Assignment-3 –The SetCalculator
COMPSYS 202 / MECHENG 270

Partha Roop

1 Introduction

Discrete structures such as **sets** are foundational for the design of any system involving a computer. Hence, they are the main basis of **discrete maths**, a discipline of maths that is essential for the formalization of concepts used in computer engineering, mechatronics, and computer science.

While a set is a collection of unique members, there can be a relation that satisfies among them. For example, we have a set containing three numbers 1,2, and 3. Considering the "greater than" relation, then we have a set of relation "2 is greater than 1", "3 is greater than 2", and "3 is greater than 1". As such, a relation on a set is essentially again a set. The set of relations naturally is the inheritance relationship in the object oriented programming (OOP).

The **objective** of this assignment is to develop a program called a set calculator. While the program is running, the user can open the input file, which contains the information about a set of strings and a set of relations. After extracting the information, the set calculator program checks whether the relation is reflexive, symmetric, and/or transitive. Furthermore, in case of equivalence relation, the equivalence class is computed.

1.1 Learning outcomes

1. Discrete Structures such as *Sets, Relations, and Equivalence Classes* as object-oriented concepts.
2. Using the above mathematical entities to learn about OO concepts: encapsulation, inheritance and composition.
3. Improve the understanding of unit testing and test driven development learnt through the lectures and also the labs in week 10, 11.
4. Practice the use of `vectors` and `iterators` and develop confidence in string operations.

2 Getting started

The SetCalculator program has normal mode and unit testing mode. In normal mode, the user can dynamically interface with the program. Whereas, in testing mode, the program only the unit testing result is printed on the terminal.

2.0.1 Normal Mode

1. Download the assignment zip file, and unzip it. It contains the skeleton code source files.
2. Open the terminal in the directory where you unzipped the files. Enter `make` to compile, enter `./calculator.o` to run the program.
3. While the program is running, enter `open TestCases/a.txt`. This will open a input file in the TestCases folder.
4. Enter `list`. It will show the extracted information in the input file.
5. You can type `check -r` to check if this relation is reflexive. However, currently this function is unimplemented for you to complete in this assignment. Also, try `check -s`, `check -t`, and `check -e`.
6. There is also `eqclass` command. Enter `help` command to know more about the commands.
7. Finally, type `exit` to terminate the program.

2.0.2 Unit testing Mode

For unit testing, the program can be compiled with a special flag *debug*.

1. On the terminal, type `make clean` to remove previously generated output files.
2. Then, type `make debug` to compile the program in the unit testing mode.
3. Enter `./calculator.o` to run the program. You will see the unit testing result.
4. Open SetControl.cpp file using any text editor. At the bottom of this code, there is a function called *Testing*, where you can manage the unit testing scenario.

3 Background Information

In this section, we describe the background knowledge needed to complete this assignment.

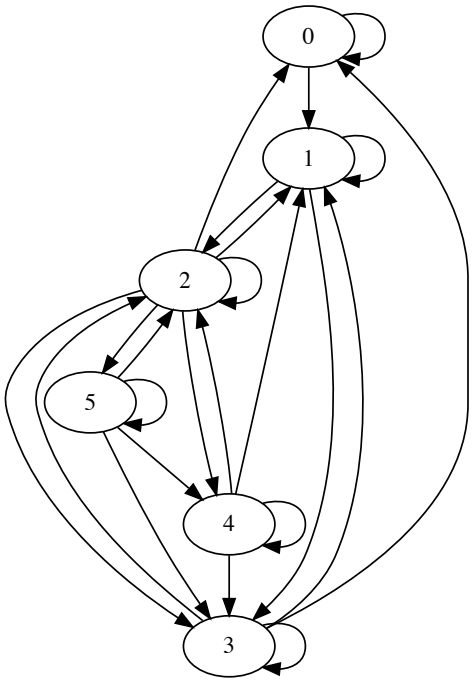
3.1 Binary Relations

A relation, by definition, is a subset of the product of sets i.e., $R \subseteq S_1 \times S_2 \times .. \times S_n$. Here $S_1, .., S_n$ are the participating sets and any subset of the product set is a relation, by definition. The above relation is known as an n-ary relation.

In this assignment, we will be only interested in one specific type of relation called a **binary relation**. Here, $R \subseteq S \times S$. Such relations have many practical applications, such as say in your GPS navigator. We will elaborate the further on this, using the concept of **graphs**, in the following section.

3.2 Input File Format and Visualization

In the input file, the information about the set and relation is stored as a graph. Graphs are mathematical objects of the form $< V, E >$ where V denotes a set of vertices and E denotes a set of edges. Graphs are used in multitude of applications. A very prominent one is is GPS navigation, where the destinations in a city, for example could be represented as the vertices and the edges between them represent the different paths.



(a) Graph visualization.

```
// 0, 1, 2, 3, 4, 5
digraph testgraph{
0 -> 0;
0 -> 1;
1 -> 1;
1 -> 2;
1 -> 3;
2 -> 0;
2 -> 1;
2 -> 2;
2 -> 3;
2 -> 4;
2 -> 5;
3 -> 3;
3 -> 0;
3 -> 1;
3 -> 2;
4 -> 2;
4 -> 3;
4 -> 1;
4 -> 4;
5 -> 5;
5 -> 4;
5 -> 3;
5 -> 2;
}
```

(b) Text representation.

Figure 1: An example graph of a set and relation.

In our example, we will consider simple graphs. Consider the example graph shown in Figure 1. Note that the set of vertices V can be considered as the set using which a relation may be created. In this example $V =$

$\{0, 1, 2, 3, 4, 5\}$. The relation captures the the edges of the graph. In this example the adjacency relation between vertices may be captured as a relation $R = \{(0,0), (0,1), (1,1), (1,2), (1,3), (2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (3,3), (3,0), (3,1), (3,2), (4,2), (4,3), (4,1), (4,4), (5,5), (5,4), (5,3), (5,2)\}$. Such a relation can be also represented using a specific text format in Figure 1b. Using a popular tool for graphs called Graphviz, a graph is visualized as in Figure 1. There is an web application for Graphviz tool at `www.webgraphviz.com`.

The input file can be visualized for manual validation of your SetCalculator program. Try to visualize the sample input files in the TestCases folder. This can be done simply by copying and pasting the entire text in the file to the Graphviz tool.

It is important that, we assume that the first line in Figure 1b must explicitly list all the nodes in the input file graph. Since the relation should be specified only between the existing nodes, the first line is read by the program to indicate what are the existing nodes. Note that, in `www.webgraphviz.com`, the graph can be drawn even without the first line. However, in this assignment, *you can assume that only the correct input files are tested during the marking process*. This means that the first line is always correctly provided.

4 Submission

In this section, we clearly indicate your tasks, and guide how to submit your assignment successfully.

4.1 Your Tasks

You are required to implement two commands: *check* and *eqclass*. These are:

1. `check -r`: this checks if the relation is reflexive. In StringRelation.h file, there is an empty function called *isReflexive()*. This function should return true if the relation is reflexive, and your task is to implement this function correctly.
2. `check -t`: this checks if the relation is transitive. For this, you must complete *isTransitive()* function In the StringRelation class.
3. `check -s`: this checks if the relation is symmetric. For this, you must complete *isSymmetric()* function in the StringRelation class.
4. `check -e`: this checks if the relation is an equivalence relation. For this, you must complete the function *isEquivalence()* in the StringRelation class.
5. `eqclass [element]`: if the relation is an equivalence relation, then this command computes the equivalence class of the specified [element]. For example, 'eqclass 1' computes the equivalence class of the node 1. For this, you must complete the function *computeEquivalenceClass(element)* in the StringRelation class.

Hint: Essentially, this assignment completes if you implement those functions. You don't have to worry about how to print on the terminal. Once you have correctly implemented those functions, the program will show the correct results.

Your implementation should be programmed only in the SetOfStrings.cpp and StringRelation.cpp files. This means that, you should not modify the provided Makefile, main.cpp, SetControl.cpp/h, and SetUI.cpp/h files. However, for unit testing, you can modify the *Testing* function in the SetControl.cpp. You are free to create any helper functions and variables in the SetOfStrings and StringRelation files.

Remember that, in the Introduction section, we have already explained that a relation is essentially a set. Thus, the StringRelation class (relation class) is designed to inherit the SetOfStrings class (set class), such that the relation also has the set properties. Also, the StringRelation class has an object called SetMembers, which is a SetOfStrings object. This shows that they also have a composition relationship. Exploit the inheritance and composition relationship to complete your logic. Note that the SetMembers object stores the nodes information (see the first line in Figure 1b).

Hint: the StringRelation class inherits an attribute called 'elements' from the SetOfStrings class. The 'elements' object of the StringRelation class stores the binary relation, e.g., (1,2). Meanwhile, the StringRelation class has an object called SetMembers, and this object has its own 'elements' object. This one stores the node information, i.e., the first line in the input file.

4.2 How to submit your code

This assignment deadline is **11.59 pm 2 October 2019**. You must submit a zip file, which contains:

1. A folder contains your SetOfStrings.cpp/h and StringsRelation.cpp/h files. **Do not** include any other files, such as SetControl.cpp/h, Makefile, main.cpp, and etc. We only need your SetOfStrings and StringRelation files. Other files will be ignored.

2. Coversheet: this should include your full name, id, upi, and your sign for declaration of originality.

Please double check your submission, because if your code fails to compile, then you will automatically lose almost every mark. There will be no exceptions if you accidentally submitted the wrong files, regardless if you can prove you did not modify them since the deadline. No exceptions.

4.3 Academic Honesty

- The work done on this assignment must be your own work. Think carefully about any problems you come across, and try to solve them yourself before you ask anyone for help. Under no circumstances should you take or pay for an electronic copy of someone else's work.
- All submitted code will be checked using software similarity tools. Code detected for suspicious similarity will result in an Investigative Meeting and will be forwarded to the Disciplinary Committee.
- Penalties for copying will be severe – to avoid being caught copying, don't do it.
- To ensure you are not identified as cheating you should follow these points:
  - Always do individual assignments by yourself.
  - Never give another person your code.
  - Never put your code in a public place (e.g., forum, git, your website).
  - Never leave your computer unattended. You are responsible for the security of your account.
  - Ensure you always remove your USB flash drive from the computer before you log off.

4.4 Late submissions

Late submissions incur the following penalties:

- 15% penalty for zero to 24 hours late
- 30% penalty for 25 to 48 hours late
- 100% penalty for over 48 hours late (dropbox automatically closes)

In case if the assignment is extended, then the late submission penalty applies identically to the extension date.