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

COMP 9601 Theory of Computation and Algorithms Design

Date: Dec 18, 2012 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 any THREE questions. All questions carry equal weight. If answers to more than three questions are given, the best three will be used to determine the final score.

1. (Decidability) It is known that the problem of determining whether the language of any given Turing machine is empty is undecidable.

   Consider the problem of determining whether the language of any given context free grammar (or pushdown automaton) is empty. Is it decidable? Justify your answer.

2. (NP-completeness) A CNF formula is a Boolean formula in the form of a conjunction of clauses, where each clauses is a disjunction of variables or their negations. The problem $\frac{1}{3}$-CNF-SAT is to determine whether a given CNF formula has a satisfying assignment in which at most one third of the variables are true. Prove that it is NP-complete.

3. (Text indexing) Let $P_1, P_2, \ldots, P_n$ be distinct strings (patterns) of length m. Suppose that a generalized suffix tree for these n strings has already been built.

   a. Show how to use this tree to determine, for any given string T of length greater than m, whether T contains more than three distinct P_i's. Your algorithm should run in $O(|T|)$ time (state clearly your assumption of the suffix tree). Note that a brute force way would use $O(|T|m)$ time.

   b. Suppose that the patterns $P_1, P_2, \ldots, P_n$ are of length either m or $m/2$, and it is possible that a pattern P_i is a prefix of another longer pattern P_j. What extra information would you store into the suffix tree so that the above problem can still be solved in $O(|T|)$ time?

4. (Online scheduling) This problem is concerned with online algorithms for scheduling jobs with deadlines on a single processor. Jobs arrive at unpredictable times. When a job arrives, its size (i.e., processing time) and deadline are known. It is assumed that the release time, size and deadline are all "integers". Jobs released may overload the system in the sense that no schedule can meet the deadlines of all jobs. The objective is to devise a schedule that maximizes the **number** of jobs completed by their deadlines. Recall that

EDF refers to the online strategy of executing the job with the earliest deadline (ties broken by smaller job ID).

It is known that in the overloaded setting, if jobs are of different sizes, then EDF is not c-competitive for any constant c. Prove that if jobs all have size exactly 1 unit, then EDF is optimal (i.e., 1-competitive). (Hint. EDF can complete all jobs in the underloaded setting. In the overloaded setting, any schedule that can complete a subset I of jobs can be transformed to an EDF schedule for I).

— END OF PAPER —