Data Structures and Algorithms II 4./5.5.2017

Exercise 5

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`.

25. Write a randomized algorithm that colours given undirected graph with as few colours as possible so that no neighbouring vertices have the same colour. Minimal colouring is NP-hard, but now we make an algorithm that finds a reasonably good solution in reasonable time. Use greedy algorithm (from course www-page) and repeat it given time (e.g., 10 seconds) using different random iteration orders. The greedy colouring algorithm uses single colour to colour as many vertices as possible, and it takes a new colour if some vertices are still uncoloured. This is then repeated using several orders of iteration. The best found colouring is returned as the result of the algorithm. You can use method *Permutation.randomPermutation*() from course www-page as a helper. Collect all vertices to an array (*GraphMaker.getVertexArray*()) and iterate the array in order of a permutation.

26. Modify the algorithm of task 25 to find the minimal colouring. Minimal colouring can be found by trying all possible iteration orders. You can generate all permutations with class *Permutation* in course www-page. What is the time complexity of the solution? Calculate (extrapolate) running time in seconds (etc.) when $n=20$, $e=70$.

27. Add to dynamic coins algorithm (*Coins.dyncoins*()) the calculation and return of the actual coins for a given sum. Coins are stored to an array *coin* during the solution as in Dijkstra algorithm. *coin*[*i*] contains the value c of the first coin for money sum i. The next coin can be found from *coin*[*i-c*], etc.

28. A company x has quorum (decision making power) in company y if and only if there exists companies $z_1, z_2, ..., z_k$ that are under quorum of compary x, and companies $x, z_1, z_2, ..., z_k$ combined own over 50% of shares of company y. Such information is needed in, e.g., co-operation negotiations and other legal situations. We model the ownership relations between companies by a graph, where each company is a vertex, and for each company x owning r% of shares company y, there is an edge (x,y) with weight r. Sketch an algorithm (pseudocode) that finds all those companies that are under quorum of company x.

The following task X2 is obligatory for all students. X-tasks must be done **oneself** by each student. Copies/versions of the same answer won't be accepted. Answers must be sent by Wednesday 3.5. 21:00 using the instructions below. You'll receive an automatic reply by email soon after successful submission. If you won't get the email reply, something went wrong. If the reply contains compiler errors, there is something wrong in the file. Then **resend a fixed version**. The answer must contain a short **self evaluation** where you evaluate the functionality, correctness, time complexity, and possible points of improvement of your solution. A correct self evaluation (for a full answer) is worth one point. The points of these tasks form a part of course evaluation.

Send your solution using a www-form, address and credentials of which you got by email. The solution should be a compilable Java source code file of name `userid.java` where `userid` is the first part of your email address. Also the **class name** of your solution must be `userid`. As the submission is Java source code, the self evaluation must be in comments of the program.

Take a skeleton from course www-page. Do not change the header (name, parameters) of the X-task method(s). Please make sure that the program/class is compilable as such, i.e., have the whole answer in the same class and do not use a package.

X2. Write an algorithm that returns a cycle of an undirected graph that contains a given vertex. Parameters are a graph and vertex v, and the return value is a list of vertices (*LinkedList*<*Vertex*>) of one cycle that contains vertex v, or *null* if there is no such cycle. The elements in the list must be in order of the cycle. The vertex v must thus be the first and the last element of the list. What is the time complexity of your algorithm?