ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

SS 2007

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

Exam
Principles of Distributed Computing

Thursday, August 23rd, 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		23
2		25
3		18
4		12
5		12
Total		90

1 Butterfly Networks (23 Points)

Let $d \in \mathbb{N}$. The d-dimensional butterfly $BF(d)$ is a graph with node set $V = [d+1] \times [2]^d$ and edge set $E = E_1 \cup E_2$ with

- $E_1 = \{\{(i,\alpha),(i+1,\alpha)\} \mid i \in [d], \alpha \in [2]^d\}$
- $E_2 = \{\{(i,\alpha),(i+1,\beta)\} \mid \alpha,\beta \in [2]^d$, α and β differ only at the i^{th} position$\}$.

The three-dimensional butterfly $BF(3)$ is depicted in Figure 1.

a) (3) In a butterfly $BF(d)$, what is the maximum number of shortest paths between any vertex on layer 0 and any vertex on layer d (so-called end-to-end-paths)? Explain!

b) (3) If the $d+1$ layers are merged together, we get a d dimensional hypercube (consisting of $n = 2^d$ vertices). What is the maximum number of shortest paths between any pair of vertices in the hypercube? Explain!

c) (4) In a butterfly $BF(d)$, describe an instance of a permutation routing problem for which many shortest paths (as many as you can get) pass through the same edge. How many paths pass through this edge? Recall that a permutation routing problem instance in a butterfly requests for each vertex on layer 0 a path to a specific vertex on layer d, so that no vertex on layer d appears more than once as a destination.

The packing problem in a butterfly $BF(d)$ takes a set of packets at a subsequence of $s \leq 2^d$ vertices on layer 0 (with one packet per vertex) and routes them to the first s vertices on layer d in such a way that the relative order of the packets is preserved. Intuitively, this "packs" packets on vertices with "spaces" in between on layer 0 into consecutive vertices on layer d (see Figure 1).

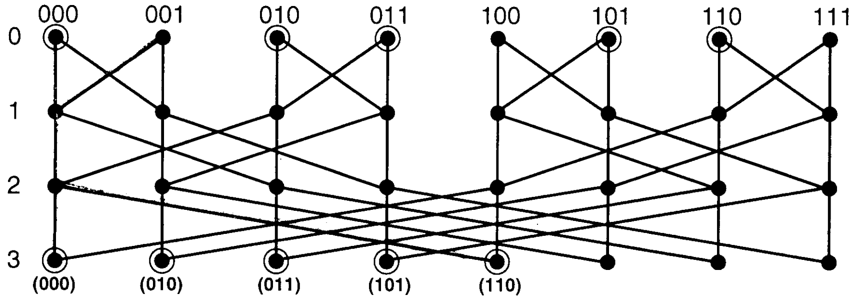

Figure 1: The subsequence in this example consists of the (circled) vertices 000, 010, 011, 101, and 110. Packets from these vertices are routed to the first 5 vertices on layer d, preserving the relative order.

d) (7) Initially, a packet at a vertex i on layer 0 does not know its destination vertex on layer d. Propose an efficient distributed algorithm that lets each packet on layer 0 find its destination vertex number, and show how many steps this takes in total.

e) (6) Assume now that each packet knows its destination. Greedy routing sends each packet on a shortest path (end-to-end). Prove that all greedy routes are vertex disjoint.
Hint: Prove first that no two such paths enter a vertex on layer 1, and then use induction over the layers.

2 Problems in Complete Graphs (25 Points)

We are given the complete graph $\mathcal{K}_n$ consisting of n nodes with undirected edges between each pair of nodes. Each node v has a unique identifier id_v and each edge e has a positive weight $w(e)$. You can assume that no two edge weights are equal. Each node further knows the weight of all incident edges and the identifiers of the nodes at the other end of the edges. Thus, every node knows all other nodes.

We use the synchronous model of communication where in each round each node can send (potentially different) messages to all its neighbors, receive messages, and perform some local computation. The size of any message is restricted in that only a constant number of node identifiers and edge weights, and additionally a constant number of other numbers of the same magnitude as identifiers and weights, can be sent in a single round.

a) (3) A specific node v wants every other node to know all its $n-1$ edge weights. Give an algorithm that achieves this goal as fast as possible, in particular requiring much less than n rounds!

In the lecture, we discussed algorithms to compute the minimum spanning tree (MST) in this model. Now we are interested in finding the n lightest edges overall, i.e., after the algorithm terminates, every node knows the weights of the n lightest among all $\binom{n}{2}$ edges and which nodes these edges connect.

Consider the following simple algorithm: *Every node sends its i^{th} lightest edge to all other nodes in round i. After a sufficiently large number of rounds, the algorithm terminates and each node knows that the n smallest weights it has learnt belong to the n lightest edges overall.*

b) (8) Show an example (that means, an assignment of edge weights to the nodes) where the above algorithm is as slow as possible!
Hint: The problem with this simple algorithm is that nodes potentially send edge weights that have already been broadcast (by other nodes) before.

In order to overcome the problem mentioned above, we modify the algorithm in the following way: *In each round, broadcast the lightest incident edge weight that has not already been broadcast before.*

c) (10) Prove an upper bound on the number of rounds required when the modified algorithm is used! Moreover, prove that your bound is asymptotically tight by providing a worst-case example (of the same asymptotic time complexity)!

Now, we are going to derive a *randomized* algorithm whose *expected* time complexity is only $O(\log n)$. Use the fact that a single node can determine the n^{th} smallest among all $\binom{n}{2}$ edge weights in $O(\log n)$ rounds in expectation.[1]

d) (4) Given that node v knows the n^{th} smallest edge weight (after $O(\log n)$ rounds), how can all nodes (v and all other nodes) learn all the weights of the n lightest edges? Describe an algorithm that solves this problem! The total time complexity must not exceed $O(\log n)$ rounds!

[1] Note that this algorithm cannot be *parallelized*! This means that this subroutine cannot be used to compute all n lightest edges *in parallel* in $O(\log n)$ time.

3 MIS on Planar Graphs (18 Points)

In the lecture, we showed that a maximum independent set (MIS) on a general graph $G = (V, E)$ can be computed using a randomized algorithm in $O(\log n)$ synchronized rounds in expectation. As a reminder, a single round of the algorithm consists of the following three steps:

1. Node v marks itself with probability $\frac{1}{2d(v)}$, where $d(v)$ is the current degree of v.
2. If no higher degree neighbor of v is also marked, node v joins the MIS. If a higher degree neighbor of v is marked, node v unmarks itself again. If the neighbors have the same degree, ties are broken arbitrarily, e.g. by identifier.
3. Delete all nodes that joined the MIS and their neighbors (that cannot join the MIS anymore).

It is now your task to prove that this algorithm constructs a MIS on a *planar* graph also in $O(\log n)$ rounds in expectation. A planar graph is a graph that can be drawn so that no edges intersect.

Note that this immediately follows from the theorem proven in the lecture, but the proof for planar graphs is substantially easier, because a planar graph can have at most $3n - 6$ edges.

a) (4) Prove that at least $n/7$ nodes in a planar graph have degree at most 6.

b) (5) Analogous to the proof in the lecture, prove that a node with degree at most 6 joins the MIS in Step 2 with probability at least $1/24$.

c) (4) Using both a) and b), prove that the algorithm terminates after $O(\log n)$ rounds in expectation.

As all planar graphs have a constant fraction of nodes whose degree is at most 6, it is also easy to compute a MIS *deterministically* using another algorithm presented in the lecture as a subroutine.

d) (5) Describe a deterministic algorithm that computes a MIS on planar graphs in $O(\log n \log^* n)$ synchronous rounds![2]

[2] $log^* n$ is the number of times the logarithm function must be iteratively applied before the result is less than or equal to 1.

4 Network Flows (12 Points)

a) (3) State the maxflow-mincut theorem. Explain the relevant terms (*maxflow* and *mincut*).

b) (4) Prove or disprove the following assertion: *A mincut remains a mincut if we uniformly increase the capacity of every edge in the network by* 1.[3]

Consider the following method to find a maximum cardinality matching (the maximum number of disjoint edges) in a general (non-bipartite) graph $G = (V, E)$ using network flows.

1. Construct a bipartite graph H, with vertex set $E \cup V$. (That is, each edge and vertex of G corresponds to a node in our bipartite graph.)
2. Add an artificial source node s and an artificial sink t.
3. Join s to each node of E, with capacity 2.
4. Join each node of V to t, with capacity 1.
5. For each edge $e = (u, v)$, put edges of capacity 1 between e and u, and e and v.
6. The cardinality of the maximum cardinality matching in G is half of the maxflow from s to t in this network H.

The intuition behind this algorithm is that each edge in the matching covers 2 vertices, so by sending a flow of 2 units to each edge, and placing a capacity of 1 at each node, we can model the matching problem as a flow problem.

c) (5) Either prove that this algorithm is always correct, or give a counterexample.

[3] In other words, suppose (A, B) is a mincut in the original network. Then, (A, B) is still a mincut (although with a different capacity value) in the modified network.

5 Network Failure (12 Points)

We are given an undirected graph G with vertex set V and edge set E. An (ϵ, k)-detection set is a set of vertices with the property that if (adversarial) deletion of up to k edges breaks the graph into two components, each containing at least an ϵ fraction of the node set V, then at least two nodes of the detection set are also disconnected. If any two nodes in the detection set fail to communicate, we declare a cut.

a) (3) Suppose the graph is a path graph (a linear chain of nodes). What is the optimal (smallest) size of a (ϵ, k)-detection set?

b) (3) Is it true that if each node in G has degree at least $k+1$, then the adversary cannot disconnect the network (by deleting at most k edges)? Justify your answer.

c) (6) Assume now that you are given an arbitrary tree T consisting of n nodes, and a detection set (a subset of the nodes) in this graph. You can further assume that the number of nodes in the detection set is even.

The goal is to match up each detector with a partner detector. Describe an asynchronous distributed algorithm, starting at the leaves of the tree, that finds for each detector a partner in such a way that all the paths connecting two partner detectors are pairwise *edge-disjoint.*