CSC 4500, Spring 2019 Homework #1 (HW1) Circles and Arrows, Part 1

Imagine there is white board. You draw non-intersecting circles on the board, numbered 1 to N, where N is an integer from 2 to 10. You next draw arrows from one circle to another, making sure that each circle has at least one out arrow and one in arrow. Now you play the following "game:"

1. Place a magnetic marker in circle #1, and put a check mark in circle #1. The circle where the marker resides is called the "current circle."
2. Randomly choose among the out arrows in the current circle. (If there is only one out arrow, that choice is trivial.) In this selection, all the out arrows should be equally likely to be picked.
3. Move the marker to the circle pointed to by the out arrow. This becomes the new current circle.
4. Put a check mark in the current circle.
5. If all the circles have at least one check mark, stop the game. If not, go to step 2 and repeat.

Your first programming assignment is to implement this game. I will announce in class which programming language you should use. The resulting program must be capable of being executed on the Windows machine at the podium in our classroom. The program will read from a textfile in the same directory as the executable program, and will write to another textfile in that same directory.

Let N and K be positive integers. For HW1, N is between 2 and 10 inclusive.

The input text file should be named HW1infile.txt. It should be in this form:

- The first line has only the number N, the number of circles that will be used in your game.
- The second line has the number K, the number of arrows you will "drawing" between the circles.
- The next K lines designate the arrows, one arrow per line. Each arrow line consists of two numbers, each number being one of circles in the game. These two numbers are separated by a single blank. The first number designates the circle that is the source (back end) of the arrow; the second number designates the circle that is the destination (pointed end) of the arrow.

The circles and arrows of this game describe a directed graph, sometimes known as a "diagraph." In order to set up the game correctly, you should describe a "strongly connected diagraph." A diagraph is strongly connected when there is a path between any two nodes. In our game, our paths are the arrows, and our nodes are circles.

As you develop this program, you should test it. Make sure that you test it with circles and arrows that describe a strongly connected digraph. Not all circles need to be connected directly to each of the other circles; but as a system, they should be connected in the sense described above. I suggest that each time you make a new HW1infile.txt you draw the desired game board and then translate into the required input file.

Shown below are three systems of circles and arrows. With respect to the definition of “connected” above, NOT ONE of them is strongly connected. In Figure 1 and Figure 3, you could make the digraph strongly connected by adding an arrow from circle 4 to circle 1. The reason we need strong connection is so that you don’t get “stuck” in your random walk around the digraph.

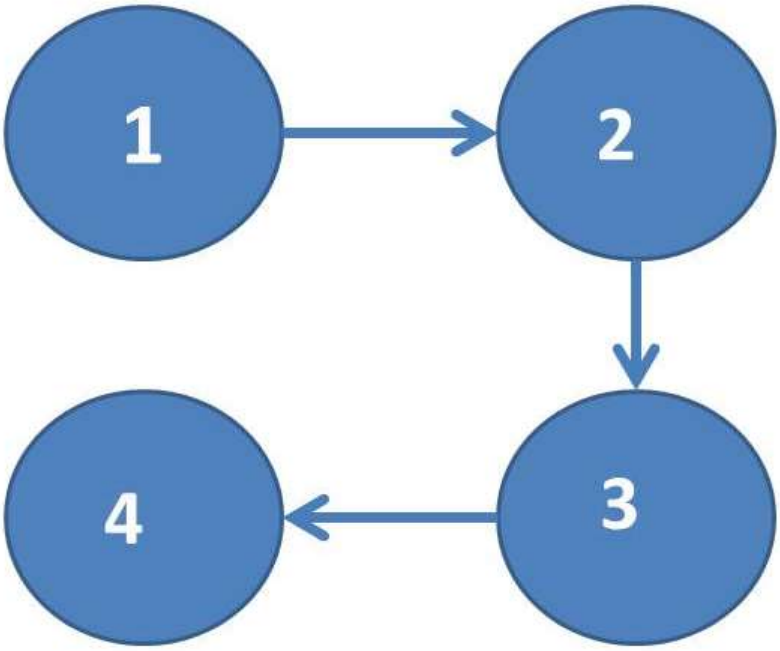

Figure 1. This system is not strongly connected; there is no path to circle 1. If you add an arrow from circle 4 to circle 1, it would be strongly connected.

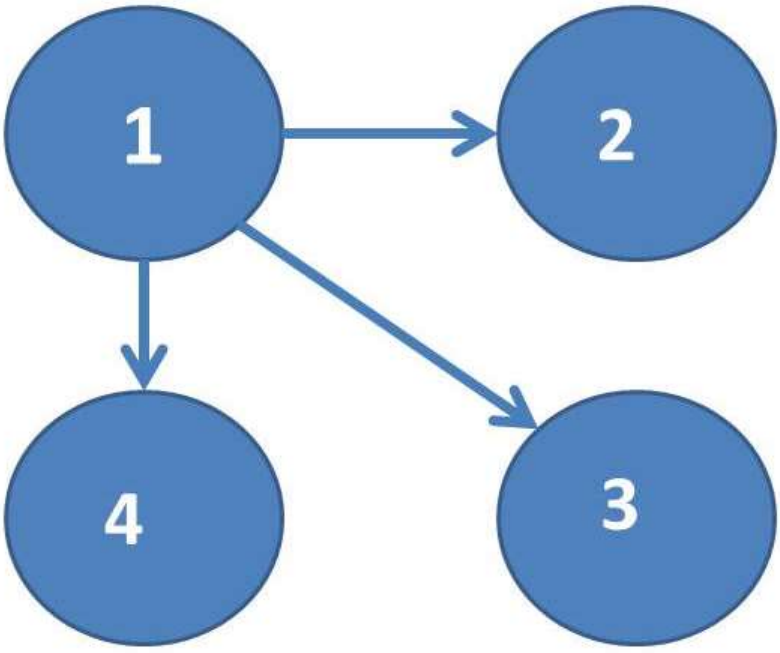

Figure 2. This system is not strongly connected. Once you follow an arrow, you are stuck.

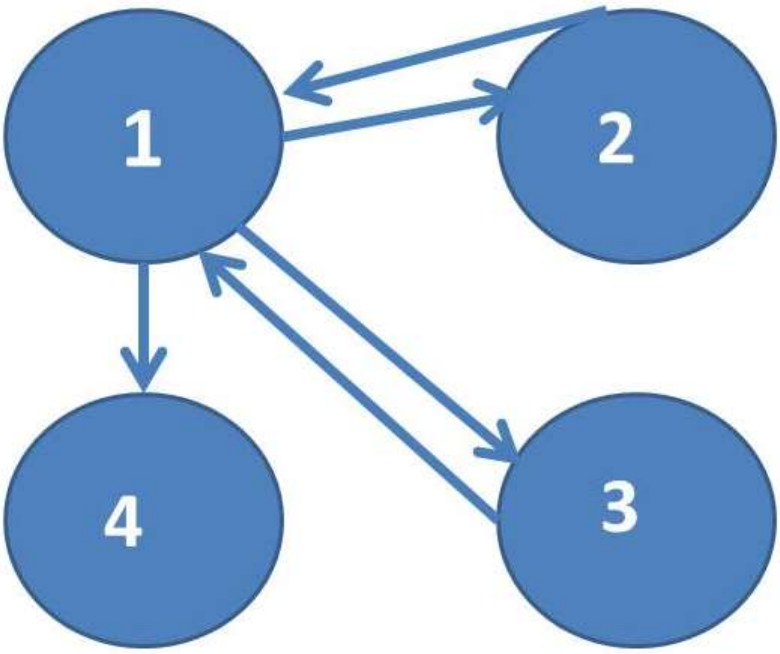

Figure 3. This system is not quite strongly connected; an added arrow from circle 4 to circle 1 would make it strongly connected.

For HW1 your program can assume this connectedness for a given input file. That is, your program need not verify that the circles and arrows described in the input file form a strongly connected digraph. A subsequent assignment will require your program to verify the connectedness.

If the text in the input file does not follow the format described above, your program should end with an error message to the screen and to an output file. The output file should be a textfile. Name your output textfile "HW1lastnameOutfile.txt" where "lastname" is replaced by your last name. My output file would be called HW1millerOutfile.txt.

If the text in the input file DOES follow the description above, then you should play the game until each circle has at least one check. When that happens, the game stops. At the end of the game, you should print out to the screen, and to the output textfile, the following numbers:

I. The number of circles that were used for this game

II. The number of arrows that were used for this game

III. The total number of checks on all the circles combined. (Thought question: how is this related to the number of arrows traversed?)

IV. The average number of checks in a circle marked during the game.

V. The maximum number of checks in any one circle. (NOTE: this number may occur in more than one circle, and that's fine.)

You do not need to put out the minimum number of checks in a circle, because we know that number. (Well, at least we all SHOULD know the minimum number; if you don't know it, think through the specification above.)

All of these numbers should be labeled clearly in both outputs, with explanations sufficient for someone who knows only vaguely what's going on with this strange game.

Start the source code for your program with an extensive opening comment that includes your name, the date, the name of our class, an explanation of what the program is designed to do, a description of the central data structures in your program, and an explanation of any external files used. Internal comments should explain important data structures where they are declared and/or used, delineate large sections of code ("paragraphing comments"), and clarify anything clever or hard to understand in your code. I grade documentation carefully and critically, so I recommend that you spend some time on your opening and internal comments.

We will test your program by recompiling it, by running it with different external input files, and by examining your external file and screen output for each run. If you have any questions, please email your instructor. Best of luck on your first homework assignment.