Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

WS 06/07

Prof. R. Wattenhofer, Prof. P. Widmayer
Thomas Locher, Yvonne Anne Oswald

Exam
Principles of Distributed Computing

Wednesday, March 5th, 2007

Do not open or turn until told so by the supervisor!

Notes

There is a total of 90 points. The number of points is given before each individual question in parentheses. The total for each group of questions is indicated after the title.

Your answers may be in English or in German. Algorithms can be specified in high-level pseudocode or as a verbal description, unless otherwise mentioned. You do not need to give every last detail, but the main aspects need to be there. Big-O notation is acceptable when giving algorithmic complexities. However, give algorithmic complexities as tight as possible.

Points

Please fill in your name and student ID before the exam starts.

Name	Legi-Nr.

Question Nr.	Achieved Points	Max Points
1		15
2		11
3		16
4		17
5		18
6		13
Total		90

1 Short Questions (15 Points)

a) (7) Describe, in sufficient detail, a distributed, asynchronous algorithm which computes the number of nodes in a graph G. Assume that the algorithm is started by a node s. What is the time complexity of your algorithm?

b) (5) Suppose that, for a graphs with maximum degree $\Delta = \log^2 n$, n the number of nodes, someone gives a synchronous leader election algorithm with running time $O(\log \log n)$. Even assuming unique identifiers (but **not** 1 through n), why is this not possible?

c) (3) Where is the problem if we apply the Arrow protocol to the ring?

2 Synchronizer (11 Points)

In the lecture, we have studied synchronous and asynchronous communication models. On the one hand, we have seen that arguing about synchronous algorithms is often easier than arguing about asynchronous algorithms. On the other hand, real-world networks are usually not perfectly synchronous systems. It would therefore be nice to be able to run a synchronous algorithm in an asynchronous system. In order to do so, we need an asynchronous algorithm (called a synchronizer) which simulates the rounds of a synchronous algorithm such that all nodes can send and receive the same messages in the same order as in a synchronous environment.

a) (4) Give a synchronizer which allows to convert a synchronous algorithm into an asynchronous one of the same time complexity. What can you say about the message complexity of the resulting algorithm?

b) (5) In some cases, saving messages is more important than saving time. Assume that we are given a synchronous algorithm with low message complexity even on dense graphs. In particular, we assume that we have a synchronous algorithm where every node sends only one message per round. Assume that you are given a (precomputed) rooted BFS tree of the network graph. Give a synchronizer which is asymptotically optimal with respect to message complexity for the given synchronous algorithm. What can you say about the time complexity of the resulting asynchronous algorithm?

c) (2) What is the definition of time complexity in the asynchronous model?

3 Leader Election (16 Points)

We are given a ring network with $n = n_r + n_b \geq 3$ nodes. All nodes of the ring are either colored red or blue such that n_r nodes are colored red and n_b nodes are colored blue. Except for the colors, the ring is anonymous (nodes of the same color can not be distinguished from each other). Assume that all nodes on the ring know n and have a sense of direction (i.e., nodes can distinguish their clockwise and counter-clockwise neighbors).

a) (9) For which combinations of n_r and n_b is it generally **not** possible to do leader election? Try to find a condition which is as general as possible.

b) (7) Is it always possible to do leader election if the condition that you found in (a) is not satisfied? If yes, argue why. If no, try to specify the cases where leader election is feasible as precisely as possible.

4 Sorting (17 Points)

We are given a two-dimensional $m \times m$-mesh on which we want to sort.

a) (4) Consider a sorting algorithm that alternately sorts all rows from left to right and all columns from top to bottom. Why doesn't this algorithm always sort, no matter how much time is allowed?

b) (7) Assume that wrap-around edges from the right end of the i^{th} row to the left end of the $(i+1)^{\text{st}}$ row ($i = 1, \ldots, m-1$) are added to the mesh. We are now modifying the algorithm of (a) by using these extra links. Every other row sorting step, we sort rows which are shifted by $m/2$. Hence, for $i = 1, \ldots, m-1$, the second half of the i^{th} row and the first half of the $(i+1)^{\text{st}}$ row are taken together and sorted. Show that the mesh is sorted after sufficiently many steps by this new algorithm.

c) (6) Can you give bounds on the number of row and column sort operations your algorithm needs to sort any given input?

5 PRAM Algorithms (18 Points)

You are given an n-element array A together with an n-element array L where $L[i]$ represents the label of element $A[i]$. The labels are in the range $L[i] \in \{1, 2, \ldots, k\}$, where k is a constant.

Consider the following task. We want to store all the elements of A with label 1 into the beginning part of A while preserving their initial ordering, followed by the elements labeled 2 with the same initial ordering, and so on. Essentially, the labels in L tell us how to sort the array A.

An example with $n = 5$ and $k = 3$: $A = [8, 1, 5, 9, 4]$, $L = [2, 1, 1, 3, 2]$, then after the execution A should be $A = [1, 5, 8, 4, 9]$.

a) (2) When is an algorithm in the EREW PRAM model "optimal"?

b) (2) How much time does an optimal sequential algorithm need for the above task?

c) (14) Develop an optimal $O(\log n)$ time EREW PRAM algorithm. You can use the algorithms from the lecture as a black box.

6 Distributed Deterministic MIS Construction (13 Points)

In the lecture, we have discussed a simple randomized algorithm to construct a maximal independent set (MIS) in expected $O(\log n)$ rounds, n the number of nodes. While an MIS can be computed very efficiently by a randomized distributed algorithm, finding a deterministic algorithm turns out to be a lot harder. The goal here is to develop a simple deterministic algorithm which computes an MIS in time $o(n)$. Here we will assume that the nodes have unique identifiers.

a) (3) Describe a simple distributed algorithm based on the nodes' identifiers which runs in time $O(n)$.

There exists a distributed and deterministic algorithm ALG[1] which computes an MIS in time $O(\Delta \log n)$, where Δ is the maximum degree of the input graph. Since Δ can be in the order of n, this does not give us the desired $o(n)$ time MIS algorithm. Therefore, we would like to reduce the degree of the graph first, before applying ALG.

b) (6) Assume that a node v is removed from the graph once v or a neighbor of v is selected into the MIS. Modify your algorithm from (a) so that it tries to quickly produce a graph of low degree. How long does it take until no node of degree at least δ remains?

c) (4) How can you combine ALG with your algorithm in (b) to obtain an $o(n)$ time distributed deterministic MIS construction algorithm? Give an expression (in Big-O notation) for the running time of your algorithm.

[1]not discussed in the lecture but otherwise well-known