Universitetet i Oslo
Institutt for Informatikk

A. Maus, R.K. Runde, I. Yu

INF2220: algorithms and data structures

Series 7

Topic More on graphs: DFS, Biconnectivity, and strongly connected components

Issued: 05. Oct. 2017

Classroom

Exercise 1 (Biconnectivity) Given the graph of Figure 1,

1. Is the graph biconnected?
2. If not, which node(s) is (are) the articulation point(s) in the graph? Show the depth-first search spanning tree. Indicate *back edge(s)* if applicable. You should also indicate the value of *Low(v)* and *Num(v)* for each node v in the graph.

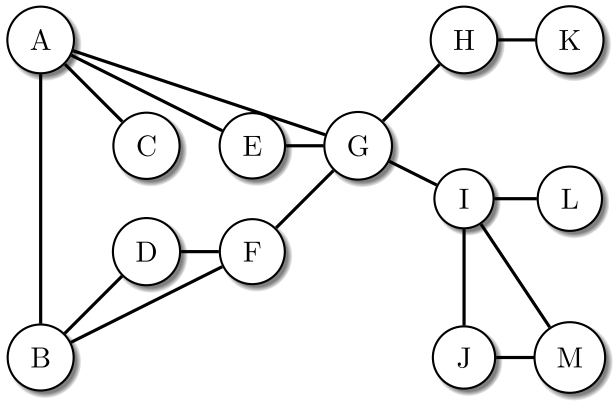


Figure 1: Connected, undirected Graph for Exercise 1

Exercise 2 (Strongly-connected components) In the lecture, we learned to compute strongly-conneced components (scc's) for a directed graph.[1] Show how this is done for the graph in Figure2.

[1]Works as well for undirected graphs, obviously, but the problem there is boring.

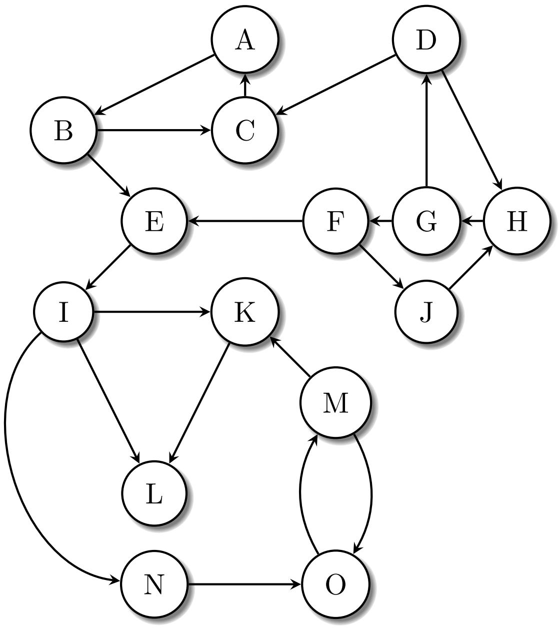
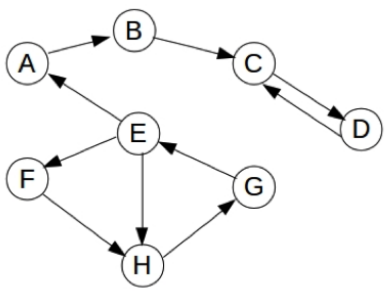
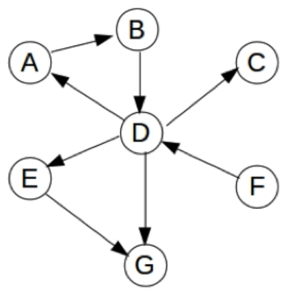


Figure 2: Directed graph for the scc exercise 2

Exercise 3 Find the stronly connected components of the following graphs. Indicate the steps in the algorithm.





Exercise 4 (Christmas party-2015 exam) In this problem you will plan the seating arrangement on a Christmas party. You have a list V of guests of type Person:

```
class Person{
 int id;

  ...

}
```

4.1 Seating arrangement

Assume that you are also given a lookup table T where $T[u.id]$ for $u \in V$ gives a list of the guests (of type Person) that u knows. If u knows v then v also knows u. Your task is to make a seating arrangement so that each guest at a table knows all the others sitting at the same table either directly, or through other guests sitting at that table. For example, if x knows y and y knows z, then x, y og z can sit at the same table.

1. If you should represent this as a graph problem, what kind of graph will that be? And what will be represented by the vertices and the edges in that graph?

2. Implement an efficient graph algorithm that, given V and T as input, returns the smallest number of tables that are necessary to satisfy that requirement.

3. What is the running time of your algorithm? Justify briefly.

4.2 Enemies

Assume that there are only 2 tables, and that you are given another lookup table S where $S[u.id]$ for $u \in V$ gives a list of guests that have a bad relationship with u. If v is in a bad relationship with u, then u is also in a bad relationship with v. Your task is to make a seating arrangement so that no guests sitting at the same table are in a bad relationship with each another. (In this problem we will not take into account whether the guests know each other.)

Figure 3 is showing two graphs where the guests are represented as vertices and an edge between two vertices means that the corresponding guests are in a bad relationship with each other. For graph **(a)** we see that it is possible to have A and C share one table and B, D and E share another table, while for **(b)** we see that this is impossible.

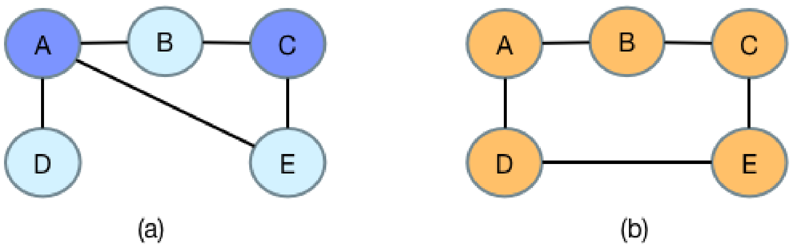


Figure 3:

Implement an efficient algorithm that given lists V and S as input returns *true* if we can place all the guests at two tables, and returns *false* otherwise.

Lab

Exercise 5 (DFS) Implement an *iterated* DFS; it should work for directed and indirected graphs. Do the DFS in such a way, that visiting times and finishing times are remembered in the nodes; and printed as well (together with the corresponding node name). The printing can be done during the traversal ("on-the-fly").

In the second half: use the implementation to solve one of the following, or, if you have time and energy, both:

1. *Topological sorting.* The input is a directed gragh. In case the graph is *cyclic*, give back an appropriate message to the user.

2. Biconnectivity, where the input is an undirected graph.