15-451 Algorithms, Spring 2012

Homework 7 (100 pts) **Due: May. 3-4**

oral presentation

Ground rules:

- This is an oral presentation assignment. You should work in groups of two or three. At some point your group should sign up for a 1.5-hour time slot on the signup sheet on the course web page.
- Each person in the group must be able to present every problem. The TA/Professor will select who presents which problem. The other group members may assist the presenter.
- This time you ARE REQUIRED to hand solutions at your presentation. If you do not hand solutions in, we will take off 30% of your grade. If you hand solutions in, the final HW grade will be determined based on the correctness of your write-up.

Question	Points	Score
1	40	
2	30	
3	30	
Total:	100	

(40) 1. **Parallel 2-Dimensional Linear Programming**

This problem aims to create a $O(\log^2 n)$ expected time, $O(n)$ expected work algorithm to solve the two dimensional linear programming problem by modifying the random incremental algorithm covered in class on March first. You may assume that you are given a bounding box as discussed in lecture.

You may assume that we can randomly permute a length n array in $O(\log n)$ time and $O(n)$ work.

(a) Show how to find the minimum and maximum element in a list of n values in $O(\log n)$ time and $O(n)$ work. Use this algorithm to solve one dimensional LP problems.

(b) Show that if we order the 2D half-planes randomly, and perform the incremental algorithm, we are expected to run into an unsatisfied constraint $O(\log n)$ times.

(c) Give an $O(\log n)$ time, $O(n)$ work algorithm that finds among a list of remaining hyperplanes, the first one that's violated.

(d) Put the above together to create a $O(\log^2 n)$ expected time, $O(n \log n)$ expected work algorithm to solve 2 dimensional LP with n constraints.

(e) Show how to modify the above algorithm to only do $O(n)$ expected work and still work in $O(\log^2 n)$ expected time.

(30) 2. **Maximum Independent Set in a Tree**

Let T be a rooted binary tree on n nodes. A subset of nodes I is **independent** if no child-parent pair are in I. It is a **maximum independent set** if it is the maximum cardinality independent set. Note that this is **not** a **maximal** independent set.

(a) Prove that there exists a maximum independent set I in T that contains all of the leaves of T.

(b) Show how to find a maximum independent set I in T in linear time.

(c) Give a parallel algorithm for finding the size of the maximum independent set in tree T in parallel $O(\log n)$ time and $O(n)$ work. Your algorithm should employ parallel tree contraction (see the note below). Please describe the RAKE and COMPRESS operations to be used in your algorithm. It is not necessary to explain how they will be used and how they will share memory.

A note on parallel tree contraction. Recall that in this paradigm, we have a processor for each node of the tree and use two main operations, RAKE and COMPRESS. RAKE moves evaluate the leaves and pass their values up the tree and COMPRESS moves reduce by half chains of vertices with only a single child each. One can think of each node as computing some operation on the values of its children. In class we looked at arithmetic parse trees where the operation at each node was to return either the sum or the product of the children. In general this could be some more complex operation. Observe that if you know how to evaluate the leaves of the tree in constant time, the RAKE operation performed in postorder DFS will give a linear time algorithm. The key to efficiently parallelizing such algorithms is to combine them with the COMPRESS operation.

(30) 3. **NP-Completeness**

Let ϕ be a 3CNF formula. A $\neq$-*assignment* to the variables of ϕ is one where each clause contains two literals with unequal truth values. In other words a $\neq$-assignment satisfies ϕ without assigning three true literals in any clause.

(a) Show that the negation of an $\neq$-assignment to ϕ is also a $\neq$-assignment.

(b) Let $\neq$SAT be the collection of 3CNF formulas that have a $\neq$-assignment. Prove that $\neq$SAT is NP-complete. Hint: consider replacing each clause $c_i = (y_1 \vee y_2 \vee y_3)$ by the two clauses $(y_1 \vee y_2 \vee z_i)$ and $(\overline{z_i} \vee y_3 \vee b)$, where z_i is a new variable for each clause c_i and b is a single additional new variable.

(c) A *cut* in an undirected graph is a partition of the vertices into two disjoint subsets S and T. The size of the cut is the number of edges that have one endpoint in S and the other in T. Let $MAXCUT = \{\langle G, k\rangle \mid G \text{ has a cut of size } k \text{ or more}\}$. Prove that $MAXCUT$ is NP-complete.

Hint: Show that $\neq$SAT $\leq_P MAXCUT$. The variable gadget for variable x is a collection of $3k$ nodes labeled with x and another $3k$ nodes labeled with $\overline{X}$, where k is the number of clauses. All nodes labeled x are connected with all nodes labeled $\overline{x}$. The clause gadget is a triangle of three edges connecting three nodes labeled with the literals appearing in the clause. Do not use the same node in more than one clause gadget. Prove that this reduction works.