Deterministic Distributed Algorithms, 10–21 May 2010, Exercises
http://www.cs.helsinki.fi/jukka.suomela/dda-2010/
Jukka Suomela, last updated on May 20, 2010

Instructions

Optional exercises are merely suggestions for further self-study: you can try to solve them if you have time, and we will discuss them during the exercise sessions if there are no further questions on compulsory exercises. You do not need to write up answers to optional exercises in your final report.

However, you must answer all *compulsory* exercises. See the course web page for more instructions. Some hints are given as endnotes.

Notation

In some exercises, we will need power towers. We will use the following notation:

$$t(0,x) = 1,\ t(1,x) = x,\ t(2,x) = 2^x,\ t(3,x) = 2^{2^x},\ t(4,x) = 2^{2^{2^x}},\ \ldots$$

In particular,

$$t(i,2) = \underbrace{2^{2^{\cdot^{\cdot^{2}}}}}_{i \text{ times}}.$$

Exercise 1

Compulsory

As the name of the course suggests, the focus is on deterministic distributed algorithms, as opposed to randomised distributed algorithms. In a deterministic algorithm, nodes do not have access to any source of randomness; all state transitions are deterministic, and the output of an algorithm is uniquely determined by the problem instance (communication graph, port numbering, and local inputs).

What would change if we changed the model and provided each node with access to a stream of random bits? Which problems become solvable? (1) Of course, if we use randomness, then the algorithm may perform poorly if the random bits are "unfortunate"; in your answers, make it explicit what might go wrong and how likely it is. (2)

This is a question that you should ponder throughout the course. Whenever we mention a negative result, ask yourself whether this holds even in the case of randomised algorithms. Whenever we present a distributed algorithm, ask yourself if we could make the algorithm simpler or faster with the help of randomness.

By the way, are there any reasons for focusing on deterministic distributed algorithms instead of randomised distributed algorithms? (3)

Optional

In the above exercise, we have seen that randomised distributed algorithms are strictly more general than deterministic distributed algorithms. And during the lectures, we have seen that the model with unique identifiers is strictly more general than the port-numbering model. In a sense, these are the only essential limitations in the models that we use throughout this course. We have made other assumptions, but they are merely a convenience and could be removed without affecting computability and time complexity of the algorithms.

The use of synchronous communication rounds is a good example of the non-essential assumptions. Show that synchronous distributed algorithms can be efficiently simulated in asynchronous networks. (4) Show that negative results for synchronous algorithms imply negative results for asynchronous algorithms as well. (5)

Exercise 2

Compulsory

Let us first study some properties of iterated logarithms.

(a) What is $\log^* 10^{10^{10}}$?

(b) Let x be a positive integer, and define $i = \log^* x$ and $y = t(i, 2)$; see above for the definition of t. What is y as a function of x? (6)

Now consider the adaptation of the Cole–Vishkin algorithm that finds a 3-colouring in a cycle or tree. Analyse the running time of the algorithm in the following cases. Provide the exact answer (exactly how many communication rounds). (7)

(c) Nodes have 48-bit unique identifiers (e.g., MAC addresses).

(d) Nodes have 128-bit unique identifiers (e.g., IPv6 addresses).

Next, analyse the running time of the same algorithm in the following cases. Provide an upper bound (as a function of n). (8)

(e) Nodes have unique identifiers from the set $\{1, 2, \ldots, n^2\}$.

(f) Nodes have unique identifiers from the set $\{1, 2, \ldots, n^c\}$ for some constant c.

(g) Nodes have unique identifiers from the set $\{1, 2, \ldots, 10^n\}$.

Optional

Design an algorithm that finds an *edge colouring* of the communication graph, assuming that the nodes have unique identifiers. (9) Analyse the running time of your algorithms. Don't hide constants in O-notation; derive a concrete upper bound.

Exercise 3

Compulsory

In this exercise, we will derive upper and lower bounds for Ramsey numbers.

(a) Prove that $R_2(3;2) = 6$. [(10)]

(b) Derive upper and lower bounds on $R_2(4;2)$. [(11)]

(c) Derive an upper bound on $R_2(n;k)$, as a function of n and k. [(12)]

Optional

In the above exercise, show that there is a constant A such that $R_2(n;k) \leq t(k, An)$ for large values of k. [(13)]

Exercise 4

Compulsory

In this exercise, we will derive more positive and negative results related to graph colouring and colour reduction. Most of these problems have very simple proofs; Ramsey's theorem and the techniques of lecture 4 are needed only in problem (k). In problem (b), you should give the algorithm in detail (exactly which messages are sent in each round); for all other positive results, it is enough to just sketch the basic idea of your algorithm.

First we will study algorithms for *path graphs* in the *port-numbering model*:

(a) Show that it is not possible to find a 2-colouring in this setting. [(14)]

(b) Show that it is possible to reduce the number of colours from 3 to 2 in $O(n)$ rounds. [(15)]

(c) Show that it is not possible to reduce the number of colours from 3 to 2 in $n/3$ rounds. [(16)]

Next, we will study algorithms for *directed paths* in the *port-numbering model* (edges are directed so that each node has outdegree ≤ 1 and indegree ≤ 1; the local input of each node indicates which ports are outgoing edges and which ports are incoming edges):

(d) Show that it is possible to find a 2-colouring in $O(n)$ rounds.

(e) Show that it is not possible to find a 2-colouring in $n/3$ rounds.

Let us then study algorithms for *directed cycles* in the *port-numbering model* (edges are directed so that each node has outdegree = 1 and indegree = 1; the local input of each node indicates which ports are outgoing edges and which ports are incoming edges):

(f) Show that it is not possible to find a 3-colouring.

(g) Show that it is not possible to reduce the number of colours from 3 to 2 in any odd cycle.

(h) Show that it is not possible to reduce the number of colours from 3 to 2 in some even cycles. (17)

(i) Is it possible to reduce the number of colours from 3 to 2 if the number of nodes is exactly 4? If yes, how many communication rounds are sufficient?

(j) Show that it is possible to reduce the number of colours from 4 to 3 in 1 communication round.

(k) Show that there exists a value k such that it is not possible to reduce the number of colours from k to 3 in 1 communication round. (18)

Optional

Consider the setting of problems (j) and (k) above. Is it possible to reduce the number of colours from 5 to 3 in 1 communication round? (19)

Exercise 5

Compulsory

In the lecture, we saw how to find a weak colouring in graphs with indegree $\neq$ outdegree. In this exercise, we will consider *directed cycles*. Such cycles have indegree = outdegree = 1, and we cannot apply the Naor–Stockmeyer algorithm. The input may be symmetric, and we need some auxiliary information.

Assume that you are given a proper k-colouring of the directed cycle as input (here k is some constant). Design an algorithm that finds a weak 2-colouring. (20) What is the running time of your algorithm? (21)

Optional

Define a *weak edge colouring* in an analogous manner: each non-isolated edge has at least one adjacent edge with a different colour. Study the solvability of this problem in the port-numbering model, with and without orientations. When can you solve this problem and how fast? Does the solvability depend on outdegrees and indegrees? (22)

Exercise 6–7

Compulsory

Either give anonymous feedback at

`https://ilmo.cs.helsinki.fi/kurssit/servlet/Valinta?kieli=en`

or explain in your report why you decided not to give any course feedback.

Optional

Contact the course instructor if you would like to write a Master's thesis on distributed algorithms, or if you are interested in doing your PhD studies in this area.

Hints

(1) Can you use randomness to break symmetry in a cycle without using unique identifiers? Can you find a large independent set in a cycle? A maximal independent set in a cycle? A graph colouring in a cycle? What about general graphs? Can you construct unique identifiers by using random bits? What kind of auxiliary information can be replaced by using randomness and what cannot?

(2) For example, a "randomised α-approximation algorithm for the vertex cover problem with running time T" might fall in any of the following categories: (i) The algorithm outputs a valid vertex cover when it terminates, and the approximation factor is at most α; with high probability, the algorithm terminates in time T. (ii) The algorithm outputs a valid vertex cover when it terminates; with high probability, the approximation factor is at most α; the algorithm terminates in time T. (iii) With high probability, the algorithm outputs a valid vertex cover; the approximation factor is at most α; the algorithm terminates in time T. Combinations of these are also possible, and instead of "with high probability", some guarantees may hold only "in expectation".

(3) Fault-tolerance. Self-stabilising algorithms.

(4) Use the α-synchroniser [1].

(5) In the worst case, an asynchronous network may accidentally operate in a completely synchronous manner.

(6) It is not necessarily x. Why? Derive as tight upper and lower bounds as possible.

(7) Note that analysing the number of "iterations" is not enough. You need to make sure that each iteration can be completed in one communication round (if this is the case).

(8) Whenever possible, try to derive an upper bound of the form $C+A \log^* n$, where A and C are some constants. Try to make the value of A as small as possible.

(9) Try to find an edge colouring with $2\Delta - 1$ colours. One possibility is to first find a vertex colouring with $\Delta + 1$ colours, and then use the vertex colouring to derive an edge colouring. However, other approaches may lead to faster running times [3].

(10) Naturally, it is enough to show that $R_2(3;2) > 5$, i.e., there is a 2-colouring of K_5 that does not contain a monochromatic triangle, and $R_2(3;2) \leq 6$, i.e., any 2-colouring of K_6 contains a monochromatic triangle.

(11) For example, you could try to show that $10 \leq R_2(4;2) \leq 100$, for some suitable values of 10 and 100. Your lower-bound construction in problem (a) gives a lower bound on $R_2(4;2)$ as well. For the upper bound, you can follow

the proof in the lecture slides: step (iii) shows that $R_2(4;2) \leq G_2(7;2)$, then you can use step (ii) to derive a bound on $G_2(7;2)$, etc. Eventually, this line of reasoning will converge to an upper bound.

(12) It is enough to have a recursive definition of your upper bound function; for example, you could show that $R_2(n;k) \leq f(n,k)$, where $f(n,k)$ is defined in terms of $f(n-1,k)$ and/or $f(n,k-1)$. The proof by induction in lecture slides already gives a recursive definition of the upper bound (using functions R_2 and G_2); you just need to put everything together and fill in the missing details. For example, in the lecture slides I just claim that $G_c(k;k)$ is finite for all c and k; you will need to derive a concrete upper bound (this is straightforward if you consider the definition of $G_c(k;k)$). Simplify your upper bound as much as you can.

(13) This is not easy if you try to follow the proof in the lectures. You can have a look at an alternative proof, e.g., in Graham et al. [2, p. 7–9].

(14) Consider an odd path, i.e., a path with an odd number of edges.

(15) Consider two different cases: odd paths and even paths.

(16) Construct an example in which a two nodes have to make different decisions, but their local neighbourhoods are identical. For example, you can consider a path with colours $0, 1, 2, 0, 1, 2, \ldots$.

(17) Consider, for example, a 6-cycle. Covering maps and covering graphs can be used as a proof technique.

(18) You do not need to derive the numerical value of k; it is OK to use, e.g., Ramsey numbers as the value of k. You can solve this problem in two steps: (i) Show that if n is sufficiently large, an algorithm with running time 1 cannot find a 3-colouring in some numbered directed n-cycle. You can follow the same idea as in the lecture slides; this time we have the simple special case $T = 1$. However, the algorithm has 3 possible outputs (3 colours) instead of 2 possible outputs (part of independent set or not). Note that you can simplify the proof a lot: you can stop as soon as you have established that there must exist a chain of at least 2 nodes such that both nodes produce the same output – this already violates the assumption that the algorithm produces a valid graph colouring. Following the full proof in the lectures would actually show that in the worst case, the colouring will be bad almost everywhere! (ii) Then you can set $k = n$ and observe that a numbered directed n-cycle can be seen as a special case of a directed cycle that is coloured with n colours. Reason that if you cannot find a 3-colouring even in the case where colours are globally unique, certainly you cannot find a 3-colouring in the general case.

(19) This question is related to the existence of a 3-colouring in a certain graph with 80 nodes.

(20) You can use the techniques from lecture 2 to first reduce the number

of colours from k to 3. Hence you only need to show how to find a weak 2-colouring if you are given a proper (non-weak) 3-colouring. You do not need any techniques from lecture 5 – just keep in mind what is the definition of a weak colouring.

(21) You can use O-notation here.

(22) You may want to consider the vertex cover algorithm that was presented in lecture 1. In the algorithm, we used a subroutine that finds a set D that consists of paths and cycles.

References

[1] Baruch Awerbuch. Complexity of network synchronization. *Journal of the ACM*, 32(4):804–823, 1985.

[2] Ronald L. Graham, Bruce L. Rothschild, and Joel H. Spencer. *Ramsey Theory*. John Wiley & Sons, New York, NY, USA, 1980.

[3] Alessandro Panconesi and Romeo Rizzi. Some simple distributed algorithms for sparse networks. *Distributed Computing*, 14(2):97–100, 2001.