CS 4820, Spring 2018 **Homework 9, Problem 1**
Name: Rongguang Wang
NetID: rw564
Collaborators: Siyuao Liu (sl2928); Yihao Chen (yc2288)

(1) *(10 points)* Design a single-tape Turing machine to evaluate the "less than" relation on two natural numbers represented in binary. The input to the Turing machine is represented as a string over the alphabet $\{0, 1, >, ?\}$. The input string is always in the format $a < b?$, where each of a and b is a string of one or more binary digits, beginning with the digit 1. Your Turing machine should terminate in the "yes" state if the number represented by a in binary is strictly less than the number represented by b in binary, and it should terminate in the "no" state if the number represented by a in binary is greater than or equal to the number represented by b in binary. If the input violates the format requirements, any behavior is acceptable as long as your algorithm terminates.

Example: if the input is $11 < 100?$ the answer is "yes". If the input is $0011 < 100?$ your algorithm should terminate, but it is fine to terminate in any of the "yes", "no", or "halt" states because the first binary string does not begin with the digit 1.

Your answer must include a description, in English (with accompanying notation as needed), of the alphabet, state set, and transition rule of your Turing machine, and a few sentences explaining how the algorithm operates and how the specified states and transitions implement those operations. You may choose to include a representation of the transition rule in tabular form (similar to those represented in Section 2 of the lecture notes) if you wish, but the tabular representation of the Turing machine is *optional* whereas the human-interpretable explanation is *mandatory*.

Your solution should include an analysis of the worst-case running time, as a function of the length of the input string. *You do not need to write a proof of correctness.*

Solution:

The Transition Rule is as below table shows.

		$\delta(q,\sigma)$		
q	σ	state	symbol	direction
s	$\triangleright$	r_s	$\triangleright$	$\rightarrow$
r_s	0	halt	$\sqcup$	—
r_s	1	r	1	$\rightarrow$
r	0	r	0	$\rightarrow$
r	1	r	1	$\rightarrow$
r	$<$	r_s	$<$	$\rightarrow$
r	?	l_b	?	$\leftarrow$
l_b	0	l	$\bar{0}$	$\leftarrow$
l_b	1	l	$\bar{1}$	$\leftarrow$
l_b	$\bar{0}$	l_b	$\bar{0}$	$\leftarrow$
l_b	$\bar{1}$	l_b	$\bar{1}$	$\leftarrow$
l_b	$<$	l_{a_be}	$<$	$\leftarrow$
l	0	l	0	$\leftarrow$
l	1	l	1	$\leftarrow$
l	$<$	l_a	$<$	$\leftarrow$
l	$\triangleright$	s	$\triangleright$	$\rightarrow$
l_a	0	l	$\bar{0}$	$\leftarrow$
l_a	1	l	$\bar{1}$	$\leftarrow$
l_a	$\bar{0}$	l_a	$\bar{0}$	$\leftarrow$
l_a	$\bar{1}$	l_a	$\bar{1}$	$\leftarrow$
l_a	$\triangleright$	yes	$\triangleright$	—
l_{a_be}	0	no	0	—
l_{a_be}	1	no	1	—
l_{a_be}	$\bar{0}$	l_{a_be}	$\bar{0}$	$\leftarrow$
l_{a_be}	$\bar{1}$	l_{a_be}	$\bar{1}$	$\leftarrow$
l_{a_be}	$\triangleright$	c	$\triangleright$	$\rightarrow$
c	$\bar{0}$	a_0	0	$\rightarrow$
c	$\bar{1}$	a_1	1	$\rightarrow$
c	0	c	0	$\rightarrow$
c	1	c	1	$\rightarrow$
c	$<$	no	$<$	—
a_0	$\bar{0}$	a_0	$\bar{0}$	$\rightarrow$
a_0	$\bar{1}$	a_0	$\bar{1}$	$\rightarrow$
a_0	$<$	b_0	$<$	$\rightarrow$
a_1	$\bar{0}$	a_1	$\bar{0}$	$\rightarrow$
a_1	$\bar{1}$	a_1	$\bar{1}$	$\rightarrow$
a_1	$<$	b_1	$<$	$\rightarrow$
b_0	$\bar{0}$	l_c	0	$\leftarrow$
b_0	$\bar{1}$	yes	1	—
b_0	0	b_0	0	$\rightarrow$
b_0	1	b_0	1	$\rightarrow$
b_1	$\bar{0}$	yes	0	—
b_1	$\bar{1}$	l_c	1	$\leftarrow$
b_1	0	b_1	0	$\rightarrow$
b_1	1	b_1	1	$\rightarrow$
l_c	0	l_c	0	$\leftarrow$
l_c	1	l_c	1	$\leftarrow$
l_c	$\bar{0}$	l_c	$\bar{0}$	$\leftarrow$
l_c	$\bar{1}$	l_c	$\bar{1}$	$\leftarrow$
l_c	$<$	l_c	$<$	$\leftarrow$
l_c	$\triangleright$	c	$\triangleright$	$\rightarrow$

The problem is aiming to design a single-tape Turing Machine to evaluate the "less than" relation on two natural numbers represented in binary. The input string is in the format of "$a < b$?", where each of a and b is a string of one or more binary digits, beginning with digit 1. The TM returns "yes" if $a < b$; returns "no" if $a \geq b$; "halt" if the input of a or b is not beginning with digit 1. The alphabet set of the TM is $\{\triangleright, \sqcup, 0, 1, \bar{0}, \bar{1}, <, ?\}$. The machine has 16 states: $\{s, yes, no, halt, r_s, r, l, l_b, l_{a\_be}, l_a, c, a_0, a_1, b_0, b_1, l_c\}$. State s is the starting state. States *yes*, *no* and *halt* represent the output of the TM is yes, no and the TM terminate, respectively. State r_s is preparing to check the beginning bit of a and b of the input string. States r and l are moving right or left, respectively. State l_b is preparing to move left since it already reach the question mark ? which is the end of the input string and is preparing to compare the cardinality of input b with input a starting from the least significant bit. State l_{a_be} is the mark that the bits of input b is exhausted when comparing the cardinality of a and b. State l_a is preparing to compare the cardinality of input a with input b starting from the least significant bit. State c is preparing to evaluate each digit of input a and b starting from the most significant bit. State a_0 is recording the bit 0 of a that we are currently evaluating to b. State a_1 is recording the bit 1 of a that we are currently evaluating to b. State b_0 is preparing to evaluate the bit of a which is recorded as 0 to the corresponding bit of b. State b_1 is preparing to evaluate the bit of a which is recorded as 1 to the corresponding bit of b. State l_c is preparing to move left since the bit of a and b evaluated are equal.

Now lets define the transition rule. Basically, there are two situation in the problem. The cardinality of the input a and b can be either equal or not equal. If the cardinality of a and b are not equal, we can immediately determine which one is larger since the larger one has greater cardinality. On the other hand, if a and b have the same cardinality, we should compare a and b bit by bit from the most significant bit (MSB) to the least significant bit (LSB). We also need to pay attention to the beginning digit of a and b. If the state r_s comes across the symbol 0, the machine should halt. To begin with, we first compare the cardinality of a and b from the LSB to the MSB. We traverse through the input string until symbol ?. The state will change to l_b which means we are preparing to compare the cardinality of b with a. l_b overwrites the symbol 0 or 1 to $\bar{0}$ or $\bar{1}$, respectively, that it first comes across and move left. The change from 0 or 1 to $\bar{0}$ or $\bar{1}$ represents that we already compared this digit. The next state will be l which just move left without changing the symbol. l_b ignores the symbol $\bar{0}$ and $\bar{1}$, and only move left. Once l_b comes across symbol $<$, the next state is l_{a_be} which means the digits of b are exhausted. If state l comes across symbol $<$, the next state will be l_a which means that the digit of b is not exhausted yet and we are preparing to compare the cardinality of a to b. Similar to l_b, l_a overwrites the symbol 0 or 1 to $\bar{0}$ or $\bar{1}$, respectively, that it first came across and move left. The next state will be l which just move left without changing the symbol. l_a ignores the symbol $\bar{0}$ and $\bar{1}$, and only move left. Once l_a comes across symbol $\triangleright$, the next state is *yes* which means the digits of a are exhausted and b is greater than a. If state l_{a_be} comes across $\bar{0}$ or $\bar{1}$, it just move left without changing the symbol. Once l_{a_be} comes across 0 or 1, the next state will be *no* which means the cardinality of a is greater than b, and thereby, a is greater than b. If l_{a_be} comes across $\triangleright$, the next state is c which means we are preparing to compare each digit of a and b from the MSB to LSB. State c overwrites the symbol $\bar{0}$ or $\bar{1}$ back to 0 or 1, respectively, that it comes across and move right. The next state is a_0 or a_1 correspondingly. State a_0 records the current evaluating bit of a is 0 and state a_1 records the current evaluating bit of a is 1. States a_0 and a_1 ignores the symbol $\bar{0}$ or $\bar{1}$ they come across and move right until symbol $<$ appear. a_0 and a_1 transit to state b_0 and b_1 respectively, which passes the digit of a and marks that we are preparing to evaluate the digit of b to the digit of a. States b_0 and b_1 ignores the symbol 0 or 1 they come across and move right. Once state b_0 comes across $\bar{1}$, the next state is *yes* which means the digit of b is greater than that of a, and thereby, b is greater than a. If state b_0 comes across $\bar{0}$, the next state is l_c which ignores symbol 0, 1, $\bar{0}$, $\bar{1}$ or $<$ and only move left. Once state b_1 comes across $\bar{0}$, the next state is *yes* which means the digit of b is greater than that of a, and thereby, b is greater than a. If state b_1 comes across $\bar{1}$, the next state is l_c. State c ignores symbol 0 or 1 it comes across and just move right. Once state c comes across $<$, the next state is *no* which means all digits of a and b were compared and they are equal.

Running Time:

The running time of the algorithm can be divided into two parts: comparing cardinality and evaluating digits. If the size of the input string is n, a and b have $n/2$ each. It will cost at most $O(n^2)$ for the cardinality comparison. Similarly, another $O(n^2)$ needed for the digits evaluation in worst case. Therefore, the total running time is $O(n^2)$.