Deadline: 5/12/2011

- See https://science.swansea.ac.uk/intranet/help/students/coursework for how to submit.
- See the course home-page http://www.cs.swan.ac.uk/~csoliver/Algorithms201112/index.html for general information on the course work.
- If you have collaborated with another student, clearly state his/her name and student number.
- The possible marks for individual exercises sum up to 130; results over 100 will be capped.

The complete coursework must be written in its entirety by you. If you used external sources, then these must be stated.

Since we cannot return your coursework to you, please keep a copy for yourself.

1. Run DFS on the following dag, and show
   (a) for each vertex \( v \) the times \( d[v], f[v] \);
   (b) the resulting “spanning forest” (which is not really a forest, since we do not have (undirected) graphs, but digraphs);
   (c) the resulting topological sorting.

Use numerical order to determine the order of vertices and edges. Explain also in words:
- how you computed \( d[v] \) and \( f[v] \) (in one paragraph);
- how you determined the topological sorting (in one sentence).

![Diagram](attachment:diagram.png)

2. Imagine that you have stored a big digraph \( G \).
   (a) How can you certify (prove) in a succinct way that \( G \) is not a dag?
   (b) And how can you certify (prove) in a succinct way that \( G \) is a dag?

In both cases the certificate should not involve computations, but should be some data for which one can easily determine whether it is a valid certificate or not (given access to \( G \)).

State a good upper bound on the length of the certificate in both cases, using big-Oh and the number of vertices and edges of \( G \).

3. Show all binary search trees with four nodes, labelled by 1, 1, 2, 2. Argue that your enumeration is complete.

4. Insert the elements 3, 8, 9, 5, 1, 7, 2, 10, 4, 6 one after another into a binary search tree (starting with the empty “tree”).
   (a) Show the sequence of binary search trees obtained.
   (b) What is the height of this tree?
   (c) Delete the root node of this tree, and show the new binary search tree.

5. Describe a heuristic that can be used with the disjoint-set forest data structure which ensures that every sequence of \( m \) MAKE-SET, UNION and FIND-SET operations, \( n \) of which are MAKE-SET, can be performed in time \( O(m \lg n) \). Prove carefully that the data structure together with your heuristic achieves the stated bound on the runtime.

6. Consider handling big graphs \( G \), with billions of vertices. We want to determine the total number of connected components, and we also want to know the sizes of the connected components (the number of vertices in them). So we assume that we can store the vertices in memory (otherwise we couldn’t do the computation), but we can not store the edges upfront. Instead we want to deliver them later, in small chunks. Such a mode of computation is called “online”.
   (a) Can we (somehow) use BFS or DFS in online-mode? Describe when which edges are to be delivered to BFS/DFS, and how many edges need to be stored.
   (b) What about using disjoint-sets for computing connected components? What requirements do we have here on edge access?
   (c) Compare BFS, DFS and disjoint sets for computing connected components: Which is best for online-purposes?