Qualifying Examination

Theoretical Computer Science

Thursday, March 2, 2017

Part I: Algorithms

Instructions:

1. This is a closed book exam.
2. The exam has four problems worth 25 points each. Read all the problems carefully to see the order in which you want to tackle them. You have all day (9am–5pm) to solve the problems.
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 force be with you.

Problem 1: A matroid $\mathcal{M}$ is a tuple $(N, \mathcal{I})$ where N is finite ground set and $\mathcal{I} \subseteq 2^N$ is a collection of *independent set*, that satisfy the following conditions.

1. $\mathcal{I}$ is non-empty, in particular, $\emptyset \in \mathcal{I}$.
2. $\mathcal{I}$ is downward closed, that is, if $A \in \mathcal{I}$ and $B \subset A$ then $B \in \mathcal{I}$.
3. If $A, B \in \mathcal{I}$ and $|A| < |B|$ then there is an element $e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$.

A *hypergraph* $G = (V, E)$ consists of a finite set of vertices V a collection of edges E where each $e \subseteq V$. Graphs are a special case when $|e| = 2$ for each e. For a set $S \subseteq V$ let $\delta_G(S) = \{e \mid e \cap S \neq \emptyset, e \cap V - S \neq \emptyset\}$ denote the set of edges that cross S. Given $s, t \in V$ an s-t the mincut value between s and t is $\min_{S: S \cap \{s,t\} = 1} |\delta_G(S)|$.

- Describe a polynomial time algorithm that given a hypergraph $G = (V, E)$ and s, t computes the mincut value between s and t. What is the running time of your algorithm as a function $n = |V|, m = |E|$ and $p = \sum_{e \in E} |e|$?

- Every graph $G = (V, E)$ implicitly defines a matroid $\mathcal{M}_G = (E, \mathcal{I})$ where $\mathcal{I} = \{A \subseteq E \mid A \text{ induces a forest}\}$. When G is a hypergraph it is not obvious what it means for A to induce a forest. Consider the following definition. We say that $A \subseteq E$ is forest-representible if one can choose for each $e \in A$ two nodes in e such that the chosen pair when viewed as edges form a forest in the graph sense. Show that when $G = (V, E)$ is a hypergraph $(E, \mathcal{I})$ where $\mathcal{I} = \{A \subseteq E \mid A \text{ is forest representible.}\}$ is a matroid. This is called the hypergraphic matroid.

Problem 2: Given a set P of n points in 2-dimensions and an integer $k \le n$, we want to find k axis-aligned unit squares that maximize the number of points covered.

1. Let b be an integer. First show that in the special case when all points P lie in $\{(x, y) \in \mathbb{R}^2 : \lfloor x \rfloor \not\equiv 0 \bmod b \text{ and } \lfloor y \rfloor \not\equiv 0 \bmod b\}$, the problem can be solved exactly in $n^{O(b^2)}$ time by dynamic programming.

2. Give a polynomial-time approximation scheme for the general problem.

Problem 3: We are given a set of n coins. Most of the coins are genuine and have the same weight, say, 1 unit, but there may be up to 2 fake coins, which have weight different from 1 unit. We have a scale that can weigh any given subset of coins, i.e., determine the exact sum of the weights in the subset. The problem is to decide whether there are any fake coins.

(You are not required to find them, but just decide existence. Note that we can't simply weigh the whole set, since the adversary could create 1 fake coin of weight $1 + \delta$ and another of weight $1 - \delta$.)

1. Prove tight upper and lower bounds (up to constant factors) on the worst-case number of weighings needed to solve the problem by a deterministic algorithm as a function of n.

2. Give a Monte Carlo randomized algorithm that solves the problem using $O(1)$ weighings with error probability at most 0.001.

(Hint: Consider the binary representation of *coin numbers* for the first part.)

Problem 4: Suppose you are given a dag G with a unique source s and a unique sink t. Two paths in G are *disjointish* if any common subpath contains at most 42 consecutive edges.

1. Describe an algorithm to find two disjointish paths from s to t in G with maximum total length. For full credit, your algorithm should run in $O(VE)$ time.

2. Describe an algorithm to compute the size of the largest set of pairwise-disjointish paths from s to t in G. For full credit, your algorithm should run in polynomial time.

[Hint: Try 1 before jumping to 42.]