Writing more Turing Machine programs

11/5/18

Adding numbers

we'll ~~add in~~ represent numbers in unary - n is rep. by n 1's.

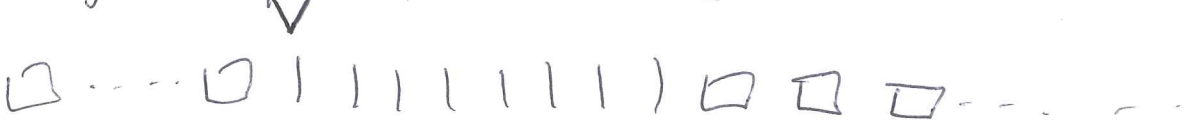
I want my TM to:

take as input
2 pos. integers ~~numbers~~ in unary, seperated by a 0.

output - the sum of the numbers (between lots of □'s, with nothing else)

3 + 5

my output should be



8

Thinking of this as shifting numbers, I can ~~erase~~ replace the 0 w/ 1, and replace a 1 on the end w/ a blank. (assume I want the machine to be over the left of output - then it's easier to erase the 1 on the left)

Conventions: We're given the machine starts at the beginning of the input and the input is well-formed.

$(q_0, 1) \longrightarrow (q_1, \square, R)$

$(q_1, 1) \longrightarrow (q_1, 1, R)$

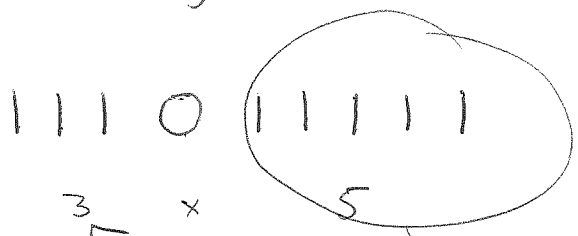
$(q_1, 0) \longrightarrow (q_2, 1, L)$

$(q_2, 1) \longrightarrow (q_2, 1, L)$

$(q_2, \square) \longrightarrow (q_3, \square, R)$

$(q_3, 1) \longrightarrow$ ~~?~~

Multiplying 2 numbers:



probably easiest not to overwrite input.

~~erase~~ erase one of these 1's and copy one ~~block~~ block of 5, then erase another and copy another block of 5 next to it, etc.

$(q_0, 1) \longrightarrow (q_1, \square, R)$

$(q_1, 1) \longrightarrow (q_1, 1, R)$

$(q_1, 0) \longrightarrow (q_2, 0, R)$

$(q_2, 1) \longrightarrow (q_3, a, R)$

$(q_3, 1) \longrightarrow (q_3, 1, R)$

$(q_3, \square) \longrightarrow (q_4, \square, R)$

$(q_4, \square) \longrightarrow (q_5, 1, L)$

$(q_4, 1) \longrightarrow (q_4, 1, R)$

q_1 - we've erased and need to copy

q_2 - we ~~are~~ start copying

q_3 - ~~copying~~ moving to right while ~~copying~~ while in input

q_4 - moving to right while in output

$(q_5, 1) \longrightarrow (q_5, 1, L)$

$(q_5, \square) \longrightarrow (q_6, \square, L)$

$(q_6, 1) \longrightarrow (q_6, 1, L)$

$(q_6, a) \longrightarrow (q_2, a, R)$

$(q_2, \square) \rightsquigarrow$ replace all the a's w/ 1's, then move to beginning of input.
if we see a 1, repeat
if we see a 0, blank out the input, move to beginning of output.

Check if a (unary) number is prime:

1. Copy the number ~~twice~~. once
2. Subtract 1 from rightmost copy.
3. Divide ~~middle~~ left copy by rightmost, find remainder.
4. If ~~the~~ remainder is 0, halt, say no
5. If the remainder is not 0, erase remainder, go back to 2
6. If rightmost copy has become 1, halt, say yes

Trying to make more ~~Turing~~ powerful Turing Machines:

We could have a TM with a stay option instead of ~~move~~ being force to move L/R.

- we could ~~cepta~~ replicate this functionality in a standard TM by just having extra states ~~and~~ which remember to move back and go to a certain state (i.e. - instead of staying put and going to state 17, move right and go to state 17', from which all you do is move left and go to state 17)

TM with several tapes (but moving simultaneously on all of them)

- replicate this w/ ~~a~~ a tape TM by making the tape alphabet

$$\Gamma_1 \times \Gamma_2 \times \cdots \times \Gamma_k \quad (k \text{ tapes}),$$

Γ_i = ~~alphabet~~ tape alphabet for k-th tape.