

Compulsory assignment 3 - INF 102 - Autumn 2016

Deadline: Nov 18th , 16:00

Organizational notes

This compulsory assignment is an individual task, however you are allowed to work together with at most 1 other student. If you do, remember to write down both your name and the name of team member on everything you submit (code + answer text). In addition to your own code, you may use the entire Java standard library, the booksite and the code provided in the github repository associated with this course.

The assignments are pass or fail. If you have made a serious attempt but the result is still not sufficient, you get feedback and a new (short, final) deadline. You need to pass all (3) assignments to be admitted to the final exam.

Your solutions (including all source code and textual solutions in PDF format) must be submitted to the automatic submission system accessible through the course before Nov 18th , 16:00. Instructions on how to submit will follow shortly. Independent of using the submission system, you should always keep a backup of your solutions for safety and later reference.

If you have any questions related to the exercises, send an email to:

`gunnar.schulze@ii.uib.no`

In addition you have the opportunity to ask some questions in the Friday review session and at the workshops.


Amaz(e)ing meeting

A boolean maze (see Figure 1) consists of different tiles on a rectangular grid where each tile has 4 arrows (north, south, west, east) pointing to its neighbouring tiles. There are two kinds of tiles - those that can be passed (represented by 1) and those that cannot (represented by 0). Any path through the maze can only contain tiles (vertices) that have value 1.

Construct a graph from the boolean maze resulting from the matrix in the file `simple_maze20x30.txt`. This maze consists of 20 rows and 30 columns and has two entrances (marked by 1's occurring at the border). Represent the tiles as vertices and connections between tiles (arrows) as edges.

Consider the problem of two persons wanting to meet in the middle of the maze. Both are a bit late and need to find the most efficient route to meet the other person. In this case "efficient" means the least number of steps for both, i.e. they meet

Boolean maze

0	1	0	0	0	0	0
0	1	1	1	0	1	0
0	1	0	1	1	1	0
0	1	1	0	0	1	0
0	0		1	1	0	0
0	0	1	0	1	1	0
0	0	0	0	0	1	0

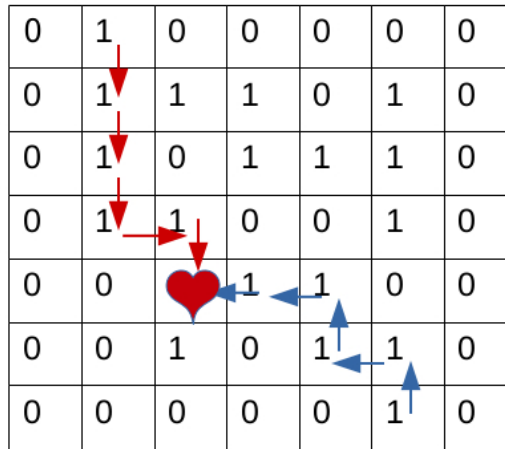


Figure 1: *Example for a meeting in a boolean maze*

in the middle ± 1 tile. Develop and implement a method to find a pair of such paths and apply your method to the graph constructed from the maze. Print out the paths you found.

Hint: Have a look at `Maze.java` on the booksite for ideas on how to represent walls.

Preorder vs Postorder DFS

Perform depth-first search (DFS) on the graph `tinyDAG.txt` provided in `algs4-data.zip` archive.

Implement the following print methods for depth-first search:

- 1 Preorder: print the nodes in the order in which DFS arrives at them.
- 2 Postorder: print the nodes in the order in which DFS leaves them.
- 3 Reverse postorder: After the recursive call to `dfs()` returns, add the node currently considered to a stack. Print the stack after the top call to `dfs()` returns.

Answer the following question: Which of these print methods provides a topologically sorted output, provided the graph is directed and acyclic (DAG)?