BU CLA CS 530: Analysis of Algorithms

Spring 1999

Problem Set 8

Due Thursday March 18, by 5:00 pm, in my mailbox in the main office of the CS department. Late submissions not accepted.

---

1. Exercise 30.2-3, page 709, in [CLR]. (*Hint*: Although you are asked to design an n-processor CRCW algorithm, you will get full credit for a constant time n^2-processor CRCW algorithm.)

2. Let A and B be two arbitrary $n \times n$ matrices (of real numbers or boolean values).

   (a) Design a CREW algorithm to multiply A and B in $\mathcal{O}(\log n)$ time using n^3 processors. (*Hint*: Make minor modifications in the standard sequential algorithm to multiply two matrices, given at the bottom of page 554 in [CLR]. Your CREW algorithm should work in two phases, the first of which computes $a_{ij} \cdot b_{jk}$ for every $i, j, k \in \{1, \ldots, n\}$ in constant time.)
   (Exercise 30.2-5 in [CLR], page 709, asks for an $\mathcal{O}(\log n)$-time EREW algorithm for matrix multiplication, which is a bit more complicated to design.)

   (b) Describe a CREW algorithm to compute the powers A^1, A^2, A^3, $\ldots$, A^n, in $\mathcal{O}(\log^2 n)$ time using $\mathcal{O}(n^4)$ processors. (*Hint*: Multiplication of square matrices is an associative operation.)

3. Let $G = (V, E)$ be a directed graph, with a non-negative weight function. Describe a CREW algorithm to compute all-pairs shortest paths in G in $\mathcal{O}(\log^2 n)$ time using $\mathcal{O}(n^4)$ processors. Assume G is represented by an adjacency-weight matrix $W = (w_{ij})$. (*Hint*: Consult Section 26.1 in [CLR], and solve Problem 2 first.) Can you reduce the number of processors to $\mathcal{O}(n^3)$?

4. Design a CREW algorithm for finding the preorder numbering of a binary tree with $n \geqslant 1$ nodes, in $\mathcal{O}(\log n)$ parallel time using n processors. The algorithm should label each node of the tree with the number of nodes visited before it in a preorder traversal. Use the method of *pointer jumping*. For simplicity, assume the binary tree is *full*, i.e. every node which is not a leaf has exactly 2 children.
   (Exercise 30.1-6, page 701 in [CLR], asks the same question but does not restrict the number of processors to be n. Without this restriction, a solution based on the *Euler-tour technique* is the simplest.)

Created: 99.03.04