Massachusetts Institute of Technology
6.892: Theory of Parallel Systems
Professor Charles E. Leiserson

Handout 3
February 4, 1997

Problem Set 1

This problem set is due in class on *Thursday, February 20.*

Problem 1-1. Segmented prefix

Like an ordinary prefix computation, a **segmented prefix computation** is defined in terms of a binary associative operator $\oplus$. It takes an input sequence $x = \langle x_1, x_2, \ldots, x_n \rangle$ whose elements are drawn from a domain S, and a **segment** sequence $b = \langle b_1, b_2, \ldots, b_n \rangle$ whose elements are drawn from the domain $\{0, 1\}$. It produces an output sequence $y = \langle y_1, y_2, \ldots, y_n \rangle$ over the domain S. The bits of b determine a partitioning of x and y into segments; a new segment begins wherever $b_i = 1$, and the current one continues if $b_i = 0$. The segmented prefix computation performs an independent prefix computation within each segment of x to produce the corresponding segment of y.

Here is an example that illustrates segmented prefix for the case where the associative operator is ordinary addition:

$$\begin{array}{ccrrrrrrrrrrrr}
b & = & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
x & = & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
y & = & 1 & 3 & 6 & 4 & 9 & 6 & 7 & 15 & 24 & 34 & 45 & 57
\end{array}$$

Show how to implement any segmented prefix computation on an n-element list in $O(\lg n)$ time on a binary tree.

Problem 1-2. Brent's theorem

Brent's theorem states that for any dag with work T_1 and critical-path length T_∞, a greedy scheduler can execute the dag on P processors in time at most

$$T_P \leq \frac{T_1}{P} + T_\infty \ .$$

Prove the tighter bound

$$T_P \leq \frac{T_1 - T_\infty}{P} + T_\infty \ .$$

Problem 1-3. Routing on butterflies

A **packing** problem consists of routing any collection of $N \leq M$ packets contained (one per processor) in level $\lg N$ of an N-input butterfly into the first M processors in level 0 of the butterfly so that the relative order of the packets is unchanged. For example, the routing problem $1 \to 0$, $3 \to 1$, $4 \to 2$, $7 \to 3$ is a packing problem (we mean that a packet goes from processor 1 in level $\lg N$ to processor 0 in level 0, and so on.)

(a) Give an algorithm for packing that requires at most $O(\lg N)$ steps on an N-input butterfly. (*Hint:* Before routing, do a prefix operation.)

A **monotone** routing problem is one for which the relative order of packets is unchanged. For example, the routing problem $1 \to 2$, $4 \to 3$, $5 \to 4$, $7 \to 6$ is monotone. The routing problem $2 \to 6$, $3 \to 5$, $6 \to 7$ is *not* monotone, because the relative order of the packets originating at nodes 2 and 3 is reversed by the routing.

(b) Give an $O(\lg N)$-time algorithm for monotone routing on an N-input butterfly.

(c) Give an $O(\lg^2 N)$-time algorithm for routing an arbitrary permutation on an N-input butterfly.

Problem 1-4. Sorting on a matrix

Do Exercise 28.5-6 on page 650 of CLR.