CycleX Program User Manual

A Guide Written by Group 7

*Daniel Vu*

*Collin Haines*

*Kevin Silver*

*Julio Diaz*

CSCE 240

Columbia, South Carolina 29225

Table of Contents:

1. Getting Started

2. Program Procedure

3. Input & Output

---------------------------------------------------------------------------------------------------------------------

1. Getting Started

This manual serves as a guide for the CycleX program, a program used to parse an input file containing data for a series of nodes, compute the parent-child path between the nodes using a union-find algorithm, and then output the result in a text file. All inputs and outputs are handled by the argument in the command line; no user input is required after the command line input.

To start using CycleX:

a. Open up Terminal.

b. Navigate to directory where this file is located (e.g. 'Downloads/', '~/')

c. Compile all .cpp and .h files with the command:

make

d. Once compiling is complete run the program with the command:

./Aprog zin1.txt zoutx.txt zlogx.txt

e. To read the output and/or the log file:

cat zoutx.txt

cat zlogx.txt

---------------------------------------------------------------------------------------------------------------------

2. Program Procedure

The program is run from the command line with three parameters: the input file, the output file, and the log file, in that order. After initializing UnionFind, the primary operating class of the program, and opening the input and output streams, the main class counts the number of Node values in the input file using a for loop to count integer pairs, as each set of two integers is a usable input.

Node values are stored in instances of the Node class, a container that stores the node's value as well as its root. In this program, the value of a Node is referred to as the current value, and the value of the root of the Node is referred to as its parent value.

For each pair of integers, the loop calls the "addLink" method of the UnionFind class. This method initializes a pair of int variables "smaller" and "larger", which are used to organize the pair of values in the input, and a pair of Node instances "parentNode" and "currentNode", which will store the two integers in Node containers after they have been organized. The program first sorts the two values into smaller and larger, then checks if the Node values of either of the input integers are default values.

If the node value of the smaller integer is a default value (in this program, the default value of a node is DUMMYX, defined as a -1 integer), then the current and parent values of that node are changed to the smaller integer. Similarly, if the node value of the larger integer is a default value, then the current value of the node is changed to the larger integer, and the parent value is changed to the smaller value. The disparity between the two node values are because we wish for the smaller of the two values to be the root node, so its root must be itself.

If neither of the nodes for the two integers is a default value, then the smaller integer becomes the parent value of the new node, and the larger value becomes the node's current value.

Once the loop has concluded, the "toString" method of the UnionFind class is called to the output stream. The toString method uses an iterator and a for loop to crawl across the Nodes obtained from the input. A vector is initialized and then filled with the Node data reaching from front to back of a complete Node path, and this vector is then sent to the "toStringPath" method of the UnionFind class, in which a single string is filled with each Node's contents. Each individual Node's data is obtained and formatted by the "toString" method of the Node class.

After all of the Node paths have been sent to the output file, all streams are closed and the program terminates.

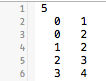
---------------------------------------------------------------------------------------------------------------------

3.Input & Output

The CycleX Program requires only one input file and produces one output file and one log file a set in the command line argument.

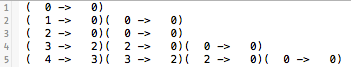
Input File

The input file may seem confusing but it is organized in a very simple manner. The first line of the input file consist of a integer n that represents the number of arcs in the graph. Following this line will be n lines of data, each line consisting of a pair of integers. Each pair of integers (a, b) represents an arc from node a to node b. (See Figure 1.)



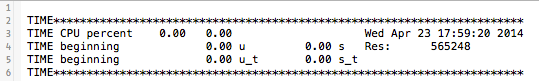
*Figure 1. Example of Input File*

Output File

Under normal execution, CycleX produces an output file that contains an output of integers and arrows within parenthesis. In each parenthesis arrows indicate the connection between a child to its parent node. The line of parenthesis represents the path that each initial node takes, ending with the root of the path. All together this makes up a tree. The output does what the program is intended to do and does not include the redundant paths. (See Figure 2.)

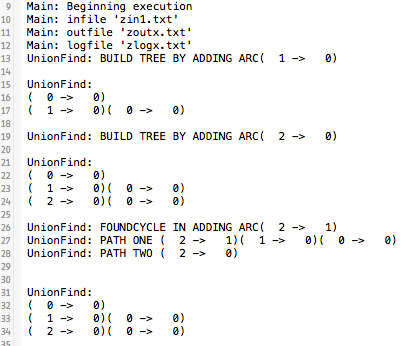
*Figure 2. Example Output File*

Log File

The log file is the file with the most amount of information. It begins and ends with information about the time CPU usage and date. (Figure 3.)

*Figure 3. Log File Beginning*

The next piece of data is what file CycleX uses as the input file, the file name of the output file and log file. It then shows what arc it is adding to the tree and then prints out the tree with the arcs and its path to the root. When there is a cycle found it shows the arc it was adding when the cycle was found. The next piece of information is the first path that node has to the root and then the second path to the root. It then shows the tree again with the path that is closest to the root of the tree.(See Figure 4.)

*Figure 4. Log File Main Body*