INTRODUCTION TO PROGRAMMING
DM550, DM562, DM857, DS801 (Fall 2020)
Exam project: Part I — Deadline: 23h59 on Friday, November 6th, 2020

Overview

In this phase of the project, your task is to implement the top-level class `RunSimulation`. This class should be executable, i.e., it should provide a `main` method that, when invoked, prompts the user for all the simulation parameters, initializes the graph, the set of ants, and the appropriate instances of `Simulator` and `Visualizer`. It then performs a simulation loop, iteratively running the simulation for the desired number of steps, and displaying information according to the user's specification (see details below).

The parameters to be provided are the following:

- the probability that a node will start as a node containing sugar;
- the average amount of sugar in such a node;
- the number of units of sugar that an ant can carry;
- the number of units of pheromones dropped by an ant when passing through a node;
- the total number of ant colonies;
- the number of ants initially at each colony;
- the type of graph to use (randomly generated grid or read from a file – check the documentation of class `Graph`);
- additional information needed to generate the graph:
  - for a grid, the width and height of the grid;
  - otherwise, the name of the file storing the graph;
- the total simulation time;
- the desired mode of viewing information:
  - a textual summary of the state of the simulation printed at regular intervals (the frequency of these reports should be given by the user);
  - a graphical representation updated at each tick.

Since each ant must belong to a colony, the total number of colonies must be at least 1. However, it is possible to have 0 ants belonging to any colony.

Important notes.

- Start by carefully reading the documentation for all the classes provided, since they have methods that do most of the work required. Class `RunSimulation` is mostly responsible for interacting with the user and calling methods from other classes in the appropriate order.
- Class `Visualizer` works with any graph, but it only draws graphs that are grids. It is correct to choose graphical representation for non-grid graphs, but the results might be disappointing :-)
- The source material includes a file `graph1` that can be used as input if the user wants to read the graph from a file.

Expected results

You must hand in a zip file containing no subdirectories and the following two files.

- A Java source file `RunSimulation.java` implementing the functionality described above. Your class should be a client of the remaining classes, which are provided in compiled form. It should run directly from the command line, print all relevant information on the screen (except if the user chooses the graphical visualization mode), and ask for any relevant input using only command-line interaction.
- A pdf document `report.pdf` containing a report describing the algorithm implemented, any design choices that you think are important to document, and examples. One of the members of the group must be clearly identified on the title page as the coordinator for this part of the project. The source code for the developed classes should be included as appendix.

If you do not follow these rules, your project may be rejected and not graded. The report will be the basis for the evaluation.

Testing and examples

Include some examples of how your program works. Rather than copying the output of a complete run in your report, you should describe any auxiliary tests that were made and explain how they can be used to ensure that specific parts of the code are running as expected.