Data Structures and Algorithms II 6./7.4.2017

Exercise 2

No obligatory X-exercise yet this week, but next week there will be an X-exercise on directed graphs.

In the following "write an algorithm" tasks, you should make a working Java method that gets as parameter input and possibly returns a new collection in accordance with the assignment, but does nothing else. Thus, for example, does not alter the input data (unless requested to do so) or print anything (at least in the final version). Please take the input generating main program from course web page. At exercise classes, we'll show your answers using projector, thus bring it with you by saving it to the cs.uef.fi server, somewhere else in the network, or a memory stick.

As the algorithms use trees and/or graphs, we need to use our data structure library which you can find from course www-page. There are instructions how to use it from command line, Eclipse, IntelliJ IDEA, and Netbeans. At cs.uef.fi you can use `trajc` and `traj`.

7. Recap from DSA I course: Write an algorithm that collects all leaf nodes of a binary tree (i.e., those nodes that have no children). Algorithm gets as a parameter a binary tree, and returns a set of binary tree nodes. What is the time complexity of your algorithm? Hint: recursion.

In the following drawing exercises 8-10 it is easiest to create graphs according to the number of edges. Drawing the same vertices and/or edges to different order/place does not make graphs different. Be careful not to draw the same graph twice.

8. Draw all different directed graphs that have exactly two vertices and no multiple edges between two vertices in the same direction. Graphs may have loops (edge from a vertex to itself). (Hint: 10 graphs).
9. Draw all different directed graphs that have exactly three vertices and no multiple edges between two vertices in the same direction and have no loops. (Hint: 16 graphs).
10. Draw all different strongly connected directed graphs that have exactly four vertices and no multiple edges between two vertices (in same or different direction) and have no loops. Strongly connected means that there is a path from every vertex to every other vertex. (Hint: 4 graphs).

Use the *GraphMaker* class from course www-page to create example graphs. The skeleton and examples have examples how to use it.

11. Write an algorithm that finds all the vertices that are reachable from a given vertex. Parameters are a graph and a vertex, and return value is a set of vertices. Hint: depth first search. What is the time complexity of your algorithm?
12. Write an algorithm that finds any path (list of vertices) between given two vertices in a directed graph. Do not use Dijkstra algorithm, but depth first search. What is the time complexity of your algorithm?