THE FACES OF TORIC ARRANGEMENTS

JUNSHAN CHEN AND AMANDA YAO
MENTOR: CHRISTIN BIBBY

ABSTRACT. This research is about toric arrangements which is a multiplicative analog of hyperplane arrangements. We extend the Deletion-restriction Theorem and Whitney’s Theorem to toric arrangement to prove the recurrence of number of regions and the recurrence of characteristic polynomial. Also, we give two alternative proofs for [ERS09] counting the number of regions of an essential toric arrangement. We also proved the cd-indexability and coefficient-symmetry for the reduced flag h-polynomial of the poset of faces. And we studied a special case of toric arrangement, “coordinate toric arrangement”, explicitly. On the other hand, we implemented a toric arrangement class by SageMath to generate examples, especially those of high dimensions which is hard to write by hand and made some conjectures based on our data. Some conjectures is well proved in our research, and some still remains to be studied.

1. INTRODUCTION

Traditionally, combinatorial topologists have been interested in hyperplane arrangements. Ehrenborg, Readdy and Slone did a multiplicative analog of hyperplane arrangement by studying arrangements on the torus, which is called toric arrangement. Our research is conducted based on some of their work and we are also inspired a lot by Stanley’s study in hyperplane arrangement.

In our research, we did some analog versions of Stanley’s theorem in toric arrangement, and modified some definitions from ERS and observed patterns worth study. In Section 2, we give basic definitions on hyperplane arrangements (from [Sta07]) and our analog on toric arrangements. But different from ERS’s definition, we removed the empty face when construct the poset of layers and poset of faces, and defined the same polynomials on the modified posets: characteristic polynomial, f-polynomial, h-polynomial, reduced flag f-polynomial, reduced flag h-polynomial. And states some interesting relations between the coefficients of those polynomials.

In Section 3, We discovered some properties of the coefficient relationship between f-polynomial and h-polynomial, flag f-polynomial and flag h-polynomial. and gave an alternative formula to [ERS09, Theorem 4.2], to generate the coefficients of flag f-polynomial from characteristic polynomial under Regular Cell Complex Assumption, and give an intuitive proof to visualize the inner meaning of this formula. Last but not least, we give
In Section 4, we did an analog version of the Deletion-Restriction Theorem in [Sta07, Lemma 2.2] from hyperplane arrangement to toric arrangement and find modified recurrence for regions on the deletion and restriction. We also use the Cross-Cut Theorem to prove a toric arrangement analog of Whitney’s Theorem, with which we prove the recurrence of characteristic polynomial that holds for hyperplane arrangement still hold for toric arrangement. ERS also brought out another theorem [ERS09, Theorem 3.6] for counting regions of an essential toric arrangement. Then we provided two alternative proof, one is a recurrent proof using the recurrence of regions and characteristic polynomial, and the other is using Möbius inversion.

In Section 5, we prove some properties which can be observed after removing the empty face in both posets. As an extension of [ERS09, Corollary 2.12], we proved that the flag h-polynomial is always cd-indexable, i.e., the cd-index form exists under regular cell complex. We established and proved an algorithm to generate the cd-index from flag h-polynomial, which is used in our Sage program. Also, we observed a nice symmetry of the coefficients in flag h-polynomial for poset of faces, which we have also proved by cd-indexibility.

In Section 6, we discuss a special case called coordinate toric arrangement, which is a direct analog from the coordinate arrangement in hyperplane arrangement. We give the explicit formula for characteristic polynomial, f-polynomial, flag f-polynomial for poset of layers and poset of faces (respectively).

In Section 7, we introduce our Sage program and some algorithms we used in the program. We also attached our code, hoping mathematicians studying this topic can save time using this program to generate examples especially for those of higher dimension which is hard to write by hand.

In Section 8, we paste three collections of our data to support our theorems and lemmas and make some conjectures based on the data we observed.

Acknowledgements. We are grateful to the University of Michigan REU program for their support and the opportunity to pursue this project. Also, we want to give special thanks to our mentor Christin Bibby.

2. Preliminaries

2.1. Hyperplane Arrangement. Hyperplane Arrangement is a useful tool to study polytopes in the field of geometry. Let \( V \cong K^n \) be a vector space, where \( K \) is a field. But for the convenience of our further discussion, we take \( K = \mathbb{R} \).

Definition 2.1. A Hyperplane is a vector subspace \( H \subseteq V \), whose dimension is one less than that of its ambient vector space \( V \).

There are two categories of hyperplanes, linear hyperplane and affine hyperplane. Let \( \alpha = (a_1, \cdots, a_n) \in V \), define a linear transformation \( T_\alpha \in \text{Hom}(V, \mathbb{R}) \) by \( T_\alpha(v) = \alpha \cdot v = \sum_{i=1}^{n} a_i v_i \) where \( v = (v_1, \cdots, v_n) \in V \), \( \alpha \cdot v \) is just the normal dot product.
**Definition 2.2.** A linear hyperplane is an \((n-1)\) dimensional subspace \(H\) of \(V\), i.e.

\[ H = \{ v \in V : T_\alpha(v) = 0 \} \]

where \(T_\alpha \in \text{Hom}(V, \mathbb{R})\). Note that \(H\) is really the kernel of \(T_\alpha\).

An affine hyperplane is a translate \(J\) of a linear hyperplane, i.e.

\[ J = \{ v \in V : T_\alpha(v) = c \} \]

where \(T_\alpha \in \text{Hom}(V, \mathbb{R})\), \(c \in \mathbb{R}\).

We will call \(\alpha\) the normal vector of \(H\) and \(J\).

Now we can collect a finite set of affine hyperplanes and study its arrangement.

**Definition 2.3.** A Hyperplane Arrangement \(A\) is a finite set of affine hyperplanes in some vector space \(V \cong \mathbb{R}^n\).

Let \(A = \{H_i | i \in \mathbb{N}, 1 \leq i \leq l\}\) be an hyperplane arrangement defined by \(H_i = \{ v \in V : T_{\alpha_i}(v) = c_i \}\) where \(\alpha_i \in V\), \(c_i \in \mathbb{R}\). Then we can use the matrix taking \([-\alpha_i | c_i]\) as rows to represent \(A\).

\[
\begin{bmatrix}
-\alpha_1 & c_1 \\
-\alpha_2 & c_2 \\
\vdots \\
-\alpha_l & c_l
\end{bmatrix}
\]

For example, the following matrix indicate a hyperplane arrangement with four hyperplanes:

\[
\begin{bmatrix}
1 & -1 & 0 \\
1 & 1 & 0 \\
1 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix} \rightarrow \begin{cases} x - y = 0 \\ x + y = 0 \\ x = 0 \\ y = 0 \end{cases}
\]

Given a set of hyperplanes in \(V\), there will be intersections of different dimensions. Let \(L(A) = \bigcap_{H \in \mathcal{B}} \{B \subseteq A\}\). Then \(L(A)\) is the collection of all possible intersections of hyperplanes in \(A\).

A subarrangement of \(A\) is a subset \(\mathcal{B} \subseteq A\), where \(\mathcal{B}\) is also an arrangement in \(V\). For \(x \in L(A)\), define the subarrangement \(A_x \subseteq A\) by

\[ A_x = \{ H \in A : x \subseteq H \} \]

Also define an arrangement \(A^x\) in the affine subspace \(x \in L(A)\) by

\[ A^x = \{ x \cap H \neq \emptyset : H \in A - A_x \} \]

**Definition 2.4.** Fix \(H_0 \in A\). The deletion of an arrangement \(A\) on \(H_0\) is defined as \(A' = A - \{H_0\}\).

**Definition 2.5.** Fix \(H_0 \in A\). The restriction of an arrangement \(A\) on \(H_0\) is defined as \(A'' = A^{H_0} = \{ H_0 \cap H \neq \emptyset : H \in A - A_{H_0} \}\). But really \(A'' = A^{H_0}\)
2.2. Toric Arrangement. Toric arrangement is defined on a space where each dimension is a unit circle $S^1 = \{ z \in \mathbb{C} | |z| = 1 \}$. Note that $S^1$ is really a quotient group $S^1 \cong \mathbb{R} \setminus \mathbb{Z}$. The groups $(\mathbb{R}, +)$ and group $(S^1, \cdot)$ are homomorphic since
\[ e^{i\theta_1}e^{i\theta_2} = e^{i(\theta_1 + \theta_2)} \]
\[ e^{2\pi i} = 1. \]
Therefore, we can construct a multiplicative analog of hyperplane arrangement

**Definition 2.6.** An $n$-torus (or simply a torus when $n$ is understood) is the set
\[ T := \{(x_1, x_2, \ldots, x_n) | \forall i, x_i \in S^1 \} \]
We can also write $T := (S^1)^n$.

The multiplication on $S^1$ induced a (coordinate-wise) multiplication on $T$ as follows:
\[ (x_1, x_2, \cdots, x_n) \cdot (y_1, y_2, \cdots, y_n) = (x_1y_1, x_2y_2, \cdots, x_ny_n) \]

Before define the analog of hyperplane in our n-torus space, let’s define our group homomorphism with multiplication operation first.

Define a group isomorphism $g : Hom(T, S^1) \to \mathbb{Z}^n$

**Definition 2.7.** Given $\alpha \in Hom(T, S^1)$, with $g(\alpha) = (a_1, a_2, \ldots, a_n) \in \mathbb{Z}^n$. Then for $x = (x_1, x_2, \ldots, x_n) \in T$, define $\alpha(x_1, x_2, \ldots, x_n) = x_1^{a_1}x_2^{a_2}\cdots x_n^{a_n}$.

In facet, all group homomorphisms $T \to S^1$ are of the form described above.

Now we can define our hypertorus:

**Definition 2.8.** Given $\alpha \in Hom(T, S^1)$ with $\alpha \neq 0$, we can define a linear hypertorus as the set $H_\alpha := \{ x \in T | \alpha(x) = 1 \}$. Note $H_\alpha$ is really the kernel of $\alpha$ since $1 = e^{2\pi i}$ is the identity in $S^1$.

Since a hypertorus is defined as the kernel of a linear transformation, it should be of dimension $n-1$.

Similarly, we can also define affine hypertorus as a translate of a linear hypertorus:

**Definition 2.9.** An affine hypertorus is a translate $J$ of a linear hypertorus, i.e.
\[ J_\alpha := \{ x \in T | \alpha(x) = c, c \in (0, 1] \}. \]

Now we can collect a finite set of affine hypertorus and study its arrangement.

**Definition 2.10.** A toric arrangement is a finite set of affine hypertori in $T$. We may denote the arrangement by $\mathcal{A} = \{ H_{\alpha_1c_1}, H_{\alpha_2c_2}, \ldots, H_{\alpha_n c_n} \}$, where $H_{\alpha_i c_i} = \{ x \in T | \alpha_i(x) = c_i \}$.

Using the bijection $g : Hom(T, S^1) \to \mathbb{Z}^n$, we can use an associated matrix to represent our toric arrangement:
where $c_i \in (0, 1]$.

In this case, each row represents a hypertorus: If $g(\alpha_i) = (a_{i1}, a_{i2}, \ldots, a_{in})$, then the equation of the hypertorus is $x_1^{a_{i1}}x_2^{a_{i2}}\ldots x_n^{a_{in}} = e^{2\pi i c_i}$.

We can also study the deletion and restriction on toric arrangement.

**Definition 2.11.** Given a toric arrangement $\mathcal{A}$, fix $H_0 \in \mathcal{A}$, we define:
the deletion of $H_0$ to be $\mathcal{A}' = \mathcal{A} - \{H_0\}$,
and the restriction of $\mathcal{A}$ on $H_0$ to be $\mathcal{A}'' = \{\text{the connected components of } H \cap H_0 \neq \emptyset | H \in \mathcal{A} \}'$.

2.3. Poests. We are interested in the poset of layers, poset of faces of our toric arrangement.

**Definition 2.12.** We say that a layer of $\mathcal{A}$ is a connected component of an intersection of some hypertori.

Let $(\mathcal{A})$ be the collection of all layers, i.e. $L(\mathcal{A}) = \{\text{connected components of } \bigcap_{H \in B} H | B \subseteq \mathcal{A} \}$.

**Definition 2.13.** Order $L(\mathcal{A})$ by inclusion, that is $Y \leq Z \iff Y \subseteq Z$, we can construct the poset of layers $\mathcal{P}$.

Notice that in the literature, it is more common to define the posets by reverse inclusion, which we will denote by $\mathcal{P}^{op}$. We use inclusion since it will be helpful for our future proof.

Our definition of poset of layers is different from [ERS09], since do not include empty space as an element in our poset.

**Example 2.14.** For example, a toric arrangement with associated matrix:

\[
\begin{bmatrix}
2 & 0 & 1 \\
0 & 2 & 1 \\
1 & 1 & 1 \\
1 & -1 & 1
\end{bmatrix}
\]

has the following poset of layers: see Figure 1

Another poset we are interested in is called the poset of faces.

**Definition 2.15.** A region is a connected component of $\mathcal{T} \setminus \bigcup_{H \in \mathcal{A}} H$, denoted by $R$.
We denote the number of regions of an arrangement $\mathcal{A}$ to be $r(\mathcal{A})$ and denote the collection of all regions in $\mathcal{A}$ to be $\mathcal{R}(\mathcal{A})$. 
Definition 2.16. A (closed) face of a real arrangement $A$ is a set $\emptyset \neq F = \overline{R} \cap x$, where $R \in \mathcal{R}(A)$ is a region of $A$ and $x \in L(A)$ is an element in poset of layers. A $k$-face is a $k$-dimensional face of $A$. An (open) face is just the interior of a closed face.

Definition 2.17. The poset of faces $\mathcal{F}$ collect all (open) faces in $A$, ordering by $F \leq G \iff F \subseteq \overline{G}$, where $\overline{G}$ is the closure of $G$.

Similarly, we do not include empty face as an element of our poset of faces.

Take the same example 2.14, we get the following poset of faces: see Figure 2.

This toric arrangement $C$ divide our n-torus space into 4 points, (the first level of our poset of faces with dimension 0 and we call them $0$-face), 12 line segments, excluding their end points (the second level of our poset of faces with dimension 1 and we call them $1$-face), and 8 triangles excluding their edges (the top level of our poset of faces with dimension 2 and we call them $2$-face).

2.4. Polynomials. Let $A$ be a toric arrangement with poset of layers $\mathcal{P}$ and poset of faces $\mathcal{F}$. There are some polynomials associated with our posets.

Definition 2.18. Define a function $\mu : \mathcal{P} \to \mathbb{Z}$, called the Möbius function of $\mathcal{P}$, by the condition:

1. $\mu(T) = 1$
2. $\sum_{x \leq y \leq T} \mu(y) = 0$, for all $x < T$
Note this is the same with the Möbius function of normal sense, instead we are applying the Möbius function to the dual of $\mathcal{P}$, $\mathcal{P}^{op}$.

**Definition 2.19.** The characteristic polynomial is defined as:

$$
\chi_A(t) = \sum_{x \in \mathcal{P}(A)} \mu(x)t^{\dim(x)}
$$

In example 2.14, we have its characteristic polynomial being $\chi_A(t) = t^2 - 6t + 8$.

**Definition 2.20.** The $f$-polynomial of $A$ is defined by:

$$
f(q) := \sum_{F \in \mathcal{F}} q^{\dim(F)}
$$

In example 2.14, we have its $f$ polynomial being $f(q) = 8q^2 + 12q + 4$.

**Definition 2.21.** The $h$-polynomial of $A$ is defined by:

$$
h(q) := (1 - q)^{\rho(F)} f\left(\frac{q}{1-q}\right)
$$

where $\rho(F)$ is the rank of top-dimentional faces, i.e. $\rho(F) = \max\{\dim(F) | F \in \mathcal{F}\}$

In example 2.14, we have its $f$ polynomial being $h(q) = (1 - q)^28\left(\frac{q}{1-q}\right)^2 + 12\left(\frac{q}{1-q}\right) + 4 = 4q+4$.

**Definition 2.22.** Let $n = \rho(F)$, and $[n] = 0, 1, ..., n$. Given subset $\emptyset \neq S = \{a_1, \cdots, s_k\} \subseteq [n]$, rank the elements in $S$ in the increasing order by $\mathcal{S} = \{s_1 < s_2 < \cdots < s_k\} \subseteq [n]$, let $f_S$ be the number of chains

$$
F_1 < F_2 < \cdots < F_k \text{ in } \mathcal{F}, \text{ such that } \text{rank}(F_i) = s_i.
$$
The reduced flag $f$-polynomial of $\mathcal{A}$ is
\[ \hat{f}(q_0, q_1, ..., q_n) = \sum_{\emptyset \neq S \subseteq [n]} f_S q^S \]
where $q^S = q_{s_1}q_{s_2}...q_{s_k}$

Note here our definition of flag $f$-polynomial is different from [ERS09] as well since we don't include empty chain of faces.

**Definition 2.23.** Define the reduced flag $h$-polynomial of $\mathcal{A}$ to be
\[ \hat{h}(q_0, q_1, ..., q_n) := (1 - q_0)...(1 - q_n)\hat{f}(\frac{q_0}{1 - q_0}, ..., \frac{q_n}{1 - q_n}). \]

### 3. Polynomials and Its Coefficients

Based on the definition of those polynomials associating with a toric arrangement, we can obtain some relations between the coefficients of different polynomials.

We can obtain the coefficients of $f$-polynomial be the restricted characteristic polynomial as following:

**Lemma 3.1.** The coefficient of $q^k$ in $f$-polynomial is given by:
\[ f_k = \text{number of } k\text{-faces} = \sum_{y \in \mathcal{P}, \dim(y) = k} \left| \chi_{\mathcal{P}^\geq_y}(0) \right| \]

**Proof.** Let $\mathcal{A}$ be an toric arrangement with poset of layers $\mathcal{P}$ and poset of faces $\mathcal{F}$.
Since each $k$ dimensional layer can only come from a $k$ dimensional layer. Therefore, to count the number of $k$-faces, we can sum over all layers of dimension $k$. For each $y \in \mathcal{P}$ with $\dim(y) = 0$, we count the number of regions of the restricted arrangement on $y$. By Theorem 4.9, the number of regions on $y$ created by the restricted arrangement is given by $\left| \chi_{\mathcal{P}^\geq_y}(0) \right|$. 

\[ \square \]

After we obtain the coefficients of $f$-polynomial, by the definition of $h$-polynomial, we can express the coefficients of $h$-polynomial in a clearer way.

**Lemma 3.2.** The coefficient of $q^k$ in $h$-polynomial with respect to the coefficient of $q^k$ in $f$-polynomial is given by:
\[ h_k = \sum_{i=0}^{k} \binom{n-i}{k-i} (-1)^{k-i} f_i \]

**Proof.** Recall from Definition 2.22,
\[ f(q) := \sum_{F \in \mathcal{F}} q^{\dim(F)} = \sum_{i=0}^{n} f_i q^i \]
where \( f_k \) = number of \( k \)-faces.
Recall \( \rho(F) = \max\{\dim(F) | F \in F\} = n, \)
\[
\begin{align*}
    h(q) & := (1 - q)^{\rho(F)} f\left(\frac{q}{1 - q}\right) \\
             & = (1 - q)^{n} \sum_{i=0}^{n} f_i (\frac{q}{1 - q})^i \\
             & = \sum_{i=0}^{n} f_i (1 - q)^{n-i} q^i
\end{align*}
\]
Therefore, we can obtain
\[
h_k = \sum_{i=0}^{n} f_i \binom{n-i}{k-i} (-1)^{k-i}
\]
□

Now we can express the coefficients of \( h \)-polynomials by the coefficients of \( f \)-polynomial, there is a nice relation between the coefficients of \( h \)-polynomials:

**Lemma 3.3.** The summation of the coefficients of \( h \)-polynomial is:
\[
h_0 + h_1 + \ldots + h_n = f_n
\]

**Proof.** From Lemma 4.7 and Lemma 3.5, we know that:
\[
h_n = (-1)^n f_0 + (-1)^{n-1} f_1 + \ldots - f_{n-1} + f_0 = 0
\]
From Lemma 3.5, we know that:
\[
\begin{align*}
    h_{n-1} & = (-1)^{n-1} n f_0 + (-1)^{n-2} (n-1) f_1 + \ldots + 3 f_{n-3} - 2 f_{n-2} + f_{n-1} \\
    h_{n-2} & = (-1)^{n-2} \left(\frac{n}{2}\right) f_0 + (-1)^{n-3} \left(\frac{n-1}{2}\right) f_1 + \ldots - 3 f_{n-3} + f_{n-2} \\
    & \ldots \\
    h_2 & = (-1)^2 \binom{n}{n-2} f_0 + (-1)^1 \binom{n-1}{n-2} f_1 + (-1)^0 \binom{n-2}{n-2} f_2 \\
    h_1 & = (-1)^1 \binom{n}{n-1} f_0 + (-1)^0 \binom{n-1}{n-1} f_1 \\
    h_0 & = (-1)^0 \binom{n}{n} f_0
\end{align*}
\]
Sum the above coefficients over, we get:
\[
\begin{align*}
    h_0 + h_1 + \ldots + h_n & = \sum_{i=0}^{n-1} (-1)^i \binom{n}{i} f_0 + \sum_{i=0}^{n-2} (-1)^i \binom{n-1}{i} f_1 + \ldots + \sum_{i=0}^{1} (-1)^i \binom{2}{i} f_{n-2} + f_{n-1} \\
    & = (-1)^n f_0 - (-1)^{n-1} f_1 - \ldots - (-1)^3 f_{n-3} - (-1)^2 f_{n-2} - (-1)^1 f_{n-1} \\
    & = (-1)^{n-1} f_0 + (-1)^{n-2} f_1 + \ldots + f_{n-3} - f_{n-2} + f_{n-1} \\
    & = f_n
\end{align*}
\]
To discover a easier way to obtain the coefficients in flag f-polynomial, we modified [ERS09, Theorem 3.13] using the restricted characteristic polynomials, and give a more intuitive proof for this formula. This formula contribute a lot in our program to generate flag f-polynomial.

**Lemma 3.4.** Assume that for every layer \( y \), \( \exists |\chi_{F^{\text{op}} \leq y}(-1)| \) regions of \( A \) incident to \( y \), that is the toric arrangement satisfies the Regular Cell Complex Assumption. Let \( S = \{ s_1 < s_2 < \ldots < s_k \} \subseteq [n] \), the coefficient of \( q_{s_1} q_{s_2} \ldots q_{s_k} \) is given by:

\[
\tilde{f}_S = \sum_{c \in \mathcal{P}} |\Pi_{i=1}^{k-1} \chi_{[y_i, y_{i+1}]}(-1)||\chi_{F^{\text{op}} \geq y_k}^{\text{op}}(0)|
\]

**Proof.** Let \( \Psi: \{\text{chains in } F^{\text{op}}\} \rightarrow \{\text{chains in } P^{\text{op}}\} \) be a surjective mapping \( (F_1 < F_2 < \ldots < F_k) \mapsto c = (y_1 < y_2 < \ldots < y_k) \), where \( y_i \) is the minimal element (i.e. a connected component of intersection) containing \( F_i \).

Note that \( \text{rank}(F_i) = \text{rank}(y_i) \).

In order to count number of chains generated by \( S \), we need to sum over number of chains in \( \Psi^{-1}(c) \).

Let \( c = (y_1 < y_2 < \ldots < y_k) \) be a chain in \( P^{\text{op}} \).

Recall from Lemma 3.1, there are \( |\chi_{P^{\text{op}} \geq y_k}^{\text{op}}(0)| \) choices for \( s_i \)-faces whose minimal covering element is \( y_k \).

Also recall from the Definition 2.10 that there is a bijection between toric arrangement and hyperplane arrangement.

From Richard P. Stanley Theorem 2.5, number of regions in a hyperplane arrangement \( A \) is given be \( r(A) = (-1)^n \chi_A(-1) = |\chi_A(-1)| \).

Then we iterate through \( y_i \):

For the subposet between interval \( [y_i, y_{i+1}] \), by our assumption, there will be \( |\chi_{[y_i, y_{i+1}]}(-1)| \) different hyperplanes incident to \( y_{i+1} \) (i.e. really is \( s_i \)-faces whose minimal covering element is \( y_{i+1} \)).

We multiply all the \( |\chi_{[y_i, y_{i+1}]}(-1)| \) with \( |\chi_{P^{\text{op}} \geq y_k}^{\text{op}}(0)| \) to get number of chains in \( F \) which corresponding to \( c \).

To get \( \tilde{f}_S \), we sum over all the possible chain \( c \) defined by \( S \). \( \square \)

We can also express the coefficients of flag h-polynomial by the coefficients of flag f-polynomial.

**Lemma 3.5.** The coefficient of \( q_S \) in reduced flag h-polynomial with respect to the coefficient of \( q_S \) in reduced flag f-polynomial is given by:

\[
\tilde{h}_S = \sum_{A \subseteq S} (-1)^{|S-A|} \tilde{f}_A
\]


Proof. Given \( \tilde{f}(q_1, \cdots, q_n) = \sum_{S \subseteq [n]} \tilde{f}_S q^S = \sum_{S \subseteq [n]} \tilde{f}_S \prod_{i \in S} q_i \) By the definition of flag h-polynomial, we have

\[
\tilde{h}(q_1, \cdots, q_n) = \prod_{i=0}^{n} (1 - q_i) \sum_{S \subseteq [n]} \tilde{f}_S (\prod_{j \in S} (1 - q_j))
\]

Therefore we can obtain

\[
\tilde{h}_S = \sum_{A \subseteq S} (-1)^{|S-A|} \tilde{f}_A
\]

□

Then there is also a nice relation between the coefficients in flag h-polynomial.

Lemma 3.6. The summation of the coefficients of reduced flag h-polynomial is:

\[
\sum_{S \subseteq [n]} \tilde{h}_S = \tilde{f}_S
\]

Proof. First, notice that \( \tilde{h}_i = \tilde{f}_i \), where \( i \in [n] \).

From Lemma 3.5, we know that:

\[
\tilde{h}_S = \sum_{A \subseteq S, |A|=|S|} \tilde{f}_A - \sum_{A \subseteq S, |A|=|S|-1} \tilde{f}_A + \ldots + \sum_{A \subseteq S, |A|=2} \tilde{f}_A + \sum_{A \subseteq S, |A|=1} \tilde{f}_A
\]

Therefore, \( \sum_{S \subseteq [n]} \tilde{h}_S = \tilde{f}_S \). □

4. Recurrences and Counting the Number of Regions

There is a well known theorem saying the number of regions in a toric arrangement can be obtained by taking the absolute value of its characteristic polynomial at 0. Here we will give two alternative proofs, one using the m"obius number and the other using the recurrence of regions and the recurrence of characteristic polynomials. Those proofs are taken as an analog of what [Sta07] did to count the number of regions in hyperplane arrangements.

We first want to show it’s proper to assume each hypertorus in the arrangement is primitive (connected) for the convenience of further proof.

Lemma 4.1. It’s proper to assume \( \forall \alpha \in T \) is primitive.

Proof. We just need to prove, \( \forall \alpha = (a_1, a_2, \cdots, a_n) \in T \), without loss of generality, we can assume \( \gcd(a_1, a_2, \cdots, a_n) = 1 \).

Let \( T = (S^1)^n \). Fix arbitrary \( \alpha : T \rightarrow S^1 \) with \( \alpha = (a_1, a_2, \cdots, a_n) \in \mathbb{Z} \).

Let \( d = \gcd(a_1, \cdots, a_n) \).

Let \( H_\alpha = \{ t \in T | t_1^{a_1} \cdots t_n^{a_n} = 1 \} \) (\( H_\alpha \) is the hypertorus association with \( \alpha \))

Since \( d = \gcd(a_1, \cdots, a_n) \), we have

\[
\alpha(t) = (t_1^{a_1/d} \cdots t_n^{a_n/d}) = 1 \iff t_1^{a_1/d} \cdots t_n^{a_n/d} = s_k
\]
where \( s_1, \ldots, s_d \) is \( d \)th root of unity. Note that \( \left( \frac{a_1}{d}, \cdots, \frac{a_d}{d} \right) \) is now primitive, otherwise, we can keep conducting this process until it become primitive. Recall from Definition 2.10, since we are working with the affine toric arrangement, we can write

\[
H_\alpha = \bigcup_{i=1}^{d} H_s \quad s_i
\]

Since each \( H_s \) is disjoint, we can work with them individually, therefore, WLOG, we can assume \( \forall \alpha \in \mathcal{T} \) is primitive.

**Lemma 4.2.** The poset of layers of \( A'' \) (the restriction of \( A \) on \( H_0 \)) is a subposet of the poset of layers of \( A \), i.e. \( \mathcal{P}(A'') \cong \{ x \in \mathcal{P}(A) | x \leq H_0 \} \).

**Proof.** We show two way containment.

Show \( \mathcal{P}(A'') \subseteq \{ x \in \mathcal{P}(A) | x \leq H_0 \} \):

\[
\forall y \in \mathcal{P}(A''), \exists H \in A \text{ with } H! = H_0 \text{ s.t. } y = \bigcap H, \text{ thus } y \subseteq H_0, \text{ also } y \in \mathcal{P}(A), \text{ Then } y \in \{ x \in \mathcal{P}(A) | x \leq H_0 \}.
\]

Show \( \{ x \in \mathcal{P}(A) | x \leq H_0 \} \subseteq \mathcal{P}(A'') \):

\[
\forall y \in \{ x \in \mathcal{P}(A) | x \leq H_0 \}, \exists H_1, \cdots, H_l \in A \text{ s.t. } y = \bigcap H_1 \cap \cdots \cap H_l. \text{ Therefore } y \in \mathcal{P}(A''). \]

In [Lemma 2.1] [Sta07], there is a recurrence for number of regions with respect to deletion and restriction in hyperplane arrangements. Here we make an analog to toric arrangements:

**Lemma 4.3.** Recurrence for regions of deletion and restriction:

\[
r(A) = \begin{cases} 
    r(A') + r(A'') & \text{if } r_k(A) = r_k(A') \\
    r(A'') & \text{if } r_k(A) > r_k(A')
\end{cases}
\]

**Proof.** Let \( A = \{ H_0, H_1, \cdots, H_l \} \) be a toric arrangement where each \( H_i \) is a connected affine hypertorus (if there is a hypertorus having multiple pieces of connected components, we can just separate them to be different hypertori). Also, WLOG, we can assume \( A \) is essential, otherwise, we can just essentialize \( A \).

Let \( A' = A - \{ H_0 \} = \{ H_1, \cdots, H_l \} \) be the deletion of \( H_0 \) in \( A \).

Let \( A'' = \{ \text{connected components of } H_i \cap H_0 \neq \emptyset | H_i \in A' \} \) be the restriction of \( A \) on \( H_0 \).

Let \( R(A), R(A'), R(A'') \) denote the regions in \( A, A', A'' \) respectively. Note that each region in \( A'' \) is created by intersect \( H_0 \) with regions in \( A' \), so the following bijection holds:

\[
R(A'') \leftrightarrow \bigcup_{R \in R(A')} \{ \text{nonempty connected components of } R \cap H_0 \}
\]

Also, for each region \( R \in R(A'), R \) is still a region in \( A \) if \( R \cap H_0 = \emptyset \), or \( R \) will be cut into some number of regions in \( A \) by \( H_0 \). So we have the following bijection:

\[
R(A) \leftrightarrow \bigcup_{R \in R(A')} \{ \text{nonempty connected components of } R - R \cap H_0 \}
\]

Case I: \( r_k(A) = r_k(A') \)

Since \( A \) is essential, \( r_k(A) = r_k(A') \), \( \forall R \in R(A'), R \cong \mathbb{R}^n \). Fix an arbitrary \( R \in R(A') \), if \( R \cap H_0 = \emptyset \), \( R \) contribute 0 to \( R(A'') \) and contribute 1 to \( R(A) \). If \( R \cap H_0 \neq \emptyset \), note since
$R \cong \mathbb{R}^{n-1}$, if $R \cap H_0$ has $k$ connected components, $R$ will be cut into $k+1$ pieces by $H_0$ in $A$, thus $R$ contribute 1 more to $R(A)$ than to $R(A'')$. Notice that $R$ always contribute 1 more in $R(A)$ than in $R(A'')$, which yields,

$$r(A) = r(A') + r(A'')$$

Case II: $rk(A) > rk(A')$

Since $A'$ is obtained by only removing $H_0$ from $A$, if we have $rk(A) > rk(A')$, it must be $rk(A) = rk(A') + 1$. So we will have $\forall R \in R(A'), R \cong \mathbb{R}^{n-1} \times S^1$. Also, since now all regions in $A'$ is isomorphic to $\mathbb{R}^{n-1} \times S^1$ but each region in $A$ is isomorphic to $\mathbb{R}^n$, every region in $A'$ need to be cut by $H_0$, meaning $\forall R \in R(A'), R \cap H_0 \neq \emptyset$. But also notice, but intersecting $R \in R(A')$, we are cutting $R \cong \mathbb{R}^{n-1} \times S^1$ into $R \cong \sqcup \mathbb{R}^n$, but the number of connected components in $R \cap H_0$ is the same as the number of connected components in $R - R \cap H_0$, which yields

$$r(A) = r(A'')$$

\[\square\]

**Theorem 4.4.** (The Cross-Cut Theorem) Let $L$ be a finite lattice. Let $X$ be a subset of $L$ such that $\hat{0} /\in X$, and such that if $y \in L$, $y \neq \hat{0}$, then some $x \in X$ satisfies $x \leq y$. Let $N_k$ be the number of $k$-elements subsets of $X$ with join $\hat{1}$. Then

$$\mu_L(\hat{0}, \hat{1}) = N_0 - N_1 + N_2 - \cdots$$

In [Sta07], the recurrence of characteristic polynomial with respect to deletion and restriction for hyperplane arrangements used a Whitney theorem [Theorem 2.4] [Sta07]. Here we will begin the proof of the recurrence of characteristic polynomial by proving an analog of Whitney’s theorem in toric arrangements.

**Theorem 4.5.** (Toric Arrangement analog of Whitney’s Theorem) Let $A$ be an arrangement in a $n$-dimensional vector space. Let $B \subset A$ be a central sub-arrangement of $A$, then denote the number of connected components of the intersection of $B$ by $m(B)$, that is

$$m(B) = \# \{\text{connected components of } \bigcap_{H \in B} H\}.$$  

Then,

$$\chi_A(t) = \sum_{B \subseteq A} (-1)^{\#B} m(B)t^{n-rank(B)}.$$  

**Proof.** Let $z \in \mathcal{P}$. Let $A_z = \{H \in A : H \leq z \text{(i.e., } z \subseteq H)\}$. Let $[\hat{0}, z]$ denote the subposet below $z$ in $\mathcal{P}^{op}$. This interval is then a finite lattice since it has meet $\hat{0}$ and join $z$. Note that for $B \subseteq A_z$, $\bigvee_{H \in B} H = z$ (the join of all hypertori in $B$ is $z$) in $[\hat{0}, z]$ if and only if $z$ is a connected components of $\bigcap_{H \in B} H$. Apply Theorem 4.4 to $[\hat{0}, z]$, we have

$$\mu(z) = \sum_k (-1)^k N_k(z)$$
where $N_k(z)$ is the number of $k$-subsets of $A_z$ with join $z$ in $[\hat{0}, z]$. In other words,

$$
\mu(z) = \sum_{B \subseteq A_z, z = \bigvee_{H \in B} H} (-1)^{\#B}
$$

Note that $z = \bigvee_{H \in B} H$ in $[\hat{0}, z]$, meaning $z \in \bigcap_{H \in B} H$, implies that $\text{rank}(B) = n - \text{dim}(z)$, then we can multiply both sided by $t^{\text{dim}(z)}$, and obtain

$$
\mu(z)t^{\text{dim}(z)} = \sum_{B \subseteq A_z, z = \bigvee_{H \in B} H} (-1)^{\#B} t^{n - \text{rank}(B)}
$$

Recall the definition for characteristic polynomial is

$$
\chi_A(t) = \sum_{x \in P} \mu(x)t^{\text{dim}(x)}
$$

Since each element in the poset of layers is formed by the intersection of some hypertori, we can construct the characteristic polynomial by summing up over all sub-arrangement $B$ of $A$ and it’s not necessarily to be central since $m(B) = 0$ if the intersection do not exist. But notice that since $\bigcap_{H \in B} H$ can have multiple connected components, it will contribute $m(B)(-1)^{\#B} t^{n - \text{rank}(B)}$ to the characteristic polynomial, which yields,

$$
\chi_A(t) = \sum_{B \subseteq A} (-1)^{\#B} m(B)t^{n - \text{rank}(B)}
$$

Then really the recurrence for characteristic polynomial of hyperplane arrangement [Lemma 2.2][Sta07] still holds for toric arrangements.

**Lemma 4.6.** Let $A$ be a toric arrangement. For arbitrary $H_0 \in A$, let $A' = A - H_0$ be the deletion of $H_0$ on $A$. Let $A'' = \{\text{the connected components of } H \cap H_0 | H \in A' \}$ be the restriction of $A$ on $H_0$. Then we have the recurrence for the characteristic polynomial on the deletion and restriction to be

$$
\chi_A(t) = \chi_{A'}(t) - \chi_{A''}(t)
$$

**Proof.** Note by Theorem 4.5,

$$
\chi_A(t) = \sum_{B \subseteq A} (-1)^{\#B} m(B)t^{n - \text{rank}(B)}
$$

$$
= \sum_{H_0 \notin B \subseteq A} (-1)^{\#B} m(B)t^{n - \text{rank}(B)} + \sum_{H_0 \in B \subseteq A} (-1)^{\#B} m(B)t^{n - \text{rank}(B)}
$$

Here for the formula for $\chi_A(t)$, we split the sum at the right hand to be two sums depending on if the central sub-arrangement includes $H_0$ or not. The for the first part we have

$$
\sum_{H_0 \notin B \subseteq A} (-1)^{\#B} m(B)t^{n - \text{rank}(B)} = \chi_{A'}(t)
$$
For the second part, let $B'' = (B - \{H_0\})^{H_0}$, which is really the restriction of a central sub-arrangement on $H_0 \cong (s^1)^{n-1}$. Since $\#B'' = \#B - 1$ and $\text{rank}(B'') = \text{rank}(B) - 1$, $m(B'') = m(B)$ we have the second part of summation to be

$$
\sum_{H_0 \in B \subseteq A} (-1)^{\#B} m(B) t^{n-\text{rank}(B)} = \sum_{B'' \subseteq A''} (-1)^{\#B''} m(B'') t^{n-\text{rank}(B'')} = -\chi_{A''}(t)
$$

Then we will have

$$
\chi_A(t) = \sum_{H_0 \in B \subseteq A} (-1)^{\#B} m(B) t^{n-\text{rank}(B)} + \sum_{H_0 \in B \subseteq A} (-1)^{\#B} m(B) t^{n-\text{rank}(B)} = \chi_{A'}(t) - \chi_{A''}(t)
$$

Lemma 4.7. For $\Delta$ being a torus, its Euler characteristic $\psi(\Delta) = f_0 - f_1 + f_2 - \cdots = 0$; Also, $\psi(\mathbb{R}^n) = f_0 - f_1 + f_2 - \cdots = (-1)^n$

Lemma 4.8. (Möbius Inversion) [Theorem 1.1] [Sta07] Let $P$ be a finite poset with Möbius function $\mu$, and let $f, g : P \to K$ ($K$ is a field). Then the following two conditions are equivalent:

$$f(x) = \sum_{y \geq x} g(y), \text{ for all } x \in P$$

$$g(x) = \sum_{y \geq x} \mu(x, y) f(y), \text{ for all } x \in P$$

Theorem 4.9. The number of regions in the complement to an essential toric arrangement $A$ is given by $r(A) = (-1)^{\rho(A)} \chi_A(0)$.

Proof, version 1. Now by Lemma 4.3 and Lemma 4.6, we have

$$r(A) = \begin{cases} 
 r(A') + r(A'') & \text{if } r(k(A)) = r(k(A')) \\
 r(A'') & \text{if } r(k(A)) > r(k(A'))
\end{cases}$$

and

$$\chi_A(t) = \chi_{A'}(t) - \chi_{A''}(t)$$

To show $r(A) = (-1)^{\rho(A)} \chi_A(0)$, we just need to show this expression satisfies the recurrence in Lemma 4.3.

Base case: Empty arrangement.
Since we only consider the essential arrangement, the empty arrangement can only exist when the ambient vector space is of dimension 0. Then $r(\emptyset) = 1$, $\chi_\emptyset = t^0 = 1$ by convention, satisfying $r(\emptyset) = |\chi_\emptyset(0)|$.
Now we prove that $r(A) = (-1)^{\rho(A)} \chi_A(0)$ satisfy the recurrence for number of regions. Case 1:
\( \text{rk}(A) = \text{rk}(A') \) Then we want to show \( r(A) = (-1)^{\text{rk}(A)} \chi_A(0) \) satisfies \( r(A) = r(A') + r(A'') \), which is just

\[
(-1)^n \chi_A(0) = (-1)^n \chi_{A'}(0) + (-1)^{n-1} \chi_A(0)
\]

\[\Leftrightarrow \chi_A(0) = \chi_{A'}(0) - \chi_{A''}(0)\]

But then this equation holds because of Lemma 4.6.

Case 2: \( \text{rk}(A) = \text{rk}(A') + 1 \) Then we want to show \( r(A) = (-1)^{\text{rk}(A)} \chi_A(0) \) satisfies \( r(A) = r(A'') \), which is just

\[
(-1)^n \chi_A(0) = (-1)^{n-1} \chi_A(0)
\]

\[\Leftrightarrow \chi_A(0) = -\chi_{A''}(0)\]

By Lemma 4.6 we have \( \chi_A(t) = \chi_{A'}(t) - \chi_{A''}(t) \), but notice since \( \text{rk}(A') = \text{rk}(A) - 1 \), \( A' \) is not essential, thus it doesn't have constant term in its characteristic polynomial, so \( \chi_{A'}(0) = 0 \). Therefore \( \chi_A(0) = -\chi_{A''}(0) \) holds.

\[\square\]

Proof, version 2. Let \( T = (S^1)^n \) and let \( A \) be an toric arrangement on \( T \). Let \( P \) be the poset of layers and let \( P^{op} \) by the poset of layers ordered by reverse inclusion.

Recall the definition, \( \chi_A(q) = \sum_{Y \in P} \mu_{P^{op}}(T, Y) q^{\text{dim}(Y)} \), we have

\[
\chi_A(0) = \sum_{x \in P} \mu_{P^{op}}(T, x)
\]

Let \( f_k(A) \) denote the number of \( k \)-faces of \( A \), it follows that

\[
\psi(T) = f_0(A) - f_1(A) + f_2(A) - \cdots
\]

Every \( k \)-face is exactly one region of \( A^y \) for some \( y \in P(A) \) with \( \text{dim}(y) = k \), so we have

\[
f_k(A) = \sum_{y \in P(A)} r(A^y)_{\text{dim}(y) = k}
\]

We can multiply both side of the equation by \( (-1)^k \) thus obtain

\[
(-1)^k f_k(A) = \sum_{y \in P(A)} (-1)^k r(A^y)_{\text{dim}(y) = k}
\]

Sum over \( k \) to get

\[
\psi(T) = \sum_{k=0}^{n} (-1)^k f_k(A) = \sum_{k=0}^{n} \sum_{y \in P(A)} (-1)^k r(A^y)_{\text{dim}(y) = k} = \sum_{x \in P(A)} (-1)^{\text{dim}(x)} r(A^x)
\]

By Lemma 4.2, we can restrict the arrangement on \( y \), thus can replace \( T \) by \( y \) in this equation,
Möbius Inversion (Lemma 4.8) yields

\[-1]^{\dim(y)} r(\mathcal{A}^y) = \sum_{x \in \mathcal{P} \atop x \geq y} \mu_{\mathcal{P}}^\text{op}(y, x) \psi(x)

Note that, for \( x \in \mathcal{P}(\mathcal{A}) \) with \( \dim(x) > 0 \), \( x \) is a torus, so by Lemma 4.7, \( \psi(x) = 0 \). But for \( x \in \mathcal{P}(\mathcal{A}) \) with \( \dim(x) = 0 \), \( \psi(x) = \psi(\mathbb{R}^0) = (-1)^0 = 1 \).

Note that we know \( \{ x \in \mathcal{P} \mid \dim(x) = 0 \} \neq \emptyset \) since now we only consider the essential arrangements. Thus we can obtain

\[ \sum_{x \in \mathcal{P} \atop x \geq y} \mu_{\mathcal{P}}^\text{op}(y, x) \psi(x) = \sum_{x \in \mathcal{P} \atop \dim(x) = 0} \mu_{\mathcal{P}}^\text{op}(y, x) \psi(x) \]

Note that here \( y \) is any layer in \( \mathcal{A} \), so we can replace \( y \) by \( T \), which yields,

\[ (-1)^n r(\mathcal{A}) = \sum_{x \in \mathcal{P} \atop \dim(x) = 0} \mu_{\mathcal{P}}^\text{op}(T, x) = \chi(0) \]

Therefore we can obtain

\[ r(\mathcal{A}) = (-1)^{\rho(\mathcal{A})} \chi(0) \]

\[ \square \]

5. Symmetry of reduced flag h-polynomial

All lemmas in this section will contribute to prove a nice symmetry of coefficients in flag h-polynomial, that is \( \tilde{h}_S = \tilde{h}_{[n] - S} \). In order to prove this symmetry, we will use the ab-index and cd-index form of flag h-polynomial:

**Definition 5.1.** Let \( \mathcal{A} \) be a toric arrangement, and let \( \mathcal{P} \) be the poset of layers of \( \mathcal{A} \). Then given a nonempty subset \( S \subset [n] \), let \( u_S = u_0 u_1 \cdots u_n \) be the \((n + 1)\)-letter word defined by

\[ u_i = \begin{cases} b & \text{if } i \in S \\ a & \text{if } i \notin S \end{cases} \]

The ab-index of \( \mathcal{P} \) is defined by

\[ \Psi(\mathcal{P}) = \sum_{S \subset [n]} h_S u_S \]

**Definition 5.2.** Let \( \Psi \) be the ab-index of a toric arrangement \( \mathcal{A} \), then the cd-index of \( \mathcal{A} \) is the expression \( \Psi \) written using the variable \( c = a + b \) and \( d = ab + ba \).

We mentioned that in [ERS09], they includes the empty face and empty layer when construct the poset of layers and poset of faces of a toric arrangement. And they have a theorem saying the ab-index form of flag h-polynomial can be expressed in a homogeneous cd-polynomial of degree \( n + 1 \) plus \((a - b)^{n+1}\).
Lemma 5.3. [ERS09, Corollary 2.12] Let $\Omega$ be a regular cell complex whose geometric realization is the $n$-dimensional torus $T^n$. Then the ab-index of the face poset $P$ of $\Omega$ has the following form:

$$\Psi(P) = (a - b)^{n+1} + \Phi$$

where $\Phi$ is a homogeneous cd-polynomial of degree $n + 1$ and $\Phi$ does not contain the term $c^{n+1}$.

But in this paper, we construct the poset of layers and poset of faces by excluding the empty face, and then we can modify Lemma 5.3 to show the flag $h$-polynomial in this case is cd-indexable.

Lemma 5.4. Let $P$ be the poset of faces excluding the empty face, then the ab-index of $P$ can be expressed in the following form:

$$\Psi(P) = \Phi$$

where $\Phi$ is a homogeneous cd-polynomial of degree $n + 1$ and $\Phi$ does not contain the term $c^{n+1}$ i.e. the cd-index form of $P$ exists.

Proof. Let $P'$ be the poset of faces including the empty face. Let $\tilde{h}'_S$ denote the coefficient of $q_S$ in the flag $h$-polynomial of $P'$. By Lemma 3.5, we have

$$\tilde{h}_S = \sum_{A \subseteq S \setminus \emptyset} (-1)^{|S - A|} \tilde{f}_A$$

Then we have

$$\tilde{h}'_S = \tilde{f}_\emptyset (-1)^{|S|} + \sum_{A \subseteq S \setminus \emptyset} (-1)^{|S - A|} \tilde{f}_A$$

$$= (-1)^{|S|} + \tilde{h}_S$$

Which is also

$$\tilde{h}_S = \tilde{h}'_S + (-1)^{|S|+1}$$

Given $S \subset [n]$, let $u_S$ be the ab-index of $q_S$, we have

$$\Psi(P) = \sum_{S \subset [n]} \tilde{h}_S u_S$$

Also, by [ERS09, Corollary 2.12],

$$\Psi(P') = \sum_{S \subset [n]} \tilde{h}'_S u_S = (a - b)^{n+1} + \Phi$$
Since we showed $\tilde{h}_S = \tilde{h}'_S + (-1)^{|S|+1}$, we can express $\Psi(P)$ in terms of $\Psi(P')$:

$$\Psi(P) = \sum_{S \subseteq \{n\}} \tilde{h}_S u_S$$

$$= \sum_{S \subseteq \{n\}} (\tilde{h}'_S + (-1)^{|S|+1}) u_S$$

$$= \sum_{S \subseteq \{n\}} \tilde{h}'_S u_S + \sum_{S \subseteq \{n\}} (-1)^{|S|+1} u_s$$

$$= \Psi(P') + \sum_{S \subseteq \{n\}} (-1)^{|S|+1} u_s$$

$$= \Phi + (a - b)^{n+1} + \sum_{S \subseteq \{n\}} (-1)^{|S|+1} u_s$$

Now it's enough to show $(a - b)^{n+1} + \sum_{S \subseteq \{n\}} (-1)^{|S|+1} u_s$, that is

$$\sum_{S \subseteq \{n\}} (-1)^{|S|} u_s = (a - b)^{n+1}$$

Recall our ab-indexing, we have $u_S = u_1 u_2 \cdots u_n$ where

$$u_i = \begin{cases} b & \text{if } i \in S \\ a & \text{if } i \notin S \end{cases}$$

and

$$(a + b)^{n+1} = \sum_{S \subseteq \{n\}} u_S$$

Now modify $u_s$ to be $u'_S = u'_1 u'_2 \cdots u'_n$ as following:

$$u_i = \begin{cases} -b & \text{if } i \in S \\ a & \text{if } i \notin S \end{cases}$$

which yields

$$(a - b)^{n+1} = \sum_{S \subseteq \{n\}} u'_S$$

Also,

$$u'_S = (-1)^{|S|} u_s$$

Therefore we have

$$\sum_{S \subseteq \{n\}} (-1)^{|S|} u_s = \sum_{S \subseteq \{n\}} u'_S = (a - b)^{n+1}$$

After showing the cd-index form of $\Phi(P)$ exists, the following algorithm can give a way to calculate the cd-index of a cd-indexable poset directly from its reduced flag h-polynomial. This poset is not necessarily to come from a toric arrangement, we just requires it to have a cd-indexable reduced flag h-polynomial.

The following two lemmas serve to prove algorithm 1.
Algorithm 1: Algorithm to obtain cd-index from flag h polynomial

1. Collect all cd-monomials of degree n+1 in a list cd-list. Set cd and sort this list by increasing dictionary order. Construct an empty dictionary taking those monomials as keys and their coefficients being the value for future use, denote this dictionary as cd-dict.

2. Set the coefficient of those cd-monomials with one 'd' in cd-dict as base case. Label the digits from 0 to n-1, for cd-monomial with one 'd' at the kth digit, by Lemma 5.5, we can set its coefficient to be \( \sum_{i=0}^{k} (-1)^{k-i} \tilde{h}_i \), where \( \tilde{h}_i \) is the coefficient for \( q_i \) in flag h-polynomial.

3. Loop through integer b in range \([1, 2^{n+1} - 2]\) for step 4-6.

4. For integer b in its binary form, construct its corresponding ab-monomial \( u_S \) in its ab-index form by

\[
\begin{cases} 
  u_i = b, & \text{if } b[i] = 1 \\
  u_i = a, & \text{if } b[i] = 0 
\end{cases}
\]

and denote its coefficient by \( \tilde{h}_S \).

5. Let b-list be the list of all cd-monomials with \( u_S \) being a term in its expansion form.

6. By Lemma 5.6, there is at most one cd-monomial in b-list whose coefficient is not in cd-dict yet and we can fill in that coefficient by solving

\[
\tilde{h}_S = \sum_{k \in \text{b-list}} \text{cd-dict}[k].
\]

7. When the loop ends, we can obtain the coefficients for all cd monomials.

Lemma 5.5. Let \( A \) be a toric arrangement over \( T = (S^1)^n \). Let its flag h-polynomial be \( \sum_{S \subseteq [n]} \tilde{h}_S q_S \) and its ab-index form be \( \sum_{S \subseteq [n]} \tilde{h}_S u_S \). For cd-monomial of degree n+1 with one 'd' at the kth digit, its coefficient in the cd-index of \( A \) is \( \sum_{i=0}^{k} (-1)^{k-i} \tilde{h}_i \), where \( \tilde{h}_i \) is the coefficient for \( q_i \) in flag h-polynomial.

Proof. We will prove this lemma by induction. Let \( s \) be a cd-monomial with n digit and n+1 degree, meaning \( s \) only has one d in its expression. We want to show if \( s[k] = d \), then the coefficient for \( s \) in the cd-index of \( A \) is \( \sum_{i=0}^{k} (-1)^{k-i} \tilde{h}_i \). Let \( s_k \) denote such cd-monomial with only \( s_k[k] = d \). Let \( \text{coef } f(s_k) \) denote the coefficient for \( s_k \).

Base Case: show \( \text{coef } f(s_0) = \tilde{h}_0 \)

Since we don’t have the pure c monomial in the cd-index of \( A \), the only cd-monomial that can include \( u_{(0)} \) as a term in its expansion form is \( s_0 \). So the coefficient for \( s \) and \( u_{(0)} \) must match. Therefore the coefficient for \( s_0 \) is \( h_0 \).

Now suppose \( \text{coef } f(s_k) = \sum_{i=0}^{k} (-1)^{k-i} \tilde{h}_i \). Then we want to show \( \text{coef } f(s_{k+1}) = \sum_{i=0}^{k+1} (-1)^{k-i} \tilde{h}_i \).

Consider the coefficient \( h_{k+1} \) for \( u_{(k+1)} \). There are only two cd-monomials can take \( u_{(k+1)} \) as a
term in their expansion form, which are \(s_k\) and \(s_{k+1}\). Then we have

\[
\begin{align*}
\text{coeff}(s_k) + \text{coeff}(s_{k+1}) &= \tilde{h}_{k+1} \\
\text{coeff}(s_k) &= \sum_{i=0}^{k} (-1)^{k-i} \tilde{h}_i
\end{align*}
\]

Therefore we can solve for \(\text{coeff}(s_{k+1})\) to be

\[
\text{coeff}(s_{k+1}) = \sum_{i=0}^{k+1} (-1)^{k+1-i} \tilde{h}_i
\]

\[\square\]

Lemma 5.6. Let \(A\) be a toric arrangement over \(T = (S^1)^n\). Let its flag h-polynomial be \(\sum S \subseteq [n] \tilde{h}_S q_S\) and its ab-index form be \(\sum S \subseteq [n] \tilde{h}_S u_S\). Let \(M = \{\text{cd-monomials with degree } n+1\}\), set order \(c < d\), and rank the elements in \(M\) in increasing order with labels. In Algorithm 1, if \(i < j\), then at the time for us to compute \(\text{coeff}(M[j])\) in step 6, we will already have \(\text{coeff}(M[i])\) known.

Proof. For the convenience of indexing and comparing, we substitute \(\tilde{d}'\) in each cd-monomial by \(\tilde{d}_0'\) to represent that \(\tilde{d}'\) has degree 2. Also, in this indexing, all cd-monomial will have length \(n+1\).

We will prove this lemma by contradiction. Suppose \(\exists i < j\) with \(M[i] < M[j]\) in the dictionary order. Let \(t = \min\{t \in [n]|M[i][t] \neq M[j][t]\}\). Then since \(M[i] < M[j]\), we need \(M[i][t] = 'c'\) and \(M[j][t] = 'd'\).

Construct two ab-monomial as following:

\[
\begin{align*}
\text{if } M[i][k] = 0(\text{i.e. } M[i][k-1] = d) \quad &\Rightarrow b, \\
\text{otherwise} \quad &\Rightarrow a,
\end{align*}
\]

Then construct two binary strings as following:

\[
\begin{align*}
b_i[k] &= \begin{cases} 
1, & \text{if } u_i[k] = b \\
0, & \text{if } u_i[k] = a 
\end{cases} \\
b_j[k] &= \begin{cases} 
1, & \text{if } u_j[k] = b \\
0, & \text{if } u_j[k] = a 
\end{cases}
\end{align*}
\]

Then following the algorithm, the first appear for \(M[i]\) is \(b_i - \text{list}\), and the first appear for \(M[j]\) is \(b_j - \text{list}\). But since \(t = \min\{t \in [n]|M[i][t] \neq M[j][t]\}\), \(M[i][t] = 'c'\) and \(M[j][t] = 'd'\), we have \(b_i < b_j\). So we will obtain the coefficient for \(M[i]\) first, a contradiction. \[\square\]

Now we can begin to prove the symmetricity of the coefficients in flag h-polynomial.
**Theorem 5.7.** For $A$ being a toric arrangement on $T = (S^1)^n$, let $\tilde{h}(q_1 q_2 \cdots q_n) = \sum_{S \subset [n]} \tilde{h}_S q_S$ be the flag $h$-polynomial associating to $A$, where $q_S = \prod_{i \in S} q_i$. Then the coefficient of $\tilde{h}$ is symmetric, i.e. $\tilde{h}_S = \tilde{h}_{[n]\setminus S}$

**Proof.** Let $P$ be the poset of faces associating to $A$. Then by Lemma 5.4, we have $\Psi(P) = \Phi$ where $\Psi(P)$ is the ab-indexing of $P$ and $\Phi$ is the cd-indexing of $P$.

Note that for the ab-indexing of $P$, we have $\Psi(P) = \sum_{S \subset [n]} \tilde{h}_S u_S$. To show $\tilde{h}_S = \tilde{h}_{[n]\setminus S}$, we just need to show $u_S$ and $u_{[n]\setminus S}$ have the same coefficient.

Recall our ab-indexing, we have $u_S = u_1 u_2 \cdots u_n$ where

$$u_i = \begin{cases} b & \text{if } i \in S \\ a & \text{if } i \notin S \end{cases}$$

But then notice that for $u_{[n]\setminus S} = u'_1 u'_2 \cdots u'_n$, we have

$$u'_j = \begin{cases} b & \text{if } j \in S \\ a & \text{if } j \notin S \end{cases}$$

Then we have $u_i = a \iff u'_i = b$

Recall for the cd-indexing rule, we have

$$\begin{aligned} c &= a + b \\
 d &= ab + ba \end{aligned}$$

So $a$ and $b$ are symmetric in $c$, $d$ respectively. But then since the cd-index form of $\tilde{h}$ exists, $a$ and $b$ are symmetric in $\Psi(P)$, i.e. $\Psi(P)(a, b) = \Psi(P)(b, a)$

Since $u_i = a \iff u'_i = b$, we have

$$\begin{aligned} \Psi(P)(a, b) = \sum_{S \subset [n]} h_S u_S \\
 \Psi(P)(b, a) = \sum_{S \subset [n]} h_S u'_S \end{aligned}$$

Which yields $\tilde{h}_S = \tilde{h}_{[n]\setminus S}$\qed

6. **Special Case: Coordinate Toric Arrangement**

We will study a special case called coordinate toric arrangement, which is a direct analog from the coordinate arrangement in hyperplane arrangement.
Definition 6.1. A coordinate toric arrangement $A(n, k)$ is an essential central toric arrangement in $T = (S^1)^n$ with the associated matrix to be the following form:

$$
\begin{bmatrix}
  k & 0 & 0 & \ldots & 0 & 1 \\
  0 & k & 0 & \ldots & 0 & 1 \\
  0 & 0 & k & \ldots & 0 & 1 \\
  \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
  0 & 0 & 0 & \ldots & k & 1 \\
\end{bmatrix} \in \text{Mat}(n \times (n+1)), k \geq 2
$$

Note that this matrix has all elements in the last column to be one, and is a diagonal matrix with the diagonal elements all be $k$ after removing the last column.

Note that since we only study the toric arrangement satisfying the regular cell complex for now. We need $k \geq 2$ to make sure the coordinate toric arrangement is under regular cell complex assumption. Also, each hypertori in this arrangement has $k$ connected component parallel with coordinates (with one piece being the coordinates) and any two connected component from different hypertori are perpendicular. Note that all hypertori are cyclic and all connected component in this arrangement are cyclic, which is very important for the following propositions.

Proposition 6.2. Let $\mathcal{P}(n, k)$ denote the poset of layers of a coordinate toric arrangement $A(n, k)$. Then the $i$th level (the level with dimension $n - i$) has $\binom{n}{i} k^i$ elements and the Möbius number for all elements with dimension $n - i$ is $\left(-\frac{1}{2}\right)^i$, i.e. $\forall x \in \mathcal{P}$ with $\text{dim}(x) = n - i$, we have $\mu(x) = \left(-\frac{1}{2}\right)^i$.

Proof. To count the number of elements in the $i$th level, we just need to count the number of ways to form such $I_i$.

We first show the $i$th level (the level with dimension $n - i$) has $\binom{n}{i} k^i$ elements. Each element in this level is of dimension $n - i$, which must be formed by intersecting $i$ connected components of hypertori. Let $x_i \in \mathcal{P}$ with $\text{dim}(x_i) = n - i, x = \bigcap_{j \in I_i} H_j$, where $I_i$ is a index set with $|I_i| = i$.

Note that the connected components come from the same hypertori can’t intersect, so to choose $i$ connected component to intersect, they must come from different hypertori, and there are $\binom{n}{i}$ way to do so. After we’ve chosen $i$ hypertori to intersect, for each hypertori, there are $k$ connected component, so there are $k^i$ way in total. Therefore there are $\binom{n}{i} k^i$ elements in the $i$th level of $\mathcal{P}$.

Now we show $\forall x \in \mathcal{P}$ with $\text{dim}(x) = n - i$, we have $\mu(x) = \left(-\frac{1}{2}\right)^i$. Since all elements in each level are cyclic, let $\mu_i$ denote the Möbius number for each element in the $i$th level. Then by definition, first of all, we have $\mu_0 = \mu(T) = 1$. Fix arbitrary $x_i \in \mathcal{P}$ with $\text{dim}(x_i) = n - i$. Let $\mathcal{P}_i$ denote the subposet above $x_i$. Then we have $\sum_{y \in \mathcal{P}_i} \mu(y) = 0$. Now consider each level in $\mathcal{P}_i$. $T$ is the top element for $\mathcal{P}_i$, and $\{H_j|j \in I_i\}$ forms the first level. $x_i$ is the intersection of $i$ independent hypertori, that is $\bigcap_{j \in I_i} H_j$. Then any intersection of some of those hypertori will include $x_i$, thus is a vertex in $\mathcal{P}_i$. And any element includes in $\mathcal{P}_i$ is an intersection of some $H_j, j \in I_i$. Therefore, $\forall y \in \mathcal{P}_i$ with $\text{dim}(y) = n - j$, its subposet in $\mathcal{P}_i$ is the same as its subposet.
in $\mathcal{P}$, thus $y$ has the same mobius number $\mu_j$ in both posets. In $\mathcal{P}_1$, there are $\binom{i}{j}$ elements with dimension $n - j$, i.e. with mobius number $\mu_j$, so we have the recurrence

$$
\begin{cases}
\mu_0 = 1 \\
\sum_{j=0}^{i} \binom{i}{j} \mu_j = 0
\end{cases}
$$

Let $\mu_j = (-1)^j$, then it satisfies the first condition obviously. For the second condition, we have

$$
LHS = \sum_{j=0}^{i} \binom{i}{j} (-1)^j = (1 - 1)^i = 0
$$

□

Proposition 6.3. Let $\mathcal{A}(n,k)$ be a coordinate toric arrangement. Then the characteristic polynomial of $\mathcal{A}(n,k)$ is $\chi(t) = (t - k)^n$.

Proof. Let $\mathcal{P}$ be the poset of layers of $\mathcal{A}(n,k)$, then follow from Proposition 6.2, the $i$th level of $\mathcal{P}$ has $\binom{n}{i}k^i$ element, with dimension $n - i$, has mobius number $(-1)^i$. Therefore we have

$$
\chi(t) = \sum_{x \in \mathcal{P}} \mu(x) t^{\dim(x)}
= \sum_{i=0}^{n} (-1)^i \binom{n}{i} k^i t^{n-i}
= \sum_{i=0}^{n} \binom{n}{i} (-k)^i t^{n-i}
= (t - k)^n
$$

□

Proposition 6.4. Let $\mathcal{A}(n,k)$ be a coordinate toric arrangement. Then the $f$ polynomial of $\mathcal{A}(n,k)$ is $f = k^n(t + 1)^n$.

Proof. Let $\mathcal{P}$ be the poset of layers of $\mathcal{A}(n,k)$, then follow from Proposition 6.2, the $i$th level of $\mathcal{P}$ has $\binom{n}{i}k^i$ element, with dimension $n - i$. Let $\mathcal{F}$ be the poset of faces of $\mathcal{A}$. Now we just need to know the number of connected component of each layer. Take an arbitrary layer $y \in \mathcal{P}$ with $\dim(y) = n - i$. Label the hypertori in $\mathcal{A}$ to be $H_1, H_2, \cdots , H_n$ with each $H_i$ has $k$ connected component $H_i, j (j \in [k])$. WLOG, let $y$ be a connected component of the intersection of the first $i$ hypertori, that is let $y \in \bigcap_{l=1}^{i} H_l$.

Consider $\mathcal{A}^y = \{y \cap H \neq \emptyset | y \not\subseteq H, H \in \mathcal{A}\}$. We take the subposet of $\mathcal{P}$ below $y$ to be $\mathcal{P}_y$. To know the number of regions $y$ have, we want to know the characteristic polynomial $\chi_y(t)$ of $\mathcal{P}_y$. Notice that $\forall H_i \in \mathcal{A}$, if $i \leq l \leq n$, then $H_l \cap y \neq \emptyset$ and $H_l \cap y$ has $k$ connected component, since $H_l$ has $k$ connected component and all of them are perpendicular to $y$. Then consider the $l$th level of $\mathcal{P}_l$, it has $\binom{n-i}{l}k^l$ elements, with mobius number $(-1)^{n-i-l}$. Therefore the characteristic polynomial of $\mathcal{P}_y$ is $\chi_y(t) = \sum_{l=0}^{n-i} \binom{n-i}{l} k^l (-1)^{n-i-l}$.
polynomial of
\[ \chi_y(t) = \sum_{i=0}^{n-i} (-1)^{n-i-l} \binom{n-i}{l} k^l t^{n-i-l} \]

Then by Theorem 4.9,
\[ r(A^y) = |\chi_y(0)| = \binom{n-i}{n-i} k^{n-i} = k^{n-i} \]

Now consider \( f, i \in 0, 1, \ldots, n \). There are \( \binom{n}{i} k^i \) layers in \( \mathcal{P} \) with dimension \( i \), and each layer of dimension \( i \) has \( k^{n-i} \) connected components. Therefore, we have the \( f \) polynomial to be:
\[ f(t) = \sum_{y \in \mathcal{F}} t^{\dim(y)} = \sum_{i=0}^{n} \binom{n}{i} k^i t^{n-i} = k^n (t + 1)^n \]

**Proposition 6.5.** Let \( A(n, k) \) be a coordinate toric arrangement. Then the \( h \) polynomial of \( A(n, k) \) is \( h = k^n \).

**Proof.** By Proposition 6.4, we have \( f = k^n (t + 1)^n \). Then we have
\[
\begin{align*}
   h(t) &= (1 - t)^n f \left( \frac{t}{1-t} \right) \\
   &= (1 - t)^n k^n \left( \frac{t}{1-t} + 1 \right)^n \\
   &= \left[ (1 - t) \left( \frac{t}{1-t} + 1 \right) \right]^n k^n \\
   &= k^n 
\end{align*}
\]

**Proposition 6.6.** Let \( A(n, k) \) be a coordinate toric arrangement. Then the flag \( f \) polynomial for the poset of layers of \( A(n, k) \) is given by
\[
\tilde{f}_P(q_0, \ldots, q_n) = \sum_{S \subseteq [n]} k^{n-s_1} \binom{n}{s_1, s_2 - s_1, \ldots, s_{l-1} - s_1, n - s_l} q_S
\]

where \( [n] = \{0, 1, \ldots, n\} \), \( q_S = \prod_{i \in S} q_i \).

**Proof.** We just need to show that if given subset \( S = \{s_1 < s_2 < \cdots < s_l\} \subseteq [n] \ (|n| = \{0, 1, \cdots, n\}) \), \( \tilde{f}_S = \# \{c = y_1 < \cdots < y_l \in \mathcal{P} | \dim(y_i) = s_i \} = k^{n-s_1} k^{n-s_1} \binom{n}{s_1, s_2 - s_1, \ldots, s_{l-1} - s_1, n - s_l} \).

Fix \( S = \{s_1 < s_2 < \cdots < s_l\} \subseteq [n] \). We are trying to count the number of chains \( c = y_1 < y_2 < \cdots < y_l \) in \( \mathcal{P} \) with \( \dim(y_i) = s_i \). Note that each \( y_i \) is the intersection of \( n - s_i \) hypertori.
First we count number of layers that is possible to be $y_1$, the minimal element in such chain. $y_1$ is the intersection of $n - s_1$ hypertori and there are \( \binom{n}{n-s_1} \) ways to choose such hypertori. And for each hypertori, there are $k$ possible connected components to choose. Therefore there are $k^{n-s_1} \binom{n}{n-s_1}$ possible choices for $y_1$. Now fix a $y_i$, we count the number of choices for $y_{i+1}$. Note that $y_i$ is the intersection of $n - s_i$ connected components of $n - s_i$ different hypertori. Note $y_{i+1}$ is the intersection of $n - s_{i+1} < n - s_i$ connected components of $n - s_{i+1}$ different hypertori, and since all those chains are cyclic, we just need to show given subset $\tilde{S}$ of hypertori, there are $k^{n-s_{i+1}} \binom{n}{n-s_{i+1}}$ possible choices for $y_{i+1}$. Therefore, given such $S \subset [n]$, the number of chains satisfying our requirements is:

$$
\tilde{f}_S = k^{n-s_1} \binom{n}{n-s_1} \binom{n-s_1}{n-s_2} \cdots \binom{n-s_1}{n-s_l} \binom{n}{n-s_{l+1}} \cdots \binom{n}{n-s_1}
$$

Also notice that,

$$
\binom{n}{n-s_1} \frac{(n-s_1)!}{(n-s_2)! (n-s_1)!} \cdots \frac{(n-s_{l-1})!}{(s_l-s_{l-1})! n!} \frac{n!}{s_1!} = \binom{n-s_{i+1}}{s_1, s_2 - s_1, \ldots, s_{l-1} - s_1, n-s_1}
$$

Therefore we conclude: $\tilde{f}_S = k^{n-s_1} \binom{n}{s_1, n-s_1} \cdots \binom{n}{s_{l-1}, s_{l-1} - s_1, n-s_1}$.

Proposition 6.7. Let $A(n, k)$ be a coordinate toric arrangement. Then the flag $f$ polynomial for the poset of faces of $A(n, k)$ is given by

$$
\tilde{f}_F(q_0, \cdots, q_n) = \sum_{S \subset [n]} (-1)^{n-s_1} k^n (k+1)^{n-s_1} (s_1, s_1 - s_1, \cdots, s_2, s_1) q_S
$$

where $[n] = \{0, 1, \ldots, n\}$, $q_S = \prod_{i \in S} q_i$.

Proof. We just need to show given subset $S = \{s_1 < s_2 < \cdots < s_l\} \subset [n]$ ($[n] = \{0, 1, \ldots, n\}$),

$$
\tilde{f}_S = (-1)^{s_1-s_l} k^n (k+1)^{s_1-s_1} (s_1, s_1 - s_1, \cdots, s_2 - s_1, s_1)
$$

We will use the method in Lemma 3.4 to calculate $\tilde{f}_S$ here. Note that by Proposition 6.6, we already know that there are $k^{n-s_1} k^{n-s_1} (s_1, s_1 - s_1, \cdots, s_2 - s_1, s_1)$ chains in the poset of layers corresponding to $S$. And since all those chains are cyclic, we just need to know how many chains of faces associating with each chain of layers.

Fix an arbitrary chain of layers $y_1 > \cdots > y_{i+1} \in \mathcal{P}$, where $y_1$ is the minimal element in this chain, since we are ordering the poset of layers and the poset of faces by reverse inclusion. We want to know how many chains of faces corresponding to this chain of layers. We first see how many pieces does $y_1$ have. Note that $y_1$ is the intersection of $n - s_1$ hypertori. We want to see the subposet above $y_1$, which should be the restriction of other $s_1$ hypertori which are not used forming $y_1$. Then the characteristic polynomial of this subposet is:

$$
\chi_{y_1} = \sum_{i=0}^{s_1} (-1)^i \binom{s_1}{i} k^i t^{s_1-i}
$$
Then the number of regions (i.e. the number of $s_1$-faces corresponding to $y_1$) is

$$\chi_{y_1}(0) = k^{s_1}$$

Now we consider the subposet between each interval $[y_i, y_{i+1}]$. Note $y_{i+1}$ is the intersection of $n - s_{i+1}$ hypertori, and let $I_{i+1}$ be the index set for those hypertori. Also, $y_i$ is the intersection of $n - s_i$ hypertori, and let $I_i$ be the index set for those hypertori. Then the subposet between interval $[y_{i+1}, y_i]$ is really a restriction of $\{H_j | j \in I_{i+1} \setminus I_i\}$ on $y_i$. So the characteristic polynomial between interval $[y_{i+1}, y_i]$ is

$$\chi_{[y_{i+1}, y_i]} = \sum_{j=1}^{s_{i+1} - s_i} (-1)^j \left( \binom{s_{i+1} - s_i}{j} \right) t^{s_{i+1} - s_i - j}$$

By Lemma 3.4, the number of chains in the poset of faces corresponding to interval $y_{i+1}, y_i$ is

$$\chi_{[y_{i+1}, y_i]}(-1) = \sum_{j=0}^{s_{i+1} - s_i} (-1)^{s_{i+1} - s_i} \binom{s_{i+1} - s_i}{j}$$

Therefore we have the coefficient to be

$$\tilde{f}_S = \sum_{c \in P} |\Pi_{i=1}^{k-1} \chi_{[y_{i+1}, y_i]}(-1)||\chi_{P \leq s_1}(0)|$$

$$= |k^{n-s_1}(s_1, s_2 - s_1, \ldots, s_{l-1} - s_l, n - s_l)|\sum_{j=0}^{s_{i+1} - s_i} (-1)^{s_{i+1} - s_i} \binom{s_{i+1} - s_i}{j}$$

$$= k^n \left( s_1, s_2 - s_1, \ldots, s_{l-1} - s_l, n - s_l \right) \prod_{i=1}^{k-1} 2^{s_{i+1} - s_i}$$

7. Sagemath Program

7.1. Introduction.

`Toric_Arrangement_Polynomials.sage` is the program we wrote to help us generate lots of examples including higher dimensions where we cannot imaging. The `ToricArrangement` class will take in a matrix of the form we discussed in Section 2.2 (Associated Matrix), and will automatically generate `poset of layers` and store it as a variable called `poset of layers`, while `dict` is another variable which will return elements of the poset of layers (aka connected components of intersections). `arr_max` will return you the associated matrix of the toric arrangement and `dim` will return you the n-torus space the toric arrangement is in.

Here's a list of useful functions in the `ToricArrangement` class:

- `characteristic_polynomial()`: will return the characteristic polynomial of the toric arrangement.
The `f_polynomial()` function will return the f-polynomial of the toric arrangement. The `h_polynomial()` function will return the h-polynomial of the toric arrangement. The `flag_f_polynomial()` function will return the reduced flag f-polynomial of the toric arrangement. The `flag_h_polynomial()` function will return the reduced flag h-polynomial of the toric arrangement. The `cd_index()` function will return the cd-index of the reduced flag h-polynomial of the toric arrangement (the algorithm is based on [ERS09]).

The `cd_index_new_alg()` function will return the cd-index of the reduced flag h-polynomial of the toric arrangement (this is our new recursive algorithm directly derived from reduced flag h-polynomial and will be explained in the next subsection; note that our algorithm will be a little bit slower for runtime than [ERS09] but save more memory, please choose either algorithm to your favor). Note that our program only works under Regular Cell Complex Assumption, and you can use the `check_qualification()` function (a function outside our ToricArrangement class) to check if your associated matrix satisfies our assumption. But our `check_qualification()` function will check a slightly stricter assumption.

### 7.2. Sage Code

```python
class ToricArrangement(object):
    def __init__(self, m):
        TEST:
        sage: m = matrix(QQ, 4, 3, [2, 0, 1, 0, 2, 1, 1, 1, 1, -1, 1])
        sage: T = ToricArrangement(m)
        sage: T
        <__main__.ToricArrangement object at 0x3359db950>
        sage: sage: T.poset_of_layers
        Finite poset containing 11 elements
        sage: T.dict
        {0: [0 0 1],
         1: [ 1 -1  1],
         2: [ 1  1  1],
         3: [0 1 1/2],
         4: [0 1 1],
         5: [ 1  0 1/2],
         6: [1 0 1],
         7: [ 1  0 1/2]
         [ 0 1 1/2],
         8: [ 1  0 1]
         [ 0 1 1/2],
         9: [ 1  0 1/2]
         [ 0 1 1],
         10: [1 0 1]
         [0 1 1]}
BUGS:
sage: m = matrix(QQ, 2, 3, [2, -1, 1, 2, 1, 1/2])
''
# below defines the private variables
self.arr_mat = m
self.poset_of_layers = poset_of_layers(m)
self.dict = poset_dictionary(m)
self.dim = m.ncols() - 1

def characteristic_polynomial(self):
    '''
    TEST:
sage: T.characteristic_polynomial()
q^2 - 6*q + 8
    '''
    # the characteristic polynomial is defined on the dual of our poset
    pop = self.poset_of_layers.dual()
    return pop.characteristic_polynomial()

def f_polynomial(self):
    '''
    OUTPUT: the f_polynomial of an arrangement
    TEST:
sage: m = matrix(QQ, 4, 3, [2, 0, 1, 0, 2, 1, 1, 1, 1, 1, -1, 1])
sage: T = ToricArrangement(m)
sage: T.f_polynomial()
8*q^2 + 12*q + 4
    ALGORITHM:
We count the faces number of each dimension i in range [1, dim]
Step1: For each i, construct a list of layers of dimension i
Step2: For each layer of dimension i, count the number of
    dimension-i faces included.
    
dim = self.arr_mat.ncols() - 1
q = polygen(ZZ, 'q')
res = 0
for i in range(dim + 1):
    S = [i]
    temp = 0
    # following count for the faces number of dimension i
    # chai is the list of layers of dimension i
    chai = self.flag_chain_of_layers(S)
    for c in chai:
        # here increase temp by the number of dim-i face
```python
    # corresponding to each layer of dim i
    temp += self.num_chains_of_faces(c)
    # here temp count for f_i, the coefficient for q^i
    res += temp * q^i
    return res

def h_polynomial(self):
    '''
    OUTPUT: return the h polynomial
    TEST:
    sage: m = matrix(QQ,4,3,[2,0,1,0,2,1,1,1,1,1,-1,1])
    sage: T = ToricArrangement(m)
    sage: T.h_polynomial()
    4*q + 4
    '''
    q = polygen(ZZ, 'q')
    f = self.f_polynomial()
    d = f.degree()
    # following comes from the definition of h polynomial
    f = (1 - q) ** d * f(q = q/(1 -q))
    return q.parent(f)

def flag_f_polynomial(self):
    '''
    OUTPUT: the flag_f_polynomial of an arrangement
    TEST:
    sage: m = matrix(QQ,4,3,[2,0,1,0,2,1,1,1,1,1,-1,1])
    sage: T = ToricArrangement(m)
    sage: T.flag_f_polynomial()
    48*q0*q1*q2 + 24*q0*q1 + 24*q0*q2 + 24*q1*q2 + 4*q0 + 12*q1 + 8*q2
    ALGORITHM:
    We loop though each subset S of [n] to obtain the term \tilde{f}_S q_S
    Step1: For each S, collect all corresponding chains of layers
    Step2: For each chain of layer, collect all corresponding chains of faces
    '''
    n = self.arr_mat.ncols() - 1
    poly = PolynomialRing(ZZ, 'q', n+1)
    q = poly.gens()
    L = list_subsets_of_n(n)
    res = 0
```
# each run of this loop, we add a term $\tilde{f}_S q_S$ for res
for S in L:
    # coeff count for the coefficient of $q_S$
    coeff = 0
    # chai is a list of all chain of layers corresponding to S
    chai = self.flag_chain_of_layers(S)
    for c in chai:
        # following increase coeff by the number of chains of faces
        # corresponding to each chain of layers
        coeff += self.num_chains_of_faces(c)
        # following construct temp as the term $\tilde{f}_S q_S$
        temp = coeff
        for i in S:
            temp *= q[i]
        res += temp

return res

def flag_h_polynomial(self):
    '''
    OUTPUT: the flag h_polynomial of an arrangement
    TEST:
    sage: m = matrix(QQ,4,3,[2,0,1,0,2,1,1,1,1,1,-1,1])
    sage: T = ToricArrangement(m)
    sage: T.flag_h_polynomial()
    8*q0*q1 + 12*q0*q2 + 4*q1*q2 + 4*q0 + 12*q1 + 8*q2
    '''
    n = self.arr_mat.ncols() - 1
    poly = PolynomialRing(ZZ, 'q', n+1)
    q = poly.gens()
    L = list_subsets_of_n(n)
    res = 0
    # following construct the flag h polynomial by a modified version of
    # our original formula (we put the product term inside summand
    # to avoid fraction)
    for S in L:
        coeff = 0
        chai = self.flag_chain_of_layers(S)
        for c in chai:
            coeff += self.num_chains_of_faces(c)
            temp = coeff
            # following represent what flag h polynomial is really doing:
            # change monomial $q_S$ in flag f polynomial to be
            # \prod \limits_{i \in S} q_i \prod \limits_{i \not \in S} (1-q_i)
            for i in S:
temp *= q[i]

for i in range(n + 1):
    if not i in S:
        temp *= (1 - q[i])
res += temp
return res

def cd_index(self):
    
    Return: a cd polynomial

    TEST:
    sage: m = matrix(QQ, 3, 3, [3, -1, 1, 1, -2, 1, 0, 1, 1/5])
    sage: T = ToricArrangement(m)
    sage: T.cd_index()
    8*c*d + 7*d*c
    sage: m = matrix(QQ, 3, 3, [3, -1, 1, 1, -2, 1, 0, 1, 1/5])
    sage: T = ToricArrangement(m)
    sage: T.cd_index()
    8*c*d + 7*d*c

    cd = star_func(omega_func(a_s_b(H_prime(s))))
    F.<c,d> = FreeAlgebra(ZZ, 2)
    return F(add_mult(cd))/2

def cd_index_new_alg(self):
    
    TEST:
    sage: m = matrix(QQ, 3, 3, [3, -1, 1, 1, -2, 1, 0, 1, 1/5])
    sage: T = ToricArrangement(m)
    sage: T.cd_index_new_alg()
    8*c*d + 7*d*c
    sage: m = matrix(QQ, 3, 3, [3, -1, 1, 1, -2, 1, 0, 1, 1/5])
    sage: T = ToricArrangement(m)
    sage: T.cd_index_new_alg()
    8*c*d + 7*d*c

    n = self.dim
    # cd_list is all possible cd monomial of degree n
    cd_list = possible_cd_str_of_n(n)
    # we use cd monomials to be the keys and construct a dictionary,
# where the value indicate the coefficient for its key monomial

cd_dict = {}

# we let the default coefficient to be -1
for i in cd_list:
    cd_dict[i] = -1

# here need give initial coeff for some in cd_dict
# we initial the coefficient for cd-monomial with only one d
poly = PolynomialRing(ZZ, 'q', n+1)
q = poly.gens()
flag_h = self.flag_h_polynomial()

# face_num is a list of face number where face_num[i] = f_i
face_num = []
for i in range(n + 1):
    face_num.append(flag_h[q[i]])

# each loop bellow construct a cd monomial with only one d
# and assign the value for that monomial in cd_dict
# the coefficient for such monomial is showed in paper
# an base case in recursion
for k in range(n):
    temp = ''
    for i in range(k):
        temp += 'c'
        temp += 'd'
    for i in range(k + 1, n):
        temp += 'c'
    val = 0
    for i in range(k + 1):
        val += (-1)^(k - i)*face_num[i]
    cd_dict[temp] = val

    # above initialized cd strings with only one d

# below each loop generate a binary string represent a monomial q_S
# by loop through all \~\{h\}_{S}, we can obtain coefficients for all cd monomials

    #(see proof for this in paper)
    # by k in this range, we can generate all monomial q_S where S \subset [n]
    for k in range(1, 2^(n+1) - 1):
        # b is the binary string for k
        b = bin(k)
        b = b[2:]
        # b_str extend b to be a binary string of n+1 digit
        b_str = ''
        for i in range(n - len(b) + 1):
            b_str += '0'
        for i in range(len(b)):
            b_str += b[i]
# b_list is a list of all possible cd monomials to use q_S as a term
b_list = cd_str_for_bin(b_str)
c = ''
for i in range(len(b_str)):
c += 'c'
if c in b_list:
b_list.remove(c)
# below we use mono to construct the monomial q_S
mono = 1
for i in range(len(b_str)):
    if b_str[i] == '1':
        mono *= q[i]
# coeff = \tilde{h} \cdot S
coeff = flag_h[mono]
# we use target to record the only cd monomial in b_list which doesn't have its coefficient yet
# the sum count the sum of the coefficient for all other cd monomials in b_list
# See the proof for there is at most one unkown cd monomial coefficient in b_list in paper
target = ''
sum = 0
for s in b_list:
    if cd_dict[s] == -1:
        target = s
    else:
        sum += cd_dict[s]
if not target == '':
cd_dict[target] = coeff - sum
# below need to construct free algebra form of the cd monomial from the dictionary
F.<c,d> = FreeAlgebra(ZZ,2)
res = 0
for s in cd_list:
temp = cd_dict[s]
for i in range(len(s)):
    if s[i] == 'c': temp *= c
    else: temp *= d
res += temp
return res

# The following code are some helper functions to construct poset and polynomials
# ERS09 Algorithm
def dual_with_empty_top(self):
    '''
OUTPUT: the dual poset of poset of layers with a top element (empty face)

TEST:

sage: T = ToricArrangement(m)
sage: T.poset_of_layers
Finite poset containing 11 elements
sage: T.dual_with_empty_top()
Finite poset containing 12 elements

sage: T = ToricArrangement(m)
sage: T.poset_of_layers
Finite poset containing 34 elements
sage: T.dual_with_empty_top()
Finite poset containing 35 elements

# Get the cover relations from our poset of layers and its cardinality
cover_relations = self.poset_of_layers.cover_relations()
num_element = self.poset_of_layers.cardinality()
vertex = list(range(0,num_element + 1))

# Add some new cover relations: empty face is included in all the 0-faces
dim_zero_elts = T.flag_chain_of_layers([0])
for i in dim_zero_elts:
    i.insert(0,num_element)
cover_relations.extend(dim_zero_elts)

# Create a new poset with empty face and return its dual
P = Poset([vertex,cover_relations],cover_relations=False)
return P.dual()

def str_ab_index(self):

    OUTPUT: an ab-string derived from the dual of poset of layers with a top (i.e. the empty face)

    TEST:

    sage: m=matrix(QQ,3,3,[[3,-1,1,1,1,1,1,1,1,1/5])
    sage: T=ToricArrangement(m)
    sage: T.str_ab_index()
    '6bb+2ba+6ab+aa'

    # Get the flag h-polynomial from the dual of poset of layers with top (i.e. empty face), split each term as a string and store them in a list
    P = self.dual_with_empty_top()
    from sage.combinat.posets.posets import Poset
    h = P.flag_h_polynomial()
    h_str = str(h)
h_list = h_str.split('+')
l = list(range(1,P.height()-1))
new_h_list = []
ab_list = []
# Fix ' ' problem in string
for i in h_list:
    if i[0] == ' ':
        new_h_list.append(i[1:])
    else:
        new_h_list.append(i)

# First notice that we don't need the term x_i where i is the rank of the
top element (i.e. empty face); then we run a loop in new_h_list: if x_i is in
the term, we add a 'b'; if not, we add an 'a'; finally we append our ab-
string for each term to a new list called ab_list
for i in new_h_list:
    t = split_string(i)
s = t[0]
    for j in l:
        if 'x'+ str(j) in i:
            s = s + 'b'
        else:
            s = s + 'a'
    ab_list.append(s)
# Return the ab polynomial as a string
return get_string_from_list(ab_list)

def num_chains_of_faces(self, C):
    '''
    INPUT: C, a list of integers, representing a chain
    OUTPUT: number of chains of faces corresponding to C
    
    NOTE: this function serves for flag f_polynomial
    
    TEST:
    sage: m = matrix(QQ,4,3,[2,0,1,0,2,1,1,1,1,1,-1,1])
sage: T = ToricArrangement(m)
sage: C = [0,10]
sage: T.num_chains_of_faces(C)
8
    '''
    # The following code works for cases fulfill our requirement
    p = self.poset_of_layers
    res = 1
    # sub is all elements in the poset of layers below C[0]
sub = p.principal_order_ideal(C[0])
#sub is the subposet below C[0]
subp = p.subposet(sub)
#but plug 0 into poly, we obtain the number of faces corresponding to C[0]
poly = subp.dual().characteristic_polynomial()
res *= abs(poly(0))
#following each loop we construct a subposet between interval [c[i],c[i+1]]
#by plug in -1 to its characteristic polynomial, we obtain the number of
chains in the poset of faces corresponding to c[i]>c[i+1] in the poset of
for i in range(len(C) - 1):
itv = p.closed_interval(C[i], C[i + 1])
subp = p.subposet(itv)
poly = subp.dual().characteristic_polynomial()
res *= abs(poly(-1))
return res

def flag_chain_of_layers(self, S):
    ",",
    INPUT: S \subset [n]
    OUTPUT: return a set of chains. Each chain is a tuple representing
    a chain of layers corresponding to S.
    
    NOTE: this function serves for flag f_polynomial
    
    TEST:
    sage: m = matrix(QQ,4,3,[2,0,1,0,2,1,1,1,1,1,-1,1])
    sage: T = ToricArrangement(m)
    sage: S
    [0, 2]
    sage: c = T.flag_chain_of_layers(S)
    sage: c
    [[10, 0], [9, 0], [8, 0], [7, 0]]
    
    ALGORITHM:
    Step1: collect all elements in the poset of layers with dimension in S
    Step2: construct a new poset using above elements ordered by
    inclusion
    Step3: use build in functions to obtain a list of chains in that
    subposet
    
    dim = self.arr_mat.ncols() - 1
    sset = set()}
```python
t = {} for k in S:
    sset.add(k)
#following we collect all elements in self.dict with dimension in S
#put the qualified elements with its original label of vertex
for i in self.dict:
    m = self.dict[i]
    if dim - rank(charact_part(m)) in sset:
        t[i] = m
        cmp_fn = lambda p,q: is_subtorus(t[p],t[q])
from sage.combinat.posets.posets import Poset
#subp is the subposet with all elements having dimension in S
subp = Poset((t, cmp_fn))
#C is all chains of layers we want
C = subp.chains()
res = list(C.elements_of_depth_iterator(len(S)))
return res

#FOLLOWING CODE IS NOT IN THE DECLARATION OF TORICARRANGEMENT CLASS
#The following code are some useful functions
def check_qualification(m):
    return if we can use our algorithm to calculate flag f_polynomial
    TEST:
sage: m = matrix(QQ,4,3,[2,0,1,0,2,1,1,1,1,-1,1])
sage: m
\[
\begin{pmatrix}
2 & 0 & 1 \\
0 & 2 & 1 \\
1 & 1 & 1 \\
1 & -1 & 1 \\
\end{pmatrix}
\]
sage: check_qualification(m)
True
sage: m = matrix(QQ,2,3,[2,-1,1,2,1,1/2])
sage: m
\[
\begin{pmatrix}
2 & -1 & 1 \\
2 & 1 & 1/2 \\
\end{pmatrix}
\]
sage: check_qualification(m)
False
sage: m = matrix(QQ,3,3,[1,0,1,0,1,1,2,-1,1])
sage: m
\[
\begin{pmatrix}
1 & 0 & 1 \\
0 & 1 & 1 \\
2 & -1 & 1 \\
\end{pmatrix}
\]
```

sage: check_qualification(m)
False

sage: m = matrix(QQ,4,3,[1,0,1,0,1/2,0,1,0,1,0,1/2])
sage: m
[ 1 0 1]
[ 1 0 1/2]
[ 0 1 1]
[ 0 1 1/2]
sage: check_qualification(m)
True

...,
c = charact_part(m)
dim = m.ncols() - 1
if rank(c) < dim: return false
categ = []
for i in range(c.nrows()):
categ.append(-1)
#the ith elt of categ mark if c[i] belongs to some category already
curr = 0
#curr mark which category should the next independent vector be
#curr should be at most dim - 1
count = []
#count[i] record #of element in ith category
#count should be at most dim - 1 long
for i in range(c.nrows()):
    if categ[i] == -1:
        #meaning c[i] is not discovered before
        categ[i] = curr
        num = gcd(c[i])
        for j in range(i + 1, c.nrows()):
            #now see if i and j are parallel
            tl = []
            for k in range(c.ncols()):
                tl.append(c[i][k])
            for k in range(c.ncols()):
                tl.append(c[j][k])
            temp = matrix(QQ,2,c.ncols(),tl)
            tl = []
            if rank(temp) == 1:#m[j] is parallel with m[i]
                categ[j] = categ[i]
                num += gcd(c[j])
        count.append(num)
        curr += 1
    ind = 0
for i in count:
if i >= 2: ind += 1
if ind < dim: return false
return true

# The following code are some helper functions
# Helper functions for cd-index
def list_subsets_of_n(n):
    
    OUTPUT: all nonempty subset of n
    
    TEST:
sage: l = list_subsets_of_n(5)
sage: len(l)
31
    
    # base case for recursive algorithm
    if n == 0:
        return [[0]]
    else:
        # let l be all subsets of n-1
        l = list_subsets_of_n(n - 1)
        # first collect all elements in l in our result representing all subsets
        # of [n] without n
        res = l
        # now collect all subsets of [n] with n
        for i in range(len(l)):
            temp = []
            for j in l[i]:
                temp.append(j)
            temp.append(n)
            res.append(temp)
        res.append([n])
        res.sort()
        return res

def add_mult(s):
    
    Input: a cd polynomial with each term a cd-string
    Return: a cd polynomial adding '*' to each term
    
    TEST:
sage: s = str('12ccd+2cdc+24dd')
sage: add_mult(s)
'12*c*c*d+2*c*d*c+24*d*d'
    
    # split ab polynomial by '*' and store each term in a list
    

s = s.split('+')
cd_poly = []
# add '*' between each 'c'/'d' variable and return a polynomial-like string
for i in s:
t = split_string(i)
cd = t[0]
for j in t[1]:
    cd += '*'
    cd += j
    cd_poly.append(cd)
return get_string_from_list(cd_poly)

def a_s_b(s):
    
    Input: an ab polynomial
    Return: an ab polynomial adding 'a' in the front and 'b' in the back for each term

    TEST:
sage: s = str('12abbaa+8abaaa+9aabba+2aaaaa')
sage: a_s_b(s)
'12aabbaabaab+8aabaaab+9aababaab+2aaaaaab'

    # split ab polynomial by '+' and store each term in a list
s = s.split('+')
adding_a_b = []
for i in s:
    # for each term, we split the number part and variable part
    t = split_string(i)
    # add an 'a' in the front and 'b' in the back and append to the new list
    adding_a_b.append(t[0] + 'a' + t[1] + 'b')
# return a polynomial-like string
return get_string_from_list(adding_a_b)

def omega_func(s):
    
    Input: an ab polynomial
    Return: a polynomial replacing each 'ab' with '2d' and other letters with 'c'

    TEST:
sage: s = str('12abbaa+8abaaa+9aabba+2aaaaa')
sage: omega_func(s)
'24dccc+16dccc+18dccc+2cccc'

    # split ab polynomial by '+' and store each term in a list
s = s.split('+')
```python
cd_string = []
for i in s:
    cd_term = ''
    # for each term, we split the number part and variable part
    t = split_string(i)
    # check if there is no variable part
    if t[0] == '':
        t0 = 1
    else:
        t0 = int(t[0])
    j = 0
    # replace each 'ab' with '2d' and other letters with 'c' and append the
    # result to the new list
    while j < len(t[1]):
        if j < len(t[1]) - 1 and t[1][j] + t[1][j+1] == 'ab':
            cd_term += 'd'
            t0 *= 2
            j += 2
        else:
            cd_term += 'c'
            j += 1
    cd_string.append(str(t0) + cd_term)
# return a polynomial-like string
return get_string_from_list(cd_string)

def H_prime(s):
    '''
    Input: an ab polynomial
    Return: a polynomial with last letter removed for each term
    '''
    TEST:
    sage: s = str('ab+ba')
    sage: s
    'ab+ba'
    sage: H_prime(s)
    'a+b'
    sage: s = str('abbaa+abaaa+aabba+aaaaa')
    sage: s
    'abbaa+abaaa+aabba+aaaaa'
    sage: H_prime(s)
    'abba+abaa+aabb+aaaa'
    sage: H_prime(s)
    'abba+abaa+aabb+aaaa'
    sage: s = str('7abba')
    sage: s
    '7abba'
    sage: H_prime(s)
    '7abb'
```
```python
647     # split ab polynomial by '+' and store each term in a list
648     s = s.split('+')
649     last_removed = []
650     for i in s:
651         # for each term, we split the number part and variable part
652         t = split_string(i)
653         # remove the last letter of the variable part if the variable part exists
654         if len(t[1]) > 1:
655             last_removed.append(t[0] + t[1][:-1])
656         # return a polynomial-like string
657         return get_string_from_list(last_removed)
658
659     def star_func(s):
660         '''
661         Input: an ab polynomial
662         Return: a polynomial with ab-part order reversed
663         TEST:
664         sage: s = str('abbaa+abaaa+aabba+aaaaa')
665         sage: star_func(s)
666         'aabba+aaaba+abbaa+aaaaa'
667         sage: s = str('12abbaa+8abaaa+9aabba+2aaaaa')
668         sage: star_func(s)
669         '12aabba+8aaba+9aabba+2aaaaa'
670         '''
671     # split ab polynomial by '+' and store each term in a list
672     s = s.split('+')
673     reversed = []
674     for i in s:
675         # for each term, we split the number part and variable part
676         t = split_string(i)
677         # reverse the ab-variable part and append the result to a new list
678         reversed.append(t[0] + t[1][::-1])
679     # return a polynomial-like string
680     return get_string_from_list(reversed)
681
682     def get_string_from_list(l):
683         '''
684         Input: a list of strings where each string is a term of a polynomial
685         Return: a string of polynomial
686         TEST:
687         l = ['ab', 'ba', 'aa']
688         sage: get_string_from_list(l)
```
'ab+ba+aa'

s = ''

# add a '+' between each term and form a polynomial-like string
for i in l:
s += i
s += '+'
s = s[:-1]
return s

def check_number(s):
    '''
    Return true if the first element of the string is a number
    '''
    result = False
    if s[0] == '0' or s[0] == '1' or s[0] == '2' or s[0] == '3' or s[0] == '4' or s[0] == '5' or s[0] == '6' or s[0] == '7' or s[0] == '8' or s[0] == '9':
        result = True
    return result

def split_string(s):
    '''
    Separate the number part and the letter part and return a tuple
    '''
    num = ''
    while check_number(s):
        num += s[0]
s = s[1:]
    return (num, s)

# Our new algorithm

def possible_cd_str_of_n(n):
    '''
    given dimension n, output all possible cd strings
    '''
    TEST:
sage: s = str('22ab')
sage: split_string(s)
('22', 'ab')
sage: s = str('7aabba')
sage: split_string(s)
('7', 'aabba')

    num = ''
while check_number(s):
    num += s[0]
s = s[1:]
return (num, s)
#d has degree 2, c has degree 1, we want to come up with all cd monomials of degree n

#the pure c monomial is removed since it can’t appear in the cd-index form of flag h-polynomial

#base case for recursive algorithm
if n == 0: return ['c']
if n == 1: return ['d', 'cc']
S = set ()
l1 = possible_cd_str_of_n(n - 1)  #should insert 'c' inside
l2 = possible_cd_str_of_n(n - 2)  #should insert 'd' inside
#l1 is a list of cd monomial with degree n-1, we insert 'c' into all possible position
for i in l1:
    for j in range (len (i)):
        temp = str_insert (i,j,'c')
        S.add(temp)
#l2 is a list of cd monomial with degree n-1, we insert 'd' into all possible position
for i in l2:
    for j in range (len (i)):
        temp = str_insert (i,j,'d')
        S.add(temp)
#above we used S to collect all monomials in order to remove duplication
res = []
#following we collect all elements in S into a list res[]
c = ''
for i in range(n + 1):
    c += 'c'
for n in S:
    res.append(s)
#remove the pure c monomial in res[]
if c in res:
    res.remove(c)
def str_insert(s, i, c):
    '''
    TEST:
    sage: s = 'abcde'
    sage: str_insert(s, 3, 'g ')
    'abcgde'
    sage: s = 'abc'
    sage: str_insert(s, 3, 'g ')
    'abcg'
    '''
    temp = ''
    for k in range(i):
        temp += s[k]
    temp += c
    for k in range(i, len(s)):
        temp += s[k]
    return temp

def cd_str_for_bin(b):
    '''
    sage: s = '101'
    sage: cd_str_for_bin(s)
    ['dc ', 'ccc ', 'cd ']
    sage: s = '10101'
    sage: cd_str_for_bin(s)
    [' cdd ', 'ccdc ', 'dcd ', 'cdcc ', 'dccc ', 'cccd ', 'ddc ', 'ccccc ']
    '''
    # base cases for recursive algorithm
    if b == '0' or b == '1': return ['c ']
    if b == '01 ' or b == '10 ': return ['d ', 'cc ']
    res = []
    S = set()
    n = len(b)
    # if b[0] != b[1], we can replace the first two digits by a 'd'
    if b[0] != b[1]:
        tail = b[2:]  # call cd_str_for_bin recursively, l is the list of all possible cd-
                      # monomial after remove the first two digit of b, then we add a 'd' to the
                      # front of all elements in l
        l = cd_str_for_bin(tail)
        for i in l:
            S.add(str_insert(i, 0, 'd '))
        S.add(str_insert(str_insert(tail, 0, 'c '), 0, 'c '))
    else:
        l = cd_str_for_bin(b)
        for i in l:
            S.add(str_insert(i, 0, 'd '))
        S.add(str_insert(str_insert(b, 0, 'c '), 0, 'c '))
# we can always replace the first digit by a 'c'

tail = b[1:]
l = cd_str_for_bin(tail)
for i in l:
    S.add(str_insert(i,0,'c'))

# if the last two digits of b is different, we are allowed to replace the last
two digit by a 'd'
if b[n - 1] != b[n - 2]:
    pre = b[: -2]
l = cd_str_for_bin(pre)
for i in l:
    S.add(str_insert(i,len(i),'d'))
    S.add(str_insert(str_insert(i,len(i),'c'),len(i) + 1,'c'))

# we can always replace the last digit by 'c'
pre = b[: -1]
l = cd_str_for_bin(pre)
for i in l:
    S.add(str_insert(i,len(i),'c'))
for i in S:
    res.append(i)
return res

# Helper functions for constructing poset of layers

def poset_of_layers(m):
    
    sage: m = matrix([[1,0,1],[0,1,1]])
    sage: m
    
    [1 0 1]
    [0 1 1]
    
sage: P = poset_of_layers(m)
    sage: P
    
    Finite poset containing 4 elements

    sage: m = matrix(QQ,2,3,[1,-1,1,1,1,1])
    sage: m
    
    [ 1 -1 1]
    [ 1 1 1]
    
sage: P = poset_of_layers(m)
    sage: P
    
    Finite poset containing 5 elements

    sage: m = matrix(QQ,3,3,[2,-1,1,1,0,1,0,1,1])
    sage: m
    
    [2 -1 1]
    [1 0 1]
    [0 1 1]
```
sage: P = poset_of_layers(m)
sage: P
Finite poset containing 6 elements

sage: m = matrix(QQ,4,3,[2,0,1,0,2,1,1,1,1,1,-1,1])
sage: m
[ 2 0 1]
[ 0 2 1]
[ 1 1 1]
[ 1 -1 1]
sage: P = poset_of_layers(m)
sage: P
Finite poset containing 11 elements
```
l = list_of_conn_comp(m)
p_elt = poset_element(l)
L = []
dim = m.ncols() - 1
wl = []
# use matrix [0, ..., 0, 1] to represent the whole space T = (S^1)^n
for i in range(0, dim):
    wl.append(0)
    wl.append(1)
whole_space = matrix(QQ, 1, len(wl), wl)
L.append(whole_space)
# append L by the connected components of all intersections in p_elt
for i in p_elt:
    if intersection_exist(i):
        temp = conn_comp_intersection(i)
        L = L + temp
# use rem_dup to collect all elements in L after remove duplication
rem_dup = []
for i in L:
    exist = False
    for j in rem_dup:
        if is_subtorus(i, j) and is_subtorus(j, i):
            exist = True
    if not exist:
        rem_dup.append(i)
# construct the dictionary for rem_dup which is the final elements in the poset of layers
t = {}
for i in range(len(rem_dup)):
    t[i] = rem_dup[i]
return t

def dict_find_key(t, m):
    #TEST:
sage: m = matrix(QQ, 3, 3, [2, -1, 1, 1, 0, 1, 0, 1, 1])
sage: m
[ 2 -1 1]
[ 1 0 1]
[ 0 1 1]
sage: P = poset_of_layers(m)
sage: P
Finite poset containing 6 elements
sage: t = poset_dictionary(m)
sage: t
{
    0: [0 0 1],
    1: [0 1 1],
    2: [1 0 1],
    3: [2 -1 1],
    4: [0 1 0]
}

sage: m1 = matrix(QQ,2,3,[1,0,1/2,0,1,1])

sage: i = dict_find_key(t, m1)
sage: i
5

for i in t:
    if t[i] == m1:
        return i;
return -1

def poset_element(l):
    TEST:
    sage: m = matrix(QQ,3,4,[3,6,9,1,0,4,6,1,0,0,4,1])
    sage: m
    [3 6 9 1]
    [0 4 6 1]
    [0 0 4 1]
    sage: l = list_of_conn_comp(m)
    sage: l
    [[[1, 2, 3, 1/3], [1, 2, 3, 2/3], [1, 2, 3, 1]],
     [[0, 2, 3, 1/2], [0, 2, 3, 1]],
     [[0, 0, 1, 1/4], [0, 0, 1, 1/2], [0, 0, 1, 3/4], [0, 0, 1, 1]]]
    sage: k = poset_element(l)
    sage: len(k)
59

sage: m = matrix(QQ,2,3,[1,-1,1,1,1,1])

sage: l = list_of_conn_comp(m)

sage: l
[[[1, -1, 1]], [[1, 1, 1]]]

sage: k = poset_element(l)

ALGORITHM:
This is a recursive algorithm to return all possible intersection of a poset.

Step 1: remove the last hypertori (last row of l)

Step 2: call poset_element recursively to obtain all possible intersection for all other hypertori, obtain a list pre.

Step 3: for each element in pre, intersect with all connected components in the last hypertori to obtain a list of new intersections, return this list.

Append all connected components in the last hypertorus and all elements in pre.

```python
res = []

# base case for recursion
if len(l) == 1:
    return matrices_from_nested_list(l)

# remove and record the last hypertori
last_row = l.pop(len(l) - 1)
l_last = matrices_from_nested_list([last_row])

# pre is the list of all possible intersection for the rest of hypertori
pre = poset_element(l)

# append list with the singleton of connected components in the last hypertorus and append res with all intersections without the last hypertorus
res = res + l_last
res = res + pre

# intersect each intersection in pre with one connected components in the last hypertorus
for i in l_last:
    for j in pre:
        temp = matrix_append_row(j, i)
        res.append(temp)
return res
```

def conn_comp_intersection(m):
    '''
    INPUT: a matrix with multiple rows representing an intersection
    OUTPUT: a list of matrices representing the connected component of the input intersection
    
    TEST:
    sage: m = matrix(QQ,2,3,[1,-1,1,1,1,1])
    sage: m
    [ 1 -1  1]
    [ 1  1  1]
    sage: c = conn_comp_intersection(m)
    sage: c
    []
    '''
```
sage: m = matrix(QQ,2,4,[3,6,9,1,0,4,6,1])
sage: m
[3 6 9 1]
[0 4 6 1]
sage: c = conn_comp_intersection(m)
sage: c
[
[ 1 0 0 5/6] [ 1 0 0 1/6] [ 1 0 0 1/3]
[ 0 2 3 1/2] [ 0 2 3 1/2] [ 0 2 3 1],
[ 1 0 0 1/2] [ 1 0 0 2/3]
[ 0 2 3 1/2] [ 0 2 3 1]]
sage: m = matrix(QQ,2,3,[2,-1,1,1,1,1])
sage: m
[ 2 -1 1]
[ 1 1 1]
sage: c = conn_comp_intersection(m)
sage: c
[
[ 1 0 1/3] [ 1 0 2/3] [ 1 0 1]
[ 0 1 2/3] [ 0 1 1/3] [ 0 1 1]]
sage: m = matrix(QQ,3,3,[2,0,1,0,2,1,1,1,1,1,-1,1])
sage: m
[2 0 1]
[ 0 2 1]
[ 1 1 1]
[ 1 -1 1]
sage: c = conn_comp_intersection(m)
sage: c
[
[ 1 0 1/2] [ 1 0 1]
[ 0 1 1/2] [ 0 1 1]]
sage: m1 = matrix(QQ,3,3,[1,-2,1,3,4,1,2,1,2,1,3])
sage: m1
[ 1 2 1]
[ 3 4 1/2]
```
[ 1  2  1/3]
sage: c = conn_comp_intersection(m1)
sage: c
[]'''
res = []
if not intersection_exist(m):
    return res
# here separate each hypertorus in m to be a list of connected components
l = list_of_conn_comp(m)
k = matrices_from_nested_list(l)
# unimodulized k
pre_res = matrix_list_unimodulize(k)
# use S to remove the duplicated matrix in k
S = set(())
for k in pre_res:
    if (intersection_exist(k)):
        k.set_immutable()
        S.add(k)
for k in S:
    res.append(k)
return res

def matrix_list_unimodulize(L):
    '''
    INPUT: L is a list of matrix
    OUTPUT: a list of unimodular matrix
    TEST:
    sage: m1 = matrix(QQ,2,4,[1,2,3,1,0,4,6,1])
    sage: m2 = matrix(QQ,2,4,[1,2,-1,2,0,1,1,1])
    sage: M = [m1,m2]
sage: M
    [  [1 2 3 1]  [ 1  2 -1  2]
    [0 4 6 1], [ 0  1  1  1]  ]
sage: K = matrix_list_unimodulize(M)
sage: K
    [  [ 1  0  0  1/2]  [1 0 0 1]  [ 1  0 -3  1]
    [ 0  2  3 1/2], [0 2 3 1], [ 0  1  1  1]  ]
sage: is_unimodular(K[0])
    True
```
sage: is_unimodular(K[1])
True
sage: is_unimodular(K[2])
True
'''
res = []
# below each loop, we want to change an element in L to be a list of its
connected components in unimodular form
for m in L:
    # temp_ech is the integer echelon form of m
temp_ech = trace_integer_echelon(m)
    # if temp-ech is connected we append it to res
    if is_unimodular(temp_ech):
        res.append(temp_ech)
    # if not, we separate temp_ech to be a list of connected component temp_l
    # call matrix_list_unimodulize recursively to unimodulize it
    # and append its unimodular form to res
    else:
        temp_l = list_of_conn_comp(temp_ech)
        temp_k = matrices_from_nested_list(temp_l)
        res = res + matrix_list_unimodulize(temp_k)
return res

def trace_integer_echelon(m):
    '''
    trying new way to implement integer_echelon
    sage: m = matrix(ZZ,4,3,[2,0,1,0,2,1,1,-1,1,1,1,1])
    sage: m
    [ 2  0  1]
    [ 0  2  1]
    [ 1 -1  1]
    [ 1  1  1]
    sage: t = trace_integer_echelon(m)
    sage: t
    [ 1  1  1]
    [ 0  2  1]
    '''
    # we need append a ext_col x ext_col identity matrix to trace the row
    # operations
    ext_col = m.nrows()  
    char_m = charact_part(m)
    # l_trace collect the elements used in m_trace
    l_trace = []
for i in range(char_m.nrows()):
    for j in range(char_m.ncols()):
        l_trace.append(m[i][j])
    for k in range(0, i):
        l_trace.append(0)
    l_trace.append(1)
    for k in range(i + 1, ext_col):
        l_trace.append(0)
# m_trace is the matrix we obtained after appending extra identity matrix
# after char_m
m_trace = matrix(ZZ, char_m.nrows(), char_m.ncols() + ext_col, l_trace)
# now the append part give us the clue for row operations
int_ech_trace = m_trace.echelon_form()
l_int = []
for i in char_m:
    for j in i:
        l_int.append(j)
char_m_int = matrix(ZZ, char_m.nrows(), char_m.ncols(), l_int)
char_m_ech = char_m_int.echelon_form()
const = []
# each loop will count for the constant for the ith row of m
for i in range(char_m.nrows()):
    temp_const = 0
    # each loop below, we count the number of multiples of the kth row in m to
    # be used
    # to form the ith row in the echelon form of char_m
    for k in range(ext_col):
        temp_const += m[k][char_m.ncols()] * int_ech_trace[i][char_m.ncols() + k]
    # below we restrict each constant to be in range (0,1]
    while temp_const <= 0:
        temp_const += 1
    while temp_const > 1:
        temp_const -= 1
    const.append(temp_const)
res = matrix_append_column(char_m_ech, const)
rk = rank(char_m)
# below we erase all rows with 0 vector as characteristic part
res = matrix_erase_rows(res, rk)
return res

def intersection_existent(m):
INPUT: a matrix

OUTPUT: return a boolean value indicating if the intersection exists
(just need to see if some torus in m is parallel but not the same)

TEST:

```
sage: m1 = matrix(QQ,3,3,[1,2,1,3,4,1/2,1,2,1/3])
sage: m1
[ 1 2 1]
[ 3 4 1/2]
[ 1 2 1/3]
sage: intersection_exist(m1)
False
```

```
sage: m1 = m1 = matrix(QQ,3,3,[1,2,1,3,4,1/2,1,2,1])
sage: m1
[ 1 2 1]
[ 3 4 1/2]
[ 1 2 1]
sage: intersection_exist(m1)
True
```

NOTE: if two hypertori is not parallel, they must have at least one
intersection

The only case for a matrix representing empty intersection is that some
hypertori
inside is parallel with each other but not the same

```
char_m = charact_part(m)

# following loop verify if m[i] and m[j] has the same characteristic part but
different constant term
# in which case, intersection does not exists
for i in range(m.nrows()):
    for j in range(i + 1, m.nrows()):
        if char_m[i] == char_m[j] and m[i][m.ncols() - 1] != m[j][m.ncols() - 1]:
            return false;
return true
```

def new_is_subtorus(m1, m2):
    
    TEST:
    ```
sage: m1 = matrix(QQ,2,4,[1,0,0,1,0,2,3,1])
sage: m2 = matrix(QQ,2,4,[1,0,0,1/2,0,2,3,1/2])
sage: new_is_subtorus(m1, m2)
```
```python
if not is_unimodular(m1) or not is_unimodular(m2):
    print("Inout unimodular matrix!")
    return False
m1 = trace_integer_echelon(m1)
m2 = trace_integer_echelon(m2)
V = ZZ^(m1.ncols() - 1)

m1_int = charact_part(m1)
m1_int = m1_int.change_ring(ZZ)
m1_int_list = matrix_to_list(m1_int)

m2_int = charact_part(m2)
m2_int = m2_int.change_ring(ZZ)
m2_int_list = matrix_to_list(m2_int)

W1 = V.submodule(m1_int_list)
W2 = V.submodule(m2_int_list)

if not W2.is_submodule(W1):
    return False

for i in range(len(m2_int_list)):
    coord = W1.coordinates(m2_int_list[i])
temp = 0
    for j in range(len(coord)):
        temp += coord[j] * m1[i][m1.ncols()-1]
    if not temp == m2[i][m2.ncols()-1]:
        return False

return True
```

```python
def is_subtorus(m1, m2):
    '''
    Input: two matrices indicating two connected intersections
    Note that they are all primitive
    Output: return true if m1 is a subtorus of m2
    TEST:
    sage: m1 = matrix(QQ,2,4,[1,2,3,1,4,5,6,1])
sage: m2 = matrix(QQ,1,4,[1,2,3,1])
sage: m1
```
1312 [1 2 3 1]
1313 [4 5 6 1]
1314 sage: m2
1315 [1 2 3 1]
1316 sage: is_subtorus(m1,m2)
1317 True
1318 sage: is_subtorus(m2,m1)
1319 False
1320 sage: m1 = matrix(QQ,3,3,
1321 [4 5 6
1322 1 2 3
1323 7 8 9]
1324 sage: m2 = matrix(QQ,2,3,
1325 [7 8 9
1326 1 2 3]
1327 sage: m1
1328 [4 5 6]
1329 [1 2 3]
1330 [7 8 9]
1331 sage: m2
1332 [7 8 9]
1333 [1 2 3]
1334 sage: is_subtorus(m1,m2)
1335 True
1336 sage: is_subtorus(m2,m1)
1337 False
1338 
1339 V = ZZ^(m1.ncols() - 1)
1340 
1341 m1_int = charact_part(m1)
1342 m1_int = m1_int.change_ring(ZZ)
1343 m1_int_list = matrix_to_list(m1_int)
1344 
1345 m2_int = charact_part(m2)
1346 m2_int = m2_int.change_ring(ZZ)
1347 m2_int_list = matrix_to_list(m2_int)
1348 
1349 W1 = V.submodule(m1_int_list)
1350 W2 = V.submodule(m2_int_list)
1351 
1352 if not W2.is_submodule(W1):
1353    return False
1354    #in case we can't have a solution, since we only solved the equation in real number
1355    try:
1356    p = point_of_subtorus(m1)
1357    except:
1358    return False
1359    l = m2.ncols()
for n in m2:
    # see if that point in m1 in also in m2
    multiplication = p[0]^n[0]
    for i in range(1,l - 1):
        multiplication *= (p[i] ^ n[i])
    if not multiplication == e ^ (2 * pi * I * n[l - 1]):
        return False
return True

def point_of_subtorus(m):
    '''
    Input: a matrix represent a subtorus
    Output: a point in the subtorus (as a vector)
    TEST:
    sage: m1=matrix(QQ,2,3,[1,2,1,2,1,1])
    sage: m1
    [1 2 1]
    [2 1 1]
    sage: point_of_subtorus(m1)
    [e^(2/3*pi), e^(2/3*pi)]
    '''
    b = m.column(m.ncols()-1)
    A = charact_part(m)
    # first solve the equation in R^n
    v = A.solve_right(b)
    l = []
    for x in v:
        l.append(e ^ (2 * pi * I * x))
    return l

def list_of_conn_comp(m):
    '''
    INPUT: a matrix (a toric arrangement)
    OUTPUT: a list of nested list
    l[i], a nested list, represents the connected component of m[i]
    TEST:
    sage: m = matrix(QQ, 2, 4, [3,6,9,1,0,4,6,1])
    sage: m
    [3 6 9 1]
    [0 4 6 1]
    sage: l = list_of_conn_comp(m)
```python
sage: l
[[[1, 2, 3, 1/3], [1, 2, 3, 2/3], [1, 2, 3, 1]],
 [[0, 2, 3, 1/2], [0, 2, 3, 1]]]

sage: m = matrix(QQ,3,4,[3,6,9,1,0,4,6,1,0,0,4,1])

sage: m
[3 6 9 1]
[0 4 6 1]
[0 0 4 1]

sage: l = list_of_conn_comp(m)
sage: l
[[[1, 2, 3, 1/3], [1, 2, 3, 2/3], [1, 2, 3, 1]],
 [[0, 2, 3, 1/2], [0, 2, 3, 1]],
 [[0, 0, 1, 1/4], [0, 0, 1, 1/2], [0, 0, 1, 3/4], [0, 0, 1, 1]]]

'''
l = []
# each loop will append l with a nested list
# l[i] represent a list of all connected components of m[i]
for i in range(0, m.nrows()):
    temp = []
    for j in m[i]:
        temp.append(j)
tl = [temp]
    ma = list_to_matrix(tl)
k = conn_comp_torus(ma)
l.append(k)
return l

def is_unimodular(m):
    '''
    we should only verify if the matrix
    formed by removing the last column of m is unimodular
    this function is really seeing if a layer has only one connected
    component
    TEST:
    sage: m = matrix(QQ,2,4,[1,2,3,1,0,4,6,1])
    sage: m
    [1 2 3 1]
    [0 4 6 1]
    sage: is_unimodular(m)
    False
    sage: m = matrix(QQ,2,4,[1,2,3,1,0,2,3,1])
    sage: m
```

\[\begin{align*}
\text{ALGORITHM:} \\
\text{This comes from a theorem in...}
\end{align*}\]

```python
m = charact_part(m)
r = m.rank()
m = m.minors(r)
d = gcd(m)
return abs(d) == 1
```

```python
def conn_comp_torus(m):
    
    m should be a single torus,
    this function will return the connected component of m as a list of torus
    (matrix with only one row)
    Say if the constant is c, it represents \(e^{2\pi c}\)

    TEST:
    sage: m = matrix(QQ, [0,4,6,1])
    sage: r = conn_comp_torus(m)
    sage: r
    [[0, 2, 3, 1/2], [0, 2, 3, 1]]
    sage: m = matrix(QQ, [1,2,3,1])
    sage: m
    sage: r = conn_comp_torus(m)
    sage: r
    [[0, 1, 3/4], [0, 1, 1/4]]
    
    ALGORITHM:
    For a single torus m not being primitive, it has different connected
    components.
    We want to recover the nested list form representing several hypertori,
    each representing a connected component of m
```

Step1:
c = matrix_to_list(m)
c = flatten(c)
# we first remove the constant term
if d == 0: return m
# d shows how many connected components should m have
d = gcd(c)
res = [[]]
# each loop below will add a connected component of m to res
for i in range(1, int(d) + 1):
    temp = []
    for j in c:
        temp.append(j)
    while cons > 1: cons = cons - 1
    while cons <= 0: cons = cons + 1
    temp.append(cons)
    res.append(temp)
res.pop(0)
def charact_part(m):
    for m being a toric arrangement in matrix form
    remove the last column
    return the matrix of characteristics
    TEST:
sage: m = matrix(QQ, 2, 4, [1, 2, 3, 1, 0, 2, 3, 1])
sage: m
1 2 3 1
0 2 3 1
sage: c = charact_part(m)
sage: c
1 2 3
def matrix_append_row(m1, m2):
    '''
    NOTE: This is a helper function for poset_of_layer function
    INPUT: two matrices, m1 and m2 have the same number of columns,
    but m2 is restricted to have only one row (representing a torus)
    OUTPUT: return a matrix that append m2 as the last row of m1,
    remaining m1, m2 unchanged
    TEST:
    sage: m1 = matrix(QQ,2,4,[1,2,3,1,4,5,6,1])
    sage: m2 = matrix(QQ,1,4,[7,8,9,1])
    sage: m1
    [1 2 3 1]
    [4 5 6 1]
    sage: m2
    [7 8 9 1]
    sage: m3 = matrix_append_row(m1,m2)
    sage: m3
    [1 2 3 1]
    [4 5 6 1]
    [7 8 9 1]
    sage: m1
    [1 2 3 1]
    [4 5 6 1]
    '''
    l = []
    for i in range(0, m2.ncols()):
        l.append(m2[0][i])
    return matrix(m1.rows() + [l])

def matrix_erase_rows(m, k):
    '''
    INPUT: m, a matrix; k, number of rows to remain
    OUTPUT: the matrix obtained by erasing rows after the kth row
    TEST:
    sage: m = matrix(QQ,3,3,[1,0,0,1,0,0,0,0,0])
    sage: m
    [1 0 0]
sage: k = matrix_erase_rows(m, 2)
sage: k
\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix}
\]

l = []
for i in range(k):
    for j in m[i]:
        l.append(j)
return matrix(QQ, k, m.ncols(), l)

def matrix_append_column(m, v):
    l = []
    for i in range(m.nrows()):
        for j in m[i]:
            l.append(j)
l.append(v[i])
res = matrix(QQ, m.nrows(), m.ncols() + 1, l)
return res

def matrices_from_nested_list(L):
    INPUT: list of list of list
    OUTPUT: take one connected component in each torus in the arrangement, return the list of all possible matrices
    TEST:
    sage: L = [[[1,1,1],[1,1,1/2]],[[0,1,1],[0,1,1/2]]]
sage: L
    [[[1, 1, 1], [1, 1, 1/2]], [[0, 1, 1], [0, 1, 1/2]]]
sage: M = matrices_from_nested_list(L)
```python
sage: M
[
[1 1 1] [ 1 1 1] [ 1 1 1/2] [ 1 1 1/2]
[0 1 1], [ 0 1 1/2], [ 0 1 1], [ 0 1 1/2]
]
''

if (len(L) == 1):
    lm = []
    for i in L[0]:
        lm.append(matrix(i))
    return lm
else:
    fr = L.pop(len(L) - 1)
    lr = matrices_from_nested_list(L)
    res = []
    for m in lr:
        for r in fr:
            #r is already a list
            temp = matrix(m.rows() + [r])
            res.append(temp)
    return res

def list_to_matrix(l):
    '''
since we will be switch from list to matrix a lot,
I implemented it for convenience

TEST:
sage: l = [[1,2,3],[4,5,6]]
sage: m = list_to_matrix(l)
sage: m
[1 2 3]
[4 5 6]
'''
    f = flatten(l)
    m = matrix(QQ, len(l), len(l[0]), f)
    return m

def matrix_to_list(m):
    '''
since we will be switch from matrix to list a lot,
I implemented it for convenience

TEST:
sage: m = matrix(QQ,2,4,[1,2,3,1,0,2,3,1])
```

sage: m
[[1 2 3 1]
[0 2 3 1]]
sage: l = matrix_to_list(m)
sage: l
[[[1, 2, 3, 1], [0, 2, 3, 1]]
]
l = [[]]
for i in m:
    temp = []
    for j in range(0, m.ncols()):
        temp.append(i[j])
    l.append(temp)
l.pop(0)
return l

LISTING 1. Sage Code

7.3. Sample Usage.

sage: m=matrix(4,3,[1,-1,1,1,1,2,0,1,0,2,1])
sage: check_qualification(m)
True
sage: T=ToricArrangement(m)
sage: P = T.poset_of_layers
sage: P
Finite poset containing 11 elements
sage: T.characteristic_polynomial()
q^2 - 6*q + 8
sage: T.f_polynomial()
8*q^2 + 12*q + 4
sage: T.h_polynomial()
4*q + 4
sage: T.flag_f_polynomial()
48*q0*q1*q2 + 24*q0*q1 + 24*q0*q2 + 24*q1*q2 + 4*q0 + 12*q1 + 8*q2
sage: T.flag_h_polynomial()
8*q0*q1 + 12*q0*q2 + 4*q1*q2 + 4*q0 + 12*q1 + 8*q2
sage: T.cd_index()
8*c*d + 4*d*c
sage: T.cd_index_new_alg()
8*c*d + 4*d*c

LISTING 2. Sample Usage
8. Data and Discoveries

8.1. Data Collection I. The following is a collection of data with respect to Coordinate Toric Arrangement.

<table>
<thead>
<tr>
<th>Toric Arrangement</th>
<th>Char-Poly</th>
<th>f-poly</th>
<th>h-poly</th>
<th>flag f-poly</th>
<th>flag h-poly</th>
<th>cd-index</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \begin{pmatrix} 2 &amp; 0 &amp; 1 \ 0 &amp; 2 &amp; 1 \end{pmatrix} )</td>
<td>( q^2 - 4q + 4 )</td>
<td>( 4q^2 + 8q + 4 )</td>
<td>4</td>
<td>( 32q_0q_1q_2 + 16q_0q_2 + 16q_1q_2 + 4q_0 + 8q_1 + 4q_2 )</td>
<td>( 4q_0q_1 + 8q_0q_2 + 4q_1q_2 + 4q_0 + 8q_1 + 4q_2 )</td>
<td>( 4cd + 4dc )</td>
</tr>
<tr>
<td>( \begin{pmatrix} 3 &amp; 0 &amp; 1 \ 0 &amp; 3 &amp; 1 \end{pmatrix} )</td>
<td>( q^2 - 6q + 9 )</td>
<td>( 9q^2 + 18q + 9 )</td>
<td>9</td>
<td>( 72q_0q_1q_2 + 36q_0q_1 + 36q_0q_2 + 36q_1q_2 + 9q_0 + 18q_1 + 9q_2 )</td>
<td>( 9q_0q_1 + 18q_0q_2 + 9q_1q_2 + 9q_0 + 18q_1 + 9q_2 )</td>
<td>( 9cd + 9dc )</td>
</tr>
<tr>
<td>( \begin{pmatrix} 4 &amp; 0 &amp; 1 \ 0 &amp; 4 &amp; 1 \end{pmatrix} )</td>
<td>( q^2 - 8q + 16 )</td>
<td>( 16q^2 + 32q + 16 )</td>
<td>16</td>
<td>( 128q_0q_1q_2 + 64q_0q_1 + 64q_0q_2 + 64q_1q_2 + 16q_0 + 32q_1 + 16q_2 )</td>
<td>( 16q_0q_1 + 32q_0q_2 + 16q_1q_2 + 16q_0 + 32q_1 + 16q_2 )</td>
<td>( 16cd + 16dc )</td>
</tr>
<tr>
<td>( \begin{pmatrix} 5 &amp; 0 &amp; 1 \ 0 &amp; 5 &amp; 1 \end{pmatrix} )</td>
<td>( q^2 - 10q + 25 )</td>
<td>( 25q^2 + 50q + 25 )</td>
<td>25</td>
<td>( 200q_0q_1q_2 + 100q_0q_1 + 100q_0q_2 + 100q_1q_2 + 25q_0 + 50q_1 + 25q_2 )</td>
<td>( 25q_0q_1 + 50q_0q_2 + 25q_1q_2 + 25q_0 + 50q_1 + 25q_2 )</td>
<td>( 25cd + 25dc )</td>
</tr>
<tr>
<td>( \begin{pmatrix} 6 &amp; 0 &amp; 1 \ 0 &amp; 6 &amp; 1 \end{pmatrix} )</td>
<td>( q^2 - 12q + 36 )</td>
<td>( 36q^2 + 72q + 36 )</td>
<td>36</td>
<td>( 288q_0q_1q_2 + 144q_0q_1 + 144q_0q_2 + 144q_1q_2 + 36q_0 + 72q_1 + 36q_2 )</td>
<td>( 36q_0q_1 + 72q_0q_2 + 36q_1q_2 + 36q_0 + 72q_1 + 36q_2 )</td>
<td>( 36cd + 36dc )</td>
</tr>
<tr>
<td>Toric Arrgement</td>
<td>Char-Poly</td>
<td>f-poly</td>
<td>h-poly</td>
<td>flag f-poly</td>
<td>flag h-poly</td>
<td>cd-index</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>--------</td>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td>$\begin{pmatrix} 2 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 2 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 2 &amp; 1 \end{pmatrix}$</td>
<td>$q^3-6q^2+12q-8$</td>
<td>$8q^3 + 24q^2 + 24q + 8$</td>
<td>8</td>
<td>$384q_0q_1q_2q_3 + 192q_0q_1q_2 + 192q_0q_1q_3 + 192q_0q_2q_3 + 48q_0q_1 + 96q_0q_2 + 96q_1q_2 + 64q_0q_3 + 96q_1q_3 + 48q_2q_3 + 8q_0 + 24q_1 + 24q_2 + 8q_3$</td>
<td>$8q_0q_1q_2 + 24q_0q_1q_3 + 24q_0q_2q_3 + 8q_1q_2q_3 + 16q_0q_1 + 64q_0q_2 + 48q_1q_2 + 48q_0q_3 + 64q_1q_3 + 16q_2q_3 + 8q_0 + 24q_1 + 24q_2 + 8q_3$</td>
<td>$32d^2 + 8c^2d + 16cdc + 8d^2$</td>
</tr>
<tr>
<td>$\begin{pmatrix} 3 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 3 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 3 &amp; 1 \end{pmatrix}$</td>
<td>$q^3-9q^2+27q-27$</td>
<td>$27q^3 + 81q^2 + 81q + 27$</td>
<td>27</td>
<td>$1296q_0q_1q_2q_3 + 648q_0q_1q_2 + 648q_0q_1q_3 + 648q_0q_2q_3 + 648q_1q_2q_3 + 64q_0q_1 + 324q_0q_2 + 324q_1q_2 + 162q_1q_3 + 216q_0q_3 + 324q_1q_3 + 162q_2q_3 + 27q_0 + 81q_1 + 81q_2 + 27q_3$</td>
<td>$27q_0q_1q_2 + 81q_0q_1q_3 + 81q_0q_2q_3 + 27q_1q_2q_3 + 54q_0q_1 + 216q_0q_2 + 162q_1q_2 + 162q_0q_3 + 216q_1q_3 + 54q_2q_3 + 27q_0 + 81q_1 + 81q_2 + 27q_3$</td>
<td>$108d^2 + 27c^2d + 54cdc + 27dc^2$</td>
</tr>
<tr>
<td>$\begin{pmatrix} 4 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 4 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 4 &amp; 1 \end{pmatrix}$</td>
<td>$q^3-12q^2 + 48q - 64$</td>
<td>$64q^3 + 192q^2 + 192q + 64$</td>
<td>64</td>
<td>$3072q_0q_1q_2q_3 + 1536q_0q_1q_2 + 1536q_0q_1q_3 + 1536q_0q_2q_3 + 1536q_1q_2q_3 + 384q_0q_1 + 768q_0q_2 + 768q_1q_2 + 512q_0q_3 + 768q_1q_3 + 384q_2q_3 + 64q_0 + 192q_1 + 192q_2 + 64q_3$</td>
<td>$64q_0q_1q_2 + 192q_0q_1q_3 + 192q_0q_2q_3 + 128q_0q_1 + 512q_0q_2 + 384q_1q_2 + 384q_0q_3 + 512q_1q_3 + 128q_2q_3 + 64q_0 + 192q_1 + 192q_2 + 64q_3$</td>
<td>$256d^2 + 64c^2d + 128cdc + 64dc^2$</td>
</tr>
</tbody>
</table>
Table 3. I-3

<table>
<thead>
<tr>
<th>Toric Arrangement</th>
<th>Char-Poly</th>
<th>f-poly</th>
<th>h-poly</th>
<th>flag f-poly</th>
<th>flag h-poly</th>
<th>cd-index</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \begin{pmatrix} 5 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 5 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 5 &amp; 1 \end{pmatrix} )</td>
<td>( q^3 - 15q^2 + \frac{75q - 125}{75q^2 + 375q + 125} )</td>
<td>( 125q^3 + 375q^2 + 375q + 125 )</td>
<td>125</td>
<td>1250q_0q_1q_2q_3 + 375q_0q_1q_3 + 375q_0q_2q_3 + 125q_1q_2q_3 + 1000q_0q_2 + 750q_1q_2 + 750q_0q_3 + 1000q_1q_3 + 250q_2q_3 + 125q_0 + 375q_1 + 375q_2 + 125q_3</td>
<td>( 500d^2 ) + ( 125c^2d ) + ( 250cd ) + ( 125dc^2 )</td>
<td></td>
</tr>
<tr>
<td>( \begin{pmatrix} 6 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 6 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 6 &amp; 1 \end{pmatrix} )</td>
<td>( q^3 - 18q^2 + \frac{108q - 216}{648q^2 + 648q + 216} )</td>
<td>( 216q^3 + 648q^2 + 648q + 216 )</td>
<td>216</td>
<td>2160q_0q_1q_2q_3 + 648q_0q_1q_3 + 648q_0q_2q_3 + 315q_1q_2q_3 + 432q_0q_2 + 1296q_1q_2 + 1296q_0q_3 + 1728q_1q_3 + 432q_2q_3 + 216q_0 + 648q_1 + 648q_2 + 216q_3</td>
<td>( 864d^2 ) + ( 216c^2d ) + ( 432cd ) + ( 216dc^2 )</td>
<td></td>
</tr>
<tr>
<td>Toric Arrangement</td>
<td>Char-Poly</td>
<td>f-poly</td>
<td>h-poly</td>
<td>flag f-poly</td>
<td>flag h-poly</td>
<td>cd-index</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------</td>
<td>----------------</td>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
<td>-----------</td>
</tr>
<tr>
<td>( \begin{pmatrix} 2 &amp; 0 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 2 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 2 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 0 &amp; 2 &amp; 1 \end{pmatrix} )</td>
<td>( q^4 - 8q^3 + 24q^2 - 32q + 16 )</td>
<td>( 16q^4 + 64q^3 + 96q^2 + 64q + 16 )</td>
<td>16</td>
<td>6144q_0q_1q_2q_3q_4 + ( 3072q_0q_1q_2q_3 ) + ( 3072q_0q_1q_2q_4 ) + ( 3072q_0q_1q_3q_4 ) + ( 3072q_0q_2q_3q_4 ) + ( 3072q_1q_2q_3q_4 ) + ( 768q_0q_1q_2 + 1536q_0q_1q_3 + 1536q_0q_2q_3 + 1536q_1q_2q_3 + 1024q_0q_1q_4 + 1536q_0q_2q_4 + 1536q_1q_2q_4 + 1024q_0q_3q_4 + 1536q_1q_3q_4 + 768q_2q_3q_4 + 128q_0q_1 + 384q_0q_2 + 384q_1q_2 + 768q_1q_3 + 384q_2q_3 + 256q_0q_4 + 384q_2q_4 + 16q_0 + 64q_1 + 96q_2 + 64q_3 + 16q_4 ) + ( 16q_0q_1q_2q_3 ) + ( 64q_0q_1q_2q_4 ) + ( 96q_0q_1q_3q_4 ) + ( 64q_0q_2q_3q_4 ) + ( 16q_1q_2q_3q_4 + 48q_0q_1q_2 + 272q_0q_1q_3 + 432q_0q_2q_3 + 224q_1q_2q_3 + 224q_0q_1q_4 + 640q_0q_2q_4 + 432q_0q_3q_4 + 272q_1q_3q_4 + 48q_2q_3q_4 + 48q_0q_1 ) + ( 272q_0q_2 + 224q_1q_2 ) + ( 432q_0q_3 + 640q_1q_3 ) + ( 224q_2q_3 + 224q_0q_4 ) + ( 432q_1q_4 + 272q_2q_4 ) + ( 48q_3q_4 + 16q_0 + 64q_1 + 96q_2 + 64q_3 + 16q_4 ) +</td>
<td>160cd^2 + 192dcd + 160d^2c + 16c^3d + 48c^2dc + 48cd^2 + 16c^3d</td>
<td></td>
</tr>
<tr>
<td>Toric Arrangement</td>
<td>Char-Poly</td>
<td>f-poly</td>
<td>h-poly</td>
<td>flag f-poly</td>
<td>flag h-poly</td>
<td>cd-index</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>--------</td>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td>(\begin{pmatrix} 3 &amp; 0 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 3 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 3 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 0 &amp; 3 &amp; 1 \end{pmatrix})</td>
<td>(q^4 - 12q^3 + 54q^2 - 108q + 81)</td>
<td>(81q^4 + 324q^3 + 486q^2 + 324q + 81)</td>
<td>81</td>
<td>31104q_0q_1q_2q_3q_4</td>
<td>81q_0q_1q_2q_3</td>
<td>810cd^2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15552q_0q_1q_2q_3</td>
<td>324q_0q_1q_2q_4</td>
<td>972dcd</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15552q_0q_1q_2q_4</td>
<td>486q_0q_1q_3q_4</td>
<td>810d^2c</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15552q_0q_1q_3q_4</td>
<td>324q_0q_2q_3q_4</td>
<td>81c^3d</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15552q_0q_2q_3q_4</td>
<td>1377q_0q_1q_3</td>
<td>243^2dc</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15552q_0q_2q_4q_4</td>
<td>2187q_0q_2q_3</td>
<td>243dcd^2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>818q_0q_1q_2q_4</td>
<td>1134q_0q_1q_3q_4</td>
<td>81dc^3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7776q_0q_1q_2q_3</td>
<td>1134q_1q_2q_3</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7776q_0q_2q_3</td>
<td>1134q_0q_1q_4</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7776q_1q_2q_3</td>
<td>3240q_0q_2q_4</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5184q_0q_1q_4</td>
<td>2187q_1q_2q_4</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7776q_0q_2q_4</td>
<td>1134q_0q_3q_4</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7776q_1q_2q_4</td>
<td>1377q_1q_3q_4 + 243q_2q_3q_4</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5184q_0q_3q_4</td>
<td>243q_0q_1 + 1377q_0q_2</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7776q_1q_3q_4</td>
<td>1134q_1q_2 + 2187q_0q_3</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3888q_2q_3q_4 + 648q_0q_1</td>
<td>3240q_1q_3 + 1134q_2q_3</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1944q_0q_2 + 1944q_1q_2</td>
<td>1134q_0q_4 + 2187q_1q_4</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2592q_0q_3 + 3888q_1q_3</td>
<td>1377q_2q_4 + 243q_3q_4</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1944q_2q_3 + 1296q_0q_4</td>
<td>81q_0 + 324q_1 + 486q_2</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2592q_1q_4 + 1944q_2q_4</td>
<td>324q_3 + 81q_4</td>
<td>+</td>
</tr>
<tr>
<td>Toric Arrangement</td>
<td>Char-Poly</td>
<td>f-poly</td>
<td>h-poly</td>
<td>flag f-poly</td>
<td>flag h-poly</td>
<td>cd-index</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------</td>
<td>--------</td>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>( 2 0 0 0 0 0 )</td>
<td>( q^5 )</td>
<td>32 ( q^5 + 160q^4 + 61440 )</td>
<td>32</td>
<td>122880q_0q_1q_2q_3q_4q_5 + 320q_0q_1q_2q_3q_4q_5 + 160q_0q_1q_2q_3q_4q_5 + 20480d^3 + 576d^2d^2 + 1280cdcd</td>
<td></td>
<td>204d3 + 1280cdcd</td>
</tr>
<tr>
<td>( 0 2 0 0 0 0 )</td>
<td>( 10q^4 + 40q^3 + 80q^2 + 80q )</td>
<td></td>
<td></td>
<td>61440q_0q_1q_2q_3q_4 + 320q_0q_1q_2q_3q_4 + 160q_0q_1q_2q_3q_4 + 1280q_0q_1q_2q_3q_4 + 320q_0q_1q_2q_3q_4 + 1280q_0q_1q_2q_3q_4 + 576d^2d^2 + 128c^3d + 192d^2d^2+</td>
<td>192c^3d^2+</td>
<td>192c^3d^2+</td>
</tr>
<tr>
<td>( 0 0 2 0 0 0 )</td>
<td>( 2048q_0q_1q_2q_3q_4q_5 + 30720q_0q_1q_2q_3q_4q_5 + 30720q_0q_1q_2q_3q_4q_5 + 30720q_0q_1q_2q_3q_4q_5 + 30720q_0q_1q_2q_3q_4q_5 + 800q_0q_1q_2q_3q_4q_5 + 928q_0q_1q_2q_3q_4q_5 + 928q_0q_1q_2q_3q_4q_5 + 192c^3d^3+ + 32cd^3</td>
<td>192c^3d^3+</td>
<td>192c^3d^3+</td>
<td>32cd^3+</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The data above support our formulas for characteristic polynomial, f-polynomial, h-polynomial, reduced flag f-polynomial of Coordinate Toric Arrangement in Section 5. Here are some other observations that satisfy our Lemma:

**Proposition 8.1.** The coefficient for $cc...cd$ and $dc...cc$ are given by: $f_n = f_0 = k^n$ (f-polynomial is symmetric). (Lemma 5.5)

**Proposition 8.2.** The coefficient of $cc...cdc...cc$ where $d$ is at the $i$th/$(n-i)$th position is given by: $f_i - f_{i+1} + f_{i+2} - ... + (-1)^{n-i} f_n$. (Lemma 5.5)

**Proposition 8.3.** If we order the cd-strings with only one d by moving d one unit towards right each time, the coefficients satisfies the nth level of Pascal's Triangle. (This can be proved by Lemma 5.5)

**Conjecture 8.4.** The cd-index has the format $k^n \ast (\psi(cd))$, where $\psi(cd)$ is a cd-polynomial that does not depend on $k$, but only depend on $n$ (the dimension of the torus space).

**Conjecture 8.5.** The symmetry of the coefficients of cd-index: given a cd string, the coefficient of itself is the same as the coefficient of its reverse string. For example, the coefficient of $ccdc$ is the same as the coefficient of $cdcc$. 
8.2. **Data Collection II.** The following is a collection of data with adding hypertori to our Coordinate Toric Arrangement (when \( k = 2 \)).

<table>
<thead>
<tr>
<th>Toric Arrangement</th>
<th>Char-Poly</th>
<th>f-poly</th>
<th>h-poly</th>
<th>flag f-poly</th>
<th>flag h-poly</th>
<th>cd-index</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
2 & 0 & 1 \\
0 & 2 & 1 \\
1 & -1 & 1
\end{bmatrix}
\] | \( q^2 - 5q + 6 \) | \( 6q^2 + 10q + 4 \) | \( 2q + 4 \) | \( 40q_0q_1q_2 + 20q_0q_1 + 20q_0q_2 + 20q_1q_2 + 4q_0 + 10q_1 + 6q_2 \) | \( 6q_0q_1 + 10q_0q_2 + 4q_1q_2 + 4q_0 + 10q_1 + 6q_2 \) | \( 6cd + 4dc \) |
| \[
\begin{bmatrix}
2 & 0 & 1 \\
0 & 2 & 1 \\
2 & -1 & 1
\end{bmatrix}
\] | \( q^2 - 5q + 8 \) | \( 8q^2 + 14q + 6 \) | \( 2q + 6 \) | \( 56q_0q_1q_2 + 28q_0q_1 + 28q_0q_2 + 28q_1q_2 + 6q_0 + 14q_1 + 8q_2 \) | \( 8q_0q_1 + 14q_0q_2 + 6q_1q_2 + 6q_0 + 14q_1 + 8q_2 \) | \( 8cd + 6dc \) |
| \[
\begin{bmatrix}
2 & 0 & 1 \\
0 & 2 & 1 \\
1 & 1 & 1 \\
1 & -1 & 1
\end{bmatrix}
\] | \( q^2 - 6q + 8 \) | \( 8q^2 + 12q + 4 \) | \( 4q + 4 \) | \( 48q_0q_1q_2 + 24q_0q_1 + 24q_0q_2 + 24q_1q_2 + 4q_0 + 12q_1 + 8q_2 \) | \( 8q_0q_1 + 12q_0q_2 + 4q_1q_2 + 4q_0 + 12q_1 + 8q_2 \) | \( 8cd + 4dc \) |
| \[
\begin{bmatrix}
2 & 0 & 1 \\
0 & 2 & 1 \\
1 & 1 & 1 \\
1 & -1 & 1 \\
2 & -1 & 1
\end{bmatrix}
\] | \( q^2 - 7q + 14 \) | \( 14q^2 + 22q + 8 \) | \( 6q + 8 \) | \( 88q_0q_1q_2 + 44q_0q_1 + 44q_0q_2 + 44q_1q_2 + 8q_0 + 22q_1 + 14q_2 \) | \( 14q_0q_1 + 22q_0q_2 + 8q_1q_2 + 8q_0 + 22q_1 + 14q_2 \) | \( 14cd + 8dc \) |
<table>
<thead>
<tr>
<th>Toric Arrangement</th>
<th>Char-Poly</th>
<th>f-poly</th>
<th>h-poly</th>
<th>flag f-poly</th>
<th>flag h-poly</th>
<th>cd-index</th>
</tr>
</thead>
<tbody>
<tr>
<td>([2 \ 0 \ 0 \ 1 \ 0 \ 2 \ 0 \ 1 \ 0 \ 0 \ 2 \ 1 \ 1 \ -1 \ 0 \ 1] )</td>
<td>(q^3 - 7q^2 + 16q - 12)</td>
<td>(12q^3 + 32q^2 + 28q + 8)</td>
<td>(4q + 8)</td>
<td>(480q_0q_1q_2q_3 + 240q_0q_1q_2 + 240q_0q_1q_3 + 240q_0q_2q_3 + 56q_0q_2 + 120q_0q_2 + 120q_1q_2 + 80q_0q_3 + 120q_1q_3 + 64q_2q_3 + 8q_0 + 28q_1 + 32q_2 + 12q_3)</td>
<td>(12q_0q_1q_2 + 32q_0q_1q_3 + 28q_0q_2q_3 + 8q_1q_2q_3 + 20q_0q_1 + 80q_0q_2 + 60q_1q_2 + 60q_0q_3 + 80q_1q_3 + 20q_2q_3 + 8q_0 + 28q_1 + 32q_2 + 12q_3)</td>
<td>(40d^2 + 12c^2d + 20cd + 8dc^2)</td>
</tr>
<tr>
<td>([2 \ 0 \ 0 \ 1 \ 0 \ 2 \ 0 \ 1 \ 0 \ 0 \ 2 \ 1 \ 2 \ -1 \ 0 \ 1] )</td>
<td>(q^3 - 7q^2 + 18q - 16)</td>
<td>(16q^3 + 44q^2 + 40q + 12)</td>
<td>(4q + 12)</td>
<td>(672q_0q_1q_2q_3 + 336q_0q_1q_2 + 336q_0q_1q_3 + 336q_0q_2q_3 + 80q_0q_1 + 168q_0q_2 + 168q_1q_2 + 112q_0q_3 + 168q_1q_3 + 88q_2q_3 + 12q_0 + 40q_1 + 44q_2 + 16q_3)</td>
<td>(16q_0q_1q_2 + 44q_0q_1q_3 + 40q_0q_2q_3 + 12q_1q_2q_3 + 28q_0q_1 + 112q_0q_2 + 84q_1q_2 + 84q_0q_3 + 112q_1q_3 + 28q_2q_3 + 12q_0 + 40q_1 + 44q_2 + 16q_3)</td>
<td>(56d^2 + 16c^2d + 28cd + 12dc^2)</td>
</tr>
<tr>
<td>([2 \ 0 \ 0 \ 1 \ 0 \ 2 \ 0 \ 1 \ 0 \ 0 \ 2 \ 1 \ 1 \ -1 \ 0 \ 1] )</td>
<td>(q^3 - 8q^2 + 20q - 16)</td>
<td>(16q^3 + 40q^2 + 32q + 8)</td>
<td>(8q + 8)</td>
<td>(576q_0q_1q_2q_3 + 288q_0q_1q_2 + 288q_0q_1q_3 + 288q_0q_2q_3 + 64q_0q_1 + 144q_0q_2 + 144q_1q_2 + 96q_0q_3 + 144q_1q_3 + 80q_2q_3 + 8q_0 + 32q_1 + 40q_2 + 16q_3)</td>
<td>(16q_0q_1q_2 + 40q_0q_1q_3 + 32q_0q_2q_3 + 8q_1q_2q_3 + 24q_0q_1 + 96q_0q_2 + 72q_1q_2 + 72q_0q_3 + 96q_1q_3 + 24q_2q_3 + 8q_0 + 32q_1 + 40q_2 + 16q_3)</td>
<td>(48d^2 + 16c^2d + 24cd + 8dc^2)</td>
</tr>
<tr>
<td>Toric Arrangement</td>
<td>Char-Poly</td>
<td>f-poly</td>
<td>h-poly</td>
<td>flag f-poly</td>
<td>flag h-poly</td>
<td>cd-index</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------</td>
<td>--------</td>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td>[ (2 \ 0 \ 0 \ 1) ] [ (0 \ 0 \ 2 \ 1) ] [ (1 \ -1 \ 0 \ 1) ] [ (1 \ 1 \ 0 \ 1) ] [ (1 \ 0 \ 1 \ 1) ] [ (1 \ 0 \ -1 \ 1) ]</td>
<td>[ q^3 - 9q^2 + 26q - 24 ]</td>
<td>[ 24q^3 + 56q^2 + 24q + 8 ]</td>
<td>[ 16q + 8 ]</td>
<td>[ 736q_0q_1q_2q_3 + 368q_0q_1q_2q_3 + 368q_0q_2q_3 + 368q_1q_2q_3 + 80q_0q_1 + 184q_0q_2 + 184q_1q_2 + 120q_0q_3 + 184q_1q_3 + 112q_2q_3 + 8q_0 + 40q_1 + 56q_2 + 24q_3 ]</td>
<td>[ 24q_0q_1q_2 + 56q_0q_1q_3 + 40q_0q_2q_3 + 32q_0q_1 + 120q_0q_2 + 88q_1q_2 + 88q_0q_3 + 120q_1q_3 + 32q_2q_3 + 8q_0 + 40q_1 + 56q_2 + 24q_3 ]</td>
<td>[ 56d^2 + 24c^2d + 32dc + 8dc^2 ]</td>
</tr>
<tr>
<td>[ (2 \ 0 \ 0 \ 1) ] [ (0 \ 0 \ 2 \ 1) ] [ (1 \ -1 \ 0 \ 1) ] [ (1 \ 1 \ 0 \ 1) ] [ (1 \ 0 \ 1 \ 1) ] [ (1 \ 0 \ -1 \ 1) ]</td>
<td>[ q^3 - 10q^2 + 32q - 32 ]</td>
<td>[ 32q^3 + 72q^2 + 48q + 8 ]</td>
<td>[ 24q + 8 ]</td>
<td>[ 896q_0q_1q_2q_3 + 448q_0q_1q_2q_3 + 448q_0q_2q_3 + 448q_1q_2q_3 + 96q_0q_1 + 224q_0q_2 + 224q_1q_2 + 144q_0q_3 + 224q_1q_3 + 144q_2q_3 + 8q_0 + 48q_1 + 72q_2 + 32q_3 ]</td>
<td>[ 32q_0q_1q_2 + 72q_0q_1q_3 + 48q_0q_2q_3 + 8q_1q_2q_3 + 40q_0q_1 + 144q_0q_2 + 104q_1q_2 + 104q_0q_3 + 144q_1q_3 + 40q_2q_3 + 8q_0 + 48q_1 + 72q_2 + 32q_3 ]</td>
<td>[ 64d^2 + 32c^2d + 40dc + 8dc^2 ]</td>
</tr>
<tr>
<td>Toric Arrangement</td>
<td>Char-Poly</td>
<td>f-poly</td>
<td>h-poly</td>
<td>flag f-poly</td>
<td>flag h-poly</td>
<td>cd-index</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------</td>
<td>--------</td>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td>$\begin{pmatrix} 2 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 2 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 2 &amp; 1 \ 1 &amp; -1 &amp; 0 &amp; 1 \ 1 &amp; 1 &amp; 0 &amp; 1 \ 1 &amp; 0 &amp; 1 &amp; 1 \ 1 &amp; 0 &amp; -1 &amp; 1 \ 0 &amp; 1 &amp; 1 &amp; 1 \end{pmatrix}$</td>
<td>$q^3 - 11q^2 + 38q - 40$</td>
<td>$40q^3 + 84q^2 + 52q + 8$</td>
<td>$4q^2 + 28q + 8$</td>
<td>1024$q_0q_1q_2q_3 + 512q_0q_1q_2 + 512q_0q_1q_3 + 512q_0q_2q_3 + 512q_1q_2q_3 + 104q_1q_2 + 256q_0q_3 + 256q_1q_3 + 168q_2q_3 + 8q_0 + 52q_1 + 84q_2 + 40q_3$</td>
<td>$40q_0q_1q_2 + 84q_0q_1q_3 + 52q_0q_2q_3 + 8q_1q_2q_3 + 44q_0q_1 + 164q_0q_2 + 120q_1q_2 + 120q_0q_3 + 164q_1q_3 + 44q_2q_3 + 8q_0 + 52q_1 + 84q_2 + 40q_3$</td>
<td>$72d^2 + 40c^2d + 44cd + 8d^2$</td>
</tr>
<tr>
<td>$\begin{pmatrix} 2 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 2 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 2 &amp; 1 \ 1 &amp; -1 &amp; 0 &amp; 1 \ 1 &amp; 1 &amp; 0 &amp; 1 \ 1 &amp; 0 &amp; 1 &amp; 1 \ 1 &amp; 0 &amp; -1 &amp; 1 \ 0 &amp; 1 &amp; 1 &amp; 1 \end{pmatrix}$</td>
<td>$q^3 - 12q^2 + 44q - 48$</td>
<td>$48q^3 + 96q^2 + 56q + 8$</td>
<td>$8q^2 + 32q + 8$</td>
<td>1152$q_0q_1q_2q_3 + 576q_0q_1q_2 + 576q_0q_1q_3 + 576q_0q_2q_3 + 576q_1q_2q_3 + 112q_0q_1 + 288q_0q_2 + 288q_1q_2 + 192q_0q_3 + 288q_1q_3 + 192q_2q_3 + 8q_0 + 56q_1 + 96q_2 + 48q_3$</td>
<td>$48q_0q_1q_2 + 96q_0q_1q_3 + 56q_0q_2q_3 + 8q_1q_2q_3 + 48q_0q_1 + 184q_0q_2 + 136q_1q_2 + 136q_0q_3 + 184q_1q_3 + 48q_2q_3 + 8q_0 + 56q_1 + 96q_2 + 48q_3$</td>
<td>$80d^2 + 48c^2d + 48cd + 8d^2$</td>
</tr>
<tr>
<td>Toric Arrangement</td>
<td>Char-Poly</td>
<td>f-poly</td>
<td>h-poly</td>
<td>flag f-poly</td>
<td>flag h-poly</td>
<td>cd-index</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>--------</td>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td>( \begin{pmatrix} 2 &amp; 0 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 2 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 2 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 0 &amp; 2 &amp; 1 \ 1 &amp; -1 &amp; 0 &amp; 0 &amp; 1 \end{pmatrix} )</td>
<td>( q^4 - 9q^3 + 30q^2 - 44q + 24 )</td>
<td>( 24q^3 + 88q^2 + 72q + 16 )</td>
<td>( 8q + 16 )</td>
<td>( 24q_0q_1q_2q_3q_4 + )</td>
<td>( 24q_0q_1q_2q_3 + )</td>
<td>( 208cd^2 + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 8q_0q_1q_2q_4 + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 120q_0q_1q_3q_4 + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 72q_0q_2q_3q_4 + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 16q_1q_2q_3q_4 + 64q_0q_1q_2 + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 352q_0q_1q_3 + 544q_0q_2q_3 + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 280q_1q_2q_3 + 288q_0q_1q_4 + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 800q_0q_2q_4 + 536q_1q_2q_4 + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 272q_0q_3q_4 + 328q_1q_3q_4 + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 56q_2q_3q_4 + 56q_0q_1 + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 328q_0q_2 + 272q_1q_2 + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 536q_0q_3 + 800q_1q_3 + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 288q_2q_3 + 280q_0q_4 + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 544q_1q_4 + 352q_2q_4 + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 64q_3q_4 + 16q_0 + 72q_1 + )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 120q_2 + 88q_3 + 24q_4 )</td>
</tr>
<tr>
<td>Toric Arrangement</td>
<td>Char-Poly</td>
<td>f-poly</td>
<td>h-poly</td>
<td>flag f-poly</td>
<td>flag h-poly</td>
<td>cd-index</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------</td>
<td>--------</td>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>( 2 \ 0 \ 0 \ 0 \ 1 )</td>
<td>( q^4 - 10q^3 + 36q^2 - 56q + 32 )</td>
<td>( 32q^3 + 112q^2 + 144q + 80 )</td>
<td>( 16q + 16 )</td>
<td>( 9216q_0q_1q_2q_3q_4 )</td>
<td>( 32q_0q_1q_2q_3 )</td>
<td>( 256cd^2 )</td>
</tr>
<tr>
<td>( 0 \ 2 \ 0 \ 0 \ 1 )</td>
<td>( 4608q_0q_1q_2q_3 )</td>
<td>( + )</td>
<td>( + )</td>
<td>( 112q_0q_1q_2q_4 )</td>
<td>( + )</td>
<td>( 288dcd )</td>
</tr>
<tr>
<td>( 0 \ 0 \ 2 \ 0 \ 1 )</td>
<td>( 4608q_0q_1q_2q_4 )</td>
<td>( + )</td>
<td>( + )</td>
<td>( 144q_0q_1q_3q_4 )</td>
<td>( + )</td>
<td>( 224d^2c )</td>
</tr>
<tr>
<td>( 0 \ 0 \ 0 \ 2 \ 1 )</td>
<td>( 4608q_0q_1q_3q_4 )</td>
<td>( + )</td>
<td>( + )</td>
<td>( 80q_0q_2q_3q_4 )</td>
<td>( + )</td>
<td>( 32c^3d )</td>
</tr>
<tr>
<td>( 1 \ -1 \ 0 \ 0 \ 1 )</td>
<td>( 4608q_0q_2q_3q_4 )</td>
<td>( + )</td>
<td>( + )</td>
<td>( 16q_1q_2q_3q_4 + 80q_0q_1q_2 + 432q_0q_1q_3 + 656q_0q_2q_3 + 336q_1q_2q_3 + 352q_0q_1q_4 + 960q_0q_2q_4 + 640q_1q_2q_4 + 320q_0q_3q_4 + 384q_1q_3q_4 + 64q_2q_3q_4 + 64q_0q_1 + 384q_0q_2 + 320q_1q_2 + 640q_0q_3 + 960q_1q_3 + 352q_2q_3 + 336q_0q_4 + 656q_1q_4 + 432q_2q_4 + 80q_3q_4 + 16q_0 + 80q_1 + 144q_2 + 112q_3 + 32q_4 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 1 \ 1 \ 0 \ 0 \ 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Toric Arrangement</td>
<td>Char-Poly</td>
<td>f-poly</td>
<td>h-poly</td>
<td>flag f-poly</td>
<td>flag h-poly</td>
<td>cd-index</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>--------</td>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td>( \begin{pmatrix} 2 &amp; 0 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 2 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 2 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 0 &amp; 2 &amp; 1 \ 1 &amp; -1 &amp; 0 &amp; 0 &amp; 1 \ 1 &amp; 1 &amp; 0 &amp; 0 &amp; 1 \ 1 &amp; 0 &amp; 1 &amp; 0 &amp; 1 \end{pmatrix} )</td>
<td>( q^4 - 11q^3 + 44q^2 - 76q + 48 )</td>
<td>( 48q^4 + 160q^3 + 192q^2 + 96q + 16 )</td>
<td>( 32q^4 + 16 )</td>
<td>( 11776q_0q_1q_2q_3q_4 ) +</td>
<td>( 48q_0q_1q_2q_3q_4 + 160q_0q_1q_2q_3q_4 + 192q_0q_1q_2q_3q_4 + 96q_0q_2q_3q_4 + 336cd^2 + 352dcd + 272d^2c + 48c^3d + 112c^2dc + 80cd^2 + 16dc^3 )</td>
<td></td>
</tr>
<tr>
<td>Toric Arrangement</td>
<td>Char-Poly</td>
<td>f-poly</td>
<td>h-poly</td>
<td>flag f-poly</td>
<td>flag h-poly</td>
<td>cd-index</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------</td>
<td>----------------</td>
<td>----------------</td>
<td>------------------</td>
<td>------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>$\begin{pmatrix} 2 &amp; 0 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 2 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 2 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 0 &amp; 2 &amp; 1 \ 1 &amp; -1 &amp; 0 &amp; 0 &amp; 1 \ 1 &amp; 1 &amp; 0 &amp; 0 &amp; 1 \ 1 &amp; 0 &amp; 1 &amp; 0 &amp; 1 \ 1 &amp; 0 &amp; -1 &amp; 0 &amp; 1 \end{pmatrix}$</td>
<td>$q^4 - 12q^3 + 52q^2 - 96q + 64$</td>
<td>$64q^4 + 208q^3 + 240q^2 + 112q + 16$</td>
<td>$48q^4 + 16$</td>
<td>$14336q_0q_1q_2q_3q_4$</td>
<td>$16q_0q_1q_2q_3q_4 + 208q_0q_1q_2q_4 + 240q_0q_1q_3q_4 + 112q_0q_2q_3q_4</td>
<td>$64q_0q_1q_2q_3$</td>
</tr>
</tbody>
</table>
Conjecture 8.6. The Coordinate Toric Arrangement with $k = 2$ is our "minimum" under Regular Cell Complex Assumption, meaning that the coefficients of characteristic polynomial, $f$-polynomial, $h$-polynomial, reduced flag $f$-polynomial, reduced flag $h$-polynomial, and cd-index are the smallest in that dimension.

Conjecture 8.7. Adding hypertori to any toric arrangement will only increase the coefficients of reduced flag $f$-polynomial, reduced flag $h$-polynomial, and cd-index (or remain the same if adding a hypertorus parallel to a hypertorus in the original toric arrangement).
8.3. **Data Collection III.** Other collection of data.

<table>
<thead>
<tr>
<th>Toric Arrangement</th>
<th>Char-Poly</th>
<th>f-poly</th>
<th>h-poly</th>
<th>flag f-poly</th>
<th>flag h-poly</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \begin{pmatrix} 1 &amp; 0 &amp; 1 \ 0 &amp; 1 &amp; 1 \ 1 &amp; 0 &amp; \frac{1}{2} \ 0 &amp; 1 &amp; \frac{1}{2} \ 3 &amp; 2 &amp; 1 \end{pmatrix} )</td>
<td>( q^2 - 5q + 12 )</td>
<td>( 12q^2 + 22q + 10 )</td>
<td>( 2q + 10 )</td>
<td>( 88q_0q_1q_2 + 44q_0q_1 + 44q_0q_2 + 44q_1q_2 + 10q_0 + 22q_1 + 12q_2 )</td>
<td>( 12q_0q_1 + 22q_0q_2 + 10q_1q_2 + 10q_0 + 22q_1 + 12q_2 )</td>
</tr>
<tr>
<td>( \begin{pmatrix} 1 &amp; 0 &amp; 1 \ 0 &amp; 1 &amp; 1 \ 1 &amp; 0 &amp; \frac{1}{2} \ 0 &amp; 1 &amp; \frac{1}{2} \ 1 &amp; -1 &amp; 1 \ 1 &amp; 2 &amp; 1 \end{pmatrix} )</td>
<td>( q^2 - 6q + 12 )</td>
<td>( 12q^2 + 20q + 8 )</td>
<td>( 4q + 8 )</td>
<td>( 80q_0q_1q_2 + 40q_0q_1 + 40q_0q_2 + 40q_1q_2 + 8q_0 + 20q_1 + 12q_2 )</td>
<td>( 12q_0q_1 + 20q_0q_2 + 8q_1q_2 + 8q_0 + 20q_1 + 12q_2 )</td>
</tr>
<tr>
<td>( \begin{pmatrix} 1 &amp; 0 &amp; 1 \ 0 &amp; 1 &amp; 1 \ 0 &amp; 0 &amp; 1 \ 1 &amp; 0 &amp; \frac{1}{2} \ 0 &amp; 1 &amp; \frac{1}{2} \ 1 &amp; 1 &amp; 1 \end{pmatrix} )</td>
<td>( q^3 - 7q^2 + 18q - 16 )</td>
<td>( 16q^3 + 44q^2 + 36q + 8 )</td>
<td>(-4q^2 + 12q + 8)</td>
<td>( 576q_0q_1q_2q_3 + 288q_0q_1q_2 + 288q_0q_1q_3 + 288q_0q_2q_3 + 144q_0q_1q_2 + 144q_1q_2 + 88q_0q_1q_3 + 88q_2q_3 + 8q_0 + 36q_1 + 44q_2 + 16q_3 )</td>
<td>( 16q_0q_1q_2 + 44q_0q_1q_3 + 36q_0q_2q_3 + 8q_1q_2q_3 + 28q_0q_1 + 92q_0q_2 + 64q_1q_2 + 64q_0q_3 + 28q_2q_3 + 8q_0 + 36q_1 + 44q_2 + 16q_3 )</td>
</tr>
</tbody>
</table>
\begin{table}
\centering
\begin{tabular}{|c|c|c|c|c|c|}
\hline
Toric Arrangement & Char-Poly & f-poly & h-poly & flag f-poly & flag h-poly \\
\hline
\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & \frac{1}{2} \\ 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & \frac{1}{2} \\ 2 & 1 & 1 & 1 \end{pmatrix} & \begin{align*}
q^3 &- 7q^2 \\
18q - 20 &
\end{align*} & 20q^3 + 56q^2 + 48q + 12 & -4q^2 + 12q + 12 & 768q_0q_1q_2q_3 + 384q_0q_1q_2 + 384q_0q_2q_3 + 384q_1q_2q_3 + 96q_0q_1 + 192q_0q_2 + 192q_1q_2 + 120q_0q_3 + 192q_1q_3 + 112q_2q_3 + 12q_0 + 48q_1 + 56q_2 + 20q_3 & 20q_0q_1q_2 + 56q_0q_1q_3 + 48q_0q_2q_3 + 12q_1q_2q_3 + 36q_0q_1 + 124q_0q_2 + 88q_1q_2 + 88q_0q_3 + 124q_1q_3 + 36q_2q_3 + 12q_0 + 48q_1 + 56q_2 + 20q_3 \\
\hline
\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & \frac{1}{2} \\ 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & \frac{1}{2} \\ 3 & 1 & 1 & 1 \end{pmatrix} & \begin{align*}
q^3 &- 7q^2 \\
18q - 24 &
\end{align*} & 24q^3 + 68q^2 + 60q + 16 & -4q^2 + 12q + 16 & 960q_0q_1q_2q_3 + 480q_0q_1q_2 + 480q_0q_2q_3 + 480q_1q_2q_3 + 120q_0q_1 + 240q_0q_2 + 240q_1q_2 + 152q_0q_3 + 240q_1q_3 + 136q_2q_3 + 16q_0 + 60q_1 + 68q_2 + 24q_3 & 24q_0q_1q_2 + 68q_0q_1q_3 + 60q_0q_2q_3 + 16q_1q_2q_3 + 44q_0q_1 + 156q_0q_2 + 112q_1q_2 + 112q_0q_3 + 156q_1q_3 + 44q_2q_3 + 16q_0 + 60q_1 + 68q_2 + 24q_3 \\
\hline
\end{tabular}
\caption{III-2}
\end{table}
<table>
<thead>
<tr>
<th>Toric Arrangement</th>
<th>Char-Poly</th>
<th>f-poly</th>
<th>h-poly</th>
<th>flag f-poly</th>
<th>flag h-poly</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \begin{pmatrix} 2 &amp; 0 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 2 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 2 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 0 &amp; 2 &amp; 1 \ 1 &amp; -1 &amp; 2 &amp; -1 &amp; 1 \end{pmatrix} )</td>
<td>( q^4 - 9q^3 + 32q^2 - 56q + 48 )</td>
<td>( 48q^4 + 184q^3 + 256q^2 + 144q + 24 )</td>
<td>( 8q^3 - 32q^2 + 48q + 24 )</td>
<td>( 13824q_0q_1q_2q_3q_4 + 6912q_0q_1q_2q_3 + 6912q_0q_1q_2q_4 + 6912q_0q_1q_3q_4 + 6912q_0q_2q_3q_4 + 6912q_1q_2q_3q_4 + 3456q_0q_1q_3q_4 + 3456q_0q_2q_3q_4 + 3456q_1q_2q_3q_4 + 2144q_0q_3q_4 + 2144q_1q_2q_4 + 2048q_2q_3q_4 + 1072q_2q_3 + 1024q_2q_4 + 1152q_1q_4 + 368q_3q_4 + 256q_2 + 184q_3 + 48q_4 )</td>
<td>( 48q_0q_1q_2q_3 + 184q_0q_1q_2q_4 + 256q_0q_1q_3q_4 + 144q_0q_2q_3q_4 + 24q_1q_2q_3q_4 + 136q_0q_1q_2 + 720q_0q_1q_3 + 960q_0q_2q_3 + 424q_1q_2q_3 + 584q_0q_1q_4 + 1400q_0q_2q_4 + 864q_1q_2q_4 + 464q_0q_3q_4 + 584q_1q_3q_4 + 120q_2q_3q_4 + 120q_0q_1 + 584q_0q_2 + 464q_1q_2 + 864q_0q_3 + 1400q_1q_3 + 584q_2q_3 + 424q_0q_4 + 960q_1q_4 + 720q_2q_4 + 136q_3q_4 + 24q_0 + 144q_1 + 256q_2 + 184q_3 + 48q_4 )</td>
</tr>
<tr>
<td>Toric Arrangement</td>
<td>Char-Poly</td>
<td>f-poly</td>
<td>h-poly</td>
<td>flag f-poly</td>
<td>flag h-poly</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>--------</td>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>((20001))</td>
<td>(q^4 - 9q^3 + 32q^2 - 56q + 64)</td>
<td>(64q^4 + 248q^3 + 352q^2 + 208q + 40)</td>
<td>(8q^3 - 32q^2 + 48q + 40)</td>
<td>19968(q_0q_1q_2q_3q_4) + (9984q_0q_1q_2q_3) + (9984q_0q_1q_2q_4) + (9984q_0q_1q_3q_4) + (9984q_0q_2q_3q_4) + (9984q_0q_1q_2q_3q_4) + (4992q_0q_1q_3q_4) + (4992q_0q_2q_3q_4) + (4992q_1q_2q_3q_4) + (3328q_0q_1q_4) + (4992q_0q_1q_2q_4) + (4992q_0q_1q_3q_4) + (3168q_0q_1q_3q_4) + (2816q_2q_3q_4) + (416q_0q_1) + (1248q_0q_2) + (1248q_1q_2) + (1584q_0q_3) + (2496q_0q_1) + (1408q_2q_3) + (752q_0q_4) + (1664q_1q_4) + (1408q_2q_4) + (496q_3q_1 + 40q_0 + 208q_1) + (352q_2 + 248q_3 + 64q_4)</td>
<td>64(q_0q_1q_2q_3 + 248q_0q_1q_2q_4) + 352(q_0q_1q_3q_4 + 208q_0q_1q_2q_4) + 40(q_1q_2q_3q_4) + 184(q_0q_1q_2) + 992(q_0q_1q_3) + 1392(q_0q_2q_3) + 648(q_1q_2q_3) + 808(q_0q_1q_4) + 2040(q_0q_2q_4) + 1296(q_1q_2q_4) + 688(q_0q_3q_4) + 856(q_1q_3q_4) + 168(q_2q_3q_4) + 168(q_0q_1) + 856(q_0q_2) + 688(q_1q_2) + 1296(q_0q_2q_4) + 2040(q_0q_2q_4) + 688(q_0q_3q_4) + 856(q_1q_3q_4) + 808(q_2q_3) + 648(q_0q_4) + 1392(q_1q_4) + 992(q_2q_4) + 184(q_3q_1 + 40q_0 + 208q_1) + 352(q_2 + 248q_3 + 64q_4)</td>
</tr>
<tr>
<td>Toric Arrangement</td>
<td>Char-Poly</td>
<td>f-poly</td>
<td>h-poly</td>
<td>flag f-poly</td>
<td>flag h-poly</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>--------</td>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>(2 0 0 0 0 0 1)</td>
<td>q^5 - 0</td>
<td>48q^5 +</td>
<td>16q + 32</td>
<td>48q_0 q_1 q_2 q_3 q_4 q_5 +</td>
<td>48q_0 q_1 q_2 q_3 q_4 + 224q_0 q_1 q_2 q_3 q_5 +</td>
</tr>
<tr>
<td>(0 2 0 0 0 0 1)</td>
<td>11q^4 +</td>
<td>224q_4 +</td>
<td>76800q_0 q_1 q_2 q_3 q_4 + 76800q_0 q_1 q_2 q_3 q_5 + 76800q_0 q_1 q_2 q_3 q_4 q_5 +</td>
<td>76800q_0 q_1 q_2 q_3 q_4 q_5 + 32q_1 q_2 q_3 q_4 q_5 +</td>
<td></td>
</tr>
<tr>
<td>(0 0 2 0 0 0 1)</td>
<td>48q^3 -</td>
<td>416q^3 +</td>
<td>176q + 32</td>
<td>176q_0 q_1 q_2 q_3 q_4 q_5 +</td>
<td>176q_0 q_1 q_2 q_3 q_4 q_5 + 1232q_0 q_1 q_2 q_3 q_4 +</td>
</tr>
<tr>
<td>(0 0 0 0 2 1)</td>
<td>104q^2 +</td>
<td>384q_2 +</td>
<td>18816q_0 q_1 q_2 q_3 q_4 + 38400q_0 q_1 q_2 q_3 q_4 q_5 +</td>
<td>2832q_0 q_1 q_2 q_3 q_4 q_5 + 2976q_0 q_1 q_2 q_3 q_4 q_5 +</td>
<td></td>
</tr>
<tr>
<td>(1 1 0 0 0 1)</td>
<td>112q -  32</td>
<td>32q_1 +</td>
<td>38400q_0 q_1 q_2 q_3 q_4 q_5 + 25600q_0 q_1 q_2 q_3 q_4 q_5 +</td>
<td>1200q_0 q_1 q_2 q_3 q_4 q_5 q_6 + 1056q_0 q_1 q_2 q_3 q_4 q_5 q_6 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td>48</td>
<td></td>
<td>153600q_0 q_1 q_2 q_3 q_4 q_5 +</td>
<td>416q_0 q_1 q_2 q_3 q_4 q_5 q_6 + 384q_1 q_2 q_3 q_4 q_5 q_6 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>38400q_0 q_1 q_2 q_3 q_4 + 38400q_0 q_1 q_2 q_3 q_4 q_5 +</td>
<td>176q_0 q_1 q_2 q_3 q_4 q_5 q_6 + 32q_1 q_2 q_3 q_4 q_5 q_6 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>38400q_0 q_1 q_2 q_3 q_4 q_5 + 25600q_0 q_1 q_2 q_3 q_4 q_5 q_6 +</td>
<td>1200q_0 q_1 q_2 q_3 q_4 q_5 q_6 + 1056q_0 q_1 q_2 q_3 q_4 q_5 q_6 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>25600q_0 q_1 q_2 q_3 q_4 q_5 + 38400q_0 q_1 q_2 q_3 q_4 q_5 q_6 +</td>
<td>4288q_0 q_1 q_2 q_3 q_4 q_5 q_6 + 6000q_0 q_1 q_2 q_3 q_4 q_5 q_6 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>19584q_2 q_3 q_4 q_5 + 3008q_0 q_1 q_2 q_3 q_4 q_5 q_6 +</td>
<td>2944q_0 q_1 q_2 q_3 q_4 q_5 q_6 + 1600q_0 q_1 q_2 q_3 q_4 q_5 q_6 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9408q_0 q_1 q_3 + 9408q_0 q_2 q_3 + 9408q_0 q_2 q_3 q_4 +</td>
<td>4122q_0 q_1 q_2 q_3 q_4 q_5 q_6 + 2688q_0 q_1 q_2 q_3 q_4 q_5 q_6 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12800q_0 q_1 q_4 + 19200q_0 q_2 q_4 +</td>
<td>944q_0 q_1 q_3 q_4 q_5 q_6 + 1088q_0 q_1 q_2 q_3 q_4 q_5 q_6 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>19200q_0 q_2 q_4 + 12800q_0 q_3 q_4 +</td>
<td>144q_0 q_2 q_3 q_4 q_5 q_6 + 240q_0 q_1 q_2 q_3 q_4 q_5 q_6 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>19200q_0 q_3 q_4 + 9792q_2 q_3 q_4 +</td>
<td>3200q_0 q_2 q_3 q_4 q_5 q_6 + 3280q_0 q_1 q_2 q_3 q_4 q_5 q_6 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6400q_0 q_1 q_5 + 12800q_0 q_2 q_5 +</td>
<td>10240q_0 q_2 q_3 q_4 q_5 + 7184q_1 q_2 q_3 q_4 q_5 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12800q_0 q_2 q_5 + 12800q_0 q_3 q_5 +</td>
<td>5440q_0 q_3 q_4 q_5 + 7216q_0 q_2 q_3 q_4 q_5 q_6 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>19200q_0 q_3 q_5 + 9792q_2 q_3 q_5 +</td>
<td>1824q_0 q_2 q_3 q_4 q_5 q_6 + 5440q_0 q_1 q_2 q_3 q_4 q_5 q_6 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6400q_0 q_4 q_5 + 12800q_0 q_1 q_4 +</td>
<td>17184q_0 q_3 q_4 q_5 + 10240q_0 q_1 q_2 q_3 q_4 q_5 q_6 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9792q_2 q_4 q_5 + 3392q_3 q_4 q_5 + 352q_0 q_1 +</td>
<td>3280q_0 q_2 q_3 q_4 q_5 q_6 + 1776q_0 q_1 q_2 q_3 q_4 q_5 q_6 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1504q_0 q_2 + 1504q_1 q_2 + 3136q_0 q_3 +</td>
<td>1840q_0 q_2 q_3 q_4 q_5 + 240q_3 q_4 q_5 + 144q_0 q_1 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4704q_0 q_3 + 2400q_2 q_3 + 3200q_0 q_4 +</td>
<td>1088q_0 q_2 + 944q_1 q_2 + 2688q_0 q_3 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6400q_1 q_4 + 4896q_2 q_4 + 1696q_3 q_4 +</td>
<td>4112q_1 q_3 + 1600q_2 q_3 + 294q_0 q_1 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12800q_0 q_5 + 3200q_1 q_5 + 3264q_2 q_5 +</td>
<td>6000q_1 q_4 + 4288q_2 q_4 + 1056q_3 q_4 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1696q_3 q_5 + 448q_4 q_5 + 32q_0 + 176q_1 +</td>
<td>1200q_0 q_5 + 2976q_1 q_5 + 2832q_2 q_5 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>384q_2 + 416q_3 + 224q_4 + 48q_5</td>
<td>1232q_3 q_5 + 176q_4 q_5 + 32q_0 + 176q_1 +</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>384q_2 + 416q_3 + 224q_4 + 48q_5</td>
<td></td>
</tr>
</tbody>
</table>
Conjecture 8.8. The coefficients for flag h-polynomial and cd-index are non-negative.

Conjecture 8.9. The coefficient cycle of ab-index (flag h-polynomial): start with any ab-string, the alternating sum of the coefficients of changing one variable (i.e. a to b or b to a) in order (i.e. from right to left) is equal to 0 if n is odd and itself if n is even.

For example:
n = 3: given an ab-string bbba, we have: \( \text{coeff}(bbba) - \text{coeff}(bbaa) + \text{coeff}(baba) - \text{coeff}(abba) = \text{coeff}(bbba) \).

Alternatively, \( h_{012} - h_{01} + h_{02} - h_{12} = h_{012} \).

Take the second toric arrangement on Page 84 as an example, we have: \( 24 - 44 + 156 - 112 = 24 \).

n = 4: given an ab-string bbbaa, we have: \( \text{coeff}(bbbaa) - \text{coeff}(bbbab) + \text{coeff}(bbbba) - \text{coeff}(bbaaa) + \text{coeff}(babaa) - \text{coeff}(abbaa) = 0 \).

Alternatively, \( h_{012} - h_{0124} + h_{0123} - h_{01} + h_{02} - h_{12} = 0 \).

Take the toric arrangement on Page 85 as an example, we have: \( 136 - 184 + 48 - 120 + 524 - 464 = 0 \).

References


Department of Mathematics, University of Michigan, Ann Arbor, MI