Boston University

Programming Assignment 2 : SCC Documentation

MET CS566 A1: Analysis of Algorithms

Austin J. Alexander :: austinja@bu.edu

22 April 2015

b) Java 8, Mac OS 10.9.5, Sublime Text, Terminal

c) In a UNIX-like environment, if javac is installed, using a terminal, change the current directory to the directory of this file (and its associated files). Since there is an included Makefile, if Make is installed, simply run `make`, which will run `javac *.java java StronglyConnectedComponents p3_input.txt`; if Make is not installed, run the latter combination of commands or whatever is appropriate for the user's system configurations.

Once the program has been successfully compiled and run with the included input file, a different (text) input file (formatted according to the instructions) may be used by running `java StronglyConnectedComponents FILENAME`, where `FILENAME` is the name of the input file with the proper format extension. The initial section of the program will print/display the contents of the input file.

d) The data structures implemented to represent a graph (using the Graph class) are two arrays of vertex objects (using the Vertex class): a 1-dimensional array that holds each vertex in the graph and a 2-dimensional array that holds each sub adjacency list; the two arrays are synchronized by index.

The Graph class has a number of methods for getting vertex information, reorganizing/sorting vertices, and printing/displaying and outputting graph/vertex information. Naming conventions and comments before each method explain individual functionality.

While the DFS and DFS-VISIT algorithms (see below) are separate static methods, the SCC algorithm is implemented as a series of statements in the main program, StronglyConnectedComponents. Thus, the SCC algorithm will only run once per program run. During the program's execution, this particular implementation of SCC prints/displays information for each stage. The final stage, as instructed, prints/displays and outputs to a file each strongly-connected component, with vertices listed in increasing order of $u.f$.

However, since print/display and output formats were not otherwise further specified, the first section of the final stage lists each strongly-connected component as 'SCC' followed by a counting integer (distinguishing each strongly-connected component) followed by the set of vertices for that component listed. This format was inspired by the mathematical notation on p.617 of Corment, Leiserson, Rivest, and Stein's *Introduction to Algorithms*, 3rd ed. (hereafter, CLRS), where parenthetical set notation was used (although, in this implementation, each vertex is separated by a single space as opposed to commas).

The second section of the final stage prints/displays and outputs the graph under consideration in an adjacency-list representation, although the vertices are given in increasing order of $u.f$ (as in the previous section of the final stage).

As instructed, the implementation of the DFS, DFS-VISIT, and SCC algorithms follow, as precisely as possible in Java, the algorithms presented as pseudocode in CLRS. From a software design perspective, the code is efficient to be sure but, in my opinion, at the expense of human readability. Moreover, common software design principles are ignored; for example, instance variables (e.g., pi, name, etc.) are accessed/modified directly (and, thus, are public) as opposed to being accessed/modified only through 'getter'/'setter' methods.