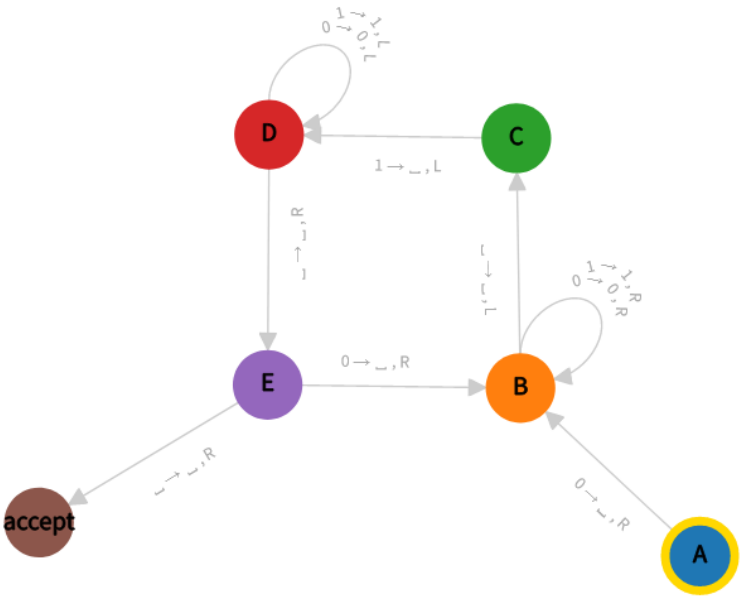
Full Name: Kaan Karaçanta

Id Number: 2448546

Answer 1



State A: The start state, it can just read 0, and makes it empty, otherwise machine rejects the input at the beginning.

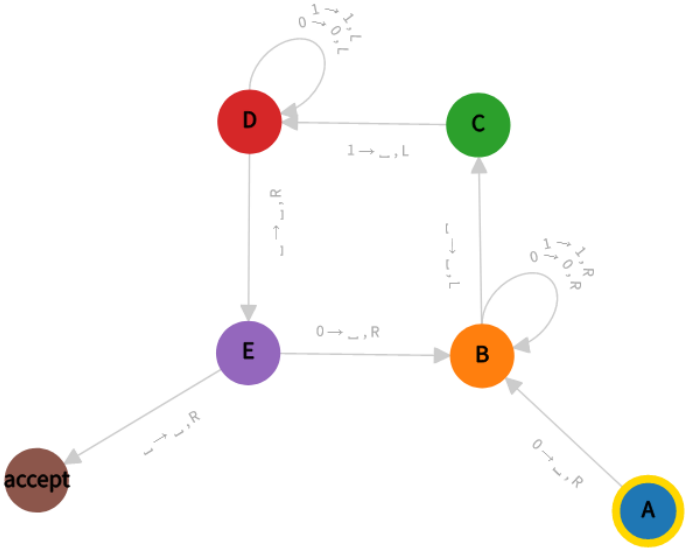
State B: Reads the input (without changing) until tape head reaches the end of the input, then moves tape head to the left and goes to state C.

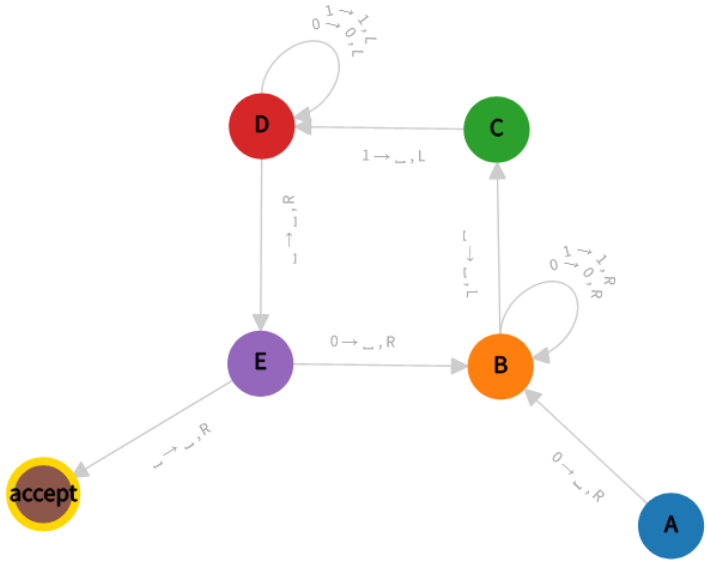
State C: This state needs to read 1, which is the rightmost 1 actually, and makes it empty with moving the tape head to the left.

State D: This moves the tape head to the leftmost non-empty input symbol without changing any input, when it sees the first empty character, it changes the state to E with moving the tape head to the right.

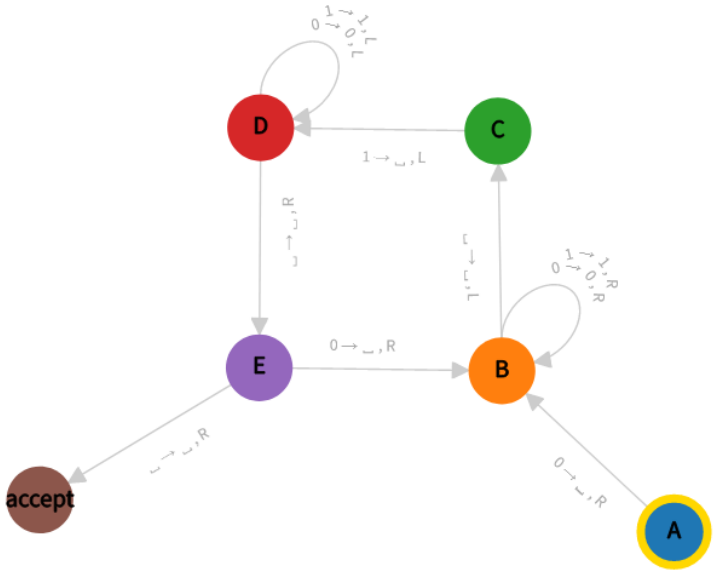
State E: This decides whether all the input is read and cleared, or there are more to read. If there is an input 0, the state goes to B and starts a new cycle, and if there is no input left, it accepts the given string.

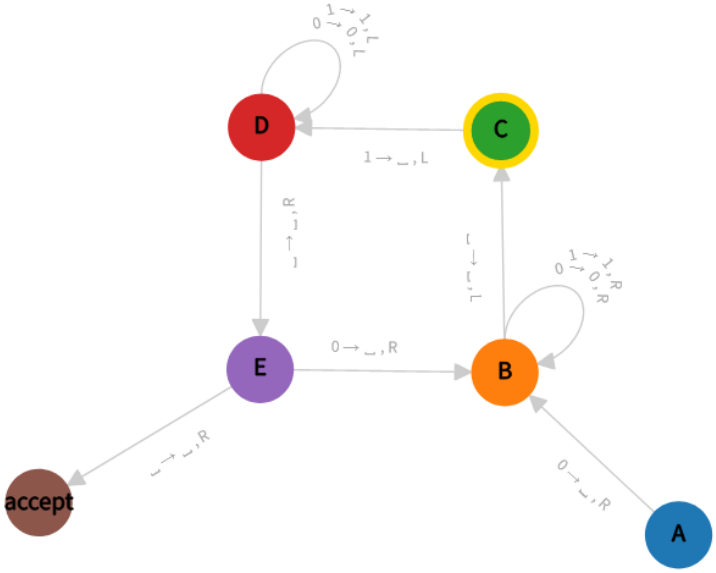
Sample 1 (initial - end):

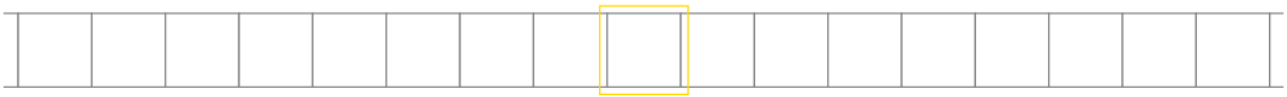




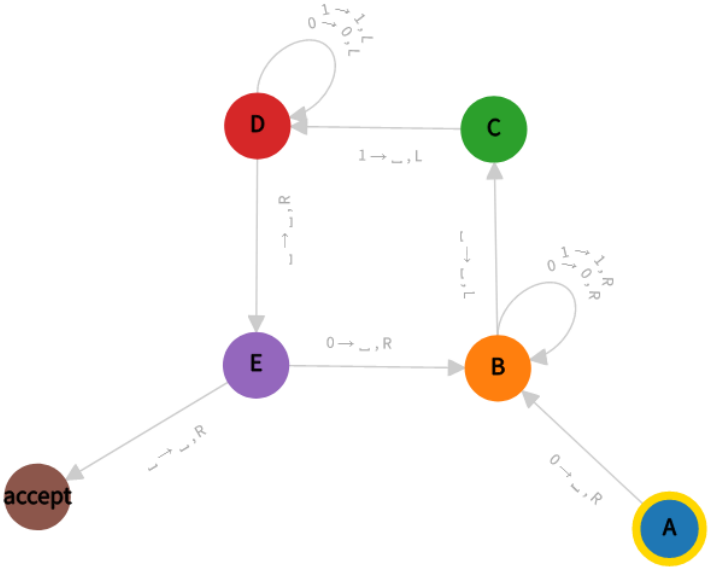
Sample 2 (initial - end):

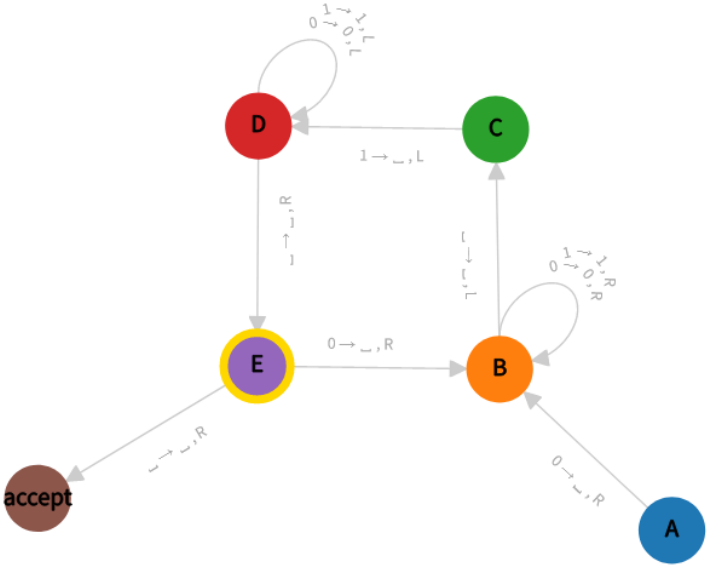




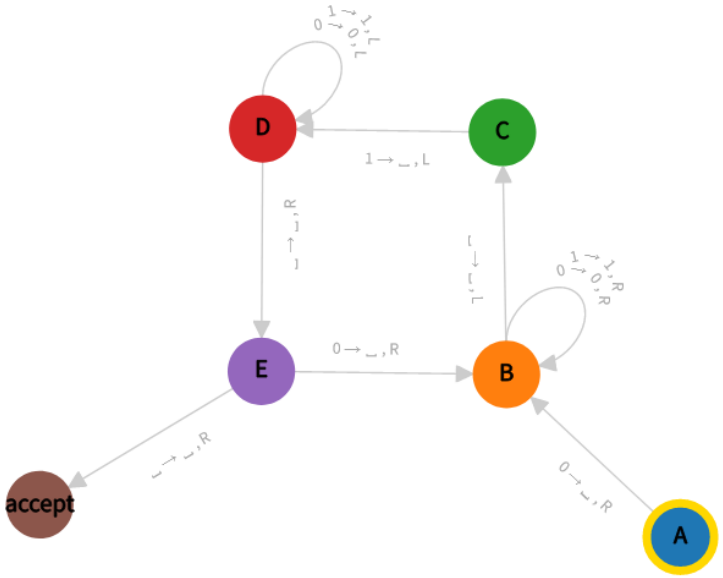


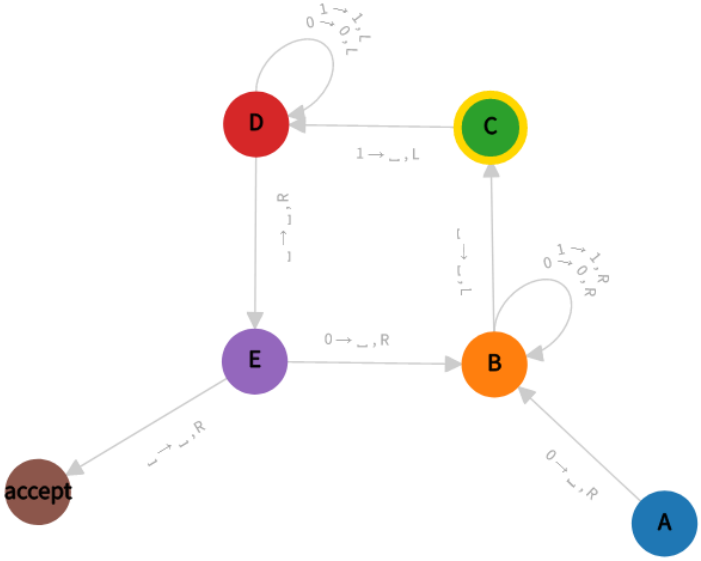
Sample 3 (initial - end):



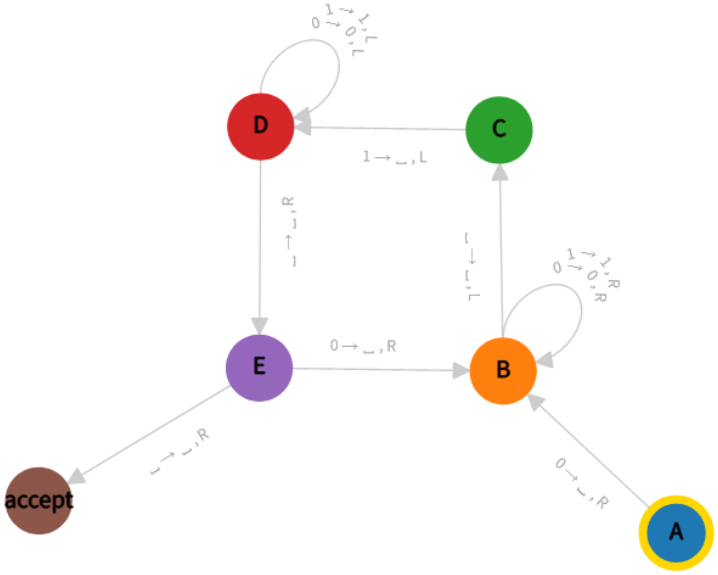


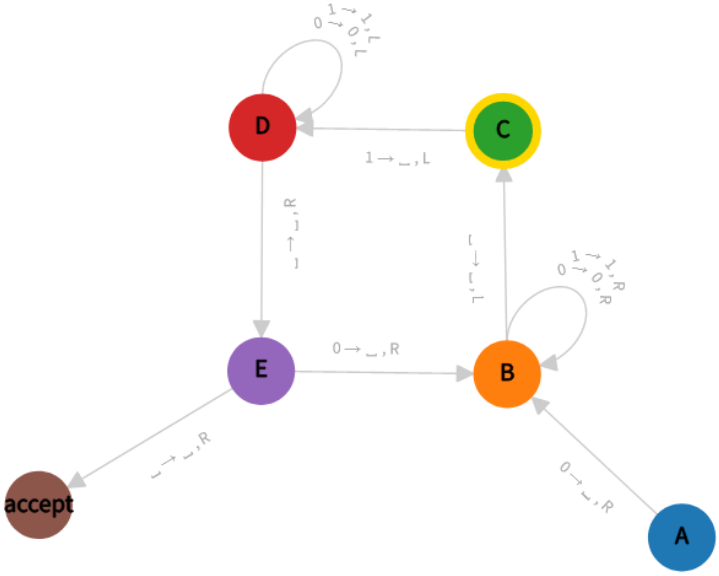
Sample 4 (initial - end):



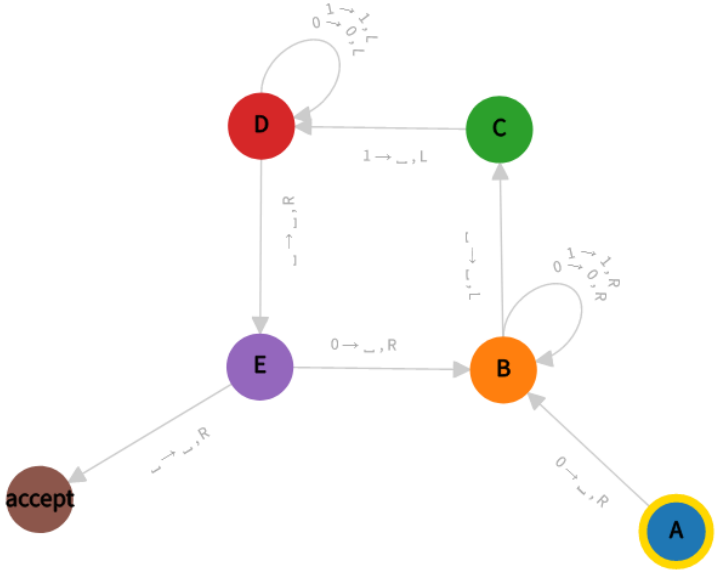


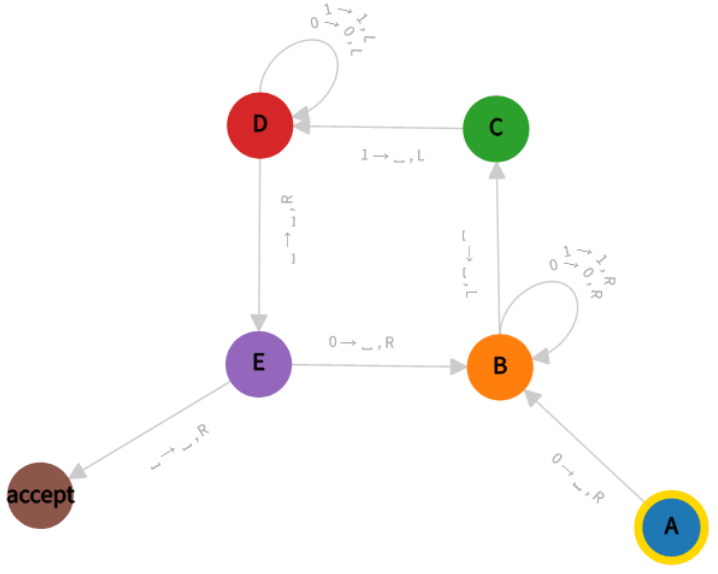
Sample 5 (initial - end):



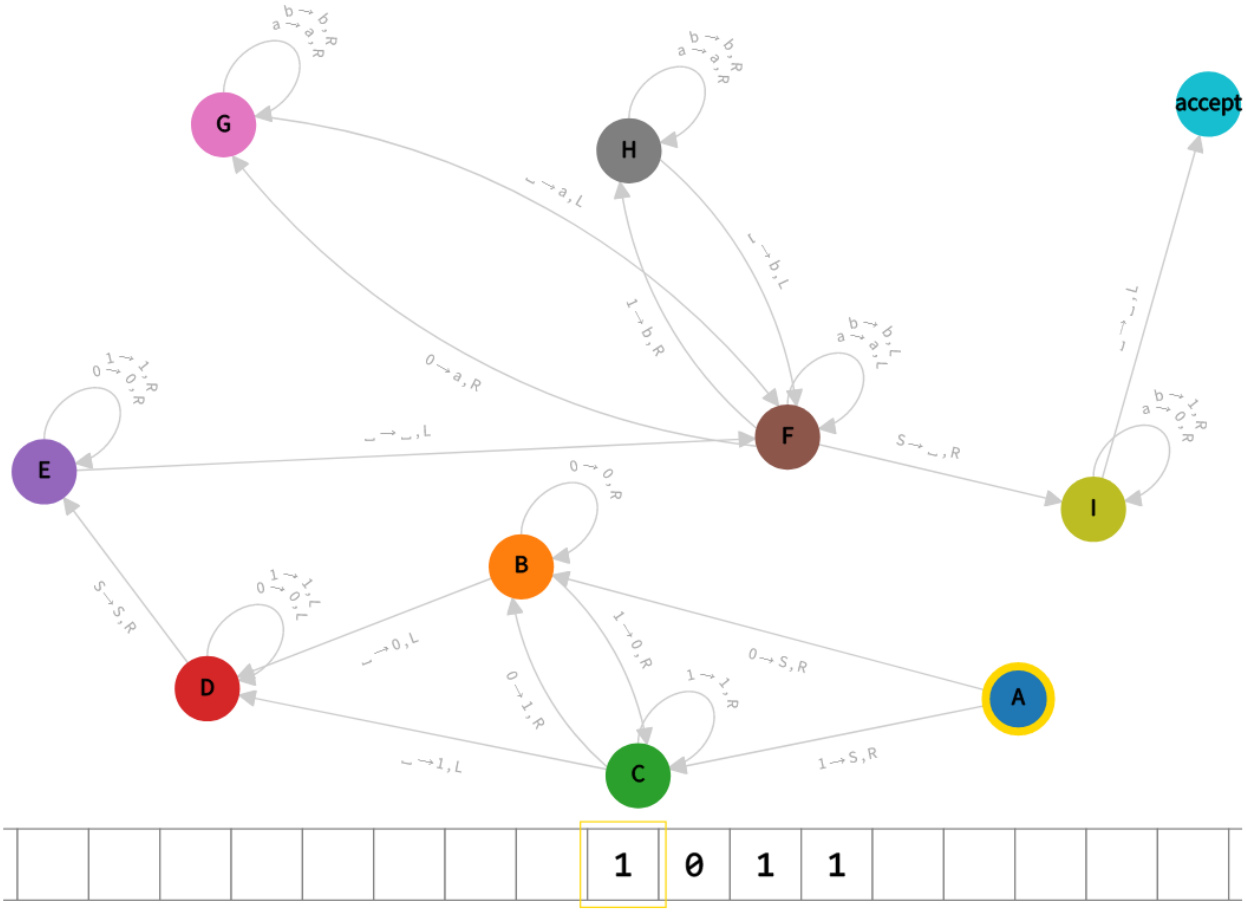


Sample 6 (initial - end):





Answer 2



State A: The start state, and from this to D, these states are just adding S to the beginning to indicate the left end of the string, so, assuming none of the inputs starting with empty string, it changes the written 0 or 1 with S and moves B or C accordingly.

State B: This state changes the input it reads to 0, if the input is already 0, stays in the same state, otherwise with the inputs 1 and ‘ ‘ goes to C and D.

State C: This is just doing the same thing as state B, yet writes 1 instead of 0.

State D: This state moves the tape head to the beginning of the actual string, the input after added S, and moves to state E to start the real function.

State E: It reads all the input until the end of it, then moves tape head to the last character of the input, and moves to state F.

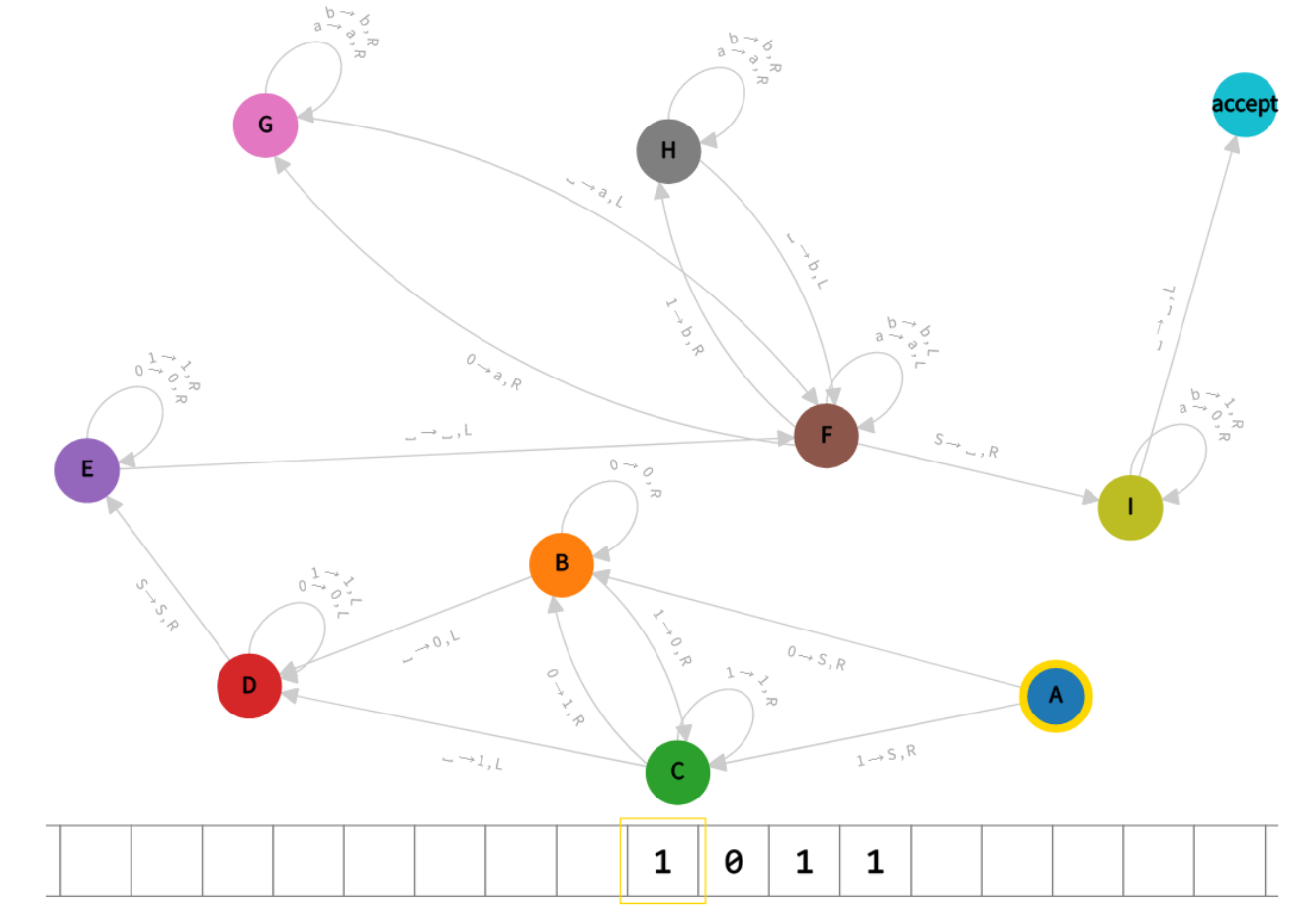
State F: This state starts to change the actual characters 0 and 1 with a and b respectively. Before that if the character it reads is a or b it does not change and moves the head to the left. If the input to be changed is 1 the state goes to H, else G.

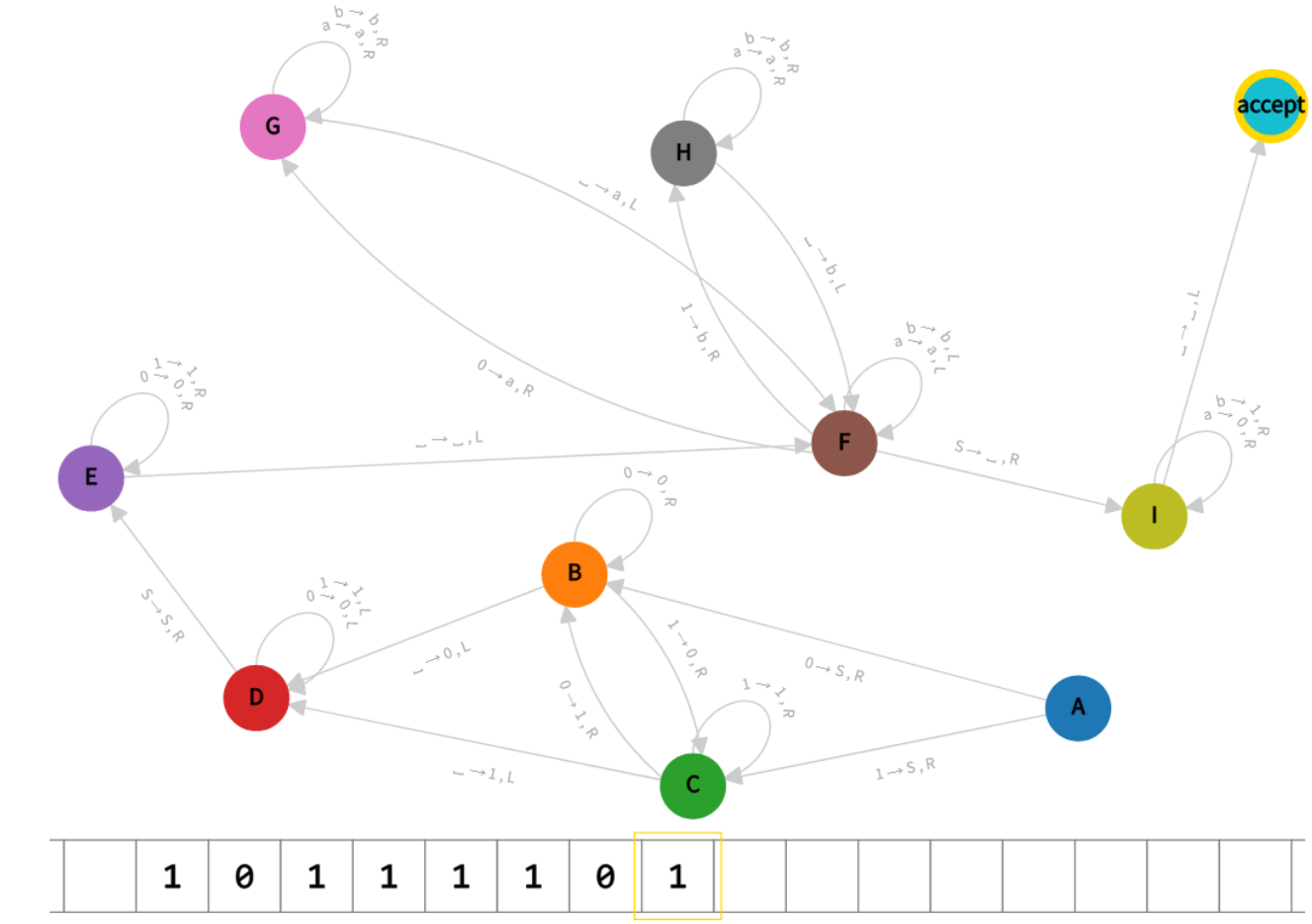
State G: It goes back to the end of the string again without changing a’s or b’s, by making the first empty character a and going back to F.

State H: It does the same thing as state G, but this time it replaces the empty character with b.

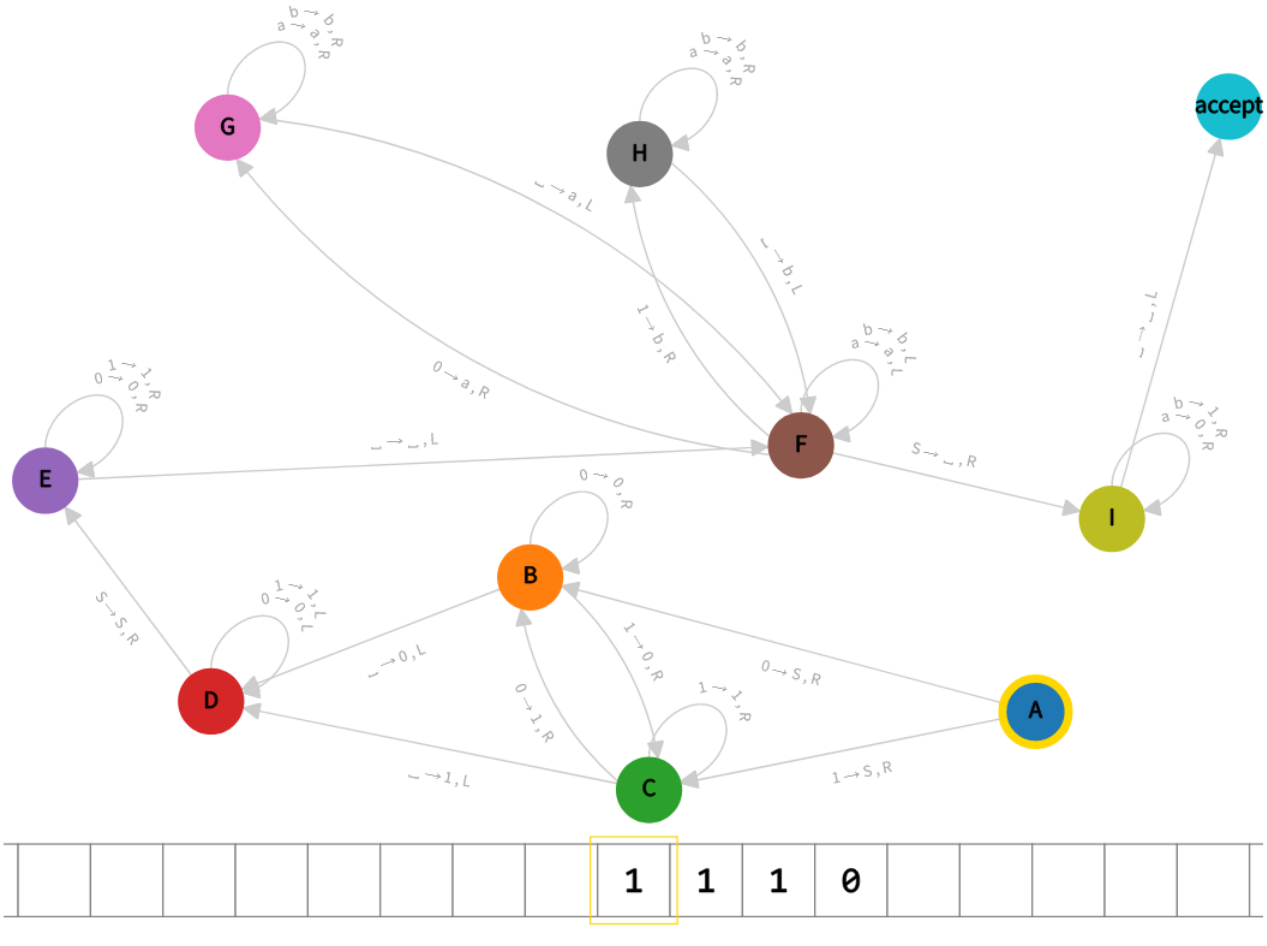
State I: After all of the process done between F, G and H, this state substitutes a’s and b’s with the real inputs 0’s and 1’s respectively, when it reaches to the end of the string, it goes to accept state.

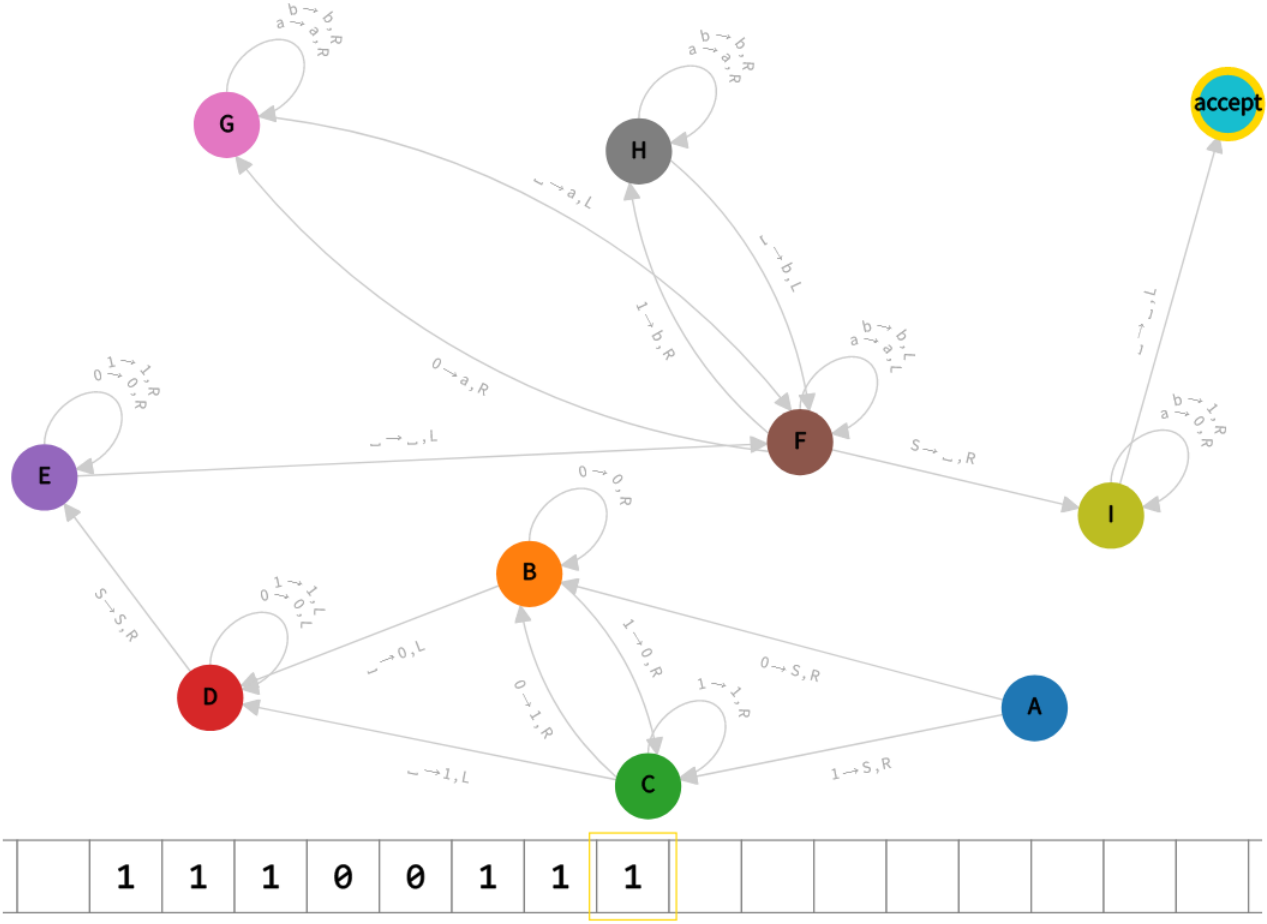
Sample 1 (initial - end):



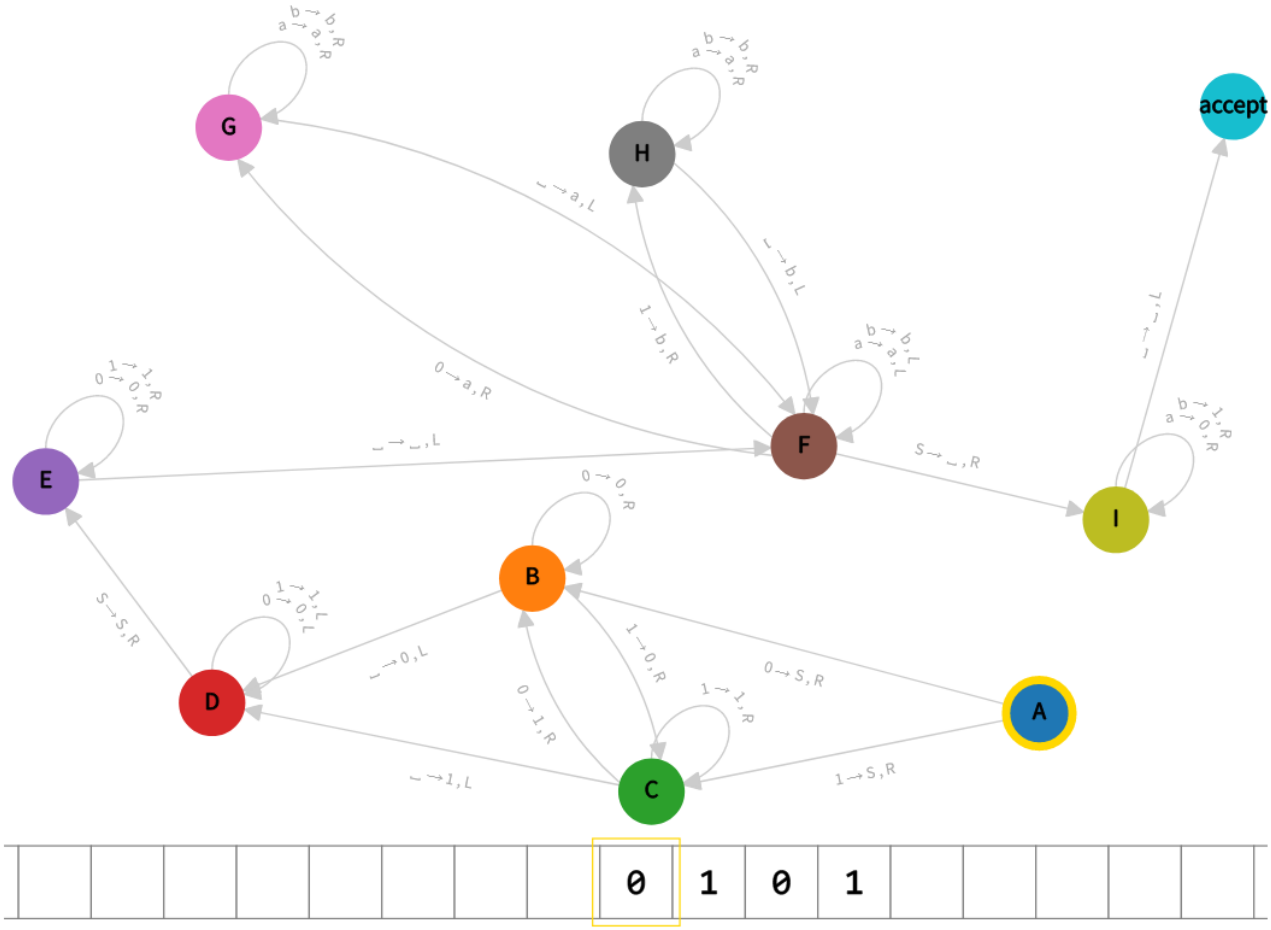


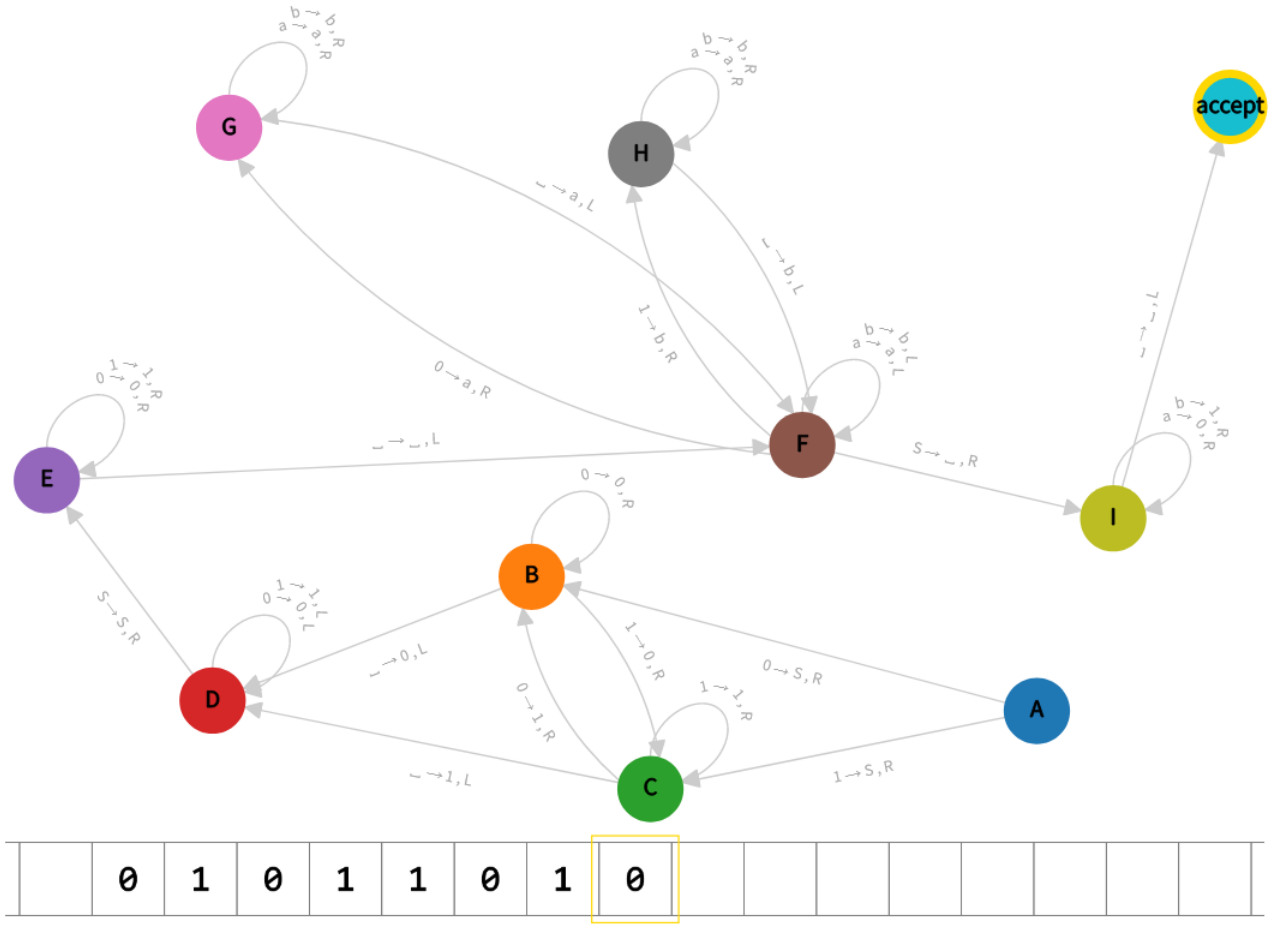
Sample 2 (initial - end):



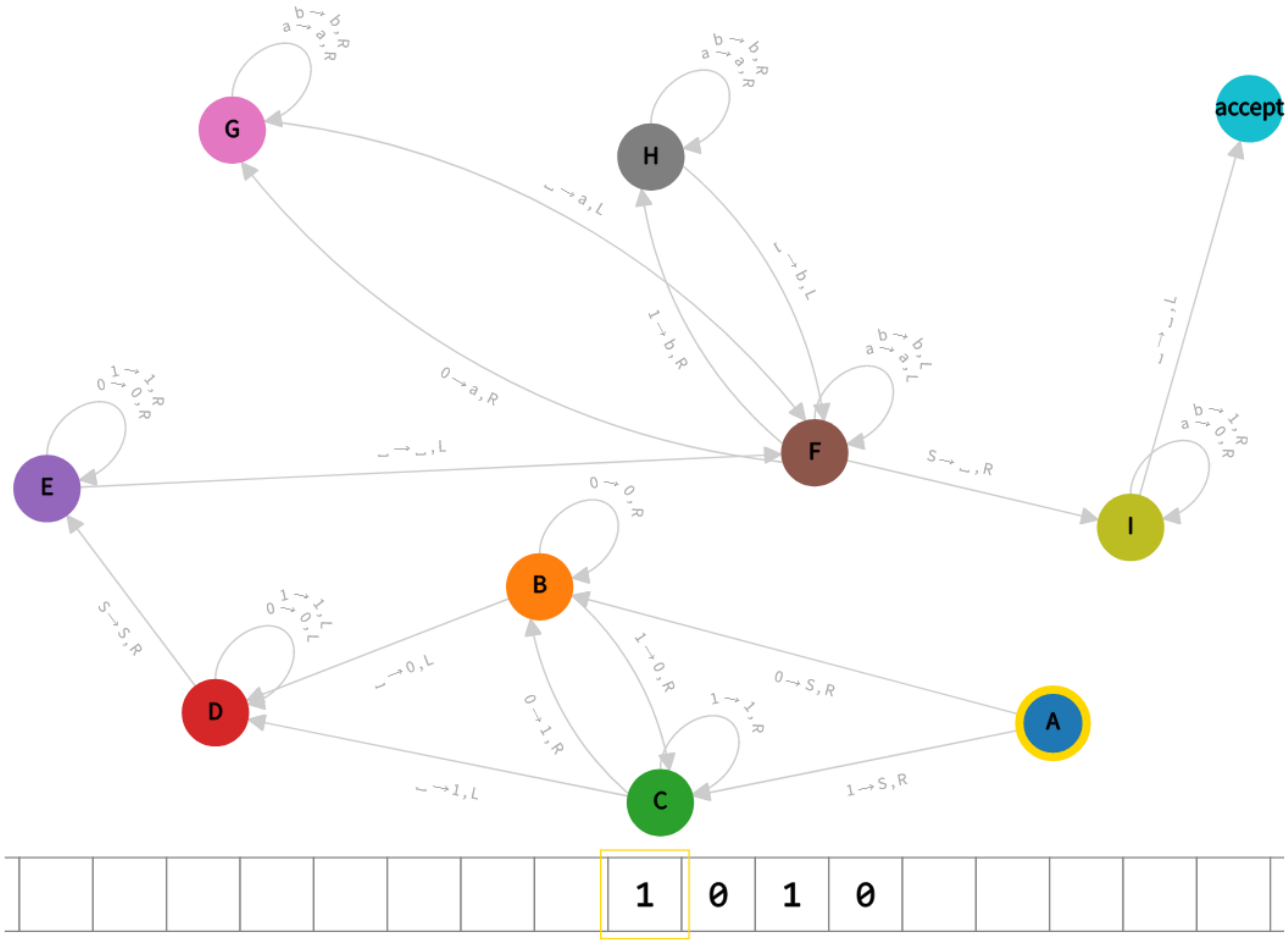


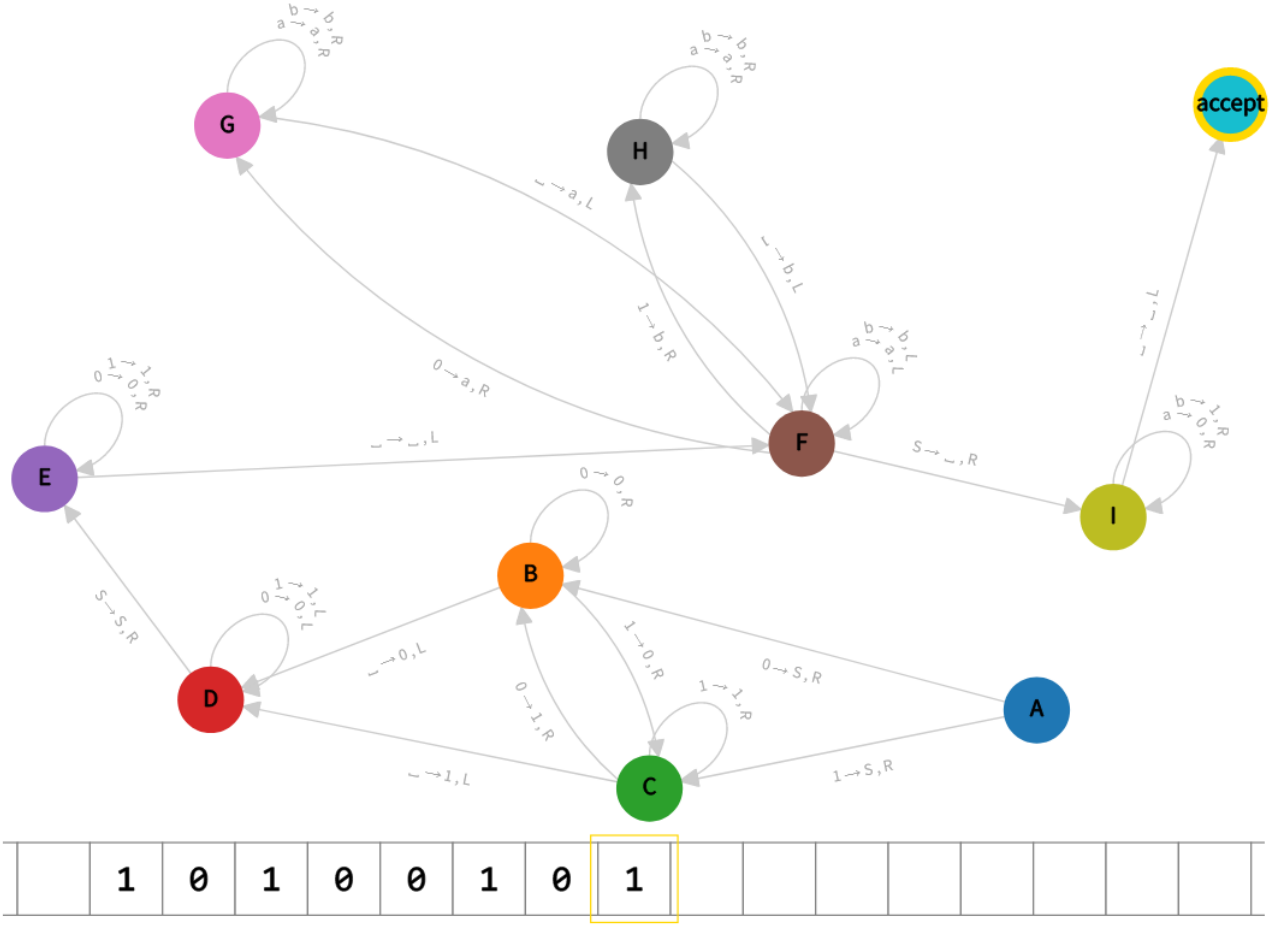
Sample 3 (initial - end):



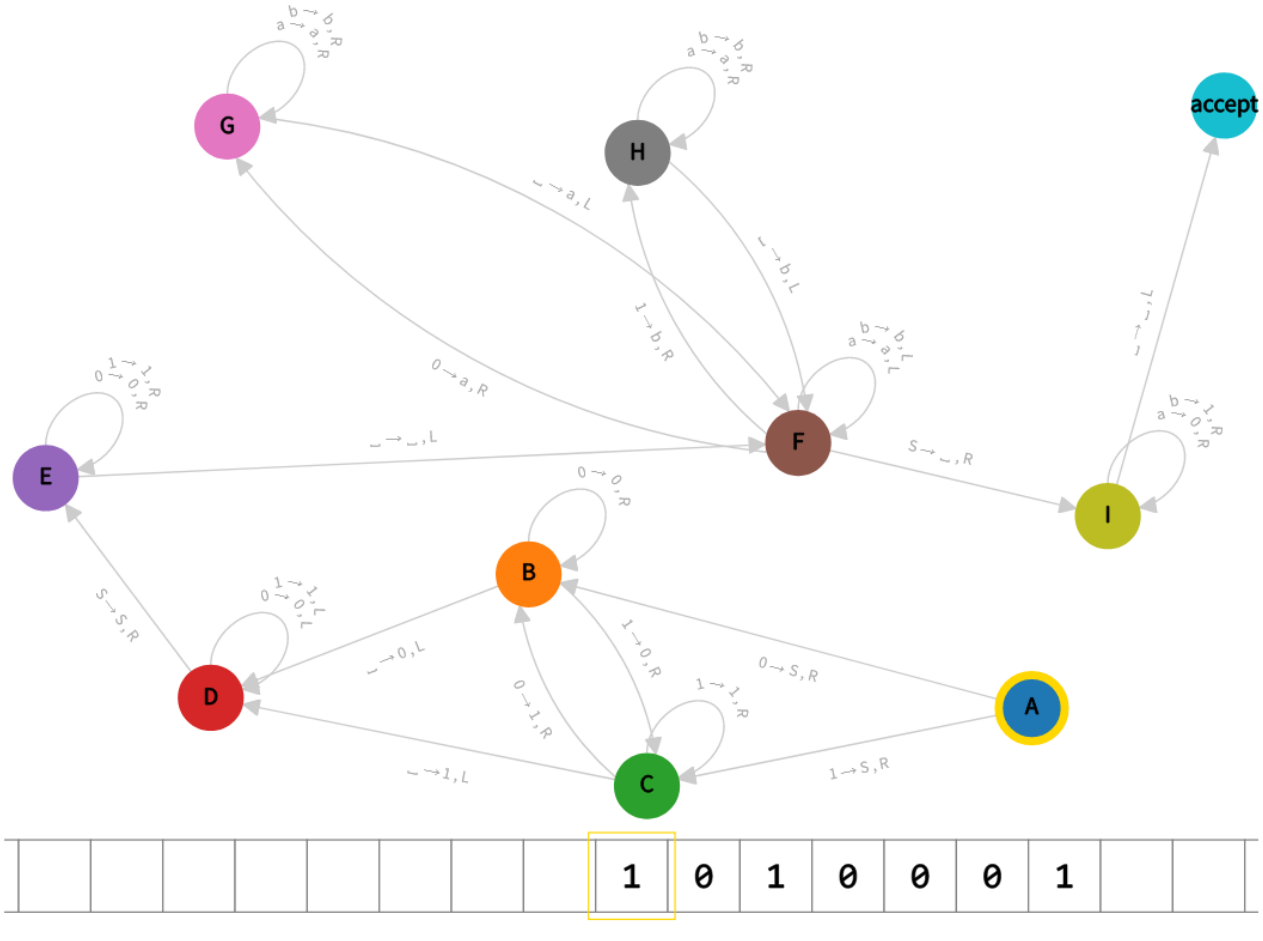


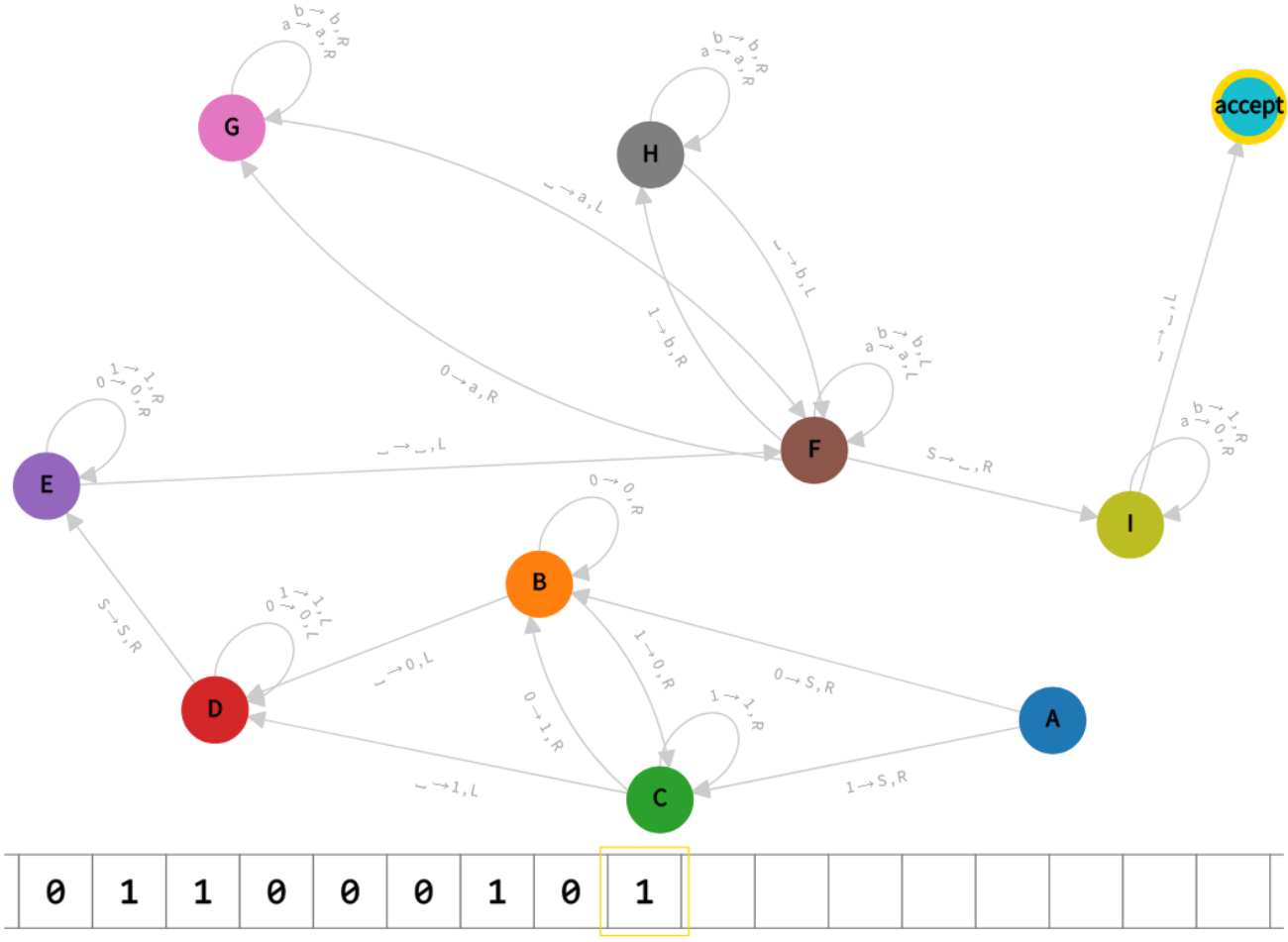
Sample 4 (initial - end):



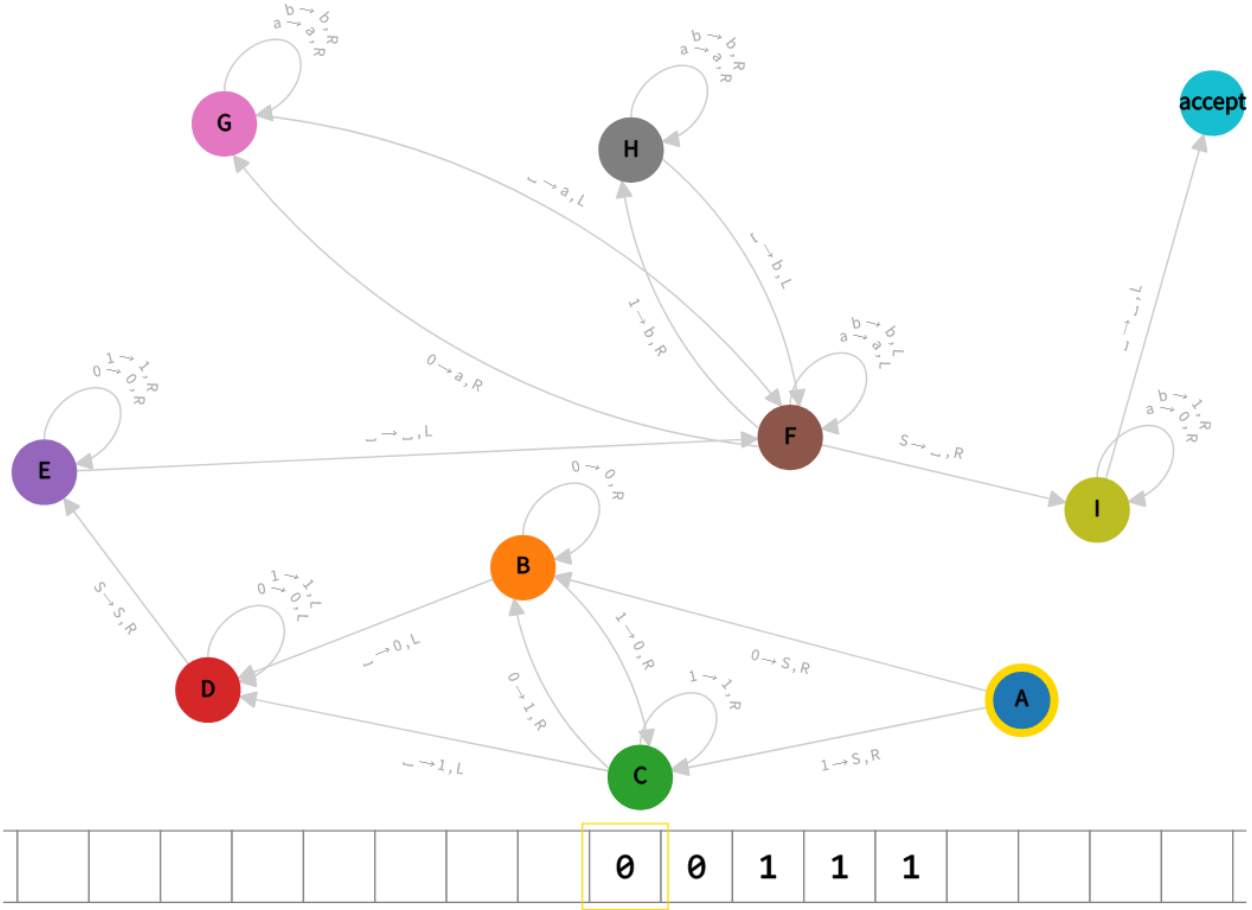


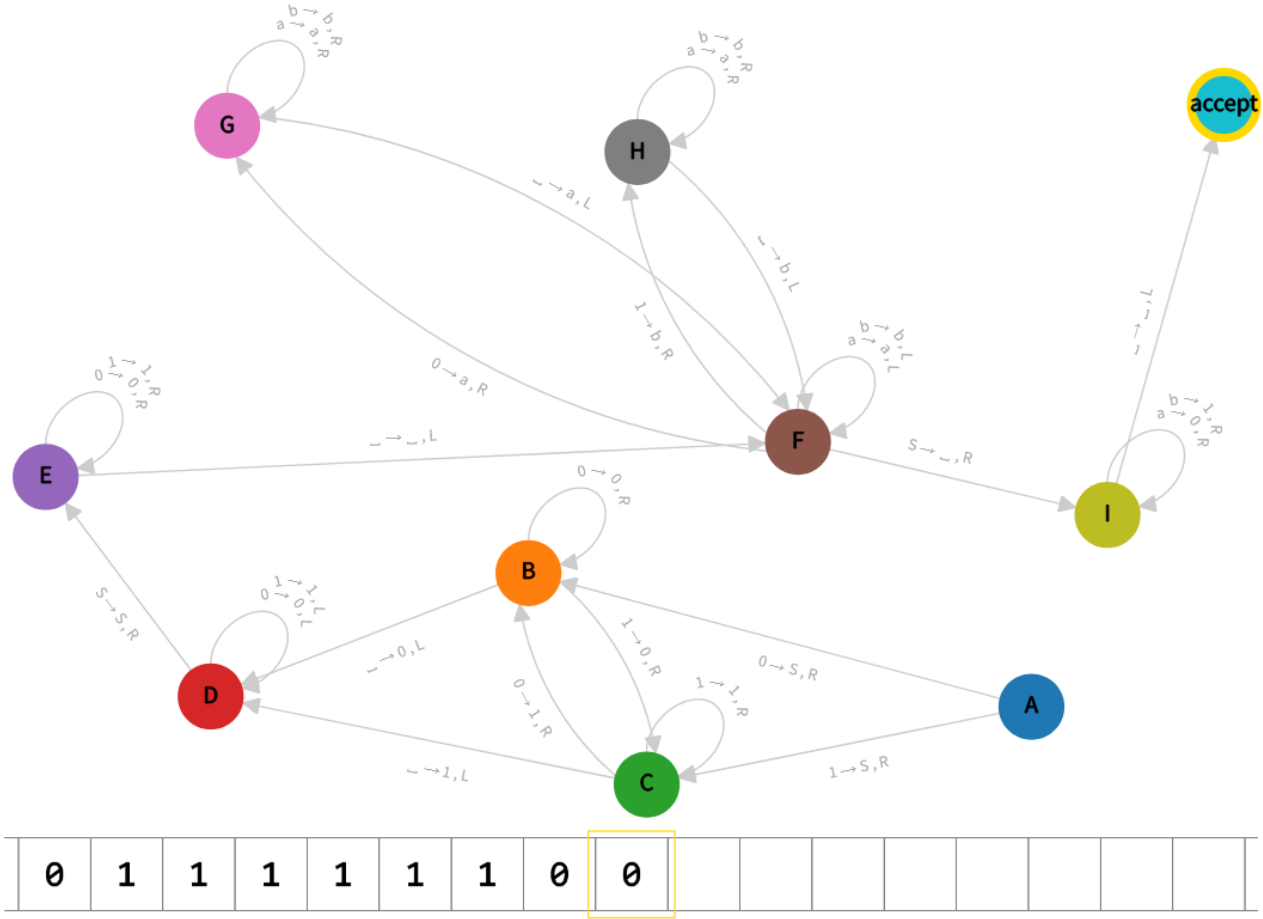
Sample 5 (initial - end):





Sample 6 (initial - end):





Answer 3

Turing machines with a 2-dimensional tape are just like an ordinary Turing machine with 2-dimensional array of cells; that is, there are no difference between these and the ordinary ones in terms of computational power. In this kind of machines, there are one finite control, one read-write head and one 2-dimensional tape. The tape is limited from its top and left ends, yet it can extend to any level to the down and right ends. Also, as expected, the tape head can move in 4 directions, instead of 2.

Formal definition of a 2-dimensional Turing machine can be given as a five-tuple $M = (K, \Sigma, \delta, s, H)$, where K, s, and H are the same as the ones in a normal Turing machine, yet Σ, also includes a bottom marker for the tape, and where δ is a function from $K \times \Sigma$ to $K \times (\Sigma \cup \{\uparrow, \rightarrow, \downarrow, \leftarrow\})$.

A configuration of a 2-dimensional Turing machine may be written as:

$$K \times N \times N \times T$$

Where Ns are the indexes of the tape, so in a configuration of these machines we have current state, current head position's x and y indexes, and a list of all non-blank squares on the input/tape. Moreover, the calculations this kind of machine needs to do may be that it will need to increment and decrement 2 indexes of x and y for the 2-dimensional tape, reading the input from the coordinates of the tape head, and writing the given input to the input tape.

Let w a string, and $t_w \in T$. When there is a Turing machine with 2-dimensional tape M with different accept and reject states y and n respectively, for this string w one of the following must be true, $(s,1,1, t_w) \vdash^* (y,i,j,t')$ or $(s,1,1, t_w) \vdash^* (n,i,j,t')$ for some i and $j \in N$ and $t' \in T$, that means the machine halts in one of those states. In this case we can say that this machine decided a language L that includes such strings.

For the time analysis, first, let i and j be the largest x and y coordinates that the machine has, so neither of them is larger than the number of cells t. Thus, the time required to simulate one move of the tape head is bounded by the quadratic polynomial in t. Thus, the time to carry out t steps, after finishing the initial setup (filling the table) is $O(t^3)$. Since the initial setup is an 2n-step computation of the operations this machine can make, writes, and moves, this whole process can be done in time $O(n^3)$. Hence, the operation asked in the question can be done in time polynomial in t and n.