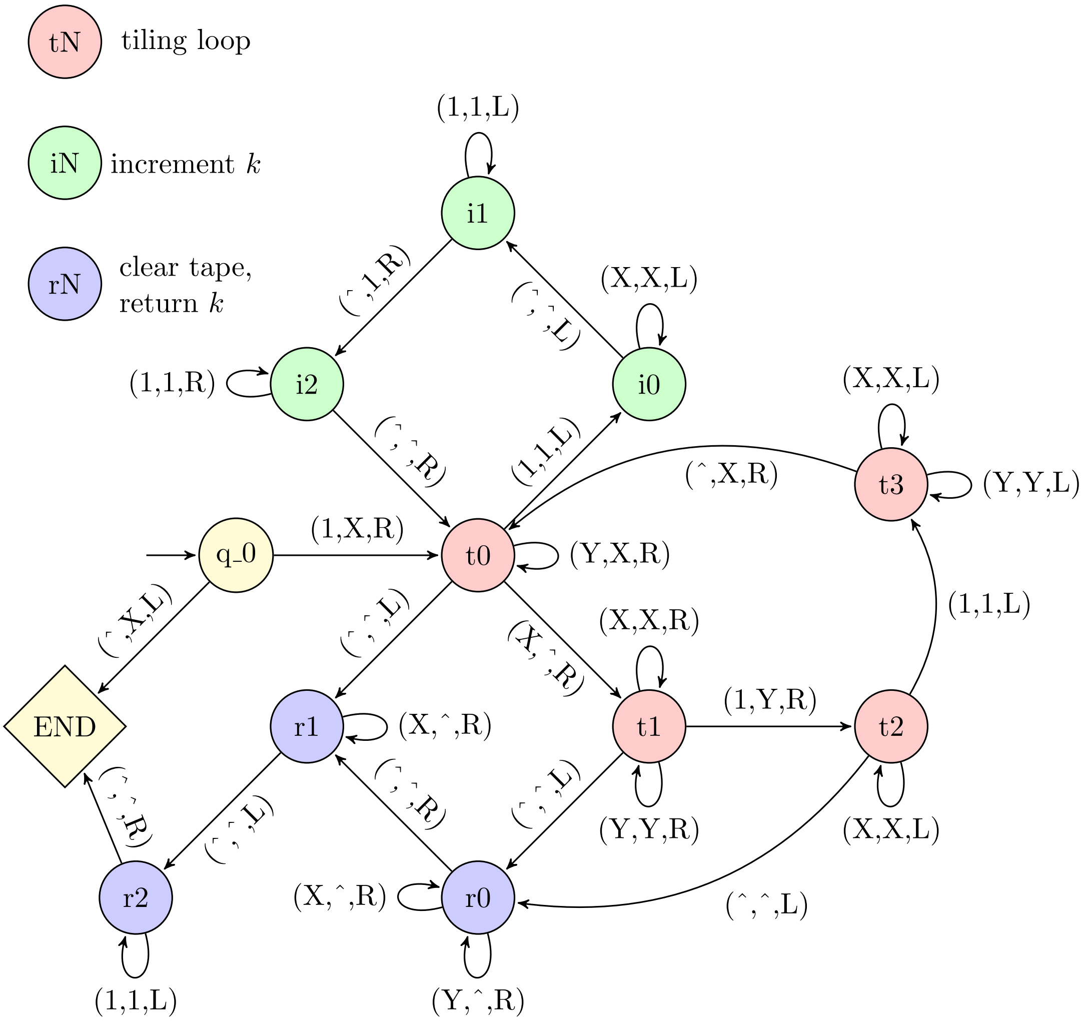
Logarithm Turing Machine Individual Coursework, F29FB, Spring2021, H00302342

1.



2.

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	ˆ	ˆ	1	1	1	ˆ
				∧ q_0			

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	ˆ	ˆ	X	1	1	ˆ
					∧ t0		

The leftmost 1 of the input is replaced with an X so that tiling can begin. The head moves right to see if there are more 1s (if there weren't it would delete the X and return 0).

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	ˆ	ˆ	X	1	1	ˆ
				∧ i0			

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	ˆ	ˆ	X	1	1	ˆ
			∧ i0				

Since there are more 1s to the right of the X, the head changes state to i0 (i0 is the first of the iNUM states, denoting the group of states in charge of incrementing k). The head passes the X and the first ˆ to the left of the input.

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	ˆ	ˆ	X	1	1	ˆ
		∧ i1					

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	1	ˆ	X	1	1	ˆ
			∧ i2				

The second ˆ is encountered, and a 1 is written in its place. The head starts moving to the right again.

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	1	ˆ	X	1	1	ˆ
				∧			
				t0			

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	1	ˆ	ˆ	1	1	ˆ
					∧		
					t1		

An X is encountered and the head changes state to t0 (t0 is the first of the tNUM states, denoting the group of states in charge of tiling the input with X). The X is erased and the head moves right until it encounters a 1.

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	1	ˆ	ˆ	Y	1	ˆ
						∧	
						t2	

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	1	ˆ	ˆ	Y	1	ˆ
					∧		
					t3		

The encountered 1 is replaced with a Y and the head moves right to check if the end of the input has been reached. Since that's not the case, it moves left again.

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	1	ˆ	ˆ	Y	1	ˆ
				∧			
				t3			

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	1	ˆ	X	Y	1	ˆ
					∧		
					t0		

The head keeps moving left until the first ˆ is encountered and writes an X in its place.

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	1	ˆ	X	X	1	ˆ
						∧	
						t0	

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	1	ˆ	X	X	1	ˆ
					∧		
					i0		

The Ys are replaced by Xs and the head moves right to check if the end of the input has been reached.

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	1	ˆ	X	X	1	ˆ
				∧			
				i0			

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	1	ˆ	X	X	1	ˆ
			∧				
			i0				

Since that was not the case, the head switches to the i0 state and increments k again.

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	1	ˆ	X	X	1	ˆ
		∧					
		i1					

-4	-3	-2	-1	0	1	2	3
ˆ	ˆ	1	ˆ	X	X	1	ˆ
	∧						
	i1						

-4	-3	-2	-1	0	1	2	3
ˆ	1	1	ˆ	X	X	1	ˆ
		∧					
		i2					

-4	-3	-2	-1	0	1	2	3
ˆ	1	1	ˆ	X	X	1	ˆ
			∧				
			i2				

-4	-3	-2	-1	0	1	2	3
ˆ	1	1	ˆ	X	X	1	ˆ
				∧			
				t0			

-4	-3	-2	-1	0	1	2	3
ˆ	1	1	ˆ	ˆ	X	1	ˆ
					∧		
					t1		

k has been incremented so the head is in state t0 again, to double the Xs tiling the input.

-4	-3	-2	-1	0	1	2	3
ˆ	1	1	ˆ	ˆ	X	1	ˆ
						∧	
						t1	

-4	-3	-2	-1	0	1	2	3
ˆ	1	1	ˆ	ˆ	X	Y	ˆ
							∧
							t2

During the doubling of the tiling, the end of the input is encountered. The head changes state to r0 (r0 is the first of the rNUM states, denoting the group of states in charge of blanking the tape and returning k).

-4	-3	-2	-1	0	1	2	3
ˆ	1	1	ˆ	ˆ	X	Y	ˆ
						∧	
						r0	

-4	-3	-2	-1	0	1	2	3
ˆ	1	1	ˆ	ˆ	X	ˆ	ˆ
					∧		
					r0		

When in r0, the head replaces any symbol it reads with a ˆ while moving left. When the first ˆ is read, the state changes to r1 and the blanking continues (see **tiling loop** in item 4C).

-4	-3	-2	-1	0	1	2	3
ˆ	1	1	ˆ	ˆ	ˆ	ˆ	ˆ
				∧			
				r0			

-4	-3	-2	-1	0	1	2	3
ˆ	1	1	ˆ	ˆ	ˆ	ˆ	ˆ
			∧				
			r1				

When the mahine is in state r1 and encounters a ˆ, the symbol to the left of the current head position is the rightmost digit of k. The state updates to r2.

-4	-3	-2	-1	0	1	2	3
ˆ	1	1	ˆ	ˆ	ˆ	ˆ	ˆ
		∧					
		r2					

-4	-3	-2	-1	0	1	2	3
ˆ	1	1	ˆ	ˆ	ˆ	ˆ	ˆ
	∧						
	r2						

In r2, the head moves left until it encounters a ˆ, which is at the left end k.

-4	-3	-2	-1	0	1	2	3
ˆ	1	1	ˆ	ˆ	ˆ	ˆ	ˆ
∧							
r2							

-4	-3	-2	-1	0	1	2	3
ˆ	1	1	ˆ	ˆ	ˆ	ˆ	ˆ
	∧						
	END						

When the head reaches the ˆ, it moves right once and now points to the leftmost digit of k, where the machine enters the state END, which ends the calculation.

3.

{ ((q_0, ˆ), (END, 0, L)), ((q_0, 1), (t0, X, R)),
((t0, ˆ), (r1, ˆ, L)), ((t0, X), (t1, ˆ, R)), ((t0, Y), (t0, X, R)), ((t0, 1), (i0, 1, L)),
((t1, Y), (t1, Y, R)), ((t1, 1), (t2, Y, R)), ((t1, X), (t1, X, R)), ((t1, ˆ), (r0, ˆ, L)),
((t2, X), (t2, X, L)), ((t2, 1), (t3, 1, L)), ((t2, ˆ), (r0, ˆ, L)),
((t3, Y), (t3, Y, L)), ((t3, ˆ), (t0, X, R)), ((t3, X), (t3, X, L)),
((i0, X), (i0, X, L)), ((i0, ˆ), (i1, ˆ, L)),
((i1, 1), (i1, 1, L)), ((i1, ˆ), (i2, 1, R)),
((i2, 1), (i2, 1, R)), ((i2, ˆ), (t0, ˆ, R)),
((r0, Y), (r0, ˆ, L)), ((r0, X), (r0, ˆ, L)), ((r0, ˆ), (r1, ˆ, L)),
((r1, X), (r1, ˆ, L)), ((r1, ˆ), (r2, ˆ, L)), ((r2, 1), (r2, 1, L)), ((r2, ˆ), (END, ˆ, R)) }

4. A The machine works by ’tiling’ the input with increasing powers of 2. First, the input is checked, and if it’s 0, an X is written on the tape and the machine finishes. Otherwise, the first 1 of the input is replaced by an X, and the main loop begins. A counter for k (to the left of the input) is incremented, and the number of Xs doubles, covering more 1s. If at any point during the doubling, the end of the input (a ˆ) is encountered, all the tiling symbols (Xs and Ys) are replaced by ˆ, and the head moves to the leftmost digit of k, where the machine stops running.

B There are 4 symbols the machine uses:

- **1**, used for number representation in unary notation
- ˆ, used as a space symbol between numbers
- **X**, used to ’tile’ the input with powers of 2 and also break the machine in case of invalid input
- **Y**, used to help copy Xs

C **Main loop:** it runs, until the entirety of the input has been replaced by either Xs or Ys. In each iteration, k (stored to the left of the input) is incremented by 1 and then the number of Xs tries to double (see tiling loop).

Tiling loop: If an iteration is complete in it's entirety, the number of Xs at the begining of the input is doubled. This is done, by first replacing an X with a ˆ. Then a Y is written over the leftmost 1 still left in the input. The head keeps going left, skipping over all Xs and Ys until it encounters a ˆ, at which point it writes down an X and moves a step to the right, and the loop repeats, if there are still Xs left. Once this process has been repeated for all the Xs, there are an equal number of Xs and Ys covering the input. At this point, all the Ys are changed to Xs and when the head returns to the leftmost X, an iteration of the tiling loop has been completed. If the doubling reaches the end of the input at any point, all the Xs and Ys are replaced by ˆ and the value for k is returned. The replacing of Xs and Ys with ˆs relies on the fact that there is at most 1 ˆ in the tiling of the input, and that there are only Xs to the left of it.

D There were no failures for any inputs from the comprehensive test.

E I tried keeping track of 2^k separately and comparing it to the input, but the machine was too slow and inefficient. At this point I came up with the idea of tiling the input with Xs, doubling it and keeping count of the number of doublings, since the number of Xs is always of the form 2^k and the number of doublings corresponds to k. When the initial X is written over the leftmost 1 of the input, it corresponds to 2^0, and k is at that point equal to 0. If a doubling of Xs fails (i.e. reaches the end of the input before completion), we know for sure that $2^{k-1} + 1 \leq \text{input} \leq 2^k$, and since $\mathit{cblog}(n) = k \ \forall \ 2^{k-1} + 1 \leq n \leq 2^k$, we can return k to produce the correct result.

F I performed calculations for all n from 0 to 2048 inclusive and plotted the number of steps required for calculation in relation to n. I discovered that the graph is linear between $2^{k-1}+1$ and 2^k, with slope 2^k+4, for all integers $k \geq 2$. Since $k = \mathit{cblog}(n) \ \forall \ 2^{k-1}+1 \leq n \leq 2^k$, we can conclude that for any input $n \geq 3$, the number of steps it takes to compute $\mathit{cblog}(n)$ is $(2^{\mathit{cblog}(n)} + 4)n + c$, where c is a constant integer, equal for all $2^{k-1}+1 \leq n \leq 2^k$. I obtained c using the LINEST function in Excel, but it can also be computed manually using the equation of the line, $y = mx + c$).

Suppose that a position on the tape counts as used if the head points to it at any point during the computation. Then the portion of the tape used to compute $\mathit{cblog}(n)$ is equal to $n + \mathit{cblog}(n) + 3$. The n comes from the length of the input, $\mathit{cblog}(n)$ is the length of the computed result (k), and the 3 is from the blank spaces that the head goes over: one immediately to the right of the input, one between n and $\mathit{cblog}(n)$ and one immediately to the left of $\mathit{cblog}(n)$.

5.

In both cases it is possible to use the fact that a k-digit binary number without leading zeroes x always satisfies $2^k \leq x \leq 2^k + 1$, and thus $\mathit{cblog}(x) = k + 1$.

If the input and output were both binary, the approach is different than for unary input and output. Once k would be obtained and written in unary (by overwriting the 0s in the input with 1s and adding a 1 at the end) it is just a matter of converting it to binary.

The computation would be even simpler with binary input and unary output. The machine would simply overwrite all the 0s in the input into 1s, write a 1 at the end and thus produce the correct output in unary notation.

Both of these problems are much easier to solve beacuse there is a very clear connection between the length of the representation of a natural number in base n and the logarithm with base n.

6.