

Exercises for Computability and Complexity, Spring 2017, Sheet 3 – Solutions

Please return your solutions in class, in the Thursday lecture on February 23

Exercise 1. Consider the ultra-simple TM M with tape alphabet $\Sigma = \{0, 1, \sqcup, \triangleright\}$ and states $\{s, \text{yes}, \text{no}\}$ that has the following transition table:

$p \in K$	$\sigma \in \Sigma$	$\delta(q, \sigma)$
s	0	(yes, 0, \rightarrow)
s	1	(s , 1, \rightarrow)
s	\sqcup	(no, \sqcup , \rightarrow)
s	\triangleright	(s , \triangleright , \rightarrow)

What is the language $L(M)$ decided by M ? Describe that language in plain English. Write a RAM program that decides the same language, in the following sense. Your RAM should compute a string function $f: \Sigma^* \rightarrow \{0,1\}$, such that $f(w) = 1$ iff w is in $L(M)$.

Solution. M decides the language of all words that contain at least one 0.

```

1   read
2   if c(0) = 0 goto 5
3   if c(0) = 1 goto 1
4   if c(0) =  $\sqcup$  goto 7
5   print 1
6   end
7   print 0
8   end

```

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

One solution (by Corneliu-Claudiu Prodescu):

ow, 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.

Here is my proof:

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: $| X _ | \Rightarrow _ | X |$

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 \Rightarrow k+1, L, DecrState

- symbol $_$ \Rightarrow 1, L, DecrState

DecrState:

- symbol k \Rightarrow k - 1, R, IncrState

- symbol 1 \Rightarrow $_$, R, Final_IncrState

Final_IncrState:

- symbol k \Rightarrow k+1, L, Some next state

- symbol $_$ \Rightarrow 1, L, Some next state

Lemma 2:

We can compare adjacent values for "consecutivity"

Ex: $| X | Y | \Rightarrow _ | _ |$, if $Y = X + 1$
 \Rightarrow 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: $| X _ | \Rightarrow _ | X | X |$

Dem:

This will be similar to Lemma 1, just that we'll use 1 decremter, 2 incremterers 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.

