Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Ernst W. Mayr
Chris Pinkau, Stefan Toman, Jeremias Weihmann

Sommersemester 2013
Aufgabenblatt 1
15.04.2013

Praktikum Diskrete Optimierung

Due date: Monday, 22th April 2013, 14:00

Aufgabe 1 (First steps with LEDA)

Get used to the LEDA-library. Create a folder which you use for your programs and a folder where you save your final solutions and copy the Makefile as well as the source files `dfs.cpp` and `control.h` from the website of the practical course. Now, it should be possible to compile the program `dfs.cpp` by `make dfs`.

Aufgabe 2 (Breadth first search `bfs`)

Implement and animate the breadth first search on connected undirected graphs by using the LEDA framework. To display the graph on the screen use the class `GraphWin`. Use the program `dfs.cpp` as a template. The user should be able to select a start node s with the mouse, and your program should traverse the graph, and compute for all other nodes v the length of a shortest path (w.r.t. the number of edges) from s to v.

The user should be able to follow in which order the program visits the nodes, which nodes are already visited, and which nodes are currently stored in the queue. To this end, use the visualization capabilities which are provided by `GraphWin` for nodes and edges.

The distance of the nodes from the start node s should be visualized on the screen as user labels. The edges which are part of the breadth first search tree should also be highlighted in an appropriate way.

Aufgabe 3 (Topological sorting `topsort`)

Implement and animate an algorithm which accepts any directed graph $G = (V, E)$ as an input, and tries to assign a unique number $f(v) \in \{1, 2, \ldots, |V|\}$ to each node $v \in V$ such that, for each edge $(u, v) \in E$, $f(u) < f(v)$ holds. (That is, for each edge, the start node of the edge must have a smaller number than the end node of the edge.)
Such a numbering exists if and only if the graph doesn't contain a (directed) cycle. If the input graph does contain a cycle, then your program should color one of the cycles. Try to implement this algorithm as efficient as possible!