CS 1510 Midterm 2
Fall 2016

Instructions: Answer as many of problems 1, 2, 3, and 4 as you can. Answer at most one of problem 5 or 6. If you answer both problem 5 and problem 6, an arbitrary one will be graded.

1. (20 points)

   (a) Consider a problem $\mathcal{P}$ with input $\mathcal{I}$ and output $\mathcal{X}$. Consider a problem $\mathcal{Q}$ with input $\mathcal{J}$ and output $\mathcal{Y}$. Explain how to show using a reduction that if there is a linear time algorithm for $\mathcal{Q}$ then there is a linear time algorithm for $\mathcal{P}$.

   (b) Consider the Matrix Multiplication problem, where the input consists of two n by n matrices A and B, and the output is the product $A \cdot B$. Consider the Matrix Squaring problem where the input is an m by m matrix C, and the output is the product $C \cdot C$. Explain how to show that if there is a linear time algorithm for Matrix Squaring (linear time for Matrix Squaring means time $O(m^2)$ since the input size is m^2) then there is a linear time algorithm for Matrix Multiplication (linear time for Matrix Multiplication means time $O(n^2)$ since the input size is $2n^2$). Do not repeat your explanation from the first part of this question, just explain how to implement the strategy.

2. (20 points)

   (a) Define CRCW-common PRAM. Use one sentence to define what a PRAM and one sentence to say what CRCW-common means.

   (b) Give an algorithm to compute the maximum of n numbers in $O(1)$ time on a CRCW-common machine using $p = n^2$ processors. So $T(n, p = n^2) = O(1)$. Give the complete algorithm. Don't use other parallel algorithms as a black box.

   (c) What is the efficiency of the algorithm in the previous subproblem? Start with a definition of efficiency. You need not have solved the previous subproblem to answer this question.

   (d) What would the Folding Principle say about the time for the algorithm in the first subproblem if there were only $n^{1/6}$ processors? Start with a definition of the Folding Principle. You need not have solved the previous subproblems to answer this question.

3. (20 points) Show that the Vertex Cover Problem is self-reducible. The decision problem is to take a graph G and an integer k and decide if G has a vertex cover S of size k or not. The optimization problem takes a graph G, and returns a smallest vertex cover S in G. Recall that a vertex cover is a collection S of vertices with the property that every edge is incident to a vertex in S. The size of S is just the number of vertices in S. Starting by stating what it means for a problem to be self-reducible.

4. (20 points) Consider the problem of merging two sorted arrays A and B, each containing n integers, into one sorted array C of size $2n$.

   (a) Give an algorithm that runs in time $O(\log n)$ on an EREW PRAM with $p = n$ processors.

   (b) Give an algorithm that runs in time $O(1)$ on a CRCW-common PRAM with $p = n^2$ processors. You can use as a black box an algorithm to compute the maximum of n numbers in $O(1)$ time on a CRCW-common machine using $p = n^2$ processors (so you don't need to explain how to accomplish this).

Remember, answer at most one of problem 5 or 6. If you answer both problem 5 and problem 6, an arbitrary one will be graded.

5. (20 points) Consider the following problem. The input is a graph $G = (V, E)$, a subset R of vertices of G, and a positive integer k. The problem is to determine if there is a subset U of V such that

   - All the vertices in R are contained in U, and
   - the number of vertices in U is at most k, and
   - for every pair of vertices x and y in R, one can walk from x to y in G only traversing vertices that are in U.

   Show that this problem is NP-hard using a reduction from Vertex Cover. Recall that the input for the vertex cover problem is a graph H and an integer ℓ, and the problem is to determine whether H has a vertex cover of size ℓ or not. A vertex cover S is a collection of vertices with the property that every edge is incident on at least one vertex in S. Start with a definition of what it means for a problem to be NP-hard.

6. (20 points) Design a parallel algorithm that takes a binary expression tree, where the leaves are Boolean values 0 or 1, and the internal nodes are the three standard logical operations: NOT, OR, and AND. The output should be the value of the expression represented by the tree. Your algorithm should run in $O(\log^2 n)$ time on a CREW PRAM with n processors, where n is the number of nodes in the tree. You may assume that each processor initially has a pointer to a unique node in the tree.

   In your answer you can use the Euler tour technique as a black box. The Euler technique computes the parallel prefix of a list L, where the positions in the L correspond to an "Euler tour" of the tree where each node in the tree can appear up to three times. But when you call the Euler tour technique, you should specify what the three values in L will be for each node in the tree.