Exercises for Computability and Complexity, Spring 2019, Sheet 2 -- Solutions

Please return in class on Tuesday Feb 19

Exercise 1 Show that $L = \{w \in \{1\}^* \mid |w| \text{ is a power of } 2\} \in$ **TIME**($O(n \log n)$), by describing in words (and maybe sketches of interesting configurations) a TM (with possibly several tapes) that does this job.

Solution. Set up a 2-tape TM, as follows. The first tape contains the input word, is read-only, and the cursor here never moves left. While the first cursor moves right, on the second tape a binary-coded count of the number of 1's visited is constructed. Whenever the first cursor moves to the right, the count on tape 2 is updated (which may take some operations where the first cursor does not move). The update is a combination of the add-1 and shift-right, single-tape TMs from the lecture notes, which per add-1 operation may require 2 full back-and-forth traversals of the word b written on tape 2 up to that point, that is, 4 |b| TM cycles. When the last 1 on tape 1 has been processed, our TM enters a final round of checking whether the 2nd tape word b is of the form 10....0. If yes, the input is accepted, if no, not. This final check can be clearly effected in another $|b|$ steps. Since $|b| \leq \log_2(|w|)$, we find that our TM uses at most $\log_2(|w|)(4\, n) + \log_2(|w|) = O(n \log n)$ steps.

Exercise 2 (a) Are the functions $f(n) = \exp(n)$ and $g(n) = \exp(2n)$ polynomially related? **(b)** What about $f(n) = \exp(n)$ and $g(n) = \exp(n^2)$? Prove your answers.

Solution. (a) Yes, by the quadratic polynomial $p(n) = n^2$. We clearly have $f(n) \leq p(g(n))$, and conversely, $g(n) = (\exp(n))^2 = p(f(n))$.
(b) No. Assume there were a polynomial $p(n) = n^a$ such that $g(n) = \exp(n^2) = \leq p(f(n)) = \exp(na)$. Then for $m > a$, we would have $g(m) = \exp(mm) \geq \exp(ma)$, contradiction.

Challenge problem (***optional***) Let $\Sigma_n = \{1,..., n\}$ and $L_n = \{12...n\}$ (i.e. the language that contains only the word 12...n). Prove or disprove: a single-tape TM deciding L_n must have at least n states.

One solution (by Corneliu-Claudiu Prodescu, copied here verbatim; simpler solution sketched at end):

I believe the # of states is actually constant (with respect to n). Here is my sketch of a proof. I might have some slip overs, but I think the main argument is right. The MAIN idea:

Step 1: Test if the last non-empty slot of the tape contains "n" (easily 2 states).
Step 2: Go through the tape from end of input to the beginning of the tape, checking if adjacent entries are consecutive (i.e. | X |X+1|)).
Step 3: One reached |_>_| X |, check if X = 1, then YES, else NO.

Also, if any of the Steps 1 or any point of 2 fails, a halt with NO is implied.

LEMMAS:
I'll state a few actions, each of which can be done by a constant (with respect to "n") number of states.

Lemma 1:

We can shift a cell item left or right by one cell (the acceptor cell was empty before).
Ex: | X |___| => |____| X |

Dem:
We'll use a decrement and an increment state to basically move X one by one. Here is a sketch of the transitions of these states (We start in DecrState and finish in "Some next state"):

IncrState:
- symbol k => k+1, L, DecrState
- symbol ___ => 1, L, DecrState

DecrState:
- symbol k => k - 1, R, IncrState
- symbol 1 => ___, R, Final_IncrState

Final_IncrState:
- symbol k => k+1, L, Some next state
- symbol ___ => 1, L, Some next state

Lemma 2:
We can compare adjacent values for "consecutivity"
Ex: | X | Y | => |___|___|, if Y = X + 1
=> Halt with NO otherwise

Dem:
We'll use a state to increment X (if it is "n" we halt with NO) and then start a Decrement left, Decrement right race (using 2 states) until one (or both) are blank and proceed accordingly.
We are going to use some additional states to couple the events, but this will clearly be still constant.

Lemma 3:
We can do a "double move" of a cell value into two adjacent empty cells
Ex: | X |___|___| => |___| X | X |

Dem:
This will be similar to Lemma 1, just that we'll use 1 decrementer, 2 incrementers and another dummy state to bring back the cursor to position 1. An additional number of 2 states may be necessary to bring back the cursor to pos 1 in the end, but the number remains constant.

BACK to the MAIN IDEA:

Phase 1 is easy:
One state walks blindly until an ___ is encountered and then moves one Left and goes to state 2.
State 2 halts with NO if the input is not "n" and otherwise the cursor is moved to the Right and phases 2-3 start.

Phase 2 and 3 will be described using the lemmas:
We will be generally in the position:
... | X | Y |___|___| ....

^

and first we use 4 states to go to the cell with X and check if it is _>_. If it is indeed _>_, we have reached the beginning of the tape and we only need to check phase 3. We move to Y check if it is 1, halt with YES or NO accordingly.

Now, if it is not _>_, we want to check if X and Y are consecutive and then somehow reduce Y. Here is what will happen on the tape, during a few sets of moves (a move will actually be a lemma usage)

```
    ... | X | Y |___|___| ....   => (1 - move Y right)
    ... | X |___| Y |___| ....   => (1 - move Y right)
    ... | X |___|___| Y | ....   => (3 - double move X right)
    ... |___| X | X | Y | ....   => (1 - move X left)
    ... | X |___| X | Y | ....   => (1 - move X left)
    ... | X | X |___| Y | ....   => (1 - move Y left)
    ... | X | X | Y |___| ....   => (2 - compare X and Y for
"consecutivity")
    ... | X |___|___|___| ....  (on success of Y = X+1)
```

and we continue the recursion, as desired.

Using a main idea of this proof, another (shorter) proof comes to mind: run the cursor backwards and forward across the input (set a delimiter at its end in a preparation phase), decrementing every symbol found by 1 at each pass. If symbol 1 is read, decrement to empty cell symbol. Fail if in any of these passes a condition different from "find only empties at beginning, followed by non-empty cells, followed by right delimiter" is encountered, else accept. If I remember correctly, this solution was suggested by Josip Djolonga.