Introduction

In this TP we will implement the fundamental algorithms studies during the course in Erlang using arbitrary topologies.

Question 1

a) Build a utility function that can be used to spawn a graph topology. It should take as an input some graph representation of your choice (e.g. adjacency list, assumed to represent an undirected graph) and return a dictionary that assigns respective PIDs to the vertices of the graph.

Question 2

a) [Echo algorithm] Implement the algorithm that performs the traversal of the topology. Initiator process is chosen randomly and starts a wave in the network by sending a request to its neighbours. An arbitrary non-initiator process propagates the request to its neighbours (all except the one who sent the request on the first place) and responds to the request with its token (some value unique to every process) and the tokens collected from all its neighbors. The algorithm terminates when the initiator received the tokens from all its neighbors.

b) Adapt the algorithm to compute the sum over some values stored on the processes.

c) [Leader election with known identities] Implement leader election algorithm on the network that chooses as the leader the process with the smallest PID. How would you implement such leader election without using the identities of the processes?

d) Assume that multiple initiators can start waves of the echo algorithm. How would you adapt the previous leader election algorithm to work in this setting. The algorithm should ensure that, no matter how many waves are started, only one wave will run to a conclusion (elected leader), namely, the wave of the initiator with the smallest PID.

e) [Termination detection] Assume that an initiator node starts a wave that results in every process computing some function (which takes some time to compute). Implement an algorithm that using repetitive waves from the initiator detects if all the processes finished computing their functions.

Question 3

a) Modify your topology spawning utility to be able to spawn both directed and undirected networks.

b) Write a function that takes as an input the graph representation of your choice and computes the diameter of the graph.

c) [Phase algorithm] Let \( D \) be the diameter of the graph underlying the network topology. Each process sends exactly \( D \) messages to each out-neighbor. Only after \( i \) messages have been received from each in-neighbor is the \((i + 1)\)-th message sent to each out-neighbor. See more details in the following pseudocode:
**Data:** $D$ - diameter of the network, $Out_p$ - out-neighbors of $p$, $In_p$ - in-neighbors of $p$, $Rec_p$ - number of received messages, $Sent_p$ - number of sent messages

\[
Rec_p := 0;
Sent_p := 0;
\]

if $p$ is initiator then
def each $n \in Out_p$ do
\[
\text{send message to } n;
Sent_p := Sent_p + 1;
\]
else
end
\[\text{while } \min_{n \in In_p} Rec_p[n] < D \text{ do}\]
\[
\text{receive from } m;
Rec_p := Rec_p + 1;
\]
if $\min_{n \in In_p} Rec_p[n] \geq Sent_p$ and $Sent_p < D$ then
\[\text{foreach } n \in Out_p \text{ do}\]
\[
\text{send message to } n; Sent_p := Sent_p + 1;
\]
end
end
decide;

**Algorithm 1:** Phase algorithm

d) Assume each process $p \in P$ stores some value $i_p$. Adapt the phase algorithm to computing the set $S = \{(p, i_p) : p \in P\}$ and the sum over the inputs.