CS 125 Algorithms & Complexity — Fall 2016

Problem Set 4

Due: 11:59pm, Friday, September 30th

See homework submission instructions at `http://seas.harvard.edu/~cs125/fall16/schedule.htm`

Problem 1

Although it is not known how to prove $\omega(n)$ lower bounds for any natural offline problems, it *is* known how to prove lower bounds on the time required to perform any operation for various *data structural* problems. Here we will focus on how to do so for a variant of the word RAM model we saw in class. More specifically, in this problem we consider the model variant in which w, the word size, has some fixed value in the beginning that never changes.

A common way to argue time lower bounds for solving data structural problems in the word RAM model is to show a lower bound in a different model: the *cell probe model*. In this model, there is the *memory*, which is an array of S cells `M[0]`, ..., `M[S-1]`, each holding a value in $\{0, \ldots, 2^w - 1\}$. The memory stores all the data maintained by the data structure. There is also an *algorithm* $\mathcal{A}$. The algorithm can send memory two types of messages:

- `read(i)`: the memory responds with a w-bit message containing the contents of `M[i]`
- `store(x, i)`: the memory performs the change `M[i]` $\leftarrow x$

Whenever there is a data structural operation, i.e. an update or query, the algorithm can adaptively decide to send a sequence of the above types to the memory. "Adaptive" here refers to the fact that the algorithm can wait to hear the return message from a `read` before deciding what the next message it sends should be.

The *worst-case running time* $T(n)$ of a data structural operation `op` in the cell probe model is defined to be the maximum number of messages sent between $\mathcal{A}$ and `M` over all possible states of the data structure before `op`, over all datasets of size at most n.

(a) (2 points) Suppose that for some data structural problem, *any* cell probe algorithm requires worst-case running time $T(n)$ to implement operation `op`. Show that this implies the worst-case running time of any *word RAM* solution must be $\Omega(T(n))$.

(b) (2 points) Consider the following `PrefixXOR` data structural problem on bits: we are to maintain n bits $x_1, \ldots, x_n$, all initialized to 0, subject to the following operations:

- `query(i)`: return $x_1 \oplus \ldots \oplus x_i$
- `update(i, z)`: set $x_i \leftarrow z$ (where $z \in \{0, 1\}$)

Our goal will be to show that for any correct cell-probe algorithm solving `PrefixXOR`, either `query` or `update` requires $\Omega(\log n / \log\log n)$ worst-case time when $w = 1$. This will be accomplished via a *reduction*. Consider the following other data structural problem which we call the `Guess` problem. In `Guess`, the data structure must maintain a value $z \in \{1, \ldots, n\}$ under the following operations:

- `set(v)`: sets $z \leftarrow v$
- `less?(j)`: returns a Boolean indicating whether $j < z$

Suppose we are only interested in sequences of operations in which there is exactly one `set` operation, in the beginning, followed by some number of `less?` operations. Show that if there is a space-S cell-probe algorithm for `PrefixXOR` with update time t_u and query time t_q, then there is a space-S cell-probe algorithm for `Guess` implementing `set` in time t_u and `less?` in time t_q. Conclude it suffices to prove an $\Omega(\log n / \log\log n)$ lower bound on the maximum runtimes for `set` and `less?` in a correct cell-probe implementation of `Guess`.

(c) (3 points) Consider the following game: there is a collection $\mathcal{D}$ of binary vectors $D_1, \ldots, D_n \in \{0,1\}^b$, each with support size at most r (i.e. at most r non-zero entries). There are also two friends Alice and Bob. Alice and Bob both know the entire collection $\mathcal{D}$, and in addition, Bob also knows an index $i \in \{1, \ldots, n\}$. Alice likes to play a guessing game to figure out i: she can repeatedly ask some $j \in \{1, \ldots, b\}$ to Bob, and Bob replies with the jth bit of D_i. Suppose Alice has a strategy that is always guaranteed to successfully identify i using at most t questions. Then prove that

$$n \leq \sum_{i=0}^{r} \binom{t}{i}$$

(d) (3 points) Use (c) to show that for any correct cell probe algorithm solving `Guess` with $w = 1$, the worst-case runtime of either `set` or `less?` must be $\Omega(\log n / \log\log n)$, irrespective of S. You are allowed to use, without proof, that for any $0 \leq r \leq t$, $\sum_{i=0}^{r} \binom{t}{i} \leq (100t/r)^r$.

Problem 2

Give a 3-tape TM that computes the multiplication function. The input alphabet should be $\{0, 1, \times\}$ and given an input of the form $x \times y$, where $x, y \in \{0,1\}^*$ are interpreted as binary representations of nonnegative integers, the TM should output a binary representation of the product. You do not need to worry about improperly formatted inputs, and the output can contain extra leading zeroes. Your TM heads can also stay still in a transition; they do not need to move at every time step. Provide both an implementation-level description of your TM (in prose) and an annotated state diagram. Your state diagram can use shorthands such

as "$(0, \Gamma, \sqcup) \mapsto (1, \Gamma, \sqcup, R, R, S)$," which means $\delta(0, \gamma, \sqcup) = (1, \gamma, \sqcup, R, R, S)$ for all $\gamma \in \Gamma$. The S denotes "stand still".

Suggestion: think through a few different possible implementations before deciding which one to formalize. A judicious choice can make for a much simpler state diagram!

Problem 3

The goal of this question is to prove that any single-tape Turing Machine deciding if a string xy is an even-length palindrome must take time $\Omega(n^2)$ where $n = |x| = |y|$ (the EVENPAL language from class). We do so by exploring "crossing sequences". Given a Turing Machine M and input xy, the crossing sequence at location i is a sequence of states $(q_1, q_2, \ldots, q_\ell)$ such that q_1 is the state of Turing machine when it first crosses over from the ith tape cell to the $(i+1)$st cell, and q_2 is the state at the next crossing (from $(i+1)$ to i) and so on. You may assume that the input is of the form $\{0,1\}^*$.

(a) (5 points) Prove that if M decides EVENPAL then the nth crossing sequences must be distinct for every pair of distinct even length palindromes.

(b) (5 points) Expand on the idea above to prove that M must take time $\Omega(n^2)$.

Problem 4

The term rewriting problem is defined below:

Input: Finite alphabet Σ, source string $s \in \Sigma^*$, target $t \in \Sigma^*$ and collection of rewrite rules $\alpha_1 \to \beta_1, \ldots, \alpha_m \to \beta_m$.

Question: Decide if the string s can be rewritten as the string t by applying a series of rewrite steps, i.e., $s = s_0, s_1, s_2, \ldots, s_m = t$ where each rewrite step $s_j \to s_{j+1}$ is obtained by replacing some occurrence of some string α_{i_j} in s_j by β_{i_j}.

Prove that term rewriting is a complete model of computing. Specifically given a Turing Machine M and input x, compute an input $(s, t, (\alpha_i \to \beta_i)_i)$ for the term-rewriting problem such that M halts and accepts x if and only if s can be rewritten as t.