CS 1510 Midterm 2
Fall 2009

The test is a bit long. I suggest keeping your answers short and to the point.

1. (a) (5 points) According to your instructor, what is the most important reason that multiplicative constants are not taken into account when computing run times of algorithms? That is, why settle on this level of detail? You could consider more detail, e.g. consider multiplicative constants but ignore lower order additive terms, or you could consider less detail, e.g. ignore multiplicative logarithmic terms.

   Your answer should contain two sentences, one starting with "The reason that we generally don't consider more detail is ..." and one starting with "The reason that we generally don't consider less detail is ..."

   (b) (5 points) For the problems that we discussed in class, what is the most commonly used algorithmic design paradigm to develop EREW parallel algorithms? Give some informal definition of this paradigm.

   (c) (5 points) Cook and Karp won a Turing awards for discovering the class of NP-complete problems. According to your instructor, why was this viewed as such an important contribution? Start with a definition of what it means for a problem to be NP-complete.

   (d) (5 points) Give an example of a formally defined problem that we discussed in class, for which there is no algorithm that solves that problem. Start with a formally defining "problem" and formally defining what it means for an "algorithm to solve a problem".

2. (20 points) Consider the following two problems:

   SORTING

   INPUT: distinct numbers $z_1, \ldots, z_n$

   OUTPUT: The numbers in increasing sorted order

   CONVEX HULL

   INPUT: points $(x_1, y_1), \ldots (x_n, y_n)$ in the Euclidean plane.

   OUTPUT: The smallest perimeter polygon P that contains each point on either the interior or boundary of P. P is specified by giving the vertices of P in clockwise order starting from an arbitrary vertex of P.

   Show using a reduction that if there is an $O(n)$ time algorithm for CONVEX HULL then there is an $O(n)$ time algorithm for SORTING. Make sure to explain the general set up of how one reduces one problem to another problem in addition to the details specific to this reduction.

3. (20 points) Show that the vertex cover problem is self-reducible.

   Recall the that decision version of vertex cover takes as input a graph G and an integer k and asks whether there is a vertex cover of size k. The optimization version of vertex cover takes as input a graph G and expects as output a minimum cardinality collection of vertices that form a vertex cover. A vertex cover is a collection of S vertices with the property that every edge is incident on at least one vertex in S. A problem is self-reducible if the optimization version can be reduced to the decision version in polynomial time.

4. (a) (10 points) Assume that we have a doubly linked list of n items. Assume that there are n processors, each with a pointer to a unique, but arbitrary, item in the linked list. Assume that the goal is to convert this linked list into an array of of n items maintaining the order in the linked list. Give an algorithm that runs an EREW machine in time $O(\log n)$. You may assume a reasonable (constant) amount of additional memory per node of the linked list that you may use in your algorithm.

   (b) (5 points) 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.

   (c) (5 points) What would the Folding Principle say about the time for the algorithm in the first subproblem if there were only $n^{1/3}$ processors? Start with a definition of the Folding Principle. You need not have solved the first subproblem to answer this question.

5. (20 points) Consider the problem of adding two n bit numbers stored in an array (here n may be much larger than the machine word size). Give an EREW algorithm for this problem that runs in time $O(\log^2 n)$ with n processors.