Distributed Algorithms - Homework 1

1. Give a definition for the time complexity of an asynchronous algorithm: synchronous time, longest-chain time, bounded delay time. Give example(s) that show that these three definitions are distinct.

2. The question relates to Dijkstra’s 1st algorithm for mutual exclusion for ring networks.
   
   (a) We saw in class that Dijkstra’s 1st self-stabilizing algorithm for mutual exclusion on a ring network, that is using \( k \)-state machines, self-stabilizes iff \( k \geq n - 1 \) with a centralized scheduler. Prove a similar claim for a case of a distributed scheduler, but for \( k \geq n \). (Recall that a ‘distributed’ scheduler is a scheduler that at each step allows any non-empty subset of privileged processors to simultaneously make a move.)

   (b) Prove: \( \forall N \geq 4, \forall K, 1 < K \leq N - 2 \), there is an input with \( N \) processors and an execution in which the system never stabilizes.

   (c) Estimate the number of steps that it takes the system to self-stabilize in the worst case, for the following cases:
      - Dijkstra’s 1st algorithm with \( k=n \) under a centralized scheduler
      - Dijkstra’s 1st algorithm with \( k=n \) under a distributed scheduler

   Note: In each case, give a function \( f(n) \) which is as bad as possible, and show a family of initial configurations with \( n \) processors which takes \( g(n) \) steps to stabilize, for infinitely many \( n \)’s. Give functions \( f(n) \) and \( g(n) \) which are as close as possible to each other.

3. The question relates to the algorithm discussed in class on clock synchronization.

   Assume that in a synchronous network each processor has a local clock. Each clocks tick at the same rate, but each starts at a different time. At each time pulse, a processor has to change its local clock, according to its current time and the time shown on clocks of its neighbors. The aim is to get in a finite number of steps to a situation where all clocks are equal to some \( k \), in the following step they are all equal to \( k + 1 \), and so on.

   Denoting by \( t[i] \) the local time of processor \( i \); the protocol for each processor \( i \) is as follows:

   - choose fairly a neighbor \( x \).
   - if \( t[i] \leq t[x] \) then \( t[i] := t[i] + 1 \)
• Prove that then the algorithm stabilizes.

• Denote by $T(n)$ the time it takes the system to self-stabilize in the worst case, on a network of size $n$, when the neighbors are chosen in a round-robin fashion. Give the best estimate you can for $T(n)$, prove its correctness and give a family of examples where $T(n)$ is as large as possible, for infinitely many $n$’s.