CS 332: *Elements of Theory of Computation* — Professor Sofya Raskhodnikova
Boston University — November 14, 2019

Homework 9 – Due Friday, November 22, 2019 before noon

This homework contains 4 mandatory problems, worth 10 points each.

Reminder Collaboration is permitted, but you must write the solutions *by yourself without assistance*, and be ready to explain them orally to the instructor if asked. You must also identify your collaborators. Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.

Problems

1. (**Closure properties of P**) For both parts of this problem, analyze the running time of your algorithms using O-notation. Prove that P is closed under

   (a) (**3 points**) concatenation;

   (b) (**7 points**) star.
   Hint: Use dynamic programming. On input $y = y_1 \cdots y_n$ for $y_i \in \Sigma$, build a table indicating for each $i \leq j$ whether the substring $y_i \cdots y_j \in A^*$ for any $A \in$ P.

   Think about union and complement on your own.

2. (**NP**) In parts (a) and (b), you are asked to prove that the given languages are in NP. For one of them (of your choice), give a verifier; for the other, a nondeterministic TM.

   (a) You would like to schedule final exams to ensure that there are no conflicts. You are given k class lists, where each class list is a set of ID numbers of students taking the class. You would like to determine if t time slots are *sufficient for a conflict-free schedule*; that is, whether there exists a partition of k classes into t sets, such that classes in each set have disjoint class lists. For example, if student IDs are integers 1 to 4, and you are given three class lists $L_1 = \{1, 2, 3\}$, $L_2 = \{1, 4\}$, and $L_3 = \{2, 3\}$, then 2 time slots are sufficient: classes 2 and 3 can be scheduled together, while class 1 is allotted the second time slot.
   We formalize this problem as a language as follows: $SCHEDULE = \{\langle L_1, L_2, \ldots, L_k, t\rangle \mid L_1, \ldots, L_k$ are class lists and t time slots are sufficient for a conflict-free schedule$\}$.
   Show that $SCHEDULE \in NP$.

   (b) Recall the language POST (about a puzzle with dominos) from the lecture on the computational history method. (This problem is called PCP in Sipser, Chapter 5.2). Let kPOST $= \{\langle S, k\rangle \mid S$ is a finite set of dominoes over Σ; k is an integer written in unary, and there is a sequence of at most k dominoes (allowing repeats) for which the top and bottom sequences are equal$\}$.
   Prove that kPOST is in NP.

   (c) If k was written in binary, would your solution to part (b) still work? Why or why not?

3. (**Search vs. decision**) You would like to ensure that a system you are building has the set U of n capabilities. You have m available software packages. The ith software package provides the set $S_i \subseteq U$ of capabilities. You want to achieve all n capabilities using the smallest number of packages.

In this problem, you will prove that if P=NP, you can *find* the smallest set of software packages in polynomial time.

(a) Consider the following language:
$SOFTWARE = \{\langle n, U, S_1, \ldots, S_m, k\rangle \mid$ there is a collection of k software packages such that the union of their sets of capabilities is equal to $U\}$. Prove that $SOFTWARE \in$ NP.

(b) **If** P=NP, the proof you just gave for part (a) implies that $SOFTWARE \in P$, that is, in polynomial time you can decide whether k packages are enough. Assume you have a subroutine that does it, and use it repeatedly to *find* the smallest set of software packages that achieve all n capabilities in polynomial time.
Hint: Similar to problem 7.40, solved in the book.

4. (**Student representatives**) You are given a collection C of rosters for all clubs in a university and a positive integer k. Each roster lists all students that are members of the club. Is there a set of students S of size at most k such that S contains at least one student from each club?

   (a) Formulate this problem as a language REP.
   (b) Show it is in NP.
   (c) Give a polynomial time reduction from VERTEX-COVER to REP.
   (*Careful:* make sure you are doing it in the direction specified.)