Problems with a star are more difficult than the others

Problem Set 1 (Due Sept 27, 2000)

Problem 1.

Under which circumstances is speed more important than efficiency when designing a parallel algorithm? (Give an example to highlight your answer).

Problem 2.

Consider two algorithms for solving a problem of size N, one that runs in N steps on an N-processor machine and one that runs in $\sqrt{N}$ steps on an N^2 processor machine. Which algorithm is more efficient?

Problem 3.

Give an CREW algorithm for solving the problem of multiplying an $n \times n$ matrix A and vector x in $O(\lg n)$ time. How many processors does your algorithm require? How much work does it require? Comparing your algorithm to its sequential counterpart, what is its efficiency and speedup? (Use $O(.)$ notation to describe various results).

Problem 4.

Make the algorithm in Problem 3 to work on an EREW PRAM. What is its running time?

Problem 5.

You are given n binary values $X_1, \ldots, X_n$. Find the logical OR of these n values in constant time on a CRCW PRAM with n processors.

Problem 6.

For the algorithm in Problem 3/4, let us assume that we have only p processors available where $p < n$ and n is a multiple of p. How fast can we solve the matrix-vector multiplication problem on an EREW PRAM? Explain. Express parallel time, speedup and efficiency in terms of n and p.

Problem 7.

Show how the comparison of two n-bit numbers can be computed by a parallel prefix computation.

Problem 8*.

Show that the bisection width of an $\sqrt{n} \times \sqrt{n}$ mesh is at least $\sqrt{n}$.

Problem 9*.

Design an algorithm for sorting n numbers on an $O(\log n)$-processor complete binary tree that has $\Theta(1)$ efficiency. (You may assume that each processor can process and store $O(n/\log n)$ numbers, and you can take advantage of the fact that a single sequential processor can sort N numbers in $O(N \log N)$ steps. You may also allow I/O at each processor of the network).

Problem 10*.

(a) A sequence S of n keys $x_1, \ldots, x_n$ is given. For a given input key y, the *rank* of y with respect to S is the number of keys in S whose key-values are less than or equal to the key value of y. Given $y, x_1, \ldots, x_n$ describe an $O(\lg n)$ time algorithm that ranks y in S using n processors of a EREW PRAM.

(b) Given a sequence of n keys $x_1, \ldots, x_n$ sort these keys on a n^2 processor EREW PRAM in $O(\lg n)$ time. (*Hint:* Use Part (a) to rank each input key in the input key sequence).