CS4438/9668 Internet Algorithmics
Assignment 2: Graduate and Undergraduate Students
Due October 25 at 11:55 pm

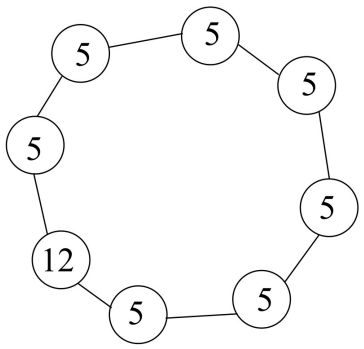
For each one of the last three questions you need to use the simulator for synchronous distributed algorithms written by Daniel Servos. You can download the simulator and read the documentation for it from the link posted in the OWL site (FAQ → Documentation for the Simulator of Synchronous Distributed Algorithms).

You need to submit through OWL the java files implementing your algorithms and a document in pdf containing your answers to the questions.

If you do not have any programming experience and are not able to write java code, we will allow you to submit your algorithms in detailed pseudocode similar to the one used in the lectures. However, this should be the exception, not the rule; you are strongly encouraged to write your algorithms in java and test them using the simulator. A test file will be provided for each algorithm. Additional tests will be conducted on your code to assess its correctness.

1. (Optional 10 marks) Consider a ring network of $n > 1$ processors in which $n - 1$ processors have the same identifier and one processor has a different identifier; the identifier of this last processor is larger than the identifiers of all other processors. An example of such a network is shown in the followig figure. In class we discussed an algorithm for the leader election problem on a synchronous ring with communication complexity $O(n \log n)$. Assume that this algorithm is modified so that whenever a processor with identifier id receives a message of the form $\langle id', d', R \rangle$, where $id = id'$ and the value of variable $d = 1$ then the processor changes its status to NotLeader and then it discards the received message. For completeness, the modified algorithm is given in the last page of this assignment; the modification is highlighted in red.

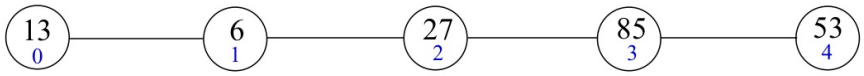
   Compute the time complexity and communication complexity of the algorithm when executed on this kind of network. You need to explain how you computed the complexities and you need to indicate the order of the complexities.



In all problems below it is assumed that each processor has a unique identifier. Also, each processor knows who its neighbours are, but it does not know the number of processors in the network. Whenever you are asked to compute the time complexity or communication complexity of an algorithm **you must explain** how you computed the complexities and you must also give the order of the complexities.

2. Consider a set of processors connected in a line. Complete the provided java class `Position.java` by implementing a synchronous distributed algorithm called `findPosition(String id)` that returns the number of processors to the left of each processor.

   Assume that each processor knows its right neighbour (if any) and its left neighbour (if any). Please read the notes at the end of the assignment to learn how to specify the configuration file for the simulator and how a processor can determine whether it has only a right neighbour or only a left neighbour. The small numbers in blue in the lower part of each node in the following figure show the expected output of your algorithm.

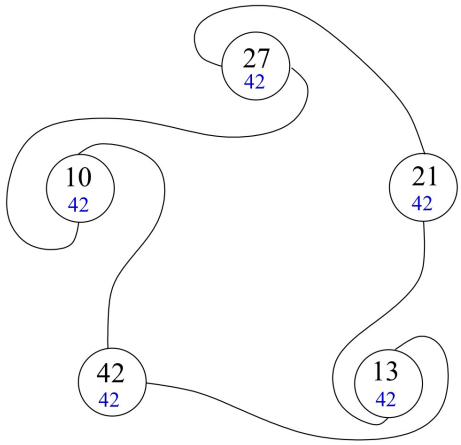


   - (4 marks) Give a high level description of the algorithm in English.
   - (14 marks) Submit a java implementation of your algorithm.
   - (5 marks) Compute the time complexity and communication complexity of your algorithm; denote by n the number of processors.

3. Complete the provided java class `LargestID.java` by implementing a distributed synchronous algorithm called `findLargest(String id)` that solves the leader election problem on a synchronous ring, by returning the largest processor identifier in the network.

   This problem is the same discussed in class except that now a processor knows who its two neighbors are but it does not know which processor is on the left and which processor is on the right, so the algorithms discussed in class will not work.

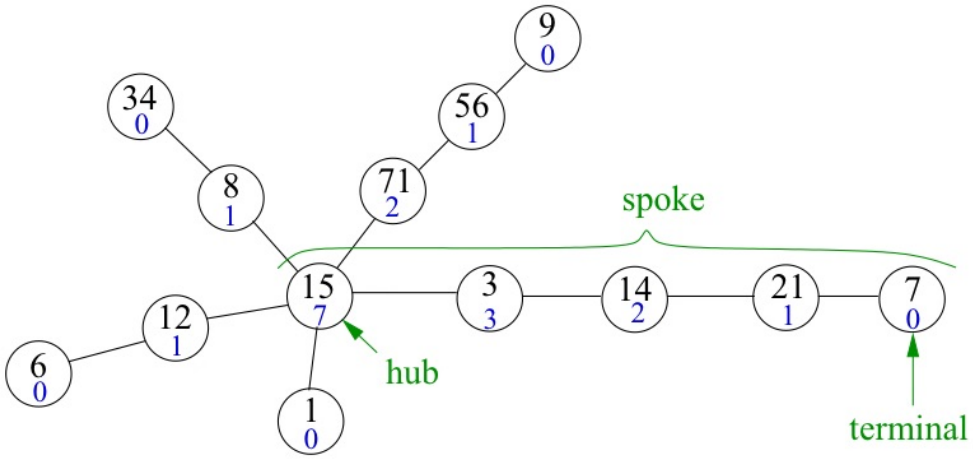
   The following figure shows an example of a ring network, where the expected output from your algorithm are the small blue numbers that appear in the lower part of each node.



- (4 marks) Give a high level description of the algorithm in English.
- (18 marks) Submit a java implementation of your algorithm.
- (10 marks) Prove that your algorithm terminates.
- (5 marks) Prove that your algorithm produces the correct output.
- (5 marks) Compute the time complexity and communication complexity of your algorithm.

4. Complete the java class `Diameter.java` by implementing algorithm `findDiameter(String id)` that computes the diameter of a *hub and spoke network*. In a hub and spoke network the processors are connected in $k > 2$ lines (called *spokes*) that all start at the same processor called the *hub*. The processor at the other end of each line is called a *terminal*. An example of a hub and spoke network is shown in the following figure.

   The copy of the algorithm being executed by the hub must return the diameter of the network, for all other processors the algorithm must return the distance from the processor to the terminal in its spoke. Remember that the diameter of a network is the maximum distance between two processors in the network. In the figure the small blue numbers that appear in the lower part of each node are the expected output of your algorithm. Note that for this network the diameter is 7 as this is the distance between processors 9 and 7.



- (4 marks) Give a high level description of the algorithm in English.
- (18 marks) Submit a java implementation of your algorithm.
- (8 marks) Prove that your algorithm produces the correct output. (No proof of termination is required.)
- (5 marks) Compute the time complexity and communication complexity of your algorithm.

Simulator Notes

Make it sure that the first line of the network configuration file is

```
Mode Identical
```

- The following comments are for the second question only. Add the following line as the second line of the network configuration file so the simulator correctly displays a network with a line topology:

  ```
  GUILayout line
  ```

  In the configuration file the identifier 0 cannot be assigned to any processor, as 0 has a special meaning in the simulator. When specifying the set of neighbours of a processor, the list of neighbours in the `AddNode` command can include one or more times the identifier 0 to indicate the lack of a neighbour. For example if the network configuration file for a network with a line topology contains the command

  ```
  AddNode 23:  0 54
  ```

  This means that the processor with id 23 does not have a left neighbour and its right neighbour has id 54. You can use the following java code to get the neighbours of a processor and to determine whether the processor is the leftmost processor or the rightmost processor.

  ```
  Vector<String> v = neighbours();
  String leftNeighbour = (String) v.elementAt(0);

  String rightNeighbour = (String) v.elementAt(1);
  boolean leftmostProcessor, rightmostProcessor;
  if (equal(leftNeighbour,"0")) leftmostProcessor = true;
  else leftmostProcessor = false;
  if (equal(rightNeighbour,"0")) rightmostProcessor = true;
  else rightmostProcessor = false;
  ```

- The following comments are for the fourth question only. In the configuration file a terminal has only one neighbour. If a processor has 2 neighbours, the first neighbour is the processor closer to the hub or the hub. So, for example, for the hub and spoke figure above, the following line is in the configuration file:

  ```
  AddNode 14:  3 21
  ```

  processor 14 belongs to the spoke marked in the figure and processor 3 is its neighbour closer to the hub while processor 21 is its neighbour closer to the terminal of their spoke. The following line is also in the configuration file

  ```
  AddNode 3:  15 14
  ```

  the first neighbour, processot 15 is the hub and procesor 14 is the neihgbour of 3 closer to the terminal of its spoke.

  Note that the hub is the only processor that has at least 3 neighbours.

Keep in mind that the algorithm used by the simulator to render the network on the screen might not show the networks in the test files as the above figures. If the simulator does not draw the nodes in the position that you expect, you will have to manually re-arrange them so a network is displayed as you wish.

```
Algorithm LeaderElection3(id)
In: Processor id
Out: Leader if id is the largest identifier in the ring, or NotLeader otherwise
status ← UNKNOWN
d ← 1
mssg ← ⟨id, 1, R⟩
loop {
    if mssg ≠ null then {
        if direction in message is R then send mssg to right neighbour
        else send mssg to left neighbour
        if mssg = ⟨END, 0, R⟩ then return status
        mssg ← null
    }
    Receive messages
    if a message m was received then
        if m is of the form ⟨id', d', R⟩ then
            if id' < id then mssg ← null
            else if id' > id then {
                status ← NotLeader
                d' ← d' − 1
                if d' = 0 then mssg ← ⟨id', 0, L⟩
                else mssg ← ⟨id', d', R⟩
            }
            else if id = id' and d = 1 then status ← NotLeader
            else { // id = id'
                status ← Leader
                mssg ← ⟨END, 0, R⟩
            }
        else if m is of the form ⟨id', 0, L⟩ then
            if id' ≠ id then mssg ← m
            else { // id' = id
                if status = UNKNOWN then {
                    d ← 2 × d
                    mssg ← ⟨id, d, R⟩
                }
            }
        else mssg ← m
}
```