Distributed Algorithms
371-1-0301
Exercise 1

Rotem Frumer & Oren Zaharia
TODO IDs numbers

March 19, 2019

1. (a) We have a synchronous ring, which holds $n-1$ processors with the same identifier and another one with different identifier.
   Therefore, we may thing about an algorithm which looks for the odd identifier, that is the leader.
   We know that $n > 2$, therefore we may conclude the following algorithms.
   Each processor will hold two counters, using that counters the processor can identify the major identifiers.
   Therefore, only one counter will be ≥ 2.
   In each stage, each processor send its id and receive its neighbors IDs.
   After one iteration each node will know what is the major identifier.

Algorithm 1 Leader election

Input $myID$

```
1:  myCounter = 1
2:  oddCounter = 0
3:  send myID to left and right {send is always to both sides}
4:  receive leftID from left
5:  receive rightID from right
6:  if leftID == myID then
7:      myCounter += 1
8:  else
9:      oddCounter += 1
10: end if
11: if rightID == myID then
12:     myCounter += 1
13: else
14:     oddCounter += 1
15: end if
16: if oddCounter == 0 then
17:     receive newID
18:     while newID == myID do
19:         receive newID {wait to receive the leader ID.}
20:     end while
21:     send newID
22: else
23:     if myCounter == 1 then
24:         send myID. {I'm the leader}
25:     else
26:         if myID ≠ leftID then
27:             send leftID. {Left is the leader}
28:         else
29:             send rightID. {Right is the leader}
30:         end if
31:     end if
32: end if
```

(b) No, that not always possible.
Let the following ring be our impossible case:

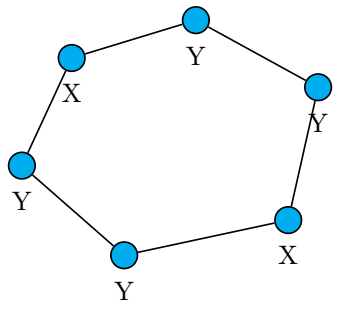

Figure 1: Counterexample ring graph

We have two nodes with identical properties (ID), that is, each phase they will act exactly the same, due to their point of view.

2. In each phase, the number of active nodes is being rolling out by factor of minimum 2.

   Each following phase continue in much the same way. Each active node now sends its new ID (that is the maximum of the last three).

   At the end, the node that sees that the ID it receives from its immediate active "parent" is the as its ID, the it knows that it's the only one left, that is the last phase. In that case, the node elects itself as the leader.

   Therefore, at most $\log n$ phases, and $2n$ messages in phase totally, so $O(n \log n)$.

3. (a)
   (b) An arbitrary node will inititate the process, sends to its whole neighbours an initiate message.
   (c) After receiving the first message, the receiving node sends its ID to its whole neighbours and sum the number of neighobours. The received IDs will be logged.
   (d) Each node, will advertise to its neighbours the number of neighbours from its point of view.
   (e) Each node, will compare its whole received numbers of number of neighbours, and will compare them. If the numbers are identical, continue to step 'e', else return False not-complete graph.
   (f) Each node advertise the maximum from whole received IDs include its own ID. Each node checks the maximum, if there is a mismatch the node advertise False, otherwise True.

   Analsys:

   Each message contains the ID, that is $\log n$ bits length.

   The messages propogates on all edges that are connected to the initiated node, and its next-neighbours. Therefore, $O(|E|)$.

   In complete graph the diameter is 1, therefore $O(d)$ is $O(1)$, our algorithms is by constant time, $O(3)$ that is $O(1)$.