Fall 2016

# Deep Data Analysis on the Web

Xuanyu Liu

*San Jose State University*

# DEEP DATA ANALYSIS ON WEB

A Writing Project

Presented To

The Faculty Of The Department Of Computer Science

San Jose State University

In Partial Fulfillment

Of The Requirement for The Degree

Major of Science

By

Xuanyu Liu

DEEP DATA ANALYSIS ON WEB

By

Xuanyu Liu

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

Dec 2016

| | |
|---|---|
| Dr. Tsau Young Lin | Department of Computer Science |
| Dr. Teng Moh | Department of Computer Science |
| Dr. Kong Li | Department of Computer Science |

# ABSTRACT

## DEEP DATA ANALYSIS ON WEB

By

Xuanyu Liu

Search engines are well known to people all over the world. People prefer to use keywords searching to open websites or retrieve information rather than type typical URLs. Therefore, collecting finite sequences of keywords that represent important concepts within a set of authors is important, in other words, we need knowledge mining. We use a simplicial concept method to speed up concept mining. Previous CS 298 project has studied this approach under Dr. Lin. This method is very fast, for example, to mine the concept, FP-growth takes 876 seconds from a database with 1257 columns 65k rows, simplicial complex only takes 5 seconds. The collection of such concepts can be interpreted geometrically into simplicial complex, which can be construed as the knowledge base of this set of documents. Furthermore, we use homology theory to analyze this knowledge base (deep data analysis). For example, in mining market basket data with {a, b, c, d}, we find out frequent item sets {abc, abd, acd, bcd}, and the homology group $H_2$ = Z (the integer Abelian group), which implies that very few customers buy four items together {abcd}, then we may analysis possible causes, etc.

Roughly speaking, this can be regarded as a new discipline in data science based on Dr. Vasant Dhār's (Ph.D., Editor-in-Chief of Big Data, a professor of NYU) definition of data science ("Data Science is a study of generalizable extraction of knowledge from data.").

Table of Contents

## List of Figures

## 1. Introduction

The development of science and technology makes the exchange of information more rapidly and instant. People can access information more conveniently through a variety of clients and terminals. Nowadays, the Internet is the main way of disseminating information because it is cheap and information can spread rapidly over a wide range. There will be a variety of emerging information on the Internet every day, and how people quickly find the information they want is through the keyword search. Each keyword has its concept behind it, and this concept signifies what kind of information people are looking for. Therefore, concept mining is imperative.

In topology, mathematicians examine holes (also known as calculation of homology [4]) to distinguish different shapes in high-dimensional spaces. When we put each keyword as a point in space and connect these points, a high-dimensional geometry can be formed. By calculating homology, we can examine these holes in the geometry constituted by those concepts, and these holes can be understood as concepts that we have not invented or undiscovered. It is also valuable and helpful in scientific research because applying the same method to different areas will have new discoveries [1]. For example, in biology, we can find out genes that are associated with each other in some way that we are not aware of yet.

In the high-dimensional space, there are shapes called the simplicial complexes [12]. A simplicial complex is a high-dimensional geometric combination of triangles composed of lines segments and points. Converting concepts into a simplicial complex [14] is the primary method we use here. Through the establishment of a chain complex, we analyze the holes in a simplicial complex. We also refer to the Smith normal form [9] in the calculation to simplify the calculation. However, in the present study, we can determine whether the holes exist or not or find out the number of holes. A further study and exploration are still needed to discover what generators cause the holes.

# 2. Background

## 2.1 Fundamental Terminologies in Concept Based Search Engine

The Concept Based Search Engine [12] is a program that generates semantic concepts from websites. Some terminologies used in it are defined below.

- Token: A token is a single English word that produced by the tokenizer.
- Token frequency: Token frequency is an informal representation of concept frequency. It represents the total number of occurrences of a concept in all the documents.
- Document frequency: It represents the total number of documents that contain a concept.
- Concept: A concept is a finite sequence of data pattern produced by the Search Engine. A concept is an abstract or general idea or the notion that can be derived from a particular instance of an English word or some combinations of them. A concept is used to represent human thoughts and ideas in symbol notion or language basis. For example:
    1) Wall Street: A wall is a physical structure that divides an area and provides security. A street is a thoroughfare built by human to provide convenient transportations. However, Wall Street physically denotes a street in New York City, but over time it refers to the financial market of the United States as a whole.
    2) White House: While house represents the government building of the president of the U.S. or the government of the U.S.
- Knowledge base: A knowledge base is a large organized collection of concepts produced by the Search Engine.

- Stop words: A stop word is an English word that has no substantial significance in mining concepts or search queries. Stopwords should be removed in data processing. Some examples of stop words are: about, above, across, also, always, have, please, yet, yourself, and so on.

## 2.2 Overview of the Concept Based Search Engine

The Concept Based Search Engine takes a set of documents as input and produces a list of concepts with information such as token count, token frequency, document frequency as well as the documents that contain them. It tokenizes input streams into single tokens and takes a paragraph of tokens as an input for generating concepts. Within a paragraph of tokens, the program generates different combinations of individual tokens to form concepts and stores them into a hashmap. After processing all the documents, the program will eliminate concepts based on the input settings. The eliminating process guarantees that most useless and nonsense concepts will be removed from the map. The last step is to transport all the useful concepts to a database. Figure 2-1 shows the workflow of the search engine. More specific details and algorithm in the search engine are not shown here.

The Concept-based Search Engine takes a major role because it produces our knowledge-based for homology calculation.

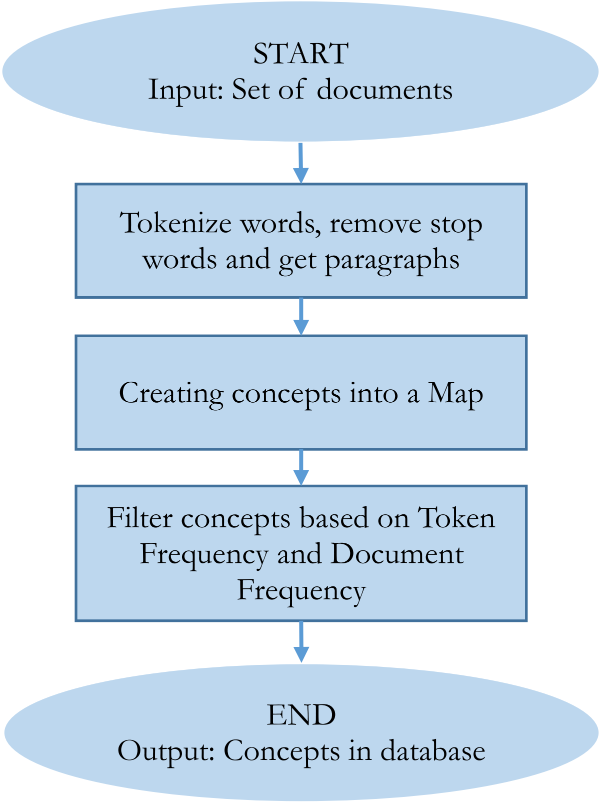

START
Input: Set of documents
Tokenize words, remove stop words and get paragraphs
Creating concepts into a Map
Filter concepts based on Token Frequency and Document Frequency
END
Output: Concepts in database

Figure 2-1

### 2.3 Secondary Storage Management

The current Search Engine works well in generating concepts, but there is a problem with it. For a real world large input data set of documents, the Concept Based Search Engine will eventually run into error due to insufficient memory. For example, to generate concepts up to 4 tokens from 2000 documents, the memory usage is about 7 Gigabytes. As a result, machines with memory size less than 4 gigabytes will not successfully process all the input data in a single execution, and today, the average memory size of PCs is 8 Gigabytes, which implies that some tricks are needed in the process of generating concepts.

In ancient time, when programmers deal with data larger than the memory size, a secondary storage management technic needs to be implemented. A solution to our problem is to use secondary storage management, for example, an external merge sort.

At some proper phases of the program when the memory is taken large enough, we export all the data from the Hashmap to a temporary file on the local hard drive. After all the concept finding, we merge all the temporary files into one single file and then do the eliminating process. This idea is based on the theory that external space is infinity that we do not need to worry about. Temporary files produced by the program would take a tremendous amount of space, and the maximum space taken is twice as final merged file because of the merging process. Though the speed of hard drives is hundreds to thousands of times slower than the main memory, memory is not easily expandable and cannot expand to ideally infinity as we desire.

The algorithm of the external merge sort for the Search Engine is:

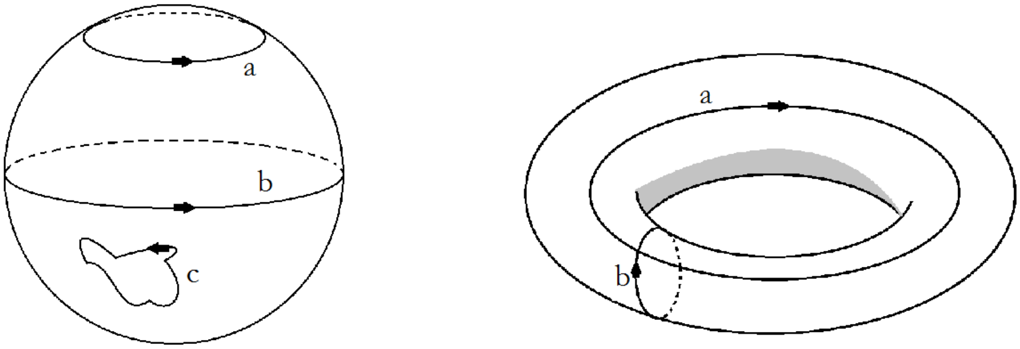
```
Output sorted concepts to temporary files $T[n]$
While T.size() > 1
    Read each line L1 and L2 from $T[1]$ and $T[2]$
    If $(\text{order}(L1) < \text{order}(L2))$ Write L1 and L2 to file T'
    Else If $(\text{order}(L1) > \text{order}(L2))$ Write L2 and L1 to file T'
    Else If $(\text{order}(L1) == \text{order}(L2))$ Merge L2 and L1 to file T'
    Place file T' into the last position of files $T[n]$
End
```

The idea of secondary storage management is used later in homology calculation because the data set processed is usually large and takes a significant amount of memory.

## 2.4 Homology

Homology [2][4][5] in topology is a theory used to distinguish and describe spaces. We are particularly interested in simplicial homology because our concepts can be triangulated into a simplicial complex, and the homology can tell the holes in the simplicial complex. These holes can be understood as concepts that we have not invented or undiscovered.

The dictionary meaning of homology is the quality of being similar or corresponding in relation. Therefore, we can interpret homology in such a way. For any shape in space, we look at cycle paths on its surface to determine whether those cycle paths can expand or shrink to each other without cutting the shape. If those cycle paths can turn into each other by expanding and shrinking, they are considered being in the same homology class and homologous to each other. Furthermore, if those cycle paths can shrink into a point, they are also said to be homologous to 0. For example, in Figure 2-2(a), a sphere S has $cycle\ a,\ b$ and $c$ on its surface in the counterclockwise rotation. By shrinking $cycle\ b$ and expanding $cycle\ \mathrm{c}$, they can transform into $cycle\ a$, and by the same operation, they can transform into each other. They can also shrink to a point without cutting the sphere. Therefore, cycle $a,\ b$ and $c$ are homologous to 0. If we traverse cycle $a$ in the direction, which is counterclockwise rotation in the figure, the traversal can iterate any number of times. If we say each cycle is p, then we can add all the iterations together as $p+p+\cdots+p$. This addition is the same as the integer space $\mathbf{Z}$, and $p$ is called the generator. On the other hand, if we traverse an iteration of cycle $a$ clockwise, then each cycle would be represented by $-p$, and a clockwise traversing followed by a counterclockwise traversing would result in $p-p=0$. However, not all cycle path can be transformed to each other. In Figure 2-2(b), cycle $a$ and cycle $b$ in a torus cannot shrink into a point nor transform to each other without cutting the torus. We say that they are not in a same homologous group. If we say each cycle of $a$ is $p$ and each cycle of $b$ is $q$, then we can add all the iterations together as $p+p+p+\cdots+q+q+\cdots+q$, which is homology group $\mathbf{Z}\times\mathbf{Z}$.



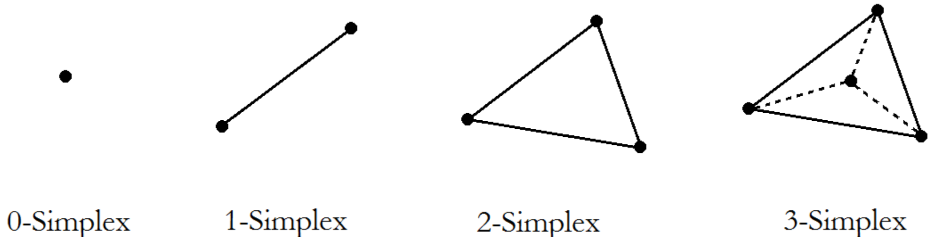
(a) Sphere

(b) Torus

Figure 2-2

## 2.5 Simplex and Simplicial Complex

An (open) n-simplex[13][15] is an n-dimensional geometry that consists of (n+1) vertices in Euclidean space $\mathbb{R}^n$. It can be viewed as the complete graph on (n+1) vertices in n dimensions. For example, if we have a single point in space, then it can be seen as an open 0-simplex, an open 0-simplex has only one single vertex. If we add another point in the space and connect those two points, then we have a line segment, forming an open 1-simplex. An open 1-simplex can be interpreted as the 1-simplex has only the line segment, and the points connecting it are not included. Adding a third point and connecting each two of them will result in an open 2-simplex, which is also a triangle. An open 2-simplex is the open triangle connected by the three points, but the points and the edges are not included. As we continue adding the fourth point and connecting each 2 of them, an open 3-simplex can be built in three-dimensional space, and an open 3-simplex is known as a tetrahedron. The open 3-simplex can be seen as a tetrahedron without its faces, which are four open 2-simplex (triangles), six 1-simplex (line), and four 0-simplex (vertices). A closed n-simplex is a simplicial complex that has an open n-simplex and all its faces. A simplicial complex [14] is a collection of simplexes such that any set consisting of one vertex is a simplex and any nonempty subset of a simplex is a simplex.



0-Simplex 1-Simplex 2-Simplex 3-Simplex

Figure 2-3

Table 3.2 shows the number of faces for an n-simplex.

| | 0-faces (vertices) | 1-faces (edges) | 2-faces (triangle) | 3-faces | 4-faces | 5-faces | 6-faces |
|---|---|---|---|---|---|---|---|
| 0-simplex (point) | 1 | | | | | | |
| 1-simplex (line) | 2 | 1 | | | | | |
| 2-simplex (triangle) | 3 | 3 | 1 | | | | |
| 3-simplex (tetrahedron) | 4 | 6 | 4 | 1 | | | |
| 4-simplex | 5 | 10 | 10 | 5 | 1 | | |
| 5-simplex | 6 | 15 | 20 | 15 | 6 | 1 | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **6-simplex** | **7** | **21** | **35** | **35** | **21** | **7** | **1** |
| **…** | | | | | | | |

Table 3.2

## 2.6 Simplicial Homology and Chain Complex

Before we define simplicial homology, we need to introduce another property of simplex, which is orientation. The orientation of a simplex is simply the order of its vertices as we write it. Therefore, every simplex has two orientations. For example, an orientated simplex $\Delta(a,b,c)$ is equal to the negative of a pair-reverse order of the simplex, that is

$$\Delta(a,b,c) = -\Delta(a,c,b) = \Delta(c,a,b) = -\Delta(c,b,a) = \Delta(b,c,a) = -\Delta(b,a,c)$$

Having the idea of Simplicial Complex, we can see that the n-dimensional simplexes of space are not unique, but the homology groups of the same Euclidean space are the same. The construction of the homology groups depends on the free groups of n-simplex and the boundary operator that maps an-simplex to an (n-1)-simplex. We can think of simplicial homology group of the n-dimensional cycle bounded by the (n+1)-dimensional cycle. What we need to do is to determine whether the (n+1)-dimensional boundary exists for the n-dimensional cycle or not. If the boundary cycle exists, the (n+1) dimensional hole is covered, and the solid enclosing can be shrunk to a point.

We use chain complex to construct homology group and discern the relation between cycles and boundaries in different dimensional spaces. The boundary operator is written as $\partial$. Given an-simplex $\Delta[x_0, x_1, x_2, \cdots, x_n]$, the boundary operator [3] $\partial[x_0, x_1, x_2, \cdots, x_n]$ is the (n-1)-chain of the chain complex, defined as the weighted sum of the facets of the n-simplex:

$$\partial[x_0, x_1, x_2, \cdots, x_n] = \sum_{i=0}^{n} (-1)^i [x_0, x_1, \cdots, \hat{x}_i, \cdots, x_n],$$

Where $[x_0, x_1, \cdots, \hat{x}_i, \cdots, x_n]$ is the facet opposite vertex $x_i$, which is removed from the calculation. For example,

$$\begin{aligned} \partial[a,b,c,d] &= [b,c,d] - [a,c,d] + [a,b,d] - [a,b,c] \\ \partial[a,b,c] &= [b,c] - [a,c] + [a,b] \\ \partial[a,b] &= [b] - [a] \\ \partial[a] &= [] \end{aligned}$$

From above four differential equations, we can see that for a 3-simplex, the three-dimensional tetrahedron bounds four faces of triangles. A two-dimensional triangle bounds three faces of line segments. A one-dimensional line bounds two faces of vertices. A vertex is just a basis in 0-dimension.

Formally, a chain complex [10] is a sequence of $\mathbf{Z}\text{–}vector$ space $C_1, C_2, \cdots, C_n$, connected by the homomorphism $\partial_n : C_n \to C_{n-1}$, which is the boundary operator defined above.

$$0 \longrightarrow C_n \xrightarrow{\partial_n} C_{n-1} \xrightarrow{\partial_{n-1}} \cdots \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \longrightarrow 0$$

The group $C_n$ is the set of open n-simplexes, and 0 is the trivial group that consists of only one element. The chain complex has a property that any two consecutive boundary operation will result in trivial group 0. That is,

$$\textit{For all the n, } \partial_n \circ \partial_{n+1} = 0,$$

where $\circ$ is the operator connects two consecutive boundary operation. This represents that the continuous mapping sends all the elements in $C_n$ to the group identity in $C_{n-1}$. This property tells us that the boundary of a boundary is trivial as in mathematical definition:

$$\mathrm{im}(\partial_{n+1}) \subseteq \ker(\partial_n),$$

where $\mathrm{im}(\partial_{n+1})$ is the image of the boundary operator $\partial_{n+1}$ and $\ker(\partial_n)$ is the kernel of the boundary operator $\partial_n$. We define elements in $B_n(X) = \mathrm{im}(\partial_{n+1})$ as boundaries and elements in $\mathrm{Z}_n(X) = \ker(\partial_n)$ as cycles. Then we can compute the *n*th homology group of X by identifying the quotient group,

$$H_n(X) = \frac{\mathrm{Z}_n(X)}{B_n(X)} = \frac{\ker(\partial_n)}{\mathrm{im}(\partial_{n+1})}$$

However, the computation of the boundaries and cycles is difficult in general because the size of generators is usually large. By the fundamental theorem of the finitely generated abelian groups [11],

"*Every finitely generated abelian group is isomorphic to a direct sum of cyclic groups.*"

We have, $H_n(X) \cong \mathbf{Z}^{\beta_n} \oplus \bigoplus_i (\mathbf{Z} / d_i\mathbf{Z})$

$\beta_n$ is an integer value called Betti number of the nth dimension, and the components in the direct sum, $\mathbf{Z}, d_i\mathbf{Z}$ is the torsion subgroups. We use a Smith Normal Form method that will change the basis of the $C_n$ but gives a well formatted linear boundary mapping that we can easily find kernels and images.

## 2.7 Smith Normal Form

Smith normal form [9] is a well-defined method to calculate homology, in particular for the simplicial complex. Let R and S be an $m \times n$matrix, and $R = U * S * V$ , where U is an $m \times m$ matrix, and V is an $n \times n$matrix. If matrix S has the property that all the elements in S are integers and all entries are 0 except for its diagonal entries, $diag[d_1, d_2, \cdots, d_n]$, where $d_{n-1}$ dives $d_n$. Then, we say matrix S is the Smith Normal Form of matrix R, An example of Smith Normal Form:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

As the example shows above, some $7 \times 7$matrices have the same Smith Normal Form as this one. The differences are their left and right invertible matrix U and V. To get the Smith Normal Form of a matrix, only basic row and column operations are needed, such as multiplying a row or column by an integer, adding a multiple of one row or column to another, and interchanging two rows or columns. The matrix U and V are formed from the basic operations done over the matrix R. Before any operations over R, initialize diagonal of U' and V' with all 1s and 0's for all other entries. Apply any basic row operations over R on U' and apply any basic column operations over R on V'. After getting the Smith Normal Form of R, taking the inverse of U' and V' will result in U and V.

# 3. Implementation

## 3.1 Triangulation of Concepts

The first step is to read the knowledge base, and then convert each English word to vertices of a simplex. For example, the concept, “Deep Data Analysis” is a three token concept. Each concept “Deep”, “Data”, and “Analysis” would be represented by open 0-simplexes. “Deep Data”, “Deep Analysis”, and “Data Analysis” would be represented by open 1-simplexes, and “Deep Data Analysis” would represent by open 2-simplex. From the example, we can clearly understand the idea of an “open” simplex. “Deep data Analysis” as a whole has its concept, and it does not have a concept of “Deep Data” or “Deep Analysis”. So, we represent it as an open 2-simpex. A close 2-simplex, on the other hand, has all the faces of the open 2-simplex. Those faces are “Deep Data”, “Deep Analysis”, “Data Analysis”, “Deep”, “Data”, and “Analysis”.

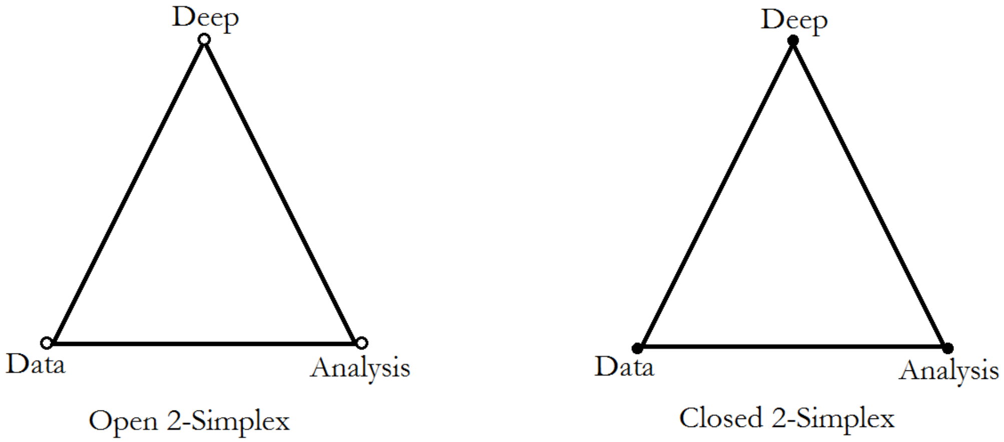
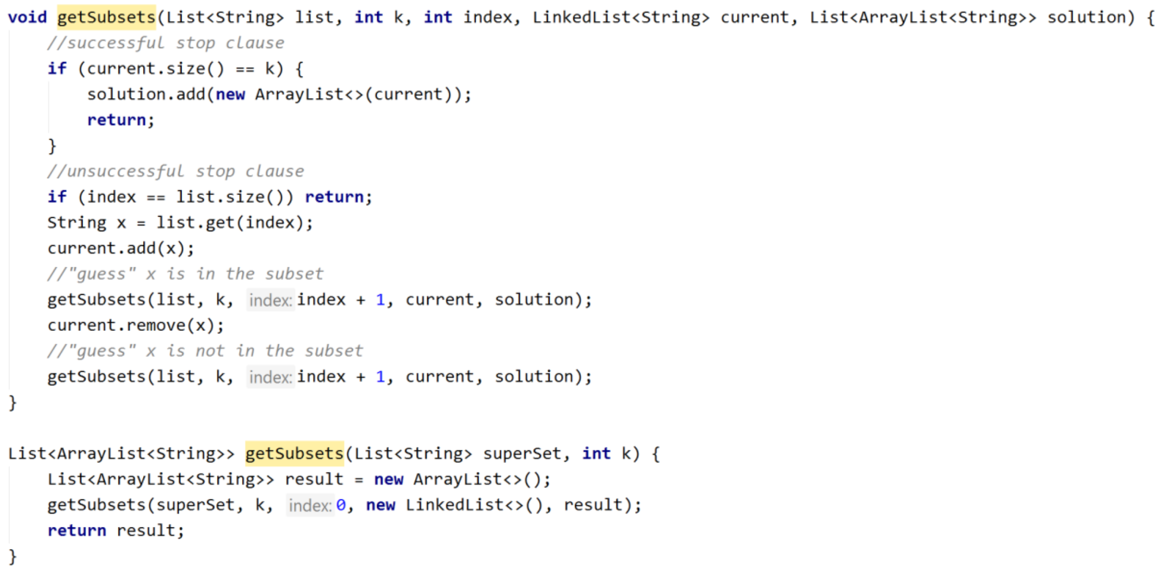


Figure 3-1

We only analysis simplicial complexes that are closed. If any faces of a simplex are missing, those are not worth analyzing because the basis is not well-formed. Our program will check if the simplicial complex is closed based on the input data. Recall that a simplicial complex is a collection of simplexes such that any set consisting of one vertex is a simplex and any nonempty subset of a simplex is a simplex. Therefore, for any input simplex, its subset is generated by the algorithm shown in Figure 3-2, and the existence of each result in the subset is checked in the knowledge base.



```java
void getSubsets(List<String> list, int k, int index, LinkedList<String> current, List<ArrayList<String>> solution) {
    //successful stop clause
    if (current.size() == k) {
        solution.add(new ArrayList<>(current));
        return;
    }
    //unsuccessful stop clause
    if (index == list.size()) return;
    String x = list.get(index);
    current.add(x);
    //"guess" x is in the subset
    getSubsets(list, k, index: index + 1, current, solution);
    current.remove(x);
    //"guess" x is not in the subset
    getSubsets(list, k, index: index + 1, current, solution);
}

List<ArrayList<String>> getSubsets(List<String> superSet, int k) {
    List<ArrayList<String>> result = new ArrayList<>();
    getSubsets(superSet, k, index: 0, new LinkedList<>(), result);
    return result;
}
```

Figure 3-2

To understand the homology with triangulated concept vertices, see Figure 3-3, 0-dimensional simplex “Deep”, “Data”, and “Analysis” have “Deep Data”, “Data Analysis”, and “Deep Analysis” in 1-dimensinal space as their boundary. Therefore, there are no 1-demensional holes. 1-dimensional simplex “Deep Data”, “Data, analysis”, and “Deep Analysis” have “Deep Data Analysis” in 2-dimensional space as their boundary. Therefore, there are no 2-deminsional holes.

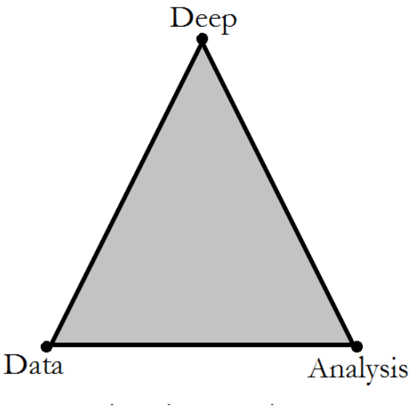


Figure 3-3

Another example without boundary is that suppose “apple pencil”, “pencil store”, and “apple store” are frequent concepts in document sets. Then, the simplex can be formed as Figure 3-4

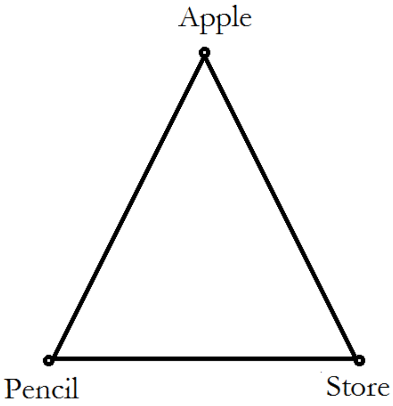


Figure 3-4

In this example, an Apple Store is the store that is selling Apple products such as MacBook’s and iPhones. An Apple Pencil is the product for iPad Pro designed by Apple. A Pencil Store, in general, a store that sells pencils. As we know, the concept, “Apple pencil store” does not exist because there are no stores just selling Apple pencil only. As a result, the concepts “Apple

Pencil", "Pencil Store", and "Apple Store" are not a boundary cycle in 2-dimenstional space, leading to a hollow triangle in space. The cycle, therefore, cannot shrink to a point, and the homology group for the 1-dimensinal space is $\mathbf{Z}$.

### 3.2 Generation of Chain Complex

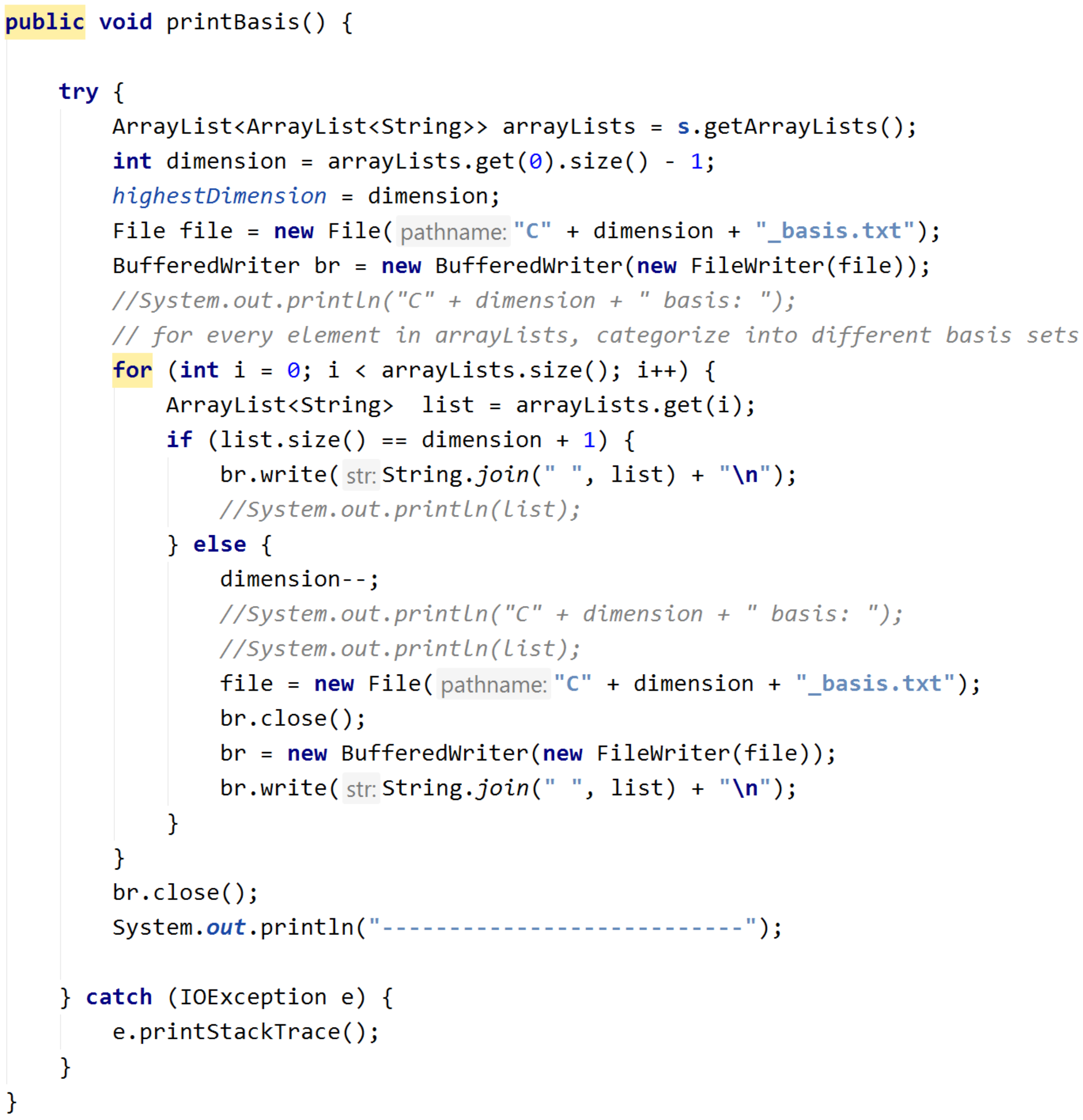
Recall that a chain complex is a sequence of $\mathbf{Z}\text{–}vector$ space $C_1, C_2, \cdots, C_n$, connected by the homomorphism $\partial_n : C_n \to C_{n-1}$, which is the boundary operator defined above.

$$0 \longrightarrow C_n \xrightarrow{\partial_n} C_{n-1} \xrightarrow{\partial_{n-1}} \cdots \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \longrightarrow 0$$

After treating each one-token concept as vertices, the basis for $C_0$ would be all the one-token concepts. Similarly, all the two-token concepts form the basis of $C_0$, and the same idea applies to all the higher dimensions. For example,

$C_0$ : $\Delta(deep)$, $\Delta(data)$, $\Delta(analysis)$
$C_1$ : $\Delta(deep\ data)$, $\Delta(deep\ analysis)$, $\Delta(data\ analysis)$
$C_2$ : $\Delta(deep\ data\ analysis)$

Our program will find the basis for all the dimensions and store basis of each dimension in (dimension)_basis.txt file rather than keep this information in memory.

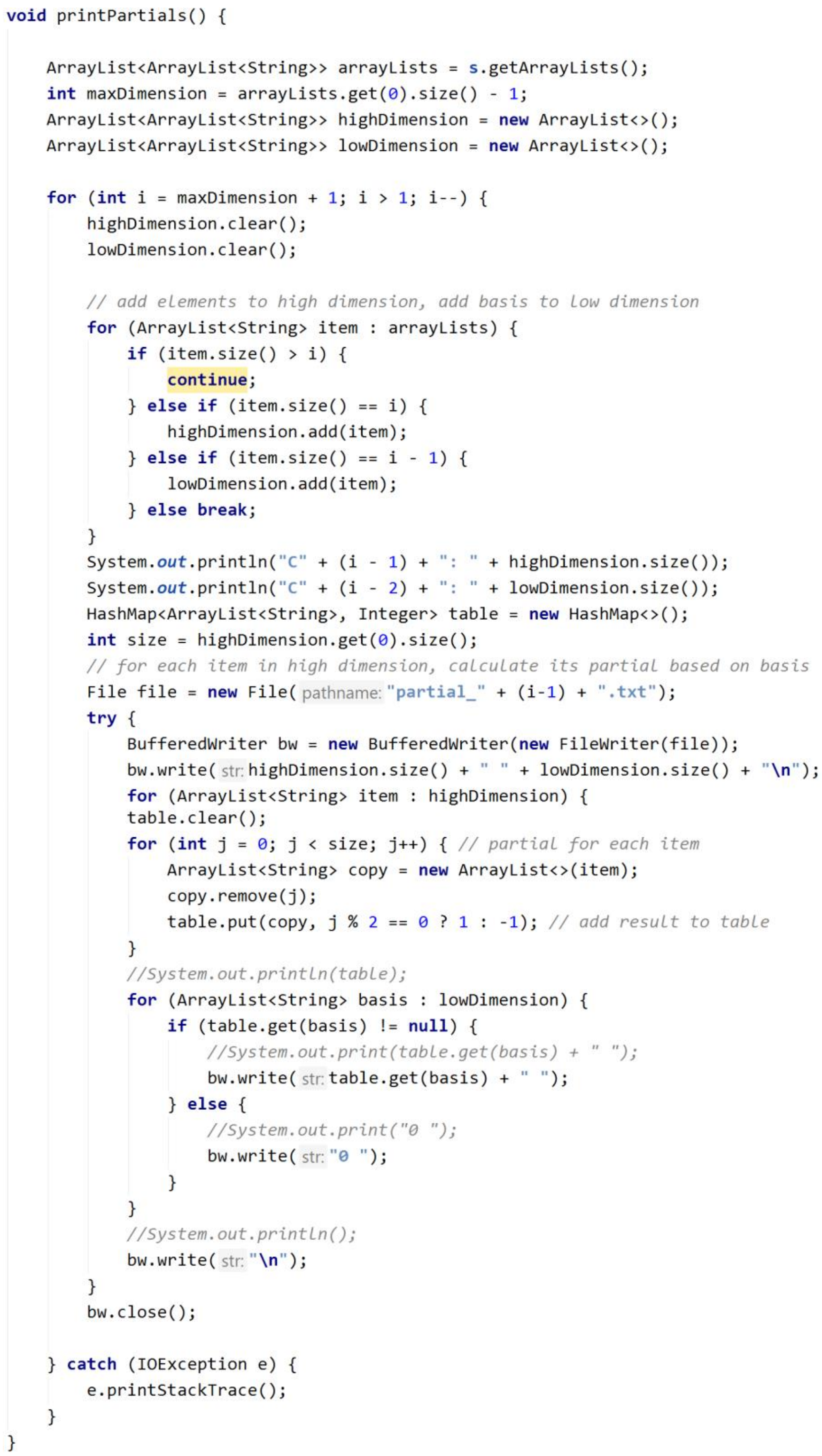


```
public void printBasis() {

    try {
        ArrayList<ArrayList<String>> arrayLists = s.getArrayLists();
        int dimension = arrayLists.get(0).size() - 1;
        highestDimension = dimension;
        File file = new File( pathname: "C" + dimension + "_basis.txt");
        BufferedWriter br = new BufferedWriter(new FileWriter(file));
        //System.out.println("C" + dimension + " basis: ");
        // for every element in arrayLists, categorize into different basis sets
        for (int i = 0; i < arrayLists.size(); i++) {
            ArrayList<String>  list = arrayLists.get(i);
            if (list.size() == dimension + 1) {
                br.write( str: String.join(" ", list) + "\n");
                //System.out.println(list);
            } else {
                dimension--;
                //System.out.println("C" + dimension + " basis: ");
                //System.out.println(list);
                file = new File( pathname: "C" + dimension + "_basis.txt");
                br.close();
                br = new BufferedWriter(new FileWriter(file));
                br.write( str: String.join(" ", list) + "\n");
            }
        }
        br.close();
        System.out.println("---------------------------");

    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Figure 3-5

Once all the basis has been formed, the boundary operations can calculate the linear mapping between any two consecutive dimensions.

$$\partial[x_0, x_1, x_2, \cdots, x_n] = \sum_{i=0}^{n} (-1)^i [x_0, x_1, \cdots, \hat{x}_i, \cdots, x_n]$$



```java
void printPartials() {

    ArrayList<ArrayList<String>> arrayLists = s.getArrayLists();
    int maxDimension = arrayLists.get(0).size() - 1;
    ArrayList<ArrayList<String>> highDimension = new ArrayList<>();
    ArrayList<ArrayList<String>> lowDimension = new ArrayList<>();

    for (int i = maxDimension + 1; i > 1; i--) {
        highDimension.clear();
        lowDimension.clear();

        // add elements to high dimension, add basis to low dimension
        for (ArrayList<String> item : arrayLists) {
            if (item.size() > i) {
                continue;
            } else if (item.size() == i) {
                highDimension.add(item);
            } else if (item.size() == i - 1) {
                lowDimension.add(item);
            } else break;
        }
        System.out.println("C" + (i - 1) + ": " + highDimension.size());
        System.out.println("C" + (i - 2) + ": " + lowDimension.size());
        HashMap<ArrayList<String>, Integer> table = new HashMap<>();
        int size = highDimension.get(0).size();
        // for each item in high dimension, calculate its partial based on basis
        File file = new File( pathname: "partial_" + (i-1) + ".txt");
        try {
            BufferedWriter bw = new BufferedWriter(new FileWriter(file));
            bw.write( str: highDimension.size() + " " + lowDimension.size() + "\n");
            for (ArrayList<String> item : highDimension) {
            table.clear();
            for (int j = 0; j < size; j++) { // partial for each item
                ArrayList<String> copy = new ArrayList<>(item);
                copy.remove(j);
                table.put(copy, j % 2 == 0 ? 1 : -1); // add result to table
            }
            //System.out.println(table);
            for (ArrayList<String> basis : lowDimension) {
                if (table.get(basis) != null) {
                    //System.out.print(table.get(basis) + " ");
                    bw.write( str: table.get(basis) + " ");
                } else {
                    //System.out.print("0 ");
                    bw.write( str: "0 ");
                }
            }
            //System.out.println();
            bw.write( str: "\n");
        }
        bw.close();

    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Figure 3-6

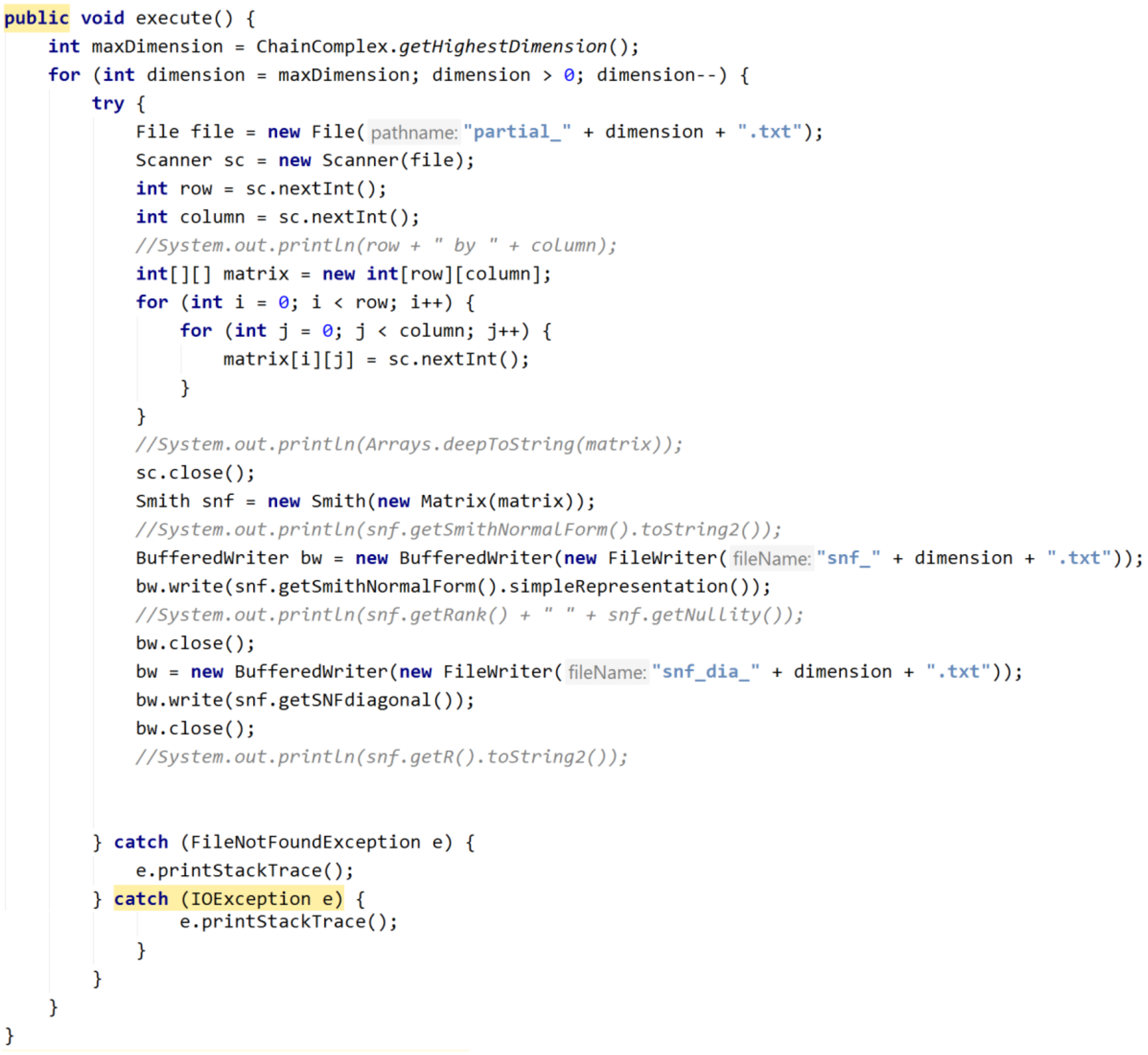
Similarly, the linear mapping is stored in partial_(higher_demension – 1).txt file.

### 3.3 Basis Transformation Using Smith Normal Form

The idea of using Smith Normal Form (SNF) to calculate homology is to generate a corresponding SNF to the boundary operator $\partial_n$. Then, the $U_n$ and $V_n$ matrices could be thought as the new bases in their dimensions. After we get each $\partial_n$ for $n^{th}$ dimensional space, based on the knowledge of Smith Normal Form, we can generate a new linear mapping $\delta_n$ for the same space with a complex new basis. The $\delta_n$ is the Smith Normal Form of $\partial_n$ such that $\partial_n = U_n * \delta_n * V_n$, where $U_n$ and $V_n$ are the new basis of $C_n$ and $C_{n-1}$. Since $\partial_n$ is in a diagonal form, kernels and images of $C_n$ and $C_{n-1}$ would be easy to see. In Figure 3-7, the calculation of Smith Normal Form of each $\partial_n$ is stored in text files as well to save the memory. The algorithm used to calculate Smith Normal Form is the same as the one in Matlab, and an example of calculation of Smith Normal Form is shown is Appendix A. The new construction of the chain complex is shown below:

$$\begin{tikzcd}
C_2 \arrow[r, "\partial_2"] \arrow[d, shift left, "U_2"] & C_1 \arrow[r, "\partial_1"] \arrow[d, shift left, "U_1"] & C_0 \arrow[r, "\partial_0"] \arrow[d, shift left, "U_0"] & 0 \\
N_2 \arrow[r, "\delta_2"] \arrow[u, shift left] & N_1 \arrow[r, "\delta_1"] \arrow[u, shift left, "V_2"] & N_0 \arrow[r, "\delta_0"] \arrow[u, shift left, "V_1"] & 0 \arrow[u, "V_0"]
\end{tikzcd}$$

Figure 3-7



```
public void execute() {
    int maxDimension = ChainComplex.getHighestDimension();
    for (int dimension = maxDimension; dimension > 0; dimension--) {
        try {
            File file = new File( pathname: "partial_" + dimension + ".txt");
            Scanner sc = new Scanner(file);
            int row = sc.nextInt();
            int column = sc.nextInt();
            //System.out.println(row + " by " + column);
            int[][] matrix = new int[row][column];
            for (int i = 0; i < row; i++) {
                for (int j = 0; j < column; j++) {
                    matrix[i][j] = sc.nextInt();
                }
            }
            //System.out.println(Arrays.deepToString(matrix));
            sc.close();
            Smith snf = new Smith(new Matrix(matrix));
            //System.out.println(snf.getSmithNormalForm().toString2());
            BufferedWriter bw = new BufferedWriter(new FileWriter( fileName: "snf_" + dimension + ".txt"));
            bw.write(snf.getSmithNormalForm().simpleRepresentation());
            //System.out.println(snf.getRank() + " " + snf.getNullity());
            bw.close();
            bw = new BufferedWriter(new FileWriter( fileName: "snf_dia_" + dimension + ".txt"));
            bw.write(snf.getSNFdiagonal());
            bw.close();
            //System.out.println(snf.getR().toString2());


        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Figure 3-8

### 3.4 Smith Normal Form Based Homology Calculation

Recall that we have $H_n(X) \cong \mathbf{Z}^{\beta_n} \oplus \bigoplus_i (\mathbf{Z}/d_i\mathbf{Z})$ because of the homology isomorphism. We now have

$$\delta_{i+1} = \begin{bmatrix} b_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & b_2 & \cdots & 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & b_t & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$H_i \cong \mathbf{Z}^{r-t} \oplus (\mathbf{Z}/b_1\mathbf{Z}) \oplus \cdots \oplus (\mathbf{Z}/b_t\mathbf{Z})$$

In the formula, $r = nullity(\delta_i)$, $t = rank(\delta_{i+1})$, and $b_1, b_2, \cdots, b_t$ are corresponding to the diagonal of $\delta_{i+1}$.

## 4. Computation of Homology

In this section, the homology calculation of some known object will be shown to make sure the correctness of the program, and a real world data is calculated in 4.6.

### 4.1 Hollow Triangles

A hollow triangle is a simplicial complex that consists of faces {“A B”, “A C”, “B C”, “A”, “B”, “C”} as shown in Figure 4-1,

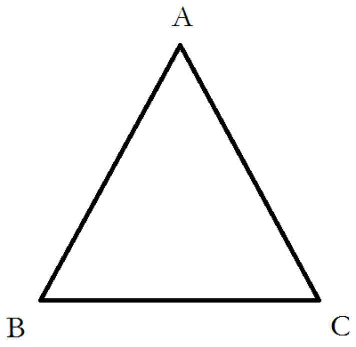
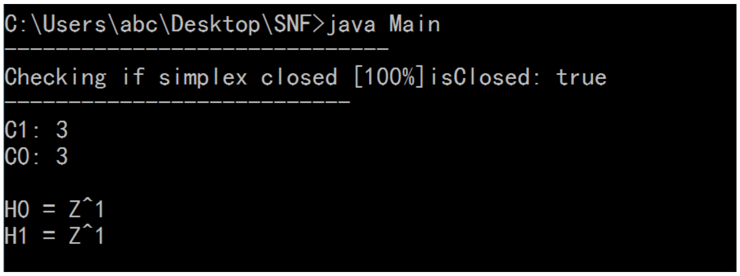


Figure 4-1

Since it is a single path-connected, which means there is no isolated points, and the boundary “A B C” is missing, the homology group should be

$$H_0 = \mathbf{Z}$$
$$H_1 = \mathbf{Z}$$

The calculation from the program is shown in Figure 4-2, which is the expected result.



```
C:\Users\abc\Desktop\SNF>java Main
------------------------------
Checking if simplex closed [100%]isClosed: true
---------------------------
C1: 3
C0: 3

H0 = Z^1
H1 = Z^1
```

Figure 4-2

## 4.2 Solid Triangles

A Solid triangle is a simplicial complex that consist of faces {"A B C", "A B", "A C", "B C", "A", "B", "C"} as shown in Figure 4-3

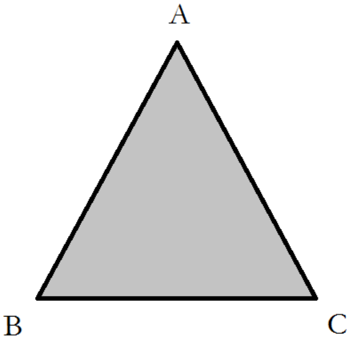
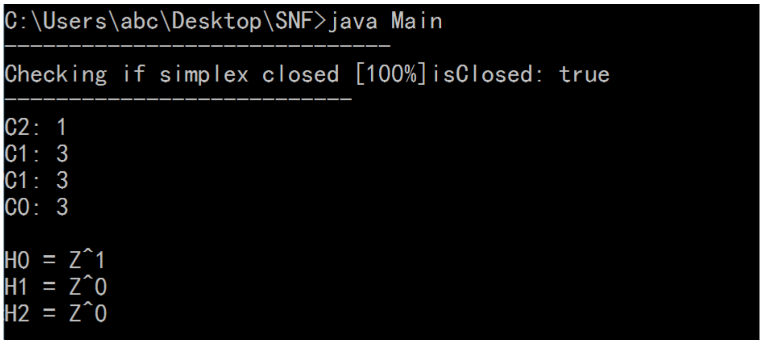


Figure 4-3

Since it is a single path-connected, which means there is no isolated points, and the boundary "A B C" exist, the homology group should be

$$H_0 = \mathbf{Z}$$
$$H_1 = 0$$
$$H_2 = 0$$

The calculation from the program is shown in Figure 4-4, which is the expected result.



```
C:\Users\abc\Desktop\SNF>java Main
------------------------------
Checking if simplex closed [100%]isClosed: true
---------------------------
C2: 1
C1: 3
C1: 3
C0: 3

H0 = Z^1
H1 = Z^0
H2 = Z^0
```

Figure 4-4

## 4.3 Two Triangles with A Shared Edge

The simplicial complex shown in Figure 4-5 has faces {“A B”, “A C”, “B C”, “B D”, “C D”, “A”, “B”, “C”, “D”}

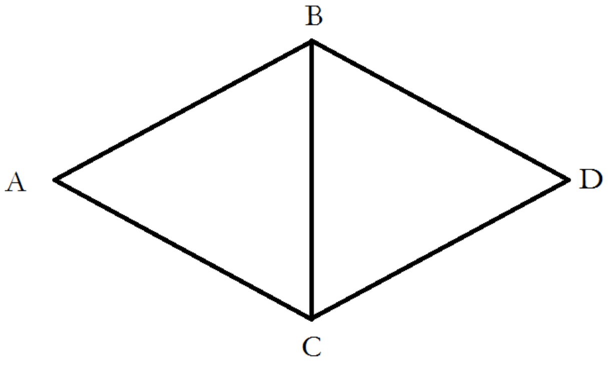
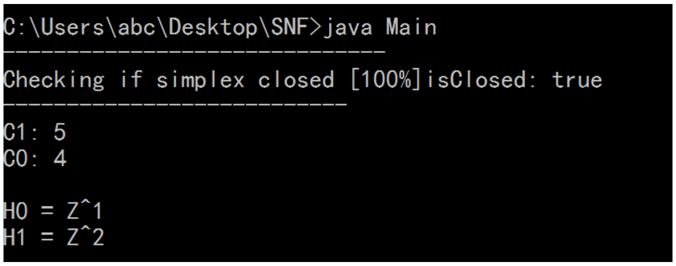


Figure 4-5

Since it is a single path-connected, which means there are no isolated points, and the boundary “A B C” and “B C D” do not exist, the homology group should be

$$H_0 = \mathbf{Z}$$
$$H_1 = \mathbf{Z} \oplus \mathbf{Z}$$

The calculation from the program is shown in Figure 4-6, which is the expected result.



```
C:\Users\abc\Desktop\SNF>java Main
------------------------------
Checking if simplex closed [100%]isClosed: true
---------------------------
C1: 5
C0: 4

H0 = Z^1
H1 = Z^2
```

Figure 4-6

The result matches each other, where the notation $\mathbf{Z}^{\wedge}2$ is the simple expression of $\mathbf{Z} \oplus \mathbf{Z}$.

### 4.4 Two Triangles with A Shared Edge and Two Extra Vertices

The simplicial complex shown in Figure 4-7 has faces {“A B C”, “A B”, “A C”, “B C”, “B D”, “C D”, “A”, “B”, “C”, “D”, “E”, “F”}

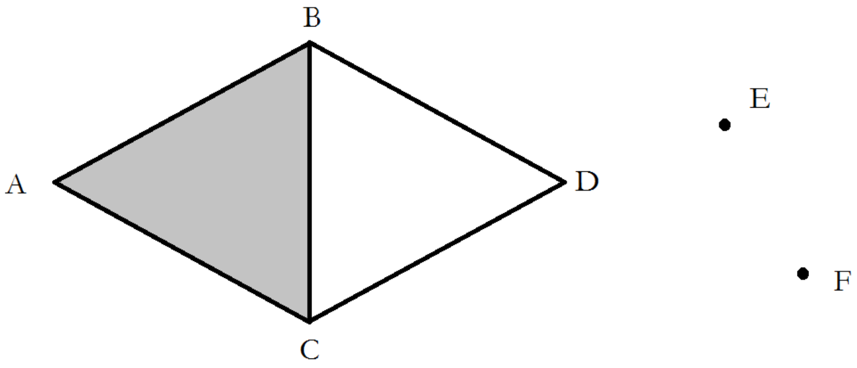
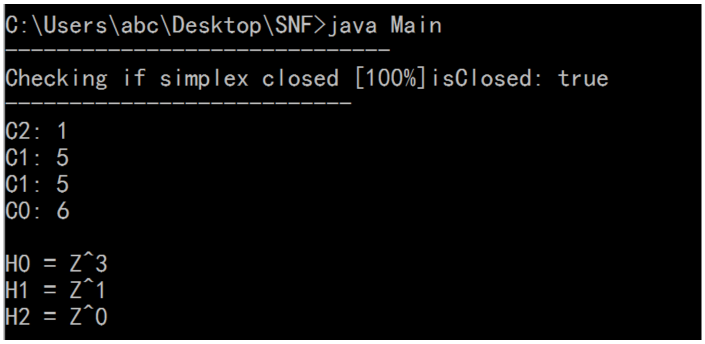


Figure 4-7

Since it has three single connected components, and boundary “B C D” do not exist, the homology group should be,

$$H_0 = \mathbf{Z} \oplus \mathbf{Z} \oplus \mathbf{Z}$$
$$H_1 = \mathbf{Z}$$
$$H_2 = 0$$

The calculation from the program is shown in Figure 4-8, which is the expected result.



```
C:\Users\abc\Desktop\SNF>java Main
------------------------------
Checking if simplex closed [100%]isClosed: true
---------------------------
C2: 1
C1: 5
C1: 5
C0: 6

H0 = Z^3
H1 = Z^1
H2 = Z^0
```

Figure 4-8

### 4.5 Real Projective Plane

The simplicial complex shown in Figure 4-9 has faces {"A B E", "A B F", "A C D", "A C F", "A D E", "B C D", "B C E", "B D F", "C E F", "D E F", "A B", "A C", "A D", "A E", "A F", "B C", "B D", "B E", "B F", "C D", "C E", "C F", "D E", "D F", "E F", "A", "B", "C", "D", "E", "F"}

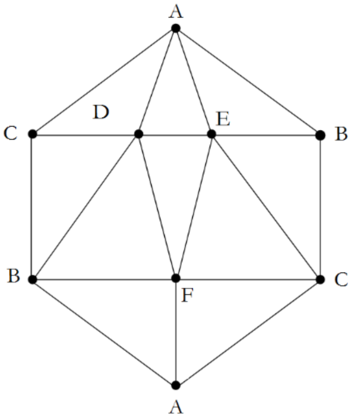
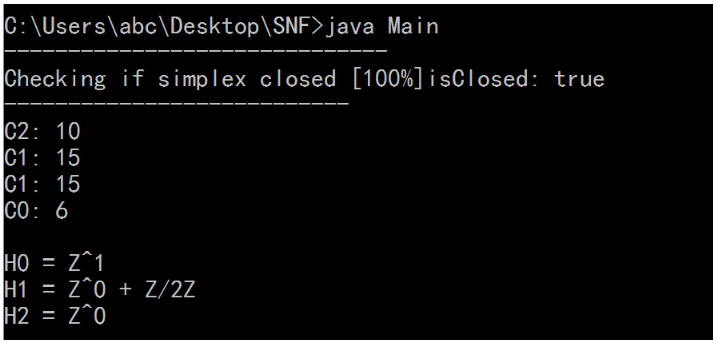


Figure 4-9

The homology group for projective plane 2 is known as,

$$H_0 = \mathbf{Z}$$
$$H_1 = \mathbf{Z}/2\mathbf{Z} = \mathbf{Z}_2$$
$$H_2 = 0$$

The calculation from the program is shown in Figure 4-10, which is the expected result.

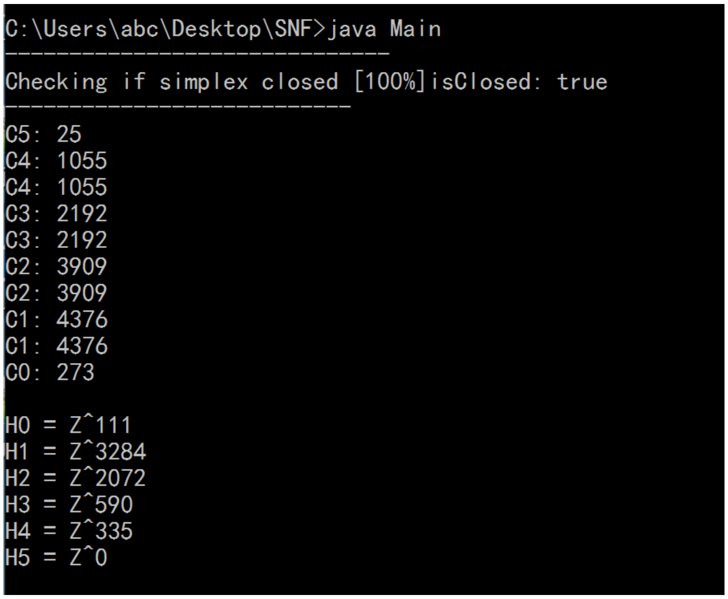


```
C:\Users\abc\Desktop\SNF>java Main
------------------------------
Checking if simplex closed [100%]isClosed: true
---------------------------
C2: 10
C1: 15
C1: 15
C0: 6

H0 = Z^1
H1 = Z^0 + Z/2Z
H2 = Z^0
```

Figure 4-10

### 4.6 Document Data Set

For a simplicial complex with 11830 row of oriented database data generated from the previous students' work [6], the homology calculation is,



```
C:\Users\abc\Desktop\SNF>java Main
------------------------------
Checking if simplex closed [100%]isClosed: true
---------------------------
C5: 25
C4: 1055
C4: 1055
C3: 2192
C3: 2192
C2: 3909
C2: 3909
C1: 4376
C1: 4376
C0: 273

H0 = Z^111
H1 = Z^3284
H2 = Z^2072
H3 = Z^590
H4 = Z^335
H5 = Z^0
```

Figure 4-11

The calculation shows that in the database, there are 111 separate data sets, called concept cluster in (WI2008, ICDM2006) [7][8] (connected components) – distinct clusters are totally unrelated. Also, there are at least 3284 missing data (1-dim holes) – each hole represents that there are three 2-itemsets that surround an un-existed 3-itemsets homologously. The rest of the homology has the same idea of missing high dimensional information in one-dimensional higher itemsets.

## 5. Conclusion and Future Work

Concept mining is an important task because concepts can be treated as a catalog of all knowledge. Even though we can apply some technics such as external merge sort to the current Concept-based Search Engine to solve some problems, the search engine still needs to be improved to capture concepts more intelligently such that it should work with semantic and grammar approaches, not just a frequency-based algorithm.

More importantly, the homology analysis is a new discipline of data science. In this project, we combined homology with text, and it shows us there are hidden concepts behind the data. If we apply the same idea to another area of science, some new breakthrough may be made. However, current work is just an indicator to tell if there is and approximately how many hidden relations are there, the generators for that hidden relation are not as clearly as we saw currently, and we are still working on it.

## Appendix A

### An example of finding the Smith Normal Form of a matrix

$$R = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 2 & 4 & 5 & 6 \\ 1 & 2 & 2 & 2 & 3 \\ 3 & 2 & 3 & 2 & 1 \end{bmatrix}$$

Step 0:

From the beginning, we have $R_0 = R$, $U_0$ and $V_0$ are initialized with diagonal 1's:

$$\begin{array}{ccc} R_0 & U_0 & V_0 \\ \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 2 & 4 & 5 & 6 \\ 1 & 2 & 2 & 2 & 3 \\ 3 & 2 & 3 & 2 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Step 1:

Subtract 3 times row 1 from row 2

Subtract 1 times row 1 from row 3

Subtract 3 times row 1 from row 4

$$\begin{array}{ccc} R_1 & U_1 & V_1 \\ \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & -4 & -5 & -7 & -9 \\ 0 & 0 & -1 & -2 & -2 \\ 0 & -4 & -6 & -10 & -14 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -3 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Step 2:

Subtract 2 times column 1 from column 2

Subtract 3 times column 1 from column 3

Subtract 4 times column 1 from column 4

Subtract 5 times column 1 from column 5

$$\begin{array}{ccc} R_2 & U_2 & V_2 \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & -4 & -5 & -7 & -9 \\ 0 & 0 & -1 & -2 & -2 \\ 0 & -4 & -6 & -10 & -14 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -3 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & -2 & -3 & -4 & -5 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Step 3:

Swap rows 2 and 3

Swap columns 2 and 3

Negate row 2

Add 5 times row 2 to row 3

Add 6 times row 2 to row 4

$$\begin{array}{ccc} R_3 & U_3 & V_3 \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & 2 \\ 0 & 0 & -4 & 3 & 1 \\ 0 & 0 & -4 & 2 & 2 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 2 & 1 & -5 & 0 \\ 3 & 0 & -6 & 1 \end{bmatrix} & \begin{bmatrix} 1 & -3 & -2 & -4 & -5 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Step 4:

Subtract 2 times column 2 from column 4

Subtract 2 times column 2 from column 5

$$\begin{array}{ccc} R_4 & U_4 & V_4 \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -4 & 3 & 1 \\ 0 & 0 & -4 & 2 & -2 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 2 & 1 & -5 & 0 \\ 3 & 0 & -6 & 1 \end{bmatrix} & \begin{bmatrix} 1 & -3 & -2 & 2 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & -2 & -2 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Step 5:

Swap columns 3 and 5

Add 2 times row 3 to row 4

$$\begin{array}{ccc} R_5 & U_5 & V_5 \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & -4 \\ 0 & 0 & 0 & 8 & -12 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 2 & 1 & -5 & 0 \\ 7 & 2 & -16 & 1 \end{bmatrix} & \begin{bmatrix} 1 & -3 & 1 & 2 & -2 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & -2 & -2 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

Step 6:

Subtract 3 times column 3 from column 4

Add 4 times column 3 to column 5

$$\begin{array}{ccc} R_6 & U_6 & V_6 \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 8 & -12 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 2 & 1 & -5 & 0 \\ 7 & 2 & -16 & 1 \end{bmatrix} & \begin{bmatrix} 1 & -3 & 1 & 2 & -2 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & -2 & 4 & 8 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & -3 & 4 \end{bmatrix} \end{array}$$

Step 7:

Add 1 times column 4 to column 5

$$\begin{array}{ccc} R_7 & U_7 & V_7 \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 8 & -4 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 2 & 1 & -5 & 0 \\ 7 & 2 & -16 & 1 \end{bmatrix} & \begin{bmatrix} 1 & -3 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & -2 & 4 & -4 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & -3 & 1 \end{bmatrix} \end{array}$$

Step 8:

Swap columns 4 and 5

Negate row 4

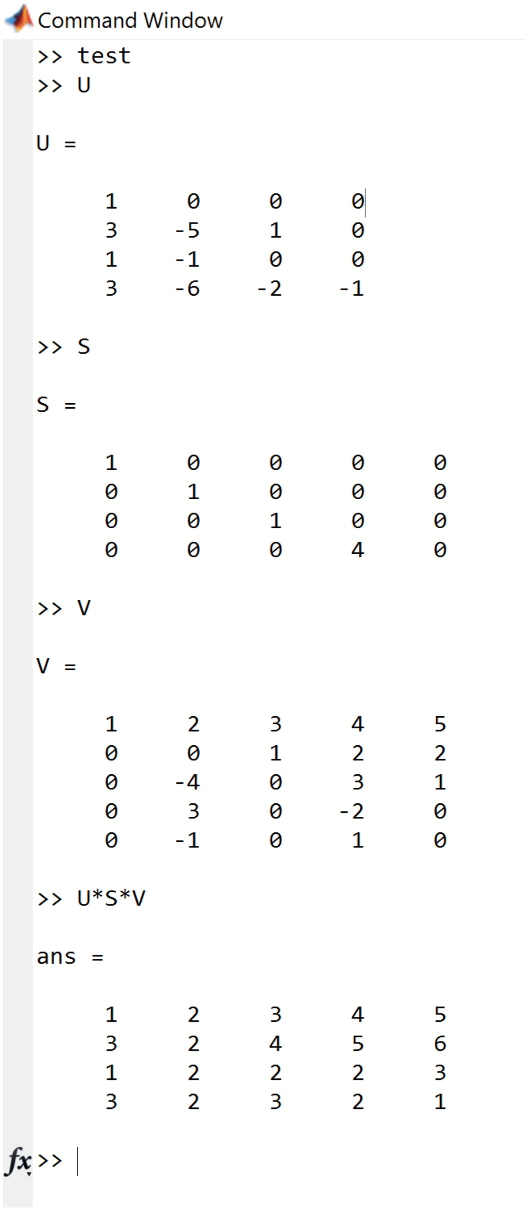
Add 2 times column 4 to column 5

$$\begin{array}{ccc} R_8 & U_8 & V_8 \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 2 & 1 & -5 & 0 \\ -7 & -2 & 16 & -1 \end{bmatrix} & \begin{bmatrix} 1 & -3 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 2 \\ 0 & 1 & -2 & -4 & -4 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & -1 \end{bmatrix} \end{array}$$

Step 9 (Last step):

$$S = R_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \end{bmatrix}, \ U = U_8^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & -5 & 1 & 0 \\ 1 & -1 & 0 & 0 \\ 3 & -6 & -2 & -1 \end{bmatrix}, \ V = V_8^{-1} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 1 & 2 & 2 \\ 0 & -4 & 0 & 3 & 1 \\ 0 & 3 & 0 & -2 & 0 \\ 0 & -1 & 0 & 1 & 0 \end{bmatrix}$$

We can verify $R = U * S * V$



```
>> test
>> U

U =

     1     0     0     0
     3    -5     1     0
     1    -1     0     0
     3    -6    -2    -1

>> S

S =

     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     4     0

>> V

V =

     1     2     3     4     5
     0     0     1     2     2
     0    -4     0     3     1
     0     3     0    -2     0
     0    -1     0     1     0

>> U*S*V

ans =

     1     2     3     4     5
     3     2     4     5     6
     1     2     2     2     3
     3     2     3     2     1

fx >> 
```

Figure 0-1

## Appendix B

### Example of Homology Group $H_1$ in $RP^2$

A chain complex is a sequence of $\mathbb{Z}$-vector spaces (Abelian groups) $C_1, C_2, \dots, C_n$, connected by homomorphism $\partial_2 : C_2 \longrightarrow C_1$, where $\partial_2$ is known as the boundary operator. The Smith Normal Form of $\partial_2$ is $\partial_2 = U_2 * \delta_2 * V_2$, where $\delta_2$ is a diagonal matrix with integers arranged in increasing order at entries. $\mathbb{Z}$-matrices in such forms permit us to compute the images and kernels easier.

Taking advantage of such special matrices, we can compute the homology groups; we will illustrate the idea of computing the $H_1$ of $RP^2$, real projective plane. Using the triangulation given in Figure 0-1, its chain complex can be denoted by $C_2 \longrightarrow C_1 \longrightarrow C_0$. Our first goal is to find a $\mathbb{Z}$-basis of $C_1$, whose basis contains the $\mathbb{Z}$-basis of $\ker(\partial_1)$, moreover the latter $\mathbb{Z}$-basis may express the $\mathbb{Z}$-basis of $im(\partial_2)$ by a diagonal matrix with integers.

$\ker(V_2 U_1 \delta_1)$ is isomorphic to $\ker(\partial_1)$ because both V and U are isomorphisms, and the same isomorphism map $im(\delta_2)$ isomorphically onto $im(\partial_2)$, so the quotient group $\frac{\ker(V_2 U_1 \delta_1)}{im(\delta_2)}$ is isomorphic to $\frac{\ker(\partial_1)}{im(\delta_2)}$. Therefore, $H_1 = \frac{\ker(\partial_1)}{im(\delta_2)} = \frac{\ker(V_2 U_1 \delta_1)}{im(\delta_2)}$, where the row vectors of $V_2 * U_1$ is the row vectors of $U_1$ represented in terms of the basis of $V_2$.

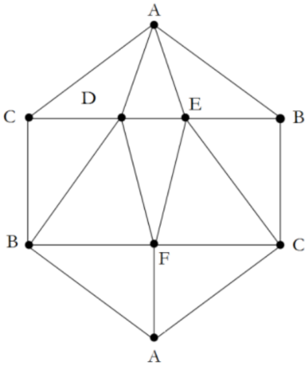


Figure 0-1

$C_2$ natural basis (geometric basis):

Δ (A, B, E), Δ (A, B, F), Δ (A, C, D), Δ (A, C, F), Δ (A, D, E), Δ (B, C, D), Δ (B, C, E), Δ (B, D, F), Δ (C, E, F), Δ (D, E, F)

$C_1$ natural basis (geometric basis):

Δ (A, B), Δ (A, C), Δ (A, D), Δ (A, E), Δ (A, F), Δ (B, C), Δ (B, D), Δ (B, E), Δ (B, F), Δ (C, D), Δ (C, E), Δ (C, F), Δ (D, E), Δ (D, F), Δ (E, F)

$C_0$ natural basis (geometric basis):

Δ(A), Δ(B), Δ(C), Δ(D), Δ(E), Δ(F)

$$\begin{tikzcd}
C_2 \arrow[r, "\partial_2"] \arrow[d, shift left, "U_2"] & C_1 \arrow[r, "\partial_1"] \arrow[d, shift left, "U_1"] & C_0 \arrow[r, "\partial_0"] \arrow[d, shift left, "U_0"] & 0 \\
N_2 \arrow[r, "\delta_2"] \arrow[u, shift left] & N_1 \arrow[r, "\delta_1"] \arrow[u, shift left, "V_2"] & N_0 \arrow[r, "\delta_0"] \arrow[u, shift left, "V_1"] & 0 \arrow[u, "V_0"]
\end{tikzcd}$$

$\partial_2$ (the $\mathbb{Z}$-linear map from $C_2$ to $C_1$)

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

According to Smith Normal Form $\partial_2 = U_2 * \delta_2 * V_2$, we can get $U_2$ and $V_2$

$U_2$ (the matrix whose row vectors form a new basis of $C_2$, called normal basis of $N_2$)

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

$\delta_2$ (the linear map $\partial_2$ from $N_2$ to $N_1$, expressed in terms of normal basis of $N_2$ and $N_1$, where $N_1$ is the row vectors of $V_2$ in terms of geometric basis).

$$\begin{matrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0
\end{matrix}$$

$V_2$ (the matrix whose row vectors form a new basis of $C_1$, called normal basis of $N_1$)

$$\begin{array}{ccccccccccccccc}
1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 1 & 1 & -1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & -2 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}$$

$\partial_1$ (the linear map from $C_1$ to $C_0$)

| | | | | | |
|---|---|---|---|---|---|
| -1 | 1 | 0 | 0 | 0 | 0 |
| -1 | 0 | 1 | 0 | 0 | 0 |
| -1 | 0 | 0 | 1 | 0 | 0 |
| -1 | 0 | 0 | 0 | 1 | 0 |
| -1 | 0 | 0 | 0 | 0 | 1 |
| 0 | -1 | 1 | 0 | 0 | 0 |
| 0 | -1 | 0 | 1 | 0 | 0 |
| 0 | -1 | 0 | 0 | 1 | 0 |
| 0 | -1 | 0 | 0 | 0 | 1 |
| 0 | 0 | -1 | 1 | 0 | 0 |
| 0 | 0 | -1 | 0 | 1 | 0 |
| 0 | 0 | -1 | 0 | 0 | 1 |
| 0 | 0 | 0 | -1 | 1 | 0 |
| 0 | 0 | 0 | -1 | 0 | 1 |
| 0 | 0 | 0 | 0 | -1 | 1 |

Again, according to Smith Normal Form $\partial_1 = U_1 * \delta_1 * V_1$, we can get $U_1$ and $V_1$

$U_1$ (the matrix whose row vectors form a new basis of $C_1$, called normal basis of $N_1{}'$)

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

$\delta_1$ (the linear map $\partial_1$ from $N_1$ to $N_0$, expressed in terms of normal basis of $N_1'$ and $N_0$, where $N_0$ is the row vectors of $V_1$ in terms of geometric basis).

$$\begin{matrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{matrix}$$

$V_1$ (the matrix whose row vectors form a new basis of $C_0$, called normal basis of $N_0$)

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

As we discussed, $H_1 = \frac{\ker(\partial_1)}{im(\delta_2)} = \frac{\ker(V_2 U_1 \delta_1)}{im(\delta_2)}$, we need to find $\ker(V_2 U_1 \delta_1)$ and $im(\delta_2)$.

From $\delta_2$, we can see the $im(\delta_2)$ is,

*AB-AE+BE*

*AC-AD+CD*

*AD-AF-CD+CF*

*AE-AF-BE+BF*

*BC-BD+CD*

*BD-BE-CD+CE*

*BE-BF-CD+CF-DE*

*-2CD+CE+CF-DE-DF*

*2CD-2CF+DE+DF+EF*

*2(CD-CF+DF)*

where the first 9 vectors are multiplied by 1, and the 10th vector is multiplied by 2.

To get the basis of $\ker(V_2 U_1 \delta_1)$, we let

$$\begin{bmatrix} x_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & x_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & x_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_{10} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_{11} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_{12} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_{13} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_{14} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_{15} \end{bmatrix} * U_2 * V_1 * \delta_1 = 0$$

Solve the equation we get

$$\begin{cases} x_1, x_2, \dots, x_{10} \in \mathbb{Z} \\ x_{11} = 0 \\ x_{12} = 0 \\ x_{13} = 0 \\ x_{14} = 0 \\ x_{15} = 0 \end{cases}$$

Therefore, we can choose the basis as follows

$x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 0, \dots, x_{10} = 0, x_{15} = 0$ as the 1st vector basis for the kernel

$x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 0, \dots, x_{10} = 0, x_{15} = 0$ as the 2nd vector basis for the kernel

…

$x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 0, \dots, x_{10} = 1, x_{15} = 0$ as the 10th vector basis for the kernel that is,

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

Where each row represents the coordinates of normal basis vector of $V_2$

Therefore, the basis of $\ker(V_2 U_1 \delta_1)$ is

*AB-AE+BE*

*AC-AD+CD*

*AD-AF-CD+CF*

*AE-AF-BE+BF*

*BC-BD+CD*

*BD-BE-CD+CE*

*BE-BF-CD+CF-DE*

*-2CD+CE+CF-DE-DF*

*2CD-2CF+DE+DF+EF*

*CD-CF+DF*

That is, $\ker(V_2 U_1 \delta_1) = \mathbb{Z}(AB - AE + BE) \oplus \mathbb{Z}(AC - AD + CD) \oplus \mathbb{Z}(AD - AF - CD + CF) \oplus \mathbb{Z}(AE - AF - BE + BF) \oplus \mathbb{Z}(BC - BD + CD) \oplus \mathbb{Z}(BD - BE - CD + CE) \oplus \mathbb{Z}(BE - BF - CD + CF - DE) \oplus \mathbb{Z}(-2CD + CE + CF - DE - DF) \oplus \mathbb{Z}(2CD - 2CF + DE + DF + EF) \oplus \mathbb{Z}(CD - CF + DF)$

Therefore, we can calculate homology group $H_1$

$$H_1 = \frac{\ker(V_2 U_1 \delta_1)}{im(\delta_2)} = \frac{\begin{array}{c} \mathbb{Z}(AB - AE + BE) \oplus \\ \mathbb{Z}(AC - AD + CD) \oplus \\ \mathbb{Z}(AD - AF - CD + CF) \oplus \\ \mathbb{Z}(AE - AF - BE + BF) \\ \mathbb{Z}(BC - BD + CD) \oplus \\ \mathbb{Z}(BD - BE - CD + CE) \oplus \\ \mathbb{Z}(BE - BF - CD + CF - DE) \oplus \\ \mathbb{Z}(-2CD + CE + CF - DE - DF) \oplus \\ \mathbb{Z}(2CD - 2CF + DE + DF + EF) \oplus \\ \mathbb{Z}(CD - CF + DF) \end{array}}{\begin{array}{c} \mathbb{Z}(AB - AE + BE) \oplus \\ \mathbb{Z}(AC - AD + CD) \oplus \\ \mathbb{Z}(AD - AF - CD + CF) \oplus \\ \mathbb{Z}(AE - AF - BE + BF) \\ \mathbb{Z}(BC - BD + CD) \oplus \\ \mathbb{Z}(BD - BE - CD + CE) \oplus \\ \mathbb{Z}(BE - BF - CD + CF - DE) \oplus \\ \mathbb{Z}(-2CD + CE + CF - DE - DF) \oplus \\ \mathbb{Z}(2CD - 2CF + DE + DF + EF) \oplus \\ \mathbb{Z}\big(2(CD - CF + DF)\big) \end{array}}$$

$$= \frac{\mathbb{Z}(AB - AE + BE)}{\mathbb{Z}(AB - AE + BE)} \oplus \ldots \oplus \frac{\mathbb{Z}(CD - CF + DF)}{\mathbb{Z}\big(2(CD - CF + DF)\big)}$$

$$= \frac{\mathbb{Z}}{\mathbb{Z}} \oplus \ldots \oplus \frac{\mathbb{Z}}{2\mathbb{Z}} = \frac{\mathbb{Z}}{2\mathbb{Z}}$$

$$r = nullity(\delta_1) = 10$$

$$t = rank(\delta_2) = 10$$

$$H_i \cong \mathbb{Z}^{r-t} \oplus (\frac{\mathbb{Z}}{b_1\mathbb{Z}}) \oplus \cdots \oplus \left(\frac{\mathbb{Z}}{b_t\mathbb{Z}}\right) = \mathbb{Z}^{10-10} \oplus (\frac{\mathbb{Z}}{\mathbb{Z}}) \oplus \cdots \oplus \left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right) = \frac{\mathbb{Z}}{2\mathbb{Z}}$$

## Bibliography

[1] Ahuja, H. K. (2015). *Clustering Web Concepts Using Algebraic Topology.* Master's Project.

[2] Darrell Allgaier, David Perkinson, Sarah Ann Stewart, John Thurber. (2004). *Homology of Simplicial Complexes.*

[3] Erickson, J. (2013). *Computational Topology* .

[4] *Homology (mathematics).* (2016, December). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Homology_(mathematics)

[5] Jonsson, J. (2011). *Introduction to simplicial homology.*

[6] Le, D. (2016). *Analyze Large Multidimensional Datasets Using.* Master's Project.

[7] Lin, T. Y., & Hsu, J. D. (2008). Knowledge Based Search Engine: Granular Computing on the Web. *IEEE / WIC / ACM International Conference on Web Intelligence.* Sydney: IEEE Computer Society.

[8] Lin, T. Y., Sutojo, A., & Hsu, J.D. (2006). Concept Analysis andWeb Clustering using Combinatorial Topology. *ICDM Workshops.* Hong Kong: IEEE Computer Society.

[9] Morandi, P. J. (2005). *The Smith Normal Form of a Matrix.*

[10] Mridul Aanjaneya; Monique Teillaud. (2007). *Triangulating the Real Projective Plane.* Sophia Antipolis.

[11] Munkres, J. R. (1984). *Elements of Algebraic Topology.* Menlo Park: Addison Wesley.

[12] Nadathur, P. (2007). *An Introduction to Homology.*

[13] Robins, V. (2000). *Computational Topology at Multiple Resolutions: Foundations and Applications to Fractals and Dynamics.* Doctor's Project.

[14] Roy, P. (2014). *Concept Based Semantic Search Engine.* Master's Project.

[15] *Simplex.* (2016). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Simplex

[16] *Simplicial complex.* (2016, December). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Simplicial_complex