CS466 — Fall 1999

Problem Set 3

These problems are due Thursday, November 18, in class. Do your own work.

1. [10 marks]
   CLR problem 30-4, p. 728.

2. [15 marks]
   For this problem you will design a parallel algorithm to compute minimum spanning trees. The input will be an adjacency matrix G, with $G[u, v]$ holding the weight of the edge between u and v. For simplicity, assume that

   - The nodes of the graph are the integers from 0 to $n - 1$.
   - All edge weights are distinct. As a consequence, the minimum spanning tree is unique.

   Neither assumption is actually necessary, but they eliminate some complications in the algorithm.

   For a node u, let $m(u)$ be the neighbor of u such that $G[u, m(u)]$ has minimum weight over all edges incident to u.

   (a) Show that each edge $(u, m(u))$ belongs to the minimum-weight spanning tree.

   (b) Show that in the directed graph with directed edges $(u, m(u))$, each component has one cycle of length two and no other cycles.

   (c) Using the above results, derive and analyze an efficient PRAM algorithm to find minimum spanning trees. Time efficiency is most important, but you should keep the processor count low if you can. You do not have to show optimality of your algorithm.

   (d) Does your algorithm require concurrent writes? Concurrent reads?

3. [15 marks]
Consider a network of $n = 2^d$ processors (d even) arranged in a hypercube; that is, processors p and q are connected if and only if $\text{bin}(p)$ and $\text{bin}(q)$ differ by one bit. Packets in the network are routed by straightforward bit-switching—a packet with source p and destination q is routed along the path corresponding to changing bits in the processor numbers. In other words, if $\text{bin}(p) = a_1a_2a_3\ldots a_i$ and $\text{bin}(q) = b_1b_2b_3\ldots b_i$, then the route of the packet is

$$a_1a_2a_3\ldots a_i \to b_1a_2a_3\ldots a_i \to b_1b_2a_3\ldots a_i \to \ldots \to b_1\ldots b_ja_{j+1}\ldots a_i \to \ldots \to b_1b_2b_3\ldots b_i.$$

(A node may appear more than once in the route, but that does not affect the results in a meaningful way.)

(a) Suppose that the network stores a $\sqrt{n} \times \sqrt{n}$ matrix, with entry k, l at processor number $k\sqrt{n} + l$. They want to compute the transpose of the matrix; that is, to send the value at processor number $k\sqrt{n} + l$ to processor number $l\sqrt{n} + k$ for all k and l between 0 and $\sqrt{n} - 1$.

Show that applying the above bit-switching method to the transpose permutation routes $\Omega(\sqrt{n})$ packets through some nodes in the network. Thus routing all of the packets requires time $\Omega(\sqrt{n})$ in the case that a node can send or receive only a constant number of packets at one time.

(One can show that any routing method for any degree-k graph on n nodes has a worst case of at least $\sqrt{n}/(2k)$. But you don't have to do the general case.)

(b) Suppose that each processor in the network chooses uniformly and independently a destination processor for its packet, and these packets are routed simultaneously. Show that the probability of having r or more packets routed through any particular node of the network is at most $(c/r)^r$ for some constant c.

(c) Using the result of part (b) (whether you solved it or not), describe a randomized routing method that for any given permutation routes $o(\sqrt{n})$ packets through every node with high probability. (Full marks require good bounds; part marks for weaker results.)