Final

Wednesday, 3 May 2017
4:30 PM

This is a take home final. It will be released to all students and be due at the date and time given above.

You are to work independently, you can reference materials, but be careful if your answers too closely repeat publicly available sources, or I will not be able to evaluate your own grasp of the materials.

The problems contain enough depth that it is always possible, regardless of the materials referenced, for you to demonstrate personal competence on the tested material.

Name: ______________________________

Problem	Credit
1	
2	
3	
4	
5	
Total	

1. The SATISFIABILITY problem is, given a collection of clauses, such as,

$$\{\,(x_1 \vee x_2 \vee \overline{x_3} \vee x_4), (\overline{x_2} \vee x_3 \vee x_1), (\overline{x_1} \vee \overline{x_4})\,\}$$

assign values to all x_i such that each clause evaluates true.

This problem reduces to 3-SAT, where each clause has exactly three literals.

(a) Show how a single clause that is not in the proper 3-SAT form is made by the reduction into several clauses each in proper 3-SAT. In particular, show the reduction for the following clause:

$$x_1 \vee x_2 \vee \overline{x_3} \vee \overline{x_4} \vee x_5 \vee x_6$$

Name by y_i any new variables introduced.

(b) Why does your reduction work? Why is the reduced collection of clauses simultaneously satisfiable if and only if the original clause is satisfiable?

2. The book gave the following reduction from 3-SAT to VERTEX-COVER: For each variable x, create a "variable widget" of two nodes x and $\overline{x}$ connected by an edge. For each clause $x \vee y \vee z$ create a "clause widget" of three nodes x, y and z connected in a triangle. Connect each node in a clause widget with a node in a variable widget according to the variable or negation of variable that the nodes represent. Set $k = n + 2m$, where n is the number of variables, and m is the number of clauses.

   Draw the resulting VERTEX-COVER instance of the following 3-SAT instance and find a vertex cover. Give the satisfying assignment implied by that cover.

$$\{\, (x_1 \vee x_2 \vee x_3),\ (\overline{x_1} \vee \overline{x_2} \vee x_3),\ (x_1 \vee \overline{x_2} \vee \overline{x_3}),\ (\overline{x_1} \vee x_2 \vee x_3),\ (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \,\}$$

3. Addition of two positive integers takes $O(k)$ time by the standard addition algorithm, with k the total length of the representation in binary of the integers.

   The following recursive function M defines multiplication,

$$M(x,y) = \begin{cases} x & \text{for } y = 1 \\ 2 * M(x, y/2) & \text{for } y \text{ even} \\ x + 2 * M(x, (y-1)/2) & \text{for } y \text{ odd and not } 1 \end{cases}$$

   for positive integers x and y.

   For instance,

$$M(6,5) = 6 + 2 * M(6,2) = 6 + 2 * 2 * M(6,1) = 6 + 2 * 2 * 6 = 30.$$

   Show that multiplication is in P-time by analyzing this algorithm. The run time will be in Big-Oh notation as a function of k, the total length of the representation in binary of the numbers.

4. The set of Turing Machines that accept nothing, E_{TM}, is undecidable.

   (a) If a set is undecidable, can it be both R.E. and co-R.E.? Give proof.

   (b) If a set is undecidable, can it be neither R.E. and co-R.E.? Give an example.

   (c) Is E_{TM} R.E., co-R.E., or neither? Give proof.

5. Given an NP problem A, there is a non-deterministic TM such that if a is in the language then there is at least one computation path leading to accept, but if a is not in the language then no computation path leads to accept.

   An important class of problems inside NP is *Randomized Polynomial Time*, or RP. A problem is in RP if for a in the language then a proportion of at least 1/100 of all computation paths are paths leading to accept, and for a not in the language, there are no computation paths leading to accept.

   A *Randomized Turing Machine* is a deterministic Turing Machine that in addition to the tape contents to guide the transitions, can flip a coin and use the coin outcome to guide the transitions. The output of these machines has an associated probability, which is the probability that the coin flips will lead the computation to that output.

   Prove the following:

   > For all problems in RP, there is a Randomized Turing Machine such that, if a is not in the language, the machine rejects always, if a is in the language, the machine accepts with probability $1 - \epsilon$ and and rejects with probability ϵ, for any real $\epsilon > 0$.

   While the class RP seems strange, it has practical significance. RP problems are those NP problems for which one can trade not being able to know the answer with being able to know the answer, but there is a slim chance that the answer is wrong.