Iowa State University, Computer Science Department

COM S 612
Assignment 1

Ibne Farabi Shihab

September 22, 2020

1 Problem 3.3

Answer. Before going to the answer, we assume that there is no failure happens. Now, It is possible, as at a given time, one processor has a different id from every other processors id. Most likely there are few algorithms out here. One of the naive algorithm can be as followed to choose processor with different id:

```
Algorithm 1: Leader Election
  Input: Election function,current processor id
  Output: Elected leader's id
1 Start the election
2     elect(Election function, current processor id) to the left and right processor in the ring.
3      If there is two processors always send to the right one
4      When receive message(Election, current processor id) from both left
5      and right neighbors.In case of two, from the left one
6      if current processor id not equal to one of neighbours id or the left one(in case of 2)
   then
7  |        Current processor is the leader
8  |        elect(elected, current processor id) to my right neighbor
9 elect(elected, current processor id) is always forwarded to the right till it come back to the
   leader processor(it )
10 return leader id
```

2 Problem 3.5

This problem will be solved by the process of reduction. Lets named the algorithm with assumption as A and the algorithm without assumption as B. We can say algorithm A is reducible to algorithm B if for a given algorithm of A, we can construct algorithm for A or in mathematical notation it is A $\leq$ B.

We know that the minimum cost of the lower bound of Algorithm A is $\Omega(nlogn)$. In other words the best possible way to solve this problem is in nlogn time. As algorithm is reducible to algorithm B, the lower bound for algorithm B need to be $\Omega(nlogn)$. The reduction from algorithm A to algorithm B is given below:

Lets say we got a temporarily leader from sub routine A with message complexity of $o(nlogn)$ which is known. As for algorithm B, we still don't know who is the final leader,the temporarily leader will send it's id not the right of it in the ring. if happens that any of the next processor has greater id than the temporary leader, that processor will pass it's own id to the next one. In this way, the temporary processor will know if it is the leader or not. n number of messages need to be passed in this process. To make aware of the leader's id to all processors as it is possible that only few processors know leader's id, we need to pass the leader's id throughout the ring which will need another n number of messages. Hence, we need another 2n number of messages. In total the message complexity for B becomes, $o(nlogn)$+2n=$o(nlogn)$ (asymptotically) which proves that A $\leq$ B. This gives the reduction for us.

Now,We will assume that we can convert from algorithm A to algorithm B fast enough and we can go from algorithm A to algorithm B in fast enough time or vice versa.

Now lets assume, the run time for algorithm B is O(n) which is a better one. So , one way to solve this problem will be to formulating algorithm A to B, solve it using algorithm B and go back to algorithm A. Keeping the assumption that we can convert algorithm A to algorithm B fast enough this process would give us the O(n). However, it contradicts as we already know that the lower bound of algorithm A is $\Omega(nlogn)$. The problem with our assumption is the O(n) time which indicates that algorithm B at least would tak e nlogn time. hence, the lower bound for algorithm B is $\Omega(nlogn)$

3 Problem 3.7

In synchronous system,we say an execution is admissible if it is infinite whereas in asynchronous system, it is not the case. it only wakes up if it needs to send messages to forward or it is the contender of becoming a leader.

An execution is said to be synchronous when every process in the system observes the same total order of all the processes which happen within it. An non-synchronous execution,only processors that are still competing to be a leader or need to pass a messages are active .So, there is no guarantee for total order of all the processes.

Hence, a non-synchronous model need to have a certain number of execution and forwarding of messages to satisfy the admissible execution condition while total order of executions is not guaranteed (unless all the processors competing for becoming a leader) within a system to satisfy the condition of executions.

4 Problem 3.10

1. Lets assume by contradiction that there exists an uniform synchronous algorithm A that solve the problem. Also assume a ring with the number of nodes ≥ 3 and all input of 1. In any of the round, the states of all nodes are identical which we proved already one of the lemma in class. As we are saying this algorithm to be uniform and the ring does not depend on the number of nodes, state of nodes also does not depend on the number of nodes say n , of the ring. We also conclude from this that there exists some round number say t, which does not depend on n and because of this all nodes must terminate and output 1 in a particular round t. This will hold as we assumed the algorithm to be correct. Lets apply this algorithm A on a ring sized 2*t+2. Lets assume only one node named n_1 will have input bit of 0. From the next node till the t'th node, all other nodes have the input value of 1. According to the argument (In any of the round, the states of all nodes are identical),till t, node n_1 will have same state as if in a ring where all nodes have input 1. Thus, in round t, n_1 will terminate and output 1 which contradicts with the assumption of A being correct and should output 0 at all nodes.

2.Algorithm for computing AND: In this particular algorithm, all input values need to be send all around the ring. We will keep track of the number of nodes a message traveled. Say, if we have n nodes and we get the count of our tracking as n, we know that for sure that it have arrived to the position from where it has started which is the worst case. This calculation is made for a single node.

For n nodes the count will: n number of messages* number of node , n=n*n=n^2 of messages Hence, the worst case will O(n^2).

3.Lets say we have n number of nodes in a ring. For computing AND, all input values need to be send all around the rings. For a particular node , hence, we need to send a message to other n-1 nodes. There is no way we can send less than this. For n number of nodes we need to do it for n*(n-1) times. This the least we need to do. So, the message complexity turned into n*(n-1)=$n^2 - n$ or in terms of complexity $\Omega(n^2)$.It is the lower bound.

4.

Algorithm 2: `Synchronous algorithm for computing AND`

```
1 if input bit==0 then
2     pass 0 to adjacent nodes in the ring
3 foreach i:=2 to n do
4     if 0 received and has not sent then
5         pass 0 to adjacent nodes in the ring
6 if 0 received at least once then
7     result:=0
8 else
9     result:=1
```

Here, we will send only one message if the result is 0, otherwise not. Hence, the only n messages will be sent and in worst case it will be O(n) messages.

5 Acknowledgement

Me and Qiao Qiao discussed from a abstract level about 3.10 and 3.7 and 3.5.

I also went through the research paper name Computing on an Anonymous Ring by Hagit Attiya AND Marc Snir.