IN.5022 — Concurrent and Distributed Computing

Series 5 and 6

Due dates: 26.10.2023 and 2.11.2023, 12:00, on Moodle

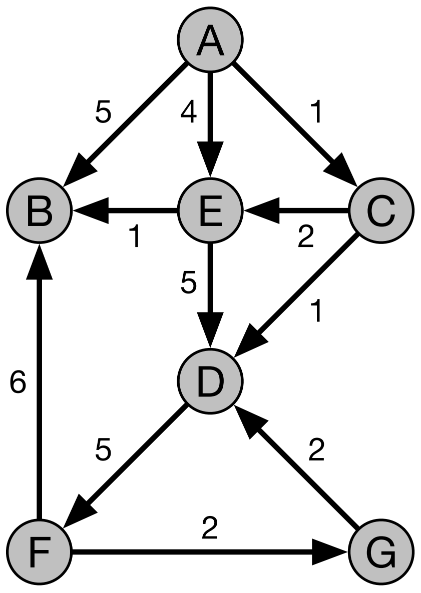
First of all…

Download the paper from Chandry-Misra and read it (available on Moodle or online at https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.104.8179).

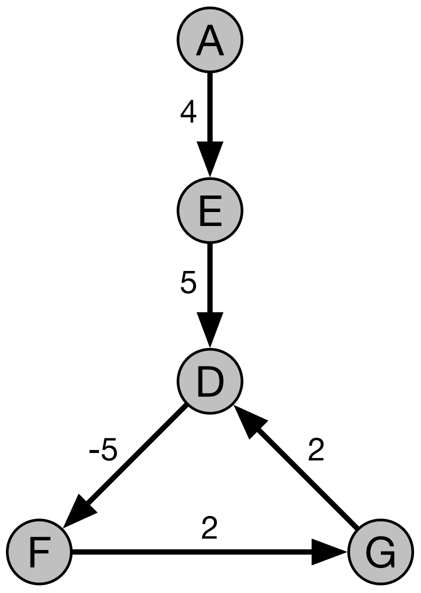
Exercise 1 (due 26.10.2023)

Compute Chandy-Misra's shortest path algorithm for non-negative graphs by hand, starting from a source *A* to all other nodes of the directed weighted graphs in the figure below. Write the steps of the algorithms and the messages exchanged using the working sheet template provided on Moodle for the following scenarios.

1. Without termination (i.e., without acknowledgment messages) on the first graph (left).
2. With termination (i.e., with ack and stop messages) on the first graph (left).
3. With termination on the second graph (right), i.e., with negative weights. Does the algorithm still produce correct results?



Graph without negative weights.



Graph with negative weights.

Exercise 2 (due 2.11.2023, feedback provided if returned by 26.10.2023)

Write an Erlang or Elixir program that implements the Chandry-Misra distributed shortest path algorithm using the template provided on Moodle. For simplification we make the following assumptions:

1. The network topology is statically configured in the program (provided in the template).
2. The network is simulated in a single process, i.e., no need to deploy the program on multiple machines.

3. Processes simply output their shortest path to the console, i.e., there is no need to send them back to the source or write them into a file.

In this first implementation, we assume a topology without negative edge, and you can omit termination (phase II), i.e., acknowledgements are not necessary. You can use a timeout to let enough time for the algorithm to complete before printing the shortest paths.

Exercise 3 (due 2.11.2023)

Extend the program of the previous exercise to support termination (phase II).

Exercise 4 (optional)

Extend the program of the previous exercise to support negative cycles and test with the alternative topologies provided in the template.

Exercise 5 (optional)

Deploy and test your programs on multiple machines (or *Dockers*).