# Qualifying Examination
## Theoretical Computer Science
Thursday, March 1, 2012
## 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:** You are given $n$ real numbers $x_1, \ldots, x_n$. Many of the numbers are repeated, and they are really taken from a much smaller set $T$ that has $t$ elements (and $t < n$). In particular, for a number $t \in T$, let $n(t)$ be the number of times it appears in the input. That is $\sum_{t \in T} n(t) = n$. Let $p_t = n(t)/n$ be the "probability" of $t$. We would like to sort these $n$ numbers in time

$$O\left(n + n \sum_{t \in T} p_t \ln \frac{1}{p_t}\right).$$

(We assume that if $p_i = 0$ then $p_i \ln \frac{1}{p_i} = 0$.) Unfortunately, all you can do is to compare two elements (i.e., you can not use hashing, bit tricks, etc).

(A) (5 points.) As a concrete example, consider the case where for every integer $i > 0$, there is an item that appears $\Theta(n/i^2)$ times in the input, for $i = 1, \ldots, O(\sqrt{n})$. What is the running time of the above sorting algorithm for this input?

(B) (20 points.) Assume you are given an algorithm that can find the median in linear time. That is, given $k$, you can assume that one can find the $k$ smallest numbers in the input in $O(n)$ time (naturally, these numbers are not necessarily distinct and the $k+1$ smallest element might be equal to the $k$ smallest element, etc).

Show a sorting algorithm with the desired running time. Prove the running time of your algorithm.

*Hint:* You might want to first try the extreme case that $p_t = 1/n$ for all elements. Then consider the less extreme case, where $p_t = 1/k$, for some $k \leq n$. Then consider the case when some of the $p_t$s are very large and some are very small.

**Problem 2:** Let $G = (V, E)$ be a directed graph with non-negative edge-capacities; we let $c(e)$ denote the capacity of edge $e$. Let $s$ be a source and $t$ be a sink node. The standard $s$-$t$ flow can be thought of as assigning a non-negative number to each path $p \in \mathcal{P}$ where $\mathcal{P}$ is the set of all directed simple paths from $s$ to $t$ in $G$; that is $f : \mathcal{P} \to \mathbb{R}_+$ is a flow and it satisfies the capacities if the total flow one each edge $e$ from all the paths that use $e$ is at most $c(e)$. In the context of fault-tolerant networks there is a generalization of flows to what are called multi-route flows. For an integer $k \geq 1$ a $k$-route $s$-$t$ flow is defined as follows. Let $\mathcal{Q}_k = \{(p_1, p_2, \ldots, p_k) \in \mathcal{P}^k \mid p_1, p_2, \ldots, p_k \text{ are mutually edge-disjoint}\}$, that is, $\mathcal{Q}_k$ is the set of all $k$-tuples of edge-disjoint $s$-$t$ paths. A $k$-route flow is an assignment of non-negative numbers to each tuple in $\mathcal{Q}_k$ such that the total flow on any edge $e$ is at most $c(e)$.

(A) (13 points) Write a linear program to find the maximum $s$-$t$ $k$-route flow that follows the definition above. Write its dual. Show that the separation oracle for the dual can be efficiently solved in polynomial time. What does this tell you about solving the maximum $s$-$t$ $k$-route flow?

(B) (12 points) Define a $k$-route $s$-$t$ cut a set of edges $E'$ such that the $k$-route flow from $s$ to $t$ in $G \setminus E'$ is 0. Given an example of a graph $G$ such that $s$ can reach $t$ but the 2-route flow is 0. Describe a polynomial-time algorithm to find the minimum-cost 2-route $s$-$t$ cut. Can you generalize your algorithm to any fixed $k$? *Hint:* Reduce to the standard $s$-$t$ mincut problem after some preprocessing.

**Problem 3:** Let $X$ be a string of $n$ bits. An *increasing binary partition* of $X$ is a partition of $X$ into substrings with strictly increasing binary values. For example, the 50-bit string

`11001100011001110010000000001011011010000110001001`

has the following increasing binary partitions, among *many* others. (Numbers under the brackets are the decimal values of the bracketed substrings.)

$$\underbrace{\texttt{1}}_{1}|\underbrace{\texttt{1001}}_{9}|\underbrace{\texttt{10001}}_{17}|\underbrace{\texttt{10011}}_{19}|\underbrace{\texttt{100100}}_{36}|\underbrace{\texttt{0000000101101}}_{45}|\underbrace{\texttt{1010000}}_{80}|\underbrace{\texttt{110001001}}_{393}$$

$$\underbrace{\texttt{1}}_{1}|\underbrace{\texttt{10}}_{2}|\underbrace{\texttt{011}}_{3}|\underbrace{\texttt{00011001}}_{25}|\underbrace{\texttt{1100100}}_{100}|\underbrace{\texttt{000000010110110}}_{182}|\underbrace{\texttt{10000110001001}}_{8585}$$

$$\underbrace{\texttt{11001100011001110010000000001011011010000110001001}}_{898973165265289}$$

The *weight* of an increasing binary partition is the length of its last substring. The partitions shown above have weight 9, 14, and 50, respectively.

Describe and analyze an efficient algorithm to compute, given a bitstring $X$, the weight of a lightest increasing binary partition of $X$. For example:

- If $X =$ `011011100`, your algorithm should return 3, for the partition `0|1|10|11|100`.
- If $X =$ `111111111`, your algorithm should return 4, for the partition `11|111|1111`.
- If $X =$ `1011010110110`, your algorithm should return 5, for either of the partitions `101|10101|10110` or `10|11|0101|10110`.
- If $X =$ `000000000000`, your algorithm should return 12, for the trivial partition.

Your algorithm should run on a standard integer RAM with $O(\log n)$-bit words; the input is a standard array of $n$ words, each holding the value 0 or 1.

**Problem 4:** Consider the following stochastic process which happens in rounds. We start with a collection of $n$ balls and 2 bins. In a given round each ball is thrown uniformly at random into one of the two bins and independently of other balls. If $X_1$ and $X_2$ are the number of balls that land in the bins then we retain $|X_1 - X_2|$ balls and remove the rest (thus $2\min\{X_1, X_2\}$ balls are removed). The process ends when there are no more balls left.

- (15 points) Prove that the expected number of balls at the end of the first round is $O(\sqrt{n})$.
- (10 points) Prove the best upper bound you can on the expected number of rounds for the process to terminate if it starts with $n$ balls.