Written Qualifying Exam
Theory of Computing

Spring 2001

Friday, May 18, 2001

This is a *three hour* examination. All questions carry the same weight. Answer all of the following *six* questions.

• Please check to see that your name and address are correct as printed on your blue-card.
• Please *print* your name on each exam booklet. Answer each question in a *separate* booklet, and *number* each booklet according to the question.

Read the questions carefully. Keep your answers legible, and brief but precise. Assume standard results, except where asked to prove them.

Problem. 1 New booklet please. [10 points]

1. Consider the language consisting of the set of all strings with sequence of 0's followed by the same number of 1's.

$$\mathcal{L} = \{o^n 1^n | n \geq 0\}.$$

   Show that it's not regular.

2. Using a homomorphism reduction show that the set of palindromes PAL(Σ) over $\Sigma = \{0, 1\}$ is not regular either. (DO NOT USE THE PUMPING LEMMA DIRECTLY.)

$$\text{PAL}(\{0,1\}) = \{\sigma \in \{0,1\}^* | \sigma = \sigma^R\}.$$

3. Consider the set of reverse palindromes RPAL(Σ) over $\Sigma = \{A, T, C, G\}$, defined as follows: The complementation operation is a homomorphism defined by $A^C = T$, $T^C = A$, $C^C = G$, and $G^C = C$. Show that the language RPAL(Σ) is not regular. (Again, DO NOT USE THE PUMPING LEMMA DIRECTLY.)

$$\text{RPAL}(\{A,T,C,G\}) = \{\sigma \in \{A,T,C,G\}^* | \sigma^R = \sigma^C\}.$$

Solution

1. Consider the string $uvw = 0^n 1^n$ for some fixed n. Since $|uv| < n$ and $|v| > 0$, we have $v = 0 \cdots 0$ and $|v| < n$. But then, $uw = 0^{n-|v|}1^n$ (the string uv^iw, with $i = 0$) must be in the same language. An impossibility.

2. Let $\mathcal{L}' = \text{PAL}(\{0,1\})$. Consider two homomorphisms:

$$\begin{aligned} h_1 &: \{a,b,c\}^* \to \{0,1\}^* \\ &: a \mapsto 0; b \mapsto 1; c \mapsto 0 \\ h_2 &: \{a,b,c\}^* \to \{0,1\}^* \\ &: a \mapsto 0; b \mapsto \epsilon; c \mapsto 1 \end{aligned}$$

Now consider the language

$$\begin{aligned} &h_2[h_1^{-1}(\mathcal{L}' \cap 0^*10^*) \cap a^*bc^*] \\ &= h_2[h_1^{-1}(\{0^n10^n | n \geq 0\}) \cap a^*bc^*] \\ &= h_2[\{a^nbc^n | n \geq 0\}] \\ &= \{0^n1^n | n \geq 0\} = \mathcal{L}] \end{aligned}$$

Since 0^*10^* and a^*bc^* are regular and regular sets are closed under intersection and homomorphisms, we conclude that if PAL($\{0,1\}$) is regular then so is $\{0^n1^n | n \geq 0\}$.

3. Consider the following surjective homomorphism h

$$\begin{aligned} h &: \{A,T,C,G\}^* \to \{0,1\}^* \\ &: A \mapsto 0; T \mapsto 0; C \mapsto 1; G \mapsto 1 \end{aligned}$$

Thus $h(X^C) = h(X) = 0$, if $X \in \{A,T\}$ and $h(X^C) = h(X) = 1$, if $X \in \{C,G\}$. Thus for all $\sigma \in \{A,T,C,G\}^*$, $h(\sigma^C) = h(\sigma)$. And for all $\sigma \in \mathrm{RPAL}(\{A,T,C,G\})$, $h(\sigma^C) = h(\sigma) = h(\sigma)^R$. Thus

$$h(\mathrm{RPAL}(\{A,T,C,G\})) = \mathrm{PAL}(\{0,1\} = \mathcal{L}',$$

which is non-regular.

Problem. 2 New booklet please. [10 points]

Graph 3-Colorability: Given an undirected graph $G = (V,E)$, it is said to be K-colorable, if there is a mapping $f : V \to [1..K]$ such that every pair of adjacent vertices are assigned distinct colors.

- 3-Colorability(G)
- Input: An undirected graph $G = (V,E)$.
- Output: If G is 3-colorable then return true; Otherwise, return False.

3-Colorability problem is known to be NP-complete. Use this fact, to show that the following problem is NP-complete: Let $n = |V|$ and $m = |E|$. Given: a system of multivariate polynomials over the field of complex numbers, involving n variables and $m+n$ equations each of degree 3 or less. Decide whether this system of polynomial equations is solvable.

Solution

Let **C** denote the field of complex numbers. Let $\{1, \omega, \omega^2\}$ denote the three cube roots of unit. These three constants will be used to represent three colors. For each vertex v_i, associate a variable x_i and introduce the following equations into the system:

$$x_i^3 - 1 = 0.$$

This enforces that the vertex v_i takes a color in $\{1, \omega, \omega^2\}$. Now the condition that each pair of adjacent vertices $[v_i, v_j] \in E$ are assigned distinct colors can be enforced by the following set of equations:

$$x_i^2 + x_i x_j + x_j^2 = 0, \quad \text{where } [v_i, v_j] \in E.$$

Note that the graph G is 3-colorable if and only if the constructed system of equations in $\mathbf{C}[x_1, \ldots, x_n]$ has a solution in $\mathbf{C}^n$.

Problem. 3 New booklet please. [**10 points**] Show that it is undecidable if a Turing Machine with alphabet $\{0, 1, B\}$ ever prints three consecutive 1's on its tape.

Solution

For each Turing Machine M_i, construct $\hat{M}_i$, which on blank tape simulates M_i on blank tape. However, $\hat{M}_i$ uses 01 to encode a 0 and 10 to encode a 1. If M_i's tape has a 0 in cell j, $\hat{M}_i$ has 01 in cells $2j - 1$ and $2j$. If M_i changes a symbol, $\hat{M}_i$ changes the corresponding 1 to 0 and then the paired 0 to 1. With this design $\hat{M}_i$ never has three consecutive 1's on its tape. Now further modify $\hat{M}_i$ so that if M_i accepts, $\hat{M}_i$ prints three consecutive 1's and halts. Thus $\hat{M}_i$ prints three consecutive 1's iff $\epsilon \in \mathcal{L}(M_i)$. Thus the question of whether an arbitrary Turing Machine ever prints three consecutive 1's is undecidable.

Please turn over.

Problem. 4 New booklet please. [10 points]

Let T be a binary tree. For each node v in T, let $h(v)$ be the length of the longest downward path from v to a leaf, and let $d(v)$ be the length of the shortest downward path from v to a leaf. (Thus at a leaf v, $d(v) = h(v) = 0$.) Define a *whole* binary tree to be one in which every internal node has two children, and in addition, $h(v) \leq 2 \cdot d(v)$ for all nodes v.

Let $T(\ell)$ be the smallest possible number of nodes in a whole binary tree with $h(root) = \ell$. Determine $T(\ell)$ exactly.

Solution

Let Tree(ℓ) denote a whole binary tree with $h(root) = \ell$ and the smallest possible total number of nodes. Without loss of generality, assume that the left-most path of the tree is the longest downward path from the root to a leaf. Then it is easy to see that the left subtree of Tree(ℓ) must be Tree($\ell-1$) and the right subtree must be a complete binary tree of depth $\lceil \frac{\ell}{2} \rceil - 1$. Hence the left subtree has $T(\ell-1)$ many nodes and the right subtree has $2^{\lceil \ell/2 \rceil} - 1$ many nodes. Hence the recurrence relation for $T(\ell)$ is as follows:

$$\begin{aligned} T(\ell) &= 1 + T(\ell-1) + 2^{\lceil \ell/2 \rceil} - 1 \\ &= T(\ell-1) + 2^{\lceil \ell/2 \rceil} \end{aligned}$$

Solving the above equation by telescoping, we get

$$T(\ell) - T(0) = \sum_{i=1}^{\ell} 2^{\lceil i/2 \rceil}.$$

The above equation can be simplified as follows:

$$T(\ell) - T(0) = \begin{cases} 4 \cdot 2^{\ell/2} - 4, & \text{if } \ell = \text{even;} \\ 3 \cdot 2^{(\ell+1)/2} - 4, & \text{if } \ell = \text{odd.} \end{cases}$$

Note that $T(0) = 1$. Hence

$$T(\ell) = \begin{cases} 4 \cdot 2^{\ell/2} - 3, & \text{if } \ell = \text{even;} \\ 3 \cdot 2^{(\ell+1)/2} - 3, & \text{if } \ell = \text{odd.} \end{cases}$$

Problem. 5 New booklet please. [10 points]

(a) [5 points] The input is a sequence of n elements $x_1, x_2, \ldots, x_n$ that we can read *sequentially*. We want to use a memory that can only store $O(k)$

elements at a time. Give a high level description of an algorithm that finds the kth *smallest element* in $O(n)$ time.
Hint: Use the linear-time median algorithm.
(b) [**5 points**] Suppose that you are given an algorithm that finds the kth smallest element in a given set. Prove that that the comparisons used by this algorithm are sufficient to partition the set in two groups of elements: the ones that are smaller than the kth element and the ones that are larger than the kth element.

Solution
(a) We first store $2k$ elements and find the median. All the elements greater than the median are eliminated. We then read the next k elements, find the median, eliminate the k larger than the median and so on, until all n elements have been read. Such phases are repeated at most $n/k+1$ times. Each phase requires finding a median of at most $2k$ elements and can be performed in $O(k)$ time. Thus the complete algorithm algorithm requires $O(n)$ time.
(b) Let x be the kth smallest element. Suppose that it is not possible to divide the set into the two groups, as required. This implies that there exists at least one element, say y, such that with the given number of comparisons we can distinguish whether this element y is smaller or larger than x. Thus, the outcome of all the comparisons is consistent with both y being larger or smaller than x. This is possible, as depending on which case we choose, we will obtain a different kth smallest element.

Problem. 6 New booklet please. [**10 points**]

Define a *common subsequence* (not necessarily contiguous) of two strings $V = v_1 \cdots v_n$ and $W = w_1 \cdots w_m$ as a pair of sequence of indices:

$$1 \leq i_1 < \cdots < i_k \leq n \quad \text{and} \quad 1 \leq j_1 < \cdots < j_k \leq m,$$

such that

$$\forall_{1 \leq t \leq k} v_{i_t} = w_{j_t}.$$

Let $s(V, W) = k$ be the length of a *longest common subsequence* (LCS) of V and W. For example, the LCS of two strings $V = ATCTGAT$ and $W = TGCATA$ is $TCTA$ and $s(V, W) = 4$. Devise an efficient algorithm to compute $s(V, W)$.

Solution

A simple dynamic programming algorithm to compute $s(V, W)$ is as follows: Let $s_{i,j}$ be the length of LCS between i-prefix $V_i = v_1 \cdots v_i$ of V and the j-prefix $W_j = w_1 \cdots w_j$ of W. Thus $s_{i,0} = s_{0,j} = 0$ for all $1 \leq i \leq n$ and $1 \leq j \leq m$. Then $s_{i,j}$ can be computed by the following recurrence:

$$s_{i,j} = \max \begin{cases} s_{i-1,j} \\ s_{i,j-1} \\ s_{-1,j-1}, & \text{if } v_i = w_j \end{cases}$$

The complete dynamic programming algorithm with trace-back pointers is as follows:

```
LCS(V,W)
for i = 1 to n do
   s[i,0] := 0;
for j = 1 to m do
   s[0,j] := 0;
for i = 1 to n do
   for j = 1 to m do
      if v[i] = w[j] then
         s[i,j] := s[i-1,j-1] + 1;
         b[i,j] := [diag];
      else if s[i-1,j] >= s[i,j-1] then
         s[i,j] := s[i-1,j];
         b[i,j] := [up];
      else
         s[i,j] := s[i,j-1];
         b[i,j] := [left];
return s and b.
```