HKU ID: _____

THE UNIVERSITY OF HONG KONG
Faculty of Engineering
Department of Computer Science

COMP 9601 Theory of Computation and Algorithms Design

Date: Dec 9, 2014 Time: 2:30 PM - 4:30 PM.

Only approved calculators as announced by the Examinations Secretary can be used in this examination. It is candidates' responsibility to ensure that their calculator operates satisfactorily, and candidates must record the name and type of the calculator used on the front page of the examination script.

Answer all questions. The question whose answer has the lowest score will be given a half weight (14.2%). and each other question will be given a full weight (28.6%).

Give your answers on this question paper. Do not use an answer book.

1. Let $A = a_1 a_2 \cdots a_n$ and $B = b_1 b_2 \cdots b_n$ be two length-n DNA fragments terminated by a special symbol "\$" (i.e., $a_n = b_n =$ "\$"). Assume that the second half of A is identical to the second half of B, i.e., $a_{n/2+1} = b_{n/2+1}$, $a_{n/2+2} = b_{n/2+2}$, $\cdots$, $a_n = b_n$. We want to index A and B based on all the suffixes of A and the suffixes of B that start from its first half. There are a total of $n + n/2$ suffixes. Suppose we sort all these suffixes (in lexicographic order) into a table $T[1, n+n/2]$, in which each entry is a suffix in the form of $A[i..n]$ or $B[j..n]$. Based on T, show how to define the compressed suffix array $\Psi[i]$, where $1 \leq i \leq n + n/2$, and show how to use Ψ (but without T) to report whether a given pattern P exists in A or B in $O(|P| \log n)$ time. Your answer may make use of an auxiliary array *count* which stores the total number of times each character occurs in A and the first half of B.

   Answer:

2. Let A be a one-state pushdown automaton that has two kinds of stack operations: (1) Pop: A reads and removes the top element z from the stack, and then depending on its current state, next input symbol a, and z, A makes a move which can push zero or more elements back to the stack. (2) Pop & peek: A reads and removes the top element z from the stack, and reads (without removing) the next top element z'; then A makes a move depending on its current state, next input symbol a, z and z', and may push zero or more elements back to the stack (on top of z'). Note that A has only one state and cannot memorise any stack symbol using the state information. Also, pushdown automata, by definition, are non-determinstic in nature.

   This question is concerned with how to construct another one-state pushdown automaton B using only pop operations to simulate A. Let q be the only state in A, and let Γ and δ be the stack alphabet and transition function of A, respectively. A plausible approach is to let B use a bigger stack alphabet $\Gamma' = \Gamma \times \Gamma$; such a compound stack symbol represents a stack entry of A and an extra copy of the entry below it. For example, if A's stack contains three symbols, say, $z_3 z_2 z_1$, where z_1 is at the top, B's stack contains $(-, z_3)(z_3, z_2)(z_2, z_1)$. Based on this approach, show how to define B's transition function δ' to simulate a move of A in each of the following four cases.

   Pop: (i) $\delta(q, a, z)$ contains $(q, z_1 z_2 \cdots z_k)$ for some $k \geq 1$; (ii) $\delta(q, a, z)$ contains (q, ϵ) where ϵ denotes the empty string;

   Pop and peek: (iii) $\delta(q, a, z', z)$ contains $(a, z_1 z_2 \cdots z_k)$ where z and z' denote the pop and peek symbol, respectively (i.e., z will be removed from the stack, and z' will stay); and (iv) $\delta(q, a, z', z)$ contains (q, ϵ).

   Answer:

3. Suppose that there is a Turing machine T when, starting with an empty tape, will generate an infinite sequence of C++ programs $C_1, C_2, \cdots$. Let L denote this set of C++ programs.

   Note that L can be recognizable by a Turing machine M which, given any program C, runs T until it has an output C_i equal to C. If C is in L, M will halt and accept C; otherwise M will not halt.

   Show how to modify each program C_i in L to another program C_i' such that C_i and C_i' have the same functions and the set L' of all the modified programs $C_1', C_2', \cdots$ is decidable. Justify your answer. [Hint. Make $C_1', C_2', \cdots$ strictly increasing in size].

   Answer:

   First of all, as David mentioned, undecidability is a property of a language, not a Turing machine.

   As for your question, the hint kind of gives it up. Note that we only need to recognize the programs C_i, not programs who are equivalent (which would be undecidable), so were only dealing with syntax, not semantics.

   Let C_i' be the Turing machine which on input x operates the same way as C_i, but with enough additional states (which were not going to use, i.e. they will be unreachable from the initial state), such that the length of the encoding $|C_i|$ will satisfy $\forall j < i : |C_j| < |C_i|$. In that case, to decide L', given an encoding of a Turing machine $\langle M \rangle$, run the machine T' which generates C_i'. If you meet the program $\langle M \rangle$ accept, if you pass the length of the encoding of M (i.e. you reached i such that $|C_i| > |\langle M \rangle|$) halt and reject. The difference here is that if we reject, we can be sure we will never meet the encoding of M, since C_i is a series of programs with strictly increasing encoding length.

4. This question is concerned with courier pick-up service in HK Island. There are $k \geq 2$ pick-up vans, initially all located in the CYC Building of HKU. Consider a sequence of requests $r_1, r_2, \cdots, r_n$, where each r_i is a building in HK Island. To serve a request r_i, one of the pick-up vans must visit the required building r_i. Given a sequence of requests, we want to minimize the total distance traveled by the vans. Show that the greedy algorithm, which moves the nearest van to serve a request, has unbounded competitive ratio.

Answer:

— END OF PAPER —