CSCI 230 Data Structures and Algorithms
Laboratory - Graph Traversals
Jonathan Limpus

Introduction

This assignment is based on material from the course primary textbook, "Data Structures and Algorithms in Java" by Michael Goodrich, chapters:

- Section 14.3 Graph Traversals
- Section 14.4 Transitive Closure

These problems may require access to supplied source code which is available on the course GitHub under libraries.

Assignment

Problem 1. The file `UDGraph.java` contains code for a class `UDGraph`, a directed graph represented by a `boolean` adjacency matrix `adjMatrix[][]`. For simplicity, vertices are denoted by integers in the range $0, \ldots, n-1$, where n is the number of vertices. Each entry in `adjMatrix[i][j]` is `true` if (i, j) is an edge of the graph; otherwise, `false`.

Complete the defintition `UDGraph length2Paths()` which constructs and returns a `UDGraph` with the same number of vertices as the referenced `UDGraph`. The new graph contains the edge (v, w) if and only if there is a path of length 2 from v to w in the referenced graph—in otherwords, there is some vertex u such that (v, u) and (u, w) are both edges of the referenced graph.

Note that a length-2 path can start and end at the same vertex: if the referenced graph contains th edges (v, w) and (w, v), then it contains a length-2 path $\langle v, w, v\rangle$ from v to itself, and the new graph should contain the self-edge (v, v). Moreover, if the referenced graph contains the self-edge (v, v), then $\langle v, v, v\rangle$ is a length-2 path so the new graph should contain (v, v).

If a vertex w can be reached from a vertex v by a length-1 path (one edge) in the referenced graph but *not* by a length-2 path, the new graph should *not* contain (v, w).

Though a näive $\mathcal{O}(n^3)$ solution exists by simply using a triple-nested for loop, you are asked to find a more efficient algorithm.

Code: `GraphOperations.java`

Solution.

```java
    /**
     * Returns a new UDGraph with the same vertices as "this" UDGraph. The new graph
     * has an edge (v, w) if and only if there is a path of length 2 from v to w in
     * "this" graph.
     *
     * @return the new UDGraph.
     */
    public UDGraph length2Paths() {
        UDGraph newGraph = new UDGraph(vertices);
        // Put your answer to Part I here.

        int[][] matrix = new int[vertices][vertices];
        // Square the matrix
        for(int i = 0; i < vertices; i++) {
            for(int j = 0; j < vertices; j++) {
                matrix[i][j] = 0;
                for(int k = 0; k < vertices; k++) {
                    // mildly hacky way to convert the booleans into int for calculation
                    int boolToInt1 = adjMatrix[k][j] ? 1 : 0;
                    int boolToInt2 = adjMatrix[i][k] ? 1 : 0;
                    matrix[i][j] += boolToInt1 * boolToInt2;

                }
            }
        }

        for(int i = 0; i < vertices; i++) {
            for(int j = 0; j < vertices; j++) {
                if (matrix[i][j] == 0)
                    newGraph.adjMatrix[i][j] = false;
                else
                    newGraph.adjMatrix[i][j] = true;
            }

        }

          return newGraph;
    }
```

□

Submission Guidelines

Modify this LaTeX document by inserting your solutions into the `solution` environments above. Submit this document along with any source code files `[*.java]` and archives `[*.jar]` to the course LMS. Finally, comment out the `\input{TexFiles/SubmissionGuidelines.tex}` line in `main.tex` to hide this section.