

CS 1510 Midterm 2  
Fall 2008

1. (20 points) Consider the following two problems:

MATRIX MULTIPLICATION

INPUT:  $n$  by  $n$  matrices  $A$  and  $B$

OUTPUT: The product  $AB$

MATRIX SQUARING

INPUT:  $m$  by  $m$  matrix  $C$

OUTPUT: The square of the matrix  $C$

Assume that Matrix Multiplication is a well known problem, and that none of the famous computer scientists that worked on the problem were able to find an  $O(n^2)$  time algorithm for this problem. Explain/Prove, using the concept of a reduction, that finding an  $O(m^2)$  time algorithm for Matrix Squaring is just as hard as finding an  $O(n^2)$  time algorithm for Matrix Multiplication.

Make sure to give some explanation of your approach. The purpose of this problem is mostly to test whether you understand what a reduction is, and how one can use a reduction to show the relative hardness of problems.

2. (20 points)

- (a) (7 points) Will an algorithm designed for a CRCW-Arbitrary machine necessarily run correctly on a CRCW-Common machine? Will an algorithm designed for a CRCW-Common machine necessarily run correctly on a CRCW-Arbitrary machine? Start with definitions of “CRCW-Arbitrary machine” and “CRCW-Common machine”. Justify your answers.
- (b) (7 points) Give a parallel algorithm for finding the minimum of  $n$  numbers on a CRCW-Common machine that runs in time  $O(1)$  with  $n^2$  processors.
- (c) (6 points) Is this algorithm sufficiently reasonably efficient that one might reasonably consider using it in practice? Start with a definition of efficiency, then calculate the efficiency of this algorithm, and then interpret what the result of this calculation means. Don't concern yourself with any other properties of the algorithm other than efficiency.

3. (20 points) Show that if one of the following three problems has a polynomial time algorithm then they all do.

- The Independent Set Problem: The input is a graph  $G$  and an integer  $k$ . The problem is to determine if  $G$  contains an independent set of size  $k$ . An independent set is a collection of mutually nonadjacent vertices.
- The Clique Problem: The input is a graph  $G$  and an integer  $k$ . The problem is to determine if  $G$  contains a clique of size  $k$ . A clique is a collection of mutually adjacent vertices.
- The Vertex Cover Problem: The input is a graph  $G$  and an integer  $k$ . The problem is to determine if  $G$  contains a vertex cover of size  $k$ . A vertex cover is a collection  $S$  of vertices with the property that every edge is incident on at least one vertex in  $S$ .

4. (20 points)

- (a) Consider the following problem: The input is an  $n$  vertex acyclic graph with weights on the edges. The output is  $n$  by  $n$  matrix giving the length of the shortest path between all pairs of vertices. Give an  $O(\log^2 n)$  time EREW algorithm to solve this problem using  $n^3$  processors.
- (b) Consider the problem of computing the length of the longest common subsequence of two sequences. Give an  $O(\log^2 n)$  time EREW algorithm for this problem that uses a polynomial number of processors.

You can use your solution to the first part as a black box. You can solve this subproblem even if you didn't solve the first subproblem.

5. (20 points) Assume that you have a binary search tree  $T$  with  $n$  nodes. Recall that a binary search tree has the property that all the keys in any node  $x$ 's left subtree are less than the key in  $x$ , and all the keys in  $x$ 's right subtree are greater than the key in  $x$ . Assume that you have  $n$  processors, where each processor has a pointer to a unique node in  $T$ . Assume that you have an empty array  $A$  of size  $n$ . Give an  $O(\log n)$  time EREW algorithm to write the keys in  $T$  into  $A$  in sorted order. You can assume that the nodes in  $T$  have a reasonable, say  $O(1)$ , extra unused space that you can use in your algorithm.

Probably you will use some known/named techniques that we discussed in class, but I want your explanation to be from first principles. I don't want to just say that you can use some known technique, without explaining that technique.