BU CLA CS 530: Analysis of Algorithms
12 May 1999
End-of-Term Test

Reminder: This is a closed-book test. You are allowed one 8.5 by 11 crib sheet which you have to hand in with your test book. Read carefully all questions before you start, organize your time, start with what you know best.

Problem 1: Order Statistics (10 points)

Let n be an even positive integer. Design an efficient algorithm to partition an input array $A[1\,..\,n]$ of n integers into $n/2$ pairs satisfying the following property: For each pair, we compute the sum of its members, and if $s_1, s_2, \ldots, s_{n/2}$ are these $n/2$ sums, the algorithm finds a partition that *minimizes* the *maximum* sum.

Problem 2: Parallel Algorithms (10 points)

Consider the search problem in which a given ordered list of $n \geq 1$ items (with possible duplicates) is to be searched for the index of an item equal to a given value X, if any such index exists. Binary search solves this problem in sequential time $\mathcal{O}(\log n)$.

(a) Design a time-efficient CREW parallel algorithm for this problem, using as many processors as you need.

(b) What is the parallel running time of your algorithm? Is your algorithm work-efficient? Justify.

Problem 3: Polynomial-time reductions (10 points)

The **clique** problem is the decision problem defined in [CLR] and in lecture: Given an undirected graph $G = (V, E)$ and a natural number k, decide whether there is a clique of size k in G.

The **maximum-clique** problem is an optimization problem defined by: Given an undirected graph $G = (V, E)$, determine a set of vertices $U \subseteq V$ which form a clique of maximum size in G.

The **maximum-clique-size** problem is another optimization problem defined as follows: Given an undirected graph $G = (V, E)$, determine the size of a maximum clique in G.

Prove the following sequence of polynomial reductions (3 of them):

$$\textbf{clique} \leq_P \textbf{maximum-clique} \leq_P \textbf{maximum-clique-size} \leq_P \textbf{clique}.$$

Not all 3 reductions are equally difficult to establish, and the 3 will be graded differently.

Problem 4: Tiling (10 points)

Consider the problem of deciding whether we can tile with dominoes a $n \times n$ chessboard C_n which is missing a few squares. A *dominoe* consists of two adjacent squares. Assuming that dominoes can be rotated, define an efficient decision procedure for each of the following cases:

Case 1: C_n is not missing any squares.
Case 2: C_n is missing exactly one corner square.
Case 3: C_n is missing exactly two corner squares at opposite ends of the same diagonal.

For full credit you have to: (1) make your algorithm as efficient as possible (i.e., make it run as fast as possible) and (2) prove it correct.

Problem 5: Coloring of Plane Figures (10 points)

Design an efficient algorithm for the following problem:

Input:

1. A $k \times \ell$ rectangle R in the plane where k and ℓ are integers $\geqslant 1$.
2. A set L of $n \geqslant 1$ straight lines in the plane.

Output:

1. *Yes/No* the set of regions in R defined by the lines in L can be colored with two colors, say *black* and *white*, so that no two adjacent regions (i.e., two regions sharing an edge) have the same color.
2. If the answer is *Yes* in 1, then construct such a coloring.

For full credit you have to: (1) make your algorithm as time-efficient as possible and (2) prove it correct. For simplicity, make two assumptions:

- the sides of R are parallel to the x-axis and y-axis of the cartesian plane.
- each line in L is parallel to one of the two axes of the cartesian plane.

These two assumptions are not necessary, but they may simplify your reasoning.

Created: 99.05.10