CSC 363, Winter 2010 — Short Assignment #3

Because of the revisions to this assignment below, and because I had to cancel my office hours, I've extended the due date on this assigment to start of tutorial on March 10. Of course, the answer now won't be gone over on March 3!

The paper version handed out had some typos, corrected below. Also, the model of computation to be used when finding running times wasn't made clear. You may use k-tape Turing machines if you wish. Note that there's no requirement that your reduction in Question (1) be optimal, as long as it runs in polynomial time. You may also be a bit loose on your running time bounds (eg, say $O(n(logn)^2)$ when the running time is also $O(nlogn)$).

Due (not any more, see above) at ***start*** *of tutorial time on March 3. Late assignments will* ***not*** *be accepted, as the tutorial instructors will immediately go over the solution. I prefer that you hand the assignment in on paper, but if you need to, you can email it (as a plain text file or PDF attachment, no Microsoft Word files please) to `radford@cdf.utoronto.ca`. (Please use this email address only for assignment submission.)*

This assignment is to be done by each student individually. You are encouraged to discuss the course material in general with other students, but you should not discuss this assignment (verbally, in writing, by email, or in any other way) with people other than the course instructor and tutors, except to clarify the meaning of the question. Handing in work that is not your own is a serious academic offense.

The text defines the VERTEX-COVER problem (on page 284), which is to decide the language

$$\text{VERTEX-COVER} \;=\; \{\, \langle G,k\rangle \,|\, G \text{ is an undirected graph that has a } k\text{-node vertex cover}\,\}$$

A "vertex cover" for an undirected graph is a set of nodes that includes at least one of the two nodes adjacent to every edge in the graph. For instance, for the graph a — b — c, the sets $\{b\}$, $\{a,c\}$, $\{a,b\}$, $\{b,c\}$, and $\{a,b,c\}$ are vertex covers, but the sets $\{a\}$ and $\{c\}$ are not. The encoding for an undirected graph G is assumed to be something reasonable. Here let's assume it consists of the number of nodes m of the graph, in standard binary notation, and a list of pairs of nodes, with nodes identified by numbers from 1 to m, in standard binary notation. For instance, the graph above would be encoded as a string something like this: 11,(1,10),(10,11).

Now let's define the NOADJ-SQUARE problem as follows. Consider a q by q square of cells, with each cell being either white or black. For a given k, we wish to know whether or not it is possible by changing k of the white cells to grey to arrange that there are no pairs of cells that are adjacent to each other either horizontally or vertically and that are both still white. For example, the white(+) and black (@) square on the left below can be changed to the white (+), black (@), and grey (%) square on the right, in which $k = 11$ cells have been changed from white to grey to satisfy the no adjacent white cell requirement.

```
+ + + @ + +        + % + @ + %
+ + + + @ @        % % % + @ @
+ + + @ + +        + % + @ + %
@ @ @ @ @ +        @ @ @ @ @ +
+ + @ @ @ @        % + @ @ @ @
+ + + + + +        + % + % + %
```

Suppose we encode an instance of the NOADJ-SQUARE problem as the q^2 symbols ("+" for white and "@" for black) in the cells, ordered left-to-right and then top-to-bottom, followed by the number k in standard binary notation. (Note that there is no need to represent q explicitly, since it is implied by the number of "+" and "@" symbols.) We are then interested in deciding the language

$$\begin{aligned}\text{NOADJ-SQUARE} \;=\; \{\, \langle S, k\rangle \mid S &\text{ is a square in which the no adjacency condition} \\ &\text{can be satisfied by changing } k \text{ white cells to gray}\,\}\end{aligned}$$

1) Show that NOADJ-SQUARE is polynomial time reducible to VERTEX-COVER, by describing an algorithm that does the reduction at a high level (but with enough detail that its running time is clear). Give the best $O(\cdot)$ bound on the running time of this algorithm that you can. Don't ignore logarithmic factors.

2) Suppose that VERTEX-COVER is decidable in time $O(n^9)$ on a deterministic Turing Machine (ie, VERTEX-COVER $\in$ TIME(n^9)). Using your reduction in part (1), give the best bound you can on the time needed to decide NOADJ-SQUARE.

3) Suppose that a polynomial time algorithm for NOADJ-SQUARE is found. Would that imply that a polynomial time algorithm for VERTEX-COVER exists?

4) Consider the language SPARSE-NOADJ-SQUARE, which is the same as NOADJ-SQUARE except that the square is encoded by giving its size, q, in standard binary notation, followed by a list of coordinates of cells that are black. The coordinates of a cell are a pair of integers from 1 to q in standard binary notation. If you adapt your reduction in part (1) to SPARSE-NOADJ-SQUARE, what is its running time?