If you have questions regarding the exercises, please ask them on the mailing list.
Please hand in your solutions by sending them to Johannes at jbund@mpi-inf.mpg.de or directly in the lecture. If possible form groups of 2-3 people.
Do *not* send them to the public mailing list. The deadline is listed on the website.
Exercises marked with a star are not mandatory.

Exercise 2: Flirting with Synchrony and Asynchrony

Task 1: Growing Balls

In this exercise, we will see how a crucial step of the γ-synchronizer works; specifically, that a desirable partition of the nodes exists.

Denote by $B(v, r)$ the ball of radius r around v, i.e., $B(v, r) = \{u \in V : \text{dist}(u, v) \leq r\}$. Consider the following partitioning algorithm. Note the difference between int*er*cluster edges and int*ra*cluster edges.

Algorithm 1 Cluster construction. $\rho \geq 2$ is a given parameter.

while there are unprocessed nodes **do**
  select an arbitrary unprocessed node v;
  $r := 0$;
  while $|B(v, r+1)| > \rho |B(v, r)|$ **do**
    $r := r + 1$
  end while
  makeCluster($B(v, r)$) // all nodes in $B(v, r)$ are now processed
  remove all cluster nodes from the current graph
end while
select intercluster edges

a) Show that Algorithm **??** constructs clusters of radius at most $\log_\rho n$.

b) Show that Algorithm **??** produces at most ρn intercluster edges.

c) For given cluster radius $k \in \{1, \ldots, \lfloor \log n \rfloor\}$, determine an appropriate choice $\rho(k) \geq 2$, proving the precondition of Corollary 2.14!

   Hint: As a short-hand, we often don't write out common terms like n that are assumed to be globally known. Specifically, $\rho(k)$ may also depend on n, as if we had written $\rho(k, n)$. If in doubt, then we weren't clear enough, so tell us!

Task 2: Showing Dijkstra, and Bellman & Ford the Ropes

a) Show that if the asynchronous Bellman-Ford algorithm from the lecture is executed synchronously, it sends only $\mathcal{O}(|E|)$ messages.

b) Use this to construct an asynchronous BFS tree construction algorithm of time complexity $\mathcal{O}(D)$ that uses $\mathcal{O}(|E|D)$ messages and terminates. You may assume that D is known here.

c) Can you give an asynchronous Bellman-Ford-based algorithm that sends $\mathcal{O}(|E| + nD)$ messages and runs for $\mathcal{O}(D^2)$ rounds?

   Hint: Either answer is feasible, provided it is backed up by appropriate reasoning!

Task 3*: Liaison with Leslie Lamport

a) Look up what the happened-before relation, Lamport clocks, and vector clocks are.

b) Contemplate their relation to synchronizers and what you've learned in the lecture.

c) Discuss your findings in the exercise session!