

## Programming Project 2

Due **12/2/15**, 11:55pm, on Blackboard

### Graphs (50 points)

The goal of this project is to practice Java, Java's drawing functionality, graph manipulation, and some graph algorithms. You will implement Breadth-First Search, Depth-First Search, and Bellman-Ford's single source shortest paths algorithm, using a graph stored in adjacency lists. You can use any data structure to store the lists, as long as the overall storage is only  $\Theta(|E|)$ .

### Files

- `project2.zip` contains java code for the project, including a `Vertex` class and basic drawing functionality. Please download this project and add your code to it. If you are using Eclipse you can import the project by importing this archive file directly (File – Import – General – Existing Projects into Workspace – Select archive file; then browse for the filename and click Finish).
- The project directory contains four data files storing example graphs:
  - `graphDirected.dat` is a directed graph with positive edge weights,
  - `graphUndirected.dat` is an undirected graph with positive edge weights,
  - `graphNeg.dat` is a directed graph with negative edge weights but without negative-weight cycles, and `graphNegCycle.dat` is a directed graph that contains a negative-weight cycle. The format of these files is as follows:
    - All numbers in the files are integers separated by spaces or line breaks. The first line contains the number  $n$  of vertices. Vertices in the file are identified with integers  $0 \dots n - 1$ .
    - The following lines contain descriptions of vertices and their adjacency lists. Each vertex and its adjacency list are described as follows:
      - \* The first line contains the ID, the x-coordinate, and the y-coordinate of the vertex  $v$  “owning” the adjacency list.
      - \* The second line contains the length  $l$  of the list (so,  $l = \deg(v)$ ).
      - \* The next  $l$  lines each contain a vertex ID and an integer weight (for the edge from  $v$  to the vertex in this line).
    - The vertices “owning” the adjacency lists are listed in increasing ID order, so first vertex 0 and its adjacency list, then vertex 1 and its adjacency list, and so on.

## Tasks

1. (10 points) Implement code that reads a directed weighted graph from a file into an adjacency list structure, and that draws the graph on the screen using the provided routines. Assume the file is given in the format described above.
2. (10 points) Implement Breadth-First Search. The edge weights are irrelevant for this algorithm. The output should at least be a visual output of the resulting tree drawn overlaid on the graph on the screen, as well as the discover times for each vertex.
3. (10 points) Implement Depth-First Search. The edge weights are irrelevant for this algorithm. The output should at least be a visual output of the resulting tree drawn overlaid on the graph on the screen, as well as the discover and finish times for each vertex.
4. (10 points) Implement Bellman-Ford's algorithm. In the absence of negative-weight cycles, the output should minimally include a) the source vertex, b) the shortest path tree, and c) the resulting d-values, and should be drawn overlaid on the graph on the screen. If the graph contains a negative-weight cycle, the algorithm should detect this and output a corresponding message.
5. (10 points) Test your algorithms using an appropriate set of input graphs, and write a short report. You may want to add screen shots to the report. For your tests, use at least the graph data files provided with the project. Run BFS, DFS, and Bellman-Ford on the same inputs and compare the outputs. What happens in the case of negative weights or negative weight cycles?

## Turnin instructions

- For this project, you have to use Java and you have to extend the provided project file.
- You may use basic data structures that come with Java, in particular for storing lists or queues. However, do not use any other fancy libraries.
- Please include comments on how you compiled the project, and how to run tests.
- **The name of your project directory should be**  
`project2_<lastName><firstName>`
- Zip up a directory with your entire project (source code and report). Turn in the zip file on Blackboard.
- All projects need to compile and run. If your program does not compile and run you will receive 0 points on this project.