Note: YOU MUST DO THE HOMEWORK BY YOURSELF. If you have difficulties in solving a question you may discuss it with friends, BUT you MUST phrase, write and formulate the answers by yourself, after you understand the solution. The language of the solution must be entirely your own.

Notice 2: Unless your hand writing is extremely clear, you should type (Word, or LaTeX, etc.) and print your homework.

Notice 3: Put your name and id number on each page of the solution.

1. Is it possible to elect a leader in a synchronous ring with \(n\) nodes, \(n\) known to the nodes, node ID’s are unique (no two ID’s are the same), but the only operation the processors can do on the ID’s is \textit{equal} or \textit{not_equal} comparisons. Provide a complete proof of your answer.

2. (a) Give the most efficient ring orientation algorithm you can come up with in the message passing model, as follows:
   The ring is anonymous and has arbitrary size. Since the ring is anonymous we refer only to algorithms that message terminate, that is: that reach a state in which no message is in transit and all processors are in idle state. But no processor is suppose to be able to detect the fact that such a state has been reached. Further assume the following:
   
   i. The number of processors on the ring is unknown.
   
   ii. Each link on the ring is equipped with a \textit{symmetry breaking marking}. That is, each pair of neighbors has agreed on an ordering between them prior to the beginning of the algorithm (such an order can be marked for example by a unique ”head” mark on each link).

   In the algorithm each node locally marks one of its incident links ”left” and the other ”right”. The problem is to design an orientation algorithm that will assign those marks at the nodes in such a way that on every link one end is marked ”left” and the other is marked ”right”.

   (b) Assume now that the \textit{symmetry breaking markings} are \textit{not} available and give a message terminating randomized orientation algorithm (a Las-Vegas algorithm, see definition below), with as good expected complexity as you can.

3. Based on the centralized Depth First Search (DFS) procedure, describe a distributed DFS traversal algorithm for an asynchronous network. In a traversal algorithm a central node, called \textit{root}, initiates a token which has to visit all the nodes of the network, one at a time. Argue that in order to visit all the nodes the token must traverse all the links of the network. (links might be traversed more than one time)
(a) Verify that your algorithm uses \( O(|E|) \) messages and \( O(|E|) \) time, where \(|E|\) is the total number of links in the network.

(b) Modify your algorithm to reduce its time complexity to \( O(n) \), where \( n \) is the total number of nodes in the network. (Hint: in the modification there must be times at which there are more than one message in transit in the network.)

(c) Modify the algorithm of (Section a) to implement a Directed DFS on a strongly connected unidirectional network. In a unidirectional network some or all the links can carry messages only in one, predetermined, direction. A unidirectional network is strongly connected if there is a path from any node to any other node. What is the message complexity of the new algorithm? (Hint: use the unique path connecting the root with the token location, to perform the backtracking). A solution exists even if all nodes are identical (no unique ids) except the initiator which is single out. If you cannot find such a solution, try to provide a solution assuming nodes have unique ids.

4. Use the DFS algorithm of the previous question (asynchronous, Bidirectional network, with unique id to each node) to derive an \( O(|E| + n \log n) \) messages election algorithm. HINT: Each initiator of the algorithm would start a DFS traversal. The question is how would colliding DFS traversals efficiently eliminate one another until only one DFS is left, which then captures the entire network and is elected as the leader.

5. Design a distributed algorithm which determines, while being executed on a (connected) network \( G = (V; E) \), whether \( G \) is a complete graph or not. More formally, a node only knows its own identity and which edges are incident on it at the beginning of the algorithm. It does not even know the identities of its neighbors. At the end of the algorithm, a single node in the network knows whether a structure of interest is present or not. In our case there is one and only one processor at which the algorithm is initiated and all other processors remain inactive until they receive a message. The algorithm should use \( O(|E|) \) messages of \( O(\log n) \) bits each and \( O(d) \) time, where \( d \) is the (actual) diameter of \( G \).