Turing Machine

By

Dr. Bhargavi K

Turing Machine

⇒ Turing M/c has infinite size tape and it is used to accept Recursive Enumerable languages

⇒ TM can move in both directions. Also it doesn't accept ϵ.

⇒ If the string inserted in not in lang, m/c will halt in non-final state.

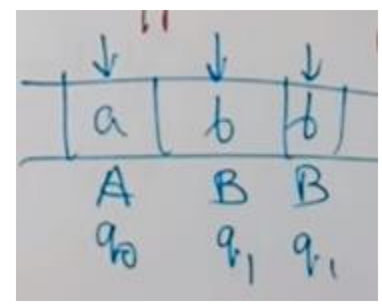
⇒ TM is a mathematical model which consists of an infinite length tape divided into cells on which i/p is given.

⇒ It consists of a head which reads the i/p tape

⇒ A state reg stores the state of T.M.

⇒ After reading an i/p symbol, it is replaced with another symbol, its internal state is changed & it moves from one cell to the right or left.

⇒ If the TM reaches the final state, the i/p string accepted, otherwise rejected.



Languague used

Formal Definition :

A TM can be formally described as $\boxed{\text{7-tuples}}$ $(Q, X, \Sigma, \delta$ X or T

X or T

$B, F)$ where :-

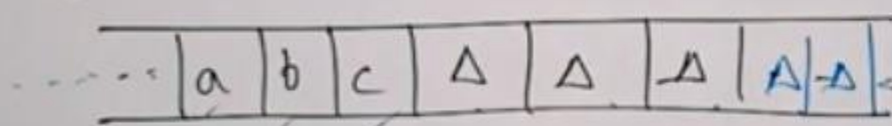
- Q is a finite set of states
- X is the tape alphabet
- Σ is the i/p alphabet
- δ is a transition function;

$$\delta : Q \times X \rightarrow Q \times X \times \{\text{left shift, Rig}$$

- q_0 is the initial state
- B is the $\boxed{\text{Blank symbol}}$ Δ $\sqcup$
- F is the set of final states

Basic Model of Turing Machine

① The i/p tape is having an infinite no: of cells, Each cell contains one i/p symbol. The empty tape i filled by blank characters

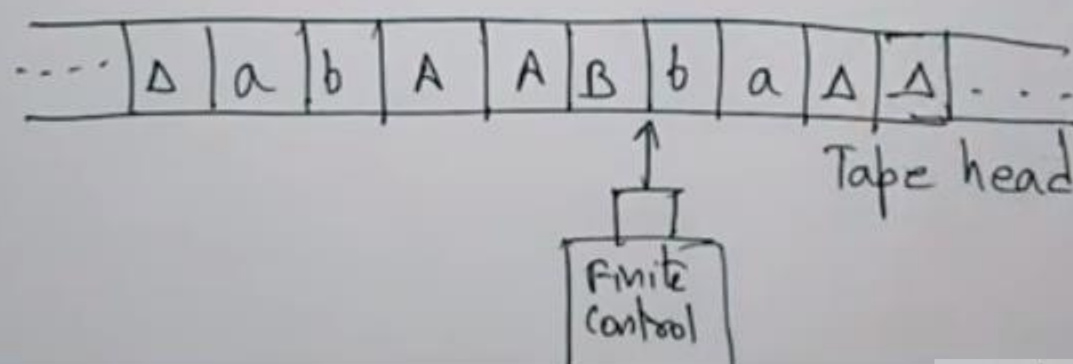


② The finite control & the tape head which is responsible for reading the current i/p symbol. The tape head can move to left to right.

3. A finite set of states through which m/c has to undergo

4. Finite set of symbols called external symbols which are used in building the logic of TM.

Δ - blank is a s
symbol
Δ ∉
used to
.. infinit



Δ - blank is a special symbol
$\Delta \notin \Sigma$
used to fill th
.. infinite tap

Language accepted by Turing Machine

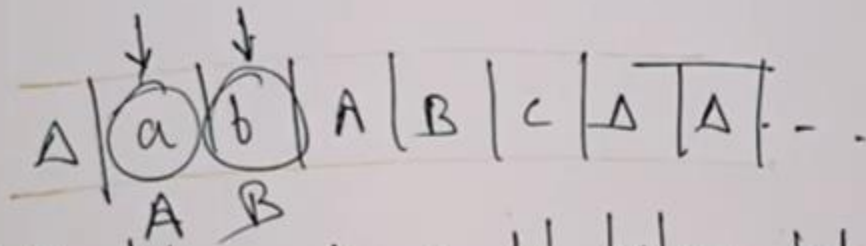
⇒ The T.M accepts all the lang even though they are recursive enumerable.

⇒ Recursive means repeating same set of rules for any no: of times.

⇒ Enumerable means a list of elements

⇒ TM also accepts the computable functions, such as additio multiplication, subtraction, division, and many more.

operation's on the tape

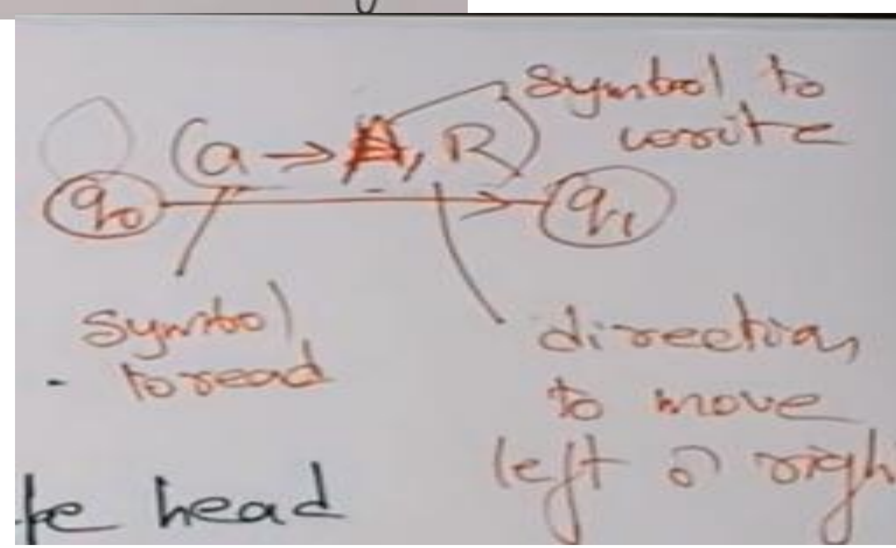


① Read/scan the symbol below tape head

② update/write a symbol below tape head

③ Move the tape head one step left

④ Move the tape head one step right



Eg:- Construct a T.M which accepts the lang of $\boxed{aba}$ over

$\Downarrow$ tape

$\Sigma = \{a, b\}$

Sol:- We will assume that on i/p tape the string 'aba' is placed

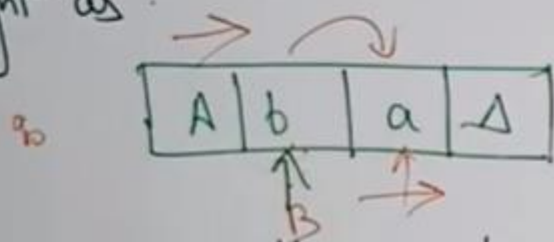
a	b	a	Δ	Δ	Δ	...

If the tape head is 'read out' 'aba' string then TM will halt after reading Δ.

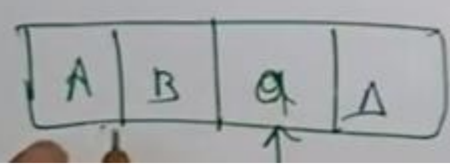
⇒ Initially, state is q_0 & head points to 'a' as:

	↓			
q_0	a	b	a	Δ

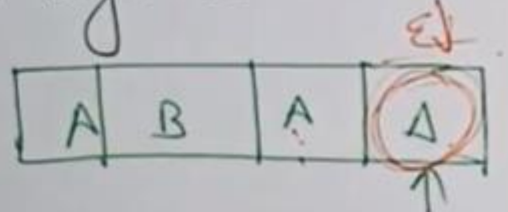
⇒ The move will be $\delta(q_0, a) = \delta(q_1, A, R)$ which means it will go to state q_1, replaced 'a' by 'A' & head will move to right as :



⇒ The move will be $\delta(q_1, b) = \delta(q_2, B, R)$ which means it will go to state q_2, replaced 'b' by 'B' and head will move to right as :

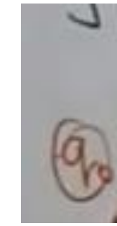


The move will be $\delta(q_2, a) = \delta(q_3, A, R)$ which means it will go to state q_3, replaced 'a' by 'A' and head will move to right as:



The move $\delta(q_3, \Delta) = (q_4, \Delta, S)$ which means it will go to state q_4 which is HALT state which is accept state for any can be represented by Transition table

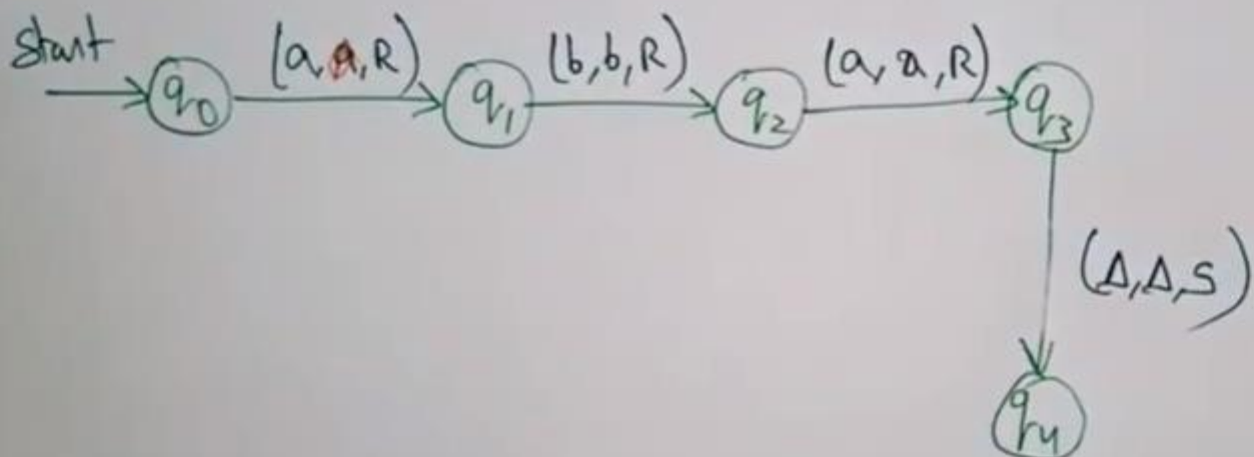
States	a	b	Δ
q_0	(q_1, A, R)	–	–
q_1	–	(q_2, B, R)	–
q_2	(q_3, A, R)	–	–
q_3	–	–	(q_4, Δ, S)
q_4	–	–	–



can be represented by Transition table

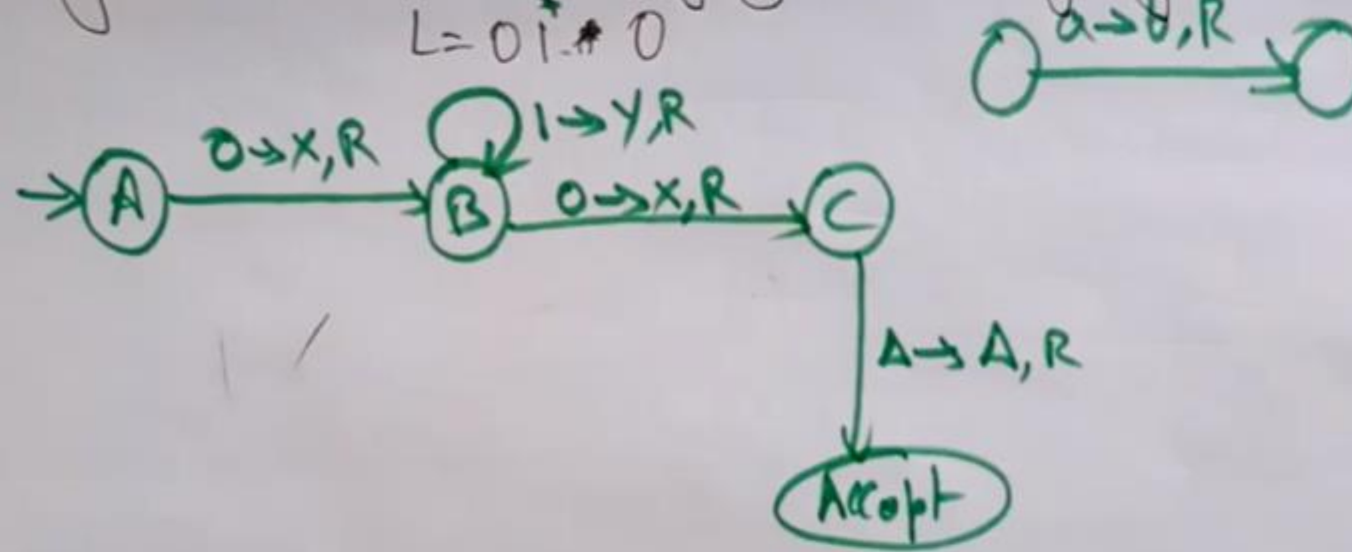
States	a	b	Δ
q_0	(q_1, A, R)	–	–
q_1	–	(q_2, B, R)	–
q_2	(q_3, A, R)	–	–
q_3	–	–	(q_4, Δ, S)
q_4	–	–	–

The Same TM can be represented by Transition diagram

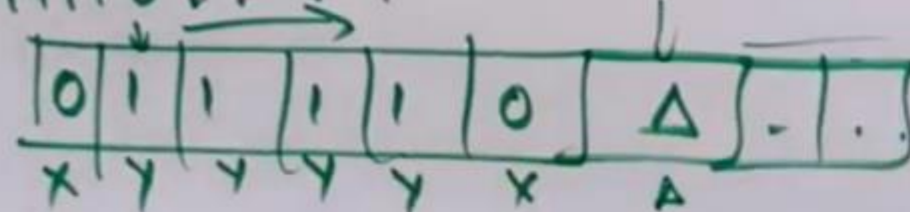


Example: Design a TM which recognize the language

$L = 0\,1^*\,0$

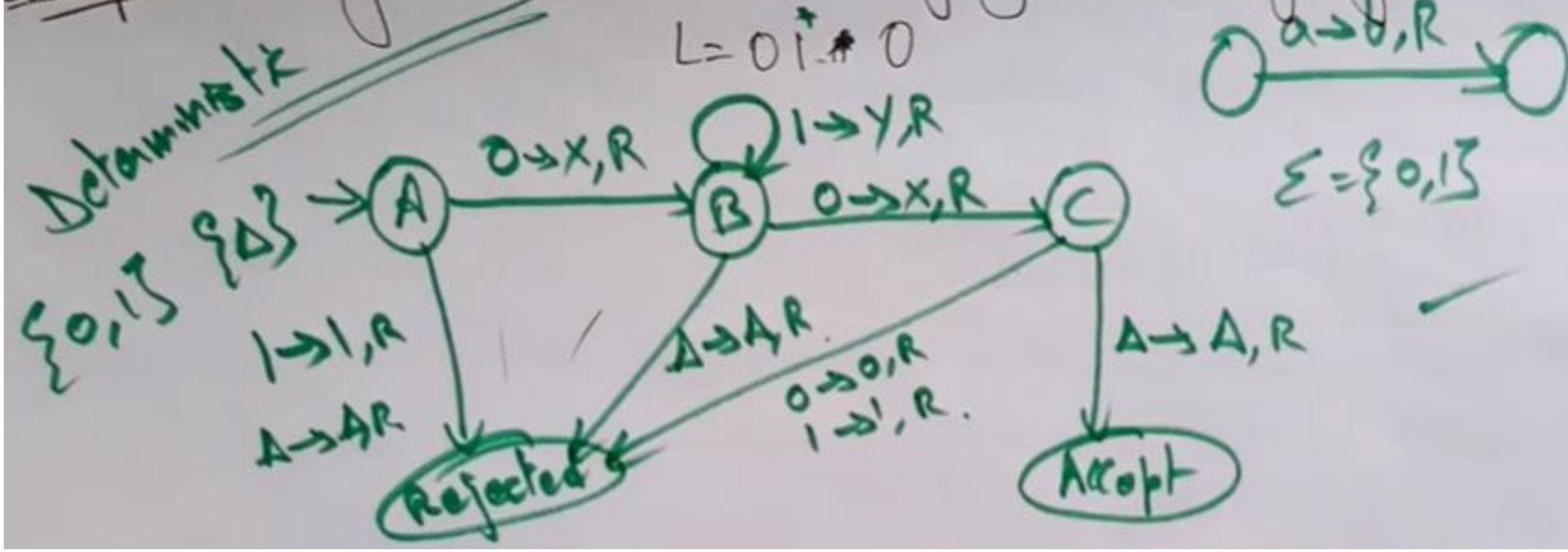


Eg: 011110 . . .



Example: Design a TM which recognize the language

$L = 01^*0$



Construct a TM to accept all strings Containing a substring "aba"

babaab, aba

Construct a TM to accept all strings Containi
substring "aba"

babaab,

ba

	a	b	B
q_0	(q_1, a, R)	(q_0, b, R)	
q_1			

construct a TM to a
substring "aba"

babaab
bab.

	a	b	B
q_0	(q_1, a, R)	(q_0, b, R)	
q_1	(q_1, a, R)	(q_2, b, R)	
q_2			

Construct a TM to accept all strings containing [illegible] substring "aba"

	a	b	B
q_0	(q_1, a, R)	(q_0, b, R)	
q_1	(q_1, a, R)	(q_2, b, R)	
q_2	(q_3, a, R)	(q_0, b, R)	

babaab, aba

babaa

baaba B

babbaba

Construct a TM to accept all strings containing a substring "aba"

	a	b	B
q_0	(q_1, a, R)	(q_0, b, R)	—
q_1	(q_1, a, R)	(q_2, b, R)	—
q_2	(q_3, a, R)	(q_0, b, R)	—
q_3	(q_3, a, R)	(q_3, b, R)	(q_4, B, R)
q_4	—	—	—

babaab, aba

babaabB

baabaB

babbaba

construct a TM to accept all strings containing a substring "aba"

babaab, aba

substring "aba"

babaab, ab

babaabB

baa

babba

	a	b	B
q_0	(q_1, a, R)	(q_0, b, R)	—
q_1	(q_1, a, R)	(q_2, b, R)	—
q_2	(q_3, a, R)	(q_0, b, R)	—
q_3	(q_3, a, R)	(q_3, b, R)	$(q_4, B, R$
q_4	—	—	—

substring "aba"

	a	b	B
q_0	(q_1, a, R)	(q_0, b, R)	—
q_1	(q_1, a, R)	(q_2, b, R)	—
q_2	(q_3, a, R)	(q_0, b, R)	—
q_3	(q_3, a, R)	(q_3, b, R)	$(q_4, B, R$
q_4	—	—	—

babaab, aba

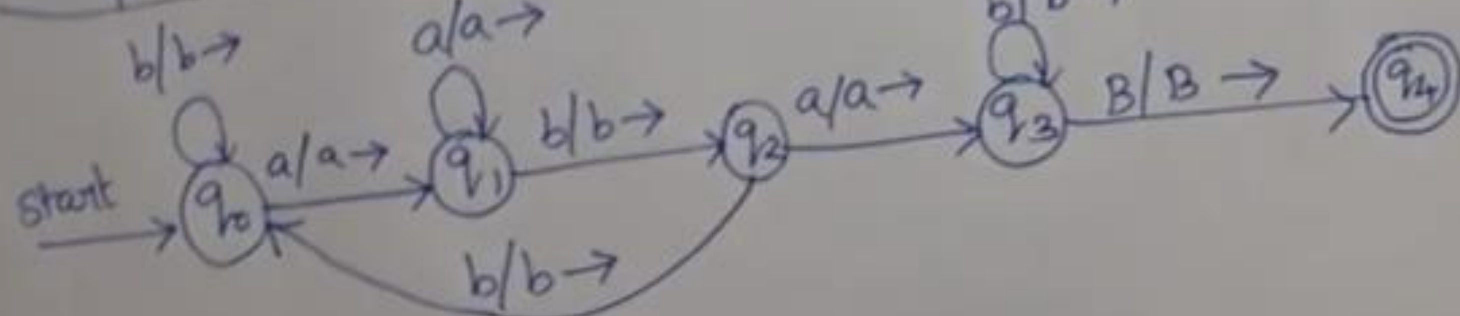
babaabB

baabaB

babbaba

Subs

q_4 | —



q_4 | —

TM, $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{a, b\}, \{a, b, B\}, \delta, q_0, B, \{q_4\})$

substring "aba"

	a	b	B
q_0	(q_1, a, R)	(q_0, b, R)	—
q_1	(q_1, a, R)	(q_2, b, R)	—
q_2	(q_3, a, R)	(q_0, b, R)	—
q_3	(q_3, a, R)	(q_3, b, R)	$(q_4, B, R$
q_4	—	—	—

babaab, aba

babaabB

baabaB

babbaba

$q_0 aaba \vdash a q_1 abaB$

$\vdash aa q_1 baB$

$\vdash aab q_2 aB$

$\vdash aaba q_3 B$

$\vdash aabaB q_4 B$

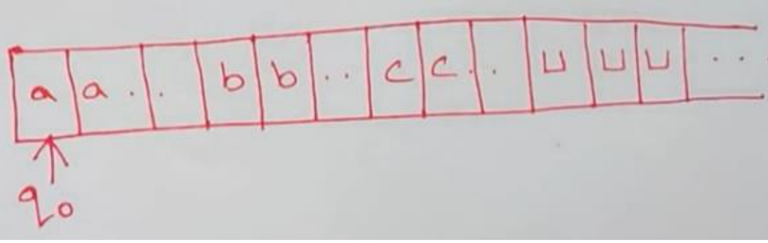
TM halts, since q_4 is a final state

TM accepts the input string aaba.

Start as q0 on a

Construction of a Turing Machine that accepts the language $L = \{a^n b^n c^n \mid n \geq 1\}$ over the alphabet $\Sigma = \{a, b, c\}$.

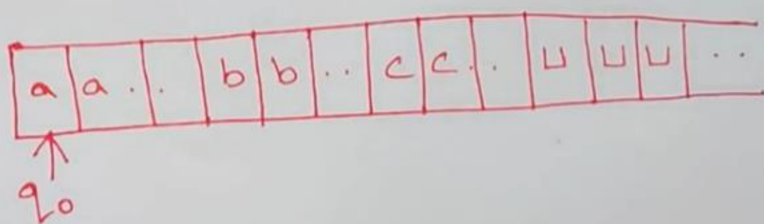
$w = \{abc, aabbcc, \ldots\}$



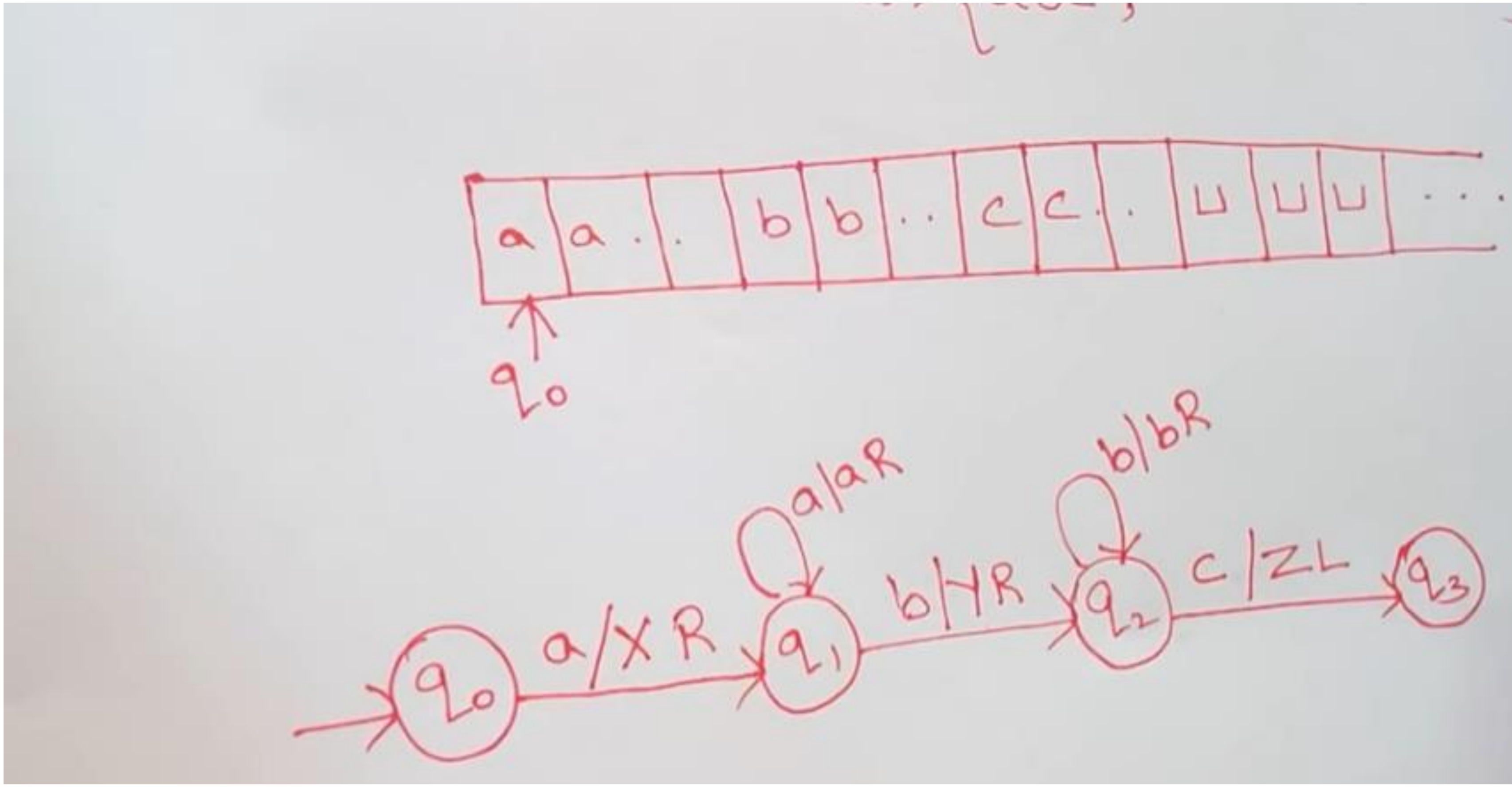
Cancel one a one b one c
Cancel second a second b second c
Repeat this in loop

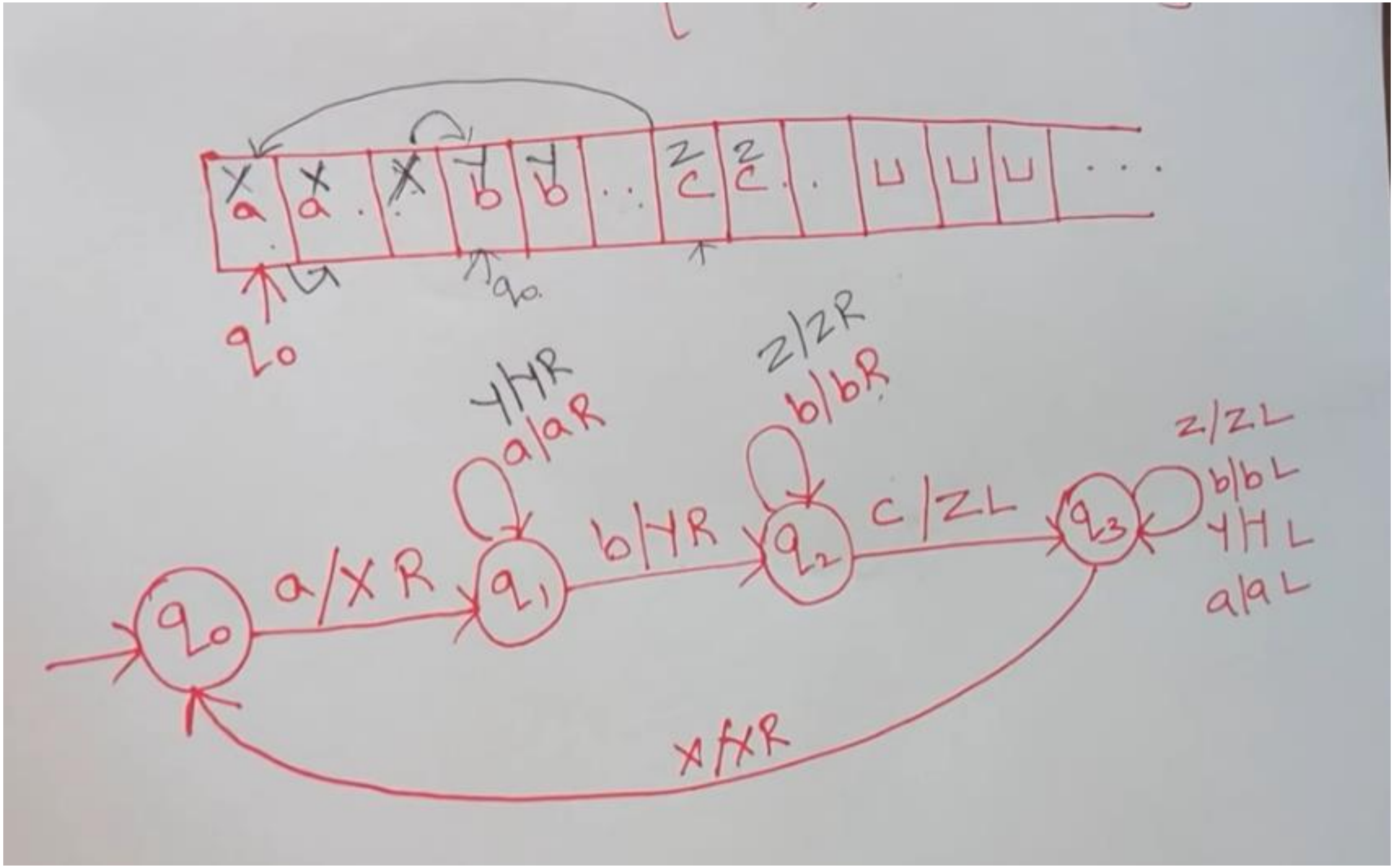
Construction of a Turing Machine that accepts the language $L = \{a^n b^n c^n \mid n \geq 1\}$ over the alphabet $\Sigma = \{a, b, c\}$.

$w = \{abc, aabbcc, \ldots\}$

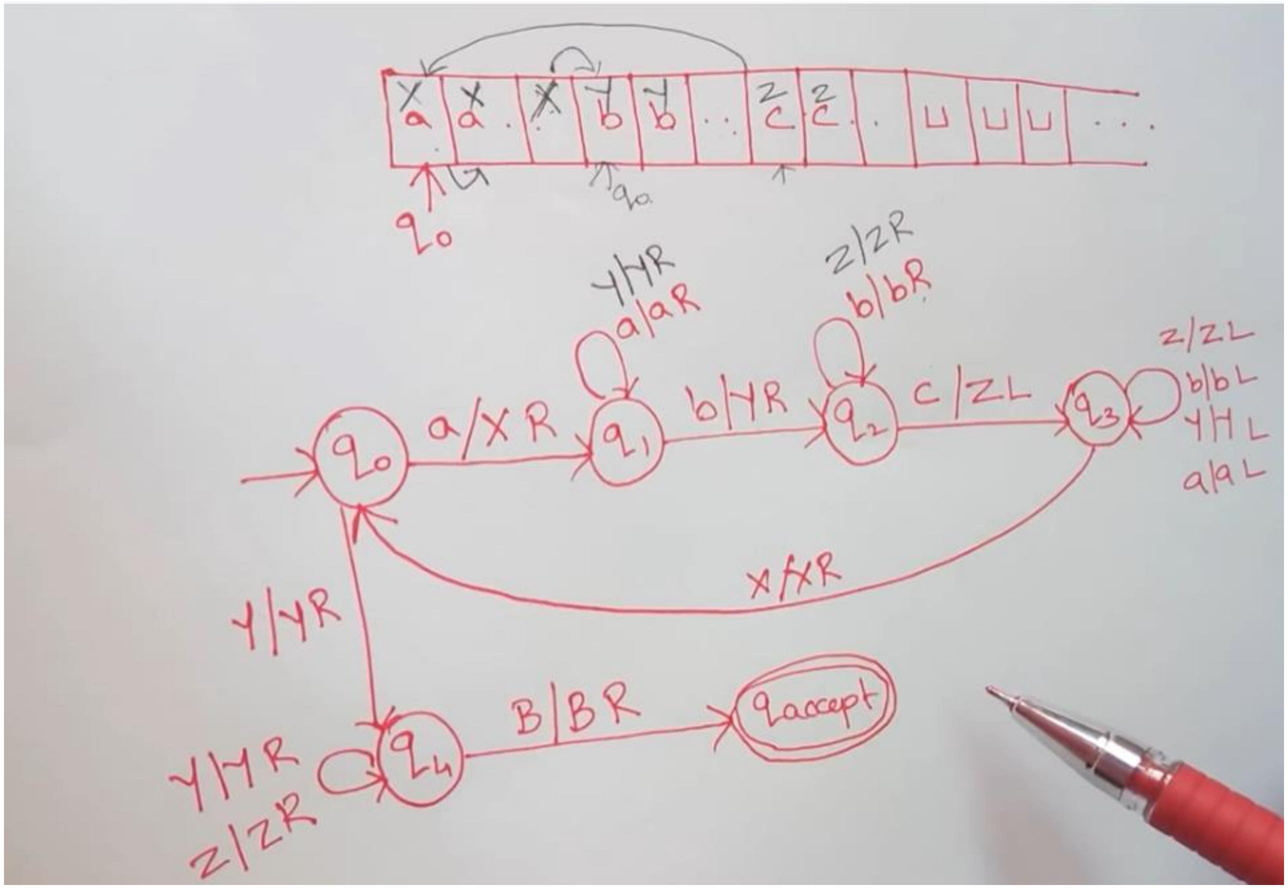


Cancel one a one b one c
Cancel second a second b second c
Repeat this in loop

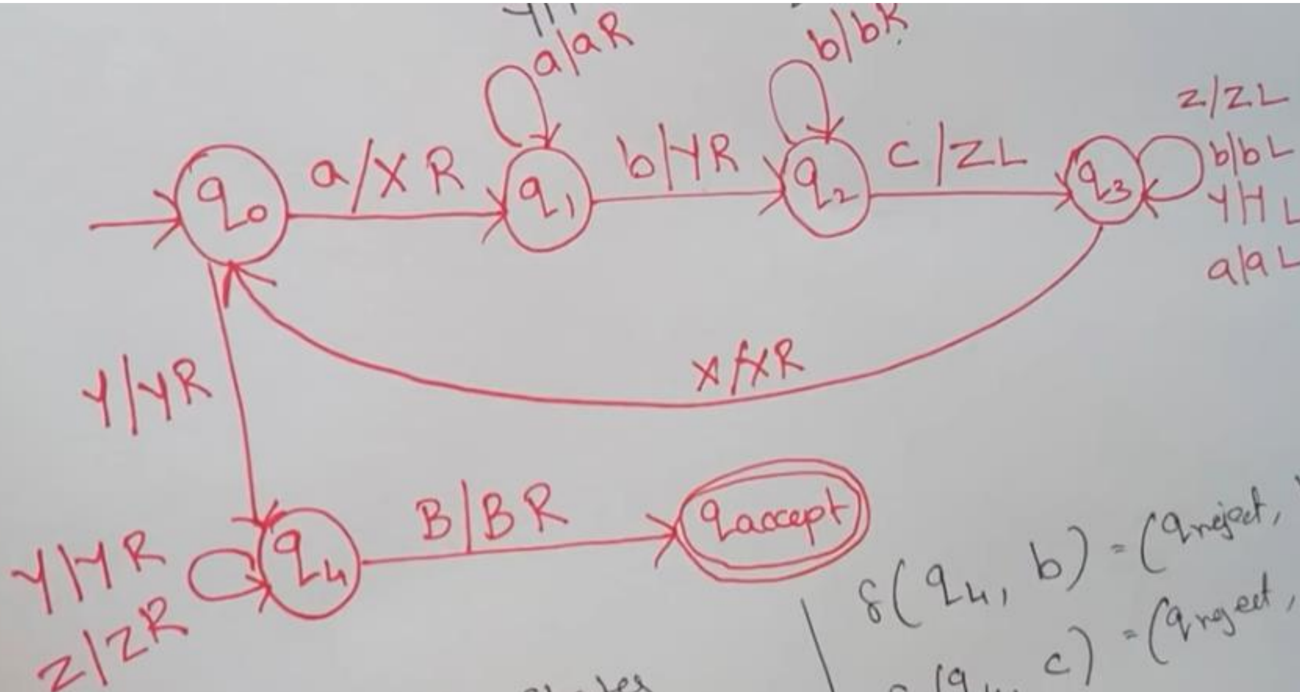

a a . . b b . . c c . . ⊔ ⊔ ⊔ . . .
q0
a/X R
a|aR
b|YR
b|bR
c/ZL
q0
q1
q2
q3



All a
All b



q0
Y|YR
a|aR
Z|ZR
b|bR
Z/ZL
b|bL
Y|YL
a|aL
a/XR
b|YR
C/ZL
X/XR
Y/YR
B|BR
qaccept
Y|YR
Z|ZR



Rejection states

$\delta(q_4, b) = (q_{reject}, b, R)$

$\delta(q_4, c) = (q_{reject}, c, R)$

$\delta(q_1, z) =$ ''

$\delta(q_1, c) =$ ''

$\delta(q_2, \sqcup) =$ ''

Construct a Turing Machine That accepts palindrome of the string over the alphabet

$\Sigma = \{a, b\} \Rightarrow \underline{ababa}$ $\cancel{a}\cancel{b}\cancel{b}\cancel{a}$

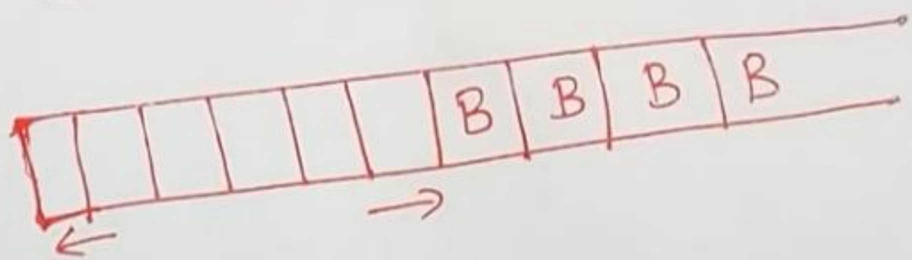
ababa.

Construct a Turing Machine ... palindrome of the string over the

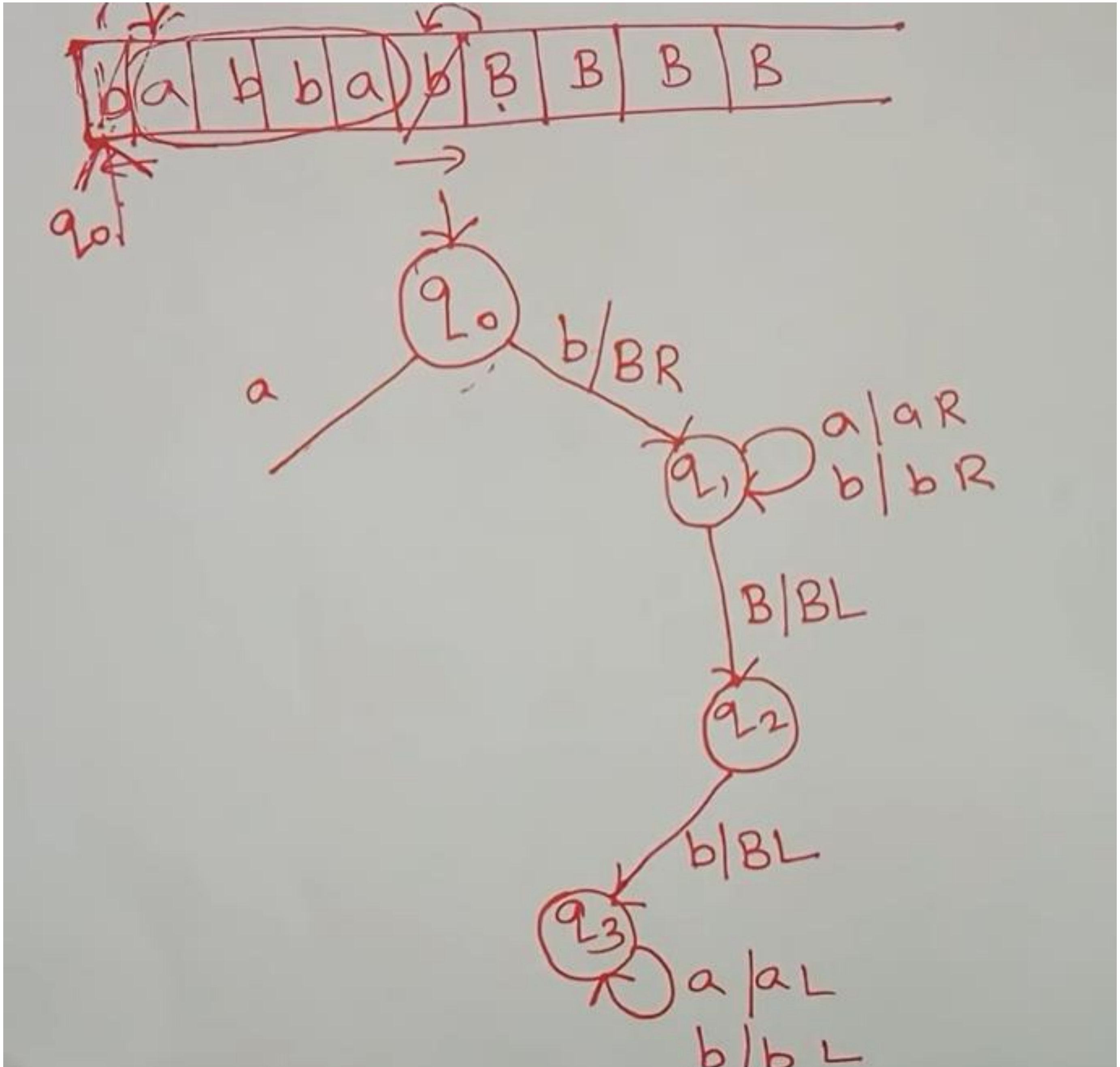
$\Sigma = \{a, b\} \Rightarrow \cancel{a}\cancel{b}a\cancel{b}\cancel{a}$ $\cancel{a}\cancel{b}\cancel{b}\cancel{a}$ $\underline{w}\ \underline{w^R}$

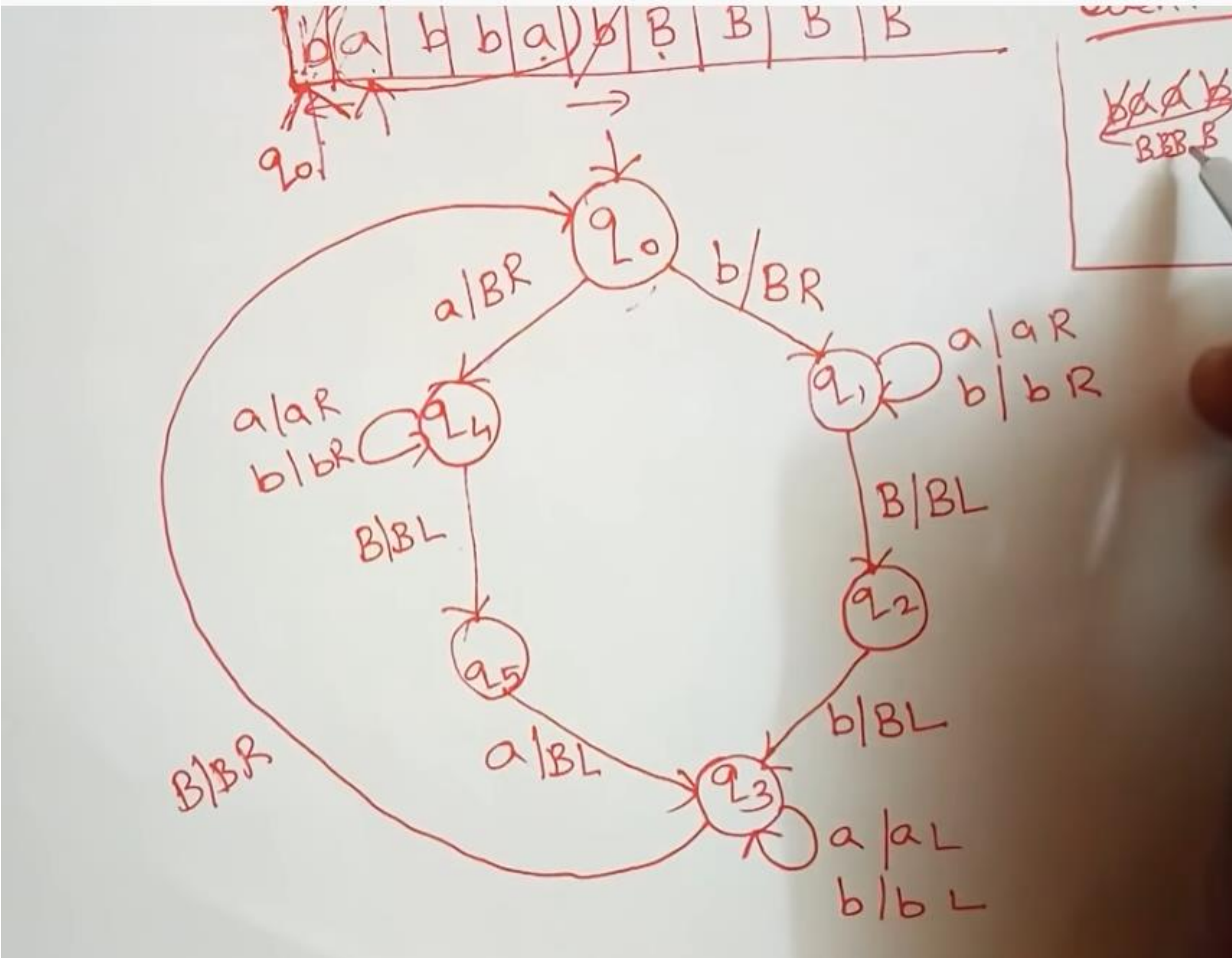
ababa.

$\cancel{b}\cancel{b}aa\cancel{b}\cancel{b}$

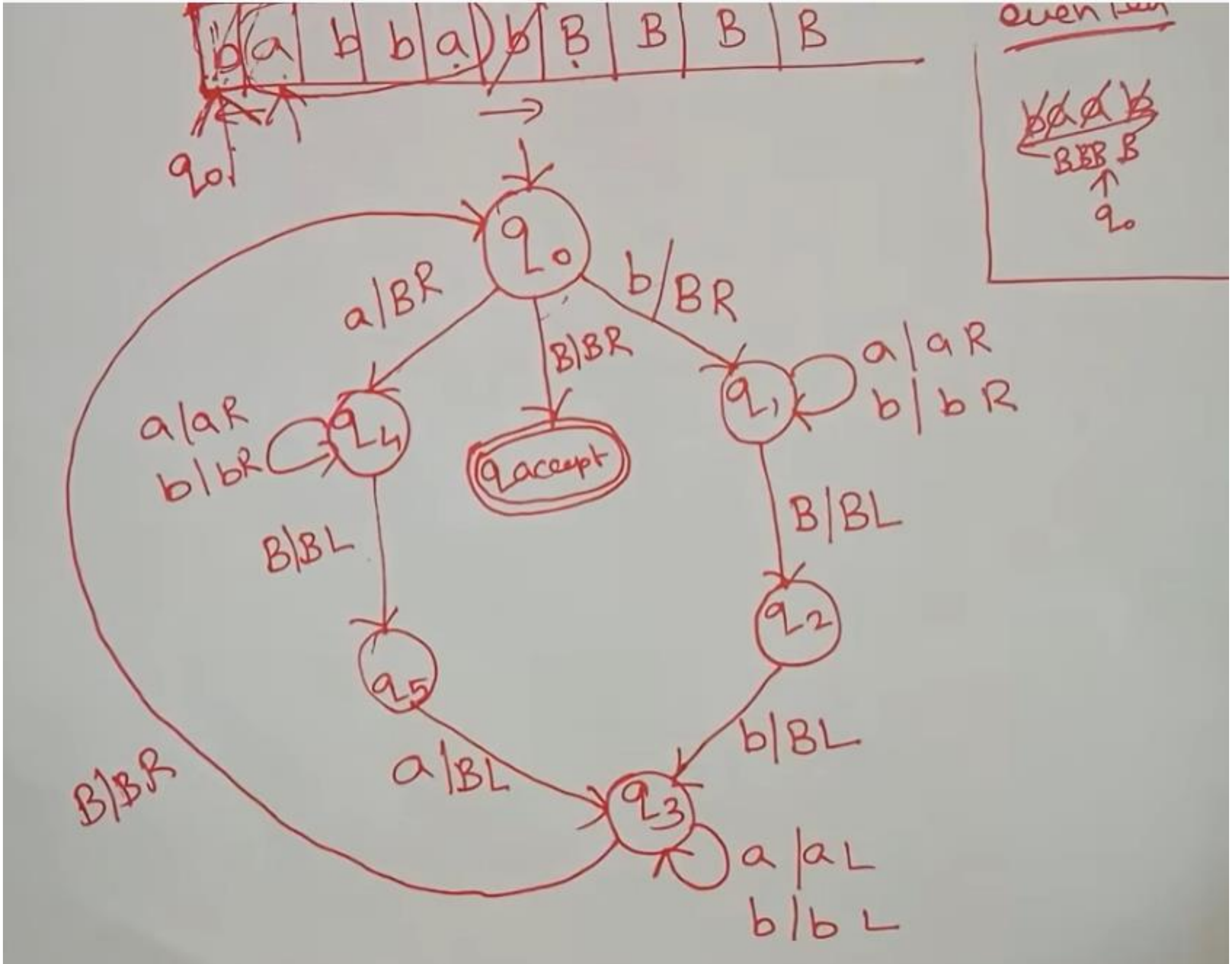


0....n
1....n-1
2....n-2
3...n-3


b a b b a b B B B B
q0
q0
b/BR
a
a/aR
b/bR
q1
B/BL
q2
b/BL
q3
a/aL
b/bL

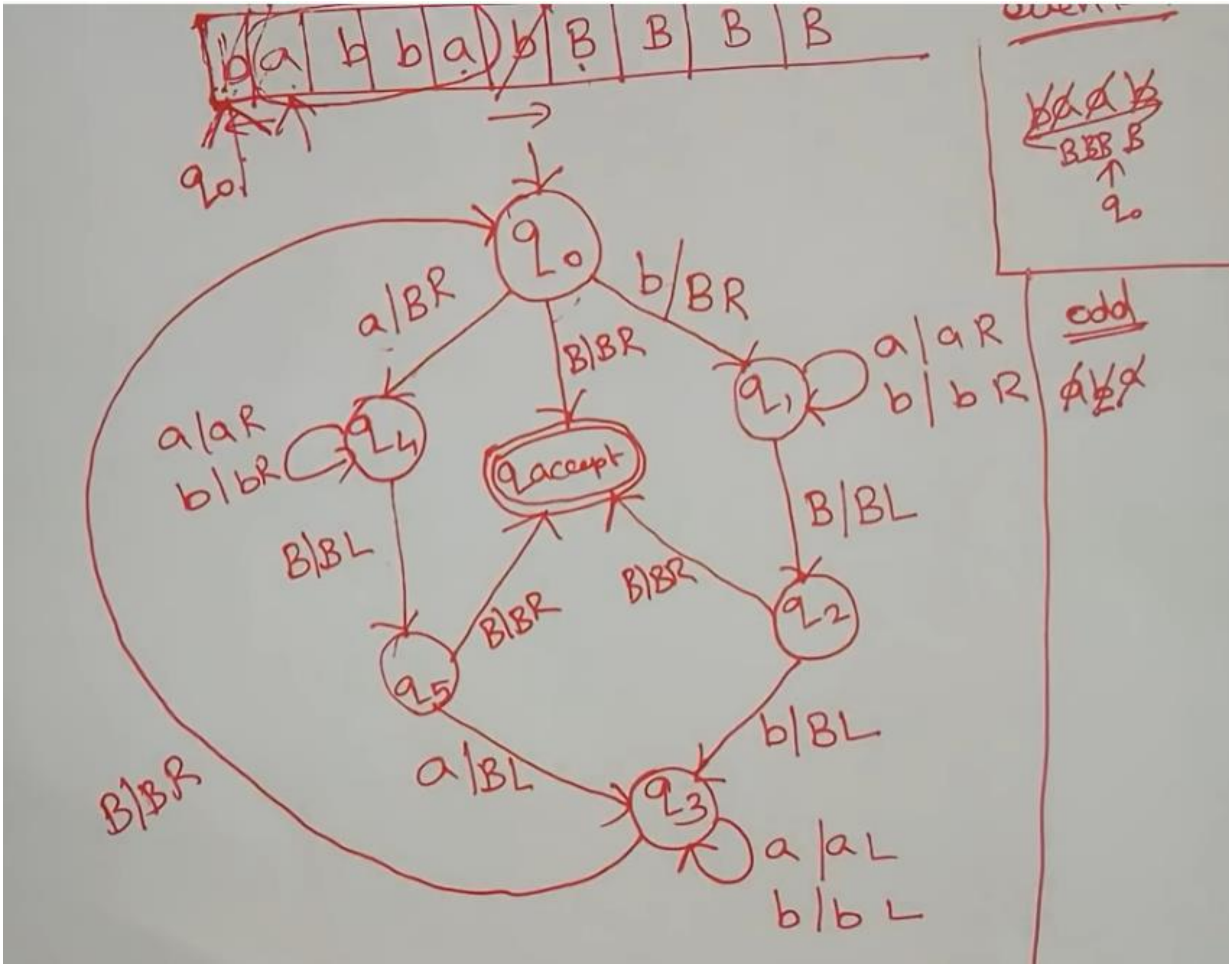


q0
a|BR
b|BR
a|aR
b|bR
B|BL
b|BL
a|BL
a|aL
b|bL
B|BR
BBB



b a b b a b B B B B
q0
q0
a|BR
b/BR
B|BR
a|aR
b|bR
q1
a|aR
b|bR
q4
qaccept
B|BL
B|BL
q2
q5
b|BL
a|BL
q3
B|BR
a|aL
b|bL
BBB B
q0

Til


b a b b a b B B B B
q0
q0
a|BR
b|BR
B|BR
q1
a|aR
b|bR
q4
a|aR
b|bR
qaccept
B|BL
B|BL
B|BR
B|BR
q2
q5
b|BL
a|BL
q3
a|aL
b|bL
B|BR
BBB B
q0
odd

3> Design a TM that accepts all palindromes over $\{a, b\}$

a b b a B
B   B

a b b a B
B B B B

(i) a b b a B
B B B B

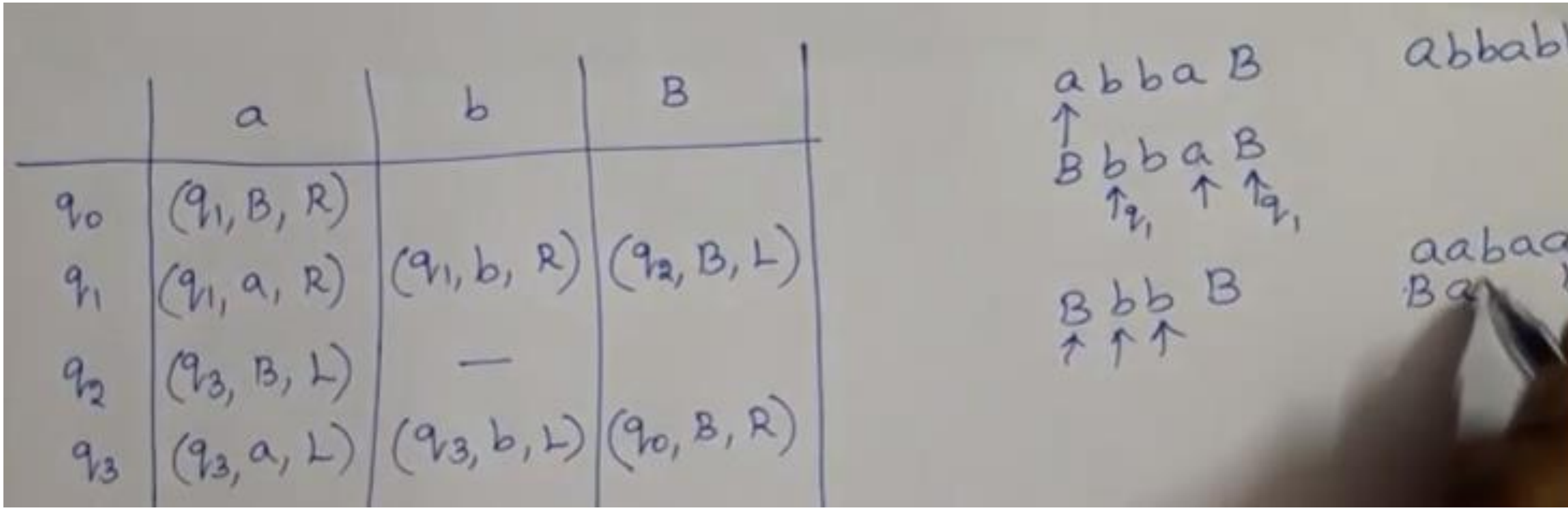
a b a a b a
B B B B B B

(ii) a b a b a
B B B B B

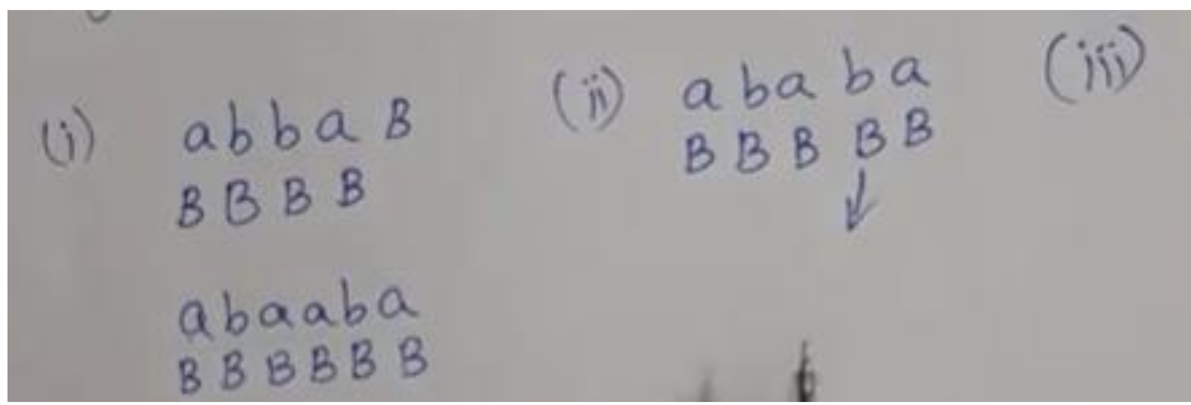
(iii) a b b b a
B B B B B

Even length palindrome

Odd length palindrome



	a	b	B
q_0	(q_1, B, R)		
q_1	(q_1, a, R)	(q_1, b, R)	(q_2, B, L)
q_2	(q_3, B, L)	—	
q_3	(q_3, a, L)	(q_3, b, L)	(q_0, B, R)



abba B
B ..
B B
BB B
BBBB

	a	b	B
q_0	(q_1, B, R)	(q_4, B, R)	(q_6, B, R)
q_1	(q_1, a, R)	(q_1, b, R)	(q_2, B, L)
q_2	(q_3, B, L)	—	
q_3	(q_3, a, L)	(q_3, b, L)	(q_0, B, R)
q_4	(q_4, a, R)	(q_4, b, R)	(q_5, B, L)
q_5	—	(q_3, B, L)	

a b b a B

B b b a B
q_1 q_1

B b b B

B B b B

B B B B
q_0

a a b a a
B a b a B

b a a b

This is for even length palindrome

(i) a b b a B
B B B B

a b a a b a
B B B B B B

(ii) a b a b a
B B B B B

(iii)

For odd length palindrome

q_0	(q_1, B, R)	(q_4, B, R)	(q_6, B, R)
q_1	(q_1, a, R)	(q_1, b, R)	(q_2, B, L)
q_2	(q_3, B, L)	—	(q_6, B, R)
q_3	(q_3, a, L)	(q_3, b, L)	(q_0, B, R)
q_4	(q_4, a, R)	(q_4, b, R)	(q_5, B, L)
q_5	—	(q_3, B, L)	(q_6, B, R)
q_6	—	—	—

$$TM, M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{a,b\}, \{a,b,B\}, \delta, q_0, B, \{q_6\})$$

q_1 q_1

B b b B

B B b B

B B B B

q_0

aabaa

B aba B

baab

ababa

B B a B B

q_0

B B B B

q_2 q_1

aba

B B B

Example

$$q_0 aba \vdash Bq_1 baB$$
$$\vdash Bbq_1 aB$$
$$\vdash Bbaq_1 B$$
$$\vdash Bbq_2 aB$$
$$\vdash Bq_3 bBB$$
$$\vdash q_3 BbBB$$
$$\vdash Bq_0 bBB$$
$$\vdash BBq_4 BB$$
$$\vdash Bq_5 BBB$$
$$\vdash BBq_6 BB$$

Halting problem

⇒ Halting problem is undecidability. It is not a problem, it ju
asks question :- "is it possible to tell whether a given m/c
will halt for some given i/p"

Eg: input A T.M & i/p string w

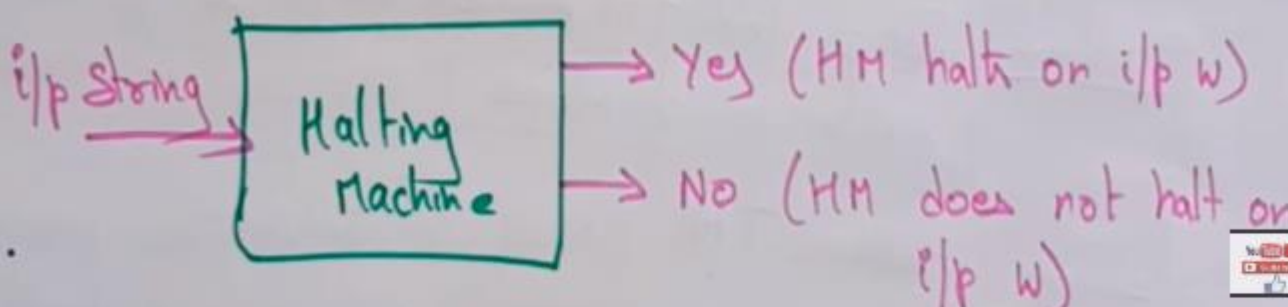
Problem: Does the TM finish computing of the string w in
finite no: of steps? The answer must be Yes or no

Proof :- Assum T.M exists to solve this problem & then
we will show it is contradicting itself

We will call this T.M as a Halting m/c that produce a 'Yes' or 'no' in a finite amount of time.

→ If the halting m/c finishes in a finite amount of time, o/p comes as 'yes' otherwise as 'no'.

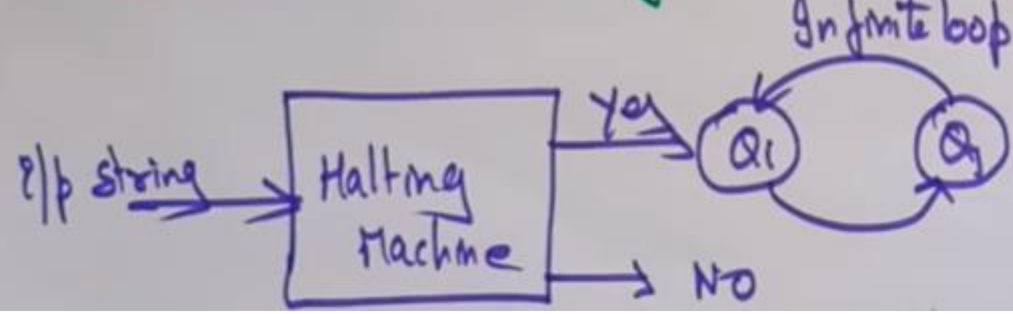
The block dig of Halting m/c :-



Now we will design an inverted halting machine as:-

- If H returns Yes, then loop forever
- If H returns No, then Halt.

The drg for inverted Halting m/c:-



After that a m/c $(HM)_2$ which i/p itself constructed as:-

- If $(HM)_2$ halts on i/p, loop forever
- Else, halt.