{x}\in\mathbb{Z}^d\cap P^{\circ}, \forall i=1,\dots,n: \mathbf{c}_i\mathbf{x}<d_i \text{ or } -\mathbf{c}_i\mathbf{x}<-d_i\\
&\Leftrightarrow \exists\mathbf{y}\in\{0,1\}^n: \begin{matrix}\mathbf{x}\in\mathbb{Z}^d\cap P^{\circ}, \forall i=1,\dots,n:\\ \mathbf{c}_i\mathbf{x}+(d_i-\lambda)\,y_i<d_i, -\mathbf{c}_i\mathbf{x}+(d_i+\lambda)\,y_i<\lambda\end{matrix} \qquad (4.6)
\end{aligned}$$

By connecting (4.5) with (4.6) one sees that φ is well-defined and surjective. Now let $\binom{\mathbf{u}}{\mathbf{v}}, \binom{\mathbf{x}}{\mathbf{y}} \in \mathbb{Z}^d\cap Q^{\circ}$, and assume $\mathbf{u}=\mathbf{x}$. If $\mathbf{v}\neq\mathbf{y}$, without loss of generality by $(*)$

we may assume that $v_i = 0$ and $y_i = 1$ for some i. By (4.5) we have $\mathbf{c}_i\mathbf{u} < d_i$. Since $\binom{\mathbf{u}}{\mathbf{y}} \in \mathbb{Z}^d \cap Q^\circ$, by (4.5) we also have

$$-\mathbf{c}_i\mathbf{x} + d_i + \lambda = -\mathbf{c}_i\mathbf{x} + (d_i + \lambda)\, y_i < \lambda$$

implying that $\mathbf{c}_i\mathbf{u} > d_i$, a contradiction. Hence $\mathbf{v} = \mathbf{y}$ and φ is injective. ■

Proposition 21 *Let $P = P(A, \mathbf{b}) \subseteq \mathbb{R}^d$ be a d-polytope and*

$$\mathcal{H} = \{H(\mathbf{c}_1, d_1), \ldots, H(\mathbf{c}_n, d_n)\}$$

be an arrangement of non-degenerate hyperplanes. Let $C := \begin{pmatrix} \mathbf{c}_1 & \cdots & \mathbf{c}_n \end{pmatrix}^\intercal$ and $D := \begin{pmatrix} d_1\mathbf{e}_1 & \cdots & d_n\mathbf{e}_n \end{pmatrix}$. If $\lambda \geq \max\{|\mathbf{c}_i^\intercal\mathbf{x}| : \mathbf{x} \in P, i = 1, \ldots, n\}$,

$$S = \begin{pmatrix} A & 0 \\ C & (D - \lambda I_n) \\ -C & (D + \lambda I_n) \\ 0 & -I_n \\ 0 & I_n \end{pmatrix}, \text{ and } \mathbf{t} = \begin{pmatrix} \mathbf{b} \\ \mathbf{d} \\ \lambda\mathbf{1} \\ \mathbf{1} \\ 2\cdot\mathbf{1} \end{pmatrix}, \tag{4.7}$$

then $\dim(P(S, \mathbf{t})) = d + n$.

Proof. Choose $\mathbf{x} \in P^\circ \backslash \bigcup \mathcal{H}$ and note that $P^\circ = \left\{x \in \mathbb{R}^d : Ax < \mathbf{b}\right\}$. For $i = 1, \ldots, n$ choose

$$y_i = \begin{cases} 0, \text{ if } \mathbf{c}_i^\intercal\mathbf{x} < d_i \\ 1, \text{ if } \mathbf{c}_i^\intercal\mathbf{x} > d_i \end{cases}.$$

We note that $\binom{\mathbf{x}}{\mathbf{y}}$ satisfies (4.4), which implies $\binom{\mathbf{x}}{\mathbf{y}} \in \left\{x \in \mathbb{R}^d : Sx < \mathbf{t}\right\}$. In particular $P\left(S, \mathbf{t}\right)$ is full dimensional. ■

For each $t \in \mathbb{Z}_{>0}$ we would like to have a method for computing $L^{\circ}_{P^{\circ},\mathcal{H}}\left(t\right)$. The proofs of Propositions 22 and 27 are aimed at providing such a method. (They are otherwise trivial as existence statements, since one can always construct an open (or closed) line segment containing any number of lattice points.)

Proposition 22 *If P is a rational polytope and $\mathcal{H}$ is a rational hyperplane arrangement, then there is a sequence of rational polytopes $\{Q_t\}$ such that,*

$$L^{\circ}_{P^{\circ},\mathcal{H}}\left(t\right) = L_{Q^{\circ}_t}\left(1\right)$$

for all $t \in \mathbb{Z}^+$.

Proof. Let $P = P\left(A, \mathbf{b}\right) \subseteq \mathbb{R}^d$ be a polytope. We may assume that

$$\mathcal{H} := \left\{H\left(\mathbf{c}_1, d_1\right), \ldots, H\left(\mathbf{c}_n, d_n\right)\right\}$$

contains no degenerate hyperplanes. We first prove the case where P is full dimensional. Let $C = \begin{pmatrix}\mathbf{c}_1 & \cdots & \mathbf{c}_n\end{pmatrix}^{\intercal}$ and $D = \begin{pmatrix}d_1\mathbf{e}_1 & \cdots & d_n\mathbf{e}_n\end{pmatrix}$. Choose $\lambda \in \mathbb{Z}^+$ such that

$$\lambda \geq \max\left\{\left|\mathbf{c}_i^{\intercal}\mathbf{x}\right| : \mathbf{x} \in P, i = 1, \ldots, n\right\}$$

and define the sequence $\{Q_t\}$ by $Q_t = P\left(A_t, \mathbf{b}_t\right)$ where

$$A_t := \begin{pmatrix} A & 0 \\ C & t\left(D - \lambda I_n\right) \\ -C & t\left(D + \lambda I_n\right) \\ 0 & -I_n \\ 0 & I_n \end{pmatrix} \text{ and } \mathbf{b}_t := \begin{pmatrix} t\mathbf{b} \\ t\mathbf{d} \\ t\lambda\mathbf{1} \\ \mathbf{1} \\ 2\cdot\mathbf{1} \end{pmatrix}. \tag{4.8}$$

A straight forward argument shows that

$$L^{\circ}_{P^{\circ},\mathcal{H}}\left(t\right) = \#\left(\mathbb{Z}^d \cap t\left[P^{\circ}\backslash \bigcup \mathcal{H}\right]\right) = L^{\circ}_{tP^{\circ},t\mathcal{H}}\left(1\right),$$

where $t\mathcal{H} := \{tH : H \in \mathcal{H}\}$. Observe that for any $\mathbf{a} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ we have $tH\left(\mathbf{a}, b\right) = H\left(\mathbf{a}, tb\right)$ and $tK\left(\mathbf{a}, b\right) = K\left(\mathbf{a}, tb\right)$. This observation implies $t\lambda \geq \max\left\{|\mathbf{c}_i^{\intercal}\mathbf{x}| : \mathbf{x} \in tP, i = 1, \ldots, n\right\}$. Together with Proposition 20 it also implies that

$$L^{\circ}_{tP^{\circ},t\mathcal{H}}\left(1\right) = \#\left(\mathbb{Z}^d \cap tP^{\circ}\backslash t\mathcal{H}\right) = \#\left(\mathbb{Z}^d \cap Q_t^{\circ}\right) = L_{Q_t^{\circ}}\left(1\right).$$

Now suppose $m := \dim(P) < d$. In the case where $\mathbb{Z}^d\cap$ aff$(P) \neq \emptyset$, using the invariance of lattice point count under integer-valued translation, we may assume that aff(P) is a linear subspace. Choose a lattice basis $\mathbf{v}_1, \ldots, \mathbf{v}_m$ of $\mathbb{Z}^d\cap$ aff(P) and define the linear isomorphism ψ : aff$(P) \to \mathbb{R}^m$ by $\psi : \mathbf{v}_i \mapsto \mathbf{e}_i$ for $i = 1, \ldots, m$ where $\{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$ is the standard basis of $\mathbb{R}^m$. A straightforward argument shows that $L^{\circ}_{P^{\circ},\mathcal{H}}(t) = L^{\circ}_{\psi(P)^{\circ},\psi(\mathcal{H})}(t)$. The desired outcome follows by applying the full dimensional case to $\psi(P)$ and $\psi(\mathcal{H})$.

Now if $\mathbb{Z}^d \cap \operatorname{aff}(P) = \emptyset$, choose k to be the least positive integer such that $\operatorname{aff}(kP) \cap \mathbb{Z}^d \neq \emptyset$. It follows that

$$L^{\circ}_{P^{\circ},\mathcal{H}}(t) = \begin{cases} L^{\circ}_{kP^{\circ},k\mathcal{H}}\left(\frac{t}{k}\right) & \text{if } t \equiv 0 \,(\operatorname{mod} k)\,, \\ 0 & \text{otherwise.} \end{cases}$$

By the previous argument there is a sequence $\{R_t\}$ such that $L^{\circ}_{kP^{\circ},k\mathcal{H}}(t) = L_{R_t^{\circ}}(1)$. Define the sequence $\{Q_t\}$ by

$$Q_t = \begin{cases} R_{\frac{t}{k}} & \text{if } t \equiv 0 \,(\operatorname{mod} k)\,, \\ \emptyset & \text{otherwise.} \end{cases}$$

It now follows that $L^{\circ}_{P^{\circ},\mathcal{H}}(t) = \#\left(\mathbb{Z}^d \cap Q_t^{\circ}\right)$. ■

If P is a rational polytope and $\mathcal{H}$ a rational arrangement the fact that $L^{\circ}_{P^{\circ},\mathcal{H}}$ has quasi-polynomial expression means that $L^{\circ}_{P^{\circ},\mathcal{H}}$ can be obtained using polynomial interpolation on the lattice point counts of the relatively open polytopes Q_t in Proposition 22.

Algorithm 23 *Mixed integer computation of* $L^{\circ}_{P^{\circ},\mathcal{H}}$.

Input. An H-description of a rational polytope $P = P(A,\mathbf{b})$ and a rational hyperplane arrangement $\mathcal{H} = \{H(\mathbf{c}_1, d_1), \ldots, H(\mathbf{c}_n, d_n)\}$. ■

Output.. A list of polynomials $f_0, \ldots, f_{p-1}$ so that $L^{\circ}_{P^\circ,\mathcal{H}}(t) = f_i$ if $t \neq i \pmod{p}$.

```
set $p :=$ denominator$(P, \mathcal{H})$
set $d := \dim(P)$
set $Q := \left\{Q_1, \ldots, Q_{p(d+1)}\right\}$ (by Proposition 22)
for $i = 0$ to $p - 1$
    for $j = 1$ to $d + 1$
        set $t := ip + j$
        set $F(z) := (-1)^{d+1}\mathrm{Ehr}_{Q_t}\left(\frac{1}{z}\right)$
        set $a_j := F'(0)$
    set $f_i :=$ polyInterpolate$([f_i(ip+1) = a_1], \ldots, [f_i(ip+d+1) = a_{d+1}])$
return $\{f_0(t), \ldots, f_{p-1}(t)\}$
```

■

Proof of algorithm. By [7, Theorem 4.1] $L^{\circ}_{P^\circ,\mathcal{H}}$ is a quasipolynomial of degree d with period dividing $p :=$ denominator$(P, \mathcal{H})$, so we may represent $L^{\circ}_{P^\circ,\mathcal{H}}$ using polynomials $f_0, \ldots, f_{p-1}$. Since each f_i has degree d [7, Theorem 4.1], we may compute $f_i(t)$ for $d+1$ values of t and use polynomial interpolation to determine f_i. To do this we use Proposition 22 to determine a set of polytopes $\left\{Q_1, \ldots, Q_{p(d+1)}\right\}$ where $L^{\circ}_{P^{\circ},\mathcal{H}}(t) = L_{Q_t^{\circ}}(1)$ for $t = 1, \ldots, p(d+1)$. Each iteration of the outer "for loop" generates one polynomial f_i. The inner "for loop" computes $d+1$ values of f_i by determining $d+1$ values of $L^{\circ}_{P^{\circ},\mathcal{H}}(t)$ for $t \equiv i \,(\mathrm{mod}\, p)$. We employ Barvinok's algorithm and Ehrhart-Macdonald reciprocity to obtain the rational generation function $\mathrm{Ehr}_{Q_t^{\circ}}(z)$. The second coefficient in the Taylor series expansion of $\mathrm{Ehr}_{Q_t^{\circ}}(z)$ centered at 0 is $\mathrm{Ehr}'_{Q_t^{\circ}}(0) = L_{Q_t}(1)$. ■

For any rational semi-open polytope S and a rational hyperplane arrangement $\mathcal{H}$, the same mixed integer programming techniques also provide a way to compute the function

$$L^{\circ}_{S,\mathcal{H}}(t) := \#\left(t^{-1}\mathbb{Z}^d \cap S \backslash \bigcup \mathcal{H}\right).$$

For this method first need to know that $L^{\circ}_{S,\mathcal{H}}(t)$ is quasipolynomial and hence can be determined by interpolation.

Proposition 24 *If $P \subseteq \mathbb{R}^d$ is a rational d-polytope and $f_0, \ldots, f_{p-1}$ are the polynomials defining $L_P(t)$, then every f_i is of degree d with leading term equal to the volume of P.*

Proof. [5, Exercise 3.29]. ■

Proposition 25 *If $S \subseteq \mathbb{R}^d$ is a rational semi-open polytope then $L_S(t)$ is a quasi-polynomial in t with period dividing denominator$\left(\overline{S}\right)$. If $f_0, \ldots, f_{p-1}$ are the polynomials defining $L_S(t)$, then every f_i is of degree* $\dim(S)$ *with leading term equal to the volume of P.*

Proof. We observe that for any polytope P we can express P as the pair-wise disjoint union $\bigcup_{F \in \text{faces}(P)} F^\circ$. (Apply Proposition 2.3 and Lemma 2.9 of [22] using induction on $\dim(P)$.) If $F \in \text{faces}(P)$ then every face of F is also a face of P. We also note that $\dim(X) = \dim\left(\overline{X}\right)$ for any convex subset of $\mathbb{R}^d$. [9, Theorem 3.4]

Choose $\mathcal{F}\subseteq \text{faces}\left(\overline{S}\right)$ such that $S = \overline{S} \setminus \bigcup \mathcal{F}$ where $\dim(F) < \dim S$ for all $F \in \mathcal{F}$. Let $\mathcal{F}' := \left\{F \in \text{faces}\left(\overline{S}\right) : F \subseteq \bigcup \mathcal{F}\right\}$. The above observation implies $\bigcup \mathcal{F} = \bigcup \mathcal{F}'$, so S is the disjoint union

$$\bigcup_{F \in \text{faces}(F)} F^\circ \setminus \bigcup_{G \in \mathcal{F}'} G^\circ = \bigcup_{F \in \text{faces}(F) \setminus \mathcal{F}'} F^\circ \setminus \bigcup \mathcal{F} \cup \bigcup_{F \in \mathcal{F}'} F^\circ \setminus \bigcup \mathcal{F} = \bigcup_{F \in \text{faces}(F) \setminus F'} F^\circ.$$

This implies

$$L_S(t) = \sum_{F \in \text{faces}(F) \setminus F'} L_{F^\circ}(t) = L_{P^\circ}(t) + \sum_{\substack{F \in \text{faces}(F) \setminus F', \\ \dim(F) < \dim(S)}} L_{F^\circ}(t).$$

The result now follows by Ehrhart's theorem for rational polytopes and Ehrhart-Macdonald reciprocity. ■

Proposition 26 *If S is a full-dimensional rational semi-open polytope and $\mathcal{H}$ an arrangement of non-degenerate rational hyperplanes, then $L^{\circ}_{S,\mathcal{H}}(t)$ is a degree d quasipolynomial in t with period dividing denominator$\left(\overline{S},\mathcal{H}\right)$. If $f_0,\ldots,f_{p-1}$ are the polynomials defining $L^{\circ}_{S,\mathcal{H}}(t)$ then every f_i is of degree d with leading term equal to the volume of S.*

Proof. Let $\mathcal{C}$ denote the connected components of $S\backslash\bigcup\mathcal{H}$, so that $S\backslash\bigcup\mathcal{H}$ is the pair-wise disjoint union $\bigcup\mathcal{C}$ and $L^{\circ}_{P,\mathcal{H}}(t)=\sum_{C\in\mathcal{C}}L_C(t)$. Every connected component of $P\backslash\bigcup\mathcal{H}$ is a semi-open polytope. Proposition 25 now implies the desired result. ■

The last proposition implies that $\mathbb{Z}^d\cap t\left(S\backslash\mathcal{H}\right)$ can be computed by knowing $L^{\circ}_{S,\mathcal{H}}(t)$ for a sufficient number of values of t. The next demonstrates a method for computing these values.

Proposition 27 *If S is a rational semi-open polytope and $\mathcal{H}$ is a rational hyperplane arrangement, then there is a sequence of polytopes $\{Q_t\}$ such that $L^{\circ}_{S,\mathcal{H}}(t)=L_{Q_t}(1)$.*

Proof. The key observation here is that

$$\left\{\mathbf{x}\in\mathbb{Z}^d:\mathbf{a}^{\intercal}\mathbf{x}<b\right\}=\left\{\mathbf{x}\in\mathbb{Z}^d:\mathbf{a}^{\intercal}\mathbf{x}\leq b-1\right\} \tag{4.9}$$

for any $\mathbf{a}\in\mathbb{Z}^d$ and $b\in\mathbb{Z}$. We now proceed using similar reasoning to Proposition

20. We may assume that

$$S = \left\{\mathbf{x} \in \mathbb{R}^d : A'\mathbf{x} \le b', A''\mathbf{x} < b''\right\}$$

for some $\binom{A'}{A''} \in \mathbb{Z}^{d\times m}$ and $\binom{\mathbf{b}'}{\mathbf{b}''} \in \mathbb{Z}^m$, and that $\mathcal{H} = \{H(\mathbf{c}_1, d_1), \ldots, H(\mathbf{c}_n, d_n)\}$ for some $\mathbf{c}_1, \ldots, \mathbf{c}_n \in \mathbb{Z}^d$ and some $\mathbf{d} \in \mathbb{Z}^n$. Let $C := \begin{pmatrix} \mathbf{c}_1 & \cdots & \mathbf{c}_n \end{pmatrix}^\intercal$, $D := \begin{pmatrix} d_1\mathbf{e}_1 & \cdots & d_n\mathbf{e}_n \end{pmatrix}$, and choose a positive integer

$$\lambda \ge \max\left\{|\mathbf{c}_i^\intercal \mathbf{x}| : \mathbf{x} \in \overline{S}, i = 1, \ldots, n\right\}.$$

Define the sequence $\{Q_t\}$ by $Q_t := P(A_t, \mathbf{b}_t)$ where

$$A_t = \begin{pmatrix} A' & 0 \\ A'' & 0 \\ C & t(D - \lambda I_n) \\ -C & t(D + \lambda I_n) \\ 0 & -I_n \\ 0 & I_n \end{pmatrix} \text{ and } \mathbf{b}_t = \begin{pmatrix} t\mathbf{b}' \\ t\mathbf{b}'' - \mathbf{1} \\ t\mathbf{d} - \mathbf{1} \\ t\lambda\mathbf{1} - \mathbf{1} \\ \mathbf{0} \\ \mathbf{1} \end{pmatrix}. \tag{4.10}$$

By (4.9) , the following statements are equivalent:

$$
\begin{aligned}
\binom{\mathbf{x}}{\mathbf{y}} \quad &\in \quad \mathbb{Z}^{d+n} \cap Q_t \\
&\Leftrightarrow \quad \begin{array}{l} \binom{\mathbf{x}}{\mathbf{y}} \in \mathbb{Z}^{d+n},\ A'\mathbf{x} \leq t\mathbf{b}', A''\mathbf{x} \leq t\mathbf{b}'' - \mathbf{1},\ \mathbf{0} \leq I_n\mathbf{y} \leq \mathbf{1}, \\ C\mathbf{x}+t\left(D-\lambda I_n\right)\mathbf{y} \leq t\mathbf{d} - \mathbf{1} - C\mathbf{x} + t\left(D+\lambda I_n\right)\mathbf{y} \leq t\lambda\mathbf{1} - \mathbf{1} \end{array} \\
&\Leftrightarrow \quad \begin{array}{l} \mathbf{y} \in \{0,1\}^n,\ \mathbf{x} \in \mathbb{Z}^d,\ A'\mathbf{x} \leq \mathbf{b}',\ A''\mathbf{x} < t\mathbf{b}'', \\ \forall i = 1,\ldots,n : \mathbf{c}_i^\intercal\mathbf{x}+t\left(d_i-\lambda\right)y_i \leq td_i - 1, \\ -\mathbf{c}_i^\intercal\mathbf{x} + t\left(d_i+\lambda\right)y_i \leq t\lambda - 1 \end{array} \\
&\Leftrightarrow \quad \begin{array}{l} \mathbf{y} \in \{0,1\}^n,\ \mathbf{x} \in \mathbb{Z}^d \cap tS,\ \forall i = 1,\ldots,n : \\ \mathbf{c}_i^\intercal\mathbf{x}+t\left(d_i-\lambda\right)y_i < td_i,\ -\mathbf{c}_i^\intercal\mathbf{x} + t\left(d_i+\lambda\right)y_i < t\lambda \end{array}
\end{aligned}
$$

as are

$$
\begin{aligned}
\mathbf{x} \quad &\in \quad \mathbb{Z}^d \cap t\left(S \backslash \mathcal{H}\right) \\
&\Leftrightarrow \quad \begin{array}{c} \mathbf{x} \in \mathbb{Z}^d \cap tS, \forall i = 1,\ldots,n : \\ \mathbf{c}_i^\intercal\mathbf{x} < td_i \ \text{ or } \ -\mathbf{c}_i^\intercal\mathbf{x} < -td_i \end{array} \\
&\Leftrightarrow \quad \mathbf{y} \in \{0,1\}^n : \begin{array}{l} \mathbf{x} \in \mathbb{Z}^d \cap tS,\ \forall i = 1,\ldots,n : \\ \mathbf{c}_i^\intercal\mathbf{x}+t\left(d_i-\lambda\right)y_i < td_i, \\ -\mathbf{c}_i^\intercal\mathbf{x} + t\left(d_i+\lambda\right)y_i < t\lambda. \end{array}
\end{aligned}
\tag{4.11}
$$

The remaining proof follows by reasoning similar to Proposition 22, in this case showing that $\varphi : \mathbb{Z}^{d+n} \cap Q_t \to \mathbb{Z}^d \cap t\left(S \backslash \mathcal{H}\right)$ is a bijection. ■

Chapter 5

Inside-out Polyhedral Library

With the goal in mind of tackling some of the larger tractable problems presented in the next chapter – namely the magic and semi-magic squares – we have created an implementation of Algorithm 17 written in C++. (We should note here that the implementation technically runs in exponential time because it uses the dual simplex algroithm [17, Section 11.7] as its linear program solver.) To support this effort we have developed a set of classes containing various optimizations to enhance speed and make efficient use of memory. We refer to these classes collectively as the Inside-out Polyhedral Library (IOP).

5.1 Dependencies and System Requirements

The Inside-out Polyhedral library has been written and tested in a LINUX environment using the gcc compiler. All of IOP's code should conform to ANSI C++ standards, however, IOP depends on pre-existing libraries that may not work on other platforms. IOP should compile properly for any architecture/compiler combination that can use these libraries. IOP directly relies on the Barvinok Library [21], the Polyhedral Library (PolyLib) [19], the CDD Library (cddlib) [13], and, optionally, the computer algebra system PARI/GP [3]. The Barvinok library also requires the Number Theoretic Library (NTL) [18] and an installation of PolyLib compiled with support for the GNU Multi-precision library (GMP) [1]. The Barvinok library provides the primary engine for computing rational Ehrhart generating functions. In addition to supporting the Barvinok library, PolyLib provides preexisting functions to support preprocessing optimizations, most notably the conversion to an equivalent full-dimensional problem (cf. Section 5.3.4). The CDD library provides an exact linear program (LP) solver, used to test whether a hyperplane is transverse to a candidate region. The sum of rational functions output by the Barvinok library is often quite unwieldy in length. GP/PARI provides support for simplifying this.

5.2 Pre-existing Data Structures

Though the libraries mentioned above are all well documented, several of the data structures used directly by the IOP library are worth mentioning here. The first pertain to the arithmetic used at runtime; the second, to the internal representation of inside-out polyhedra and to input files for the example applications mentioned later in this chapter.

Both PolyLib and cddlib can be compiled to support either floating-point or exact rational arithmetic. Numbers used by PolyLib and cddlib are stored in abstract number types `Value` and `mytype,` respectively. Both libraries use flags passed to the compiler to determine the type of arithmetic to be used at runtime. Anyone making number assignments directly using the IOP library should consult the PolyLib and cddlib documentation for more information. It is recommended that all applications be compiled with exact arithmetic flags for PolyLib, as the function calls to the Barvinok library require this. One can compile applications using either arithmetic for cddlib. The floating point arithmetic for cddlib can dramatically reduce the runtime for computing the regions of an inside-out polyhedron, but may result in errors both from register overflow and from the floating point definition of zero, which affects numeric comparisons involving equalities and inequalities [13]. This speed increase offers the ability to obtain a reliable estimate of the number of regions of an inside-out polyhedron.

At the present iteration (version 0.14) only the class `RegionTree` contains function calls to cddlib. The type of arithmetic used by cddlib affects the process used to generate the regions of an inside-out polyhedron, in particular the determination of whether or not a hyperplane is transverse to a polyhedron. In trial runs during the development of IOP, we have experienced no differences in results using cddlib's exact vs. floating-point arithmetic.

IOP uses PolyLib's `Matrix` data type for its internal representation of an inside-out polyhedron. A PolyLib `Matrix` consists of two unsigned integers, `NbRows` and `NbColumns`, giving the row and column dimensions of the matrix, together with a pointer `p` to a two dimensional `Value` array with corresponding row and column dimensions. With the exception of the function `IOPolyhedron::PreImage`, all of the functions in IOP taking `Matrix` pointers as arguments expect a `Matrix` in PolyLib format. If a polyhedron is defined by the system $A\mathbf{x} \leq \mathbf{b}$ ($A \in \mathbb{R}^{m \times d}$), the PolyLib formatted representation of this system is an $m \times d + 2$ matrix of the form $(\mathbf{v}, -A, \mathbf{b})$ where $\mathbf{v} \in \{0, 1\}^m$. A zero in the i^{th} component of $\mathbf{v}$ indicates that the i^{th} inequality in the system $A\mathbf{x} \leq \mathbf{b}$ should be treated as an equality; a one indicates weak inequality.

Example 28 *Let P be the line segment*

$$P = \{(x,y)^{\mathsf{T}} : x = y, x \leq 1, y \geq -1\}$$

and $\mathcal{H}$ is set of coordinate axes along with the line $x = \frac{1}{2}$. The matrices used by IOP to represent P and $\mathcal{H}$ would be the matrices

$$\begin{pmatrix} 0 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \textit{ and } \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -2 & 0 & 1 \end{pmatrix},$$

respectively.

5.3 New Data Structures

The IOP library consists of three C++ classes designed to support an implementation of Algorithm 17. The class `IOPolyhedron` contains the data structures for performing operations on an inside-out polyhedron that do not pertain its regions. The class `RegionTree` is responsible for all operations directly involving the regions of an inside-out polyhedron. `RegionTree` uses a small helper class called `RegionTreeNode`.

5.3.1 C++ Syntax for Non-Programmers

Many of the objects and functions described below use notation that depends on C++'s particular syntax for using classes and pointers. In order to make this notation readable to non-programmers we offer a brief introduction to the relevant syntax. We refer the reader to any introductory book on C++ or object oriented programming for a further discussion of classes and pointers.

Classes have both data and functions associated with them. These functions perform operations using the data stored in particular instances of a class. To make the syntactical explanation more concrete we use an example. Suppose we want the computer to store the time of day using hours and minutes. To accomplish this we define a class of data objects called `time` with two associated integer variables, one called `hour` and the other called `minute`. We may wish to define more than one time of day, say for instance an `arrival` time and a `departure` time. We might also want a way to print the time in a meaningful format, which we accomplish by defining a function called `PrintTime()`. To distinguish between the `hour` and `minute` labels for each time of day we use the name of the `time` variable followed by either `hour` or `minute`. In C++ syntax, to reference the hour of arrival or minute of departure we would write `arrival.hour` or `departure.minute,` respectively. To print the arrival time we would write `arrival.PrintTime()`. Though C++ class definitions are often more complex than this, the same general syntax applies.

A pointer is a reference to the memory address where a variable or instance of a class is stored. The * operator de-references a pointer by returning the value stored at the pointer's address. The & operator returns a pointer to the address where a variable is stored. As an example, if x is a integer variable that has been assigned the value 2, and if xPtr is a pointer that has been assigned the address where x is stored in memory, then we have *xPtr = x = 2 and xPtr = &x. Every instance of a class may refer to its own address in memory using a special pointer called this. In the example used above the function definition for PrintTime() might for instance contain a single line like

```
cout << (*this).hour << ":" << (*this).minute << endl;
```

telling the computer to print the hour and minute, separated by a colon and followed by a new line, (usually) to the screen. If one only wanted to print the departure time, instead of using departure.PrintTime(), one might just as well have substituted

```
cout << departure.hour << ":" << departure.minute << endl;
```

with exactly the same effect.

In the following class descriptions we use typewriter style (Courier) font to indicate C++ statements.

5.3.2 Class `RegionTree`

The class `RegionTree` contains methods used for implementation of Algorithms 13 and 17. In order to improve time and memory efficiency, the implementation of Algorithm 13 appears differently in the code than one would expect. We discuss more theory behind Algorithm 13 and show why it is theoretically equivalent to the implementation described below. We also discuss briefly why the present implementation is more efficient both in terms of memory usage and expected runtime.

Theoretical Support

Let P be a polyhedron and let $\mathcal{H}$ be a hyperplane arrangement with no hyperplane of $\mathcal{H}$ containing P. The fundamental observation motivating Algorithm 13 is that for every fixed ordering of the hyperplanes, say $\mathcal{H} := \{H(\mathbf{c}_1, d_1), \ldots, H(\mathbf{c}_n, d_n)\}$, there exists an encoding of the regions of $(P, \mathcal{H})$ as the Hasse diagram of the poset

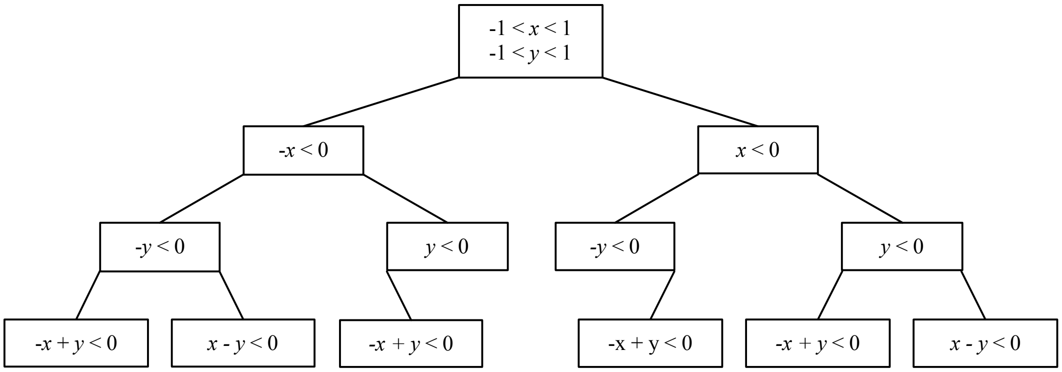
$$P \cup \bigcup_{j=1}^{n} \text{regions}\,(P, \mathcal{H}_j) \tag{5.1}$$

ordered by reverse inclusion where $\mathcal{H}_j := \{H(\mathbf{c}_1, d_1), \ldots, H(\mathbf{c}_j, d_j)\}$. The graph of this Hasse diagram forms a tree, with P the only node at the top level. For reasons that will become apparent, to the definition of this diagram we add the requirement that every level j of the diagram contain a node for each member of regions$(P, \mathcal{H}_j)$; if

R is a region of $(P, \mathcal{H}_j)$ and of $(P, \mathcal{H}_{j+1})$ we place a node representing R on the j^{th} and on the $(j+1)^{th}$ levels connecting them by an edge. We call any such representation of (5.1) a *region tree*, referring to the top node P as the *root.* We index the levels of a region tree starting with 0 instead of 1, corresponding to the fact that the root node represents the region set of $(P, \emptyset)$. The terms *parent* and *child* to refer to the relationship between nodes joined by a single edge, the parent being the closest node to the root. Children sharing a common parent are called *siblings.* Every node in a region tree has at most two children and at most one parent. Only the root node has no parent. A node with no children is a *leaf.* For a node on level j, if the associated region is contained in $K(-\mathbf{c}_j, -d_j)$, we call the node a *left child*; a node representing the region contained in $K(\mathbf{c}_j, d_j)$ is a *right child.* No node has two left (or two right) children. A node on level j has a left child and a right child if and only if $H(\mathbf{c}_{j+1}, d_{j+1})$ is transverse to the region it represents. If a node has no siblings we call it an *only child.* In drawing a diagram of a region tree, children are always placed directly beneath their parents; left children are always placed to the left of the parent, with an analogous situation holding for right children.

Example 29 *Let P the square in $\mathbb{R}^2$ with vertices $\{(x,y) : -1 \leq x, y \leq 1\}$, and let $\mathcal{H}$ be the set of hyperplanes given by the equations $x = 0$, $y = 0$ and $x - y = 0$. The following represents the region tree based on this order of equations. The labels show*

how the structure encodes the open regions of $(P, \mathcal{H})$ *.*



Example of a region tree using open regions

By definition, leaves of a region tree correspond to regions of the associated inside-out polyhedron. Every path from the root to a leaf encodes a description of the associated region by indicating containment in one of two half-spaces associated with each hyperplane in the arrangement based on each node's right/left child status. It is easy to see that a region tree can be built up level by level using Algorithm 13. Hence Algorithm 13 amounts to generating a region tree by performing a breadth-first traversal. Implementing such a traversal requires a running list, the length of which eventually equals the width of the lowest level of the region tree. Because region trees are usually much wider than they are tall, the minimal amount of memory required to perform such a traversal may grow quite large during the course of generating the regions of an inside-out polyhedron.

One can also traverse a region tree using a depth-first tree traversal. This is essentially a backtracking algorithm. One proceeds from the root node down the left-most branch of the tree until one encounters a leaf. After reaching a leaf, one travels back up the path to the lowest untraversed branch, at which point one proceeds down taking the path as far to the left as possible. In this situation, in order to perform a full tree traversal one is required to keep no more nodes in memory than the number of levels (one more than the number of hyperplanes) of the tree. Performing a region tree traversal in this manner means a much smaller memory footprint.

Both traversal methods require that one touch every node in the tree. Using either traversal method, to generate the tree one must still test whether a hyperplane is transverse to a polyhedron for every node except the leaves. This requires at some point that appropriate source matrices be created for an linear program (LP) solver. The depth-first traversal method allows one to reuse much of the information stored in the matrix for previous LP's, as movement from parent to child (or vise-versa) involves altering only a single row of a the matrix representing the region.

The half-space associated with an only child node is redundant with respect to the description encoded by a root-leaf path containing it. This follows from the fact that the hyperplane associated with the only child's level in the region tree is not transverse to its parent. In the implementation of the class `RegionTree` we use this fact to reduce the size of region descriptions (and thus decrease expected runtime) by replacing redundant half space descriptions with a description of the degenerate half-space.

Class Description

The class `RegionTree` contains an implementation of the depth-first traversal of a region tree, and is responsible for all computational operations involving the regions of an inside-out polyhedron. This includes generating region descriptions, computing the generating function $\mathrm{Erh}_{(P,\mathcal{H})}$ and, in future versions, the quasipolynomial $L_{(P,\mathcal{H})}$.

The structures that directly store data are for the most part self-explanatory. The pointers `(*this).P` and `(*this).H` point to matrices representing the inside-out polyhedron $(P,\mathcal{H})$. We use the variable `dilfactor` to implement Proposition 32 below during conversions to full-dimensional representations.

```
/* Data */
gen_fun *genfunction;
Matrix *P,*H;
RegionTreeNode *root;
unsigned currentdepth, dilfactor, numregions;

/* Data for helper objects */
dd_LPPtr lp;
FILE *fileptr;
int (RegionTree::*regionVectorFunction)(Vector*);
std::ostream *streamout;
QQ bv_one; // Used as muliplier for adding gen_fun's
```

```
unsigned counter, maxdepth;
Vector *regionvector;
```

As one can see by the comments in the above code the data stored directly by an object of type `RegionTree` includes a number of extra "helper objects". These help facilitate communication between member functions. The meaning of each object will become clear as we discuss the member functions.

```
void GenerateRegions( GenRegionSubtreeMode mode = NORMAL,
                      int depth = -1,
                      int (RegionTree::*pt2function)(Vector*) = NULL );
```

If `depth == -1`, generates an entirely new region tree for the inside-out polyhedron associated with `(*this).P` and `(*this).H` by performing a depth-first tree traversal. If a region tree has previously been computed, memory for it is deallocated and numregions set to zero before proceeding with the new tree. Otherwise it generates the region subtree formed by using the first `depth` hyperplanes associated with `(*this).H` as the arrangement. If `(*this).curentdepth` $<$ `depth`, the region tree corredponding to the first `depth` hyperplanes is assumed to exist in memory and no action is performed.

If mode == NORMAL the function keeps the entire region tree in memory during the process of tree generation and stores the pointer to the root node in `(*this).root`. If mode == CLEAN, the memory storing a RegionTreeNode is deallocated as soon as the node is no longer relevant to the generation of new regions. Running in this mode effectively erases the entire region tree by the time it has been traversed, and root is appropriately set to NULL before the function returns. In either mode numregions is incremented by one every time a leaf of the region tree is reached, so that numregions reflects the number of regions of the inside-out polyhedron after the process of generating/traversing the region tree is finished.

During the generation/traversal of the tree, two running descriptions of the region represented by each node are held in memory. The first is in the form of a linear program description, pointed to by `lp`. This linear program description is used by cddlib's LP solver to determine whether the next hyperplane is transverse to the region represented by a node. The second representation takes the form of a `Vector` with entries from $\{0, \pm 1\}$, whose length equals the number of hyperplanes in the arrangement. Each component represents a multiplier for the row of `(*this).H` with corresponding index. A zero in the i^{th} component indicates that the i^{th} hyperplane $H(\mathbf{c}_i, d_i)$ does intersect the interior of the region R; hence the half-space from $\{K(\pm\mathbf{c}_i, d_i)\}$ that contains R is redundant. A ± 1 in the i^{th} component of the vector indicates that $R \subseteq K(\pm\mathbf{c}_i, d_i)$. This vector is passed as an argument to the private member function pointed to by `regionVectorFunction` each time a leaf node (region) is reached during the traversal. (If `depth != -1` a leaf represents a region of the region tree formed using the first `depth` hyperplanes of `(*this).H` as the arrangement.) This allows one to perform various operations on regions while the region tree is generated, without having to duplicate the code for generating/traversing a region tree.

```
unsigned GetNumRegions( GenRegionSubtreeMode mode = NORMAL );
```

If the number of regions is equal to zero the function first calls `GenerateRegions( mode, NULL )` prior to returning `(*this).numregions`.

```
void MakeEhrhart( IOP_MakeEhrhartOption option );
```

Performs a full depth-first traversal of a the region tree of the inside out polytope $(P, \mathcal{H})$ represented by `(*this).P` and `(*this).H`, computing the Ehrhart generating function of each region (actually the lattice point enumerator). If `option == GENFUNCTION`, this stores a sum of the generating functions in the object `(*this).genfunction`. At the end of the traversal `(*this).genfunction` represents the rational function for $\mathrm{Ehr}_{(P,\mathcal{H})}(z)$. If `option == GENFUNCTIONFILE`, each region's Ehrhart generating function is printed immediately to `(*this).streamout`. If `(*this).dilfactor == 1` the variable used for output is `z`. If `(*this).dilfactor` >= `1`, `'( z^(*this).dilfactor )'` replaces `z` (cf. Proposition 32 below). In future implementations `option == QUASIPOLY` will spur `MakeEhrhart` to compute a running sum of Ehrhart quasipolynomials, similarly to when `option == GENFUNCTION`.

```
void PutArrangement(Matrix *M);
```

Assigns `M` to the value of `(*this).H`. It is the programmer's responsibility to ensure that the `Matrix` referenced by `M` does not change in a way that will produce undefined results for a `RegionTree` object.

```
void PutDilationFactor( unsigned dil = 1 );
```

Assigns `dil` to `(*this).dilfactor`.

```
void PutPolyhedron(Matrix *M);
```

Assigns `M` to the value of `(*this).P`. See `PutArrangement` above.

```
unsigned
TraverseTreeDepthFirst( int (RegionTree::*pt2function)(Vector*) = NULL,
                        int depth = -1 );
```

If `depth == -1` and a previously generated region tree exists in memory, performs a depth-first traversal of the existing region tree. Otherwise, if `depth <= current-depth` a depth-first traversal is performed using the nodes at level `depth` of the region tree as leaves. A running description of the region associated with a node is kept stored in a vector (cf. `GenerateRegions`). Every time a leaf is encountered the function `*pt2function` is called to perform operations with the associated region. If no previously generated region tree is found or `depth` > `currentdepth`, the function calls `GenerateRegions( CLEAN, depth, pt2function )` to generate/traverse the region tree with similar results. Returns the number of leaves encountered.

```
void WriteRegions( IOP_OutputFormat format=VECTOR, FILE * os=stdout);
```

Writes descriptions of the regions of the associated polyhedron to the file handle `os`.

5.3.3 Class RegionTreeNode

Objects of type `RegionTreeNode` store information about individual nodes in a region tree. All variable names correspond to the terminology used in Section 5.3.2.

```
class RegionTreeNode {
  friend class RegionTree;
  friend class IOPolyhedron;
  public:
    RegionTreeNode() {
      index = 0;
      leftchild = rightchild = NULL;
    }
    unsigned getData() {
      return index;
    }
  private:
    unsigned index;
    RegionTreeNode *leftchild;
    RegionTreeNode *rightchild;
};
```

5.3.4 Class IOPolyhedron

In addition to the data structures and methods used to store and retrieve data pertaining to an inside-out polyhedron, the class IOPolyhedron contains methods that support various preprocessing optimizations. The bulk of these support the conversion of the original problem to an equivalent full-dimensional representation. Others support removing redundancies from the problem description.

Theoretical Support

Proposition 30 *Let $A \in \mathbb{Z}^{m \times n}$, let $\mathbf{b} \in \mathbb{Z}^m$ and suppose $V := \{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} = \mathbf{b}\}$ has dimension d. If $V \cap \mathbb{Z}^n \neq \emptyset$, there exists a vector $\mathbf{v} \in \mathbb{Z}^n$ and a matrix $U \in \mathbb{Z}^{nxd}$, both computable in polynomial time, such that $V \cap \mathbb{Z}^d = \left\{U\mathbf{x} + \mathbf{v} : \mathbf{x} \in \mathbb{R}^d\right\}$.*

Proof. [17, Corollary 5.3c] ■

Proposition 31 *Let $P = P(A, \mathbf{b})$ $(A \in \mathbb{Z}^{m \times n}, \mathbf{b} \in \mathbb{Z}^m)$ be a rational d-polytope with aff$(P) \cap Z^n \neq \emptyset$, and suppose*

$$\text{aff}(P) \cap \mathbb{Z}^d = \left\{U\mathbf{x} + \mathbf{v} : \mathbf{x} \in \mathbb{Z}^d\right\}$$

for some $\mathbf{v} \in \mathbb{Z}^n$ and a $U \in \mathbb{Z}^{nxd}$. For every $t \in \mathbb{Z}_{\geq 1}$, the lattice points of $tP \cap \mathbb{Z}^n$ are in bijection with the lattice points of

$$tP(AU, \mathbf{b} - A\mathbf{v}).$$

Proof. For every $t \in \mathbb{Z}_{\geq 1}$, the function $\varphi_t : \mathbb{R}^d \to t\ \text{aff}(P)$ given by $\varphi_t(\mathbf{x}) = U\mathbf{x} + t\mathbf{v}$ is an affine isomorphism. Hence

$$\varphi_t^{-1}(tP) = \left\{\mathbf{x} \in \mathbb{R}^d : A(U\mathbf{x} + t\mathbf{v}) \leq t\mathbf{b}\right\} = tP(AU, \mathbf{b} - Av).$$

The desired result is obtained using the observation that φ_t defines a bijection between $\mathbb{Z}^d$ and $t\text{aff}(P) \cap \mathbb{Z}^n$. ■

Proposition 32 *If P is a polytope and k is the minimal positive integer such that $k\mathit{aff}(P)$ contains a lattice point, then $\mathit{Ehr}_P(z) = \mathit{Ehr}_{kP}\left(z^k\right)$.*

Proof. For each $i \in \mathbb{Z}_{\geq 0}$ choose $a_i, b_i \in \mathbb{R}$ such that

$$\begin{aligned} \text{Ehr}_P(z) &= a_0 + a_1 z + a_2 z^2 + \cdots \text{ and} \\ \text{Ehr}_{kP}\left(z^k\right) &= b_0 + b_1 z + b_2 z^2 + \cdots. \end{aligned}$$

Let $j \in \mathbb{Z}_{\geq 0}$ and suppose $j \not\equiv 0 \pmod{k}$. Using the fact that $j\text{aff}(P) = \text{aff}(jP)$ and $j\text{aff}(P) \cap \mathbb{Z}^d = \emptyset$, a straight-forward argument shows that $a_j = 0 = b_j$. For the case where $j \equiv 0 \pmod{k}$, choosing $n \in \mathbb{Z}_{\geq 0}$ such that $j = kn$, we obtain

$$a_j = \#\left(\mathbb{Z}^d \cap n(kP)\right) = b_j$$

from the fact that $z^j = \left(z^k\right)^n$. ■

Proposition 33 *Let $P = P(A, \mathbf{b})$ $(A \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m)$ be a rational d-polytope and let $\mathcal{H} = \{H(\mathbf{c}_1, d_1), \ldots, H(\mathbf{c}_{m'}, d_{m'})\}$ be a rational hyperplane arrangement. Let k be the minimal positive integer such that $k\mathit{aff}(P) \cap \mathbb{Z}^n \neq \emptyset$ and suppose that*

$$k\mathit{aff}(P) \cap \mathbb{Z}^d = \left\{U\mathbf{x} + \mathbf{v} : \mathbf{x} \in \mathbb{Z}^d\right\}.$$

If $P' := P(AU, \mathbf{b} - A\mathbf{v})$ and $\mathcal{H}' := \{H(\mathbf{c}_1^{\mathsf{T}} U, d_1 - \mathbf{c}_1^{\mathsf{T}}\mathbf{v}), \ldots, H(\mathbf{c}_{m'}^{\mathsf{T}} U, d_{m'} - \mathbf{c}_{m'}^{\mathsf{T}}\mathbf{v})\}$, then

$$\mathit{Ehr}_{(P,\mathcal{H})}(z) = \mathit{Ehr}_{(P',\mathcal{H}')}\left(z^k\right).$$

Proof. Because the function $\varphi(\mathbf{x}) := U\mathbf{x} + \mathbf{v}$ defines an affine isomorphism from $\mathbb{R}^d$ to aff(P) and because P' and $\mathcal{H}'$ represent the pre-images of P and $\mathcal{H}$ respectively, the regions of $(P', \mathcal{H}')$ are in bijection with the regions of $(P, \mathcal{H})$. In particular each region of $(P', \mathcal{H}')$ is the preimage of a region of $(P, \mathcal{H})$ under φ. An application of Propositions 30, 32, and 33 shows that

$$\begin{aligned}
\mathrm{Ehr}_{(P,\mathcal{H})}(z) &= \sum_{R \in \mathrm{regions}(P,\mathcal{H})} \mathrm{Ehr}_R(z) \\
&= \sum_{R \in \mathrm{regions}(P',\mathcal{H}')} \mathrm{Ehr}_R\left(z^k\right) \\
&= \mathrm{Ehr}_{(P',\mathcal{H}')}\left(z^k\right).
\end{aligned}$$

■

Proposition 34 *Let $A \in \mathbb{Z}^{m\times n}$, let $\mathbf{b} \in \mathbb{Z}^m$ and let $V = \{x \in \mathbb{R}^m : A\mathbf{x} = \mathbf{b}\}$. Let $\mathbf{a}_1^\intercal, \ldots, \mathbf{a}_m^\intercal$ denote the rows of A and for $i = 1, \ldots, m$ let $g_i := \gcd(\mathbf{a}_i)$ and $q_i := \operatorname{lcm}(g_i, b_i)$. If k is the minimal positive integer such that $kV \cap \mathbb{Z}^n \neq \emptyset$, then*

$$k = \operatorname{lcm}\left(\frac{q_1}{b_1}, \ldots, \frac{q_m}{b_m}\right).$$

Proof. Observe that any equation $\mathbf{a}^\intercal\mathbf{x} = b$ has an integer solution if and only if $\gcd(\mathbf{a})$ divides b. For such an equation the minimal positive integer t such that $\mathbf{a}^\intercal\mathbf{x} = tb$ has an integer solution is $t = \frac{\operatorname{lcm}(\gcd(\mathbf{a}),b)}{b}$. Applying this observation to every equation $\mathbf{a}_i^\intercal x = b_i$ shows that $A\mathbf{x} = k\mathbf{b}$ has an integer solution if and only if k is a multiple of $\frac{q_i}{b_i}$. ■

Class Description

The class `IOPolyhedron` represents each inside-out polyhedron internally as two PolyLib matrices (cf. Section 5.2), one for the polytope, the other for the arrangement. In the latter case each row of the matrix represents a hyperplane in the arrangement.

```
/* Data */
Matrix *P,*H; // P represents polyhedron
              // H represents hyperplane arrangement
int affdim, ambdim, codim;
bool canonicalP;
bool canonicalH;
RegionTree *regions;
gen_fun *genfunction;
```

The variables `affdim`, `ambdim`, and `codim` represent the dimension of the polyhedron's affine hull, the dimension of the ambient space, and the co-dimension of the affine hull, respectively. The flag `canonicalP` (respectively, `canonicalH`), is set to `true` if the internal H-description of P (respectively, H) is in irredundant canonical form.

We now describe the public member functions available for objects of type `IOPolyhedron`:

```
void CanonicalizePolyhedron();
```

Removes redundancies from the description of the polyhedron `P`. Rows representing equalities (*linearities*) are placed in the leading rows. This submatrix (representing the affine hull of `P`) is given in Hermite normal form. Rows representing facets are placed last. All rows $\mathbf{a}$ of the `Matrix` pointed to by `P` have $\gcd(\mathbf{a}) = 1$. Since an irredundant description of the affine hull effectively encodes the co-dimension, this also sets the data members `affdim` and `codim` to the correct values.

```
void CanonicalizeArrangement();
```

Not yet fully implemented, in future versions this will remove hyperplanes that are redundant in the affine hull of `P` and sort the rows of `H` lexicographically. All rows $\mathbf{a}$ of the `Matrix` pointed to by `H` have $\gcd(\mathbf{a}) = 1$.

```
void DilateAll( Value t );
void DilateAll( unsigned t );
```

Scales the polyhedron `P` and arrangement `H` by a factor of `t`. This effectively converts the `IOPolyhedron` object `*this` from a description of the inside-out polyhedron $(P, \mathcal{H})$ to a description of the inside-out polyhedron $(tP, t\mathcal{H})$.

```
void DilateAll( IOPolyhedron * NewIOPoly, Value t );
void DilateAll( IOPolyhedron * NewIOPoly, unsigned t );
```

Similar to the last function, this version leave `*this` untouched. The resulting dilation is stored in `*NewIOPoly`.

```
Matrix *GetAffineHullConstraints();
```

If P is not yet in canonical form, `(*this).CanonicalizePolyhedron()` is called. The submatrix of `*P` representing the linearities is copied and a pointer to the copy returned.

```
Matrix *GetArrangementConstraints();
```

Copies the matrix pointed to by`(*this).H` and returns a pointer to the copy.

```
int GetAmbientDimension();
```

Returns the dimension of the ambient space if either `P` or `H` has been intitialized with a valid description; otherwise returns `-1`.

`unsigned GetLatticeGenerators( Matrix **U, Vector **v);`

Computes a minimal set of generators for the intersection of the affine hull of `P` with the integer lattice of the ambient space. Here the `Matrix` and `Vector` pointed to by *`U` and *`v`, respectively, correspond to a description of the form $\text{aff}(P) \cap \mathbb{Z}^n = \left\{U\mathbf{x} + \mathbf{v} : \mathbf{x} \in \mathbb{Z}^d\right\}$ where $d = \dim(P)$.

`unsigned GetNumRegions( GenRegionSubtreeMode mode = NORMAL);`

Returns the number of regions of the inside-out polyhedron represented by `*this` by calling `regions->GetNumRegions( mode )`. See the description GetNumRegions in class RegionTree for an understanding different effects that `mode` has on computation and memory usage.

`int GetPolyhedronDimension();`

Returns the affine dimension of the polyhedron represented by `P`. If `affdim` is defined (i.e. greater than or equal to -1) this simply returns `affdim`. Otherwise `(*this).CanonicalizePolyhedron()` is called prior to returning `affdim`.

`Matrix *GetPolyhedronConstraints();`

Makes a copy of the `Matrix` pointed to by `(*this).H` and returns a pointer to the copy.

```
void MakeEhrhart( IOP_MakeEhrhartOption option = GENFUNCTION );
```

Computes a lattice point counting function for the inside-out polyhedron represented by *this. If option == GENFUNCTION the rational generating function $\mathrm{Ehr}_{(P,\mathcal{H})}$ is created and a pointer to its representation is stored in (*this).genfunction. If option == GENFUNCTIONFILE, then the generating function of each region is computed and immediately output to stdout on a separate line. In future implementations if option == QUASIPOLY, a representation of the Ehrhart quasipolynomial $L_{(P,\mathcal{H})}$ will be created.

```
unsigned MakeRegionTree( GenRegionSubtreeMode mode = NORMAL,
                         int depth = -1 );
```

If depth == -1 generates the region tree of the inside-out polytope represented by *this and returns the number of regions. Otherwise, generates the region subtree formed by using the first depth hyperplanes of (*this).H as the arrangement. If mode == NORMAL the entire representation is stored in memory for future use. If mode == CLEAN, in order to save memory the branches of the region tree are pruned (deleted) as soon as they are no longer useful for generating untouched nodes of a region tree. (cf. TraverseRegionSubtree, Section 5.3.2)

```
unsigned GenerateFullDimEquivalent(IOPolyhedron *newiopoly);
```

Let $(P, \mathcal{H})$ be the inside-out polytope represented by `*this`. If $\operatorname{aff}(P)$ contains a lattice point, the function computes the description of a full-dimensional equivalent inside-out polyhedron, stores the result in `*newiopoly` and returns 1. If $\operatorname{aff}(P)$ is found to be integrally empty, this computes the minimal positive integer t such that $\operatorname{aff}(tP)$ contains a lattice point, stores a representation of $(tP, t\mathcal{H})$ in `*newiopoly` and returns t.

```
void PreImage( IOPolyhedron *IOPoly, Matrix *U, Vector *v);
```

Computes the pre-image of `*this` under the affine transformation $\mathbf{x} \mapsto U\mathbf{x} + \mathbf{v}$ and stores the result in `*IOPoly`.

```
IOPolyhedron *PreImage( Matrix *U, Vector *v);
```

Similar to the previous function, this version of the function computes the preimage of the transformation $\mathbf{x} \mapsto U\mathbf{x} + \mathbf{v}$ and assigns the result to `*this`.

```
void PrintArrangement( FILE *stream = stdout,
                       IOP_OutputFormat format = PolyLib );
```

Prints a matrix representing the hyperplane arrangement to the file handle `stream`. Currently the only output format is `PolyLib`. Future implementations will include options for `format == CDD` and `format == LATTE`.

```
void PrintGenFunction( std::ostream& stream, unsigned dilfactor = 1 );
```

If a representation of $\mathrm{Ehr}_{(P,\mathcal{H})}$ has already been computed this simply outputs $\mathrm{Ehr}_{(P,\mathcal{H})}$ to stream. Otherwise $\mathrm{Ehr}_{(P,\mathcal{H})}$ is computed prior to output.

```
void PrintIOPolyhedron( FILE *stream = stdout,
                        IOP_OutputFormat format = PolyLib );
```

If *this represents the inside-out polyhedron $(P, \mathcal{H})$, a matrix representing P followed by a matrix representing $\mathcal{H}$ is output to stream. Currently the only output format is PolyLib. Future implementations will include options for format == CDD and format == LATTE.

```
void PrintPolyhedron( FILE *stream = stdout,
                      IOP_OutputFormat format = VECTOR );
```

Prints a matrix representing the polyhedron to the file handle stream. Currently the only output format is PolyLib. Future implementations will include options for format == CDD and format == LATTE.

```
void PrintRegions( FILE *stream = stdout,
                   IOP_OutputFormat format = VECTOR );
```

Prints a set of descriptions of the regions of $(P, \mathcal{H})$ to the file handle stream. Currently the only supported formats are VECTOR and PolyLib. Selection of vector format tells the function to write one vector, of length equal to the number hyperplanes in $\mathcal{H}$, to stream. A vector $\mathbf{v}$ representing a region has entries from $\{-1, 0, 1\}$.

If $v_i = 0$, then the hyperplane $H(\mathbf{c}_i, d_i)$ of $\mathcal{H}$ is not a supporting hyperplane of the region R; hence the associated half-space containing R does not contribute to the description. If $v_i = \pm 1$, then $R \subseteq P \cap K(\pm\mathbf{c}_i, \pm d_i)$.

```
int PutArrangement( FILE *stream, IOP_OutputFormat = PolyLib );
```

Reads a matrix from the file handle `stream`, converts the information to a PolyLib formatted `Matrix M,` and attempts to set `*this.H` to point to the result by calling `PutArrangement(&M).` Returns 0 if successful.

```
int PutArrangement( Matrix *M );
```

If `(*this).P` is not yet defined, makes a copy of `*M`, sets `(*this).H` to point to the copy and returns 0 indicating success. If `(*this).P` is already defined and the row dimensions match, the same action is performed. Otherwise the function halts further computation by calling `exit(1)` and prints an error message.

```
void PutDilationFactor( unsigned dil = 1 );
```

Sets `(*this).regions->dilfactor` to the value of `dil`.

```
int PutPolyhedron( FILE *stream, IOP_OutputFormat = PolyLib );
```

Similar to the function `PutArrangement( FILE *, IOP_OutputFormat ).`

```
int PutPolyhedron( Matrix *M );
```

Similar to `PutArrangement( Matrix *)`

5.4 Example Applications

The goal of tackling the magic, semi-magic, magilatin, and magisudoku examples in the next chapter motivated the development of the classes and functions outlined above. In support of this goal the following example applications have all contributed to the results in the next chapter. They have been written with integration in the UNIX style pipe system in mind. Input is always taken from the standard input stream, and relevant data is always sent the standard output stream. Progress indicators for completed computations are always directed to the standard error stream, so that a command like

```
./ioregioncount <K6.iop >K6.cnt
```

will save only the number of regions of `K6.iop` to the file `K6.cnt` while also printing a running progress update to the screen.

5.4.1 Input/output file format

Each application listed below expects to receive two PolyLib formatted matrices as input, the first representing the polyhedron, and the second, the arrangement. The text representation of these matrices requires that the first line contain the row and column dimensions of the matrix separated by white space. Each subsequent line represents a matrix row with components separated by white space.

Example 35 *To use the problem given in Example 28 as input for one of the applications below we would input the following lines:*

```
3 4
0  1 -1  0
1 -1  0  1
1  0  1  1
3  4
0  1  0  0
0  0  1  0
0 -2  0 1
```

5.4.2 `fulldim`

If $(P, \mathcal{H})$ is the inside-out polyhedron represented by the input matrices, `fulldim` outputs a full dimensional representation of $(kP, k\mathcal{H})$ where k is the smallest positive integer such that aff(kP) contains a lattice point. The output format corresponds to that described in Section 5.4.1, with the slight difference that we append a line containing the value k. The description of the full dimensional polyhedron is given in the canonical form generated by PolyLib's internal representation of a `Polyhedron`.

Currently, no redundancy removal is performed on the full-dimensional version of the arrangement matrix, nor are the hyperplane descriptions normalized so that the coefficients have no common divisor. Future versions of IOP will include this as a feature.

5.4.3 `ioregions`

Outputs a set of vectors separated by new lines, each representing a region of the inside-out polyhedron associated with the input matrices. (cf. `GenerateRegions`, Section 5.3.2)

5.4.4 `ioregioncount` and `ioregioncount_fp`

Both output the number of regions of the inside-out polyhedron associated with the two input matrices. The only difference lies in the fact that `ioregioncount_fp` uses floating point instead of exact arithmetic to test whether or not a hyperplane is transverse to a polyhedron (cf. Section 5.2). Consequently `ioregioncount_fp` runs approximately ten times faster than `ioregioncount`. To date we have encountered no differences in output using the two different arithmetics, but those who wish to be absolutely certain of the results should use `ioregioncount`.

5.4.5 ioehrhart, ioehrhart_clean, and ioehrhart_file

Each of these outputs a first line containing the dimension of the polytope associated with the first input matrix. With **ioehrhart** and **ioehrhart_clean** this is followed by a line containing an unsimplified sum of rational functions representing the generating function $\mathrm{Ehr}_{(P,\mathcal{H})}(z)$. This sum is generally so long as to be unreadable, and in some instances can be quite large (over 1gb on disk). The difference in the two applications lies in the fact that **ioehrhart** generates the entire region tree prior to beginning the computation of each region's Ehrhart generating function, storing the entire region tree in memory for the duration of its runtime, whereas **ioehrhart_clean** generates the region tree and computes the Ehrhart generating function of each region all in the same pass, deallocating the memory for each node as soon as it is no longer useful for further computations (cf. MakeEhrhart, Section 5.3.2). With **ioehrhart** the number of regions is output to the screen (actually the standard error stream) prior to beginning the process of computing generating functions; whereas with **ioehrhart_clean** no such count is given, but the application runs with a smaller memory footprint and a minimal increase in speed.

With **ioehrhart_file** each output line following the first contains the Ehrhart generating function of a single region of the inside-out polytope. This traverses the regions tree in a similar manner to **ioehrhart_clean**, and in general runs with a memory footprint that depends mostly on the resources needed to compute the generating function of any of the regions. (The internal representation used by the Barvinok library only simplifies terms with the same denominator, so memory usage with `ioehrhart` and `ioehrhart_clean` grows slightly slower than linearly in the number of regions.) Based on tests performed during debugging, the processor time needed to obtain a simplified generating function using `ioehrhart_file` exceeds that of `ioehrhart_clean` by about five percent. However, in practice `ioehrhart_file` is limited only by disk space and runtime as to number of regions it can tackle.

One can pipe the output of any of these applications to the perl script `gfsimplify.pl` described below to obtain simplified representations of the rational generating functions $\mathrm{Ehr}_{(P,\mathcal{H})}(z)$ and $\mathrm{Ehr}^{\circ}_{(P^{\circ},\mathcal{H})}(z)$. One can also use any of these applications to obtain the Erhhart generating function of a polytope by using a description of the empty hyperplane as the arrangement matrix.

5.4.6 `ioehrhart_nopt`

This performs essentially the same function as `ioehrhart_clean`, except that no preprocessing optimizations are performed. The function `IOPolyhedron::MakeEhrhart` is called using the data exactly as it was input. Because the dimension of the polyhedron is computed during the process of redundancy removal, the dimension is not printed to the first line of output as with the other `ioehrhart` applications. This is useful for situations where one has to process a large set of inside-out polyhedra with descriptions that are close to optimal (see `ioerhart_subtrees.pl` and `subtrees` below).

5.4.7 `subtrees`

This application was written specifically to enable the computation of region counts and rational generating functions in parallel using a network of computers and a network job scheduler. Based on the input description, `subtrees` outputs a set of inside-out polyhedra, whose regions taken together form the regions of the original inside-out polyhedron. Using a lower bound n this finds the highest level of the region subtree containing at least n nodes, and outputs a description of the inside-out polyhedron associated with each node having root on that level. Because each level of a region tree contains no more than double the number of nodes of the previous level, the actual number of output descriptions is between n and $2n-1$. The default value for n is 2500. One may supply other values of n using the first argument of the command line. The inside-out polyhedron description must come from `stdin`.

5.4.8 `gfimplify.pl` and `gfsimplify_hc.pl`

Takes the output produced by `ioehrhart`, `ioehrhart_clean`, `ioehrhart_file`, and `ioehrhart_subtrees.pl` and computes a simplified canonical representation of the functions $\mathrm{Ehr}_{(P,\mathcal{H})}(z)$ and $\mathrm{Ehr}^{\circ}_{(P^{\circ},\mathcal{H})}(z)$. Neither the numerator nor the denominator share a common divisor and both are in expanded form. Problems, for which `gp`'s stack allocation is insufficient should work using `gpsimplify_hc.pl`, but the instance of `gp` intitiated by the script uses considerably more memory.

5.4.9 `graph2iop.pl`

Takes as input the description of a graph and outputs a description of an inside-out polytope that can be used to compute the nowhere-zero k-flow polynomial of the graph. (cf. Section 6.1) The first line of input is expected to contain the number of edges followed by the number of vertices. Each subsequent line should contain an edge represented as a pair of vertex indices separated by white space. Using index values larger than the number of vertices specified will result in undefined behavior.

5.4.10 ioehrhart_subtrees.pl

Originally written to verify the functionality of `subtrees` and the method used for `ioehrhart2cluster.pl`, this script mimics the behavior of `ioerhrhart_file` in that it outputs the dimension of the polyhedron followed many rational generating functions. Instead of printing $\mathrm{Ehr}_R(z)$ to a separate line for each region R of the original problem, `ioehrhart_subtrees.pl` prints $\mathrm{Ehr}_{(Q,\mathcal{J})}(z)$ to a separate line for each inside-out polyhedron $(Q, \mathcal{J})$ generated by `subtrees`. For problems that are sufficiently large to eat up all available memory, this allows one to split the original problem into a (roughly) specified number of smaller problems and solves each one separately using `ioehrhart_nopt`. As with `subtrees`, problem descriptions are taken from `stdin` and one may specify a lower bound n as the first argument to the command line if one does not wish to use the default value 2500. Output of `ioehrhart_subtrees.pl` may be piped to `gfsimplify.pl` to obtain simplified forms of $\mathrm{Ehr}_{(P,\mathcal{H})}(z)$ and $\mathrm{Ehr}^{\circ}_{(P^{\circ},\mathcal{H})}(z)$ where $(P, \mathcal{H})$ represents the original inside-out polyhedron.

5.4.11 `ioehrhart2cluster.pl` and `ioregioncount2cluster.pl`

These scripts are designed to be used with the network job scheduling application Platform Lava [10]. Using `subtree` each script breaks up a region tree into a set of subtrees that together contain the same regions as the original problem, so that computations may be performed in parallel on these subtrees by a cluster of computers. Depending on the script used, each job computes the rational generating function or number of regions associated with a region subtree. After all jobs are finished a sum of the results is computed

Input syntax is the same for both scripts of the same form. At the command line type '`scriptname srcfile`' where scriptname is either '`ioehrhart2cluster.pl`' or '`ioregioncount2cluster.pl`', and `srcfile` is an optional argument specifying the name of the source file. Each script prompts for a source file name if no argument is given. They also prompt for a directory `datadir` in which to store data (this needs to be a directory that does not yet exist), and for a lower bound n on the number of subtrees to be used. The actual number N of subtrees generated satisfies $n \leq N < 2n$ and depends on the first level of the region subtree found to contain at least n nodes. Subtree descriptions are stored in the directory `datadir/jobs`. The output resulting from computations performed on each subtree is stored in a file with corresponding name in the directory `datadir/results`. Messages to `stderr` for each such computation are stored in a file with corresponding name in `datadir/log`. After computations on all subtrees are performed, the results are added together by concatenating all of the files in `datadir/results` and piping this to `simplify.pl`. The result is stored in the file `datadir/srcfile.gen` or `data/srcfile.cnt` depending on which script is used.

All jobs in the queue maintained by Lava can be found under the job name `srcfile`.

5.4.12 `simplify.pl` and `simplify_hc.pl`

This takes input from `stdin` and uses PARI/GP to compute a sum using each line of the input stream as a term in the sum. Simplified results are piped to `stdout`.

5.5 Installation

A working installation of IOP takes up roughly 25mb of disk space. The directory used to build the supporting libraries can grow to over 200mb but may be deleted once the installation process is complete. To install, download the file `ioplib-**.tar.gz`, where `**` represents the version number. After download, assuming you are in the same directory as the downloaded file, type

```
tar xvfz ioplib-**.tar.gz
```

This will create the build directory `./ioplib-**`. Installation instructions can be found in the file

`/.../ioplib-**/README`.

5.6 Known Issues and Bugs

At the present time IOP assumes that all polyhedra are actually polytopes. In particular IOP never tests for boundedness prior to attempting to compute generating functions. Whatever error handling is done in this situation is left to PolyLib and the Barvinok library. This assumption also affects the generation of the regions of an inside-out polyhedron. The LP solver in cddlib detects when an objective function is unbounded with respect to a feasible region; however, the data structures indicating this are not currently used during the process of generating regions. In future versions this assumption will be eliminated entirely using appropriate error handling.

Undefined behavior may result from feeding IOP a description in which the polyhedron is contained in one of the hyperplanes. This problem will be fixed at the same time as we implement the removal of redundant hyperplanes from an inside-out polyhedron description.

Chapter 6

Computational Examples

Based on [7], [8], and [6], we now present a number of example applications of inside-out polytopes computed using the applications from the Inside-out Polyhedral Library.

6.1 Nowhere-Zero k-Flow Polynomials for Graphs

We define the term *graph* to mean any undirected multigraph (multiple edges and loops allowed). A *flow* on a graph $G = (V, E)$ with values from an abelian group A

(*A-flow*), is a mapping $x : E \to A$ such that for every vertex $v \in V$,

$$\sum_{h(e)=v} x(e) = \sum_{t(e)=v} x(e) \tag{6.1}$$

where $h(e)$ and $t(e)$ represent the head and tail of the edge e under a fixed orientation of G. Informally x assigns a weight to every directed edge e such that the sum of edges directed into a given vertex equals the sum of edges directed out of the vertex. A *nowhere-zero flow* is an A-flow whose values are never zero. A *nowhere-zero k-flow* is a nowhere-zero $\mathbb{Z}$-flow with values in $\{\pm1, \pm2, \ldots, \pm(k-1)\}$.

In order to construct A- and k-flows one needs technically to consider a fixed orientation of a graph. (This is just the assignment of a head and tail to each edge.) The number of A- and k-flows on a graph is the same for any orientation chosen. To simplify notation we tacitly assume, where necessary, that a given graph is directed.

Tutte proved [20] that the number of nowhere zero flows on G from an abelian group A is a polynomial function in $\#A$, independent of A itself. We refer to this polynomial, $\overline{\varphi}_G(k)$, as the *(strict) modular flow polynomial* or *$\mathbb{Z}_k$-flow polynomial.* More recently Kochol proved [15] that the number of nowhere-zero k-flows on G is a polynomial, φ_G, in k, which we call the *(strict) integral flow polynomial* or *k-flow polynomial.* Beck and Zaslavsky offer an independent proof of this in [8] and compute examples of k-flow and $\mathbb{Z}_k$-flow polynomials for a number of small graphs. Here we expand on these examples using `ioehrhart` to compute integral flow polynomials.

The strong and weak flow polynomials of a graph are not in general the same. However they do share the same positive integer roots. Put another way, a graph admits a k-flow if and only if it admits a $\mathbb{Z}_k$-flow. (See [12, Theorem 6.3.3] for a proof.) This leads to the following observation, partially answering Problem 3.5 of [8].

Proposition 36 *Let G be a multigraph with real cyclespace $U \subseteq \mathbb{R}^n$ and let $d = \dim(U)$. Let $\overline{\varphi}_G(k)$ and $\varphi_G(k)$ denote the nowhere-zero $\mathbb{Z}_k$- and k-flow polynomials of G, respectively. If either $\overline{\varphi}_G(k)$ or $\varphi_G(k)$ have d as a root, then*

$$\overline{\varphi}_G(k) = \prod_{i=1}^{d} (k-i) \ \ \text{and}\ \varphi_G(k) = (d+1)\,\overline{\varphi}_G(k)\,.$$

Proof. We make the following observations: First, if k is a positive integer and $m \geq k$, then every k-flow is also an m-flow. Second, the polynomial $\overline{\varphi}_G$ is always monic and both $\overline{\varphi}_G$ and φ_G have degree d . [8] Third, both polynomials share the same positive integer roots. In particular d and (because of the nowhere-zero stipulation) 1 are roots of both φ_G and $\overline{\varphi}_G$.

We first prove $\overline{\varphi}_G(k) = \prod_{i=1}^{d}(k-i)$ by showing that $2, \ldots, d-1$ are roots of $\overline{\varphi}_G$. Let $r \in \{2, \ldots, d-1\}$ and suppose by way of contradiction that r is not a root of $\overline{\varphi}_G$. Then there exists a nowhere-zero $\mathbb{Z}_r$-flow on G and, by the second observation, a nowhere-zero r-flow. This implies the existence k-flows on G for all $k \in \mathbb{Z}_{\geq r}$. In particular, there is a nowhere-zero d-flow, meaning d is not a root of φ_G. This contradiction proves r must be a root of $\overline{\varphi}_G$. As $1, 2, \ldots, d$ are roots of $\overline{\varphi}_G$, the first observation implies $\overline{\varphi}_G(k) = \prod_{i=1}^{d}(k-i)$.

Since $1, 2, \ldots, d$ are also roots of φ_G, it follows that $\overline{\varphi}_G(k)$ divides $\varphi_G(k)$. Because φ_G and $\overline{\varphi}_G$ have the same degree, there exists a constant c such that $\varphi_G(k) = c\overline{\varphi}_G(k)$. Theorem 3.1(c) of [8] implies $\varphi_G(0) = \overline{\varphi}_G(-1)$. In particular

$$c\overline{\varphi}_G(0) = \prod_{i=1}^{d}(-1-i) = \prod_{i=1}^{d+1}(0-i) = (d+1)\,\overline{\varphi}_G(0).$$

Hence $c = d+1$. ■

Establishing stronger links in the relationship between φ_G and $\overline{\varphi}_G$ is largely an open question. For this reason we include the nowhere $\mathbb{Z}_k$-flow polynomial and the Tutte polynomial (which encodes many graph invariants) along side our results from `ioehrhart`. For these latter polynomials we use the Networks package from Maple. We also include the *weak integral and modular flow polynomials*, φ_G^0 and $\overline{\varphi}_G^0$, which respectively count the number of possibly zero-valued $\mathbb{Z}_k$- and k-flows.

Motivated by [8], we now outline the methods used to compute φ_G and φ_G^0 using inside-out polytope theory. Let $G = (V, E)$ be a directed graph with $V = \{v_1, \ldots, v_m\}$ and $E = \{e_1, \ldots, e_d\}$. The equations given by (6.1) naturally define a linear subspace U of $\mathbb{R}^d$ (via the map $x \mapsto (x(e_1), \ldots, x(e_d))$) called the *real cycle space of* G. Let $P := U \cap [-1,1]^d$ and $k \in \mathbb{Z}^+$. One can show (see Lemma 2.9, [22]) that $kP^\circ = U \cap (-k,k)^d$. In particular this means that

$$\mathbb{Z}^d \cap kP^\circ = \mathbb{Z}^d \cap (k-1)\, P.$$

If $\mathcal{H}_d$ is the arrangement of coordinate hyperplanes in $\mathbb{R}^d$, then the possibly-zero k-flows on G are in bijection with the set

$$\mathbb{Z}^d \cap U \cap (k-1)\,[-1,1]^d = \mathbb{Z}^d \cap (k-1)\, P = \mathbb{Z}^d \cap kP^\circ.$$

Similarly, the nowhere-zero k-flows on G are in bijection with the set $\mathbb{Z}^d \cap k\left(P^\circ \backslash \bigcup \mathcal{H}_d\right)$. Hence the problem of finding φ_G^0 and φ_G amounts to computing L_{P° and $L^\circ_{P^\circ,\mathcal{H}_d}$, respectively.

The *edge-vertex incidence matrix* of G is the $m \times d$ matrix $A_G := (a_{ij})$ where

$$a_{i,j} = \begin{cases} 1, \text{ if } v_i = h(e_j) \\ -1, \text{ if } v_i = t(e_j) \\ 0, \text{ otherwise} \end{cases}.$$

A straight-forward argument shows that null(A_G) is the real cycle space of G, so that

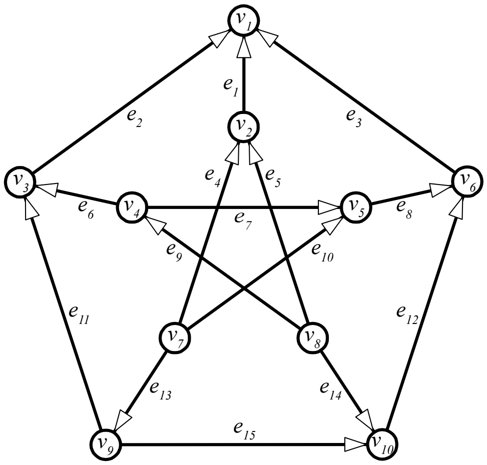
$P = P(S, \mathbf{t})$ where

$$S = \begin{pmatrix} A_G \\ \\ -A_G \\ \\ I \\ \\ -I \end{pmatrix} \text{ and } \mathbf{t} = \begin{pmatrix} \mathbf{0} \\ \\ \mathbf{0} \\ \\ \mathbf{1} \\ \\ -\mathbf{1} \end{pmatrix}. \tag{6.2}$$

With a half-space description for P in hand we now compute Ehr_{P° and $\mathrm{Ehr}_{(P^\circ, \mathcal{H}_d)}$ using `ioehrhart`. The fact that A_G is always totally unimodular [17] implies that the vertices of P and $(P, \mathcal{H}_d)$ have integer coordinates, so that $L_{P^\circ}(t)$ and $L^\circ_{P^\circ, \mathcal{H}_d}(t)$ are both polynomials in t. To recover $L_{P^\circ}(t)$ and $L^\circ_{P^\circ, \mathcal{H}_d}(t)$ we simply use polynomial interpolation on the first $\dim(P)+1$ terms of $\mathrm{Ehr}_{P^\circ}(z)$ and $\mathrm{Ehr}_{P^\circ, \mathcal{H}_d}(z)$, respectively.

Example 37 *Flow polynomials of the Peterson graph.*

Let G denote the Peterson graph and let $\varphi(t)$ and $\varphi_0(t)$ denote the strong and weak k-flow polynomials of G, respectively. We label and orient G as follows:



Oriented Peterson graph

Under this orientation the real cycle space of G is $\text{null}(A_G)$ where A_G is the matrix

$$\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1
\end{pmatrix}$$

Let $P := P(S, \mathbf{t})$ where S and $\mathbf{t}$ are defined by (6.2) and let

$$\mathcal{H}_{15} := \{H(\mathbf{e}_1, 0), \ldots, H(\mathbf{e}_{15}, 0)\}$$

where $\mathbf{e}_1, \ldots, \mathbf{e}_{15}$ represents the standard basis of $\mathbb{R}^{15}$; then $\varphi(t) = L^{\circ}_{P^{\circ}, \mathcal{H}_{15}}(t)$ and $\varphi_0(t) = L_{P^{\circ}}(t)$. Using the specified input format, we feed the matrices

$$\begin{pmatrix} \mathbf{0} & -A_G \\ \mathbf{0} & A_G \\ \mathbf{1} & -I \\ -\mathbf{1} & I \end{pmatrix} \text{ and } \begin{pmatrix} \mathbf{0} & -I \end{pmatrix}$$

to `ioehrhart` and obtain the simplified rational generating function

$$\mathrm{Ehr}^{\circ}_{P^{\circ}, \mathcal{H}_{15}}(z) = -120 \frac{z^5 (16z^2 + 19z + 20)}{(z-1)^7}.$$

We use `ioehrhart` to compute $\mathrm{Ehr}_P(z)$ by appending a description of the empty hyperplane and apply Theorem 5 to obtain the rational function

$$\mathrm{Ehr}_{P^{\circ}}(z) = -\frac{z(z^6 + 132z^5 + 1533z^4 + 3268z^3 + 1533z^2 + 132z + 1)}{(z-1)^7}.$$

Using Lemma 2 and Theorem 4, one can verify that $L_P(t)$ and $L_{P,\mathcal{H}}(t)$ have degree $6 = \dim(P)$. (One can also verify this using IOP.) We now compute the first 7 terms of the Taylor series expansions of both $\mathrm{Ehr}^{\circ}_{P^{\circ},\mathcal{H}_{15}}(z)$ and $\mathrm{Ehr}_{P^{\circ}}(z)$

$$\begin{aligned}\mathrm{Ehr}^{\circ}_{P^{\circ},\mathcal{H}_{15}}(z) &= 2400z^5+19080z^6+85080z^7+O(z^8)\\ \mathrm{Ehr}_{P^{\circ}}(z) &= z+139z^2+2485z^3+17779z^4+78631z^5\\ &\quad +259321z^6+702199z^7+O(z^8)\end{aligned}$$

Using polynomial interpolation and the definitions of the Ehrhart series for P and $(P,\mathcal{H}_{15})$ we obtain the flow polynomials φ and φ_0. We use Maple's Networks package to compute the strong $\mathbb{Z}_k$-flow polynomial. The weak $\mathbb{Z}_k$-flow polynomial is always $t^{\dim(\mathrm{null}(A_G))}$. Hence, the flow polynomials for the Peterson graph are:

$$\begin{aligned}\varphi(t) &= \tfrac{1}{6}(t-1)(t-2)(t-3)(t-4)\left(55t^2-251t+480\right),\\ \varphi_0(t) &= \frac{1}{12}\left(110t^6-330t^5+485t^4-420t^3+233t^2-78t+12\right),\\ \overline{\varphi}(t) &= (t-1)(t-2)(t-3)(t-4)\left(t^2-5t+10\right),\ \text{and}\\ \overline{\varphi}_0(t) &= t^6.\end{aligned}$$

The process outlined in Example 37 has been entirely automated through a series of Maple and UNIX shell scripts. We have generated flow polynomials for a sizeable number of small graphs, all of which are catalogued in Appendix A. A *bridge* in a graph is an edge whose removal renders the resulting graph disconnected. The number of nowhere-zero flows on a graph with a bridge is always zero [12, Corollary

6.1.2]. As flow polynomials for disconnected graphs can be obtained by taking the product of the flow-polynomials for the connected components, we consider only connected, bridgeless graphs in our examples.

6.2 Magic and Semi-magic Squares

A *semi-magic n-square* is an $n \times n$ square matrix with distinct integer entries such that every row and every column sum to the same number, the *magic sum*. A *magic n-square* is a semi-magic n-square in which entries along each main diagonal also have the same magic sum. To simplify notation we often treat $n \times n$ matrices as vectors in $\mathbb{R}^{n^2}$. If $X \in \mathbb{R}^{n\times n}$ we always assume the index assignment of $X \in \mathbb{R}^{d^2}$ is given by

$$X = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ x_{n+1} & x_{n+1} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(n-1)n+1} & x_{(n-1)n+2} & \cdots & x_{n^2} \end{pmatrix}.$$

The set of real matrices having the same line-sum, column-sum, and main diagonal-sum naturally defines a subspace S of $\mathbb{R}^{n^2}$. A vector in $\mathbb{R}^{n^2}$ has distinct entries if and only if it is off the hyperplane

$$H_{i,j} := \left\{\mathbf{x} \in \mathbb{R}^{n^2} : x_i - x_j = 0\right\}$$

for all distinct $i, j \in n^2$. Let $\mathcal{H}$ be the arrangement

$$\left\{H_{i,j} : i, j \in n^2, i \neq j\right\}; \tag{6.3}$$

then the set of magic n-squares is precisely $\mathbb{Z}^{n^2} \cap S \backslash \bigcup \mathcal{H}$.

Traditionally, the entries for magic squares come from the positive integers. Here we consider two variations on this, one in which we require that entries come from $[t] := \{1, 2, \ldots, t\}$ (*cubical count)*, and the other in which we draw from the positive integers, but consider only squares with magic sum equal to t (*affine count).* In both cases we count the number of squares as a function of t.

Let $\mathcal{C}_n(t)$ denote the number magic n-squares with entries from $[t]$. One can verify that

$$\begin{aligned} \mathcal{C}_n(t) &= \#\left[\mathbb{Z}^{n^2} \cap [1,t]^{n^2} \cap S \backslash \bigcup \mathcal{H}\right] \\ &= \#\left[\mathbb{Z}^{n^2} \cap (t+1)\left((0,1)^{n^2} \cap S \backslash \bigcup \mathcal{H}\right)\right]. \end{aligned}$$

It is easy to see that

$$C := [0,1]^{n^2} \cap S \tag{6.4}$$

is a polytope with $C^{\circ} = (0,1)^{n^2} \cap S$, so that for all t we have

$$\mathcal{C}_n\left(t\right) = L^{\circ}_{C^{\circ},\mathcal{H}}\left(t+1\right). \tag{6.5}$$

Let $\mathcal{A}_n\left(t\right)$ denote the number of n-squares with magic sum t and let B_n be the well known Birkhoff polytope,

$$B_n := \left\{(b_{i,j}) \in \mathbb{R}^{n\times n} : b_{i,j} \geq 0, \sum\nolimits_{i=1}^{n} b_{i,k} = \sum\nolimits_{j=1}^{n} b_{k,j} = 1, \forall k \in [n]\right\}.$$

By [6] we have

$$\mathcal{A}_n\left(t\right) = L^{\circ}_{A^{\circ},\mathcal{H}}\left(t\right) \tag{6.6}$$

where

$$A = B_n \cap \left\{(b_{i,j}) \in \mathbb{R}^{n\times n} : \sum\nolimits_{k=1}^{n} b_{k,k} = \sum\nolimits_{k=1}^{n} b_{k,n+1-k} = 1\right\}. \tag{6.7}$$

Though we've only mentioned magic squares here, it is easy to see that similar constructions apply to semi-magic squares by simply omitting the diagonal-sum requirements.

Beck and Zaslavsky have published affine and cubical counts of magic and semi-magic 3-squares in [6]. We have verified these using `ioehrhart` directly.

Example 38 *Cubical and affine counts of magic 3-squares. Let $\mathcal{C}_3\left(t\right)$ denote the number of magic 3-squares with distinct entries from $[t-1] := \{1,\ldots,t-1\}$, and let $\mathcal{A}_n\left(t\right)$ be the number of magic 3-squares with distinct entries from $\mathbb{Z}_{>0}$ and line/*

column/diagonal sum equal to t. In both cases the associated inside-out polytopes have 16 *regions. The rational generating functions for the power series* $\sum_{t\geq 0}\mathcal{C}_3\left(t\right)z^t$ *and* $\sum_{t\geq 0}\mathcal{A}_3\left(t\right)z^t$ *are*

$$\begin{aligned}\sum_{t\geq 0}\mathcal{C}_3\left(t\right)z^t &= 8z^{10}\frac{2z^2+1}{\left(z^2+z+1\right)\left(z^2-z+1\right)\left(z^2+1\right)\left(z+1\right)^2\left(z-1\right)^4}\\ \sum_{t\geq 0}\mathcal{A}_3\left(t\right)z^t &= -8z^{15}\frac{2z^3+1}{\left(z^6+z^3+1\right)\left(z^2-z+1\right)\left(z+1\right)\left(z-1\right)^3\left(z^2+z+1\right)^3}\end{aligned}$$

Example 39 *Cubical and affine counts of semi-magic* 3*-squares. Let* $\mathcal{C}_3\left(t\right)$ *denote the number of semi-magic* 3*-squares with distinct entries from* $\left[t-1\right]$, *and let* $\mathcal{A}_n\left(t\right)$ *be the number of semi-magic* 3*-squares with distinct entries from* $\mathbb{Z}_{>0}$ *and line/column sum equal to t. In both cases the associated inside-out polytopes have* 1296 *regions. The rational generating functions for the power series* $\sum_{t\geq 0}\mathcal{C}_3\left(t\right)z^t$ *and* $\sum_{t\geq 0}\mathcal{A}_3\left(t\right)z^t$ *are*

$$\sum_{t\geq 0}\mathcal{C}_3\left(t\right)z^t = 72z^{10}\frac{\begin{array}{c}18z^9+46z^8+69z^7+74z^6+65z^5+46z^4\\ +26z^3+11z^2+4z+1\end{array}}{\begin{array}{c}\left(z^4+z^3+z^2+z+1\right)\left(z^2+1\right)\left(z^2+z+1\right)\\ {}^2\left(z+1\right)^3\left(z-1\right)^6\end{array}} \quad \textit{and}$$

$$\sum_{t\geq 0}\mathcal{A}_3\left(t\right)z^t=-72z^{15}\frac{\begin{array}{c}18z^{19}+41z^{18}+79z^{17}+117z^{16}+166z^{15}+207z^{14}\\ +249z^{13}+268z^{12}+274z^{11}+258z^{10}+233z^{9}\\ +192z^{8}+152z^{7}+109z^{6}+73z^{5}+44z^{4}+24z^{3}\\ +11z^{2}+4z+1\end{array}}{\begin{array}{c}\left(z^6+z^4+z^5+z^3+z^2+z+1\right)\left(z^4+z^3+z^2+z+1\right)\\ \left(z^4+1\right)\left(z^2+z+1\right)^3\left(z^2-z+1\right)\left(z^2+1\right)^2\\ \left(z+1\right)^3\left(z-1\right)^5\end{array}}.$$

Example 40 *Cubical and affine counts of magic* 4*-squares.*

Computing the number of magic and semi-magic 4-squares using the method described above turns out to be much more difficult than the previous problem. In the ambient vector space $\mathbb{R}^{16}$ the arrangements of the inside-out polytopes associated with the affine and cubical count problems have 16! regions. Fortunately the dimension of the affine hull of the respective problems helps to cut the number of regions. Even so, the problems are still too large to be handled directly in a reasonable amount of time. We now describe a series of reductions of the problem size.

Given a set $\mathcal{M}$ of matrices, two matrices can be considered equivalent if one is obtainable from the other by a permutation of the entries that leaves the defining properties of $\mathcal{M}$ untouched. The dihedral symmetries of the square form an often cited example of this when considering magic squares. One can, for example, assume that the upper left corner contains the largest entry of all four corners and that the upper right entry is larger than the lower left. For each magic square X satisfying these requirements, one recognizes that there exist 8 others that can be obtained from X by simple rotation or reflection of X. For magic squares larger than 3×3 the dihedral symmetries are not the only ones that exist. We would like a general method for exploiting a set of known symmetries in order to simplify the problems involving counting magic squares. Here S_k denotes the symmetric group on k letters. We define the group action $\bullet : S_{n^2} \times \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$ by $\sigma \bullet (x_{i,j}) \mapsto \left(x_{\sigma(i,j)}\right)$.

Proposition 41 *Let $\mathcal{M}$ denote a set of $n \times n$ matrices. If*

$$G := \{\sigma \in S_{n^2} : \sigma \bullet X \in \mathcal{M} \textit{ for all } X \in \mathcal{M}\}.$$

Then G is a a subgroup of S_{n^2} and for any subgroup H of G, H acts on $\mathcal{M}$ by $\bullet$.

Proof. G clearly contains the identity. Closure follows directly from the definition of G. For all $\sigma \in G$ we have $\sigma^{-1} = \sigma^{n^2-1}$; hence $\sigma^{-1} \in G$ by the closure property. Thus G is a subgroup of S_{n^2}. If H is any subgroup of G, a straightforward argument shows that the map $\sigma \bullet (a_{ij}) \longmapsto \left(a_{\sigma(i,j)}\right)$ satisfies the axioms for a group action. ■

If $\mathcal{M}$ is the set of all (semi) magic squares, we call any member of the group G defined above a (*semi-*) *magic n-symmetry.* Recognizing a set of magic symmetries is often relatively easy. Given a set S of magic symmetries we would like to use S to generate a set of equivalence classes of equal cardinality. This way we can always count magic squares by counting the number of equivalence classes associated with S. Together with Proposition 41 the following result shows we may accomplish this goal using the sub-group (S) generated by S together with its associated orbits under the group action $\bullet$.

Proposition 42 *Let G be a subgroup of S_n and let $\mathcal{M}$ be a set of $n \times n$ matrices with distinct entries. If G acts on $\mathcal{M}$ by permutation of the tupple indices, then every orbit of G in $\mathcal{M}$ has cardinality $\#H$.*

Proof. Let O be the orbit of G in $\mathcal{M}$. A straight-forward argument shows O has at most $\#H$ members. To see that O has at least $\#H$ members, let $(a_{ij}) \in O$. Any $\sigma, \tau \in H$ satisfying $\sigma \bullet (a_{ij}) = \tau \bullet (a_{ij})$ by definition satisfies $a_{\sigma(i,j)} = a_{\tau(i,j)}$ for all $i, j \in [n]$. Because the entries of (a_{ij}) are assumed distinct, we have $\sigma(i,j) = \tau(i,j)$ for all $i, j \in [n]$, whence $\sigma = \tau$. By contraposition, $\sigma \neq \tau$ implies $\sigma \bullet (a_{ij}) \neq \tau \bullet (a_{ij})$ for all $(a_{ij}) \in O$, i.e. distinct permutations describe distinct members of O. ■

Given a group G of magic n-symmetries and a set $\mathcal{M}$ upon which G acts, the challenge of finding $\#\mathcal{M}$ reduces to the problem of counting the orbits of G in $\mathcal{M}$. We may accomplish this by counting a set of representatives of each orbit. More specifically we fix a partial ordering on the matrix entries, which ensures that the matrices satisfying the partial ordering are in bijection with the orbits of G. (Exactly what this means will become more apparent as we deal specifically with magic 4-squares below.)

Proposition 43 *Let G be a group of (semi-) magic n-symmetries. For any $t \in \mathbb{Z}^+$, G acts on the set of (semi-) magic n-squares with positive integer entries and magic sum t. Likewise, G acts on the set of magic n-squares with entries from $[t]$.*

Proof. The (semi-) magic sum is invariant under permutation by (semi-) magic symmetries, as is the set of matrix entries. ■

With this in mind, we now apply the above results to aid our computations for counting magic 4-squares. As above we assume that the entries of every magic 4-square X are index by

$$X := \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{pmatrix}. \tag{6.8}$$

It is easy to verify that the following set S represents a set of magic symmetries:

$$S := \left\{ \begin{array}{ll} \sigma_1 = & (0\ 6)\,(2\ 5)\,(3\ 8)\,(4\ 7)\,(9\ 14)\,(10\ 13)\,(11\ 16)\,(12\ 15) \\ \\ \sigma_2 = & (1\ \ 4\ \ 16\ \ 13)\,(2\ \ 8\ \ 15\ \ 9)\,(3\ \ 12\ \ 14\ \ 5)\,(6\ 7\ 11\ 10) \\ \\ \sigma_3 = & (2\ \ 5)\,(3\ \ 9)\,(4\ \ 13)\,(7\ \ 10)\,(4\ \ 13)\,(8\ \ 14)\,(12\ \ 15) \\ \\ \sigma_4 = & (2\ \ 3)\,(5\ \ 9)\,(6\ \ 11)\,(7\ \ 10)\,(8\ \ 12)\,(14\ \ 15) \end{array} \right\}. \tag{6.9}$$

The permutations σ_2 and σ_3 listed above represent the generators for the usual dihedral symmetries associated with magic 4-squares. We list the 32 members of the subgroup (S) generated by S, computed using Maple's group package, as an aid for the following proofs:

$$\begin{array}{rl} \tau_0 = & id. \\ \tau_1 = & (1\ 6)\,(2\ 5)\,(3\ 8)\,(4\ 7)\,(9\ 14)\,(10\ 13)\,(11\ 16)\,(12\ 15) \\ \tau_2 = & (2\ 3)\,(5\ 9)\,(6\ 11)\,(7\ 10)\,(8\ 12)\,(14\ 15) \\ \tau_3 = & (1\ 4)\,(5\ 12)\,(6\ 10)\,(7\ 11)\,(8\ 9)\,(13\ 16) \\ \tau_4 = & (1\ 10\ 16\ 7)\,(2\ 12\ 15\ 5)\,(3\ 9\ 14\ 8)\,(4\ 11\ 13\ 6) \\ \tau_5 = & (1\ 16)\,(2\ 8)\,(3\ 12)\,(5\ 14)\,(7\ 10)\,(9\ 15) \\ \tau_6 = & (1\ 6)\,(3\ 14)\,(4\ 10)\,(7\ 13)\,(8\ 9)\,(11\ 16) \\ \tau_7 = & (1\ 6\ 16\ 11)\,(2\ 8\ 15\ 9)\,(3\ 5\ 14\ 12)\,(4\ 7\ 13\ 10) \\ \tau_8 = & (1\ 11\ 16\ 6)\,(2\ 9\ 15\ 8)\,(3\ 12\ 14\ 5)\,(4\ 10\ 13\ 7) \\ \tau_9 = & (1\ 7\ 16\ 10)\,(2\ 3\ 15\ 14)\,(4\ 11\ 13\ 6)\,(5\ 8\ 12\ 9) \\ \tau_{10} = & (1\ 10)\,(3\ 14)\,(4\ 6)\,(5\ 12)\,(7\ 16)\,(11\ 13) \\ \tau_{11} = & (2\ 9)\,(3\ 5)\,(4\ 13)\,(6\ 11)\,(8\ 15)\,(12\ 14) \\ \tau_{12} = & (1\ 7\ 16\ 10)\,(2\ 5\ 15\ 12)\,(3\ 8\ 14\ 9)\,(4\ 6\ 13\ 11) \\ \tau_{13} = & (1\ 4\ 16\ 13)\,(2\ 8\ 15\ 9)\,(3\ 12\ 14\ 5)\,(6\ 7\ 11\ 10) \\ \tau_{14} = & (1\ 4)\,(2\ 3)\,(5\ 8)\,(6\ 7)\,(9\ 12)\,(10\ 11)\,(13\ 16)\,(14\ 15) \\ \tau_{15} = & (1\ 10\ 16\ 7)\,(2\ 14\ 15\ 3)\,(4\ 6\ 13\ 11)\,(5\ 9\ 12\ 8) \end{array}$$

$$\begin{aligned}
\tau_{16} = {} & (1\ 11)\,(2\ 12)\,(3\ 9)\,(4\ 10)\,(5\ 15)\,(6\ 16)\,(7\ 13)\,(8\ 14) \\
\tau_{17} = {} & (1\ 10)\,(2\ 9)\,(3\ 12)\,(4\ 11)\,(5\ 14)\,(6\ 13)\,(7\ 16)\,(8\ 15) \\
\tau_{18} = {} & (1\ 7)\,(2\ 8)\,(3\ 5)\,(4\ 6)\,(9\ 15)\,(10\ 16)\,(11\ 13)\,(12\ 14) \\
\tau_{19} = {} & (1\ 13\ 16\ 4)\,(2\ 5\ 15\ 12)\,(3\ 9\ 14\ 8)\,(6\ 7\ 11\ 10) \\
\tau_{20} = {} & (1\ 16)\,(2\ 12)\,(3\ 8)\,(5\ 15)\,(6\ 11)\,(9\ 14) \\
\tau_{21} = {} & (1\ 11)\,(2\ 15)\,(4\ 7)\,(5\ 12)\,(6\ 16)\,(10\ 13) \\
\tau_{22} = {} & (1\ 16)\,(2\ 14)\,(3\ 15)\,(4\ 13)\,(5\ 8)\,(9\ 12) \\
\tau_{23} = {} & (1\ 7)\,(2\ 15)\,(4\ 11)\,(6\ 13)\,(8\ 9)\,(10\ 16) \\
\tau_{24} = {} & (1\ 11\ 16\ 6)\,(2\ 3\ 15\ 14)\,(4\ 7\ 13\ 10)\,(5\ 9\ 12\ 8) \\
\tau_{25} = {} & (1\ 16)\,(2\ 15)\,(3\ 14)\,(4\ 13)\,(5\ 12)\,(6\ 11)\,(7\ 10)\,(8\ 9) \\
\tau_{26} = {} & (1\ 13\ 16\ 4)\,(2\ 9\ 15\ 8)\,(3\ 5\ 14\ 12)\,(6\ 10\ 11\ 7) \\
\tau_{27} = {} & (1\ 13)\,(2\ 15)\,(3\ 14)\,(4\ 16)\,(6\ 7)\,(10\ 11) \\
\tau_{28} = {} & (2\ 5)\,(3\ 9)\,(4\ 13)\,(7\ 10)\,(8\ 14)\,(12\ 15) \\
\tau_{29} = {} & (1\ 13)\,(2\ 14)\,(3\ 15)\,(4\ 16)\,(5\ 9)\,(6\ 10)\,(7\ 11)\,(8\ 12) \\
\tau_{30} = {} & (1\ 6\ 16\ 11)\,(2\ 14\ 15\ 3)\,(4\ 10\ 13\ 7)\,(5\ 8\ 12\ 9) \\
\tau_{31} = {} & (1\ 4\ 16\ 13)\,(2\ 12\ 15\ 5)\,(3\ 8\ 14\ 9)\,(6\ 10\ 11\ 7)
\end{aligned}$$

Proposition 44 *If $\mathcal{M}$ is a set of magic 4-squares such that $\langle S\rangle$ acts on $\mathcal{M}$ by permutation of the entries indexed by* (6.8)*, then each of the sets,*

$$\mathcal{M}' \ : \ = \left\{ X \in \mathcal{M} : \begin{array}{c} x_1 \textit{ is the smallest diagonal entry,} \\ x_4 < x_{13}, \textit{ and } x_6 < x_{11} \end{array} \right\} \textit{ and} \tag{6.10}$$

$$\mathcal{M}'' \ : \ = \left\{ X \in \mathcal{M} : \begin{array}{c} x_2 \textit{ is the smallest off-diagonal entry,} \\ x_3 < x_{14}, \textit{ and } x_5 < x_{12} \end{array} \right\}, \tag{6.11}$$

are in bijection with the orbits of G in $\mathcal{M}$.

Proof. Let ϕ be the map that takes each member of $\mathcal{M}'$ to its corresponding orbit. By enumerating the members of $\langle S\rangle$ one can verify that $\sigma(X) \notin \mathcal{M}'$ for every $X \in \mathcal{M}'$. In particular no two members of $\mathcal{M}'$ share the same orbit, implying ϕ

is injective. Now let O be an orbit of (S) in $\mathcal{M}$ and choose a representative M' of O with entries $m_1, \ldots, m_{16}$. By applying σ_1, if necessary, we may assume that the smallest diagonal entry of M' occurs at one of the four corners. By applying σ_2 (right rotation) enough times we may assume that m_1 is the smallest diagonal entry of M'. By applying σ_3 we may assume that $m_4 < m_{13}$. The application of σ_4 leaves the corners untouched. In particular σ_4 leaves x_1, x_4, and x_{13} untouched, so we may if necessary apply σ_4 to ensure that $x_6 < x_{11}$. This proves that ϕ is surjective.

A similar argument can be made for $\mathcal{M}''$. The only issue is showing that every orbit of (S) in $\mathcal{M}$ has a representative in $\mathcal{M}''$. Using the dihedral symmetries we may assume that every orbit has a representative with x_2 as the smallest off-diagonal entry. Now if $x_3 > x_{14}$ we may apply τ_6, which exchanges x_3 and x_{14} without changing the position of x_2. If $x_5 > x_{12}$ we may apply τ_3, which exchanges x_5 and x_{12} without changing the positions of x_2, x_3 or x_{14}. Hence every orbit has a representative in $\mathcal{M}''$.

■

Proposition 45 *Let $\mathcal{M}$ be a set of magic 4-squares such that (S) acts on $\mathcal{M}$ by permutation of the matrix entries indexed by* (6.8). *If*

$$\begin{aligned}\mathcal{P} \;&:\; = \{X \in \mathcal{M} : \min(x_1, \ldots, x_{16}) \text{ is on the diagonal}\}, \text{ and}\\ \mathcal{Q} \;&:\; = \{X \in \mathcal{M} : \min(x_1, \ldots, x_{16}) \text{ is off the diagonal}\},\end{aligned}$$

then G acts similarly on $\mathcal{P}$ and $\mathcal{Q}$, and

$$\#\mathcal{M} = 32\left(\#\mathcal{P}' + \#\mathcal{Q}''\right), \tag{6.12}$$

where $\mathcal{P}'$ and $\mathcal{Q}^{''}$ are defined by (6.10) *and* (6.11) , *respectively.*

Proof. Because $\mathcal{P}, \mathcal{Q} \subseteq \mathcal{M}$, in order to show G acts on $\mathcal{P}$ and $\mathcal{Q}$, it suffices to show that $\sigma(X) \in \mathcal{P}$ and $\sigma(Y) \in \mathcal{Q}$ for all $X \in \mathcal{P}$, all $Y \in \mathcal{Q}$, and all $\sigma \in (S)$. This follows from the observation that every $\sigma \in S$ maps diagonals to diagonals and off-diagonals to off-diagonals. Using the fact $\mathcal{M} = \mathcal{P} \cup \mathcal{Q}$ is a disjoint union, (6.12) follows from Propositions 44 and 44. ∎

Corollary 46 *Let $\mathcal{M}$ be a set of magic 4 squares such that (S) acts on $\mathcal{M}$ by permutation of the matrix entries. Then*

$$\#\mathcal{M} = 32\left(\#\mathcal{P} + \#\mathcal{Q}\right),$$

where

$$\mathcal{P} \; : \; = \left\{ X \in \mathcal{M} : \begin{array}{c} x_1 < x_2, x_3, \ldots, x_{16}, \\ x_4 < x_{13}, \text{ and } x_6 < x_{11} \end{array} \right\} \text{ and} \tag{6.13}$$

$$\mathcal{Q} \; : \; = \left\{ X \in \mathcal{M} : \begin{array}{c} x_2 < x_1, x_3, x_{4,} \ldots, x_{16}, \\ x_3 < x_{14}, \text{ and } x_5 < x_{12} \end{array} \right\}. \tag{6.14}$$

Let $\mathcal{C}_4\left(t\right)$ denote the number of magic 4-squares with distinct entries from $\left[t-1\right]$, and let $\mathcal{A}_4\left(t\right)$ be the number of magic 4-squares with distinct entries from $\mathbb{Z}_{>0}$ and line/column sum equal to t. The last corollary together with Proposition 42 implies that

$$\sum_{t\geq 0}\mathcal{C}_3\left(t\right)z^t = 32\left(\mathrm{Ehr}^{\circ}_{\left(P_C^{\circ},\mathcal{H}\right)}\left(z\right)+\mathrm{Ehr}^{\circ}_{\left(Q_C^{\circ},\mathcal{H}\right)}\left(z\right)\right), \tag{6.15}$$

where

$$\begin{aligned} P_C &= \left\{ [x_{i,j}]\in\mathbb{R}^{4\times 4} : \begin{array}{c} x_{1,4}+x_{2,3}+x_{3,2}+x_{4,1}=\sum_{i=1}^4 x_{i,i} \\ =\sum_{i=1}^4 x_{i,j}=\sum_{i=1}^4 x_{i,k}=\sum_{i=1}^4 x_{k,i}, \\ x_{1,1}\leq x_{j,k},\ x_{1,4}\leq x_{4,1},\ x_{2,2}\leq x_{3,3} \\ x_{i,j}\leq 1\ \forall j,k=1,\ldots,4 \end{array} \right\} \text{ and} \\ Q_C &= \left\{ [x_{i,j}]\in\mathbb{R}^{4\times 4} : \begin{array}{c} x_{1,4}+x_{2,3}+x_{3,2}+x_{4,1}=\sum_{i=1}^4 x_{i,i} \\ =\sum_{i=1}^4 x_{i,j}=\sum_{i=1}^4 x_{i,k}=\sum_{i=1}^4 x_{k,i}, \\ x_{1,2}\leq x_{j,k},\ x_{1,3}\leq x_{4,2},\ x_{2,1}\leq x_{3,4} \\ x_{i,j}\leq 1\ \forall j,k=1,\ldots,4 \end{array} \right\}. \end{aligned}$$

Applying the application `ioregioncount` (cf. Section 5.4) to the inside-out polytopes $(P_C,\mathcal{H})$ and $(Q_C,\mathcal{H})$ shows that

$$\begin{aligned} \#\text{regions}\left((P_C,\mathcal{H})\right) &= 1,147,814 \text{ and} \\ \#\text{regions}\left((Q_C,\mathcal{H})\right) &= 2,063,598. \end{aligned}$$

With the gracious help of the Center for Computing in Life Sciences (CCLS) at San Francisco State University, we have successfully computed simplified forms of the rational generating functions in (6.15) using their high performance cluster. Computation time for this particular problem was between 8-10 hours during a time of minimal cluster traffic. In a simplified form (reduced to lowest terms with expanded numerators and denominators) $\mathrm{Ehr}^{\circ}_{\left(P^{\circ}_{C},\mathcal{H}\right)}(z)$, $\mathrm{Ehr}^{\circ}_{\left(Q^{\circ}_{C},\mathcal{H}\right)}(z)$, as well as $\sum_{t\geq 0}\mathcal{C}_3(t)\,z^t$, each requires between 20 and 30 kilobytes of disk space. We include them on the compact disk that accompanies this work. A printed version of $\sum_{t\geq 0}\mathcal{C}_3(t)\,z^t$ can be found in the appendices. Using Maple to compute a Taylor expansion at 0, we give the first 20 non-zero terms of the power series:

$$\begin{aligned}\sum_{t\geq 0}\mathcal{C}_4(t)\,z^t &= 7040z^{17}+25792z^{18}+90176z^{19}+243136z^{20}+657600z^{21}\\ &\quad +1563392z^{22}+3565248z^{23}+7597376z^{24}+15347840z^{25}\\ &\quad +29272064z^{26}+53575296z^{27}+94380032z^{28}+160733632z^{29}\\ &\quad +265288128z^{30}+426468864z^{31}+668000576z^{32}\\ &\quad +1023828864z^{33}+1536693504z^{34}+2264122816z^{35}\\ &\quad +3278733632z^{36}+O(z^{37})\end{aligned}$$

A similar problem reduction may be used for the affine count. In this case we have

$$\sum_{t\geq 0} \mathcal{A}_4 (t)\, z^t = 32 \left(\mathrm{Ehr}^{\circ}_{\left(P_A^{\circ},\mathcal{H}\right)} (z) + \mathrm{Ehr}^{\circ}_{\left(Q_A^{\circ},\mathcal{H}\right)} (z) \right) \tag{6.16}$$

where

$$\begin{aligned} P_A &= \left\{ [x_{i,j}] \in \mathbb{R}^{4\times 4} : \begin{array}{c} x_{1,4} + x_{2,3} + x_{3,2} + x_{4,1} = \sum_{i=1}^4 x_{i,i} \\ = \sum_{i=1}^4 x_{i,j} = \sum_{i=1}^4 x_{j,i} = 1, \\ x_{1,1} \leq x_{j,k},\ x_{1,4} \leq x_{4,1},\ x_{2,2} \leq x_{3,3} \\ x_{i,j} \leq 1\ \forall j, k = 1, \ldots, 4 \end{array} \right\} \text{ and} \\ Q_A &= \left\{ [x_{i,j}] \in \mathbb{R}^{4\times 4} : \begin{array}{c} x_{1,4} + x_{2,3} + x_{3,2} + x_{4,1} = \sum_{i=1}^4 x_{i,i} \\ = \sum_{i=1}^4 x_{i,j} = \sum_{i=1}^4 x_{j,i} = 1, \\ x_{1,2} \leq x_{j,k},\ x_{1,3} \leq x_{4,2},\ x_{2,1} \leq x_{3,4} \\ x_{i,j} \leq 1\ \forall j, k = 1, \ldots, 4 \end{array} \right\}. \end{aligned}$$

The number of regions of the associated inside-out polytopes is the same as those found for the cubical count, namely

$$\begin{aligned} \#\text{regions}\left((P_A, \mathcal{H}) \right) &= 1,147,814, \text{ and} \\ \#\text{regions}\left((Q_A, \mathcal{H}) \right) &= 2,063,598. \end{aligned}$$

The simplified generating functions for $\mathrm{Ehr}^{\circ}_{\left(P_A^{\circ},\mathcal{H}\right)} (z)$, $\mathrm{Ehr}^{\circ}_{\left(Q_A^{\circ},\mathcal{H}\right)} (z)$ and $\sum_{t\geq 0} \mathcal{A}_3 (t)\, z^t$ each require between 200 and 300 kilobytes of disk space to store. They involve coefficients larger than 10^{58}. To print them in this work would require over 50 pages apiece. We include them on the compact disk that accompanies this work. In lieu of a printed version of the rational functions, we give the first 20 non-zero terms of the power series:

$$\begin{aligned}\sum_{t\geq 0}\mathcal{A}_4\left(t\right)z^t &= 7040z^{34}+4000z^{35}+22592z^{36}+24768z^{37}+85568z^{38}\\ &\quad +91008z^{39}+198464z^{40}+266784z^{41}+560640z^{42}\\ &\quad +654912z^{43}+1113696z^{44}+1450912z^{45}+2521440z^{46}\\ &\quad +2956736z^{47}+4543392z^{48}+5635744z^{49}+8701792z^{50}\\ &\quad +10234848z^{51}+14693504z^{52}+17695168z^{53}+O(z^{54}).\end{aligned}$$

Proposition 47 *The maximal group of magic* 4*-symmetries is the subgroup of* S_{16} *generated by* S *given in* (6.9)*.*

Proof. By Proposition 42 the order of a group of magic symmetries must divide the cardinality of any set of magic squares upon which it acts. The result follows from Proposition 43 and the fact that the first three non-zero coefficients 7040, 4000, and 22592 from $\sum_{t\geq 0}\mathcal{A}_4\left(t\right)z^t$ above have a greatest common divisor of 32. ■

Example 48 *Semi-magic* 4*-squares by cubical and affine count.*

Despite the fact that semi-magic squares have more symmetries (the group of semi-magic n-symmetries contains the group of magic n-symmetries) than magic squares, even a reduced form of the problem exploiting these symmetries is significantly more difficult to solve using Algorithm 17 than the magic 4-squares. Applying the same reasoning used for the reduction of the magic 4-squares above, it is not difficult to show that one can obtain a group of semi-magic symmetries isomorphic to $S_4 \times S_4 \times \mathbb{Z}_2$. This stems from the observation that we may permute rows and/or columns without affecting the semi-magic nature of a matrix. It is possible to construct an ordering of the entries (see the one used below) that prohibits permutation of rows or columns without violating the order but allows a flip across the diagonal. If $\mathcal{C}_4(t)$ denotes the number of semi-magic 4-squares with distinct entries from $[t-1]$, and $\mathcal{A}_4(t)$, the number of semi-magic 4-squares with distinct entries from $\mathbb{Z}_{>0}$ and line/column sum equal to t, then Propositions 42 and 43 imply that

$$\sum_{t\geq 0} \mathcal{C}_4(t)\, z^t = 1152 \ \mathrm{Ehr}^{\circ}_{\left(P_C^{\circ},\mathcal{H}\right)}(z) \ \text{ and } \sum_{t\geq 0} \mathcal{A}_4(t)\, z^t = 1152 \ \mathrm{Ehr}^{\circ}_{\left(P_A^{\circ},\mathcal{H}\right)}(z)$$

(1152 is the cardinality of $S_4 \times S_4 \times \mathbb{Z}_2$), where

$$P_C = \left\{ [x_{i,j}] \in \mathbb{R}^{4\times 4} : \begin{array}{c} \sum_{i=1}^{4} x_{i,j} = \sum_{i=1}^{4} x_{k,i}, 0 \leq x_{i,j} \leq 1 \\ x_{i,j} \leq 1, x_{1,1} \leq x_{j,k}, \ \forall j,k = 1,2,3,4 \\ x_{2,2} \leq x_{j,k} \forall j,k = 2,3,4 \\ x_{3,3} \leq x_{j,k} \forall j,k = 3,4 \\ x_{1,4} \leq x_{4,1} \end{array} \right\},$$

$$P_A = \left\{ [x_{i,j}] \in \mathbb{R}^{4\times 4} : \begin{array}{c} \sum_{i=1}^{4} x_{i,j} = \sum_{i=1}^{4} x_{k,i} = 1, x_{i,j} \geq 0 \\ x_{1,1} \leq x_{j,k}, \ \forall j,k = 1,2,3,4 \\ x_{2,2} \leq x_{j,k} \forall j,k = 2,3,4 \\ x_{3,3} \leq x_{j,k} \forall j,k = 3,4 \\ x_{1,4} \leq x_{4,1} \end{array} \right\},$$

and $\mathcal{H}$ is the braid arrangement in $\mathbb{R}^{4\times 4}$. The number of regions of each inside-out polytope as computed by `ioregioncount` (cf. Section 5.4) is

$$\#\text{regions}\left(\left(P_C^\circ, \mathcal{H}\right)\right) = \#\text{regions}\left(\left(P_A^\circ, \mathcal{H}\right)\right) = 16,972,282.$$

Using the high performance cluster at the Center for Computing in Life Sciences we have obtained a generating function for $\sum_{t\geq 0} \mathcal{C}_3\left(t\right) z^t$. A printed version of can be found in the appendices. The rational function $\mathrm{Ehr}^\circ_{\left(P_C^\circ, \mathcal{H}\right)}\left(z\right)$ can be found on the compact disk that accompanies this work. We give the first 20 non-zero terms of the

Taylor expansion at 0 as computed by Maple:

$$\begin{aligned}\sum_{t\geq 0}\mathcal{C}_4\left(t\right)z^t &= 549504z^{17}+2001024z^{18}+8683776z^{19}+29227392z^{20}\\ &+95084928z^{21}+271008000z^{22}+718776576z^{23}\\ &+1758727296z^{24}+4034877696z^{25}+8706097152z^{26}\\ &+17860399488z^{27}+35021365632z^{28}+65984512896z^{29}\\ &+119952919296z^{30}+211193611776z^{31}+361274832000z^{32}\\ &+60198963494z^{33}+979683345792z^{34}+1559749594752z^{35}\\ &+2434741618176z^{36}+O(z^{37})\end{aligned}$$

6.3 Magilatin and Magisudoku squares

A *latin square* is an $n \times n$ integer matrix in which the entries within each row and within each column are distinct. With *traditional Latin squares* the entries come from the first n positive integers, in which case the sum of entries from each row and each column is always the same. We call a Latin square *magilatin* if every row and every column have the same sum. A *sudoku n-square* is an $n^2 \times n^2$ latin square of the form

$$\begin{pmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n} \end{pmatrix} \tag{6.17}$$

where each $A_{i,j}$ is an $n \times n$ matrix with distinct entries. As with $n^2 \times n^2$ latin squares, *traditional sudoku squares* have entries from the first n^4 positive integers. A *magisudoku square* is a sudoku square in which the sum of entries from every row, every column, and every submatrix $A_{i,j}$ is the same. Traditional latin and sudoku squares are special cases of magilatin and magisudoku squares.

Latin, magilatin, and magisudoku squares to not lend themselves well to the same type of problem reductions as used with the magic and semi-magic squares. The problem here is that Proposition 42 fails because of the lack of distinct matrix entries. Though we have succeeded in producing region counts for the following examples, due to the high number of regions involved we have not been able to compute generating functions.

Example 49 *Counting magilatin* 4*-squares, affine and cubical counts.*

The inside-out polytopes $(P_c, \mathcal{H})$ and $(P_A, \mathcal{H})$ used to compute the cubical and affine counts, respectively, differ from those used for the semi-magic squares only in the arrangement. In the arrangement for magilatin squares each hyperplane corresponds to an equation $x = y$ for each pair of entries x, y sharing the same row or column of the matrix. The number of regions of $(P_A, \mathcal{H})$ is $162,437,760$.

Example 50 *Counting magisudoku* 2*-squares, affine and cubical counts.*

The number of magisudoku 2-squares with an upper bound t on the entries is expressed by the function $L^{\cdot}_{\left(P_C^{\circ},\mathcal{H}\right)}(t+1)$ where P_C is the polytope defined by the requirement that all matrix entries be weakly between 0 and 1, and that the entries of every row, every column, and every submatrix $A_{i,j}$ $(i, j \in \{1, 2\})$ corresponding to (6.17) sum to to the same number. The number of magisudoku 2-squares with line sum t can be obtained from the function $L^{\cdot}_{\left(P_A^{\circ},\mathcal{H}\right)}(t)$ where P_A is the polytope defined by the requirement that all matrix entries be non-negative, and that the entries of every row, every column, and every submatrix $A_{i,j}$ $(i, j \in \{1, 2\})$ corresponding to (6.17) sum to 1. The hyperplanes in the arrangement correspond to pair-wise equality between components within the same row, column or submatrix $A_{i,j}$. The number of regions of $(P_A, \mathcal{H})$ is $66,915,808$.

BIBLIOGRAPHY

[1] Swox AB, *The GNU Multiple Precision Arithmetic Library, Edition 4.2.1*, Electronically available at `http://gmplib.org`, May 2006.

[2] Alexander I. Barvinok, *A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed*, Math. Oper. Res. **19** (1994), no. 4, 769–779. MR MR1304623 (96c:52026)

[3] C. Batut, K. Belabas, D. Bernardi, H. Cojen, and M. Olivier, *User's Guide to PARI/GP (Version 2.3.2)*, Electronically available at `http://pari.math.u-bordeaux.fr`, March 2006.

[4] Matthias Beck, *Multidimensional Ehrhart reciprocity*, J. Combin. Theory Ser. A **97** (2002), no. 1, 187–194. MR MR1879135 (2002j:52013)

[5] Matthias Beck and Sinai Robins, *Computing the continuous discretely*, Undergraduate Texts in Mathematics, Springer, New York, 2007, Integer-point enumeration in polyhedra. MR MR2271992 (2007h:11119)

[6] Matthias Beck and Thomas Zaslavsky, *An enumerative geometry for magic and magilatin labellings*, Ann. Comb. **10** (2006), no. 4, 395–413. MR MR2293647

[7] ——, *Inside-out polytopes*, Adv. Math. **205** (2006), no. 1, 134–162. MR MR2254310 (2007e:52017)

[8] ——, *The number of nowhere-zero flows on graphs and signed graphs*, J. Combin. Theory Ser. B **96** (2006), no. 6, 901–918. MR MR2274083

[9] Arne Brøndsted, *An introduction to convex polytopes*, Graduate Texts in Mathematics, vol. 90, Springer-Verlag, New York, 1983. MR MR683612 (84d:52009)

[10] Bill Bryce, Yung-Chin Fang, Saeed Iqbal, and Keith Priddy, *Workload management and job scheduling on platform rock clusters*, Dell Power Solutions, Dell Inc., 2005, pp. 35–38.

[11] Jesús A. De Loera, Hemmecke Raymond Haws, David, Peter Huggins, Jeremiah Tauzer, and Ruriko Yoshida, *A User's Guide for LattE v1.1*, Electronically available at `http://www.math.ucdavis.edu/~latte`, December 2003.

[12] Reinhard Diestel, *Graph theory*, second ed., Graduate Texts in Mathematics, vol. 173, Springer-Verlag, New York, 2000. MR MR1743598

[13] Komei Fukuda, *cddlib Reference Manual (Version 0.94)*, Electronically available at `http://www.cs.mcgill.ca/~fukuda/soft/cdd_home/cdd.html`, February 2007.

[14] P. Hanlon and R. W. Robinson, *Counting bridgeless graphs*, J. Combin. Theory Ser. B **33** (1982), no. 3, 276–305. MR MR693368 (84d:05096)

[15] Martin Kochol, *Polynomials associated with nowhere-zero flows*, J. Combin. Theory Ser. B **84** (2002), no. 2, 260–269. MR MR1889258 (2002k:05193)

[16] Kenneth H. Rosen, *Discrete mathematics and its applications: And its applications*, McGraw-Hill Higher Education, 2006.

[17] Alexander Schrijver, *Theory of linear and integer programming*, Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons Ltd., Chichester, 1986, A Wiley-Interscience Publication. MR MR874114 (88m:90090)

[18] Victor Shoup, *NTL: A Library for doing Number Theory*, Electronically available at `http://www.shoup.net/ntl`, 2006.

[19] The Polylib Team, *Polylib User's Manual*, Electronically available at `http://icps.u-strasbg.fr/PolyLib`, September 2002.

[20] W. T. Tutte, *A ring in graph theory*, Proc. Cambridge Philos. Soc. **43** (1947), 26–40. MR MR0018406 (8,284k)

[21] Sven Verdoolaege, *barvinok: User Guide, (Version 0.23)*, Electronically available at `http://www.kotnet.org/~skimo/barvinok`, April 2007.

[22] Günter M. Ziegler, *Lectures on polytopes*, Graduate Texts in Mathematics, vol. 152, Springer-Verlag, New York, 1995. MR MR1311028 (96a:52011)

Appendix A

Flow Polynomials of Graphs

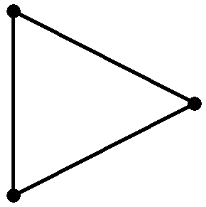
We catalogue some polynomials associated with graphs as described in Chapter 6. As a reminder φ and $\overline{\varphi}$ represent the nowhere-zero t-flow and $\mathbb{Z}_t$-flow polynomials, respectively. The functions φ_0 and $\overline{\varphi}_0$ correspond to the weak (possibly zero) t-flow and $\mathbb{Z}_t$-flow polynomials. We include the Tutte polynomial Φ because of its relationship to many graph invariants.

A Maple script was used to generate non-isomorphic bridgeless connected graphs with 3, 4, 5, and 6 vertices. Maple code for testing graph isomorphism was borrowed from the online companion to [16]. At the time of writing, it is available at

http://www.mhhe.com/math/advmath/rosen/r5/student/maple.html

and is included in the CD accompanying this work, as no reference to a printed version was found. Based on [14] we believe the lists to be exhaustive. Graphs with the same number of vertices are ordered in descending order of the number of edges. The labels for the graphs correspond the iteration at which they were generated.

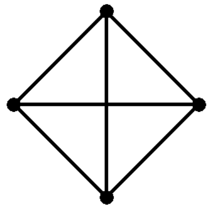
A.1 Simple, connected, bridgeless graphs with 3 vertices.



vert3_i1

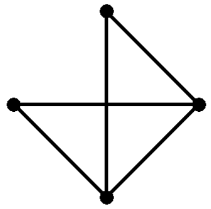
$$\begin{aligned} \varphi(t) &= 2(t-1) \\ \varphi_0(t) &= 2t-1 \\ \overline{\varphi}(t) &= (t-1) \\ \overline{\varphi}_0(t) &= t \\ \Phi(x,y) &= x^2+x+y \end{aligned}$$

A.2 Simple, connected, bridgeless graphs with 4 vertices.



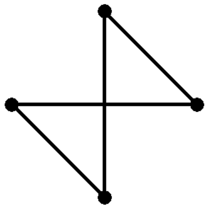
vert4_i1

$$\begin{aligned} \varphi(t) &= 4(t-1)(t-2)(t-3) \\ \varphi_0(t) &= (2t-1)(2t^2-2t+1) \\ \overline{\varphi}(t) &= (t-1)(t-2)(t-3) \\ \overline{\varphi}_0(t) &= t^3 \\ \Phi(x,y) &= x^3+3x^2+4xy+2x+y^3+3y^2+2y \end{aligned}$$



vert4_i2

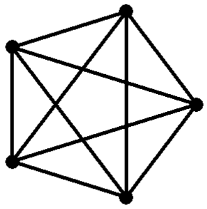
$$\begin{aligned} \varphi(t) &= 3(t-1)(t-2) \\ \varphi_0(t) &= 3t^2-3t+1 \\ \overline{\varphi}(t) &= (t-1)(t-2) \\ \overline{\varphi}_0(t) &= t^2 \\ \Phi(x,y) &= x^3+2x^2+2xy+x+y^2+y \end{aligned}$$



vert4_i3

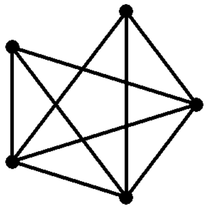
$$\begin{aligned}
\varphi\left(t\right) =& \quad 2\left(t-1\right)\\
\varphi_0\left(t\right) =& \quad 2t-1\\
\overline{\varphi}\left(t\right) =& \quad t-1\\
\overline{\varphi}_0\left(t\right) =& \quad t\\
\Phi\left(x,y\right) =& \quad x^3+x^2+x+y
\end{aligned}$$

A.3 Simple, connected, bridgeless graphs with 5 vertices.



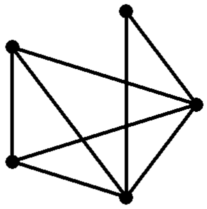
vert5_i1

$$\begin{aligned}
\varphi\left(t\right) =& \quad \tfrac{2}{3}\left(t-1\right)\left(24t^5-195t^4+715t^3-1440t^2\right.\\
& \qquad \left.+1622t-816\right)\\
\varphi_0\left(t\right) =& \quad \tfrac{1}{3}\left(2t-1\right)^2\left(12t^4-24t^3+23t^2-11t+3\right)\\
\overline{\varphi}\left(t\right) =& \quad \left(t-1\right)\left(t^5-9t^4+36t^3-79t^2+96t-51\right)\\
\overline{\varphi}_0\left(t\right) =& \quad t^6\\
\Phi\left(x,y\right) =& \quad x^4+6x^3+10x^2y+11x^2+5xy^3+15xy^2\\
& \qquad +20xy+6x+y^6+4y^5+10y^4+15y^3\\
& \qquad +15y^2+6y
\end{aligned}$$



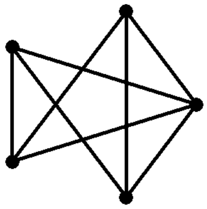
vert5_i2

$$\begin{aligned}
\varphi\left(t\right) =& \quad \tfrac{1}{10}\left(t-1\right)\left(t-2\right)\left(98t^3-541t^2+1161t-1020\right)\\
\varphi_0\left(t\right) =& \quad \tfrac{1}{10}\left(2t-1\right)\left(49t^4-98t^3+91t^2-42t+10\right)\\
\overline{\varphi}\left(t\right) =& \quad \left(t-1\right)\left(t-2\right)\left(t^3-6t^2+14t-13\right)\\
\overline{\varphi}_0\left(t\right) =& \quad t^5\\
\Phi\left(x,y\right) =& \quad x^4+5x^3+7x^2y+8x^2+2xy^3+9xy^2+13xy+4x\\
& \qquad +y^5+4y^4+8y^3+9y^2+4y
\end{aligned}$$



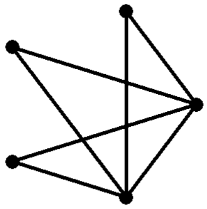
vert5_i3

$$\begin{aligned}
\varphi\left(t\right) =& \quad \tfrac{1}{6}\left(t-1\right)\left(t-2\right)\left(41t^2-151t+180\right)\\
\varphi_0\left(t\right) =& \quad \tfrac{1}{6}\left(\left(41t^4-82t^3+73t^2-32t+6\right)\right)\\
\overline{\varphi}\left(t\right) =& \quad \left(t-1\right)\left(t-2\right)\left(t^2-4t+5\right)\\
\overline{\varphi}_0\left(t\right) =& \quad t^4\\
\Phi\left(x,y\right) =& \quad x^4+4x^3+5x^2y+5x^2+xy^3+5xy^2+7xy+2x\\
& \qquad +y^4+3y^3+4y^2+2y
\end{aligned}$$



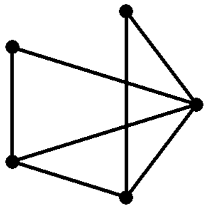
vert5_i4

$$\begin{aligned}
\varphi\left(t\right) =&\quad \tfrac{1}{3}\left(t-1\right)\left(t-2\right)\left(18t^2-86t+117\right)\\
\varphi_0\left(t\right) =&\quad \left(3t^2-3t+1\right)\left(2t^2-2t+1\right)\\
\overline{\varphi}\left(t\right) =&\quad \left(t-1\right)\left(t-2\right)\left(t^2-5t+7\right)\\
\overline{\varphi}_0\left(t\right) =&\quad t^4\\
\Phi\left(x,y\right) =&\quad x^4+4x^3+4x^2y+6x^2+4xy^2+9xy+3x\\
&\quad\qquad +y^4+4y^3+6y^2+3y
\end{aligned}$$



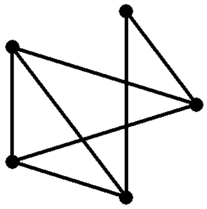
vert5_i5

$$\begin{aligned}
\varphi\left(t\right) =&\quad \tfrac{2}{3}\left(t-1\right)\left(8t^2-22t+21\right)\\
\varphi_0\left(t\right) =&\quad \tfrac{1}{3}\left(2t-1\right)\left(8t^2-8t+3\right)\\
\overline{\varphi}\left(t\right) =&\quad \left(t-1\right)\left(t^2-3t+3\right)\\
\overline{\varphi}_0\left(t\right) =&\quad t^3\\
\Phi\left(x,y\right) =&\quad x^4+3x^3+3x^2y+3x^2+3xy^2+3xy+x\\
&\quad\qquad +y^3+y^2+y
\end{aligned}$$



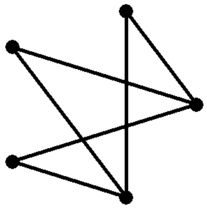
vert5_i6

$$\begin{aligned}
\varphi\left(t\right) =&\quad \tfrac{1}{3}\left(t-1\right)\left(t-2\right)\left(14t-27\right)\\
\varphi_0\left(t\right) =&\quad \tfrac{1}{3}\left(2t-1\right)\left(7t^2-7t+3\right)\\
\overline{\varphi}\left(t\right) =&\quad \left(t-1\right)\left(t-2\right)^2\\
\overline{\varphi}_0\left(t\right) =&\quad t^3\\
\Phi\left(x,y\right) =&\quad t^4+3t^3+3t^2z+3t^2+2tz^2+4tz+t\\
&\quad +z^3+2z^2+z
\end{aligned}$$



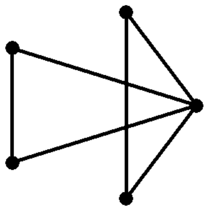
vert5_i7

$$\begin{aligned}
\varphi\left(t\right) =&\quad 4\left(t-1\right)\left(t-2\right)\left(t-3\right)\\
\varphi_0\left(t\right) =&\quad \left(2t-1\right)\left(2t^2-2t+1\right)\\
\overline{\varphi}\left(t\right) =&\quad \left(t-1\right)\left(t-2\right)\left(t-3\right)\\
\overline{\varphi}_0\left(t\right) =&\quad t^3\\
\Phi\left(x,y\right) =&\quad t^4+3t^3+2t^2z+4t^2+tz^2+5tz+2t\\
&\quad\qquad +z^3+3z^2+2z
\end{aligned}$$



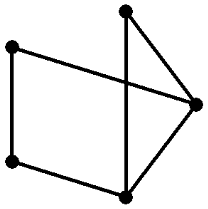
vert5_i8

$$\begin{aligned}
\varphi\left(t\right) =&\quad 3\left(t-1\right)\left(t-2\right)\\
\varphi_0\left(t\right) =&\quad 3t^2-3t+1\\
\overline{\varphi}\left(t\right) =&\quad \left(t-1\right)\left(t-2\right)\\
\overline{\varphi}_0\left(t\right) =&\quad t^2\\
\Phi\left(x,y\right) =&\quad x^4+2x^3+3x^2+3xy+x+y^2+y
\end{aligned}$$



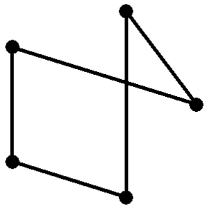
vert5_i9

$$\begin{aligned} \varphi\left(t\right) &= 4\left(t-1\right)^2 \\ \varphi_0\left(t\right) &= \left(2t-1\right)^2 \\ \overline{\varphi}\left(t\right) &= \left(t-1\right)^2 \\ \overline{\varphi}_0\left(t\right) &= t^2 \\ \Phi\left(x,y\right) &= \left(x^2+x+y\right)^2 \end{aligned}$$



vert5_i10

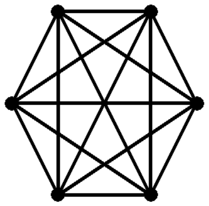
$$\begin{aligned} \varphi\left(t\right) &= 3\left(t-1\right)\left(t-2\right) \\ \varphi_0\left(t\right) &= 3t^2-3t+1 \\ \overline{\varphi}\left(t\right) &= \left(t-1\right)\left(t-2\right) \\ \overline{\varphi}_0\left(t\right) &= t^2 \\ \Phi\left(x,y\right) &= x^4+2x^3+x^2y+2x^2+2xy+x+y^2+y \end{aligned}$$



vert5_i11

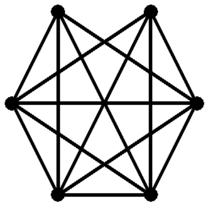
$$\begin{aligned} \varphi\left(t\right) &= 2\left(t-1\right) \\ \varphi_0\left(t\right) &= 2t-1 \\ \overline{\varphi}\left(t\right) &= t-1 \\ \overline{\varphi}_0\left(t\right) &= t \\ \Phi\left(x,y\right) &= x^4+x^3+x^2+x+y \end{aligned}$$

A.4 Simple, connected, bridgeless graphs with 6 vertices.



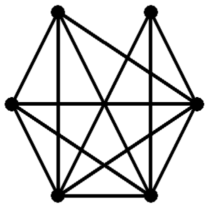
vert6_i1

$$\begin{aligned}
\varphi\left(t\right) = {} & \tfrac{1}{168}\left(t-1\right)\left(t-2\right)\left(18219t^{8}-200389t^{7}+1008604t^{6}\right. \\
& -3085170t^{5}+6335859t^{4}-9028913t^{3}+8850854t^{2} \\
& \left.-5643192t+1874880\right) \\
\varphi_{0}\left(t\right) = {} & \tfrac{1}{672}\left(2t-1\right)^{2}\left(18219t^{8}-72876t^{7}+136486t^{6}\right. \\
& -154392t^{5}+116179t^{4}-60060t^{3}+21436t^{2} \\
& \left.-4992t+672\right) \\
\overline{\varphi}\left(t\right) = {} & (t-1)(t-2)^{2}(t^{2}-4t+5) \\
& (t^{5}-6t^{4}+18t^{3}-34t^{2}+37t-28) \\
\overline{\varphi}_{0}\left(t\right) = {} & t^{7} \\
\Phi\left(x,y\right) = {} & x^{5}+10x^{4}+20x^{3}y+35x^{3}+15x^{2}y^{3}+45x^{2}y^{2} \\
& +90x^{2}y+50x^{2}+6xy^{6}+24xy^{5}+60xy^{4}+105xy^{3} \\
& +145xy^{2}+106xy+24x+y^{10}+5y^{9}+15y^{8}+35y^{7} \\
& +64y^{6}+96y^{5}+120y^{4}+120y^{3}+80y^{2}+24y
\end{aligned}$$



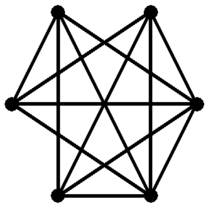
vert6_i2

$$\begin{aligned}
\varphi\left(t\right) = {} & \tfrac{1}{2520}\left(t-1\right)\left(t-2\right)\left(163951t^{7}-1634952t^{6}\right. \\
& +7481940t^{5}-20519906t^{4}+36677189t^{3} \\
& \left.-43286806t^{2}+32292696t-12277440\right) \\
\varphi_{0}\left(t\right) = {} & \tfrac{1}{5040}\left(2z-1\right)\left(163951z^{8}-655804z^{7}+1229718z^{6}\right. \\
& -1393840z^{5}+1048383z^{4}-538804z^{3}+188828z^{2} \\
& \left.-42432z+5040\right) \\
\overline{\varphi}\left(t\right) = {} & \left(t-1\right)\left(t-2\right)\left(t^{7}-11t^{6}+56t^{5}-172t^{4}+347t^{3}\right. \\
& \left.-467t^{2}+399t-171\right) \\
\overline{\varphi}_{0}\left(t\right) = {} & t^{9} \\
\Phi\left(x,y\right) = {} & x^{5}+9x^{4}+16x^{3}y+29x^{3}+9x^{2}y^{3}+33x^{2}y^{2}+68x^{2}y \\
& +39x^{2}+2xy^{6}+12xy^{5}+36xy^{4}+70xy^{3}+102xy^{2} \\
& +78xy+18x+y^{9}+5y^{8}+15y^{7}+33y^{6}+56y^{5} \\
& +76y^{4}+81y^{3}+57y^{2}+18y
\end{aligned}$$



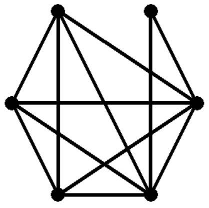
vert6_i3

$$\begin{aligned}
\varphi\left(t\right) = {} & 1/420(t-1)(t-2)(16693t^6-152677t^5+622577t^4\\
& -1467839t^3+2135826t^2-1882548t+821520)\\
\varphi_0\left(t\right) = {} & \tfrac{1}{840}(33386t^8-133544t^7+249683t^6-281645t^5\\
& +210259t^4-106911t^3+36712t^2-7940t+840)\\
\overline{\varphi}\left(t\right) = {} & (t-1)(t-2)(t^6-10t^5+45t^4-118t^3+193t^2\\
& -192t+93)\\
\overline{\varphi}_0\left(t\right) = {} & t^8\\
\Phi\left(x,y\right) = {} & x^5+8x^4+13x^3y+23x^3+6x^2y^3+24x^2y^2+49x^2y\\
& +28x^2+xy^6+6xy^5+21xy^4+44xy^3+66xy^2\\
& +52xy+12x+y^8+5y^7+14y^6+28y^5+42y^4\\
& +48y^3+36y^2+12y
\end{aligned}$$



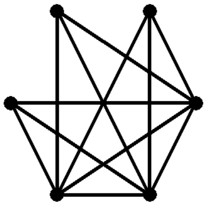
vert6_i4

$$\begin{aligned}
\varphi\left(t\right) = {} & \tfrac{1}{10080}\left(t-1\right)\left(t-2\right)\left(391841t^6-3508321t^5\right.\\
& +14503757t^4-35036447t^3+52555274t^2\\
& \left.-47852040t+21470400\right)\\
\varphi_0\left(t\right) = {} & \tfrac{1}{10080}(391841t^8-1567364t^7+2940546t^6\\
& -3335864t^5+2506329t^4-1281476t^3+441124t^2\\
& -95136t+10080)\\
\overline{\varphi}\left(t\right) = {} & \left(t-1\right)\left(t-2\right)\left(t^6-10t^5+46t^4-124t^3+209t^2\right.\\
& \left.-214t+106\right)\\
\overline{\varphi}_0\left(t\right) = {} & t^8\\
\Phi\left(x,y\right) = {} & x^5+8x^4+12x^3y+24x^3+4x^2y^3+22x^2y^2+51x^2y\\
& +31x^2+4xy^5+18xy^4+44xy^3+72xy^2+59xy\\
& +14x+y^8+5y^7+15y^6+31y^5+48y^4+56y^3\\
& +42y^2+14y
\end{aligned}$$



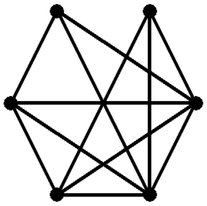
vert6_i5

$$\begin{aligned}
\varphi\left(t\right) = {} & \tfrac{1}{420}\left(t-1\right)\left(t-2\right)\left(11608t^5-83021t^4+270396t^3\right.\\
& \qquad \left.-491475t^2+512980t-257040\right)\\
\varphi_0\left(t\right) = {} & \tfrac{1}{420}\left(2t-1\right)\left(5804t^6-17412t^5+24145t^4-19270t^3\right.\\
& \qquad \left.+9531t^2-2798t+420\right)\\
\overline{\varphi}\left(t\right) = {} & \left(t-1\right)\left(t-2\right)^2\left(t^4-6t^3+17t^2-25t+19\right)\\
\overline{\varphi}_0\left(t\right) = {} & t^7\\
\Phi\left(x,y\right) = {} & x^5+7x^4+11x^3y+17x^3+5x^2y^3+18x^2y^2+33x^2y\\
& \qquad +17x^2+xy^6+4xy^5+13xy^4+26xy^3+37xy^2\\
& \qquad +28xy+6x+y^7+4y^6+10y^5+17y^4+21y^3\\
& \qquad +17y^2+6y
\end{aligned}$$



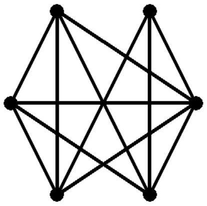
vert6_i6

$$\begin{aligned}
\varphi\,(t) =\; & \tfrac{1}{420}(t-1)(t-2)(10279t^5 - 87610t^4 + 306887t^3 \\
& \quad -572954t^2 + 599790t - 304920) \\
\varphi_0\,(t) =\; & \tfrac{1}{840}(2t-1)(10279t^6 - 30837t^5 + 42937t^4 - 34479t^3 \\
& \quad +17344t^2 - 5244t + 840) \\
\overline{\varphi}\,(t) =\; & (t-1)(t-2)(t^5 - 9t^4 + 34t^3 - 70t^2 + 82t - 46) \\
\overline{\varphi}_0\,(t) =\; & t^7 \\
\Phi\,(x,y) =\; & x^5 + 7x^4 + 10x^3y + 18x^3 + 3x^2y^3 + 18x^2y^2 + 33x^2y \\
& \quad +20x^2 + 3xy^5 + 12xy^4 + 27xy^3 + 39xy^2 + 33xy \\
& \quad +8x + y^7 + 5y^6 + 12y^5 + 20y^4 + 25y^3 \\
& \quad +21y^2 + 8y
\end{aligned}$$



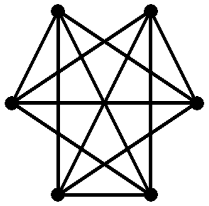
vert6_i7

$$\begin{aligned}
\varphi\,(t) =\; & \tfrac{1}{1260}(t-1)(t-2)(30556t^5 - 254489t^4 + 905660t^3 \\
& \quad -1756957t^2 + 1924338t - 990360) \\
\varphi_0\,(t) =\; & \tfrac{1}{1260}(2t-1)(15278t^6 - 45834t^5 + 64205t^4 - 52020t^3 \\
& \quad +26357t^2 - 7986t + 1260) \\
\overline{\varphi}\,(t) =\; & (t-1)(t-2)(t^5 - 9t^4 + 35t^3 - 75t^2 + 91t - 51) \\
\overline{\varphi}_0\,(t) =\; & t^7 \\
\Phi\,(x,y) =\; & x^5 + 7x^4 + 10x^3y + 18x^3 + 3x^2y^3 + 16x^2y^2 + 35x^2y \\
& \quad +20x^2 + 10xy^4 + 26xy^3 + 2xy^5 + 43xy^2 + 35xy \\
& \quad +8x + y^7 + 5y^6 + 13y^5 + 23y^4 + 29y^3 \\
& \quad +23y^2 + 8y
\end{aligned}$$



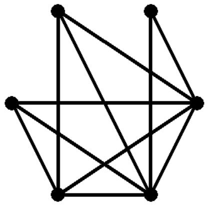
vert6_i8

$$\begin{aligned}
\varphi\,(t) =\; & \tfrac{1}{1260}(t-1)(t-2)(29769t^5 - 242038t^4 + 877737t^3 \\
& \quad -1760348t^2 + 2008308t - 1081080) \\
\varphi_0\,(t) =\; & \tfrac{1}{840}(2t-1)(9923t^6 - 29769t^5 + 41959t^4 - 34303t^3 \\
& \quad +17558t^2 - 5368t + 840) \\
\overline{\varphi}\,(t) =\; & (t-1)(t-2)(t^5 - 9t^4 + 36t^3 - 80t^2 + 101t - 59) \\
\overline{\varphi}_0\,(t) =\; & t^7 \\
\Phi\,(x,y) =\; & x^5 + 7x^4 + 9x^3y + 19x^3 + 2x^2y^3 + 14x^2y^2 + 36x^2y \\
& \quad +23x^2 + xy^5 + 8xy^4 + 25xy^3 + 46xy^2 + 41xy \\
& \quad +10x + y^7 + 5y^6 + 14y^5 + 26y^4 + 34y^3 \\
& \quad +28y^2 + 10y
\end{aligned}$$



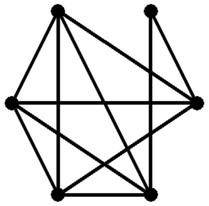
vert6_i9

$$\begin{aligned}
\varphi\left(t\right) = {} & \tfrac{1}{105}(t-1)(2432t^6-24161t^5+110274t^4-291001t^3\\
& \quad +472972t^2-451140t+195510)\\
\varphi_0\left(t\right) = {} & \tfrac{1}{105}(2t-1)(1216t^6-3648t^5+5176t^4-4272t^3\\
& \quad +2218t^2-690t+105)\\
\overline{\varphi}\left(t\right) = {} & (t-1)(t^6-11t^5+55t^4-159t^3+282t^2-290t+133)\\
\overline{\varphi}_0\left(t\right) = {} & t^7\\
\Phi\left(x,y\right) = {} & x^5+7x^4+8x^3y+20x^3+12x^2y^2+39x^2y+25x^2\\
& \quad +6xy^4+24xy^3+52xy^2+46xy+11x+y^7+5y^6\\
& \quad +15y^5+29y^4+40y^3+32y^2+11y
\end{aligned}$$



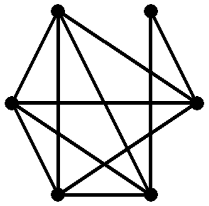
vert6_i10

$$\begin{aligned}
\varphi\left(t\right) = {} & \tfrac{1}{180}(t-1)(t-2)(3059t^4-19992t^3+52201t^2\\
& \quad -68268t+41040)\\
\varphi_0\left(t\right) = {} & \tfrac{1}{180}(3059t^6-9177t^5+12560t^4-9825t^3+4721t^2\\
& \quad -1338t+180)\\
\overline{\varphi}\left(t\right) = {} & (t-1)(t-2)(t^4-7t^3+20t^2-29t+19)\\
\overline{\varphi}_0\left(t\right) = {} & t^6\\
\Phi\left(x,y\right) = {} & x^5+6x^4+8x^3y+13x^3+2x^2y^3+12x^2y^2+22x^2y\\
& \quad +12x^2+xy^5+6xy^4+15xy^3+22xy^2+18xy\\
& \quad +4x+y^6+4y^5+8y^4+11y^3+10y^2+4y
\end{aligned}$$



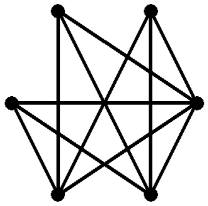
vert6_i11

$$\begin{aligned}
\varphi\left(t\right) = {} & \tfrac{1}{90}(t-1)(t-2)(1513t^4-9585t^3+26084t^2\\
& \quad -36258t+22140)\\
\varphi_0\left(t\right) = {} & \tfrac{1}{180}(3026t^6-9078t^5+12515t^4-9900t^3+4799t^2\\
& \quad -1362t+180)\\
\overline{\varphi}\left(t\right) = {} & (t-1)(t-2)(t^4-7t^3+21t^2-32t+21)\\
\overline{\varphi}_0\left(t\right) = {} & t^6\\
\Phi\left(x,y\right) = {} & x^5+6x^4+8x^3y+13x^3+2x^2y^3+11x^2y^2+23x^2y\\
& \quad +12x^2+xy^5+5xy^4+14xy^3+24xy^2+19xy\\
& \quad +4x+y^6+4y^5+9y^4+13y^3+11y^2+4y
\end{aligned}$$



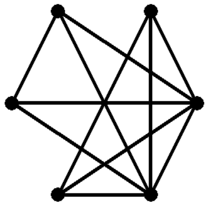
vert6_i12

$$\begin{aligned}
\varphi\left(t\right) = {} & \tfrac{2}{3}(t-1)(24t^5-195t^4+715t^3-1440t^2+1622t-816)\\
\varphi_0\left(t\right) = {} & \tfrac{1}{3}(2t-1)^2(12t^4-24t^3+23t^2-11t+3)\\
\overline{\varphi}\left(t\right) = {} & (t-1)(t^5-9t^4+36t^3-79t^2+96t-51)\\
\overline{\varphi}_0\left(t\right) = {} & t^6\\
\Phi\left(x,y\right) = {} & x^5+6x^4+7x^3y+14x^3+2x^2y^3+9x^2y^2+23x^2y\\
& \quad +15x^2+xy^5+4xy^4+13xy^3+24xy^2+24xy\\
& \quad +6x+y^6+4y^5+10y^4+15y^3+15y^2+6y
\end{aligned}$$



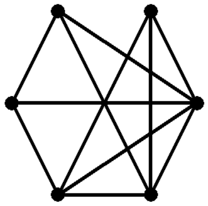
vert6_i13

$$\begin{aligned}
\varphi\left(t\right) = {} & \tfrac{1}{360}(t-1)(t-2)(5209t^4-39096t^3+117509t^2 \\
& \quad -173706t+115560) \\
\varphi_0\left(t\right) = {} & \tfrac{1}{360}(5209t^6-15627t^5+21745t^4-17445t^3+8686t^2 \\
& \quad -2568t+360) \\
\overline{\varphi}\left(t\right) = {} & (t-1)(t-2)(t^4-8t^3+26t^2-42t+30) \\
\overline{\varphi}_0\left(t\right) = {} & t^6 \\
\Phi\left(x,y\right) = {} & x^5+6x^4+6x^3y+15x^3+9x^2y^2+24x^2y+17x^2+3xy^4 \\
& \quad +14xy^3+27xy^2+27xy+7x+y^6+5y^5+12y^4 \\
& \quad +18y^3+17y^2+7y
\end{aligned}$$



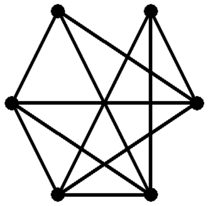
vert6_i14

$$\begin{aligned}
\varphi\left(t\right) = {} & \tfrac{1}{45}(t-1)(t-2)(t-3)(691t^3-3210t^2+5849t-4320) \\
\varphi_0\left(t\right) = {} & \tfrac{1}{180}(2764t^6-8292t^5+11\,545t^4-9270t^3+4591t^2 \\
& \quad -1338t+180) \\
\overline{\varphi}\left(t\right) = {} & (t-1)(t-2)^2(t-3)(t^2-3t+4) \\
\overline{\varphi}_0\left(t\right) = {} & t^6 \\
\Phi\left(x,y\right) = {} & t^5+6t^4+8t^3z+13t^3+2t^2z^3+10t^2z^2+24t^2z+12t^2 \\
& \quad +4tz^4+14tz^3+26tz^2+20tz+4t+z^6+5z^5 \\
& \quad +15z^3+12z^2+11z^4+4z
\end{aligned}$$



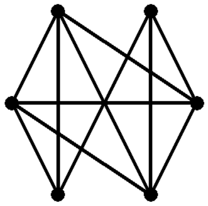
vert6_i15

$$\begin{aligned}
\varphi\left(t\right) = {} & \tfrac{1}{180}(t-1)(t-2)(2651t^4-19824t^3+59491t^2 \\
& \quad -88554t+57240) \\
\varphi_0\left(t\right) = {} & \tfrac{1}{180}(2651t^6-7953t^5+11090t^4-8925t^3+4439t^2 \\
& \quad -1302t+180) \\
\overline{\varphi}\left(t\right) = {} & (t-1)(t-2)(t^4-8t^3+26t^2-42t+29) \\
\overline{\varphi}_0\left(t\right) = {} & t^6 \\
\Phi\left(x,y\right) = {} & x^5+6x^4+7x^3y+14x^3+x^2y^3+9x^2y^2+24x^2y+15x^2 \\
& \quad +3xy^4+13xy^3+27xy^2+25xy+6x+y^6+5y^5 \\
& \quad +12y^4+18y^3+16y^2+6y
\end{aligned}$$



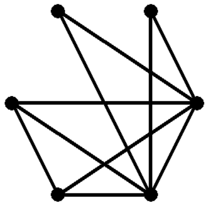
vert6_i16

$$\begin{aligned}
\varphi\left(t\right) = {} & \tfrac{1}{60}(t-1)(t-2)(861t^4-6304t^3+19441t^2 \\
& \quad -30594t+20700) \\
\varphi_0\left(t\right) = {} & \tfrac{1}{20}(287t^6-861t^5+1205t^4-975t^3+488t^2 \\
& \quad -144t+20) \\
\overline{\varphi}\left(t\right) = {} & (t-1)(t-2)(t^4-8t^3+27t^2-46t+33) \\
\overline{\varphi}_0\left(t\right) = {} & t^6 \\
\Phi\left(x,y\right) = {} & x^5+6x^4+6x^3y+15x^3+7x^2y^2+26x^2y+17x^2 \\
& \quad +2xy^4+12xy^3+30xy^2+29xy+7x+y^6 \\
& \quad +5y^5+13y^4+21y^3+19y^2+7y
\end{aligned}$$



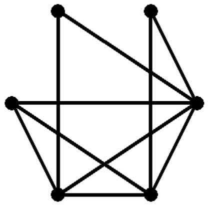
vert6_i17

$$\begin{aligned}
\varphi\left(t\right) = {} & \tfrac{1}{45}(t-1)(t-2)(643t^4-4677t^3+14423t^2 \\
& \quad -22557t+15660) \\
\varphi_0\left(t\right) = {} & \tfrac{1}{180}(2572t^6-7716t^5+10\,825t^4-8790t^3 \\
& \quad +4423t^2-1314t+180) \\
\overline{\varphi}\left(t\right) = {} & (t-1)(t-2)(t^4-8t^3+27t^2-46t+34) \\
\overline{\varphi}_0\left(t\right) = {} & t^6 \\
\Phi\left(x,y\right) = {} & x^5+6x^4+6x^3y+15x^3+x^2y^3+7x^2y^2+24x^2y \\
& \quad +18x^2+2xy^4+11xy^3+28xy^2+30xy+8x \\
& \quad +y^6+5y^5+13y^4+21y^3+20y^2+8y
\end{aligned}$$



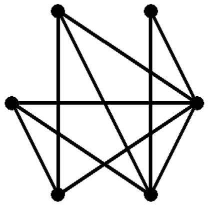
vert6_i18

$$\begin{aligned}
\varphi\left(t\right) = {} & \tfrac{1}{30}(t-1)(t-2)(361t^3-1702t^2+2777t-1980) \\
\varphi_0\left(t\right) = {} & \tfrac{1}{60}(2t-1)(361t^4-722t^3+639t^2-278t+60) \\
\overline{\varphi}\left(t\right) = {} & (t-1)(t-2)(t^3-5t^2+9t-7) \\
\overline{\varphi}_0\left(t\right) = {} & t^5 \\
\Phi\left(x,y\right) = {} & x^5+5x^4+6x^3y+9x^3+x^2y^3+8x^2y^2+13x^2y+7x^2 \\
& \quad +2xy^4+8xy^3+11xy^2+9xy+2x+y^5+3y^4 \\
& \quad +4y^3+4y^2+2y
\end{aligned}$$



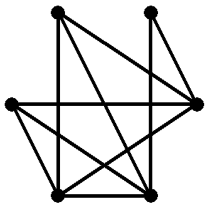
vert6_i19

$$\begin{aligned}
\varphi\left(t\right) = {} & \tfrac{1}{30}(t-1)(t-2)(351t^3-1592t^2+2897t-2160) \\
\varphi_0\left(t\right) = {} & \tfrac{1}{60}(2t-1)(351t^4-702t^3+629t^2-278t+60) \\
\overline{\varphi}\left(t\right) = {} & (t-1)(t-2)^2(t^2-3t+4) \\
\overline{\varphi}_0\left(t\right) = {} & t^5 \\
\Phi\left(x,y\right) = {} & x^5+5x^4+6x^3y+9x^3+x^2y^3+7x^2y^2+14x^2y+7x^2 \\
& \quad +2xy^4+7xy^3+12xy^2+10yx+2x+y^5+3y^4 \\
& \quad +5y^3+5y^2+2y
\end{aligned}$$



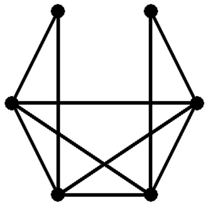
vert6_i20

$$\begin{aligned}
\varphi\left(t\right) = {} & \tfrac{1}{60}(t-1)(t-2)(618t^3-3491t^2+7011t-5580) \\
\varphi_0\left(t\right) = {} & \tfrac{1}{60}(2t-1)(309t^4-618t^3+571t^2-262t+60) \\
\overline{\varphi}\left(t\right) = {} & (t-1)(t-2)(t^3-6t^2+13t-11) \\
\overline{\varphi}_0\left(t\right) = {} & t^5 \\
\Phi\left(x,y\right) = {} & x^5+5x^4+5x^3y+10x^3+6x^2y^2+15x^2y+9x^2+xy^4 \\
& \quad +7xy^3+14xy^2+13xy+3x+y^5+4y^4+7y^3 \\
& \quad +7y^2+3y
\end{aligned}$$



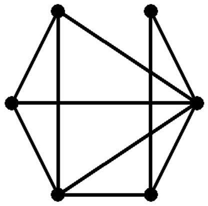
vert6_i21

$$\begin{aligned}
\varphi\,(t) = {} & \tfrac{1}{10}(t-1)(t-2)(98t^3-541t^2+1161t-1020)\\
\varphi_0\,(t) = {} & \tfrac{1}{10}(2t-1)(49t^4-98t^3+91t^2-42t+10)\\
\overline{\varphi}\,(t) = {} & (t-1)(t-2)(t^3-6t^2+14t-13)\\
\overline{\varphi}_0\,(t) = {} & t^5\\
\Phi\,(x,y) = {} & x^5+5x^4+4x^3y+11x^3+4x^2y^2+16x^2y+11x^2\\
& \quad +xy^4+6xy^3+15xy^2+16xy+4x+y^5\\
& \quad +4y^4+8y^3+9y^2+4y
\end{aligned}$$



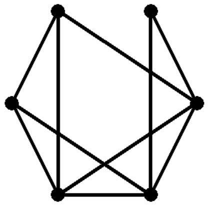
vert6_i22

$$\begin{aligned}
\varphi\,(t) = {} & \tfrac{1}{15}(t-1)(176t^4-1109t^3+3101t^2\\
& \quad -4194t+2280)\\
\varphi_0\,(t) = {} & \tfrac{1}{15}(2t-1)(44t^2-44t+15)(2t^2-2t+1)\\
\overline{\varphi}\,(t) = {} & (t-1)(t^4-7t^3+21t^2-30t+17)\\
\overline{\varphi}_0\,(t) = {} & t^5\\
\Phi\,(x,y) = {} & x^5+5x^4+6x^3y+9x^3+x^2y^3+7x^2y^2+14x^2y\\
& \quad +7x^2+2xy^4+6xy^3+13xy^2+10xy+2x\\
& \quad +y^5+3y^4+6y^3+5y^2+2y
\end{aligned}$$



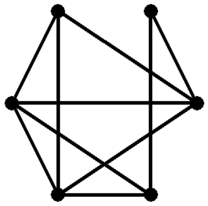
vert6_i23

$$\begin{aligned}
\varphi\,(t) = {} & \tfrac{1}{30}(t-1)(t-2)(316t^3-1797t^2\\
& \quad +3687t-2700)\\
\varphi_0\,(t) = {} & \tfrac{1}{15}(2t-1)(79t^4-158t^3+146t^2-67t+15)\\
\overline{\varphi}\,(t) = {} & (t-1)(t-2)^2(t^2-4t+5)\\
\overline{\varphi}_0\,(t) = {} & t^5\\
\Phi\,(x,y) = {} & x^5+5x^4+6x^3y+9x^3+x^2y^3+6x^2y^2+15x^2y\\
& \quad +7x^2+xy^4+6xy^3+14xy^2+11xy+2x\\
& \quad +y^5+4y^4+7y^3+6y^2+2y
\end{aligned}$$



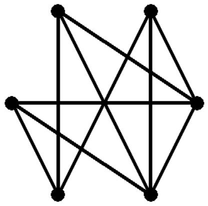
vert6_i24

$$\begin{aligned}
\varphi\,(t) = {} & \tfrac{1}{15}(t-1)(t-2)(153t^3-836t^2+1821t-1485)\\
\varphi_0\,(t) = {} & \tfrac{1}{30}(2t-1)(153t^4-306t^3+287t^2-134t+30)\\
\overline{\varphi}\,(t) = {} & (t-1)(t-2)^2(t^2-4t+6)\\
\overline{\varphi}_0\,(t) = {} & t^5\\
\Phi\,(x,y) = {} & x^5+5x^4+5x^3y+10x^3+5y^2x^2+16x^2y+9x^2\\
& \quad +xy^4+6xy^3+15xy^2+14yx+3x+y^5\\
& \quad +4y^4+8y^3+8y^2+3y
\end{aligned}$$



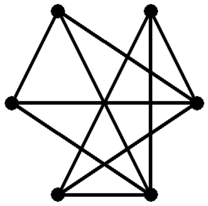
vert6_i25

$$\begin{aligned}
\varphi\left(t\right) = \;& \tfrac{1}{10}(t-1)(t-2)(98t^3-541t^2+1161t-1020)\\
\varphi_0\left(t\right) = \;& \tfrac{1}{10}(2t-1)(49t^4-98t^3+91t^2-42t+10)\\
\overline{\varphi}\left(t\right) = \;& (t-1)(t-2)(t^3-6t^2+14t-13)\\
\overline{\varphi}_0\left(t\right) = \;& t^5\\
\Phi\left(x,y\right) = \;& x^5+5x^4+5x^3y+10x^3+x^2y^3+5x^2y^2+14x^2y\\
& +10x^2+xy^4+5xy^3+13xy^2+15xy+4x\\
& +y^5+4y^4+8y^3+9y^2+4y
\end{aligned}$$



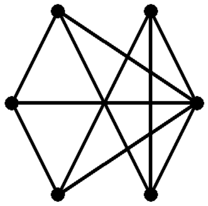
vert6_i26

$$\begin{aligned}
\varphi\left(t\right) = \;& \tfrac{1}{10}(t-1)(t-2)(t-3)(87t^2-313t+440)\\
\varphi_0\left(t\right) = \;& \tfrac{1}{60}(2t-1)(261t^4-522t^3+499t^2-238t+60)\\
\overline{\varphi}\left(t\right) = \;& (t-1)(t-2)(t-3)(t^2-4t+6)\\
\overline{\varphi}_0\left(t\right) = \;& t^5\\
\Phi\left(x,y\right) = \;& 5x^4+3x^3y+12x^3+3x^2y^2+15x^2y+14x^2+4xy^3\\
& +15xy^2+21xy+6x+y^5+5y^4+x^5+11y^3\\
& +13y^2+6y
\end{aligned}$$



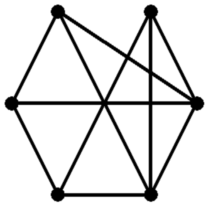
vert6_i27

$$\begin{aligned}
\varphi\left(t\right) = \;& \tfrac{1}{15}(t-1)(t-2)(136t^3-897t^2+2207t-1890)\\
\varphi_0\left(t\right) = \;& \tfrac{1}{15}(2t-1)(2t^2-2t+1)(34t^2-34t+15)\\
\overline{\varphi}\left(t\right) = \;& (t-1)(t-2)^2(t^2-5t+8)\\
\overline{\varphi}_0\left(t\right) = \;& t^5\\
\Phi\left(x,y\right) = \;& x^5+5x^4+4x^3y+11x^3+2x^2y^2+18x^2y+11x^2\\
& +4xy^3+18xy^2+18xy+4x+y^5+5y^4\\
& +11y^3+11y^2+4y
\end{aligned}$$



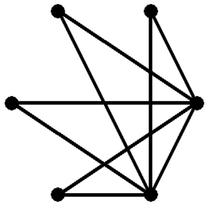
vert6_i28

$$\begin{aligned}
\varphi\left(t\right) = \;& \tfrac{5}{6}(t-1)(t-2)(t-3)(11t^2-41t+48)\\
\varphi_0\left(t\right) = \;& \tfrac{1}{12}(2t-1)(55t^4-110t^3+105t^2-50t+12)\\
\overline{\varphi}\left(t\right) = \;& (t-1)(t-2)(t-3)(t^2-4t+5)\\
\overline{\varphi}_0\left(t\right) = \;& t^5\\
\Phi\left(x,y\right) = \;& x^5+5x^4+5x^3y+10x^3+5x^2y^2+15x^2y+10x^2\\
& +5xy^3+15xy^2+16xy+4x+y^5+5y^4\\
& +10y^3+10y^2+4y
\end{aligned}$$



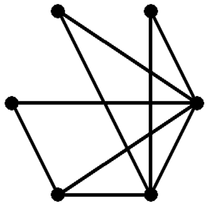
vert6_i29

$$\begin{aligned}
\varphi\left(t\right) = \;& \tfrac{1}{30}(t-1)(t-2)(267t^3-1759t^2+4274t-3870)\\
\varphi_0\left(t\right) = \;& \tfrac{1}{20}(2t-1)(89t^4-178t^3+171t^2-82t+20)\\
\overline{\varphi}\left(t\right) = \;& (t-1)(t-2)(t^3-7t^2+18t-17)\\
\overline{\varphi}_0\left(t\right) = \;& t^5\\
\Phi\left(x,y\right) = \;& x^5+5x^4+4x^3y+11x^3+3x^2y^2+16x^2y+12x^2\\
& +4xy^3+16xy^2+19xy+5x+y^5+5y^4+11y^3\\
& +12y^2+5y
\end{aligned}$$



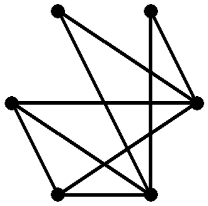
vert6_i30

$$\begin{aligned}
\varphi\,(t) =&\ \tfrac{5}{12}(t-1)(t-2)(23t^2-41t+36)\\
\varphi_0\,(t) =&\ \tfrac{1}{12}\,(115t^4-230t^3+185t^2-70t+12)\\
\overline{\varphi}\,(t) =&\ (t-1)(t-2)(t^2-2t+2)\\
\overline{\varphi}_0\,(t) =&\ t^5\\
\Phi\,(x,y) =&\ x^5+4x^4+4x^3y+6x^3+6x^2y^2+6x^2y+4x^2+4xy^3\\
&\quad +4xy^2+4xy+x+y^4+y^3+y^2+y
\end{aligned}$$



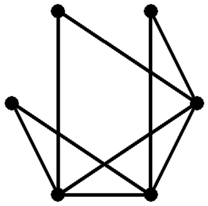
vert6_i31

$$\begin{aligned}
\varphi\,(t) =&\ \tfrac{1}{6}(t-1)(t-2)(49t^2-135t+126)\\
\varphi_0\,(t) =&\ \tfrac{1}{6}(7t^2-8t+3)(7t^2-6t+2)\\
\overline{\varphi}\,(t) =&\ (t-1)(t-2)(t^2-3t+3)\\
\overline{\varphi}_0\,(t) =&\ t^4\\
\Phi\,(x,y) =&\ t^5+4t^4+4t^3z+6t^3+4t^2z^2+8t^2z+4t^2+3tz^3\\
&\quad +6tz^2+5tz+t+z^4+2z^3+2z^2+z
\end{aligned}$$



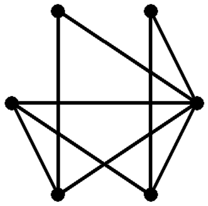
vert6_i32

$$\begin{aligned}
\varphi\,(t) =&\ \tfrac{1}{6}(t-1)(t-2)(41t^2-151t+180)\\
\varphi_0\,(t) =&\ \tfrac{1}{6}\,(41t^4-82t^3+73t^2-32t+6)\\
\overline{\varphi}\,(t) =&\ (t-1)(t-2)(t^2-4t+5)\\
\overline{\varphi}_0\,(t) =&\ t^4\\
\Phi\,(x,y) =&\ t^5+4t^4+2zt^3+8t^3+t^2z^2+10t^2z+7t^2+2tz^3\\
&\quad +8tz^2+9tz+2t+z^4+3z^3+4z^2+2z
\end{aligned}$$



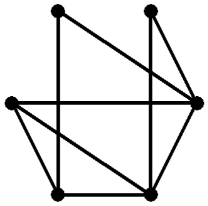
vert6_i33

$$\begin{aligned}
\varphi\,(t) =&\ 2(t-1)(4t^3-18t^2+34t-23)\\
\varphi_0\,(t) =&\ (2t-1)^2(2t^2-2t+1)\\
\overline{\varphi}\,(t) =&\ (t-1)(t^3-5t^2+10t-7)\\
\overline{\varphi}_0\,(t) =&\ t^4\\
\Phi\,(x,y) =&\ t^5+4t^4+4t^3z+6t^3+3t^2z^2+9t^2z+4t^2+3tz^3\\
&\quad +6tz^2+6tz+t+z^4+2z^3+3z^2+z
\end{aligned}$$



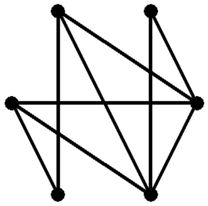
vert6_i34

$$\begin{aligned}
\varphi\,(t) =&\ \tfrac{1}{12}(t-1)(t-2)(87t^2-337t+324)\\
\varphi_0\,(t) =&\ \tfrac{1}{4}\,(29t^4-58t^3+51t^2-22t+4)\\
\overline{\varphi}\,(t) =&\ (t-1)(t-2)^3\\
\overline{\varphi}_0\,(t) =&\ t^4\\
\Phi\,(x,y) =&\ t^5+4t^4+4t^3z+6t^3+3t^2z^2+9t^2z+4t^2+2tz^3\\
&\quad +7tz^2+6tz+t+z^4+3z^3+3z^2+z
\end{aligned}$$



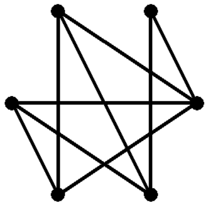
vert6_i35

$$\begin{aligned}
\varphi\,(t) ={}& \tfrac{1}{6}(t-1)(t-2)(41t^2-151t+180)\\
\varphi_0\,(t) ={}& \tfrac{1}{6}\,(41t^4-82t^3+73t^2-32t+6)\\
\overline{\varphi}\,(t) ={}& (t-1)(t-2)(t^2-4t+5)\\
\overline{\varphi}_0\,(t) ={}& t^4\\
\Phi\,(x,y) ={}& t^5+4t^4+3t^3z+7t^3+2t^2z^2+9t^2z+6t^2+2tz^3\\
&\quad +7tz^2+8tz+2t+z^4+3z^3+4z^2+2z
\end{aligned}$$



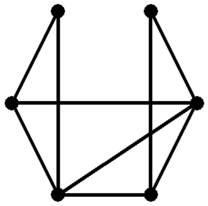
vert6_i36

$$\begin{aligned}
\varphi\,(t) ={}& \tfrac{1}{6}(t-1)(t-2)(41t^2-151t+180)\\
\varphi_0\,(t) ={}& \tfrac{1}{6}\,(41t^4-82t^3+73t^2-32t+6)\\
\overline{\varphi}\,(t) ={}& (t-1)(t-2)(t^2-4t+5)\\
\overline{\varphi}_0\,(t) ={}& t^4\\
\Phi\,(x,y) ={}& t^5+4t^4+3t^3z+7t^3+3t^2z^2+8t^2z+6t^2+2tz^3\\
&\quad +6tz^2+8tz+2t+z^4+3z^3+4z^2+2z
\end{aligned}$$



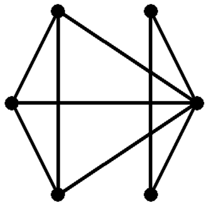
vert6_i37

$$\begin{aligned}
\varphi\,(t) ={}& \tfrac{1}{3}(t-1)(t-2)(18t^2-86t+117)\\
\varphi_0\,(t) ={}& (3t^2-3t+1)(2t^2-2t+1)\\
\overline{\varphi}\,(t) ={}& (t-1)(t-2)(t^2-5t+7)\\
\overline{\varphi}_0\,(t) ={}& t^4\\
\Phi\,(x,y) ={}& t^5+4t^4+2t^3z+8t^3+t^2z^2+9zt^2+8t^2+tz^3\\
&\quad +7tz^2+11tz+3t+z^4+4z^3+6z^2+3z
\end{aligned}$$



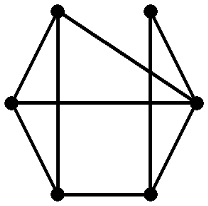
vert6_i38

$$\begin{aligned}
\varphi\,(t) ={}& \tfrac{1}{12}(t-1)(t-2)(87t^2-337t+324)\\
\varphi_0\,(t) ={}& \tfrac{1}{4}\,(29t^4-58t^3+51t^2-22t+4)\\
\overline{\varphi}\,(t) ={}& (t-1)(t-2)^3\\
\overline{\varphi}_0\,(t) ={}& t^4\\
\Phi\,(x,y) ={}& t^5+4t^4+4t^3z+6t^3+3t^2z^2+9t^2z+4t^2+2tz^3\\
&\quad +7tz^2+6tz+t+z^4+3z^3+3z^2+z
\end{aligned}$$



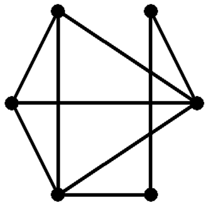
vert6_i39

$$\begin{aligned}
\varphi\,(t) ={}& 8(t-1)^2(t-2)(t-3)\\
\varphi_0\,(t) ={}& (2t-1)^2(2t^2-2t+1)\\
\overline{\varphi}\,(t) ={}& (t-1)^2(t-2)(t-3)\\
\overline{\varphi}_0\,(t) ={}& t^4\\
\Phi\,(x,y) ={}& (t^3+3t^2+4tz+2t+z^3+3z^2+2z)(t^2+t+z)
\end{aligned}$$



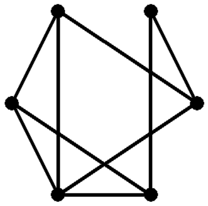
vert6_i40

$$\begin{aligned}
\varphi\,(t) ={}& \tfrac{1}{3}(t-1)(t-2)(t-3)(19t-36)\\
\varphi_0\,(t) ={}& \tfrac{1}{3}\,(19t^4-38t^3+35t^2-16t+3)\\
\overline{\varphi}\,(t) ={}& (t-1)(t-2)^2(t-3)\\
\overline{\varphi}_0\,(t) ={}& t^4\\
\Phi\,(x,y) ={}& t^5+4t^4+3t^3z+7t^3+t^2z^2+10t^2z+6t^2\\
&\quad +tz^3+8tz^2+9tz+2t+z^4+4z^3\\
&\quad +5z^2+2z
\end{aligned}$$



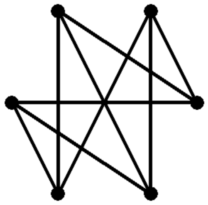
vert6_i41

$$\begin{aligned}
\varphi\,(t) ={}& \tfrac{1}{6}(t-1)(t-2)(41t^2-151t+180)\\
\varphi_0\,(t) ={}& \tfrac{1}{6}\,(41t^4-82t^3+73t^2-32t+6)\\
\overline{\varphi}\,(t) ={}& (t-1)(t-2)(t^2-4t+5)\\
\overline{\varphi}_0\,(t) ={}& t^4\\
\Phi\,(x,y) ={}& t^5+4t^4+4t^3z+6t^3+t^2z^3+3t^2z^2+7t^2z\\
&\quad +5t^2+tz^3+5tz^2+7tz+2t+z^4\\
&\quad +3z^3+4z^2+2z
\end{aligned}$$



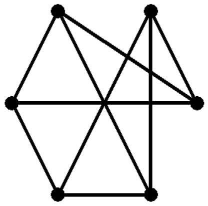
vert6_i42

$$\begin{aligned}
\varphi\,(t) ={}& \tfrac{1}{3}(t-1)(t-2)(18t^2-86t+117)\\
\varphi_0\,(t) ={}& (3t^2-3t+1)(2t^2-2t+1)\\
\overline{\varphi}\,(t) ={}& (t-1)(t-2)(t^2-5t+7)\\
\overline{\varphi}_0\,(t) ={}& t^4\\
\Phi\,(x,y) ={}& t^5+4t^4+3t^3z+7t^3+2t^2z^2+8t^2z+7t^2\\
&\quad +tz^3+6tz^2+10tz+3t+z^4+4z^3\\
&\quad +6z^2+3z
\end{aligned}$$



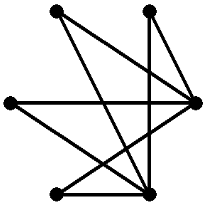
vert6_i43

$$\begin{aligned}
\varphi\,(t) ={}& \tfrac{3}{4}(t-1)(t-2)(7t^2-41t+68)\\
\varphi_0\,(t) ={}& \tfrac{1}{4}\,(21t^4-42t^3+39t^2-18t+4)\\
\overline{\varphi}\,(t) ={}& (t-1)(t-2)(t^2-6t+10)\\
\overline{\varphi}_0\,(t) ={}& t^4\\
\Phi\,(x,y) ={}& t^5+4t^4+10t^3+9t^2z+11t^2+6tz^2\\
&\quad +15tz+5t+z^4+5z^3+9z^2+5z
\end{aligned}$$



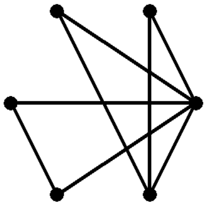
vert6_i44

$$\begin{aligned}
\varphi\,(t) ={}& \tfrac{1}{2}(t-1)(t-2)(t-3)(11t-32)\\
\varphi_0\,(t) ={}& \tfrac{1}{2}\,(11t^4-22t^3+21t^2-10t+2)\\
\overline{\varphi}\,(t) ={}& (t-1)(t-2)(t-3)^2\\
\overline{\varphi}_0\,(t) ={}& t^4\\
\Phi\,(x,y) ={}& t^5+4t^4+2t^3z+8t^3+9t^2z+8z^2+7tz^2\\
&\quad +13tz+4t+z^4+5z^3+9t^2+4z
\end{aligned}$$



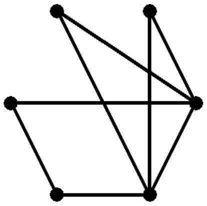
vert6_i45

$$\begin{aligned}
\varphi\,(t) &= \tfrac{2}{3}(t-1)(8t^2-22t+21)\\
\varphi_0\,(t) &= \tfrac{1}{3}(2t-1)(8t^2-8t+3)\\
\overline{\varphi}\,(t) &= (t-1)(t^2-3t+3)\\
\overline{\varphi}_0\,(t) &= t^3\\
\Phi\,(x,y) &= t^5+3t^4+6t^3+6t^2z+4t^2+4tz^2+4tz\\
&\qquad +t+z^3+z^2+z
\end{aligned}$$



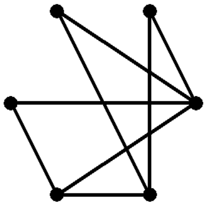
vert6_i46

$$\begin{aligned}
\varphi\,(t) &= 6(t-1)^2(t-2)\\
\varphi_0\,(t) &= (2t-1)(3t^2-3t+1)\\
\overline{\varphi}\,(t) &= (t-1)^2(t-2)\\
\overline{\varphi}_0\,(t) &= t^3\\
\Phi\,(x,y) &= (t^2+t+z)(t^3+2t^2+2tz+t+z^2+z)
\end{aligned}$$



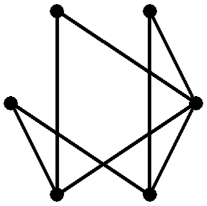
vert6_i47

$$\begin{aligned}
\varphi\,(t) &= \tfrac{2}{3}(t-1)(8t^2-22t+21)\\
\varphi_0\,(t) &= \tfrac{1}{3}(2t-1)(8t^2-8t+3)\\
\overline{\varphi}\,(t) &= (t-1)(t^2-3t+3)\\
\overline{\varphi}_0\,(t) &= t^3\\
\Phi\,(x,y) &= t^5+3t^4+2t^3z+4t^3+z^2t^2+4t^2z+3t^2\\
&\qquad +3tz^2+3tz+t+z^3+z^2+z
\end{aligned}$$



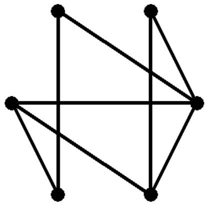
vert6_i48

$$\begin{aligned}
\varphi\,(t) &= \tfrac{1}{3}(t-1)(t-2)(14t-27)\\
\varphi_0\,(t) &= \tfrac{1}{3}(2t-1)(7t^2-7t+3)\\
\overline{\varphi}\,(t) &= (t-1)(t-2)^2\\
\overline{\varphi}_0\,(t) &=\\
\Phi\,(x,y) &= t^5+3t^4+t^3z+5t^3+5t^2z+4t^2+3tz^2\\
&\qquad +5tz+t+z^3+2z^2+z
\end{aligned}$$



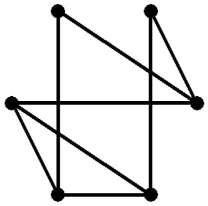
vert6_i49

$$\begin{aligned}
\varphi\,(t) &= \tfrac{1}{3}(t-1)(t-2)(14t-27)\\
\varphi_0\,(t) &= \tfrac{1}{3}(2t-1)(7t^2-7t+3)\\
\overline{\varphi}\,(t) &= (t-1)(t-2)^2\\
\overline{\varphi}_0\,(t) &=\\
\Phi\,(x,y) &= t^5+3t^4+2t^3z+4t^3+5t^2z+3t^2\\
&\qquad +3tz^2+4tz+t+z^3+2z^2+z
\end{aligned}$$



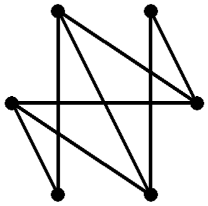
vert6_i50

$$\begin{aligned}
\varphi\,(t) &= \tfrac{1}{3}(t-1)(t-2)(14t-27)\\
\varphi_0\,(t) &= \tfrac{1}{3}(2t-1)(7t^2-7t+3)\\
\overline{\varphi}\,(t) &= (t-1)(t-2)^2\\
\overline{\varphi}_0\,(t) &=\\
\Phi\,(x,y) &= t^5+3t^4+2t^3z+4t^3+t^2z^2+4t^2z+3t^2\\
&\qquad +2tz^2+4tz+t+z^3+2z^2+z
\end{aligned}$$



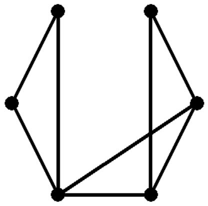
vert6_i51

$$\begin{aligned}
\varphi\,(t) &= 4(t-1)(t-2)(t-3)\\
\varphi_0\,(t) &= (2t-1)(2t^2-2t+1)\\
\overline{\varphi}\,(t) &= (t-1)(t-2)(t-3)\\
\overline{\varphi}_0\,(t) &= t^3\\
\Phi\,(x,y) &= t^5+3t^4+t^3z+5t^3+4t^2z+5t^2+2tz^2\\
&\qquad +6tz+2t+3z^2+z^3+2z
\end{aligned}$$



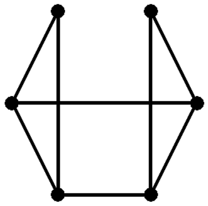
vert6_i52

$$\begin{aligned}
\varphi\,(t) &= 4(t-1)(t-2)(t-3)\\
\varphi_0\,(t) &= (2t-1)(2t^2-2t+1)\\
\overline{\varphi}\,(t) &= (t-1)(t-2)(t-3)\\
\overline{\varphi}_0\,(t) &= t^3\\
\Phi\,(x,y) &= t^5+3t^4+6t^3+5t^2z+2tz^2+6tz\\
&\qquad +5t^2+2t+z^3+3z^2+2z
\end{aligned}$$



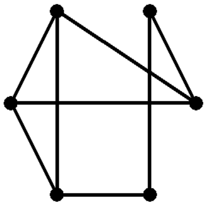
vert6_i53

$$\begin{aligned}
\varphi\,(t) &= 6(t-1)^2(t-2)\\
\varphi_0\,(t) &= (2t-1)(3t^2-3t+1)\\
\overline{\varphi}\,(t) &= (t-1)^2(t-2)\\
\overline{\varphi}_0\,(t) &= t^3\\
\Phi\,(x,y) &= (t^2+t+z)(t^3+2t^2+2tz+t+z^2+z)
\end{aligned}$$



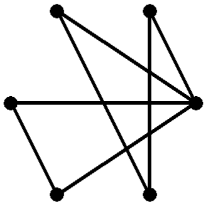
vert6_i54

$$\begin{aligned}
\varphi\,(t) &= \tfrac{1}{3}(t-1)(t-2)(14t-27)\\
\varphi_0\,(t) &= \tfrac{1}{3}(2t-1)(7t^2-7t+3)\\
\overline{\varphi}\,(t) &= (t-1)(t-2)^2\\
\overline{\varphi}_0\,(t) &= t^3\\
\Phi\,(x,y) &= t^5+3t^4+2t^3z+4t^3+5t^2z+3t^2\\
&\qquad +3tz^2+4tz+t+2z^2+z^3+z
\end{aligned}$$



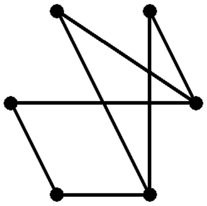
vert6_i55

$$\begin{aligned} \varphi\,(t) =&\ \ 4(t-1)(t-2)(t-3)\\ \varphi_0\,(t) =&\ \ (2t-1)(2t^2-2t+1)\\ \overline{\varphi}\,(t) =&\ \ (t-1)(t-2)(t-3)\\ \overline{\varphi}_0\,(t) =&\ \ t^3\\ \Phi\,(x,y) =&\ \ t^5+3t^4+2t^3z+4t^3+t^2z^2+3t^2z+4t^2\\ &\qquad +tz^2+5tz+2t+z^3+3z^2+2z \end{aligned}$$



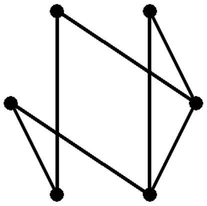
vert6_i56

$$\begin{aligned} \varphi\,(t) =&\ \ 4(t-1)^2\\ \varphi_0\,(t) =&\ \ (2t-1)^2\\ \overline{\varphi}\,(t) =&\ \ (t-1)^2\\ \overline{\varphi}_0\,(t) =&\ \ t^2\\ \Phi\,(x,y) =&\ \ (t^2+t+z)(t^3+t^2+t+z) \end{aligned}$$



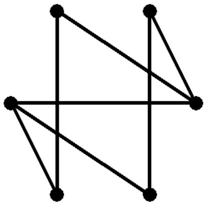
vert6_i57

$$\begin{aligned} \varphi\,(t) =&\ \ 3(t-1)(t-2)\\ \varphi_0\,(t) =&\ \ 3t^2-3t+1\\ \overline{\varphi}\,(t) =&\ \ (t-1)(t-2)\\ \overline{\varphi}_0\,(t) =&\ \ t^2\\ \Phi\,(x,y) =&\ \ t^5+2t^4+3t^3+t^2z+3t^2+3tz\\ &\qquad +t+z^2+z \end{aligned}$$



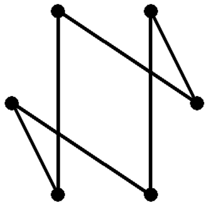
vert6_i58

$$\begin{aligned} \varphi\,(t) =&\ \ 3(t-1)(t-2)\\ \varphi_0\,(t) =&\ \ 3t^2-3t+1\\ \overline{\varphi}\,(t) =&\ \ (t-1)(t-2)\\ \overline{\varphi}_0\,(t) =&\ \ t^2\\ \Phi\,(x,y) =&\ \ t^5+2t^4+t^3z+2t^3+t^2z+2t^2+2tz\\ &\qquad +t+z^2+z \end{aligned}$$



vert6_i59

$$\begin{aligned} \varphi\,(t) =&\ \ 3(t-1)(t-2)\\ \varphi_0\,(t) =&\ \ 3t^2-3t+1\\ \overline{\varphi}\,(t) =&\ \ (t-1)(t-2)\\ \overline{\varphi}_0\,(t) =&\ \ t^2\\ \Phi\,(x,y) =&\ \ t^5+2t^4+3t^3+2t^2z+2t^2+2tz+t\\ &\qquad +z^2+z \end{aligned}$$



vert6_i60

$$\begin{aligned} \varphi\left(t\right) = &\quad 2\left(t-1\right) \\ \varphi_0\left(t\right) = &\quad 2t-1 \\ \overline{\varphi}\left(t\right) = &\quad t-1 \\ \overline{\varphi}_0\left(t\right) = &\quad t \\ \Phi\left(x,y\right) = &\quad t^5+t^4+t^3+t^2+t+z \end{aligned}$$

Appendix B

Generating Functions for Magic and Semi-magic 4-Squares

B.1 Magic 4-Squares by Cubical Count

If $\mathcal{C}_4(t)$ denotes the number of magic 4-squares with entries from $\{1, \ldots, t-1\}$, then the rational function representing $\sum_{t \geq 0} \mathcal{C}_4(t)\, z^t$ is equal to:

$32 \ast (-3211412 \ast z\hat{}533 - 22967012 \ast z\hat{}532 - 104775600 \ast z\hat{}531 - 373847822 \ast z\hat{}530$
$- 1144869440 \ast z\hat{}529 - 3136121578 \ast z\hat{}528 - 7901527168 \ast z\hat{}527 - 18619762038 \ast z\hat{}526$
$- 41546594686 \ast z\hat{}525 - 88537605566 \ast z\hat{}524 - 181413153278 \ast z\hat{}523$
$- 359221699928 \ast z\hat{}522 - 690247376746 \ast z\hat{}521 - 1291297835284 \ast z\hat{}520$
$- 2358435547724 \ast z\hat{}519 - 4214900518578 \ast z\hat{}518 - 7385192156126 \ast z\hat{}517$

$- 12707649173790 * z\hat{}516 - 21504095636652 * z\hat{}515 - 35831904242082 * z\hat{}514$
$- 58855496590326 * z\hat{}513 - 95387440225186 * z\hat{}512 - 152670539252582 * z\hat{}511$
$- 241496795397072 * z\hat{}510 - 377796888291116 * z\hat{}509 - 584877918452952 * z\hat{}508$
$- 896552696734208 * z\hat{}507 - 1361481148065544 * z\hat{}506 - 2049155528608666 * z\hat{}505$
$- 3058093029097068 * z\hat{}504 - 4526983989296130 * z\hat{}503 - 6649763574335616 * z\hat{}502$
$- 9695875402482962 * z\hat{}501 - 14037353612738518 * z\hat{}500 - 20184830048584312 * z\hat{}499$
$- 28835144860467476 * z\hat{}498 - 40933992358263550 * z\hat{}497$
$- 57757929238666274 * z\hat{}496 - 81021234544906546 * z\hat{}495$
$- 113014488272433334 * z\hat{}494 - 156783498864264964 * z\hat{}493$
$- 216359293532185186 * z\hat{}492 - 297052519452340858 * z\hat{}491$
$- 405828705002158962 * z\hat{}490 - 551784706184811378 * z\hat{}489$
$- 746751208702864190 * z\hat{}488 - 1006051777296095894 * z\hat{}487$
$- 1349455508368078172 * z\hat{}486 - 1802368381622834940 * z\hat{}485$
$- 2397317753259009950 * z\hat{}484 - 3175795777684842654 * z\hat{}483$
$- 4190540673645346596 * z\hat{}482 - 5508350549665954400 * z\hat{}481$
$- 7213542702924264986 * z\hat{}480 - 9412193031132339410 * z\hat{}479$
$- 12237315102939312414 * z\hat{}478 - 15855167938713805432 * z\hat{}477$
$- 20472915216115033652 * z\hat{}476 - 26347898205187191382 * z\hat{}475$
$- 33798829683926195422 * z\hat{}474 - 43219268579127173110 * z\hat{}473$
$- 55093794375987974502 * z\hat{}472 - 70017369133197220242 * z\hat{}471$
$- 88718452261785001248 * z\hat{}470 - 112086522344513662916 * z\hat{}469$
$- 141204759931195962676 * z\hat{}468 - 177388759384412426944 * z\hat{}467$
$- 222232264829418753260 * z\hat{}466 - 277661069795257729138 * z\hat{}465$
$- 345996380085661666142 * z\hat{}464 - 430029120376469550834 * z\hat{}463$
$- 533106864230934040638 * z\hat{}462 - 659235291231257599226 * z\hat{}461$
$- 813196320157759094514 * z\hat{}460 - 1000685341289965604776 * z\hat{}459$
$- 1228470269339691713122 * z\hat{}458 - 1504575470161683214318 * z\hat{}457$
$- 1838493973317224549952 * z\hat{}456 - 2241431779048228750044 * z\hat{}455$
$- 2726588494739390225854 * z\hat{}454 - 3309479004248581748614 * z\hat{}453$
$- 4008301374080093556430 * z\hat{}452 - 4844356746621403510236 * z\hat{}451$
$- 5842527550668936705026 * z\hat{}450 - 7031820988355200286098 * z\hat{}449$
$- 8445985420557297717134 * z\hat{}448 - 10124207986966859515240 * z\hat{}447$
$- 12111902543962711314668 * z\hat{}446 - 14461597802395826367782 * z\hat{}445$
$- 17233936376004589194330 * z\hat{}444 - 20498796330874081388378 * z\hat{}443$
$- 24336547729960423124828 * z\hat{}442 - 28839457618217203017536 * z\hat{}441$
$- 34113257860540880895220 * z\hat{}440 - 40278891253419750343424 * z\hat{}439$
$- 47474452342698703701934 * z\hat{}438 - 55857340424273288033102 * z\hat{}437$

$- 65606643235401430775496 * z\hat{}436 - 76925770895326873343254 * z\hat{}435$
$- 90045360670466944067150 * z\hat{}434 - 105226474159361394435012 * z\hat{}433$
$- 122764109452227973433360 * z\hat{}432 - 142991051762020147400384 * z\hat{}431$
$- 166282086874582016988210 * z\hat{}430 - 193058602573255225612188 * z\hat{}429$
$- 223793603873358924945756 * z\hat{}428 - 259017168509458765774190 * z\hat{}427$
$- 299322369556606759689496 * z\hat{}426 - 345371692398842796237404 * z\hat{}425$
$- 397903973374737749744378 * z\hat{}424 - 457741887403129833415318 * z\hat{}423$
$- 525800011599381259531454 * z\hat{}422 - 603093491417085125423510 * z\hat{}421$
$- 690747335054752830933734 * z\hat{}420 - 790006360848359573870504 * z\hat{}419$
$- 902245820976227267578182 * z\hat{}418 - 1028982723144753823509126 * z\hat{}417$
$- 1171887869833807098783862 * z\hat{}416 - 1332798632289320491039024 * z\hat{}415$
$- 1513732473573434710423970 * z\hat{}414 - 1716901231770375910138532 * z\hat{}413$
$- 1944726170698124602879690 * z\hat{}412 - 2199853801366722706544646 * z\hat{}411$
$- 2485172472735618525727584 * z\hat{}410 - 2803829725256945433420336 * z\hat{}409$
$- 3159250395013297418141982 * z\hat{}408 - 3555155450200430545161444 * z\hat{}407$
$- 3995581535013418678079090 * z\hat{}406 - 4484901188936720326814708 * z\hat{}405$
$- 5027843701741562752397098 * z\hat{}404 - 5629516556463910087584864 * z\hat{}403$
$- 6295427403980569788578656 * z\hat{}402 - 7031506503867689748176044 * z\hat{}401$
$- 7844129556701654969766636 * z\hat{}400 - 8740140843241014111703762 * z\hat{}399$
$- 9726876575673858768067974 * z\hat{}398 - 10812188355771134135069520 * z\hat{}397$
$- 12004466623992738361067964 * z\hat{}396 - 13312663972845625433587466 * z\hat{}395$
$- 14746318186704739118022106 * z\hat{}394 - 16315574859429626392023206 * z\hat{}393$
$- 18031209430025441992901802 * z\hat{}392 - 19904648465915791261478802 * z\hat{}391$
$- 21947990012666334234611362 * z\hat{}390 - 24174022818891151768420302 * z\hat{}389$
$- 26596244235100283187465770 * z\hat{}388 - 29228876576136670591130206 * z\hat{}387$
$- 32086881728072341768922336 * z\hat{}386 - 35185973772778994409970012 * z\hat{}385$
$- 38542629396276199395542000 * z\hat{}384 - 42174095841245197249390902 * z\hat{}383$
$- 46098396159140477929537566 * z\hat{}382 - 50334331514050548894968188 * z\hat{}381$
$- 54901480288152600473753672 * z\hat{}380 - 59820193738249126089210246 * z\hat{}379$
$- 65111587953695360349847562 * z\hat{}378 - 70797531869059546010095240 * z\hat{}377$
$- 76900631089250647967798334 * z\hat{}376 - 83444207291692204978604966 * z\hat{}375$
$- 90452272978474594927623254 * z\hat{}374 - 97949501362428058717909816 * z\hat{}373$
$- 105961191183685518088676886 * z\hat{}372 - 114513226268747457074182572 * z\hat{}371$
$- 123632029661174279172860450 * z\hat{}370 - 133344512173065690905912150 * z\hat{}369$
$- 143678015228180076408739458 * z\hat{}368 - 154660247892200244690154162 * z\hat{}367$
$- 166319218011880920771701354 * z\hat{}366 - 178683157413967584214321634 * z\hat{}365$
$- 191780441145341726230282416 * z\hat{}364 - 205639500769223137036501010 * z\hat{}363$

$- 220288731766796369912580944 * z\hat{}362 - 235756395130775213166696634 * z\hat{}361$
$- 252070513275387663801407456 * z\hat{}360 - 269258760427556649023952982 * z\hat{}359$
$- 287348347704738759102652022 * z\hat{}358 - 306365903127523724593204364 * z\hat{}357$
$- 326337346857619276230408634 * z\hat{}356 - 347287761995871103189827840 * z\hat{}355$
$- 369241261318283094241899416 * z\hat{}354 - 392220850372300463046233390 * z\hat{}353$
$- 416248287398547384571218070 * z\hat{}352 - 441343940586541646855370598 * z\hat{}351$
$- 467526643214159030669707706 * z\hat{}350 - 494813547261648708130274656 * z\hat{}349$
$- 523219976129200459657687692 * z\hat{}348 - 552759277124394390401670098 * z\hat{}347$
$- 583442674419600430965609808 * z\hat{}346 - 615279123211610731437809838 * z\hat{}345$
$- 648275165843653584707008182 * z\hat{}344 - 682434790675651397495396842 * z\hat{}343$
$- 717759294509223524723971170 * z\hat{}342 - 754247149391835909895462812 * z\hat{}341$
$- 791893874636656945950523964 * z\hat{}340 - 830691914903584416889543712 * z\hat{}339$
$- 870630525189485112197153192 * z\hat{}338 - 911695663574594091125084832 * z\hat{}337$
$- 953869892564119826067932854 * z\hat{}336 - 997132289852229247902942468 * z\hat{}335$
$- 1041458369316561258116000934 * z\hat{}334 - 1086820013028260409420582524 * z\hat{}333$
$- 1133185415032015478021738110 * z\hat{}332 - 1180519037615946955628127968 * z\hat{}331$
$- 1228781580749165839564415544 * z\hat{}330 - 1277929965318839396603287270 * z\hat{}329$
$- 1327917330745378044948724994 * z\hat{}328 - 1378693047497518432926953498 * z\hat{}327$
$- 1430202744965332654008523346 * z\hat{}326 - 1482388355082359241921679528 * z\hat{}325$
$- 1535188172014821898026887692 * z\hat{}324 - 1588536928160288261233554718 * z\hat{}323$
$- 1642365886616797413372044900 * z\hat{}322 - 1696602950200648440379502564 * z\hat{}321$
$- 1751172787003342720817571608 * z\hat{}320 - 1805996972389980021527426816 * z\hat{}319$
$- 1860994147249340440746727480 * z\hat{}318 - 1916080192214468361028660786 * z\hat{}317$
$- 1971168417478386046260979610 * z\hat{}316 - 2026169767737269263777776420 * z\hat{}315$
$- 2080993041699535978583571630 * z\hat{}314 - 2135545125508614054349282058 * z\hat{}313$
$- 2189731239336117168087957176 * z\hat{}312 - 2243455196315644689896710172 * z\hat{}311$
$- 2296619672901799425336053804 * z\hat{}310 - 2349126489659229072514348436 * z\hat{}309$
$- 2400876901408811973811673416 * z\hat{}308 - 2451771895587508465135164424 * z\hat{}307$
$- 2501712497611040177675505180 * z\hat{}306 - 2550600081969439268915036950 * z\hat{}305$
$- 2598336687730652911798022434 * z\hat{}304 - 2644825337081783539440160260 * z\hat{}303$
$- 2689970355497075814240573470 * z\hat{}302 - 2733677692091381132017482434 * z\hat{}301$
$- 2775855238693271470456586326 * z\hat{}300 - 2816413146158135345516391436 * z\hat{}299$
$- 2855264136434042442647463086 * z\hat{}298 - 2892323808896735403797118448 * z\hat{}297$
$- 2927510939480208038332085406 * z\hat{}296 - 2960747771150695590391093924 * z\hat{}295$
$- 2991960294299784128005419464 * z\hat{}294 - 3021078515671379973909992918 * z\hat{}293$
$- 3048036714482559560941750540 * z\hat{}292 - 3072773684454376585172606660 * z\hat{}291$
$- 3095232960530483488587821942 * z\hat{}290 - 3115363029133351066912555206 * z\hat{}289$

$$
\begin{aligned}
&-3133117520884757687317983702 * z\hat{}288 - 3148455384803384648865003100 * z\hat{}287\\
&- 3161341043082496830107481432 * z\hat{}286 - 3171744525649023856503042712 * z\hat{}285\\
&- 3179641583806495030938023374 * z\hat{}284 - 3185013782372345925412147466 * z\hat{}283\\
&- 3187848569829648206520829536 * z\hat{}282 - 3188139326128363165629282214 * z\hat{}281\\
&- 3185885387886287949770252374 * z\hat{}280 - 3181092050858934401203569840 * z\hat{}279\\
&- 3173770549665150401546972170 * z\hat{}278 - 3163938014875298459648089116 * z\hat{}277\\
&- 3151617407685766241141138522 * z\hat{}276 - 3136837432521405162946311898 * z\hat{}275\\
&- 3119632428020751063622983796 * z\hat{}274 - 3100042236971469743612108158 * z\hat{}273\\
&- 3078112055870050223403404786 * z\hat{}272 - 3053892264884238796722891330 * z\hat{}271\\
&- 3027438239093842077918442724 * z\hat{}270 - 2998810141979245591789467350 * z\hat{}269\\
&- 2968072702212211323975665698 * z\hat{}268 - 2935294974884183336753716844 * z\hat{}267\\
&- 2900550088378524184014450506 * z\hat{}266 - 2863914978158817951212590452 * z\hat{}265\\
&- 2825470108800870666155382946 * z\hat{}264 - 2785299185645354292195789642 * z\hat{}263\\
&- 2743488857486637017686476956 * z\hat{}262 - 2700128411745308087316438548 * z\hat{}261\\
&- 2655309463592909118655691590 * z\hat{}260 - 2609125640511560786352753210 * z\hat{}259\\
&- 2561672263774407486641653702 * z\hat{}258 - 2513046028329269236817959702 * z\hat{}257\\
&- 2463344682553688449055197812 * z\hat{}256 - 2412666709328896541112614040 * z\hat{}255\\
&- 2361111009849412319272892770 * z\hat{}254 - 2308776591548245188965444000 * z\hat{}253\\
&- 2255762261471526564566298610 * z\hat{}252 - 2202166326385057449274646564 * z\hat{}251\\
&- 2148086300835447066733160308 * z\hat{}250 - 2093618624324433017599020040 * z\hat{}249\\
&- 2038858388683504545684609580 * z\hat{}248 - 1983899076661263386418698800 * z\hat{}247\\
&- 1928832312655100480244515452 * z\hat{}246 - 1873747626435845335335513890 * z\hat{}245\\
&- 1818732230626251226261383326 * z\hat{}244 - 1763870812605532203251113030 * z\hat{}243\\
&- 1709245341419985481210112534 * z\hat{}242 - 1654934890187925781319571372 * z\hat{}241\\
&- 1601015474393151342107476118 * z\hat{}240 - 1547559906368736004641274882 * z\hat{}239\\
&- 1494637666179610684721443978 * z\hat{}238 - 1442314789021793324585270478 * z\hat{}237\\
&- 1390653769165832840021104160 * z\hat{}236 - 1339713480385516228532223390 * z\hat{}235\\
&- 1289549112727788787412879234 * z\hat{}234 - 1240212125399441538298537390 * z\hat{}233\\
&- 1191750215468110776198787188 * z\hat{}232 - 1144207302002606978860711610 * z\hat{}231\\
&- 1097623525208260950888176482 * z\hat{}230 - 1052035260049739752990894494 * z\hat{}229\\
&- 1007475143794387010282237950 * z\hat{}228 - 963972116856274255557958212 * z\hat{}227\\
&- 921551476272577718296865092 * z\hat{}226 - 880234941102107593271273744 * z\hat{}225\\
&- 840040728998601216537081312 * z\hat{}224 - 800983643181036341871620558 * z\hat{}223\\
&- 763075168997535580006695662 * z\hat{}222 - 726323579260512245936569578 * z\hat{}221\\
&- 690734047516387880090006884 * z\hat{}220 - 656308768405401351473683040 * z\hat{}219\\
&- 623047084263598208532251322 * z\hat{}218 - 590945617121771409538633918 * z\hat{}217\\
&- 559998405262831036315244638 * z\hat{}216 - 530197043511384693729708466 * z\hat{}215
\end{aligned}
$$

$- 5015308264451720747102727788 \ast z\hat{}214 - 4739868937388438271190103004 \ast z\hat{}213$
$- 4475503768744259689108142774 \ast z\hat{}212 - 4222045464809627331954564486 \ast z\hat{}211$
$- 3979309595963719581251537700 \ast z\hat{}210 - 3747096061786585935262107712 \ast z\hat{}209$
$- 3525190542294818118225086604 \ast z\hat{}208 - 3313365929317402399049369956 \ast z\hat{}207$
$- 3111383732425844507670964438 \ast z\hat{}206 - 2918995454250815272174656686 \ast z\hat{}205$
$- 2735943930439959873012534402 \ast z\hat{}204 - 2561964629947387453375267706 \ast z\hat{}203$
$- 2396786911779267209410326696 \ast z\hat{}202 - 2240135234760554534326995538 \ast z\hat{}201$
$- 2091730317320792920167557748 \ast z\hat{}200 - 1951290244730357123233477716 \ast z\hat{}199$
$- 1818531521639171079105469992 \ast z\hat{}198 - 1693170068185636298937351114 \ast z\hat{}197$
$- 1574922158342122601880551146 \ast z\hat{}196 - 1463505299552376986703807774 \ast z\hat{}195$
$- 1358639053084604727756492220 \ast z\hat{}194 - 1260045794878987825920733334 \ast z\hat{}193$
$- 1167451417000143498078897786 \ast z\hat{}192 - 1080585970120492558066804428 \ast z\hat{}191$
$- 999184247750889228450509994 \ast z\hat{}190 - 922986313206972292064044418 \ast z\hat{}189$
$- 851737970545738830191296624 \ast z\hat{}188 - 785191180933447409544848806 \ast z\hat{}187$
$- 723104426106293258967565560 \ast z\hat{}186 - 665243020765812864356794496 \ast z\hat{}185$
$- 611379375905723072255659912 \ast z\hat{}184 - 561293215202136284266300074 \ast z\hat{}183$
$- 514771746709681746795720096 \ast z\hat{}182 - 471609792198047965030984450 \ast z\hat{}181$
$- 431609876532432895429232262 \ast z\hat{}180 - 394582279553157477523752246 \ast z\hat{}179$
$- 360345052940496514886121172 \ast z\hat{}178 - 328724004566160519064283328 \ast z\hat{}177$
$- 299552652829613705964766690 \ast z\hat{}176 - 272672153460803874451070096 \ast z\hat{}175$
$- 247931201238177691480192204 \ast z\hat{}174 - 225185909027018630082095562 \ast z\hat{}173$
$- 204299666485843465859880080 \ast z\hat{}172 - 185142980722456509852842238 \ast z\hat{}171$
$- 167593301104287816342207708 \ast z\hat{}170 - 151534830344116583282045598 \ast z\hat{}169$
$- 136858323890469606119049944 \ast z\hat{}168 - 123460879555814329502779948 \ast z\hat{}167$
$- 111245719213613089103610012 \ast z\hat{}166 - 100121964290934507877260008 \ast z\hat{}165$
$- 90004406675268898065928810 \ast z\hat{}164 - 80813276545669475257743880 \ast z\hat{}163$
$- 72474008528102619259234408 \ast z\hat{}162 - 64917007465784292513156660 \ast z\hat{}161$
$- 58077414986159794102196636 \ast z\hat{}160 - 51894877939550743919485514 \ast z\hat{}159$
$- 46313319679267272483509952 \ast z\hat{}158 - 41280715051335483394342230 \ast z\hat{}157$
$- 36748869862890319427796688 \ast z\hat{}156 - 32673205503592659628786694 \ast z\hat{}155$
$- 29012549303154099908994416 \ast z\hat{}154 - 25728931121767061824618858 \ast z\hat{}153$
$- 22787386587979430619428806 \ast z\hat{}152 - 20155767321633700033350078 \ast z\hat{}151$
$- 17804558406976064820931182 \ast z\hat{}150 - 15706703314007902391263376 \ast z\hat{}149$
$- 13837436403694011678285590 \ast z\hat{}148 - 12174123095628039509587734 \ast z\hat{}147$
$- 10696107724309595967923318 \ast z\hat{}146 - 9384569062998401875128878 \ast z\hat{}145$
$- 8222383451336750969239994 \ast z\hat{}144 - 7193995425107355405561174 \ast z\hat{}143$
$- 6285295712816734089992442 \ast z\hat{}142 - 5483506434670461087194428 \ast z\hat{}141$

$- 4777073314177262294397 44 \ast z\hat{}140 - 4155564691384091241766 44 \ast z\hat{}139$
$- 3609577108893849411689 38 \ast z\hat{}138 - 3130647227566561346771 80 \ast z\hat{}137$
$- 2711169817495733258438 64 \ast z\hat{}136 - 2344321561633471167653 38 \ast z\hat{}135$
$- 2023990403707958657026 44 \ast z\hat{}134 - 1744710168934391048374 94 \ast z\hat{}133$
$- 1501600184919488317705 74 \ast z\hat{}132 - 1290309631163455489464 90 \ast z\hat{}131$
$- 1106966348183158836152 12 \ast z\hat{}130 - 948129841554774104214 66 \ast z\hat{}129$
$- 810748221671503213091 10 \ast z\hat{}128 - 692118826754928302191 10 \ast z\hat{}127$
$- 589852284270668670710 20 \ast z\hat{}126 - 501839774394414441789 28 \ast z\hat{}125$
$- 426223268229334991471 40 \ast z\hat{}124 - 361368523101126838743 44 \ast z\hat{}123$
$- 305840627178226941005 70 \ast z\hat{}122 - 258381895893104547970 82 \ast z\hat{}121$
$- 217891932947303048298 94 \ast z\hat{}120 - 183409679081825896311 40 \ast z\hat{}119$
$- 154097282098134102085 64 \ast z\hat{}118 - 129225631841187088647 68 \ast z\hat{}117$
$- 108161413854064041594 82 \ast z\hat{}116 - 90355545205900647472 14 \ast z\hat{}115$
$- 75332865462925176871 88 \ast z\hat{}114 - 62682964942693986257 02 \ast z\hat{}113$
$- 52052041173798593814 00 \ast z\hat{}112 - 43135682907449707300 70 \ast z\hat{}111$
$- 35672489026562941352 48 \ast z\hat{}110 - 29438437305241135539 28 \ast z\hat{}109$
$- 24241925139508951591 68 \ast z\hat{}108 - 19919411133670298139 48 \ast z\hat{}107$
$- 16331592754211314877 20 \ast z\hat{}106 - 13360061190532865973 52 \ast z\hat{}105$
$- 10904380070006852829 44 \ast z\hat{}104 - 8879539799656955477 32 \ast z\hat{}103$
$- 7213744038882207168 38 \ast z\hat{}102 - 5846489181961195589 60 \ast z\hat{}101$
$- 4726901743522904474 02 \ast z\hat{}100 - 3812302228083236750 50 \ast z\hat{}99$
$- 3066967429195689888 96 \ast z\hat{}98 - 2461066175992353912 24 \ast z\hat{}97$
$- 1969746331419787439 88 \ast z\hat{}96 - 1572353376062434530 74 \ast z\hat{}95$
$- 1251763192721648860 52 \ast z\hat{}94 - 993813725912878717 26 \ast z\hat{}93$
$- 786822036846620893 58 \ast z\hat{}92 - 621174931465807773 18 \ast z\hat{}91$
$- 488982816694632899 86 \ast z\hat{}90 - 383787758569986031 92 \ast z\hat{}89$
$- 300317885076968577 18 \ast z\hat{}88 - 234281313904628778 54 \ast z\hat{}87$
$- 182193700034520881 78 \ast z\hat{}86 - 141234305095172775 90 \ast z\hat{}85$
$- 109126197948564842 78 \ast z\hat{}84 - 84036816660399816 16 \ast z\hat{}83$
$- 64495663053833146 04 \ast z\hat{}82 - 49326372949324098 50 \ast z\hat{}81$
$- 37590814153873341 36 \ast z\hat{}80 - 28543218877966111 06 \ast z\hat{}79$
$- 21592662754299963 56 \ast z\hat{}78 - 16272465959917429 86 \ast z\hat{}77$
$- 12215317424658944 46 \ast z\hat{}76 - 9133116290334854 86 \ast z\hat{}75$
$- 6800689177665049 02 \ast z\hat{}74 - 5042681790400946 44 \ast z\hat{}73$
$- 3723041741851900 20 \ast z\hat{}72 - 2736609623676760 14 \ast z\hat{}71$
$- 2002419453430739 90 \ast z\hat{}70 - 1458380339473014 64 \ast z\hat{}69$
$- 1057070193000901 62 \ast z\hat{}68 - 762421567034177 68 \ast z\hat{}67 - 547120507300327 28 \ast z\hat{}66$

$- 39057313000479050 * z\hat{}65 - 27732246752986190 * z\hat{}64 - 19582102627151858 * z\hat{}63$
$- 13748319710478410 * z\hat{}62 - 9595693234501556 * z\hat{}61 - 6656647283453344 * z\hat{}60$
$- 4588793496946524 * z\hat{}59 - 3142762079855868 * z\hat{}58 - 2137938856055428 * z\hat{}57$
$- 1444257636798088 * z\hat{}56 - 968608085175728 * z\hat{}55 - 644743482283738 * z\hat{}54$
$- 425828902657602 * z\hat{}53 - 278970444306112 * z\hat{}52 - 181222746475682 * z\hat{}51$
$- 116693152304692 * z\hat{}50 - 74454694216124 * z\hat{}49 - 47051865906236 * z\hat{}48$
$- 29438157506462 * z\hat{}47 - 18225938613006 * z\hat{}46 - 11160798204014 * z\hat{}45$
$- 6755997820380 * z\hat{}44 - 4040320464406 * z\hat{}43 - 2385587358652 * z\hat{}42$
$- 1389711553854 * z\hat{}41 - 798128367696 * z\hat{}40 - 451520550514 * z\hat{}39$
$-251386284678*z\hat{}38-137602979952*z\hat{}37-73969612222*z\hat{}36-39001695498*z\hat{}35$
$- 20142939414 * z\hat{}34 - 10174367958 * z\hat{}33 - 5017538502 * z\hat{}32 - 2411195934 * z\hat{}31$
$- 1126615868 * z\hat{}30 - 510542752 * z\hat{}29 - 223736200 * z\hat{}28 - 94498774 * z\hat{}27$
$- 38313008 * z\hat{}26 - 14839024 * z\hat{}25 - 5456530 * z\hat{}24 - 1890252 * z\hat{}23 - 609864 * z\hat{}22$
$-180436*z\hat{}21-47636*z\hat{}20-10808*z\hat{}19-1906*z\hat{}18-220*z\hat{}17)/(z\hat{}533+5*z\hat{}532$
$+ 18 * z\hat{}531 + 52 * z\hat{}530 + 133 * z\hat{}529 + 308 * z\hat{}528 + 666 * z\hat{}527 + 1358 * z\hat{}526$
$+2644 * z\hat{}525 + 4944 * z\hat{}524 + 8937 * z\hat{}523 + 15674 * z\hat{}522 + 26777 * z\hat{}521$
$+ 44670 * z\hat{}520 + 72955 * z\hat{}519 + 116858 * z\hat{}518 + 183908 * z\hat{}517 + 284748 * z\hat{}516$
$+434315*z\hat{}515+653248*z\hat{}514+969859*z\hat{}513+1422483*z\hat{}512+2062659*z\hat{}511$
$+ 2958906 * z\hat{}510 + 4201700 * z\hat{}509 + 5909341 * z\hat{}508 + 8235506 * z\hat{}507$
$+ 11378092 * z\hat{}506 + 15590359 * z\hat{}505 + 21193871 * z\hat{}504 + 28594504 * z\hat{}503$
$+ 38300869 * z\hat{}502 + 50946713 * z\hat{}501 + 67316465 * z\hat{}500 + 88375798 * z\hat{}499$
$+ 115306141 * z\hat{}498 + 149545350 * z\hat{}497 + 192833153 * z\hat{}496 + 247263925 * z\hat{}495$
$+ 315345013 * z\hat{}494 + 400063487 * z\hat{}493 + 504959034 * z\hat{}492 + 634206177 * z\hat{}491$
$+ 792702882 * z\hat{}490 + 986168993 * z\hat{}489 + 1221250765 * z\hat{}488 + 1505635103 * z\hat{}487$
$+1848168812*z\hat{}486+2258986559*z\hat{}485+2749641679*z\hat{}484+3333243516*z\hat{}483$
$+4024594082*z\hat{}482+4840327582*z\hat{}481+5799044055*z\hat{}480+6921440448*z\hat{}479$
$+ 8230428658 * z\hat{}478 + 9751243544 * z\hat{}477 + 11511528631 * z\hat{}476$
$+ 13541402129 * z\hat{}475 + 15873489130 * z\hat{}474 + 18542922264 * z\hat{}473$
$+ 21587294847 * z\hat{}472 + 25046568567 * z\hat{}471 + 28962918101 * z\hat{}470$
$+ 33380514671 * z\hat{}469 + 38345229613 * z\hat{}468 + 43904260295 * z\hat{}467$
$+ 50105658613 * z\hat{}466 + 56997765222 * z\hat{}465 + 64628529581 * z\hat{}464$
$+ 73044720422 * z\hat{}463 + 82291007438 * z\hat{}462 + 92408921065 * z\hat{}461$
$+ 103435672903 * z\hat{}460 + 115402846838 * z\hat{}459 + 128334946377 * z\hat{}458$
$+ 142247812448 * z\hat{}457 + 157146901451 * z\hat{}456 + 173025443062 * z\hat{}455$
$+ 189862473194 * z\hat{}454 + 207620767864 * z\hat{}453 + 226244680305 * z\hat{}452$
$+ 245657914173 * z\hat{}451 + 265761243272 * z\hat{}450 + 286430218356 * z\hat{}449$
$+ 307512880394 * z\hat{}448 + 328827528777 * z\hat{}447 + 350160573293 * z\hat{}446$

$+ 371264525974 * z\hat{}445 + 391856171007 * z\hat{}444 + 411614975496 * z\hat{}443$
$+ 430181787910 * z\hat{}442 + 447157891982 * z\hat{}441 + 462104470030 * z\hat{}440$
$+ 474542545920 * z\hat{}439 + 483953466425 * z\hat{}438 + 489779990225 * z\hat{}437$
$+ 491428044893 * z\hat{}436 + 488269215832 * z\hat{}435 + 479644025038 * z\hat{}434$
$+ 464866053306 * z\hat{}433 + 443226956495 * z\hat{}432 + 414002413357 * z\hat{}431$
$+ 376459042988 * z\hat{}430 + 329862307119 * z\hat{}429 + 273485417096 * z\hat{}428$
$+ 206619232245 * z\hat{}427 + 128583145669 * z\hat{}426 + 38736909621 * z\hat{}425$
$- 63506632387 * z\hat{}424 - 178667994435 * z\hat{}423 - 307186743989 * z\hat{}422$
$- 449406576180 * z\hat{}421 - 605559868520 * z\hat{}420 - 775751893819 * z\hat{}419$
$- 959944830079 * z\hat{}418 - 1157941791469 * z\hat{}417 - 1369371054580 * z\hat{}416$
$- 1593670747777 * z\hat{}415 - 1830074209074 * z\hat{}414 - 2077596318529 * z\hat{}413$
$- 2335021034496 * z\hat{}412 - 2600890469090 * z\hat{}411 - 2873495745761 * z\hat{}410$
$- 3150869991488 * z\hat{}409 - 3430783706539 * z\hat{}408 - 3710742866264 * z\hat{}407$
$- 3987989981488 * z\hat{}406 - 4259508455919 * z\hat{}405 - 4522030432169 * z\hat{}404$
$- 4772048428533 * z\hat{}403 - 5005830902949 * z\hat{}402 - 5219441988709 * z\hat{}401$
$- 5408765462309 * z\hat{}400 - 5569533108918 * z\hat{}399 - 5697357449768 * z\hat{}398$
$- 5787768897384 * z\hat{}397 - 5836257188749 * z\hat{}396 - 5838317045030 * z\hat{}395$
$- 5789497778999 * z\hat{}394 - 5685456667716 * z\hat{}393 - 5522015673005 * z\hat{}392$
$- 5295221187878 * z\hat{}391 - 5001406249447 * z\hat{}390 - 4637254755722 * z\hat{}389$
$- 4199866988924 * z\hat{}388 - 3686825848467 * z\hat{}387 - 3096262971075 * z\hat{}386$
$- 2426924022082 * z\hat{}385 - 1678232231929 * z\hat{}384 - 850349367171 * z\hat{}383$
$+ 55766862682 * z\hat{}382 + 1038309834735 * z\hat{}381 + 2094576510458 * z\hat{}380$
$+ 3220926294142 * z\hat{}379 + 4412747269792 * z\hat{}378 + 5664430680990 * z\hat{}377$
$+ 6969354604379 * z\hat{}376 + 8319877600188 * z\hat{}375 + 9707343166429 * z\hat{}374$
$+ 11122095643539 * z\hat{}373 + 12553508214204 * z\hat{}372 + 13990023453273 * z\hat{}371$
$+ 15419206834890 * z\hat{}370 + 16827813408464 * z\hat{}369 + 18201867762680 * z\hat{}368$
$+ 19526757201631 * z\hat{}367 + 20787337923112 * z\hat{}366 + 21968053801836 * z\hat{}365$
$+ 23053067209599 * z\hat{}364 + 24026401133889 * z\hat{}363 + 24872091655426 * z\hat{}362$
$+ 25574349701448 * z\hat{}361 + 26117730768117 * z\hat{}360 + 26487311199329 * z\hat{}359$
$+ 26668869372214 * z\hat{}358 + 26649070082158 * z\hat{}357 + 26415650178927 * z\hat{}356$
$+ 25957603507419 * z\hat{}355 + 25265362970430 * z\hat{}354 + 24330977601983 * z\hat{}353$
$+ 23148282317481 * z\hat{}352 + 21713058155702 * z\hat{}351 + 20023180626772 * z\hat{}350$
$+ 18078754012784 * z\hat{}349 + 15882229294162 * z\hat{}348 + 13438503694706 * z\hat{}347$
$+ 10754999695569 * z\hat{}346 + 7841721776045 * z\hat{}345 + 4711289028271 * z\hat{}344$
$+ 1378942285219 * z\hat{}343 - 2137475679262 * z\hat{}342 - 5817567749355 * z\hat{}341$
$- 9638458791509 * z\hat{}340 - 13574900992399 * z\hat{}339 - 17599408536785 * z\hat{}338$
$- 21682421259654 * z\hat{}337 - 25792496937358 * z\hat{}336 - 29896531153626 * z\hat{}335$

$- 33960003711805 * z\hat{}334 - 37947249817939 * z\hat{}333 - 41821754313791 * z\hat{}332$
$- 45546466495415 * z\hat{}331 - 49084133140918 * z\hat{}330 - 52397646652482 * z\hat{}329$
$- 55450405354014 * z\hat{}328 - 58206682313551 * z\hat{}327 - 60631999257413 * z\hat{}326$
$- 62693501537743 * z\hat{}325 - 64360330385659 * z\hat{}324 - 65603988161712 * z\hat{}323$
$- 66398692667235 * z\hat{}322 - 66721716159094 * z\hat{}321 - 66553705147937 * z\hat{}320$
$- 65878976749440 * z\hat{}319 - 64685787881194 * z\hat{}318 - 62966573403643 * z\hat{}317$
$- 60718149907617 * z\hat{}316 - 57941881773826 * z\hat{}315 - 54643806804689 * z\hat{}314$
$- 50834718764631 * z\hat{}313 - 46530204895683 * z\hat{}312 - 41750636614156 * z\hat{}311$
$- 36521112361224 * z\hat{}310 - 30871351807812 * z\hat{}309 - 24835541393796 * z\hat{}308$
$- 18452131479592 * z\hat{}307 - 11763586153451 * z\hat{}306 - 4816087084617 * z\hat{}305$
$+ 2340806464985 * z\hat{}304 + 9654538985986 * z\hat{}303 + 17069978288905 * z\hat{}302$
$+ 24529869785753 * z\hat{}301 + 31975321428742 * z\hat{}300 + 39346314898580 * z\hat{}299$
$+ 46582238212434 * z\hat{}298 + 53622434615627 * z\hat{}297 + 60406762348574 * z\hat{}296$
$+ 66876159636683 * z\hat{}295 + 72973209146204 * z\hat{}294 + 78642695981398 * z\hat{}293$
$+ 83832153373069 * z\hat{}292 + 88492390125590 * z\hat{}291 + 92577994148974 * z\hat{}290$
$+ 96047806406460 * z\hat{}289 + 98865360050337 * z\hat{}288 + 100999279605798 * z\hat{}287$
$+ 102423635678086 * z\hat{}286 + 103118250818462 * z\hat{}285 + 103068952956973 * z\hat{}284$
$+ 102267773027396 * z\hat{}283 + 100713084316416 * z\hat{}282 + 98409681322938 * z\hat{}281$
$+ 95368796925851 * z\hat{}280 + 91608056924151 * z\hat{}279 + 87151372099146 * z\hat{}278$
$+ 82028768198241 * z\hat{}277 + 76276155363889 * z\hat{}276 + 69935038736925 * z\hat{}275$
$+ 63052173089466 * z\hat{}274 + 55679164476130 * z\hat{}273 + 47872022984707 * z\hat{}272$
$+ 39690670702990 * z\hat{}271 + 31198410044027 * z\hat{}270 + 22461357487590 * z\hat{}269$
$+ 13547848724938 * z\hat{}268 + 4527820972324 * z\hat{}267 - 4527820972324 * z\hat{}266$
$- 13547848724938 * z\hat{}265 - 22461357487590 * z\hat{}264 - 31198410044027 * z\hat{}263$
$- 39690670702990 * z\hat{}262 - 47872022984707 * z\hat{}261 - 55679164476130 * z\hat{}260$
$- 63052173089466 * z\hat{}259 - 69935038736925 * z\hat{}258 - 76276155363889 * z\hat{}257$
$- 82028768198241 * z\hat{}256 - 87151372099146 * z\hat{}255 - 91608056924151 * z\hat{}254$
$- 95368796925851 * z\hat{}253 - 98409681322938 * z\hat{}252 - 100713084316416 * z\hat{}251$
$- 102267773027396 * z\hat{}250 - 103068952956973 * z\hat{}249 - 103118250818462 * z\hat{}248$
$- 102423635678086 * z\hat{}247 - 100999279605798 * z\hat{}246 - 98865360050337 * z\hat{}245$
$- 96047806406460 * z\hat{}244 - 92577994148974 * z\hat{}243 - 88492390125590 * z\hat{}242$
$- 83832153373069 * z\hat{}241 - 78642695981398 * z\hat{}240 - 72973209146204 * z\hat{}239$
$- 66876159636683 * z\hat{}238 - 60406762348574 * z\hat{}237 - 53622434615627 * z\hat{}236$
$- 46582238212434 * z\hat{}235 - 39346314898580 * z\hat{}234 - 31975321428742 * z\hat{}233$
$-24529869785753 * z\hat{}232 - 17069978288905 * z\hat{}231 - 9654538985986 * z\hat{}230$
$- 2340806464985 * z\hat{}229 + 4816087084617 * z\hat{}228 + 11763586153451 * z\hat{}227$
$+ 18452131479592 * z\hat{}226 + 24835541393796 * z\hat{}225 + 30871351807812 * z\hat{}224$

$+ 36521112361224 * z\hat{}223 + 41750636614156 * z\hat{}222 + 46530204895683 * z\hat{}221$
$+ 50834718764631 * z\hat{}220 + 54643806804689 * z\hat{}219 + 57941881773826 * z\hat{}218$
$+ 60718149907617 * z\hat{}217 + 62966573403643 * z\hat{}216 + 64685787881194 * z\hat{}215$
$+ 65878976749440 * z\hat{}214 + 66553705147937 * z\hat{}213 + 66721716159094 * z\hat{}212$
$+ 66398692667235 * z\hat{}211 + 65603988161712 * z\hat{}210 + 64360330385659 * z\hat{}209$
$+ 62693501537743 * z\hat{}208 + 60631999257413 * z\hat{}207 + 58206682313551 * z\hat{}206$
$+ 55450405354014 * z\hat{}205 + 52397646652482 * z\hat{}204 + 49084133140918 * z\hat{}203$
$+ 45546466495415 * z\hat{}202 + 41821754313791 * z\hat{}201 + 37947249817939 * z\hat{}200$
$+ 33960003711805 * z\hat{}199 + 29896531153626 * z\hat{}198 + 25792496937358 * z\hat{}197$
$+ 21682421259654 * z\hat{}196 + 17599408536785 * z\hat{}195 + 13574900992399 * z\hat{}194$
$+ 9638458791509 * z\hat{}193 + 5817567749355 * z\hat{}192 + 2137475679262 * z\hat{}191$
$- 1378942285219 * z\hat{}190 - 4711289028271 * z\hat{}189 - 7841721776045 * z\hat{}188$
$- 10754999695569 * z\hat{}187 - 13438503694706 * z\hat{}186 - 15882229294162 * z\hat{}185$
$- 18078754012784 * z\hat{}184 - 20023180626772 * z\hat{}183 - 21713058155702 * z\hat{}182$
$- 23148282317481 * z\hat{}181 - 24330977601983 * z\hat{}180 - 25265362970430 * z\hat{}179$
$- 25957603507419 * z\hat{}178 - 26415650178927 * z\hat{}177 - 26649070082158 * z\hat{}176$
$- 26668869372214 * z\hat{}175 - 26487311199329 * z\hat{}174 - 26117730768117 * z\hat{}173$
$- 25574349701448 * z\hat{}172 - 24872091655426 * z\hat{}171 - 24026401133889 * z\hat{}170$
$- 23053067209599 * z\hat{}169 - 21968053801836 * z\hat{}168 - 20787337923112 * z\hat{}167$
$- 19526757201631 * z\hat{}166 - 18201867762680 * z\hat{}165 - 16827813408464 * z\hat{}164$
$- 15419206834890 * z\hat{}163 - 13990023453273 * z\hat{}162 - 12553508214204 * z\hat{}161$
$- 11122095643539 * z\hat{}160 - 9707343166429 * z\hat{}159 - 8319877600188 * z\hat{}158$
$- 6969354604379 * z\hat{}157 - 5664430680990 * z\hat{}156 - 4412747269792 * z\hat{}155$
$- 3220926294142 * z\hat{}154 - 2094576510458 * z\hat{}153 - 1038309834735 * z\hat{}152$
$- 55766862682 * z\hat{}151 + 850349367171 * z\hat{}150 + 1678232231929 * z\hat{}149$
$+ 2426924022082 * z\hat{}148 + 3096262971075 * z\hat{}147 + 3686825848467 * z\hat{}146$
$+ 4199866988924 * z\hat{}145 + 4637254755722 * z\hat{}144 + 5001406249447 * z\hat{}143$
$+ 5295221187878 * z\hat{}142 + 5522015673005 * z\hat{}141 + 5685456667716 * z\hat{}140$
$+ 5789497778999 * z\hat{}139 + 5838317045030 * z\hat{}138 + 5836257188749 * z\hat{}137$
$+ 5787768897384 * z\hat{}136 + 5697357449768 * z\hat{}135 + 5569533108918 * z\hat{}134$
$+ 5408765462309 * z\hat{}133 + 5219441988709 * z\hat{}132 + 5005830902949 * z\hat{}131$
$+ 4772048428533 * z\hat{}130 + 4522030432169 * z\hat{}129 + 4259508455919 * z\hat{}128$
$+ 3987989981488 * z\hat{}127 + 3710742866264 * z\hat{}126 + 3430783706539 * z\hat{}125$
$+ 3150869991488 * z\hat{}124 + 2873495745761 * z\hat{}123 + 2600890469090 * z\hat{}122$
$+ 2335021034496 * z\hat{}121 + 2077596318529 * z\hat{}120 + 1830074209074 * z\hat{}119$
$+ 1593670747777 * z\hat{}118 + 1369371054580 * z\hat{}117 + 1157941791469 * z\hat{}116$
$+ 959944830079 * z\hat{}115 + 775751893819 * z\hat{}114 + 605559868520 * z\hat{}113$

$$
\begin{aligned}
&+ 449406576180 * z\hat{}112 + 307186743989 * z\hat{}111 + 178667994435 * z\hat{}110\\
&+ 63506632387 * z\hat{}109 - 38736909621 * z\hat{}108 - 128583145669 * z\hat{}107\\
&- 206619232245 * z\hat{}106 - 273485417096 * z\hat{}105 - 329862307119 * z\hat{}104\\
&- 376459042988 * z\hat{}103 - 414002413357 * z\hat{}102 - 443226956495 * z\hat{}101\\
&- 464866053306 * z\hat{}100 - 479644025038 * z\hat{}99 - 488269215832 * z\hat{}98\\
&- 491428044893 * z\hat{}97 - 489779990225 * z\hat{}96 - 483953466425 * z\hat{}95\\
&- 474542545920 * z\hat{}94 - 462104470030 * z\hat{}93 - 447157891982 * z\hat{}92\\
&- 430181787910 * z\hat{}91 - 411614975496 * z\hat{}90 - 391856171007 * z\hat{}89\\
&- 371264525974 * z\hat{}88 - 350160573293 * z\hat{}87 - 328827528777 * z\hat{}86\\
&- 307512880394 * z\hat{}85 - 286430218356 * z\hat{}84 - 265761243272 * z\hat{}83\\
&- 245657914173 * z\hat{}82 - 226244680305 * z\hat{}81 - 207620767864 * z\hat{}80\\
&- 189862473194 * z\hat{}79 - 173025443062 * z\hat{}78 - 157146901451 * z\hat{}77\\
&- 142247812448 * z\hat{}76 - 128334946377 * z\hat{}75 - 115402846838 * z\hat{}74\\
&- 103435672903 * z\hat{}73 - 92408921065 * z\hat{}72 - 82291007438 * z\hat{}71 - 73044720422 * z\hat{}70\\
&- 64628529581 * z\hat{}69 - 56997765222 * z\hat{}68 - 50105658613 * z\hat{}67 - 43904260295 * z\hat{}66\\
&- 38345229613 * z\hat{}65 - 33380514671 * z\hat{}64 - 28962918101 * z\hat{}63 - 25046568567 * z\hat{}62\\
&- 21587294847 * z\hat{}61 - 18542922264 * z\hat{}60 - 15873489130 * z\hat{}59 - 13541402129 * z\hat{}58\\
&- 11511528631 * z\hat{}57 - 9751243544 * z\hat{}56 - 8230428658 * z\hat{}55 - 6921440448 * z\hat{}54\\
&- 5799044055 * z\hat{}53 - 4840327582 * z\hat{}52 - 4024594082 * z\hat{}51 - 3333243516 * z\hat{}50\\
&- 2749641679 * z\hat{}49 - 2258986559 * z\hat{}48 - 1848168812 * z\hat{}47 - 1505635103 * z\hat{}46\\
&- 1221250765 * z\hat{}45 - 986168993 * z\hat{}44 - 792702882 * z\hat{}43 - 634206177 * z\hat{}42\\
&- 504959034 * z\hat{}41 - 400063487 * z\hat{}40 - 315345013 * z\hat{}39 - 247263925 * z\hat{}38\\
&- 192833153 * z\hat{}37 - 149545350 * z\hat{}36 - 115306141 * z\hat{}35 - 88375798 * z\hat{}34\\
&- 67316465 * z\hat{}33 - 50946713 * z\hat{}32 - 38300869 * z\hat{}31 - 28594504 * z\hat{}30 - 21193871 * z\hat{}29\\
&- 15590359 * z\hat{}28 - 11378092 * z\hat{}27 - 8235506 * z\hat{}26 - 5909341 * z\hat{}25 - 4201700 * z\hat{}24\\
&- 2958906 * z\hat{}23 - 2062659 * z\hat{}22 - 1422483 * z\hat{}21 - 969859 * z\hat{}20 - 653248 * z\hat{}19\\
&- 434315 * z\hat{}18 - 284748 * z\hat{}17 - 183908 * z\hat{}16 - 116858 * z\hat{}15 - 72955 * z\hat{}14 - 44670 * z\hat{}13\\
&- 26777 * z\hat{}12 - 15674 * z\hat{}11 - 8937 * z\hat{}10 - 4944 * z\hat{}9 - 2644 * z\hat{}8 - 1358 * z\hat{}7 - 666 * z\hat{}6\\
&- 308 * z\hat{}5 - 133 * z\hat{}4 - 52 * z\hat{}3 - 18 * z\hat{}2 - 5 * z - 1)
\end{aligned}
$$

B.2 Semi-magic 4-Squares by Cubical Count

If $\mathcal{C}_4\left(t\right)$ denotes the number of semi-magic 4-squares with positive integer entries and magic sum t, then the rational function representing $\sum_{t\geq 0}\mathcal{C}_4\left(t\right)z^t$ is equal to:

1152*(16972282*z^922 + 55124778*z^921 + 166705790*z^920 + 437085856*z^919
+ 1054348331*z^918 + 2363737841*z^917 + 5064279895*z^916 + 10330051348*z^915
+ 20412253522*z^914 + 39041718921*z^913 + 72747908235*z^912
+ 132295144958*z^911 + 235664106083*z^910 + 411629001532*z^909
+ 706770208247*z^908 + 1194069382853*z^907 + 1988094683408*z^906
+ 3265015714926*z^905 + 5294895029565*z^904 + 8485075997131*z^903
+ 13447856082060*z^902 + 21091158039468*z^901 + 32755180995955*z^900
+ 50397359710190*z^899 + 76861431919270*z^898 + 116241804879895*z^897
+ 174403251965812*z^896 + 259680199626831*z^895 + 383856332150701*z^894
+ 563477923125808*z^893 + 821662320157825*z^892 + 1190509502966889*z^891
+ 1714380261192333*z^890 + 2454236629778009*z^889 + 3493474037355803*z^888
+ 4945596408102206*z^887 + 6964415549935697*z^886 + 9757388478434136*z^885
+ 13603177140733934*z^884 + 18874456413827336*z^883
+ 26067686820578904*z^882 + 35841545745093616*z^881
+ 49066688635370729*z^880 + 66889592215144776*z^879
+ 90814616335047192*z^878 + 122808653495982359*z^877
+ 165434726775022715*z^876 + 222021388827652846*z^875
+ 296877580808384328*z^874 + 395563568729234845*z^873
+ 525232520679570807*z^872 + 695058925065073446*z^871
+ 916775638352767914*z^870 + 1205344023311506094*z^869
+ 1579789446186680112*z^868 + 2064238694887073477*z^867
+ 2689206741510197473*z^866 + 3493186875549728284*z^865
+ 4524613378805835024*z^864 + 5844275842127668305*z^863
+ 7528285138437942816*z^862 + 9671705855809267724*z^861
+ 12392998731526503945*z^860 + 15839438168105133769*z^859
+ 20193709357768341027*z^858 + 25681920474354788287*z^857
+ 32583319121009204491*z^856 + 41242046287871144659*z^855
+ 52081333869699679430*z^854 + 65620613686901290731*z^853
+ 82496104318096817745*z^852 + 103485528065993266624*z^851
+ 129537742416334325840*z^850 + 161808189056212986009*z^849

$+ 201701239803866979419 * z\hat{}848 + 250920680957541578800 * z\hat{}847$
$+ 311529812073380221827 * z\hat{}846 + 386022854815889364476 * z\hat{}845$
$+ 477409677635815279526 * z\hat{}844 + 589316137628837240946 * z\hat{}843$
$+ 726102748873106244280 * z\hat{}842 + 893004781178651125756 * z\hat{}841$
$+ 1096297427629229676823 * z\hat{}840 + 1343490202372775246628 * z\hat{}839$
$+ 1643555426493339151313 * z\hat{}838 + 2007196349108290255981 * z\hat{}837$
$+ 2447161353605930328898 * z\hat{}836 + 2978611601155298140216 * z\hat{}835$
$+ 3619550628962220657758 * z\hat{}834 + 4391325594220884607251 * z\hat{}833$
$+ 5319211351383033921408 * z\hat{}832 + 6433090068456064644067 * z\hat{}831$
$+ 7768241000902122662838 * z\hat{}830 + 9366256993985069140558 * z\hat{}829$
$+ 11276106718686618881223 * z\hat{}828 + 13555364142804428173461 * z\hat{}827$
$+ 16271629821858549469675 * z\hat{}826 + 19504171770844274084702 * z\hat{}825$
$+ 23345817564356033121171 * z\hat{}824 + 27905133332512721049870 * z\hat{}823$
$+ 33308930198662459971910 * z\hat{}822 + 39705143765993941426118 * z\hat{}821$
$+ 47266138356225374444430 * z\hat{}820 + 56192494045543446829968 * z\hat{}819$
$+ 66717342126356713973990 * z\hat{}818 + 79111322531386330157715 * z\hat{}817$
$+ 93688246150991548375261 * z\hat{}816 + 110811554787885795204473 * z\hat{}815$
$+ 130901683080697247375029 * z\hat{}814 + 154444438850860286999844 * z\hat{}813$
$+ 182000532558145563110109 * z\hat{}812 + 214216401457728278471691 * z\hat{}811$
$+ 251836491458050976191612 * z\hat{}810 + 295717177927646634734481 * z\hat{}809$
$+ 346842527900597683040462 * z\hat{}808 + 406342128373659733742807 * z\hat{}807$
$+ 475511231105173237254623 * z\hat{}806 + 555833491319861747879902 * z\hat{}805$
$+ 649006608794045354963456 * z\hat{}804 + 756971212413903617191463 * z\hat{}803$
$+ 881943366685566032892497 * z\hat{}802 + 1026451117926899179509703 * z\hat{}801$
$+ 1193375542675385668834719 * z\hat{}800 + 1385996807881963640566277 * z\hat{}799$
$+ 1608045805946234880841446 * z\hat{}798 + 1863761983778368839341501 * z\hat{}797$
$+ 2157958048669400941340588 * z\hat{}796 + 2496092300465178044519764 * z\hat{}795$
$+ 2884349414859917366243950 * z\hat{}794 + 3329730581596760757232340 * z\hat{}793$
$+ 3840153990226407550029786 * z\hat{}792 + 4424566749179015510441249 * z\hat{}791$
$+ 5093069428339274949696238 * z\hat{}790 + 5857054524647698879508694 * z\hat{}789$
$+ 6729360272508789839873206 * z\hat{}788 + 7724441348667608302801416 * z\hat{}787$
$+ 8858558163792525108371163 * z\hat{}786 + 10149986581982870677153189 * z\hat{}785$
$+ 11619250075051726058197768 * z\hat{}784 + 13289376491318151955461494 * z\hat{}783$
$+ 15186181810314845795912945 * z\hat{}782 + 17338583454648772124922737 * z\hat{}781$
$+ 19778945951211057008092574 * z\hat{}780 + 22543461963998049076256471 * z\hat{}779$
$+ 25672571974543230110621469 * z\hat{}778 + 29211426149795647177900508 * z\hat{}777$
$+ 33210392227527968346952220 * z\hat{}776 + 37725613550722202105412875 * z\hat{}775$

$+ 4281962171310789889701852 * z\hat{}774 + 4856200862088507416525 7251 * z\hat{}773$
$+ 5503016315103995165952 4508 * z\hat{}772 + 6231007797517013544184 0023 * z\hat{}771$
$+ 7049723254210106145041 4633 * z\hat{}770 + 7969755865103439190666 8610 * z\hat{}769$
$+ 9002849552481356740930 7891 * z\hat{}768 + 1016201417856953822042 26994 * z\hat{}767$
$+ 1146165122719086160931 70870 * z\hat{}766 + 1291769081848832815951 28120 * z\hat{}765$
$+ 1454774096555586443637 00689 * z\hat{}764 + 1637125004330012736305 25192 * z\hat{}763$
$+ 1840968350642712729671 04298 * z\hat{}762 + 2068671596168252990762 54506 * z\hat{}761$
$+ 2322843977320362277019 84329 * z\hat{}760 + 2606359145524086457618 42186 * z\hat{}759$
$+ 2922379718780953129489 79425 * z\hat{}758 + 3274383887377664890825 19921 * z\hat{}757$
$+ 3666194224514546919769 78417 * z\hat{}756 + 4102008861712388855685 20205 * z\hat{}755$
$+ 4586435198610549106242 18468 * z\hat{}754 + 5124526326667411671825 95521 * z\hat{}753$
$+ 5721820356883569308017 79029 * z\hat{}752 + 6384382852402495374530 57462 * z\hat{}751$
$+ 7118852578326507787342 05185 * z\hat{}750 + 7932490792667472972188 61840 * z\hat{}749$
$+ 8833234314717235012851 68734 * z\hat{}748 + 9829752619551983698339 46118 * z\hat{}747$
$+ 1093150922062959445431 987303 * z\hat{}746 + 1214882761570297384317 091483 * z\hat{}745$
$+ 1349296208537973826306 594455 * z\hat{}744 + 1497617364772916750129 964693 * z\hat{}743$
$+ 1661181148727416828397 181780 * z\hat{}742 + 1841440019153793046535 603769 * z\hat{}741$
$+ 2039973314402975378226 854741 * z\hat{}740 + 2258497243808266042931 046543 * z\hat{}739$
$+ 2498875569237635650801 964713 * z\hat{}738 + 2763131016511595453757 948733 * z\hat{}737$
$+ 3053457458086895360067 860521 * z\hat{}736 + 3372232910069689392777 750022 * z\hat{}735$
$+ 3722033388374065619712 545243 * z\hat{}734 + 4105647670541073270650 691808 * z\hat{}733$
$+ 4526093011519880196655 013997 * z\hat{}732 + 4986631863430968008269 480243 * z\hat{}731$
$+ 5490789651134209170912 747449 * z\hat{}730 + 6042373657142087797459 092525 * z\hat{}729$
$+ 6645493071216662717893 510391 * z\hat{}728 + 7304580261682196972793 352896 * z\hat{}727$
$+ 8024413327254739334436 646882 * z\hat{}726 + 8810139989832412284500 402047 * z\hat{}725$
$+ 9667302890402622679719 713083 * z\hat{}724$
$+ 1060186635178366888275 0608686 * z\hat{}723$
$+ 1162024467354216667069 0467491 * z\hat{}722$
$+ 1272933202587366016847 2341655 * z\hat{}721$
$+ 1393653401073392720107 1920856 * z\hat{}720$
$+ 1524980095980119188359 6253561 * z\hat{}719$
$+ 1667766304018673803779 5636171 * z\hat{}718$
$+ 1822926723990922455942 2453155 * z\hat{}717$
$+ 1991441630627960790489 4239689 * z\hat{}716$
$+ 2174360971120110533142 9168796 * z\hat{}715$
$+ 2372808671826819011362 5369280 * z\hat{}714$
$+ 2587987162711685395482 7136593 * z\hat{}713$

$+ 28211821271056066642603363614 * z\hat{}712$

$+ 30737674844237018121297061553 * z\hat{}711$

$+ 33472106134839520065031095363 * z\hat{}710$

$+ 36430778240584636918910437432 * z\hat{}709$

$+ 39630400842694705577160461657 * z\hat{}708$

$+ 43088790113796543896676052486 * z\hat{}707$

$+ 46824931334606843507537512199 * z\hat{}706$

$+ 50859044293098141522773802566 * z\hat{}705$

$+ 55212651538657982927730833936 * z\hat{}704$

$+ 59908649562036268026259456754 * z\hat{}703$

$+ 64971382970099821794518732471 * z\hat{}702$

$+ 70426721722055878188491079323 * z\hat{}701$

$+ 76302141491378405305599481074 * z\hat{}700$

$+ 82626807214611431481808966949 * z\hat{}699$

$+ 89431659885082688348195892858 * z\hat{}698$

$+ 96749506645736849247254269084 * z\hat{}697$

$+ 104615114231383942862150094562 * z\hat{}696$

$+ 113065305806010166889964999092 * z\hat{}695$

$+ 122139061236054472432571076142 * z\hat{}694$

$+ 131877620835024661263944893186 * z\hat{}693$

$+ 142324592609199008098386201885 * z\hat{}692$

$+ 153526063027695780429496677161 * z\hat{}691$

$+ 165530711333632719310631914471 * z\hat{}690$

$+ 178389927405653604961514649101 * z\hat{}689$

$+ 192157933171563710616752701079 * z\hat{}688$

$+ 206891907567349432098300428194 * z\hat{}687$

$+ 222652115026312218744238648213 * z\hat{}686$

$+ 239502037473526922927745091330 * z\hat{}685$

$+ 257508509791254116400695919509 * z\hat{}684$

$+ 276741858710340260907347791843 * z\hat{}683$

$+ 297276045072008748174099514951 * z\hat{}682$

$+ 319188809392762825366329475148 * z\hat{}681$

$+ 342561820653437437609264087730 * z\hat{}680$

$+ 367480828220670885741869552090 * z\hat{}679$

$+ 394035816796341103591527837252 * z\hat{}678$

$+ 422321164276677569489900663150 * z\hat{}677$

$+ 452435802389017849670657918019 * z\hat{}676$

$+ 484483379959308750995988366091 * z\hat{}675$
$+ 518572428648736780014956404738 * z\hat{}674$
$+ 554816530982017990594449318484 * z\hat{}673$
$+ 593334490474249449051193693493 * z\hat{}672$
$+ 634250503646452311316533655718 * z\hat{}671$
$+ 677694333703475886080785230576 * z\hat{}670$
$+ 723801485630329886824077020555 * z\hat{}669$
$+ 772713382445818501734119553931 * z\hat{}668$
$+ 824577542334026027480247772033 * z\hat{}667$
$+ 879547756356398212916459962295 * z\hat{}666$
$+ 937784266428247211508743915032 * z\hat{}665$
$+ 999453943225232962929734684246 * z\hat{}664$
$+ 1064730463666016450897822047758 * z\hat{}663$
$+ 1133794487598700086526659625556 * z\hat{}662$
$+ 1206833833299045200879647368226 * z\hat{}661$
$+ 1284043651369757838285970505278 * z\hat{}660$
$+ 1365626596610435844979559423401 * z\hat{}659$
$+ 1451792997409161182570074008439 * z\hat{}658$
$+ 1542761022187166829507624553337 * z\hat{}657$
$+ 1638756842409717616037643784229 * z\hat{}656$
$+ 1740014791657170661973294065639 * z\hat{}655$
$+ 1846777520232459766433235985326 * z\hat{}654$
$+ 1959296144762711888639445799879 * z\hat{}653$
$+ 2077830392235819724623943426326 * z\hat{}652$
$+ 2202648737895170278519005218633 * z\hat{}651$
$+ 2334028536399967693865822463358 * z\hat{}650$
$+ 2472256145642167194980759269060 * z\hat{}649$
$+ 2617627042596698123926634738900 * z\hat{}648$
$+ 2770445930566739208949396213668 * z\hat{}647$
$+ 2931026837173205568456180788082 * z\hat{}646$
$+ 3099693202424513332834175289482 * z\hat{}645$
$+ 3276777956192140709441212496287 * z\hat{}644$
$+ 3462623584406553983244432567353 * z\hat{}643$
$+ 3657582183279903562808364649336 * z\hat{}642$
$+ 3862015500853413646546457103041 * z\hat{}641$
$+ 4076294965161938236806196198911 * z\hat{}640$
$+ 4300801698302468755297403519797 * z\hat{}639$

$+ 4535926515690966281061904970664 * z\hat{}638$
$+ 4782069909789318494716883696202 * z\hat{}637$
$+ 5039642017585166885143316622532 * z\hat{}636$
$+ 5309062571108202010403608497936 * z\hat{}635$
$+ 5590760830271134874193415538014 * z\hat{}634$
$+ 5885175497328126062802871702178 * z\hat{}633$
$+ 6192754612252018984478196530435 * z\hat{}632$
$+ 6513955428340294847186249388168 * z\hat{}631$
$+ 6849244267372453381164921190579 * z\hat{}630$
$+ 7199096353654343128264126122780 * z\hat{}629$
$+ 7563995626302212464758496124826 * z\hat{}628$
$+ 7944434529136541059125306482194 * z\hat{}627$
$+ 8340913777577625275922401783424 * z\hat{}626$
$+ 8753942101956832277852696573188 * z\hat{}625$
$+ 9184035966684184496565580282404 * z\hat{}624$
$+ 9631719264739672066798874851207 * z\hat{}623$
$+ 10097522986987393186989920148841 * z\hat{}622$
$+ 10581984865843248133612998965714 * z\hat{}621$
$+ 11085648992863642969852432853898 * z\hat{}620$
$+ 11609065409859217833502196700433 * z\hat{}619$
$+ 12152789673179391208905097128417 * z\hat{}618$
$+ 12717382390854989767238209845553 * z\hat{}617$
$+ 13303408732333015582011674787434 * z\hat{}616$
$+ 13911437910583925577774255111990 * z\hat{}615$
$+ 14542042636413482339016632569318 * z\hat{}614$
$+ 15195798544862264405844923192291 * z\hat{}613$
$+ 15873283593632340815884268787280 * z\hat{}612$
$+ 16575077433536142741380037098998 * z\hat{}611$
$+ 17301760751023475610210148291822 * z\hat{}610$
$+ 18053914582902365689484176624275 * z\hat{}609$
$+ 18832119603434529954047381116432 * z\hat{}608$
$+ 19636955384049883291689961834755 * z\hat{}607$
$+ 20468999625993394957782053598559 * z\hat{}606$
$+ 21328827366284685856570242275697 * z\hat{}605$
$+ 22217010157443005967203383086932 * z\hat{}604$
$+ 23134115221500273685636155325533 * z\hat{}603$
$+ 24080704578899931856535918780481 * z\hat{}602$

$$
\begin{gathered}
+ 25057334152951818784508866637989 \ast z\hat{\ }601 \\
+ 26064552850590557141366820682524 \ast z\hat{\ }600 \\
+ 27102901620259178404070962495049 \ast z\hat{\ }599 \\
+ 28172912487818577034097495102804 \ast z\hat{\ }598 \\
+ 29275107571458680655766390373716 \ast z\hat{\ }597 \\
+ 30409998076666948886288572018960 \ast z\hat{\ }596 \\
+ 31578083272385429631616945642830 \ast z\hat{\ }595 \\
+ 32779849449567399093391245757118 \ast z\hat{\ }594 \\
+ 34015768863419744704671108597927 \ast z\hat{\ }593 \\
+ 35286298660696289597798780712351 \ast z\hat{\ }592 \\
+ 36591879793481053077352493406660 \ast z\hat{\ }591 \\
+ 37932935920977850397110875140940 \ast z\hat{\ }590 \\
+ 39309872300894201370802603913292 \ast z\hat{\ }589 \\
+ 40723074672082388729640605211717 \ast z\hat{\ }588 \\
+ 42172908130168920969061582396018 \ast z\hat{\ }587 \\
+ 43659715997975056601375058463640 \ast z\hat{\ }586 \\
+ 45183818692595365684189400726014 \ast z\hat{\ }585 \\
+ 46745512591068290558515996089568 \ast z\hat{\ }584 \\
+ 48345068896631940433436159924049 \ast z\hat{\ }583 \\
+ 49982732507619974115662877391925 \ast z\hat{\ }582 \\
+ 51658720891105702244934037282755 \ast z\hat{\ }581 \\
+ 53373222963457851851987550680229 \ast z\hat{\ }580 \\
+ 55126397980017782791899048987254 \ast z\hat{\ }579 \\
+ 56918374436156027258783066947793 \ast z\hat{\ }578 \\
+ 58749248982004532666657084184504 \ast z\hat{\ }577 \\
+ 60619085353200952000930840510424 \ast z\hat{\ }576 \\
+ 62527913320011129416234427575278 \ast z\hat{\ }575 \\
+ 64475727657226933431069775349338 \ast z\hat{\ }574 \\
+ 66462487137256873798644233753108 \ast z\hat{\ }573 \\
+ 68488113548848201007714794172781 \ast z\hat{\ }572 \\
+ 70552490743889202906206709913341 \ast z\hat{\ }571 \\
+ 72655463714751224102632232767748 \ast z\hat{\ }570 \\
+ 74796837704629011225552533877375 \ast z\hat{\ }569 \\
+ 76976377353337721654373275771360 \ast z\hat{\ }568 \\
+ 79193805881012493963964242597492 \ast z\hat{\ }567 \\
+ 81448804312144607692816146376852 \ast z\hat{\ }566 \\
+ 83741010742363828230835154367786 \ast z\hat{\ }565
\end{gathered}
$$

$+ 86070019650352636559621436256648 * z\hat{}564$
$+ 88435381257241256811264570718645 * z\hat{}563$
$+ 90836600935796171607714946345146 * z\hat{}562$
$+ 93273138671665434477840739621632 * z\hat{}561$
$+ 95744408578895364420981412548620 * z\hat{}560$
$+ 98249778471871126497243852473362 * z\hat{}559$
$+ 100788569495772438351896218963243 * z\hat{}558$
$+ 103360055817560893003059480400825 * z\hat{}557$
$+ 105963464379441688369823704711567 * z\hat{}556$
$+ 108597974716655313215743530029373 * z\hat{}555$
$+ 111262718841368853738672761628537 * z\hat{}554$
$+ 113956781194337179355907862359784 * z\hat{}553$
$+ 116679198665906595580816001919399 * z\hat{}552$
$+ 119428960687822458184855647168843 * z\hat{}551$
$+ 122205009397193338840403672517172 * z\hat{}550$
$+ 125006239873842178912456209446619 * z\hat{}549$
$+ 127831500452155432469002161188669 * z\hat{}548$
$+ 130679593108408764184768690860753 * z\hat{}547$
$+ 133549273924418750031641336358583 * z\hat{}546$
$+ 136439253628228234432507600304684 * z\hat{}545$
$+ 139348198212395190032943158890998 * z\hat{}544$
$+ 142274729630304836023685923081016 * z\hat{}543$
$+ 145217426570779401354949657623134 * z\hat{}542$
$+ 148174825311102727814401024618016 * z\hat{}541$
$+ 151145420648425230775472571304101 * z\hat{}540$
$+ 154127666909351758947954636225953 * z\hat{}539$
$+ 157119979037358280324752569801250 * z\hat{}538$
$+ 160120733757515985802469742458234 * z\hat{}537$
$+ 163128270817841323799568430476318 * z\hat{}536$
$+ 166140894306420306416017830026975 * z\hat{}535$
$+ 169156874043293446737692533344795 * z\hat{}534$
$+ 172174447045916279166310359919300 * z\hat{}533$
$+ 175191819066848164444369981780661 * z\hat{}532$
$+ 178207166202151103978752149207163 * z\hat{}531$
$+ 181218636568819455771354689255623 * z\hat{}530$
$+ 184224352049392608871176454077188 * z\hat{}529$
$+ 187222410101744981335773203304487 * z\hat{}528$

$+ 190210885631882749004388082430190 * z\hat{}527$
$+ 193187832927423898906089771902968 * z\hat{}526$
$+ 196151287649278862374727214210955 * z\hat{}525$
$+ 199099268878902779996863287885686 * z\hat{}524$
$+ 202029781218338456465394653313880 * z\hat{}523$
$+ 204940816940131176429690954416934 * z\hat{}522$
$+ 207830358184053591137428896425451 * z\hat{}521$
$+ 210696379197450972392534683170776 * z\hat{}520$
$+ 213536848615884831315121954436229 * z\hat{}519$
$+ 216349731780636547691001962103197 * z\hat{}518$
$+ 219132993089512569103141482135204 * z\hat{}517$
$+ 221884598377289446516814278659638 * z\hat{}516$
$+ 224602517322030548218620691108807 * z\hat{}515$
$+ 227284725873417448226331930642278 * z\hat{}514$
$+ 229929208699147551691842656159286 * z\hat{}513$
$+ 232533961645376378192227407011076 * z\hat{}512$
$+ 235096994207107672546691789459970 * z\hat{}511$
$+ 237616332004378185202323544224467 * z\hat{}510$
$+ 240090019260025930215963843336614 * z\hat{}509$
$+ 242516121274792161736620279595327 * z\hat{}508$
$+ 244892726895467356794823064447562 * z\hat{}507$
$+ 247217950971771453139626299253326 * z\hat{}506$
$+ 249489936797637398067674271381188 * z\hat{}505$
$+ 251706858532566129936058785804586 * z\hat{}504$
$+ 253866923598719164394532013864692 * z\hat{}503$
$+ 255968375049433504140670100559031 * z\hat{}502$
$+ 258009493904861162248724123094606 * z\hat{}501$
$+ 259988601450473783931856100328398 * z\hat{}500$
$+ 261904061494210010449194457561814 * z\hat{}499$
$+ 263754282578101024846587903970555 * z\hat{}498$
$+ 265537720140266316597003358466608 * z\hat{}497$
$+ 267252878623248836633066317933769 * z\hat{}496$
$+ 268898313524734384654550061155461 * z\hat{}495$
$+ 270472633386796064324335784003811 * z\hat{}494$
$+ 271974501719898805638035275705433 * z\hat{}493$
$+ 273402638858013114746748011638222 * z\hat{}492$
$+ 274755823741299078127928939566884 * z\hat{}491$

$+\ 276032895622953077739802810699192 \ast z\hat{}490$

$+\ 277232755696938250075652419590183 \ast z\hat{}489$

$+\ 278354368643467323281009910129327 \ast z\hat{}488$

$+\ 279396764089250566713980527190628 \ast z\hat{}487$

$+\ 280359037979684075773695511497946 \ast z\hat{}486$

$+\ 281240353860311872458377505665983 \ast z\hat{}485$

$+\ 282039944065071220706227815983780 \ast z\hat{}484$

$+\ 282757110809001438650647207694787 \ast z\hat{}483$

$+\ 283391227183284224115829250539944 \ast z\hat{}482$

$+\ 283941738050665338512762042460717 \ast z\hat{}481$

$+\ 284408160839505290650082154130571 \ast z\hat{}480$

$+\ 284790086234897627525699073936106 \ast z\hat{}479$

$+\ 285087178765499426033225332256210 \ast z\hat{}478$

$+\ 285299177284916764695552144466685 \ast z\hat{}477$

$+\ 285425895346700162982510986137825 \ast z\hat{}476$

$+\ 285467221472208378449200088480665 \ast z\hat{}475$

$+\ 285423119310815339117661793112524 \ast z\hat{}474$

$+\ 285293627692141544330647990730576 \ast z\hat{}473$

$+\ 285078860570209741176079852518973 \ast z\hat{}472$

$+\ 284779006859632290652549607849869 \ast z\hat{}471$

$+\ 284394330164156080974772891885528 \ast z\hat{}470$

$+\ 283925168398097389976220883411479 \ast z\hat{}469$

$+\ 283371933301415393959932491090529 \ast z\hat{}468$

$+\ 282735109849376401040226301233802 \ast z\hat{}467$

$+\ 282015255557972943222428057680285 \ast z\hat{}466$

$+\ 281212999686460014599673972071254 \ast z\hat{}465$

$+\ 280329042338576518520532278403312 \ast z\hat{}464$

$+\ 279364153464210911172521465815765 \ast z\hat{}463$

$+\ 278319171763467556363097673120619 \ast z\hat{}462$

$+\ 277195003495272086881958565773472 \ast z\hat{}461$

$+\ 275992621192841477411363917498630 \ast z\hat{}460$

$+\ 274713062288515363292791009405028 \ast z\hat{}459$

$+\ 273357427650620680425798322422887 \ast z\hat{}458$

$+\ 271926880035199902103706012564148 \ast z\hat{}457$

$+\ 270422642455595209141454763777295 \ast z\hat{}456$

$+\ 268845996473024971961576060290506 \ast z\hat{}455$

$+\ 267198280411436049507338749187351 \ast z\hat{}454$

$+ 265480887500043930031191068095315 * z\hat{}453$
$+ 263695263947103649943749553515113 * z\hat{}452$
$+ 261842906948566242060664988363825 * z\hat{}451$
$+ 259925362635389067050473277459013 * z\hat{}450$
$+ 257944223963362501907729795475950 * z\hat{}449$
$+ 255901128549410858410060271487625 * z\hat{}448$
$+ 253797756458401042155229934207686 * z\hat{}447$
$+ 251635827944568970744242835770879 * z\hat{}446$
$+ 249417101151730348349718258579534 * z\hat{}445$
$+ 247143369776499535688714037287445 * z\hat{}444$
$+ 244816460698777411989301286554163 * z\hat{}443$
$+ 242438231583806642574452365619763 * z\hat{}442$
$+ 240010568460110356224421506504362 * z\hat{}441$
$+ 237535383277648086281829363251074 * z\hat{}440$
$+ 235014611450520922716802387316129 * z\hat{}439$
$+ 232450209388556176117787994667490 * z\hat{}438$
$+ 229844152022080731452326708582362 * z\hat{}437$
$+ 227198430324171507372155987412661 * z\hat{}436$
$+ 224515048834631582977790930828805 * z\hat{}435$
$+ 221796023189901256950200974175948 * z\hat{}434$
$+ 219043377663055401281165437229494 * z\hat{}433$
$+ 216259142717981411247377984803483 * z\hat{}432$
$+ 213445352581756967987382934498712 * z\hat{}431$
$+ 210604042839172955803744502281038 * z\hat{}430$
$+ 207737248053255645402975758404175 * z\hat{}429$
$+ 204846999415552679761224617904627 * z\hat{}428$
$+ 201935322429841215623459361080333 * z\hat{}427$
$+ 199004234632812559289723520887543 * z\hat{}426$
$+ 196055743355167829856144499647050 * z\hat{}425$
$+ 193091843526442119094136422367352 * z\hat{}424$
$+ 190114515526742640341417226249299 * z\hat{}423$
$+ 187125723088457745337567481299320 * z\hat{}422$
$+ 184127411250851090614866849902896 * z\hat{}421$
$+ 181121504370316662094953193704617 * z\hat{}420$
$+ 178109904188918778717487073103968 * z\hat{}419$
$+ 175094487963694322794943922768048 * z\hat{}418$
$+ 172077106659035554506608165451282 * z\hat{}417$

$+ 169059583204318401579780551821780 * z\hat{}416$

$+ 166043710818776647704884525664084 * z\hat{}415$

$+ 163031251405464116731838885218073 * z\hat{}414$

$+ 160023934015978645566573072117492 * z\hat{}413$

$+ 157023453387460190646917163821406 * z\hat{}412$

$+ 154031468553205007034854893908968 * z\hat{}411$

$+ 151049601528074992477339324805360 * z\hat{}410$

$+ 148079436069710490619607440576932 * z\hat{}409$

$+ 145122516516392289406548357042487 * z\hat{}408$

$+ 142180346702229024133285999360633 * z\hat{}407$

$+ 139254388950185558024893174250742 * z\hat{}406$

$+ 136346063143301258272150992872795 * z\hat{}405$

$+ 133456745874289931169425139630276 * z\hat{}404$

$+ 130587769673550916299852573983357 * z\hat{}403$

$+ 127740422315468639827839988567207 * z\hat{}402$

$+ 124915946202721477859781865700898 * z\hat{}401$

$+ 122115537828174904661641756460087 * z\hat{}400$

$+ 119340347313784579248204867376213 * z\hat{}399$

$+ 116591478025796724686573145010147 * z\hat{}398$

$+ 113869986265392152672106727600825 * z\hat{}397$

$+ 111176881033790689971227247280500 * z\hat{}396$

$+ 108513123870701140793205917235486 * z\hat{}395$

$+ 105879628764882017032636068254391 * z\hat{}394$

$+ 103277262135456944858522210496917 * z\hat{}393$

$+ 100706842882519295635378451772061 * z\hat{}392$

$+ 98169142505450339208840086624775 * z\hat{}391$

$+ 95664885287277098708636515822901 * z\hat{}390$

$+ 93194748543297534515551245834064 * z\hat{}389$

$+ 90759362932114357613855325602823 * z\hat{}388$

$+ 88359312827132393042174945811139 * z\hat{}387$

$+ 85995136746500276822849592866244 * z\hat{}386$

$+ 83667327839403385089971589924796 * z\hat{}385$

$+ 81376334426553195422702478473375 * z\hat{}384$

$+ 79122560592657064603329984148940 * z\hat{}383$

$+ 76906366828603248855734113556874 * z\hat{}382$

$+ 74728070721047474768724670779929 * z\hat{}381$

$+ 72587947687050711400832959282826 * z\hat{}380$

$$
\begin{aligned}
&+ 70486231751381864349221684471554 * z\hat{}379\\
&+ 68423116364074779540118297081493 * z\hat{}378\\
&+ 66398755255805361235298856569215 * z\hat{}377\\
&+ 64413263328642297298387338663059 * z\hat{}376\\
&+ 62466717579713282868152595216255 * z\hat{}375\\
&+ 60559158055327922266460528613381 * z\hat{}374\\
&+ 58690588833098390172842301958837 * z\hat{}373\\
&+ 56860979029609292355742134714720 * z\hat{}372\\
&+ 55070263831198989737807584628846 * z\hat{}371\\
&+ 53318345545435453403729278846753 * z\hat{}370\\
&+ 51605094670890792101940965668952 * z\hat{}369\\
&+ 49930350982849131546727138459376 * z\hat{}368\\
&+ 48293924632613099061368999931100 * z\hat{}367\\
&+ 46695597258113655893856628979901 * z\hat{}366\\
&+ 45135123103567278110543435343622 * z\hat{}365\\
&+ 43612230145972084898536895295343 * z\hat{}364\\
&+ 42126621226281587119385245313269 * z\hat{}363\\
&+ 40677975183149547941906656557269 * z\hat{}362\\
&+ 39265947987193428337076609112686 * z\hat{}361\\
&+ 37890173873785008842426485854897 * z\hat{}360\\
&+ 36550266472436731537403993123351 * z\hat{}359\\
&+ 35245819930918782734587764333013 * z\hat{}358\\
&+ 33976410032306911203148701862366 * z\hat{}357\\
&+ 32741595303231861469955980353925 * z\hat{}356\\
&+ 31540918111670346011579770185395 * z\hat{}355\\
&+ 30373905752691826868024579323444 * z\hat{}354\\
&+ 29240071520647548953915709127594 * z\hat{}353\\
&+ 28138915766365144899551208899336 * z\hat{}352\\
&+ 27069926937986507953589736110598 * z\hat{}351\\
&+ 26032582604165131969261405192302 * z\hat{}350\\
&+ 25026350458414806774026377358829 * z\hat{}349\\
&+ 24050689303480813283632405152759 * z\hat{}348\\
&+ 23105050014680911206268616129320 * z\hat{}347\\
&+ 22188876481242593725331065111553 * z\hat{}346\\
&+ 21301606524738884683476529485910 * z\hat{}345\\
&+ 20442672793803283319141705465302 * z\hat{}344\\
&+ 19611503634379162788794187099352 * z\hat{}343
\end{aligned}
$$

$+ 18807523934835685508466317475020 * z\hat{}342$
$+ 18030155945355206625586703954332 * z\hat{}341$
$+ 17278820071071662532222360758648 * z\hat{}340$
$+ 16552935638509927922622565289183 * z\hat{}339$
$+ 15851921634947827084076874120935 * z\hat{}338$
$+ 15175197420389989178679508604546 * z\hat{}337$
$+ 14522183411911092935267479936045 * z\hat{}336$
$+ 13892301740190077193851423065997 * z\hat{}335$
$+ 13284976878121450987385558498363 * z\hat{}334$
$+ 12699636241449975851477569107008 * z\hat{}333$
$+ 12135710761435371984941949986674 * z\hat{}332$
$+ 11592635429609600005814540301960 * z\hat{}331$
$+ 11069849814745162502148157415336 * z\hat{}330$
$+ 10566798552204255386063991380966 * z\hat{}329$
$+ 10082931805889767711009103122741 * z\hat{}328$
$+ 9617705703065794479917185490853 * z\hat{}327$
$+ 9170582742361603361308345509223 * z\hat{}326$
$+ 8741032175314804186542508691470 * z\hat{}325$
$+ 8328530361850751806037822668309 * z\hat{}324$
$+ 7932561100132089527407399666600 * z\hat{}323$
$+ 7552615931248587776437603912398 * z\hat{}322$
$+ 7188194419249365313809131596399 * z\hat{}321$
$+ 6838804407050812684506612183463 * z\hat{}320$
$+ 6503962248780570360342223279086 * z\hat{}319$
$+ 6183193019144204742160962054243 * z\hat{}318$
$+ 5876030700423454094011554794030 * z\hat{}317$
$+ 5582018347736388492469162804004 * z\hat{}316$
$+ 5300708233207395776576183283082 * z\hat{}315$
$+ 5031661969711726063355455452153 * z\hat{}314$
$+ 4774450614872418872144332620278 * z\hat{}313$
$+ 4528654755999812147194945887935 * z\hat{}312$
$+ 4293864576672668355740439248406 * z\hat{}311$
$+ 4069679905668122325779485737327 * z\hat{}310$
$+ 3855710248952487252092838702413 * z\hat{}309$
$+ 3651574805449186551762666393736 * z\hat{}308$
$+ 3456902467301183614552377664165 * z\hat{}307$
$+ 3271331805345864415249778394518 * z\hat{}306$

$+ 309451104051800148492713622245 * z\hat{}305$
$+ 292609800189366938180412618560 2 * z\hat{}304$
$+ 276576007208253555407114818822 3 * z\hat{}303$
$+ 261317412067016990582974259484 1 * z\hat{}302$
$+ 246802642640377097145441419106 7 * z\hat{}301$
$+ 233001258880622985320657892962 2 * z\hat{}300$
$+ 219883742989273394942646424726 2 * z\hat{}299$
$+ 207421488665326813545724181719 5 * z\hat{}298$
$+ 195586789495150393615107385834 1 * z\hat{}297$
$+ 184352826547768205214741162456 0 * z\hat{}296$
$+ 173693655237840051563679731992 8 * z\hat{}295$
$+ 163584191517161584349376222779 7 * z\hat{}294$
$+ 154000197453896017223801197163 3 * z\hat{}293$
$+ 144918266257146467750523219624 4 * z\hat{}292$
$+ 136315806802736784196926040667 3 * z\hat{}291$
$+ 128171027714356646224089792842 0 * z\hat{}290$
$+ 120462921052393863213932407400 5 * z\hat{}289$
$+ 113171245660982416368064059373 0 * z\hat{}288$
$+ 106276510221897687223009778254 0 * z\hat{}287$
$+ 9975995606208041282721470520 00 * z\hat{}286$
$+ 9360353975963522133304165226 22 * z\hat{}285$
$+ 8778991559126907969999491272 77 * z\hat{}284$
$+ 8230241786218442133145644618 10 * z\hat{}283$
$+ 7712504315755345999173182754 88 * z\hat{}282$
$+ 7224243255275639574728070535 20 * z\hat{}281$
$+ 6763985381769098111802545306 49 * z\hat{}280$
$+ 6330318364854336853560145522 74 * z\hat{}279$
$+ 5921888995856653478453214029 96 * z\hat{}278$
$+ 5537401425753858717254710983 46 * z\hat{}277$
$+ 5175615414777762441824690287 64 * z\hat{}276$
$+ 4835344596277449889852772392 81 * z\hat{}275$
$+ 4515454757277137533071683432 28 * z\hat{}274$
$+ 4214862137987024933097099349 68 * z\hat{}273$
$+ 3932531752359634775618314708 78 * z\hat{}272$
$+ 3667475731618000134421718298 87 * z\hat{}271$
$+ 3418751692524547820408615442 85 * z\hat{}270$
$+ 3185461132002509236597440591 71 * z\hat{}269$

$$\begin{array}{l}
+ 29667478495733856802376710639 * z\hat{}268 \\
+ 27617963989267730357582611819 9 * z\hat{}267 \\
+ 25698305698003532005368641284 4 * z\hat{}266 \\
+ 23901119012109283394758117538 5 * z\hat{}265 \\
+ 22219382269491053606076788799 5 * z\hat{}264 \\
+ 20646422541239212402239790207 1 * z\hat{}263 \\
+ 19175901754259163958339852987 8 * z\hat{}262 \\
+ 17801803156616264022222543076 4 * z\hat{}261 \\
+ 16518418130052692744270821255 8 * z\hat{}260 \\
+ 15320333353086702605238292040 6 * z\hat{}259 \\
+ 14202418317138314976333848423 3 * z\hat{}258 \\
+ 13159813197185135464712223918 7 * z\hat{}257 \\
+ 12187917077589662117043732101 9 * z\hat{}256 \\
+ 11282376532903803105808941110 5 * z\hat{}255 \\
+ 10439074562695477135392960136 6 * z\hat{}254 \\
+ 9654119878708071473687786182 9 * z\hat{}253 \\
+ 8923836542000629351106679429 7 * z\hat{}252 \\
+ 8244753947080080080658628529 2 * z\hat{}251 \\
+ 7613597149467583170608008053 3 * z\hat{}250 \\
+ 7027277532597328612132839603 7 * z\hat{}249 \\
+ 6482883809465652353764019281 4 * z\hat{}248 \\
+ 5977673353992163727713202084 2 * z\hat{}247 \\
+ 5509063856657463629223812190 3 * z\hat{}246 \\
+ 5074625298607755160764079148 5 * z\hat{}245 \\
+ 4672072238097138219230805301 1 * z\hat{}244 \\
+ 4299256402840005614606928996 9 * z\hat{}243 \\
+ 3954159581598053808649895499 1 * z\hat{}242 \\
+ 3634886808097781794167737353 5 * z\hat{}241 \\
+ 3339659830191911055215061471 4 * z\hat{}240 \\
+ 3066810857013002446167099946 7 * z\hat{}239 \\
+ 2814776576744385379542531692 0 * z\hat{}238 \\
+ 2582092437525594244259310328 1 * z\hat{}237 \\
+ 2367387183939488917124233278 4 * z\hat{}236 \\
+ 2169377641471394004109458808 7 * z\hat{}235 \\
+ 1986863741307697264108054732 8 * z\hat{}234 \\
+ 1818723777829584342087774013 5 * z\hat{}233 \\
+ 1663909891175982660025552067 7 * z\hat{}232
\end{array}$$

$+ 15214437672773198814006392974 * z\hat{}231$

$+ 13904125478157944851578471841 * z\hat{}230$

$+ 12699649426291673056817694123 * z\hat{}229$

$+ 11593075371595607781426060325 * z\hat{}228$

$+ 10577012876386322619141386423 * z\hat{}227$

$+ 9644581968103588017457931768 * z\hat{}226 + 8789381631062432846949636405 * z\hat{}225$

$+ 8005459963183686690697685382 * z\hat{}224 + 7287285929484740482600999046 * z\hat{}223$

$+ 6629722645583028021214537767 * z\hat{}222 + 6028002125942481004902085510 * z\hat{}221$

$+ 5477701433189430842324942597 * z\hat{}220 + 4974720166414765631193665752 * z\hat{}219$

$+ 4515259228065450255752584366 * z\hat{}218 + 4095800810697518742665553672 * z\hat{}217$

$+ 3713089546607378431824976285 * z\hat{}216 + 3364114765075748885988112510 * z\hat{}215$

$+ 3046093803733725232625898193 * z\hat{}214 + 2756456322299861959670029372 * z\hat{}213$

$+ 2492829568718327647560204019 * z\hat{}212 + 2253024549466436685000991757 * z\hat{}211$

$+ 2035023057565938725568772572 * z\hat{}210 + 1836965513549802270663929159 * z\hat{}209$

$+ 1657139576369353315235873378 * z\hat{}208 + 1493969482906455700469020772 * z\hat{}207$

$+ 1346006076440724796532495923 * z\hat{}206 + 1211917486050334681763508205 * z\hat{}205$

$+ 1090480420550366680583924138 * z\hat{}204 + 980572042138914454756078992 * z\hat{}203$

$+ 881162386477072297108401049 * z\hat{}202 + 791307297424295719969020232 * z\hat{}201$

$+ 710141846129809707235614376 * z\hat{}200 + 636874205598794060806426479 * z\hat{}199$

$+ 570779953249353943005902216 * z\hat{}198 + 511196775312596208817285228 * z\hat{}197$

$+ 457519548240151393543558033 * z\hat{}196 + 409195773535351652301420574 * z\hat{}195$

$+ 365721343649292779355203993 * z\hat{}194 + 326636617749242979771148005 * z\hat{}193$

$+ 291522787304522095638422182 * z\hat{}192 + 259998512515678881250577079 * z\hat{}191$

$+ 231716811663920715386617475 * z\hat{}190 + 206362186454122229905455376 * z\hat{}189$

$+ 183647967391144197934190907 * z\hat{}188 + 163313864143337389466646373 * z\hat{}187$

$+ 145123706731408175323977891 * z\hat{}186 + 128863364215576981628742584 * z\hat{}185$

$+ 114338828359419816156552532 * z\hat{}184 + 101374450507564411118052619 * z\hat{}183$

$+ 89811320644785124073264652 * z\hat{}182 + 79505778290698089119501091 * z\hat{}181$

$+ 70328045543661580317524562 * z\hat{}180 + 62160973206180121518955294 * z\hat{}179$

$+ 54898891516904215788821473 * z\hat{}178 + 48446557569405879206738664 * z\hat{}177$

$+ 42718192028595067640926804 * z\hat{}176 + 37636598251584349240815600 * z\hat{}175$

$+ 33132357392939960071476120 * z\hat{}174 + 29143093515534911413436508 * z\hat{}173$

$+ 25612803148327094574381567 * z\hat{}172 + 22491244123431234705978647 * z\hat{}171$

$+ 19733378896385146540413492 * z\hat{}170 + 17298867898651803354268974 * z\hat{}169$

$+ 15151608798668021940935368 * z\hat{}168 + 13259317851164423871609312 * z\hat{}167$

$+ 11593149801625981923685716 * z\hat{}166 + 10127353078441239225302208 * z\hat{}165$

$+ 8838957256242305254596791 * z\hat{}164 + 7707490005660364501588690 * z\hat{}163$

$+ 6714720963192496775949561 * z\hat{}162 + 5844430156191520306124094 * z\hat{}161$
$+ 5082198807444353193196428 * z\hat{}160 + 4415220518021357617059183 * z\hat{}159$
$+ 3832130990733096079490648 * z\hat{}158 + 3322854606708410596357606 * z\hat{}157$
$+ 2878466308427975123401294 * z\hat{}156 + 2491067371497656195402930 * z\hat{}155$
$+ 2153673768148158521635254 * z\hat{}154 + 1860115935738424230816657 * z\hat{}153$
$+ 1604948866596477123286977 * z\hat{}152 + 1383371529493213640272124 * z\hat{}151$
$+ 1191154720691944298321717 * z\hat{}150 + 1024576522251801308946128 * z\hat{}149$
$+ 880364619513327400268069 * z\hat{}148 + 755644797090022445257367 * z\hat{}147$
$+ 647894995342932732021970 * z\hat{}146 + 554904366053775546091078 * z\hat{}145$
$+ 474736818670601480986810 * z\hat{}144 + 405698596075064293737259 * z\hat{}143$
$+ 346309462901723914155591 * z\hat{}142 + 295277129170420449850069 * z\hat{}141$
$+ 251474568745575478053875 * z\hat{}140 + 213919925180400006597177 * z\hat{}139$
$+ 181758728022422898430079 * z\hat{}138 + 154248170026470849053567 * z\hat{}137$
$+ 130743220958786212292021 * z\hat{}136 + 110684376251438215664158 * z\hat{}135$
$+ 93586859550148092916170 * z\hat{}134 + 79031116741074290817858 * z\hat{}133$
$+ 66654456086758095849942 * z\hat{}132 + 56143704266280915103737 * z\hat{}131$
$+ 47228762033354856950568 * z\hat{}130 + 39676955550369563864 08 * z\hat{}129$
$+ 33288090810842725555177 * z\hat{}128 + 27890128443775989552864 * z\hat{}127$
$+ 23335405616528364896779 * z\hat{}126 + 19497339506777256918039 * z\hat{}125$
$+ 16267554459501671110163 * z\hat{}124 + 13553381306852782191362 * z\hat{}123$
$+ 11275683290548250326014 * z\hat{}122 + 9366968214337066432805 * z\hat{}121$
$+ 7769751185815760567386 * z\hat{}120 + 6435136440236864980530 * z\hat{}119$
$+ 5321590500505041177770 * z\hat{}118 + 4393882202107819990202 * z\hat{}117$
$+ 3622168087869313412512 * z\hat{}116 + 2981204258933224442600 * z\hat{}115$
$+ 2449668111813245172597 * z\hat{}114 + 2009575416378455376699 * z\hat{}113$
$+ 1645780025793405282694 * z\hat{}112 + 1345545089910044927383 * z\hat{}111$
$+ 1098176075074588970768 * z\hat{}110 + 894707120411771116958 * z\hat{}109$
$+ 727633370950370439347 * z\hat{}108 + 590682875616030722333 * z\hat{}107$
$+ 478622494802198274796 * z\hat{}106 + 387092990136240593846 * z\hat{}105$
$+ 312469126519206220267 * z\hat{}104 + 251741172501622684935 * z\hat{}103$
$+ 202414686892389034775 * z\hat{}102 + 162425901727911534182 * z\hat{}101$
$+ 130070392626345322594 * z\hat{}100 + 103943046339124063579 * z\hat{}99$
$+ 82887622753963631939 * z\hat{}98 + 65954447832503024640 * z\hat{}97$
$+ 52364989621449827412 * z\hat{}96 + 41482247879706710424 * z\hat{}95$
$+ 32786048682842722754 * z\hat{}94 + 25852467568577702585 * z\hat{}93$
$+ 20336723965174026026 * z\hat{}92 + 15958986961132669947 * z\hat{}91$
$+ 12492620240254453650 * z\hat{}90 + 9754465150042739033 * z\hat{}89$

$+ 7596825079281271346 * z\hat{}88 + 5900865970404594245 * z\hat{}87$

$+ 4571194445537945095 * z\hat{}86 + 3531412247947778427 * z\hat{}85$

$+ 2720479367658156444 * z\hat{}84 + 2089744840161695385 * z\hat{}83$

$+ 1600528327992056679 * z\hat{}82 + 1222154486559033326 * z\hat{}81$

$+ 930359266903148032 * z\hat{}80 + 706000605398202138 * z\hat{}79$

$+ 534018054867084596 * z\hat{}78 + 402595190403187111 * z\hat{}77$

$+ 302487097940542108 * z\hat{}76 + 226481672733845121 * z\hat{}75$

$+ 168969338444037733 * z\hat{}74 + 125600197401348791 * z\hat{}73$

$+ 93011673413771228 * z\hat{}72 + 68612694420541278 * z\hat{}71 + 50413226545875771 * z\hat{}70$

$+ 36889977806234861 * z\hat{}69 + 26880957983984055 * z\hat{}68 + 19502915614013662 * z\hat{}67$

$+ 14086923976685420 * z\hat{}66 + 10128265189415792 * z\hat{}65 + 7247590678250068 * z\hat{}64$

$+ 5160906482824844 * z\hat{}63 + 3656475430094017 * z\hat{}62 + 2577094336633797 * z\hat{}61$

$+ 1806556914093752 * z\hat{}60 + 1259344992996952 * z\hat{}59 + 872816619413251 * z\hat{}58$

$+ 601304612212699 * z\hat{}57 + 411682101148321 * z\hat{}56 + 280040960322252 * z\hat{}55$

$+ 189218325496785 * z\hat{}54 + 126960627404582 * z\hat{}53 + 84569535063645 * z\hat{}52$

$+ 55906615966000 * z\hat{}51 + 36666838286076 * z\hat{}50 + 23850076869318 * z\hat{}49$

$+ 15379661079885 * z\hat{}48 + 9827989981323 * z\hat{}47 + 6220872599178 * z\hat{}46$

$+ 3898488386409 * z\hat{}45 + 2417541432539 * z\hat{}44 + 1482640469853 * z\hat{}43$

$+ 898699750109 * z\hat{}42 + 538038461677 * z\hat{}41 + 317915388562 * z\hat{}40$

$+ 185246464839 * z\hat{}39 + 106350504743 * z\hat{}38 + 60095224039 * z\hat{}37$

$+ 33386504410 * z\hat{}36 + 18213103612 * z\hat{}35 + 9742681914 * z\hat{}34 + 5102193038 * z\hat{}33$

$+ 2611258119 * z\hat{}32 + 1303355195 * z\hat{}31 + 632958946 * z\hat{}30 + 298278811 * z\hat{}29$

$+ 135961213 * z\hat{}28 + 59731041 * z\hat{}27 + 25178481 * z\hat{}26 + 10136465 * z\hat{}25$

$+ 3869420 * z\hat{}24 + 1393384 * z\hat{}23 + 467777 * z\hat{}22 + 146193 * z\hat{}21 + 40982 * z\hat{}20$

$+ 10706 * z\hat{}19 + 2214 * z\hat{}18 + 477 * z\hat{}17)/(-z\hat{}922 - z\hat{}921 - 3 * z\hat{}920 - 6 * z\hat{}919$

$-11 * z\hat{}918 - 20 * z\hat{}917 - 35 * z\hat{}916 - 57 * z\hat{}915 - 96 * z\hat{}914 - 151 * z\hat{}913$

$-237 * z\hat{}912 - 364 * z\hat{}911 - 551 * z\hat{}910 - 818 * z\hat{}909 - 1205 * z\hat{}908 - 1745 * z\hat{}907$

$-2505 * z\hat{}906 - 3554 * z\hat{}905 - 4995 * z\hat{}904 - 6951 * z\hat{}903 - 9600 * z\hat{}902$

$-13136 * z\hat{}901 - 17851 * z\hat{}900 - 24073 * z\hat{}899 - 32248 * z\hat{}898 - 42910 * z\hat{}897$

$-56755 * z\hat{}896 - 74607 * z\hat{}895 - 97540 * z\hat{}894 - 126815 * z\hat{}893 - 164041 * z\hat{}892$

$-211123 * z\hat{}891 - 270446 * z\hat{}890 - 344823 * z\hat{}889 - 437741 * z\hat{}888 - 553306 * z\hat{}887$

$-696534 * z\hat{}886 - 873328 * z\hat{}885 - 1090829 * z\hat{}884 - 1357398 * z\hat{}883$

$-1683072 * z\hat{}882 - 2079563 * z\hat{}881 - 2560799 * z\hat{}880 - 3142987 * z\hat{}879$

$-3845262 * z\hat{}878 - 4689779 * z\hat{}877 - 5702535 * z\hat{}876 - 6913523 * z\hat{}875 - 8357679 * z\hat{}874$

$- 10075172 * z\hat{}873 - 12112525 * z\hat{}872 - 14522973 * z\hat{}871 - 17367869 * z\hat{}870$

$- 20717132 * z\hat{}869 - 24650878 * z\hat{}868 - 29260095 * z\hat{}867 - 34648504 * z\hat{}866$

$- 40933394 * z\hat{}865 - 48247884 * z\hat{}864 - 56741889 * z\hat{}863 - 66584743 * z\hat{}862$

$- 77966455 * z\hat{}861 - 91100654 * z\hat{}860 - 106226109 * z\hat{}859 - 123610153 * z\hat{}858$
$- 143550397 * z\hat{}857 - 166378625 * z\hat{}856 - 192462844 * z\hat{}855 - 222211571 * z\hat{}854$
$- 256076190 * z\hat{}853 - 294555770 * z\hat{}852 - 338199622 * z\hat{}851 - 387612617 * z\hat{}850$
$- 443458045 * z\hat{}849 - 506463299 * z\hat{}848 - 577422992 * z\hat{}847 - 657205076 * z\hat{}846$
$- 746754035 * z\hat{}845 - 847097375 * z\hat{}844 - 959348907 * z\hat{}843 - 1084715406 * z\hat{}842$
$- 1224499844 * z\hat{}841 - 1380108199 * z\hat{}840 - 1553052383 * z\hat{}839 - 1744956995 * z\hat{}838$
$- 1957561879 * z\hat{}837 - 2192728483 * z\hat{}836 - 2452441782 * z\hat{}835 - 2738816135 * z\hat{}834$
$- 3054096118 * z\hat{}833 - 3400661569 * z\hat{}832 - 3781027149 * z\hat{}831 - 4197846028 * z\hat{}830$
$- 4653907806 * z\hat{}829 - 5152140560 * z\hat{}828 - 5695606511 * z\hat{}827 - 6287501994 * z\hat{}826$
$- 6931150515 * z\hat{}825 - 7629999938 * z\hat{}824 - 8387612453 * z\hat{}823 - 9207658559 * z\hat{}822$
$- 10093903156 * z\hat{}821 - 11050195816 * z\hat{}820 - 12080452535 * z\hat{}819$
$- 13188641447 * z\hat{}818 - 14378759751 * z\hat{}817 - 15654814338 * z\hat{}816$
$- 17020793080 * z\hat{}815 - 18480639909 * z\hat{}814 - 20038219949 * z\hat{}813$
$- 21697288278 * z\hat{}812 - 23461448540 * z\hat{}811 - 25334114925 * z\hat{}810$
$- 27318463626 * z\hat{}809 - 29417387782 * z\hat{}808 - 31633441495 * z\hat{}807$
$- 33968787249 * z\hat{}806 - 36425132490 * z\hat{}805 - 39003669410 * z\hat{}804$
$- 41705003905 * z\hat{}803 - 44529088008 * z\hat{}802 - 47475141451 * z\hat{}801$
$- 50541576865 * z\hat{}800 - 53725914683 * z\hat{}799 - 57024701681 * z\hat{}798$
$- 60433419832 * z\hat{}797 - 63946399365 * z\hat{}796 - 67556722671 * z\hat{}795$
$- 71256132981 * z\hat{}794 - 75034935137 * z\hat{}793 - 78881901478 * z\hat{}792$
$- 82784171248 * z\hat{}791 - 86727156105 * z\hat{}790 - 90694440431 * z\hat{}789$
$- 94667688832 * z\hat{}788 - 98626550531 * z\hat{}787 - 102548571711 * z\hat{}786$
$- 106409107122 * z\hat{}785 - 110181241091 * z\hat{}784 - 113835709888 * z\hat{}783$
$- 117340835169 * z\hat{}782 - 120662461677 * z\hat{}781 - 123763907145 * z\hat{}780$
$- 126605919824 * z\hat{}779 - 129146649913 * z\hat{}778 - 131341631504 * z\hat{}777$
$- 133143780455 * z\hat{}776 - 134503406296 * z\hat{}775 - 135368241387 * z\hat{}774$
$- 135683488087 * z\hat{}773 - 135391884836 * z\hat{}772 - 134433793582 * z\hat{}771$
$- 132747308045 * z\hat{}770 - 130268387032 * z\hat{}769 - 126931009522 * z\hat{}768$
$- 122667358913 * z\hat{}767 - 117408030009 * z\hat{}766 - 111082268283 * z\hat{}765$
$- 103618233020 * z\hat{}764 - 94943295831 * z\hat{}763 - 84984362900 * z\hat{}762$
$- 73668235909 * z\hat{}761 - 60921996140 * z\hat{}760 - 46673429274 * z\hat{}759$
$- 30851472750 * z\hat{}758 - 13386705389 * z\hat{}757 + 5788142705 * z\hat{}756$
$+ 26737636428 * z\hat{}755 + 49523064734 * z\hat{}754 + 74201826073 * z\hat{}753$
$+ 100826799899 * z\hat{}752 + 129445675601 * z\hat{}751 + 160100272834 * z\hat{}750$
$+ 192825820478 * z\hat{}749 + 227650233516 * z\hat{}748 + 264593351810 * z\hat{}747$
$+ 303666183554 * z\hat{}746 + 344870115165 * z\hat{}745 + 388196134867 * z\hat{}744$
$+ 433624027358 * z\hat{}743 + 481121592639 * z\hat{}742 + 530643842760 * z\hat{}741$

$$+ 582132233277 * z\hat{}740 + 635513880942 * z\hat{}739 + 690700828742 * z\hat{}738$$

$$+ 747589305388 * z\hat{}737 + 806059046279 * z\hat{}736 + 865972619042 * z\hat{}735$$

$$+ 927174824283 * z\hat{}734 + 989492112423 * z\hat{}733 + 1052732091113 * z\hat{}732$$

$$+ 1116683059529 * z\hat{}731 + 1181113649733 * z\hat{}730 + 1245772506979 * z\hat{}729$$

$$+ 1310388092367 * z\hat{}728 + 1374668537454 * z\hat{}727 + 1438301637055 * z\hat{}726$$

$$+ 1500954905461 * z\hat{}725 + 1562275787097 * z\hat{}724 + 1621891942135 * z\hat{}723$$

$$+ 1679411700531 * z\hat{}722 + 1734424602554 * z\hat{}721 + 1786502121005 * z\hat{}720$$

$$+ 1835198478746 * z\hat{}719 + 1880051660347 * z\hat{}718 + 1920584526673 * z\hat{}717$$

$$+ 1956306132326 * z\hat{}716 + 1986713152552 * z\hat{}715 + 2011291519441 * z\hat{}714$$

$$+ 2029518170001 * z\hat{}713 + 2040863008033 * z\hat{}712 + 2044790977490 * z\hat{}711$$

$$+ 2040764349071 * z\hat{}710 + 2028245115728 * z\hat{}709 + 2006697596975 * z\hat{}708$$

$$+ 1975591144340 * z\hat{}707 + 1934403048050 * z\hat{}706 + 1882621533050 * z\hat{}705$$

$$+ 1819748942614 * z\hat{}704 + 1745304996476 * z\hat{}703 + 1658830217943 * z\hat{}702$$

$$+ 1559889414778 * z\hat{}701 + 1448075306910 * z\hat{}700 + 1323012182376 * z\hat{}699$$

$$+ 1184359671450 * z\hat{}698 + 1031816520525 * z\hat{}697 + 865124450145 * z\hat{}696$$

$$+ 684071978695 * z\hat{}695 + 488498292989 * z\hat{}694 + 278297045779 * z\hat{}693$$

$$+ 53420157354 * z\hat{}692 - 186118496728 * z\hat{}691 - 440239441299 * z\hat{}690$$

$$- 708794116018 * z\hat{}689 - 991561482404 * z\hat{}688 - 1288244837695 * z\hat{}687$$

$$- 1598468774110 * z\hat{}686 - 1921776392859 * z\hat{}685 - 2257626718932 * z\hat{}684$$

$$- 2605392420012 * z\hat{}683 - 2964357780509 * z\hat{}682 - 3333717029640 * z\hat{}681$$

$$- 3712572979217 * z\hat{}680 - 4099936061931 * z\hat{}679 - 4494723733472 * z\hat{}678$$

$$- 4895760319139 * z\hat{}677 - 5301777273662 * z\hat{}676 - 5711413926480 * z\hat{}675$$

$$- 6123218685593 * z\hat{}674 - 6535650760481 * z\hat{}673 - 6947082384492 * z\hat{}672$$

$$- 7355801583057 * z\hat{}671 - 7760015473435 * z\hat{}670 - 8157854130033 * z\hat{}669$$

$$- 8547375004891 * z\hat{}668 - 8926567921996 * z\hat{}667 - 9293360642204 * z\hat{}666$$

$$- 9645624999212 * z\hat{}665 - 9981183608857 * z\hat{}664 - 10297817136436 * z\hat{}663$$

$$- 10593272127691 * z\hat{}662 - 10865269370311 * z\hat{}661 - 11111512798421 * z\hat{}660$$

$$- 11329698885835 * z\hat{}659 - 11517526546789 * z\hat{}658 - 11672707471468 * z\hat{}657$$

$$- 11792976918937 * z\hat{}656 - 11876104875330 * z\hat{}655 - 11919907607396 * z\hat{}654$$

$$- 11922259496815 * z\hat{}653 - 11881105192878 * z\hat{}652 - 11794471949851 * z\hat{}651$$

$$- 11660482191548 * z\hat{}650 - 11477366150536 * z\hat{}649 - 11243474633421 * z\hat{}648$$

$$- 10957291737610 * z\hat{}647 - 10617447581302 * z\hat{}646 - 10222730853814 * z\hat{}645$$

$$- 9772101255450 * z\hat{}644 - 9264701617621 * z\hat{}643 - 8699869783646 * z\hat{}642$$

$$-8077150022004 * z\hat{}641 - 7396304065912 * z\hat{}640 - 6657321536049 * z\hat{}639$$

$$- 5860429851148 * z\hat{}638 - 5006103371235 * z\hat{}637 - 4095071892925 * z\hat{}636$$

$$- 3128328227171 * z\hat{}635 - 2107134996526 * z\hat{}634 - 1033030372397 * z\hat{}633$$

$$+ 92167096147 * z\hat{}632 + 1266354847263 * z\hat{}631 + 2487143695376 * z\hat{}630$$

$$+ 3751856537672 * z\hat{}629 + 5057528186866 * z\hat{}628 + 6400906627482 * z\hat{}627$$
$$+ 7778455486260 * z\hat{}626 + 9186358012841 * z\hat{}625 + 10620522340555 * z\hat{}624$$
$$+ 12076588324744 * z\hat{}623 + 13549935703909 * z\hat{}622 + 15035693878664 * z\hat{}621$$
$$+ 16528753033503 * z\hat{}620 + 18023776888530 * z\hat{}619 + 19515216784310 * z\hat{}618$$
$$+ 20997327382107 * z\hat{}617 + 22464183657108 * z\hat{}616 + 23909699459384 * z\hat{}615$$
$$+ 25327647299940 * z\hat{}614 + 26711679623313 * z\hat{}613 + 28055351204177 * z\hat{}612$$
$$+ 29352142919670 * z\hat{}611 + 30595486511632 * z\hat{}610 + 31778790579727 * z\hat{}609$$
$$+ 32895467402479 * z\hat{}608 + 33938960811183 * z\hat{}607 + 34902774698810 * z\hat{}606$$
$$+ 35780502376859 * z\hat{}605 + 36565856344995 * z\hat{}604 + 37252698674939 * z\hat{}603$$
$$+ 37835071563313 * z\hat{}602 + 38307228238637 * z\hat{}601 + 38663663771083 * z\hat{}600$$
$$+ 38899145958847 * z\hat{}599 + 39008745831825 * z\hat{}598 + 38987867937319 * z\hat{}597$$
$$+ 38832279948406 * z\hat{}596 + 38538141746294 * z\hat{}595 + 38102033523137 * z\hat{}594$$
$$+ 37520983048241 * z\hat{}593 + 36792491648904 * z\hat{}592 + 35914559044038 * z\hat{}591$$
$$+ 34885706594588 * z\hat{}590 + 33704999100320 * z\hat{}589 + 32372064726571 * z\hat{}588$$
$$+ 30887113187211 * z\hat{}587 + 29250951785983 * z\hat{}586 + 27464999443454 * z\hat{}585$$
$$+ 25531298338173 * z\hat{}584 + 23452523286181 * z\hat{}583 + 21231988521449 * z\hat{}582$$
$$+ 18873652002760 * z\hat{}581 + 16382116941950 * z\hat{}580 + 13762630685431 * z\hat{}579$$
$$+ 11021080682633 * z\hat{}578 + 8163987675111 * z\hat{}577 + 5198495886065 * z\hat{}576$$
$$+ 2132360348853 * z\hat{}575 - 1026068802098 * z\hat{}574 - 4267864930037 * z\hat{}573$$
$$- 7583546512870 * z\hat{}572 - 10963101626399 * z\hat{}571 - 14396015368005 * z\hat{}570$$
$$- 17871300063747 * z\hat{}569 - 21377528282686 * z\hat{}568 - 24902868492491 * z\hat{}567$$
$$- 28435123328032 * z\hat{}566 - 31961770304709 * z\hat{}565 - 35470004891211 * z\hat{}564$$
$$- 38946785772444 * z\hat{}563 - 42378882165911 * z\hat{}562 - 45752923018659 * z\hat{}561$$
$$- 49055447898002 * z\hat{}560 - 52272959406997 * z\hat{}559 - 55391976884742 * z\hat{}558$$
$$- 58399091230056 * z\hat{}557 - 61281020562627 * z\hat{}556 - 64024666566369 * z\hat{}555$$
$$- 66617171187055 * z\hat{}554 - 69045973543565 * z\hat{}553 - 71298866681553 * z\hat{}552$$
$$- 73364054048841 * z\hat{}551 - 75230205286587 * z\hat{}550 - 76886511234815 * z\hat{}549$$
$$- 78322737717973 * z\hat{}548 - 79529278036997 * z\hat{}547 - 80497203704759 * z\hat{}546$$
$$- 81218313386226 * z\hat{}545 - 81685179560341 * z\hat{}544 - 81891192898956 * z\hat{}543$$
$$- 81830603868092 * z\hat{}542 - 81498561588480 * z\hat{}541 - 80891149449145 * z\hat{}540$$
$$- 80005417559818 * z\hat{}539 - 78839411532550 * z\hat{}538 - 77392197725764 * z\hat{}537$$
$$- 75663884446393 * z\hat{}536 - 73655639296540 * z\hat{}535 - 71369702165159 * z\hat{}534$$
$$- 68809394111079 * z\hat{}533 - 65979121649301 * z\hat{}532 - 62884376743521 * z\hat{}531$$
$$- 59531732036652 * z\hat{}530 - 55928831681657 * z\hat{}529 - 52084377322774 * z\hat{}528$$
$$- 48008109654030 * z\hat{}527 - 43710785126953 * z\hat{}526 - 39204148293913 * z\hat{}525$$
$$- 34500899387926 * z\hat{}524 - 29614657683820 * z\hat{}523 - 24559920267434 * z\hat{}522$$
$$- 19352016819002 * z\hat{}521 - 14007060063855 * z\hat{}520 - 8541892550283 * z\hat{}519$$

$- 2974029439178 * z\hat{}518 + 2678401985583 * z\hat{}517 + 8396726374997 * z\hat{}516$
$+ 14161787182506 * z\hat{}515 + 19954016606589 * z\hat{}514 + 25753507810139 * z\hat{}513$
$+ 31540089654790 * z\hat{}512 + 37293403120573 * z\hat{}511 + 42992979628597 * z\hat{}510$
$+ 48618320407237 * z\hat{}509 + 54148977104672 * z\hat{}508 + 59564632769592 * z\hat{}507$
$+ 64845183386312 * z\hat{}506 + 69970819078295 * z\hat{}505 + 74922105157115 * z\hat{}504$
$+ 79680062125888 * z\hat{}503 + 84226244812497 * z\hat{}502 + 88542819748542 * z\hat{}501$
$+ 92612640966147 * z\hat{}500 + 96419323347865 * z\hat{}499 + 99947313706064 * z\hat{}498$
$+ 103181958748830 * z\hat{}497 + 106109570121666 * z\hat{}496 + 108717485714578 * z\hat{}495$
$+ 110994127434743 * z\hat{}494 + 112929054679827 * z\hat{}493 + 114513013728410 * z\hat{}492$
$+ 115737982329708 * z\hat{}491 + 116597209733715 * z\hat{}490 + 117085251499206 * z\hat{}489$
$+ 117197999342231 * z\hat{}488 + 116932705429178 * z\hat{}487 + 116288001401534 * z\hat{}486$
$+ 115263911602478 * z\hat{}485 + 113861860823686 * z\hat{}484 + 112084676113403 * z\hat{}483$
$+ 109936582989912 * z\hat{}482 + 107423195681730 * z\hat{}481 + 104551501763773 * z\hat{}480$
$+ 101329840887539 * z\hat{}479 + 97767878004692 * z\hat{}478 + 93876570859121 * z\hat{}477$
$+ 89668132169332 * z\hat{}476 + 85155986359556 * z\hat{}475 + 80354721279612 * z\hat{}474$
$+ 75280034848995 * z\hat{}473 + 69948677085624 * z\hat{}472 + 64378387528208 * z\hat{}471$
$+ 58587828523189 * z\hat{}470 + 52596514461970 * z\hat{}469 + 46424737442634 * z\hat{}468$
$+ 40093489509295 * z\hat{}467 + 33624381947278 * z\hat{}466 + 27039561847385 * z\hat{}465$
$+ 20361626409881 * z\hat{}464 + 13613535262876 * z\hat{}463 + 6818521248172 * z\hat{}462$
$- 6818521248172 * z\hat{}460 - 13613535262876 * z\hat{}459 - 20361626409881 * z\hat{}458$
$- 27039561847385 * z\hat{}457 - 33624381947278 * z\hat{}456 - 40093489509295 * z\hat{}455$
$- 46424737442634 * z\hat{}454 - 52596514461970 * z\hat{}453 - 58587828523189 * z\hat{}452$
$- 64378387528208 * z\hat{}451 - 69948677085624 * z\hat{}450 - 75280034848995 * z\hat{}449$
$- 80354721279612 * z\hat{}448 - 85155986359556 * z\hat{}447 - 89668132169332 * z\hat{}446$
$- 93876570859121 * z\hat{}445 - 97767878004692 * z\hat{}444 - 101329840887539 * z\hat{}443$
$- 104551501763773 * z\hat{}442 - 107423195681730 * z\hat{}441 - 109936582989912 * z\hat{}440$
$- 112084676113403 * z\hat{}439 - 113861860823686 * z\hat{}438 - 115263911602478 * z\hat{}437$
$- 116288001401534 * z\hat{}436 - 116932705429178 * z\hat{}435 - 117197999342231 * z\hat{}434$
$- 117085251499206 * z\hat{}433 - 116597209733715 * z\hat{}432 - 115737982329708 * z\hat{}431$
$- 114513013728410 * z\hat{}430 - 112929054679827 * z\hat{}429 - 110994127434743 * z\hat{}428$
$- 108717485714578 * z\hat{}427 - 106109570121666 * z\hat{}426 - 103181958748830 * z\hat{}425$
$-99947313706064 * z\hat{}424 - 96419323347865 * z\hat{}423 - 92612640966147 * z\hat{}422$
$- 88542819748542 * z\hat{}421 - 84226244812497 * z\hat{}420 - 79680062125888 * z\hat{}419$
$- 74922105157115 * z\hat{}418 - 69970819078295 * z\hat{}417 - 64845183386312 * z\hat{}416$
$- 59564632769592 * z\hat{}415 - 54148977104672 * z\hat{}414 - 48618320407237 * z\hat{}413$
$- 42992979628597 * z\hat{}412 - 37293403120573 * z\hat{}411 - 31540089654790 * z\hat{}410$
$- 25753507810139 * z\hat{}409 - 19954016606589 * z\hat{}408 - 14161787182506 * z\hat{}407$

$$
\begin{aligned}
&- 8396726374997 \ast z\hat{}406 - 2678401985583 \ast z\hat{}405 + 2974029439178 \ast z\hat{}404 \\
&+ 8541892550283 \ast z\hat{}403 + 14007060063855 \ast z\hat{}402 + 19352016819002 \ast z\hat{}401 \\
&+ 24559920267434 \ast z\hat{}400 + 29614657683820 \ast z\hat{}399 + 34500899387926 \ast z\hat{}398 \\
&+ 39204148293913 \ast z\hat{}397 + 43710785126953 \ast z\hat{}396 + 48008109654030 \ast z\hat{}395 \\
&+ 52084377322774 \ast z\hat{}394 + 55928831681657 \ast z\hat{}393 + 59531732036652 \ast z\hat{}392 \\
&+ 62884376743521 \ast z\hat{}391 + 65979121649301 \ast z\hat{}390 + 68809394111079 \ast z\hat{}389 \\
&+ 71369702165159 \ast z\hat{}388 + 73655639296540 \ast z\hat{}387 + 75663884446393 \ast z\hat{}386 \\
&+ 77392197725764 \ast z\hat{}385 + 78839411532550 \ast z\hat{}384 + 80005417559818 \ast z\hat{}383 \\
&+ 80891149449145 \ast z\hat{}382 + 81498561588480 \ast z\hat{}381 + 81830603868092 \ast z\hat{}380 \\
&+ 81891192898956 \ast z\hat{}379 + 81685179560341 \ast z\hat{}378 + 81218313386226 \ast z\hat{}377 \\
&+ 80497203704759 \ast z\hat{}376 + 79529278036997 \ast z\hat{}375 + 78322737717973 \ast z\hat{}374 \\
&+ 76886511234815 \ast z\hat{}373 + 75230205286587 \ast z\hat{}372 + 73364054048841 \ast z\hat{}371 \\
&+ 71298866681553 \ast z\hat{}370 + 69045973543565 \ast z\hat{}369 + 66617171187055 \ast z\hat{}368 \\
&+ 64024666566369 \ast z\hat{}367 + 61281020562627 \ast z\hat{}366 + 58399091230056 \ast z\hat{}365 \\
&+ 55391976884742 \ast z\hat{}364 + 52272959406997 \ast z\hat{}363 + 49055447898002 \ast z\hat{}362 \\
&+ 45752923018659 \ast z\hat{}361 + 42378882165911 \ast z\hat{}360 + 38946785772444 \ast z\hat{}359 \\
&+ 35470004891211 \ast z\hat{}358 + 31961770304709 \ast z\hat{}357 + 28435123328032 \ast z\hat{}356 \\
&+ 24902868492491 \ast z\hat{}355 + 21377528282686 \ast z\hat{}354 + 17871300063747 \ast z\hat{}353 \\
&+ 14396015368005 \ast z\hat{}352 + 10963101626399 \ast z\hat{}351 + 7583546512870 \ast z\hat{}350 \\
&+ 4267864930037 \ast z\hat{}349 + 1026068802098 \ast z\hat{}348 - 2132360348853 \ast z\hat{}347 \\
&- 5198495886065 \ast z\hat{}346 - 8163987675111 \ast z\hat{}345 - 11021080682633 \ast z\hat{}344 \\
&- 13762630685431 \ast z\hat{}343 - 16382116941950 \ast z\hat{}342 - 18873652002760 \ast z\hat{}341 \\
&- 21231988521449 \ast z\hat{}340 - 23452523286181 \ast z\hat{}339 - 25531298338173 \ast z\hat{}338 \\
&- 27464999443454 \ast z\hat{}337 - 29250951785983 \ast z\hat{}336 - 30887113187211 \ast z\hat{}335 \\
&- 32372064726571 \ast z\hat{}334 - 33704999100320 \ast z\hat{}333 - 34885706594588 \ast z\hat{}332 \\
&- 35914559044038 \ast z\hat{}331 - 36792491648904 \ast z\hat{}330 - 37520983048241 \ast z\hat{}329 \\
&- 38102033523137 \ast z\hat{}328 - 38538141746294 \ast z\hat{}327 - 38832279948406 \ast z\hat{}326 \\
&- 38987867937319 \ast z\hat{}325 - 39008745831825 \ast z\hat{}324 - 38899145958847 \ast z\hat{}323 \\
&- 38663663771083 \ast z\hat{}322 - 38307228238637 \ast z\hat{}321 - 37835071563313 \ast z\hat{}320 \\
&- 37252698674939 \ast z\hat{}319 - 36565856344995 \ast z\hat{}318 - 35780502376859 \ast z\hat{}317 \\
&- 34902774698810 \ast z\hat{}316 - 33938960811183 \ast z\hat{}315 - 32895467402479 \ast z\hat{}314 \\
&- 31778790579727 \ast z\hat{}313 - 30595486511632 \ast z\hat{}312 - 29352142919670 \ast z\hat{}311 \\
&- 28055351204177 \ast z\hat{}310 - 26711679623313 \ast z\hat{}309 - 25327647299940 \ast z\hat{}308 \\
&- 23909699459384 \ast z\hat{}307 - 22464183657108 \ast z\hat{}306 - 20997327382107 \ast z\hat{}305 \\
&- 19515216784310 \ast z\hat{}304 - 18023776888530 \ast z\hat{}303 - 16528753033503 \ast z\hat{}302 \\
&- 15035693878664 \ast z\hat{}301 - 13549935703909 \ast z\hat{}300 - 12076588324744 \ast z\hat{}299 \\
&- 10620522340555 \ast z\hat{}298 - 9186358012841 \ast z\hat{}297 - 7778455486260 \ast z\hat{}296
\end{aligned}
$$

$$
\begin{aligned}
&- 6400906627482 * z\hat{}295 - 5057528186866 * z\hat{}294 - 3751856537672 * z\hat{}293\\
&- 2487143695376 * z\hat{}292 - 1266354847263 * z\hat{}291 - 92167096147 * z\hat{}290\\
&\quad +1033030372397 * z\hat{}289 + 2107134996526 * z\hat{}288 + 3128328227171 * z\hat{}287\\
&+ 4095071892925 * z\hat{}286 + 5006103371235 * z\hat{}285 + 5860429851148 * z\hat{}284\\
&+ 6657321536049 * z\hat{}283 + 7396304065912 * z\hat{}282 + 8077150022004 * z\hat{}281\\
&+ 8699869783646 * z\hat{}280 + 9264701617621 * z\hat{}279 + 9772101255450 * z\hat{}278\\
&+ 10222730853814 * z\hat{}277 + 10617447581302 * z\hat{}276 + 10957291737610 * z\hat{}275\\
&+ 11243474633421 * z\hat{}274 + 11477366150536 * z\hat{}273 + 11660482191548 * z\hat{}272\\
&+ 11794471949851 * z\hat{}271 + 11881105192878 * z\hat{}270 + 11922259496815 * z\hat{}269\\
&+ 11919907607396 * z\hat{}268 + 11876104875330 * z\hat{}267 + 11792976918937 * z\hat{}266\\
&+ 11672707471468 * z\hat{}265 + 11517526546789 * z\hat{}264 + 11329698885835 * z\hat{}263\\
&+ 11111512798421 * z\hat{}262 + 10865269370311 * z\hat{}261 + 10593272127691 * z\hat{}260\\
&+ 10297817136436 * z\hat{}259 + 9981183608857 * z\hat{}258 + 9645624999212 * z\hat{}257\\
&+ 9293360642204 * z\hat{}256 + 8926567921996 * z\hat{}255 + 8547375004891 * z\hat{}254\\
&+ 8157854130033 * z\hat{}253 + 7760015473435 * z\hat{}252 + 7355801583057 * z\hat{}251\\
&+ 6947082384492 * z\hat{}250 + 6535650760481 * z\hat{}249 + 6123218685593 * z\hat{}248\\
&+ 5711413926480 * z\hat{}247 + 5301777273662 * z\hat{}246 + 4895760319139 * z\hat{}245\\
&+ 4494723733472 * z\hat{}244 + 4099936061931 * z\hat{}243 + 3712572979217 * z\hat{}242\\
&+ 3333717029640 * z\hat{}241 + 2964357780509 * z\hat{}240 + 2605392420012 * z\hat{}239\\
&+ 2257626718932 * z\hat{}238 + 1921776392859 * z\hat{}237 + 1598468774110 * z\hat{}236\\
&+ 1288244837695 * z\hat{}235 + 991561482404 * z\hat{}234 + 708794116018 * z\hat{}233\\
&+ 440239441299 * z\hat{}232 + 186118496728 * z\hat{}231 - 53420157354 * z\hat{}230\\
&- 278297045779 * z\hat{}229 - 488498292989 * z\hat{}228 - 684071978695 * z\hat{}227\\
&- 865124450145 * z\hat{}226 - 1031816520525 * z\hat{}225 - 1184359671450 * z\hat{}224\\
&- 1323012182376 * z\hat{}223 - 1448075306910 * z\hat{}222 - 1559889414778 * z\hat{}221\\
&- 1658830217943 * z\hat{}220 - 1745304996476 * z\hat{}219 - 1819748942614 * z\hat{}218\\
&- 1882621533050 * z\hat{}217 - 1934403048050 * z\hat{}216 - 1975591144340 * z\hat{}215\\
&- 2006697596975 * z\hat{}214 - 2028245115728 * z\hat{}213 - 2040764349071 * z\hat{}212\\
&- 2044790977490 * z\hat{}211 - 2040863008033 * z\hat{}210 - 2029518170001 * z\hat{}209\\
&- 2011291519441 * z\hat{}208 - 1986713152552 * z\hat{}207 - 1956306132326 * z\hat{}206\\
&- 1920584526673 * z\hat{}205 - 1880051660347 * z\hat{}204 - 1835198478746 * z\hat{}203\\
&- 1786502121005 * z\hat{}202 - 1734424602554 * z\hat{}201 - 1679411700531 * z\hat{}200\\
&- 1621891942135 * z\hat{}199 - 1562275787097 * z\hat{}198 - 1500954905461 * z\hat{}197\\
&- 1438301637055 * z\hat{}196 - 1374668537454 * z\hat{}195 - 1310388092367 * z\hat{}194\\
&- 1245772506979 * z\hat{}193 - 1181113649733 * z\hat{}192 - 1116683059529 * z\hat{}191\\
&- 1052732091113 * z\hat{}190 - 989492112423 * z\hat{}189 - 927174824283 * z\hat{}188\\
&- 865972619042 * z\hat{}187 - 806059046279 * z\hat{}186 - 747589305388 * z\hat{}185
\end{aligned}
$$

$- 690700828742 * z\hat{}184 - 635513880942 * z\hat{}183 - 582132233277 * z\hat{}182$
$- 530643842760 * z\hat{}181 - 481121592639 * z\hat{}180 - 433624027358 * z\hat{}179$
$- 388196134867 * z\hat{}178 - 344870115165 * z\hat{}177 - 303666183554 * z\hat{}176$
$- 264593351810 * z\hat{}175 - 227650233516 * z\hat{}174 - 192825820478 * z\hat{}173$
$- 160100272834 * z\hat{}172 - 129445675601 * z\hat{}171 - 100826799899 * z\hat{}170$
$-74201826073 * z\hat{}169 - 49523064734 * z\hat{}168 - 26737636428 * z\hat{}167 - 5788142705 * z\hat{}166$
$+ 13386705389 * z\hat{}165 + 30851472750 * z\hat{}164 + 46673429274 * z\hat{}163$
$+ 60921996140 * z\hat{}162 + 73668235909 * z\hat{}161 + 84984362900 * z\hat{}160$
$+ 94943295831 * z\hat{}159 + 103618233020 * z\hat{}158 + 111082268283 * z\hat{}157$
$+ 117408030009 * z\hat{}156 + 122667358913 * z\hat{}155 + 126931009522 * z\hat{}154$
$+ 130268387032 * z\hat{}153 + 132747308045 * z\hat{}152 + 134433793582 * z\hat{}151$
$+ 135391884836 * z\hat{}150 + 135683488087 * z\hat{}149 + 135368241387 * z\hat{}148$
$+ 134503406296 * z\hat{}147 + 133143780455 * z\hat{}146 + 131341631504 * z\hat{}145$
$+ 129146649913 * z\hat{}144 + 126605919824 * z\hat{}143 + 123763907145 * z\hat{}142$
$+ 120662461677 * z\hat{}141 + 117340835169 * z\hat{}140 + 113835709888 * z\hat{}139$
$+ 110181241091 * z\hat{}138 + 106409107122 * z\hat{}137 + 102548571711 * z\hat{}136$
$+ 98626550531 * z\hat{}135 + 94667688832 * z\hat{}134 + 90694440431 * z\hat{}133$
$+ 86727156105 * z\hat{}132 + 82784171248 * z\hat{}131 + 78881901478 * z\hat{}130$
$+ 75034935137 * z\hat{}129 + 71256132981 * z\hat{}128 + 67556722671 * z\hat{}127$
$+ 63946399365 * z\hat{}126 + 60433419832 * z\hat{}125 + 57024701681 * z\hat{}124$
$+ 53725914683 * z\hat{}123 + 50541576865 * z\hat{}122 + 47475141451 * z\hat{}121$
$+ 44529088008 * z\hat{}120 + 41705003905 * z\hat{}119 + 39003669410 * z\hat{}118$
$+ 36425132490 * z\hat{}117 + 33968787249 * z\hat{}116 + 31633441495 * z\hat{}115$
$+ 29417387782 * z\hat{}114 + 27318463626 * z\hat{}113 + 25334114925 * z\hat{}112$
$+ 23461448540 * z\hat{}111 + 21697288278 * z\hat{}110 + 20038219949 * z\hat{}109$
$+ 18480639909 * z\hat{}108 + 17020793080 * z\hat{}107 + 15654814338 * z\hat{}106$
$+ 14378759751 * z\hat{}105 + 13188641447 * z\hat{}104 + 12080452535 * z\hat{}103$
$+11050195816 * z\hat{}102 + 10093903156 * z\hat{}101 + 9207658559 * z\hat{}100 + 8387612453 * z\hat{}99$
$+ 7629999938 * z\hat{}98 + 6931150515 * z\hat{}97 + 6287501994 * z\hat{}96 + 5695606511 * z\hat{}95$
$+ 5152140560 * z\hat{}94 + 4653907806 * z\hat{}93 + 4197846028 * z\hat{}92 + 3781027149 * z\hat{}91$
$+ 3400661569 * z\hat{}90 + 3054096118 * z\hat{}89 + 2738816135 * z\hat{}88 + 2452441782 * z\hat{}87$
$+ 2192728483 * z\hat{}86 + 1957561879 * z\hat{}85 + 1744956995 * z\hat{}84 + 1553052383 * z\hat{}83$
$+ 1380108199 * z\hat{}82 + 1224499844 * z\hat{}81 + 1084715406 * z\hat{}80 + 959348907 * z\hat{}79$
$+ 847097375 * z\hat{}78 + 746754035 * z\hat{}77 + 657205076 * z\hat{}76 + 577422992 * z\hat{}75$
$+ 506463299 * z\hat{}74 + 443458045 * z\hat{}73 + 387612617 * z\hat{}72 + 338199622 * z\hat{}71$
$+ 294555770 * z\hat{}70 + 256076190 * z\hat{}69 + 222211571 * z\hat{}68 + 192462844 * z\hat{}67$
$+ 166378625 * z\hat{}66 + 143550397 * z\hat{}65 + 123610153 * z\hat{}64 + 106226109 * z\hat{}63$

$+ 91100654 * z\hat{}62 + 77966455 * z\hat{}61 + 66584743 * z\hat{}60 + 56741889 * z\hat{}59$
$+ 48247884 * z\hat{}58 + 40933394 * z\hat{}57 + 34648504 * z\hat{}56 + 29260095 * z\hat{}55$
$+ 24650878 * z\hat{}54 + 20717132 * z\hat{}53 + 17367869 * z\hat{}52 + 14522973 * z\hat{}51$
$+ 12112525 * z\hat{}50 + 10075172 * z\hat{}49 + 8357679 * z\hat{}48 + 6913523 * z\hat{}47 + 5702535 * z\hat{}46$
$+ 4689779 * z\hat{}45 + 3845262 * z\hat{}44 + 3142987 * z\hat{}43 + 2560799 * z\hat{}42 + 2079563 * z\hat{}41$
$+ 1683072 * z\hat{}40 + 1357398 * z\hat{}39 + 1090829 * z\hat{}38 + 873328 * z\hat{}37 + 696534 * z\hat{}36$
$+ 553306 * z\hat{}35 + 437741 * z\hat{}34 + 344823 * z\hat{}33 + 270446 * z\hat{}32 + 211123 * z\hat{}31$
$+ 164041 * z\hat{}30 + 126815 * z\hat{}29 + 97540 * z\hat{}28 + 74607 * z\hat{}27 + 56755 * z\hat{}26 + 42910 * z\hat{}25$
$+ 32248 * z\hat{}24 + 24073 * z\hat{}23 + 17851 * z\hat{}22 + 13136 * z\hat{}21 + 9600 * z\hat{}20 + 6951 * z\hat{}19$
$+ 4995 * z\hat{}18 + 3554 * z\hat{}17 + 2505 * z\hat{}16 + 1745 * z\hat{}15 + 1205 * z\hat{}14 + 818 * z\hat{}13$
$+ 551 * z\hat{}12 + 364 * z\hat{}11 + 237 * z\hat{}10 + 151 * z\hat{}9 + 96 * z\hat{}8 + 57 * z\hat{}7 + 35 * z\hat{}6 + 20 * z\hat{}5$
$+ 11 * z\hat{}4 + 6 * z\hat{}3 + 3 * z\hat{}2 + z + 1)$