



**Linnæus University**

Sweden

---

Degree project

# Visualization of Bags



*Author:* Hui Wu  
*Supervisor:* Andreas Kerren

*Date:* 2013-02-13

*Course Code:* 2DV00E, 15 credits  
*Level:* Bachelor

Department of Computer Science

## **Abstract**

The purpose of this thesis is to develop a toolkit to visualize bag calculations for teaching. We take full advantage of interaction techniques in Computer Science to achieve it, which could lead to a modern and impressive way for teaching.

In this thesis, the developed toolkit is going to show the bag calculations and corresponding animations interactively and aesthetically which make new learners easier to acquire the concept of multiset.

**Keywords:** set theory, bag, Venn-diagram, JUNG, visualization

## Table of Contents

|        |   |    |
|--------|---|----|
| 1.     | Introduction.....   | 1  |
| 1.1.   | Background .....  | 1  |
| 1.2.   | Problem .....   | 1  |
| 1.3.   | Motivation .....  | 2  |
| 1.4.   | Project Goals .....   | 2  |
| 1.5.   | Structure of the Report .....                                   | 2  |
| 2.     | Mathematical Background .....                                   | 3  |
| 2.1.   | Set.....  | 3  |
| 2.2.   | Bags.....   | 5  |
| 3.     | Related Works .....   | 9  |
| 3.1.   | Information Visualization .....                                 | 9  |
| 3.2.   | Visualization in Education Software .....                       | 10 |
| 3.3.   | Algorithm Animation .....                                       | 11 |
| 3.4.   | Tree and Hierarchies in Computer Science and Visualization..... | 12 |
| 3.4.1. | Binary Tree.....  | 13 |
| 3.4.2. | Binary Tree Traversal.....                                      | 14 |
| 3.5.   | Visualizing Trees with JUNG.....                                | 15 |
| 4.     | Visualization.....  | 18 |
| 4.1.   | Conceptual Design .....   | 18 |
| 4.2.   | Graphical User Interface .....                                  | 19 |
| 4.2.1. | The Dialog Items.....   | 19 |
| 4.2.2. | Color Legend .....  | 21 |
| 4.3.   | Interaction Design .....  | 21 |
| 4.3.1. | Interactive Buttons .....                                       | 22 |
| 4.3.2. | Brushing .....  | 22 |
| 4.4.   | Generating Tree .....   | 23 |
| 4.4.1. | Color-Coding and Animation.....                                 | 23 |
| 5.     | Implementation .....  | 25 |
| 5.1.   | Implementation Overview .....                                   | 25 |
| 5.2.   | Generate Tree .....   | 27 |
| 5.2.1. | Tree Data Structure .....                                       | 27 |
| 5.2.2. | Draw Visual Tree .....  | 28 |

|      |                                 |    |
|------|---------------------------------|----|
| 5.3. | Tree Animation.....             | 29 |
| 5.4. | Node List.....                  | 31 |
| 6.   | Conclusion and Future Work..... | 35 |
| 6.1. | Conclusion.....                 | 35 |
| 6.2. | Future work.....                | 35 |
|      | References.....                 | 37 |

# 1. Introduction

The thesis is going to develop a toolkit for teaching. It is based on combining existing mathematical theories and model technologies in Computer Science. This chapter is going to introduce all related concepts.

## 1.1. Background

Set and bag are core basic mathematical theories in this thesis. Bag is kind extension of set. The compositions of them are the same. The unique and biggest difference is the same element in a bag could appear more than once, while only once for set. For example,  $\{1,2,3\}$  is a set,  $\{1,2,3,3\}$  is a bag.

## 1.2. Problem

We are all in an “information society”, obviously this expression convey the society have an equivocal meaning. Creation, processing and transmission of kinds of information are the key factors. In this modern society, these factors are vital issues now. This derives one of the biggest tasks is how to reside them to be brief and to the point [12]. The significant nature of the information is they change our habits and behaviours. Individuals get kinds of information from the Internet, TV, radio and many other media. This leads the educationists to strongly fear that young people will be manipulated by the kinds of ways, like the enormous changes in their values, the effect among the human relations and so on. We can call this difficulty “education crisis in the information society” [13]. On the positive side, accessing to more information means more communications and more new knowledge for us. Therefore a good education should be prepared for us. Here, the major responsibility has been owned to the educationalists, which play the indisputable role since education system appeared. The education should not be only a way to inculcate the new knowledge to pupils any more, but more than a good way to arouse the study interests as well. The traditional education system is inadequate, introducing the basic sterile concepts are always the first step for new stuff. New technology floods the whole world; it keeps invading our work and life. If we could apply the new technology to education system, that would be much more attractive for education receivers. Bringing IT into the classroom, the teachers and students would be allowed to have hands-on experience and impress students more deeply. Education should give advice how IT can be used properly and what the IT want to present in front of the students.

In the thesis, the concept of bag is a kind of generalization of the notion of set. They are quite similar but also have distinct difference. To distinguish them, instantiating may be best way to introduce bag in education.

### **1.3. Motivation**

Bag could be a pure mathematics concept. It is more applied in many scientific fields to assign specify data or information, like database. Even so, the primary task is to master the operations of bag. To visualize the processing together with the result would be the best for new learners.

### **1.4. Project Goals**

Considering the advantages of visualization application, the goal in this thesis is to develop a toolkit for operations among bags.

This toolkit is a combination of visualization with processing the calculations. It is not only simply offering the results of operations, but also provides the operation processing by creating graphs and animations. This toolkit should be easy to used, the user just needs to type an expression and click buttons to visualize the operation process. The toolkit provides two modes which are automatic show and show step by step to visualize the processing. It also provides a good error message to remind the user to locate the error in the expression so the user can fix the expression easier.

The aim of this implementation is make students easier to understand what they learned by visualization. For that reason, it can be called an educational used of visualization which belongs to the topic of visualizations and animations in learning systems. With the toolkit, our goal is to create visual process models which support better and easier understanding. Especially the animations of project are used to support the directed smooth movements of graphs to explain the progressive operation [14]. We will invite some people to use the toolkit and analysis their feedback to see if our goal is achieved.

### **1.5. Structure of the Report**

The structure of this thesis will be divided into 6 chapters. The first chapter focuses on the background and problem: the application in education; introduce the motivation and the goal. Next chapter is related work, talking about all the techniques and the related concepts: InfoVis, binary tree and JUNG etc. Introducing these techniques will hopefully let readers know the blueprint of these projects and guide them to this area. The third and fourth parts are the core of whole thesis: visualization and implementation. These two chapters explain how the project processing and calculating. Lastly, the discussion and future work will be talked about. The improvements and the shortcomings will be included here.

## 2. Mathematical Background

In this chapter, we will introduce the mathematical background of set and bag. Our toolkit is used to show the calculation processing of bags, it is necessary to know the basic knowledge of it.

### 2.1. Set

Set theory in mathematics was developed by G.Cantor (1845-1918). This theory had been developed into an admirable system by him also. Till now, it is still one of the greatest creations of the human mind [1]. As a mathematics concept, set theory could be used as the foundation for all known mathematics. It can also be extended to our daily life, like a blanket of apples or a flock of birds are all sets of things. But such usage is always to be constructed as an illuminating parable only, not as a part of the theory that is being developed. In naive set theory, a set could be described as well-defined objects which we call elements or members of the set [2].

Elements or members we mentioned above are conceived. An element of a set could be single stuff or common numbers. At the same time, a set itself can also be an element of some other sets. There are full of examples of sets of sets in mathematics. For example, a circle is a set of points; the set of all circles is a nature example of a set of sets. We can also say the points are the subset of the circle. Or the circle is the superset of the points [2]. A set is considered as given when one can tell of every object whether it belongs to the set or not, the object here are set's elements. On the other hand, we completely determined all the elements belong to a set. If a set T consists of the elements a, b, c ...and of no others, then we could write  $T = \{a, b, c, \dots\}$ . We put the elements in the parentheses. a in the set is an element of T, we can also write in the form:  $a \in T$ . This relation usually is extended to sets, like the point & circle example. There is a set A, it consists of elements {a, b, c}; another set B contains elements {a, b, c, d}. Here set A could be defined as a subset of set B. For the form among sets, we have to write:  $A \subseteq B$  (or  $B \supseteq A$ ). Also if there is an arbitrary element x belongs to set A, then x must be an element of B [3].

There is a special set called null set (also named empty set). It has no element in the null set, but it is defined as the subset of any arbitrary sets [4]. One subset of the set also could be this set itself. There set A and set B, all the elements in set A belong to set B, that is  $A \subseteq B$ . At the same time, each element from set B appears in the set A also. Here we could form the similar formula:  $B \subseteq A$ . For this special situation, we could say these two sets are equal:  $A=B$ . When there is at least one element in the subset that never belongs to the set, this subset is defined proper subset [5]. For example, set A {1,2} is the subset of set B {1,2,3}. There is an element {3} from set B does not belong to set A. So set A is the proper subset of set B. The corresponding expression could be written:  $A \subset B$  or  $B \supset A$

By the mentioned example above, we can introduce a very basic but principal con-

cept of set theory, belonging and subset. Be similar to the word itself external meaning, belonging usually is used to describe the relation between two sets or between a set and an element. Element  $a$  belongs to set  $A$ . Equally, if there is a set which contains the all elements of another set, we can say the relation between them is belonging as well.

There are lots more of definitions of sets. Comparing to elementary geometry, set is analogous to this familiar axiomatic. The operations of sets are the most important parts we are going to talk about. Union (also called sum), the union among sets is the set that consists of all elements that belong to at least one of the operation sets. For set  $A, B$  we set before. The union of these sets is the set  $X$ .

$$X = A + B = \{1,2\} + \{1,2,3\} = \{1,2,3\}$$

Here, even if there are elements  $\{1\}$  and  $\{3\}$  belong to two summands, it still occurs in the sum for only once.

$$A + A = A$$

Intersection is the common part of sets. With set  $A$  and  $B$ , we can easily get  $A \cap B = A = \{1,2\}$ . Meanwhile cause set  $A \{1,2\}$  belongs to set  $B \{1,2,3\}$ , then  $A \cap B = A = \{1,2\}$ . If the intersection of two sets is empty, then we say they are disjoint\*.

Difference, the difference of two sets  $B$  and  $A$  is the set  $X$  of all those elements of set  $B$  that do not belong to  $A$ .

$$X = B - A = \{1,2,3\} - \{1,2\} = \{3\}$$

If set  $A$  belongs to set  $B$ , the difference  $X = B - A$  is also called the complement of  $A$  in  $B$  [6].

In mathematics, usually we have three major solutions to describe the set: Enumerating, Description method and Venn diagrams.

**Enumerating** is regarded as a good way bringing lots of benefits to users. It can actually show every attribute or elements of one object. For an operation, listing the all possible solutions, we can decide which result is the suitable one for the next stage. Based upon judgments, conditions and experience, we could get the final result by step and step. The most obvious advantage of this method is that final result must be correct. The application of set is to list all the elements, of course if there are some repeated elements, only list once. Like set  $\{1, 2, 3, 4\} = \{1, 1, 2, 3, 4\} = \{1, 2, 3, 3, 4\}$  is correct in Enumerating. The elements in the set here are disorder, only if they are in the set. But how to manage large number of different elements in a set is the coming problem. When there are thousands of different elements in one set, does this mean we need to list all elements one by one? So Enumerating does not fit to represent the sets so much.

**Description method** is mainly describing the relations of sets. There is a set which all elements satisfy a certain given property. In another word, we could be tempted to postulate a rule of formation for sets. If  $T$  is a property, then for any set  $X$ , there exists a set  $Y = \{x \in X : T(x)\}$  ( $x$  is the element of set  $X$ ) [6]. Intuitively, a set is a collection of all elements are the integer multiples of 3. So this set is consisted of 3, 6, 9, 12, 15, 18...

**Venn diagrams** are used to represent the logical between different collections or sets, normally they are used as draft to help deducing and understanding the relations



between sets or collections. It is ever since playing an important role in today's mathematics research. The inner area of the diagram is used to represent the sets. If two circles are crossing, that means they have public elements, or they don't have any relations. Below there are some figures showing the calculating processes with the circles.

Using Venn diagrams to express the mathematics sets' operations is a wise way. These diagrams are not only more visible, but also easy to present. The calculate relations are very clear by presenting these sets. Comparing to traditional way, visualizing the calculation processes is much more acceptable and faster to new study beginners. Figure .1 shows some basic operations of sets. The calculations processes are visualized through these circles. Visualizing sets with circles can transform the computation into attractive diagrams.

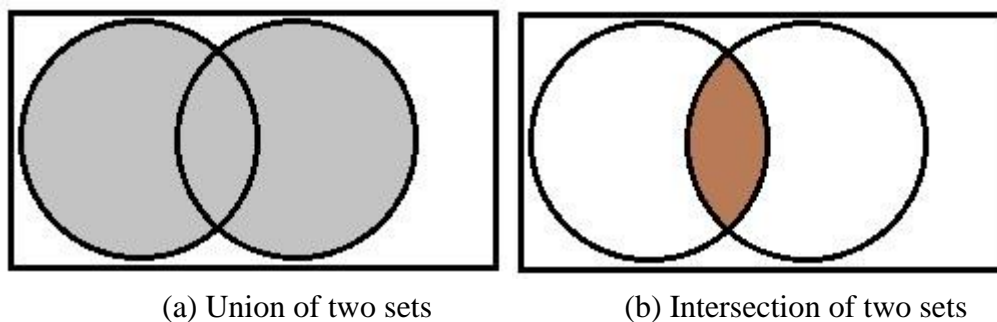


Figure 2.1: Some Basic Set Operations

From Figure .1, the union between two sets is the blue part in (a), and the intersection in (b) is the deep red part of overlapping two sets.

## 2.2. Bags

Set is an extremely basic mathematic theory as most traditional mathematics. It is a collection of distinct objects. The possible relation between them is either equality or difference. While the real situation in both science and daily life is not like this. In physical world, enormous repetition could be found everywhere [7]. If there is a random value in the set appears more than once, this is defined as *multiset* (also called *bag*).

In contrast to a set, multiset is the generalization of set. The difference is that all elements in a multiset have the same characteristic in the proper sense: multiplicity. The calculation categories and the data structure of multiset are the same to set. Usually we could say a set is a bag, but a bag may not be a set. With the operations among bags, it is quite clear to distinguish bags from sets.

Suppose there are two normal bags R and T. The values of them are {1,2,1,1} and {1,2,1,3}. Comparing to set, the unique attribute of bag is that the same element could appear in one bag, so easily get the union of these two bags:

$$\{1,2,1,1\} \cup \{1, 2,1,3\} = \{1,1,1,1,2,2,3\}$$

For intersection, one element could only appear the minimum of the number of

times of these two bags. For the same condition above, the intersection is:

$$\{1,2,1,1\} \cap \{1,2,1,3\} = \{1,1,2\}$$

The last operation difference between bag R and bag T is to get the elements as many times as they appear in bag R, minus the number of times they appear in bag T. But there is a precondition for difference; any elements here should never appear less than 0 times.

$$R-T = \{1,2,1,1\} - \{1,2,1,3\} = \{1\}$$

Nowadays, the application fields of bags are wider and wider, especially these huge data and resources. A bag can have duplicate tuples (rows). The most widely used area of multiset is Relational Database Management System (RDMS). Real RDBMSs treat relations as bags of tuples. Structured Query Language (shorted as SQL) is the most important query language for related databases to manage data in RDMSs. The results of SQL frequently contain duplicate tuples (rows) which adjust to the multiplicity of bag. So usually we also say SQL is a bag language. SQL is associated with different SQL tables. These tables (see Figure 2.2) essentially are multisets which allow duplicate elements such as rows or tuples. The duplicate rows with an ALL or DISTINCT modifier could be preserved or removed by SQL. There are two UNION statements: UNION and UNION ALL. The sample UNION could return the expressions or rows which is in either or both tables and removes redundant duplicates from the result tables. To the UNION ALL, it will save the duplicates from both tables in the result tables. EXCEPT set operate is by using the keyword MINUS to get the set difference. The principle is getting the rows which appear in the first table, while not appear in the second table. Usually SQL defines duplicate actions are based on the count of duplicate of matching rows, which make EXCEPT quite easy when none of two tables has NULLs or duplicates in it [8].

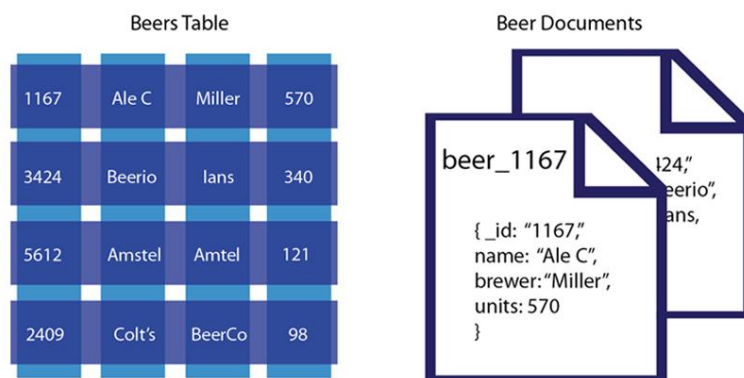


Figure 2.2: The structure of table in RDMS.

Google Books (see Figure 2.3) is a very good example of bags in database. We know every day; millions of people go to Google for searching information they need. For old typical way, usually we need to provide the title, the author or some other clue to find out the book. While for Google Books, people can get search results shown up

after entering words and phrases. Because Google will index the contents of every book after it has been submitted to the Partner Program. If there is some information in one book is relevant to the input keywords, this book will appear [9]. Before results come out, there is a track process to match the input words with the books in library.



Figure 2.3: The User Interface of Google Books

With the development of Computer Science, Google Books has become a quick service that could save both time and money. It is very convenient to end users. By entering the keywords, the results could be listed in sorted order very fast. Figure 2.4 shows lots of results, which belonged to its corresponding keywords.

Google information technology

Web Images Maps Shopping **Books** More Search tools

About 9,490,000 results (0.57 seconds)

Ad related to **information technology**

[Teknisk dokumentation | dokumentera.se](#)  
[www.dokumentera.se/](http://www.dokumentera.se/)  
 Dokumentera har gedigen erfarenhet av teknisk dokumentation.

**Information Technology**  
[books.google.com/books?isbn=1583403299](https://books.google.com/books?isbn=1583403299)  
 Pennie Stoyles, Peter Pentland, David Demant - 2003 - Preview - More editions  
 Discusses two sides of issues related to information technology--how carefully the Internet should be controlled, whether information on the Internet should be subject to copyright laws, and how much information should remain in electronic ...

**The Physics of Information Technology**  
[books.google.com/books?isbn=0521580447](https://books.google.com/books?isbn=0521580447)  
 Neil Gershenfeld - 2000 - Preview - More editions  
 This self-contained volume will help both physical scientists and computer scientists see beyond the conventional division between hardware and software to understand the implications of physical theory for information manipulation.

**Understanding Information Technology**  
[books.google.com/books?isbn=0748736093](https://books.google.com/books?isbn=0748736093)  
 Stephen Doyle - 2000 - Preview  
 This student textbook is written for the revised AS and A Level syllabuses, BTEC, GNVQ and first year degree Information Technology courses.

Figure 2.4: The Searching Result, taken from [10]

As the most important application of bag, database is considerable. Stop rum-maging information through piles of paperwork; uses just need to enter specific key-words to recall the information in database. Instead of new solution, we could not on-ly save time and money, but also high efficiency to keep away redundant information [11].

### 3. Related Works

This chapter is about the introductions and explanations of both concepts and techniques that will be used in the project: definition of Information Visualization, what graph is and the usage of the binary tree.

#### 3.1. Information Visualization

Visualization is a part of our daily life. From the weather map on TV to the vivid computer graphics in many applications are all the examples of visualization. Moreover, the definition about visualization has been brought out in many different ways. Informally, we can say visualization is a wise way to transform the data and information into pictures. The result is an attractive and effective medium for communicating complex and huge data or information [15].

Inspired by the rapid development of the software industry, people devote more and more effort and money to the different applications of Information Visualization. The applied fields of Information Visualization come to medicine, geography, biology, business and so on [16]. Compared with traditional computer graphic like 2D and 3D, Information Visualization cannot only show both static and dynamic data, but also point out the correlations even evaluate the changing trend.

The data flow pipeline below could explain the principle of Information Visualization.

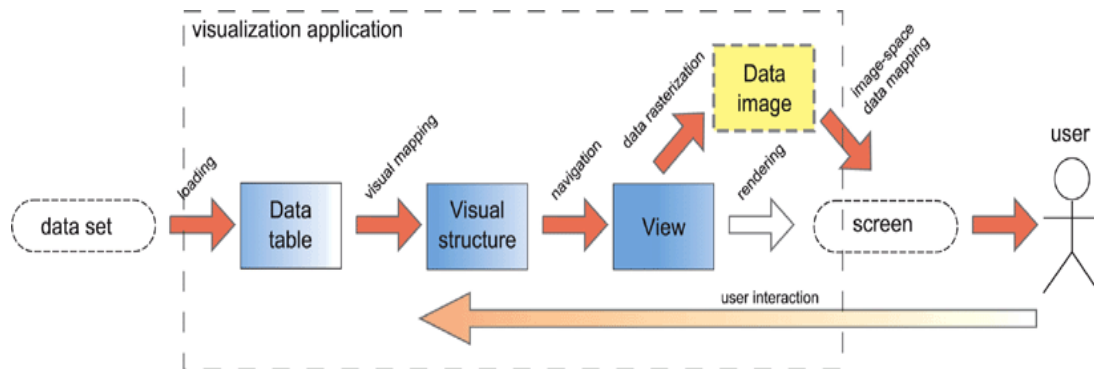


Figure 3.1: Classic InfoVis Pipeline with Transformations and States, taken from [17]

The refinement of the typically traditional Information Visualization data flow pipeline in Figure 3.1 shows the transforming steps from the raw data to the final visual views to users. Also include the completed complex human interactions in every visible and invisible part.

The obvious advantages are performance improvements. Moreover, by this pipeline method we could also use for dynamic queries with adaptive colors and scale

mappings. Particularly, this image-space step is conceptually consistent with existing visualization systems to present in all visualizations.

With the pipeline model, the raw data should be firstly filtered or transformed into a usable form. It is vital to make sure if the data could be mapped visually. Deal with different conditions is the next step, including input in error or missing values. To eliminate the data, interpolate the missing values and reenter are basic ways. The task after the modification is to decide what the data is used for. The last step is to map or visual the data to an image, which is inseparable and directed with an Application Programmers Interface (API). Usually there are quite lots of techniques in this part which are based on the library. How to get different visual layouts is an example in this thesis. This part will be discussed in the following chapters.

### **3.2. Visualization in Education Software**

As one of the most emerging area in Computer Science, Information Visualization has broad prospects. The usage and applications have been developed in the past decades. For example, TV and radio can only output the information to audience in one way, this means you can just read and look but not interactive. Comparing to traditional multimedia, for example video, visualization could provide an interactive platform. Users can choose different ways to present the result. With the development of fundamental techniques in visualization, many toolkits creation in related areas become possible. One of the most emerging areas is education. More and more educational software have been developed and applied in education with the improvement in Computer Science [18].

For lots of abstract mathematics theories and algorithms, they are hard to describe with some words or static diagrams. Meanwhile people have noted the role of computing in rebuilding the concepts of mathematics. By taking the power of generating mathematics graphics, computers acts as both direct and concrete part in visualization [19]. Visualizations based on the fundamental techniques have been developed to support learners a better view to understand.

There is a representative example: how the automatic generation and animations for visualizations can be used for education software in Computer Science and related fields. The idea does not require learners to know the graphs very well. The proceeding is separated as different steps. Specify a slightly formal computational model first; combining this model with a specified visualization system could automatically generate an interactive visualization. This generated visualization cooperates with arbitrary input data in the simulation of underlying model to get the final view. By rebuilding the models frequently, the learners can get deeper impression. In Figure , an educational software system called GANIFA is shown. This system is an electronic, HTML-based textbook on the theory of finite automata and their generation from regular expressions. GANIFA can be locally used as well as via the Internet [20].

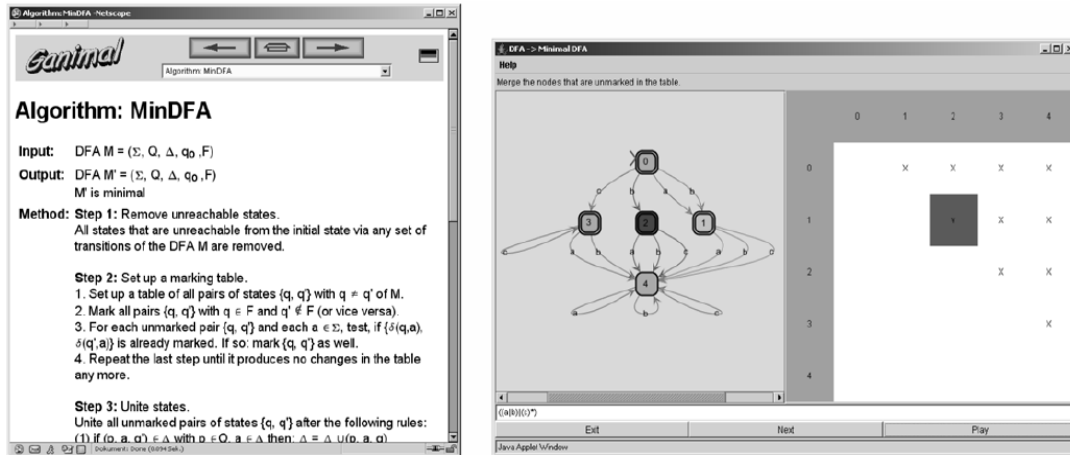


Figure 3.2: HTML-based, textual description of an algorithm on the left and corresponding interactive animation on the right, taken from [20]

The prospect of mathematics visualization in learning system concludes visualizations and animations. They are similar but different. Both of them show us clear and easily understanding representations of abstract complex aspects of mathematics.

### 3.3. Algorithm Animation

With the development of computer technology, computer scientists and programmers use animation frequently to make the program itself more impressive. The same applies to algorithms. By visualizing the data and the operations of the algorithm, it provides a clearer and more understandable idea of the algorithm.

Algorithm animation is used to simplify information about an algorithm process by creating a visual effect of its execution. Typically this is done by creating links between consecutive points in the algorithm. Thereafter these points are animated. This can be done in several different ways and with many different methods, out of which the following are of particular importance according to Kerren & Stasko [21].

Event driven algorithm animation has the user specify essential points in the algorithm. When each of these points is reached during the execution they are animated, thus giving an overview of how the algorithm is calculated.

Another way to specify the mapping between program and visualization is state driven. It constructs the mapping before the program executes to get an alternative approach. Visual programming is another common way to specify algorithm animation by visualizing executive program commands and statements with specific visual notations. The simplest way to specify the algorithm animation is automatic animation. As its name says, get the animation generated automatically.

The different techniques classified above are commonly used for connecting the algorithm to visualization. However, for an algorithm animation there is not only one dimension. Here we are going to introduce another vital dimension: visualization technique.

Nowadays there are three quite new visual aspects of algorithm animation: 3D

Algorithm Animation, Auralization, and Web Deployment. All of them should display the information that you want to show and how it works.

3D Algorithm Animation has a very distinct advantage: “the third dimension can be used for capturing time (history), uniting multiple views, and displaying additional information” [21].

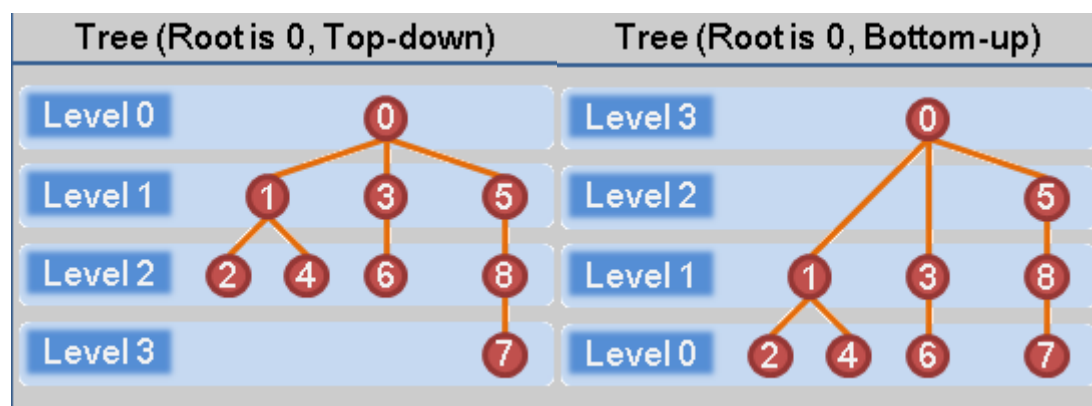
Auralization is a quite new idea to achieve the algorithm animation. By carefully mapping registers to notes, and slowing the tempo to a human timescale, the result is a cacophonous machine that offers a glimpse into the operation of various programs. We might find the resulting minimalist “music” insightful, entertaining...or maybe just incredibly grating [22].

Web Deployment is an extensive definition. All activities that make a software system available could be named Web Deployment. This solution of algorithm animation does not specify solely one activity like 3D Algorithm Animation or Auralization. As it continues to grow it becomes more developed [22].

### 3.4. Tree and Hierarchies in Computer Science and Visualization

In Computer Science, tree is kind of abstract data structure which consists of specified values or attributes in one node (or more than one node). In the most common sense the specified definition of tree, it is a set contains at least one node which forms a hierarchical structure. The typical characteristics of tree include: root node can have 0 sub nodes or much more than one. For each sub node, it can only have one parent node. One tree could be structured from several sub trees. For one tree, it can be classified undirected and directed tree. Meanwhile lots of concepts are referenced in one tree, including node, level, root, leaf, parent & child and so on.

In Figure 3.3, (a) and (b) show two opposite directed levels in a same tree. But they have the same root and the amount level. For the node labeled 1 in (a), it is the parent of node labeled 2 and 4. Meanwhile it is the child of node labeled 0 (also the root of the tree). Here labeled 2, 4, 6, 7 are the leaves of the tree.



(a) Top-down

(b) Bottom-up

Figure 3.3: Undirected Tree, taken from [23]

For directed tree shown in Figure 3.4, there are two complete opposite directions.



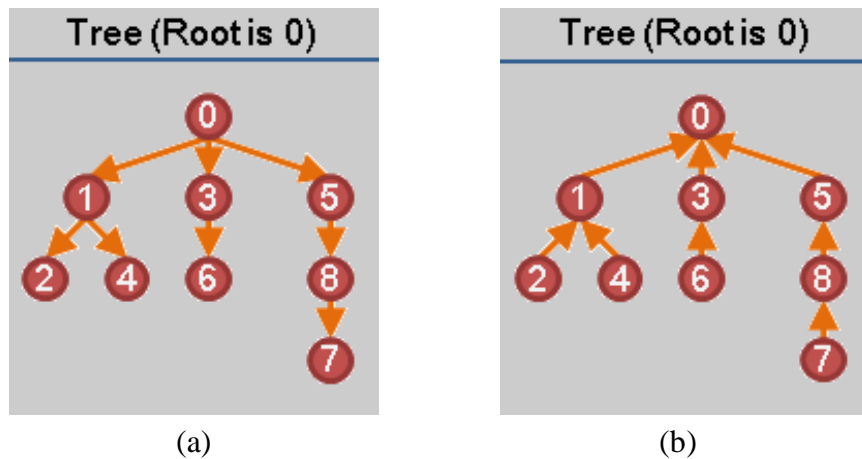


Figure 3.4: Directed Tree, taken from [23]

### 3.4.1. Binary Tree

A binary tree is a specified kind of tree. For all nodes in binary tree, they have two child nodes at most. Usually these two child nodes are positioned in “left” and “right” side of their parent node. According to judging how many children one node have, binary tree can be still classified.

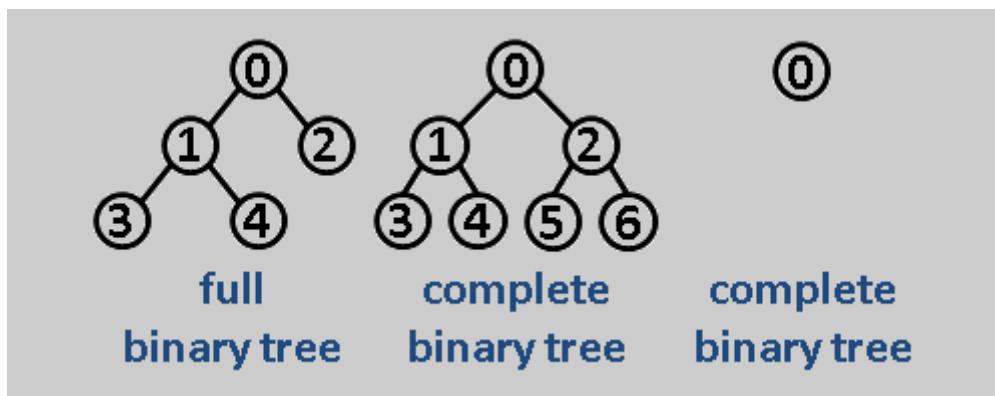


Figure 3.5: Binary Tree Classifications, taken from [23]

In Figure 3.5, it displays the classifications with examples for different binary trees: full binary tree and complete binary tree.

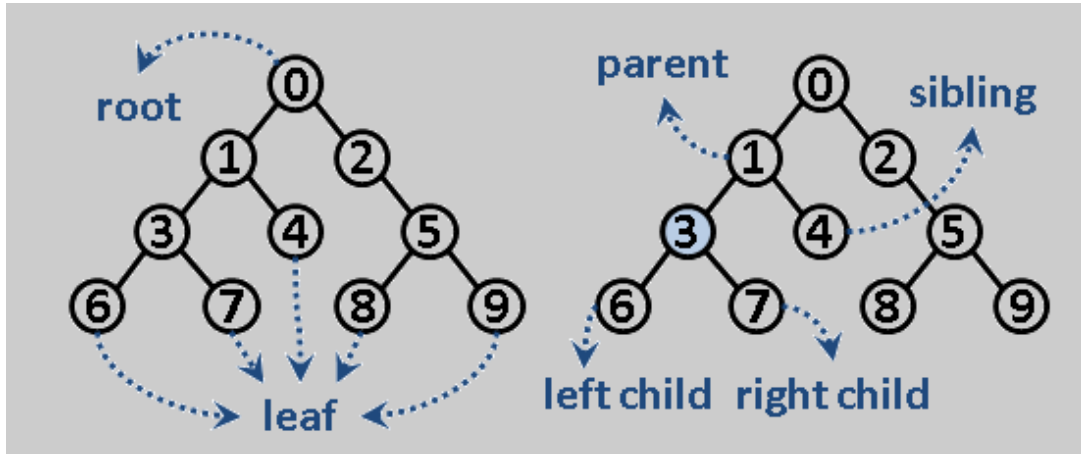


Figure 3.6: Binary Tree Components, taken from [23]

For arbitrary binary tree, attributes are the same to normal tree. Leaf, parent & child and root are basic properties. In Figure 3.6, the root is node labeled 0. It can only two children in its both sides: left child (node labeled 1) and right child (node labeled 2). For Node 1, it is the parent of Node 3 and 4. Node 4 is a leaf without child. Similar to all nodes in this binary tree, easily can get Node 4, 6, 7, 8, and 9 are leave of this binary tree.

### 3.4.2. Binary Tree Traversal

In this thesis, the basic solution idea is from the original binary tree traversal. First split the text expression into single tokens, assign these meaningful tokens to the nodes and leaves in a binary tree from bottom level to up level. Here we exploit the binary tree traversal theory with the bag algorithm to do the operations from the bottom level of the binary tree.

Binary tree traversal can be simplified: recursively visit the nodes and examine the corresponding values in both left and right subtrees of the root [23]. In this project, by splitting the input text expression into individual tokens, these symbols will be assigned their specific meanings. Imagine put these tokens in different nodes to form different subitems which could be the compositions of subtrees. Push the individual subtrees into a whole structured binary tree. After traversing the tree step by step and do the algorithm operations, the final root will be the result we need.

There are three typical different traversal ways: Pre-Order, In-Order and Post-Order traversal. The sequence of Pre-Order traversal is from root to left child, lastly right child. For In-Order traversal, the traversal sequence is left child, root and right child. The third traversal begins on the left side, ends on right side.

In Figure 3.7 below, it is a completed example to explain all the different tree traversals clearly.

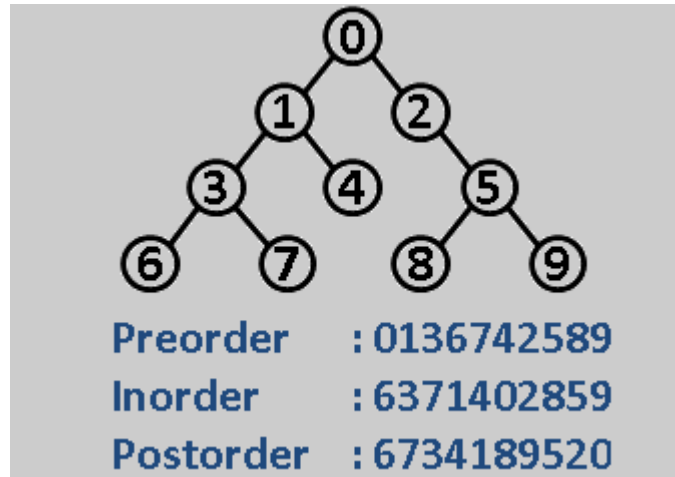


Figure 3.7: Tree Traversal, taken from [23]

Based on an existing binary tree in Figure 3.7, the Preorder Traversal begins from the top to left, ends on the right. Inorder Traversal is based on left-top-right rule. Postorder Traversal is starting with leaf from left to right and finish on the top.

### 3.5. Visualizing Trees with JUNG

JUNG is short for Java Universal Network/ Graph framework. With the high efficiency, this software library has been applied in increasing fields, especially information visualization. It can provide a language which is common and extensible to analyze, model and visual the data. Display the data as more understanding ways: graph or network. JUNG allows itself based applications to make use of the extensive built in capabilities of the java API. In one word, JUNG is a very good tool to draw graph or network [24] [25].

The basic interface of the graph structure JUNG provides is  $\text{Graph}\langle V, E \rangle$ . This is the most common sense of the definition of graph. For one pair  $\text{Graph}\langle V, E \rangle$ ,  $V$  is a set of vertices or nodes and  $E$  is a set of edges or lines. When creating a graph, add or delete for both nodes and edges become an easy job.

The most striking aspect of JUNG is the visual performance. The basic module of JUNG visualization is a subclass of Swing JPanel which called BasicVisualizationServer. Layout is the most important part in BasicVisualizationServer. The implementation for Layout actually is to “tell” JUNG how to draw nodes and edges. The message from Layout to JUNG concludes the position of nodes in a coordinate plane. Usually in order to visualizing the graph, there should be 4 steps [25]. Suppose for a circle Layout:

1: Initialize graph  $g$ ;

2: Create the Layout object with graph  $g$ ;

```
Layout<Integer, String> layout = new CircleLayout(g);
```

3: Create object BasicVisualizationServer with Layout object;

```
BasicVisualizationServer<String, String> vv = new BasicVisualizationServer<String,
```

```
String> (layout);
```

#### 4: Display the BasicVisualizationServer

```
frame.getContentPane().add(vv);  
frame.pack();  
frame.setVisible(true);
```

Figure 3.8 is the simplest graph view from the circle Layout above.

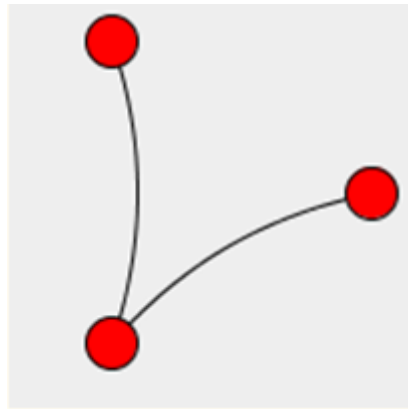
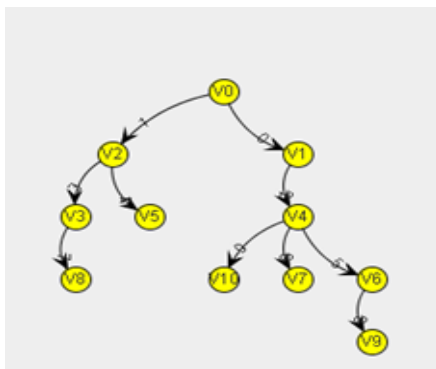


Figure 3.8: The Sample Graph View, taken from [25]

To implement a more picturesque graph, there are still lots to do in Figure 3.8. Like the descriptions of the vertices and the lines and change the color and shape.

A tree is a special case of a graph. So, JUNG provides a dedicated interface to implement tree and its internal algorithm. Take `DelegateTree` for example, it brings about the tree interface based on directed graph. The common used functions in `TreeLayout` include:

```
public boolean addVertex(V vertex): add root node;  
public boolean addChild(E edge, V parent, V child): add child node;  
public boolean addEdge(E e, V v1, V v2)
```



(a)

```
DelegateTree<String,Integer> g = new DelegateTree<String,Integer>();  
g.addVertex("V0");  
g.addEdge(edgeFactory.create(), "V0", "V1");  
g.addEdge(edgeFactory.create(), "V0", "V2");  
g.addEdge(edgeFactory.create(), "V1", "V4");  
g.addEdge(edgeFactory.create(), "V2", "V3");  
g.addEdge(edgeFactory.create(), "V2", "V5");  
g.addEdge(edgeFactory.create(), "V4", "V6");  
g.addEdge(edgeFactory.create(), "V4", "V7");  
g.addEdge(edgeFactory.create(), "V4", "V10");  
g.addEdge(edgeFactory.create(), "V3", "V8");  
g.addEdge(edgeFactory.create(), "V6", "V9");  
g.addEdge(edgeFactory.create(), "V4", "V10");  
TreeLayout layout = new TreeLayout<String,Integer>(g);
```

(b)

Figure 3.9: Tree Layout, taken from [25]

Figure 3.9 is a specific example of DelegateTree. The main codes provide shows the dedicated interface for tree drawing is really not complicated any more. And the TreeLayout on the right hand looks concise and clear [25].

## 4. Visualization

The solution for this project visualization will be introduced in this chapter. Like the conceptual design for panel and interaction.

### 4.1. Conceptual Design

In this thesis, the general design includes the interaction and layout part. The input of the raw data and to show the step animations are the major interactive parts. To show the tree and the animation are important visualizations for the user who uses the tool for education. The general design flow should be like in Figure 4.1.

Check the validity of the input expression is the first step and this would be the only job during this step. It is an interaction between end user and the computer. Tree generating is the second main part. It can be divided into two parts: abstract structure generating and the visual structure generating. By the way all visual parts could be the layout part too. Next is to show the animation on the generated visual tree to get the distinct idea of bag calculations. At last show the final result on the panel.

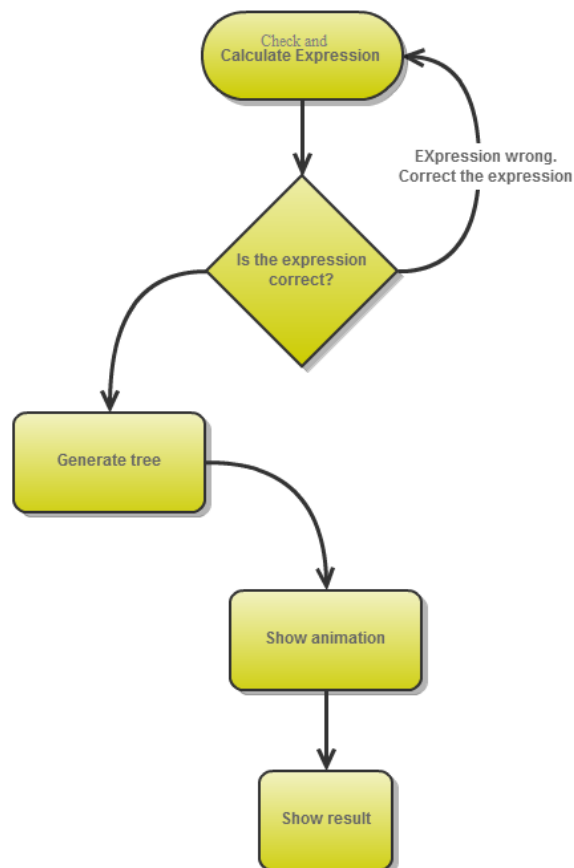


Figure 4.1: General Design Flow

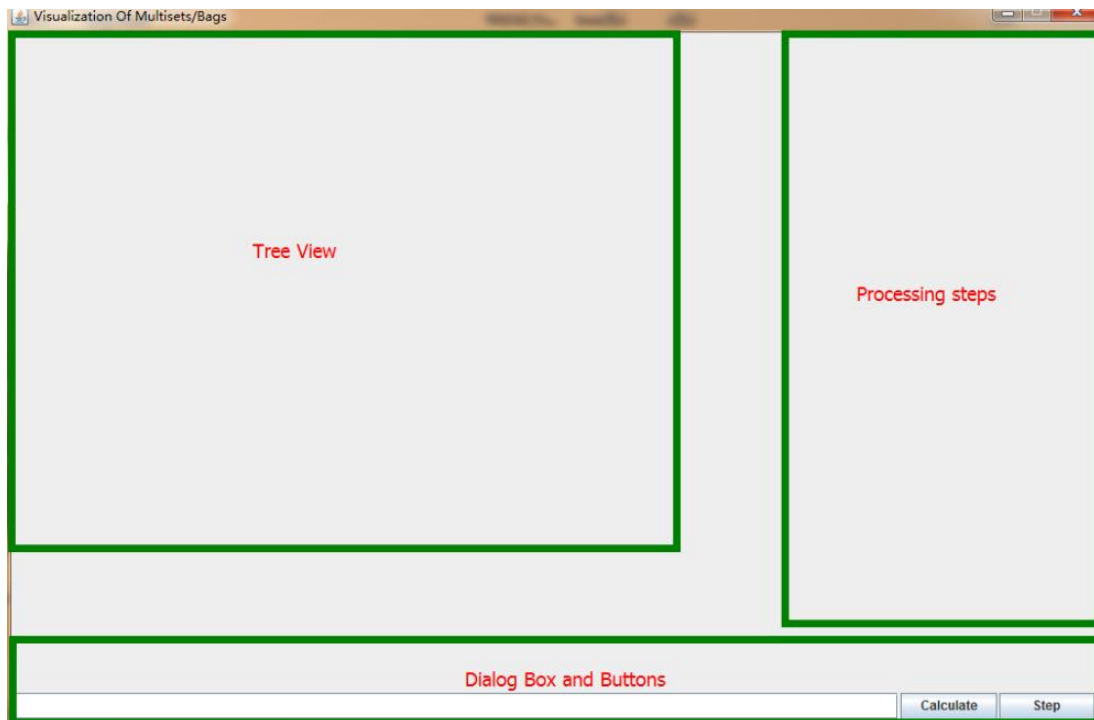


Figure 4.2: The Graphical User Interface

## 4.2. Graphical User Interface

The initial user interface is quite simple. The whole visual view panel has been divided into three major parts: tree view panel, processing step list and the dialog box for raw data (see Figure 4.2)

### 4.2.1. The Dialog Items

These three parts of layout occupy the most space of the panel to show the processing procedure and steps. On the left-top in Tree View part, there are two menus. The 'File' menu only includes one item 'Exit' to exit this program. Another menu with two items, they are named 'Help' and 'About' (see Figure 4.3).

For the 'Help' item in 'Help' menu, it describes the possible syntax of the expressions as well as the evaluation order. It also provides an example show users (see Figure 4.4).

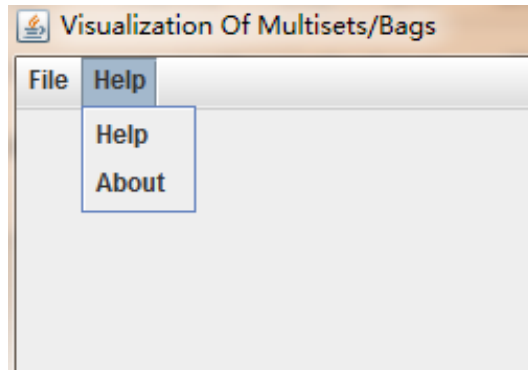


Figure 4.3: Menus and their items on Tree View Panel

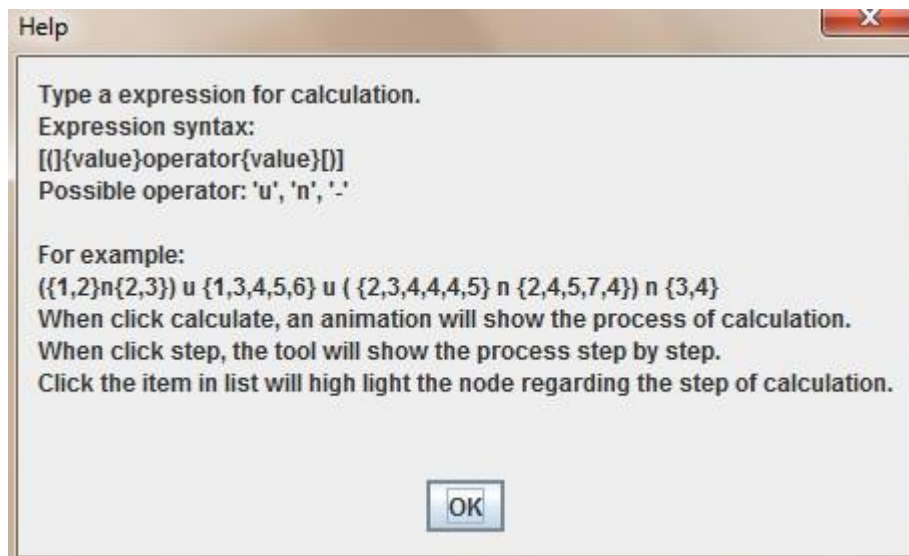


Figure 4.4: 'Help' Dialog

The 'About' item is showing a brief dialog with the purpose of this tool and its developer along with the supervisor. Also the ISOVIS Group at Linnaeus University with URL as address (see Figure 4.5).

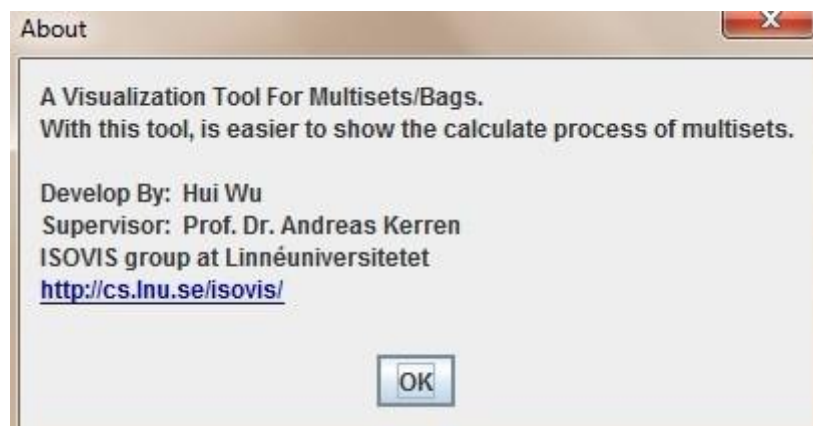


Figure 4.5: 'About' Dialog



For the dialog part, there are two different but related buttons. These buttons are important for interaction. The first button, 'Calculates', is used to start calculating automatically until the final result is computed. In addition, the calculate procedure will be shown in the Processing part in the right side of the panel. Another button is to get each calculated step for explaining it clearly to the user. By pressing once, we can get one step and then stop automatically until we press the button again. Meanwhile on the Processing Steps panel, the corresponding step appears as well.

#### 4.2.2. Color Legend

The tree nodes should be colored not only for aesthetics, but also to distinguish between the different kinds of objects. In this thesis, for the visual tree, we use yellow, red, green and grey to represent different types of tree components. Without the instruction of these color application, the user can get confused. So put a color legend on the panel is necessary.

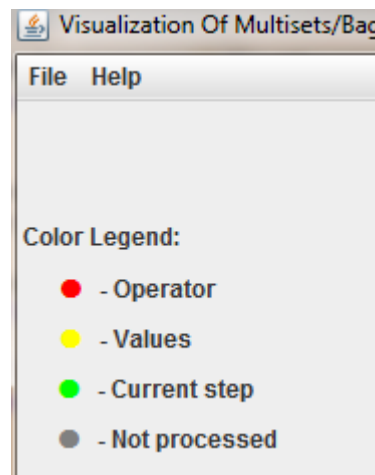


Figure 4.6: Color Legend

As Figure 4.6 shows, a red dot stands for 'Operator', yellow is for 'Value', current values are drawn in green; color grey means this object is not processed or the program suspends now.

### 4.3. Interaction Design

Interaction design patterns are used to solve the problems in specific contexts. By modeling the behavior, they make some abstract processing very understandable. These interaction models cannot only generate an interface for end users, but also provide an interactive implementation to accomplish the tasks [26].

As educational software, this project focuses on communications between the end user and the computer. This leads us to another related field, which is associated with Information Visualization: Human Computer Interaction (see Figure 4.7). Human Computer Interaction is exploring how computers and human work together to make

software and computers easier to study by building interfaces [27].

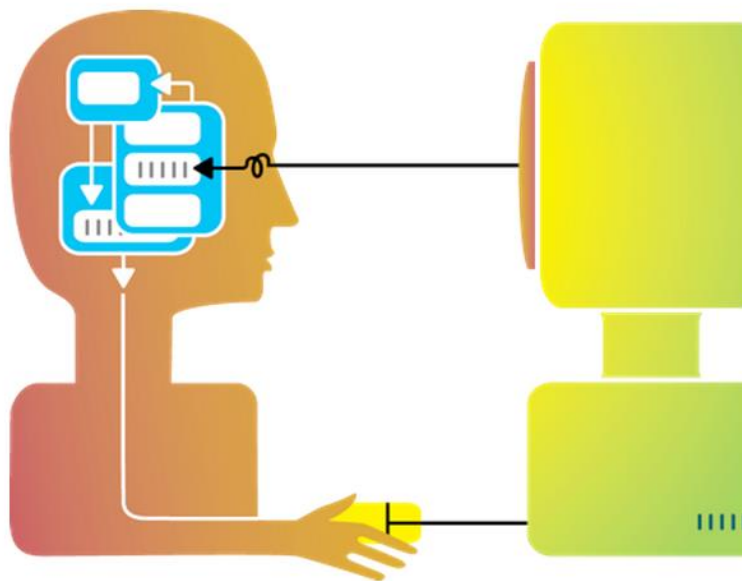


Figure 4.7: Human Computer Interaction, taken from [27]

#### 4.3.1. Interactive Buttons

On the whole panel, there are two buttons: 'Calculate' and 'Step'. Definitely these buttons are controlled by end users. As the button names suggest, 'Calculate' is to begin the calculation after the input of the verified expression. 'Step' is to show the whole calculation procedure step by step. These buttons are going to be set unavailable when the program is running or calculating unless it is finished. Only at the end, the buttons will be available again. With this design, we take how to avoid calculated errors in continuous processes into consideration.

#### 4.3.2. Brushing

Brushing is a process which could select data items in a visual representation; it is one of the interactive technologies. By highlighting chosen data items, the user gains insight into the calculation steps [28].

For the graph shown in Figure 4.8, the yellow node is the only part be highlighted. It is colored to emphasize information. This also leads to another similar way to do the brushing. For the thesis, we color all nodes gray and then step by step color the processed nodes yellow or red to show the process. The design of this step is really focus on the human interaction in a visual representation.

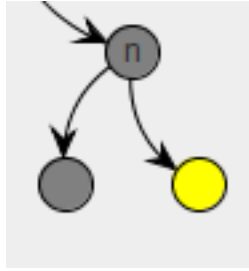


Figure 4.8: Step Processes

#### 4.4. Generating Tree

The visual tree will be generated by traversing the internal tree. By drawing the edges between parent nodes and children nodes which generated by the same rules as internal tree, an initial visual tree will be shown in front of end users (see Figure 4.9).

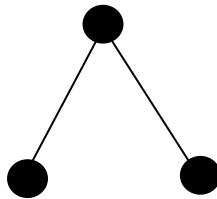


Figure 4.9: Internal Generated Tree

##### 4.4.1. Color-Coding and Animation

In Figure 4.9, it is the initial basic tree view. Color-coding for the nodes is significant and necessary. This procedure can be easily achieved to help us to distinguish the 'father' and 'children'.

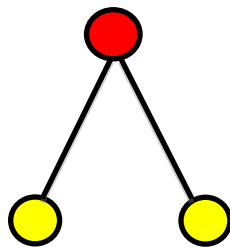


Figure 4.10: Color-coding Tree

In Figure 4.10, we use yellow to color the children nodes, and red for parent node.

In order to explain the clear idea of the calculation, we are going to use animation to show the calculating steps. This animation is shown by a kind of color coding as well. By changing the color of nodes the program show us where the specified calculating steps are. We define red as an operator process and yellow is representing the value of result. In this thesis, the idea of animation is changing the colors of nodes to

green to represent the running procedure. Meanwhile there is always a bounding box keeping track of the processed bags or expressions.

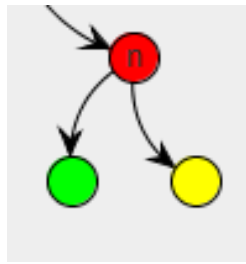


Figure 4.11: Animation: Color-Coding

The solution of showing the calculated processes is automatically achieved from the beginning. This leads to a problem that user would like to stop in a process and explore it. In response to the problem, we design the 'steps' showing at the same time.

# 5. Implementation

This part will focus on explaining how the ideas and algorithm in Chapter 3 are implemented with the software package JUNG that introduced in section 3.5.

## 5.1. Implementation Overview

With the general design flow shown in Section 4.1, it is clear that check the input expression is the first task. All expressions should be constituted by bags and operators. When a user input the wrong expression, the tool should not work and point it out.

As referenced, the data source of the thesis is an expression which is formed with bags. First, we have to ensure the input expression is legal. The union calculation between bag  $\{1,4,7\}$  and an illegal data  $(6,9,0)$  will show us how to find out the wrong expression. After pressing the 'calculate' button, there will be a dialog box warning end user to notice the errors. Any errors like the mismatching of brackets or parentheses will lead to this visual dialog box (see Figure 5.1).

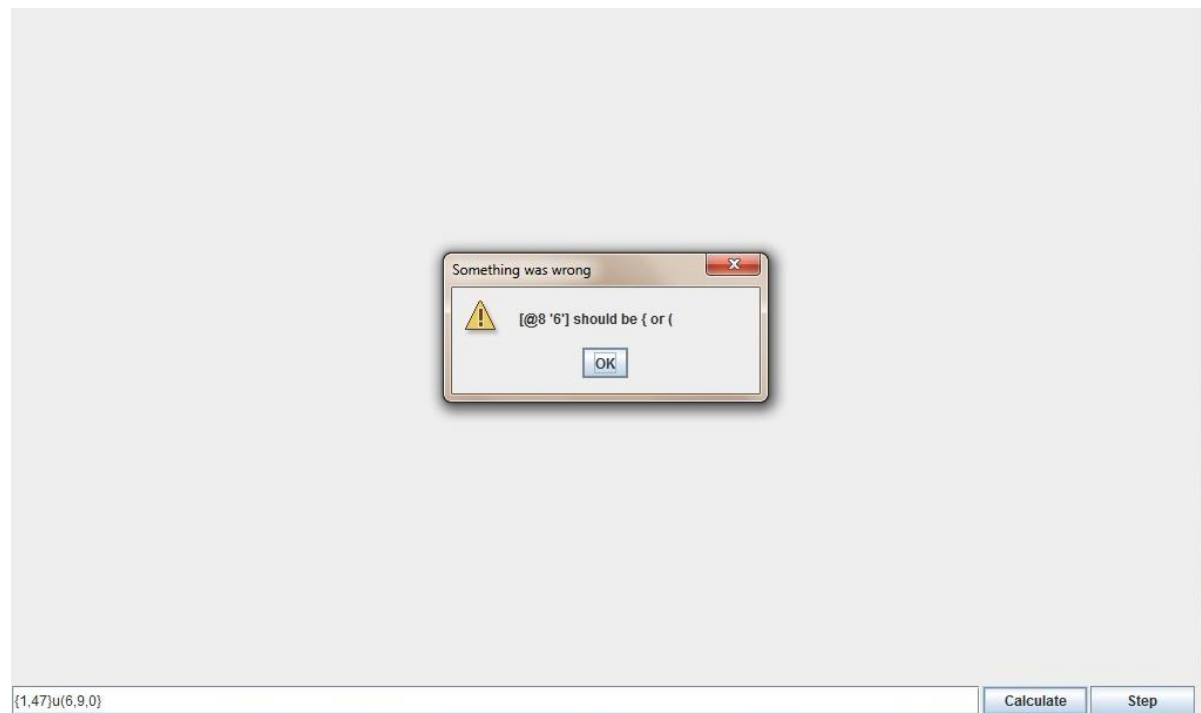


Figure 5.1: Error Warning

For the error message in the dialog box above, it says the 8th sign '6' is wrong. The long sign of '['@8'6]' means the error's position is at the 8th space of the segments of the expression. The left part of the warn message is pointing out the error.

For the expression, to make sure if it is correct, we need a variable to store the state of each character which we call 'machinestate'. Once 'machinestate' does not match the defined state value, then it will be judged illegal. For example, there is an

expression {1,0} u {1,3}. Firstly we parse it character by character, until it is parsed to '0', everything goes well. For the 4th character ')', 'machinestate' does not match with the defined state which will inform user this is a wrong expression.

```
int machinestate = 0; // 0=next ( or { 1=next , or } 11=next cannot , (
// { } ) 2=next operator u or n or - or )
```

Figure 5.2: Initialize a Machine State

Like shown in the code, 'machinestate' = 0 means next symbol must be '(' or '{'. 'machinestate' = 1 means next symbol should be a comma or '}'. So do 'machinestate' = 11 and 'machinestate' = 2. They all have been defined their different values.

Once 'machinestate' has been initialized, as shown in Figure 5.2. The initial value is set to 0. State 0 means that the next symbol is '(' or '{'. In state 1, the next symbol should be comma ',' or '}'. State 11 means that the next symbol is not allowed following a comma. These symbols are invalid for that case: ',', '(', ')', '{', '}'. State 2 means that the next symbol is an operator, like 'u', '-', 'n' or ')' (see Figure 5.3).

```
if (machinestate == 11)
    throw new SyntaxErrorException(i, exp[i],
        "cannot appear follow a comma");
if (machinestate == 2)
    throw new SyntaxErrorException(i, exp[i], "missing {");
if (machinestate == 0)
    throw new SyntaxErrorException(i, exp[i], "missing {");
machinestate = 2;
break;
```

Figure 5.3: State Value

When start checking the expression, for each symbol, assign the state machine one state value. If the next symbol is not matched the defined state value, then the provided expression is wrong. The check is continued in that way until a '(' is found. Then we start a subtree by increasing the value of 'bracketdeep'.

For every ')' we leave the subtree by decreasing 'bracketdeep' (see Figure 5.4). If finish checking all symbols and find the state value of 'bracketdeep' is not 0, it means this expression is illegal.

Depending on the value of 'bracketdeep', it can always point out how many pairs of brackets included in this expression. When there is a left bracket exit, there must be a corresponding right bracket. Only if all expressions obey this rule, it may be a correct one.

```
int bracketdeep = 0; // for (
```

Figure 5.4: Check the Completeness of Bracket

## 5.2. Generate Tree

After finishing the raw data input and check, now it is turn to generate the tree with this verified expression. The assignment activity is ongoing as well.

### 5.2.1. Tree Data Structure

For one expression in the dialog box in visual panel, we know the computer cannot recognize all these symbols immediately. Here we need to split the expression into tokens, and then assign them unique attributes or meanings.

Bags compose expressions that are connected by operators. Before assign activity, splitting the expression into tokens. For each token, it has specified function to generate the tree. For example, there is a common bag expression:  $\{1,2\}u\{3\}$ . There are 9 elements in this expression (see Figure 5.5) Encapsulate these tokens and now it is time to generate the tree.

| Expression      | Tokens |   |   |   |   |   |   |   |   |
|-----------------|--------|---|---|---|---|---|---|---|---|
| $\{1,2\}u\{3\}$ | {      | 1 | , | 2 | } | u | { | 3 | } |

Figure 5.5: Tokens from Expression

Generating abstract tree actually is a process of read-in the string of expression. For the example above, the process begins with reading “{”. Continue reading the string until get the first “}”, encapsulate all elements between “{” and “}” including them as a variant. Here this variant is supposed to be the child node of the abstract generating tree. In a similar way to get another child node, the operator “u” will be the parent node. So to speak, the bags are represented by children nodes and parent node for operator. Draw the abstract edges to connect the parent node and children nodes. Then the internal tree data structure has been built. This internal abstract tree is used for constituting its corresponding visual tree (see Figure 5.6).

```

private Point generateTree(String exp) {
    Point cur = null;
    Point tmp = null;
    exp = exp.trim();
    int pos;
    while (exp.length() > 0) {
        char x = exp.charAt(0); // Get the first character
        switch (x) {
            case '{':
                pos = exp.indexOf('}'); // Get the correspond index.
                tmp = new Point(exp.substring(1, pos)); // Create a new point,
                // and store the value

                if (cur == null) // Not processing before.
                    cur = tmp;
                else {
                    cur.setRight(tmp);
                    tmp.setParent(cur);
                }
                exp = exp.substring(pos + 1).trim(); // Get the remain string.
                break;
            case '(':
                pos = searchPairBracket(exp); // Get correspond bracket.
                tmp = generateTree(exp.substring(1, pos)); // Create a sub tree.
                if (cur == null)
                    cur = tmp; // Not processing before.
                else {
                    cur.setRight(tmp);
                    tmp.setParent(cur);
                }
                exp = exp.substring(pos + 1).trim();
                break;
            case 'n':
            case 'u':
            case '-':
                // These are the operator, they are always as parents.
                tmp = new Point(x);
                tmp.setLeft(cur);
                cur.setParent(tmp);
                cur = tmp;
                exp = exp.substring(1).trim();
        }
    }
    return cur;
}

```

Figure 5.6: Data Structure Tree

### 5.2.2. Draw Visual Tree

In order to draw the result, we need to traverse all the nodes of the tree. JUNG will connect each node by the order of the traversal of the tree. As the library JUNG has been defined to draw the tree from root to leaves, so we have to implement the tree by pre-order traversal which was introduced in Chapter 2. Recursion is used to traverse the tree structure to get the possible children points and add the edges between children and parent points.



```

private void traverseTree(Point point, DelegateTree<Point, Integer> g) {
    if (point.getLeft() != null) {
        g.addEdge(edgeFactory.create(), point, point.getLeft());
        traverseTree(point.getLeft(), g);
    }
    if (point.getRight() != null) {
        g.addEdge(edgeFactory.create(), point, point.getRight());
        traverseTree(point.getRight(), g);
    }
}

```

Figure 5.7: Draw Visual Tree

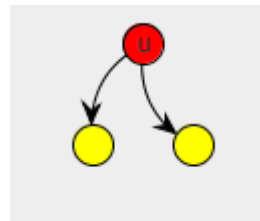
‘g.addEdge’ in Figure 5.7 is a method from the library JUNG introduced in Chapter 2. It connects the nodes in the abstract tree. After drawing the tree, we color the parent and child nodes to distinguish the operator and bag more clearly.

```

if (p.getOperator() != null)
    return Color.RED;
return Color.YELLOW;

```

(a)



(b)

Figure 5.8: Color the Nodes

### 5.3. Tree Animation

In order to get the idea of bag calculations, we use the animation to describe the processing procedure. Similar to the visual tree generation, the animation is based on postorder traversal.

For the generated visual tree, the indexes of each node are ordered using a postorder traversal. The animation is changing the node color by these specified indexes.

```

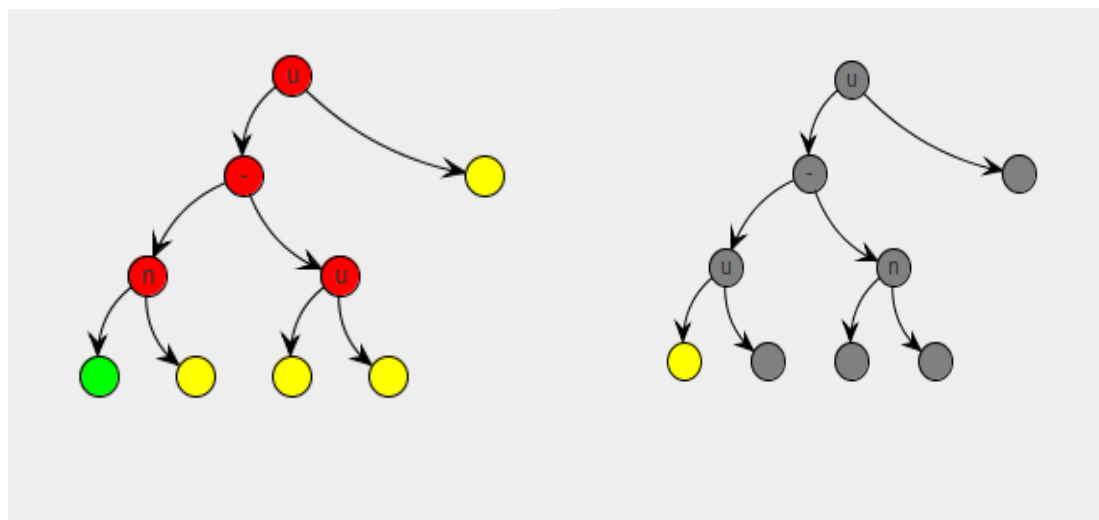
private void orderPoints(Point point) {
    if (point.getLeft() != null) {
        orderPoints(point.getLeft());
    }

    if (point.getRight() != null) {
        orderPoints(point.getRight());
    }
    points.add(point);
}

```

Figure 5.9: Order the Index

If the index of the current node equals the index of the current animation, then the color of the current node changes to green. Otherwise the color is left as it was before. We loop over all nodes to finish the coloring (see Figure 5.10 a)). It is a continuous procedure which cannot be interrupted by user unless it stops by itself after finishing the animation for all nodes.



a) Animation without interrupting

b) Calculating steps animation

Figure 5.10: Animations

The animation diagram in Figure 5.10 b) is achieved in a very similar way as the left animation. The biggest difference is when the animation is on current node; all other nodes will be changed to grey color. And it can only work with the help from end user. This function is added to help learners to control the toolkit more directly. By clicking the ‘step’ button on the visual panel, the animation will move only for once and change the node color as well. This current node will always keep the current color after the animation, same as all other nodes. Continue clicking this button, the animation will happen on all nodes in sequence. This is a feature for helping teachers to explain more clearly in class and students can understand the whole process easily.

Selecting is usually used to mark objects as interested. User can select a visual target by mouse placing or clicking. In particular, each object always has a suspending circle to highlight the details of the selected object in visual view [27]. As shown in Figure 5.11, the details of a node are visible by placing the mouse above it. This makes studying and teaching more convenient and clear.

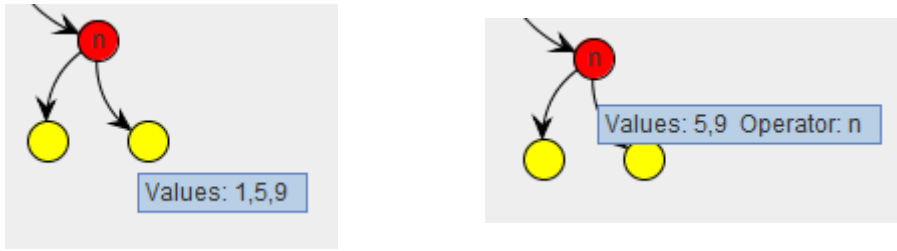


Figure 5.11: Selecting and Highlight

To get the details of an operation, the user can right click one of these nodes. The details are displayed in a separated box.

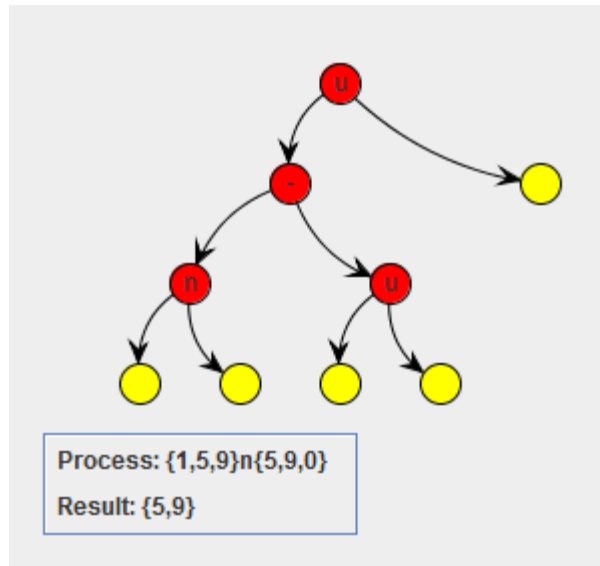


Figure 5.12: Operation Highlight

#### 5.4. Node List

For the node list on the panel, when the tool is doing the animation, the details of nodes will be shown one by one as animation moves. When the animation has finished, all details are added to the list on the right hand of the tree. By clicking any line in the node list, we can get the corresponding value on the tree which changes to green color.

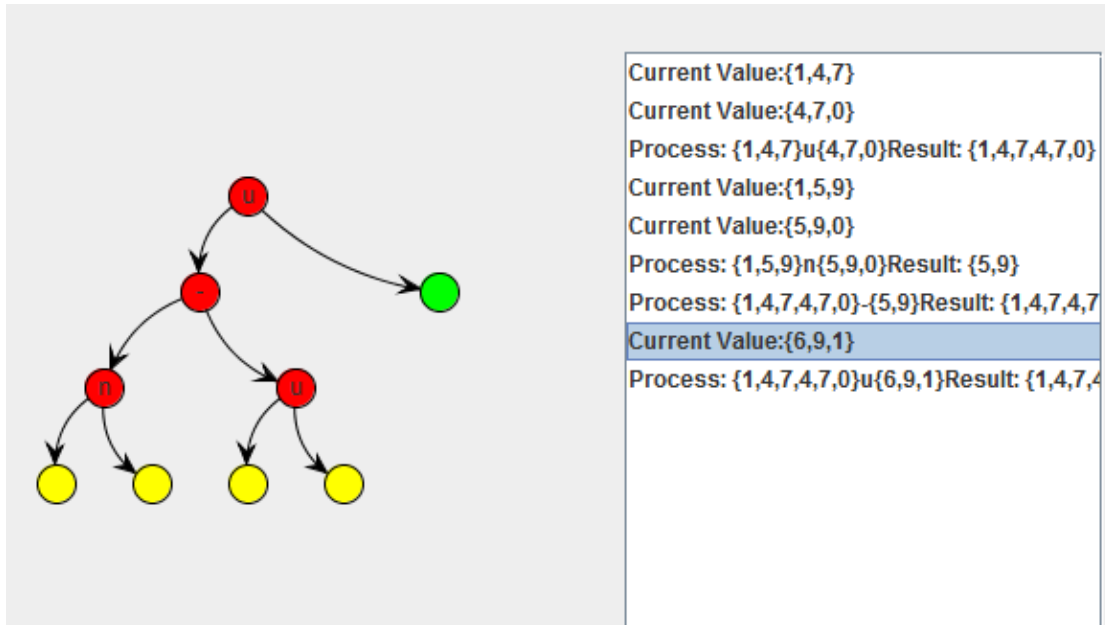


Figure 5.13: for Single Node, it shows its bag value

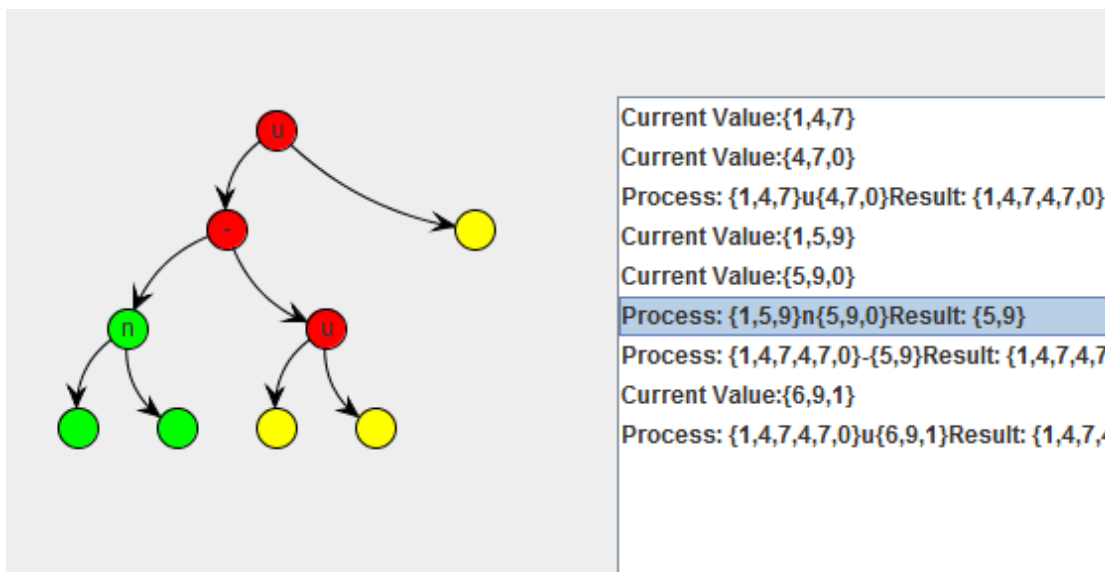


Figure 5.14: for a process, it shows the calculation between bags

For the implementation of this part, it is the same as the achievement of animation and steps animation. Save all tree nodes in sequence by the tree traversal. When the current animation comes to any node, extract the information in this node. Then put the information on an item in the list from the top down in the bounding box. The program continues working for the loop until animation has finished (see Figure 5.13). Finally we can read all the information from each node is listed on the panel.

```

if (points != null && points.size() > 0) {
    // Make sure it is the first one in the list, in order
    // to make the order correct.
    if (points.get(0) == p) {
        // It is an operator, we show details and result
        if (p.getOperator() != null) {
            StringBuffer result = new StringBuffer();
            result.append("Process: ");
            result.append("{");
            for (int i = 0; i < p.getLeft().getValues().size(); i++) {
                result.append(p.getLeft().getValues().get(i));
                if (i < p.getLeft().getValues().size() - 1)
                    result.append(",");
            }
            result.append("}");
            result.append(p.getOperator());
            result.append("{");
            for (int i = 0; i < p.getRight().getValues().size(); i++) {
                result.append(p.getRight().getValues().get(i));
                if (i < p.getRight().getValues().size() - 1)
                    result.append(",");
            }
            result.append("}");
            result.append("\n");
            result.append("Result: {");
            for (int i = 0; i < p.getValues().size(); i++) {
                result.append(p.getValues().get(i));
                if (i < p.getValues().size() - 1)
                    result.append(",");
            }
            result.append("}");
            System.out.println(result.toString());
            DrawResult.textArea.append(result.toString());
            DrawResult.textArea.append("\n");
        }
        // Show the results.
        StringBuffer result = new StringBuffer();
        result.append("Current Value:");
        result.append("{");
        for (int i = 0; i < p.getValues().size(); i++) {
            result.append(p.getValues().get(i));
            if (i < p.getValues().size() - 1)
                result.append(",");
        }
        result.append("}");
        System.out.println(result.toString());
        DrawResult.textArea.append(result.toString());
        DrawResult.textArea.append("\n");
    }
    // Remove this point from the list.
    points.remove(p);
    transed = false;
    // debug
    // We set the color as green here, if this node is
    // an operator, we set the color below.
    return Color.GREEN;
}
}

```

```

if (p.getOperator() != null) {
    // If this node is an operator, we set it to different color.
    if (p.getOperator() == "u")
        return Color.YELLOW;
    else if (p.getOperator() == "n")
        return Color.CYAN;
    else {
        return Color.RED;
    }
} else
    return Color.YELLOW;
} else {

    if (p.getOperator() != null) {
        if (p.getOperator() == "u")
            return Color.YELLOW;
        else if (p.getOperator() == "n")
            return Color.CYAN;
        else {
            return Color.RED;
        }
    } else
        return Color.YELLOW;
}
}

```

Figure 5.15: The Implementation of Node List

The animation of the calculation process is traversed by post-order. In order to record each node's information in the tree with calculation order, we need to traverse the tree by post-order, which is from bottom to top. We also use a list to store nodes by this order, and then fetch all the information from these nodes. Meanwhile add them to a JList (a component in Java). By clicking the item in JList, we can get an index. With this index we can get the specified node and highlight all nodes one by one. When the node is an operator, what we need to do is not only to highlight the node itself but also its children. If the node is a value, only this specified node will be highlighted (see Figure 5.14).

## 6. Conclusion and Future Work

In order to extend the ideas and applications for this thesis, in this section, we will mainly discuss the conclusion and further works on the current work. This will help us get clear summary of what we have done and what can be thinking and developing further.

### 6.1. Conclusion

At the beginning, this paper introduces the basic knowledge of set and bag. Both of them are the foundation of all known mathematics. And the paper also describes related technique and concept using basic knowledge of them, for example Relational Database Management System (RDMS). The aim of this paper is using visualization concept to develop a toolkit for visualizing the process of bag calculations. We try to using color and animation to explain the calculation processing step by step, users can get the results and details for each step directly. Compare with the traditional way for teaching, the most important key is interactive. Sometimes it is hard to understand how to make the bag calculation correctly, and we might confuse which step we need to do next and how to calculate the result. We invited about 10 people to testing and use the tool. They left some comment to us directly after they using the tool. After analyzing the feedback from invited people, most of them think that it is much easier to understand the calculation processing since the toolkit will show the processing step by step and with the color coding node. They think this is a good way for a beginner to learn a new knowledge in a not boring way. As a result the tool shows the result directly, it makes it easier for a user to understand how bag calculations work, and it is much funnier also since they can interactive with computer.

As education tool for mathematical theory bag, the thesis visualization not only shows the result correctly, but also combines the InfoVis to visualize the calculation animations. The animation includes the tree calculation and the calculation steps. With the dynamic animations, we can figure out the understanding of three bag operations easily and visually which leads the education of bag to vivid.

### 6.2. Future work

The primary target for this thesis is for the teaching of bags. Considering the precise calculation results from this tool and the fluent visualization, we can treat it as a calculator with its wider application in bag teaching. Efficient and flexible computing plays core roles as a calculator especially when adapted to teaching purpose. If mapping to our project, this requires the tool can perform different tasks with various visualizing speed. However, since we can only do the fast animations, do consider to the entire teaching aspect, it may not make sense enough to learners. In this case, for the future developing, adding a new function to control the speed of visualization can op-

timize the issue. This widens the usage of the toolkit effectively.

Meanwhile there also exists a limitation aspect about the visual screen. During the real computing process, there are always some long expression existing. The generated tree of long expression may not be full-scale shown. Zooming out the whole tree to make it visible is one solution, but it cannot show the animation so obviously. In this case, here provide a new idea: collapse the sub-tree to a farther-node of the tree to make the size of the tree smaller, even for a very long expression, the final layout tree must be a complete binary tree. It makes the steps animation more clearly; end users are not so easy to get confused by too many subtrees or nodes.

Because the limit of time, we just testing and survey in a very small group, the bigger survey is needed to confirm that our result is reliable since we are just testing with a very small group.



# References

- [1] E. Kamke, *Theory of Sets*. New York: Dover Publications, Inc, 1950.
- [2] P.R. Halmos, *Naive Set Theory*. Princeton: Princeton Press, 1960.
- [3] J. Venn, *On the Diagrammatic and Mechanical Representation of Propositions and Reasonings*. Dublin Philosophical Magazine and Journal of Science 9 (59): 1–18, 1880.
- [4] M. Machover, *Set Theory, Logic and their Limitations*. Cambridge: Cambridge University Press, 1996.
- [5] A.D. Aleksandrov, A.N. Kolmogorov & M.A. Lavrent'ev, *Mathematics: Its Content, Methods, and Meaning*. Cambridge: M.I.T Press, 1999.
- [6] D. Singh, A. M. Ibrahim, T. Yohana & J. N. Singh, *Complementation in Multiset Theory*. International Mathematical Forum, 38(6): 1877-1884, 2011.
- [7] K.P. Girish & S. J. John, *Multiset topologies induced by multiset relations*. Information Sciences, (188): 298-313, 2011.
- [8] P. J. Pratt & M. Z. Last, *A Guide to SQL*. Course Technology, 2008.
- [9] K. Krozser (2006), *Quick Thoughts On Book Search*. [Online]. Available: <http://booksquare.com/quick-thoughts-on-book-search/>
- [10] Google Books, [online]. Last accessed 12<sup>th</sup> February 2013 at <https://www.google.com/search?q=information+technology&btnG=Search+Books&tbm=bks&tbo=1&oq=infor>
- [11] C. Coronel, S. Morris & P. Rob, *Database Systems: Design, Implementation, and Management*. Course Technology, 2012.
- [12] R. Wesley, *Computers and Society*. McGraw Hill Text, 1972.
- [13] W. Schroeder, K. Martin & B. Lorenzen, *The Visualization Toolkit*. New Jersey: Prentice-Hall, Inc, 1998.
- [14] Y. Wang (2012), *A Space-Filling Technique for the Visualization of Planar Graphs*. Bachelor Thesis of DFM, LNU. [Online]. Available: <http://cs.lnu.se/iso-vis/theses/finished/21033.pdf>
- [15] B. McDonnell & N. Elmqvist, *Towards Utilizing GPUs in Information Visualization: A Model and Implementation of Image-Space Operations*. IEEE Transactions on Visualization & Computer Graphics, 6(15): 1105-1112, 2009
- [16] Z. Walter & S. Cunningham, *Editors' introduction: What is mathematical visualization? Visualization in teaching and learning mathematics* (1991): 1-8, 1991.
- [17] S. MacDonald, D. Szafron & J. Schaeffer, *Rethinking the pipeline as object-oriented states with transformations*. IEEE Conference Publications, 2004.
- [18] S. Diehl, C. Görg & A. Kerren, *Animating Algorithms Live and Post Mortem*. In *Software Visualization*. LNCS State-of-the-Art Survey, (2269): 46-57, 2002.
- [19] T. Müldner, E. Shakhshuki, A. Kerren, Z. Shen & X. Bai, *Using Structured Hypermedia to Explain Algorithms*. In *Proceedings of the 3rd IADIS International Conference e-Society*, (5): 499-503, 2005.

- [20]S. Diehl, A. Kerren & T. Weller, *Visual Exploration of Generation Algorithms for Finite Automata on the Web*. In Implementation and Application of Automata, Lecture Notes on Computer Science, LNCS, (2088): 327-328, 2001.
- [21]A. Kerren & J. T. Stasko, *Algorithm Animation - Chapter Introduction*. In *Software Visualization*. LNCS State-of-the-Art Survey, (2269): 1-15, 2002.
- [22]Revelation Software Ltd (2013). *Web Deployment*. [Online]. Last accessed 13<sup>th</sup> February 2013 at <http://www.revelation.com/revelation.nsf/byTitle/>
- [23]National Taiwan Normal University (2012), *Tree*. [Online]. Last accessed 19<sup>th</sup> November 2012 at <http://www.csie.ntnu.edu.tw/~u91029/Tree.html>
- [24]J. Celko (2011), *Joe Celko's SQL for Smarties*. Elsevier.
- [25]JUNG: Java Universal Network/Graph Framework [Online]. Last accessed 15<sup>th</sup> November 2012 at <http://jung.sourceforge.net/>
- [26]Y. Rogers, H. Sharp & J. Preece, *Interaction Design: Beyond Human - Computer Interaction*. New York: Multiscience Press,Inc, 2011.
- [27]H. Hege, K. Polthier & G. Scheuermann, *Topology-Based Methods in Visualization II*. New York: Springer, 2009.
- [28]J. Liu (2012), *Visualization of Weather Data: Temperature trend visualization*. Bachelor Thesis of DFM, LNU. [Online]. Available: <http://cs.lnu.se/isovis/theses/finished/20969.pdf>



# **Linnæus University**

Sweden

Faculty of Technology

SE-391 82 Kalmar | SE-351 95 Växjö

Phone +46 (0)772-28 80 00

teknik@lnu.se

[Lnu.se/faculty-of-technology?l=en](https://lnu.se/faculty-of-technology?l=en)