CSC 363, Winter 2010 — Long Assignment #2

Due at ***start*** *of lecture on March 31. You may hand in up to* ***one*** *long assignment up to* ***one*** *day late without explanation (put it under my door in PT209E or SS6026A by 7pm). Other late assignments will be accepted only with a valid excuse (presented as soon as possible). 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.

Question 1: The book defines (p. 193) a Linear Bounded Automaton (LBA) as a one-tape Turing machine in which the tape head is not allowed to move past the end of the input string. Since the tape alphabet is allowed to be larger than the input alphabet, the effective amount of memory available to an LBA can be any constant times the length of the input string.

Consider a machine which is an LBA with an additional work tape, for which the head can be moved left or right as for an ordinary Turing Machine, but for which the symbol on a tape square can be changed only if the current symbol is blank. In other words, the only write operations allowed are ones that append a non-blank symbol to the end of the non-blank part of the work tape. (Arbitrary write operations are allowed on the input tape, as for an LBA, within the range of the squares originally occupied by the input string.)

Prove that any language that is decidable in polynomial time on an ordinary deterministic Turing Machine is decidable in polynomial time on a deterministic machine of the sort described above.

Question 2: The book defines (p. 269) the SUBSET-SUM language as follows:

$$\text{SUBSET-SUM} \;=\; \{\langle S,t\rangle \mid \; S=\{x_1,\ldots,x_k\} \text{ and for some } \{y_1,\ldots,y_l\}\subseteq S \text{ we have } \textstyle\sum y_i = t\}$$

where $x_1,\ldots,x_k$ and t are positive integers, with standard binary encoding, and $\{x_1,\ldots,x_k\}$ and $\{y_1,\ldots,y_l\}$ are "multisets", in which repetitions are allowed.

For example, $\langle\{12,1,5,1,1,12\},\ 7\rangle$ is in SUBSET-SUM, because $5+1+1=7$, but on the other hand $\langle\{12,1,5,1,1,12\},\ 11\rangle$ is not in SUBSET-SUM.

For this question, consider the language SUBSET-SUM-NOREP, defined in the same way as SUBSET-SUM, except $\{x_1,\ldots,x_k\}$ and $\{y_1,\ldots,y_l\}$ are ordinary sets, in which repetitions are not allowed. Prove that SUBSET-SUM is polynomial time reducible to SUBSET-SUM-NOREP.

Question 3: Consider the language SUBSET-SUM-POW2 defined the same as SUBSET-SUM except that the integers $x_1,\ldots,x_k$ and t are not represented in standard binary notation. Instead, a positive integer is represented as a list $(b_1,\ldots,b_m)$ of non-negative integers, in standard binary notation, with $b_i<b_j$ if $i<j$, representing the number $2^{b_1}+\cdots+2^{b_m}$. For example, the integer 6 is represented as $(1,10)$, since $6=2^1+2^2$.

Prove that SUBSET-SUM-POW2 is NP-complete. Remember: You need to show *two* things to show that a language is NP-complete (see definition 7.34 on p. 276).