Qualifying Examination

Theoretical Computer Science

Friday, September 27, 2007

Part I: Algorithms

ID Number	
Pseudonym	

Problem	Maximum Points	Points Earned	Grader
1	75		
2	50		
3	25		
4	50		
Total	200		

Instructions:

1. This is a closed book exam.
2. The exam is for 3 hours and has four problems. Read all the problems carefully to see the order in which you want to tackle them.
3. Write clearly and concisely. You may appeal to some standard algorithms/facts from text books unless the problem explicitly asks for a proof of that fact or the details of that algorithm.
4. If you cannot solve a problem, to get partial credit write down your main idea/approach in a clear and concise way. For example you can obtain a solution assuming a clearly stated lemma that you believe should be true but cannot prove during the exam. However, please do not write down a laundry list of half baked ideas just to get partial credit.

May the Schwartz be with you.

1 Algorithms questions

1. Arithmetic sequences
   [75 Points]

   An arithmetic sequence of length k is a sequence of numbers $a_1, \ldots, a_k$, where $a_i = a_1 + \Delta(i-1)$, for $i = 1, \ldots k$, where both a_1 and Δ are positive numbers.

   The input to this problem is a sorted sequence of integer numbers $X = x_1, \ldots, x_n$.

   (A) **[20 Points]** Provide a quadratic time algorithm that finds the longest arithmetic subsequence in X. (Deciding if X has an arithmetic subsequence of length three is 3SUM-Hard, and as such, it is unlikely that a faster algorithm exists.)

   Note, that if an arithmetic sequence $a_1, \ldots, a_k$ appears in X, then for all i, $a_i \in X$ (i.e., you can not omit missing elements).

   (B) **[25 Points]** A ***k-partition*** of X into k sequences, is a partition of X into k disjoint arithmetic subsequences. Show an algorithm, that given X and k computes, a k-partition of X, if it exists, as quickly as possible. (A $n^{O(k)}$ time algorithm is trivial. A faster solution is possible and expected.)

   (C) **[15 Points]** A ***k-cover*** of X into k sequences, is a collection of k arithmetic subsequences such that each element of X belongs to at least one such sequence (i.e., an element can belong to several such sequences). Show an algorithm that, given X, computes a 2-cover of X, if it exists, as quickly as possible.

   (D) **[15 Points]** Given X that has a k-cover, provide an algorithm that computes a $O(k \log n)$-cover. (A $O(k \log k)$-cover can be computed in polynomial time, but it is non-trivial and not required.)

2. Bin packing.
   [50 Points]

   An algorithm **Alg** for a problem Π is a pseudo-polynomial time algorithm if its running time is polynomial when the numbers in the input are represented in unary. Consider the following multiple knapsack problem (MKP for short). You are given n items. Each item i has an integer size s_i and an integer profit p_i associated with it. Also given are m knapsacks with integer capacities $B_1, B_2, \ldots, B_m$. The goal is to pack a maximum profits subset of the items in the bins. A FPTAS (fully polynomial time approximation scheme) for an optimization problem (say maximization) is an algorithm that for any given instance of size n and an error parameter $\varepsilon \in (0, 1)$ outputs a solution of value at least $(1 - \varepsilon)OPT$ (OPT is the value of an optimum solution) and runs in time polynomial in n and $1/\varepsilon$.

   (A) **[35 Points]** Show that the MKP problem admits a pseudo-polynomial time algorithm for any fixed m.

   (B) **[15 Points]** Show that MKP with $m = 2$ does not admit an FPTAS unless P = NP. You may want to use the NP-Complete Partition problem for this. In the Partition problem you are given n integers $a_1, a_2, \ldots, a_n$ and the question

is whether there is a partition of these into two sets such that the sums of the numbers in each set is exactly the same.

3. Vertex disjoint paths
[25 Points]

You are given an undirected graph $G = (V, E)$ and integer weights on the nodes: $w(v)$ represents the weight of node v. Also given are two nodes s, t. The objective is to find two internally node-disjoint paths P_1 and P_2 between s and t to minimize the total weight of nodes on P_1 and P_2. Give a polynomial time algorithm for this problem. Extra credit if your algorithm runs in $O(m)$ time where m is the number of edges in the graph.

4. Misc.
[50 Points]

(A) **[15 Points]** Prove that sorting n numbers requires $\Omega(n \log n)$ time in the comparison model.

(B) **[15 Points]** Let $G = (V, E)$ be an undirected graph, with weight $w(\cdot)$ associated with the vertices. Show a 2-approximation algorithm for computing the minimum weight vertex cover. Note that the "standard" approximation algorithm for vertex cover works only for the unweighted case.

(C) **[20 Points]** What is the solution for the following recurrence. ***Prove*** your answer.

$$T(n) = T\left(\frac{1}{3}n\right) + T\left(\frac{2}{3}n\right) + \frac{n}{\log n}.$$

(Here and in other questions: Write concise, full and formal proofs. Hand-waving or omitting details would be met with due scorn.)

Qualifying Examination

Theoretical Computer Science

Friday, September 27, 2007

Part II: Automata and Complexity

ID Number	
Pseudonym	

Problem	Maximum Points	Points Earned	Grader
1	50		
2	50		
3	50		
4	50		
Total	200		

2 Complexity questions

1. **[50 Points]** Recall that AM(or AM[2]) is the class of languages L for which there is an Arthur-Merlin protocol[1] (A, M), such that

$$x \in L \implies \mathbf{Pr}\big[(A,M)(x) = 1\big] > 2/3$$
$$x \notin L \implies \mathbf{Pr}\big[(A,M^*)(x) = 1\big] < 1/3$$

for all possible probabilistic Turing Machines M^*.

Show that AM $\subseteq$ NP/ poly. Be sure to explain all the details (or point out any details that you omit to explain).

Hint: For partial credit, show that BPP $\subseteq$ P/*poly*.

2. **[50 Points]** A graph property $\mathcal{P}_n$ is a subset of $\mathcal{G}_n$, the set of all graphs on n vertices. $\mathcal{P}_n$ is called trivial $\mathcal{P}_n = \mathcal{G}_n$ or $\mathcal{P}_n = \phi$. A graph property $\mathcal{P}_n$ is called a *matroid* if it has the following two properties: (a) if $G \in \mathcal{P}_n$ and G' is a subgraph of G (on n vertices), then $G' \in \mathcal{P}_n$; (b) if $G \in \mathcal{P}_n$ and e is an edge not in G, then there exists and edge e' in G such that $(G - \{e'\}) \cup \{e\} \in \mathcal{P}_n$.

   (A) Is $\mathsf{cyclic}_n = \{G \in \mathcal{G}_n | G \text{ has a cycle}\}$ a matroid? How about $\mathsf{acyclic}_n = \mathcal{G}_n - \mathsf{cyclic}_n$? Prove.

   (B) Recall that a graph property $\mathcal{P}_n \subseteq \mathcal{G}_n$ is called *elusive* (for decision trees) if the decision tree complexity for $\mathcal{P}_n$ is $\binom{n}{2}$ (that is, any decision tree correctly deciding the graph property must have a path in which it queries all possible $\binom{n}{2}$ edges of the input). Show that if $\mathcal{P}_n$ is a matroid and not trivial, then it is elusive.

   Hint: For partial credit, just give an appropriate adversarial strategy that forces a decision tree to query all edges (without proof).

3. **[50 Points]** $T \subseteq \{0,1\}^*$ is said to be a *binary tree* if T is closed under prefixes, i.e., if $v \in T$ and u is a prefix of v then $v \in T$. For example a binary tree with vertices $\{0,1,2,3\}$, with 0 as the root, 1 as the left child of 0, 2 as the right child of 0 and 3 as the right child of 1, is represented by the set $\{\epsilon, 0, 1, 01\}$. Recursiveness and infiniteness of a tree is defined in terms of whether the set T is recursive or infinite. A *path* p in T is a (finite or infinite) word over $\{0,1\}$ such that every prefix of p is in T. A path being recursive (or infinite) can be defined analogously.

   (A) **König's Lemma:** Show that every infinite binary tree T has an infinite path. [30% of the total points for this problem]

   (B) Show that there an infinite, recursive tree T such that none of its infinite paths is recursive, i.e., König's Lemma is not "effective". *Hint:* You may use without proof

[1] In an AM-protocol (A, M), A is a polynomial time Turing Machine and M is a computationally unbounded probabilistic Turing Machine. The random variable $(A, M)(x)$ is 1 if and only if Arthur accepts the proof: i.e., $A(x, r, z) = 1$, where $r \leftarrow \{0,1\}^{\ell(|x|)}$ and $z \leftarrow M(x, r)$ (with $|r|$ and $|z|$ being polynomial in $|x|$).

the following observation from the Qual in Spring 2004 — There are disjoint r.e languages A, B for which there is no recursive language C such that $A \subseteq C$ and $B \cap C = \emptyset$; so there are "recursively inseparable" r.e languages. [70% of the points]

4. Some automata theory required.
[50 Points]

Recall D_n is the language over $\{(_1, \ldots, (_n,)_1, \ldots,)_n\}$ of "matched parenthesis", where $(_i$ is matched by $)_i$. More formally, D_n is generated by the following grammar

$$S \to \epsilon \mid SS \mid (_1 S)_1 \mid (_2 S)_2 \mid \cdots \mid = (_n S)_n$$

Prove that every context-free language L can be written in the form $h(g^{-1}(D_n) \cap R)$ for some n, where R is regular language and g, h are homomorphisms.