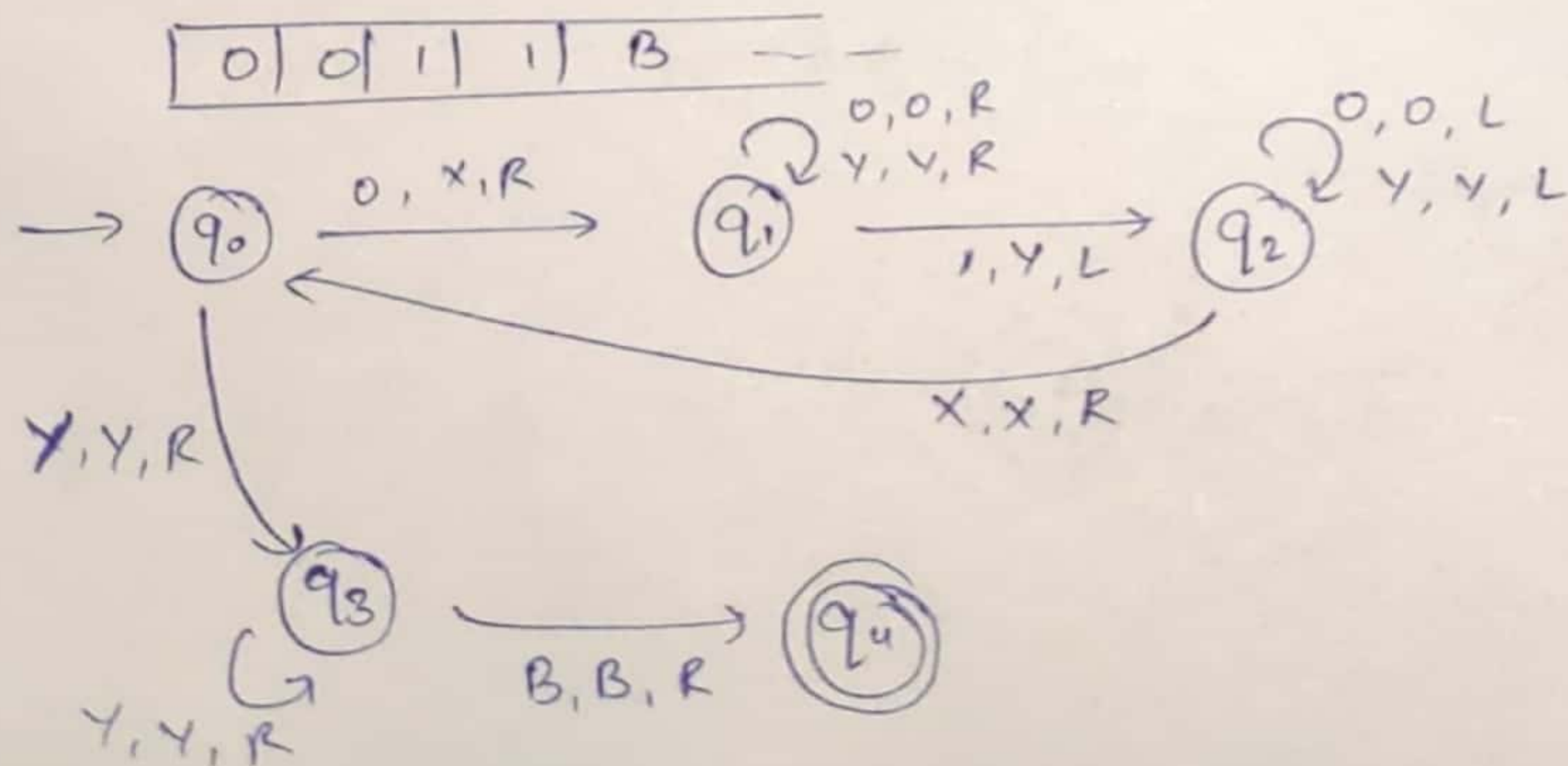


$Q = \{q_0, q_1, q_2, q_3, q_4\}$
 $\Sigma = \{1, \#\}$
 $T = \{0, 1, \#, X, Y\}$

Transition Table:

State	0	1	#	X	Y
$\rightarrow q_0$	$q_1, 0, L$	$q_0, 1, R$	$q_0, \#, R$	-	-
q_1	-	q_2, X, L	q_2, Y, L	-	-
q_2	$q_3, 0, R$	$q_2, 1, L$	$q_2, \#, L$	-	-
q_3	-	$q_3, 1, R$	$q_3, \#, R$	$q_4, 1, L$	$q_4, \#, L$
q_4	-	-	-	-	-

2.)



$$Q = \{q_0, q_1, q_2, q_3, q_4\}$$

Initial state = q_0

Final state = q_4

$$\Sigma = \{0, 1\}$$

$$\Gamma = \{0, 1, x, y, B\}$$

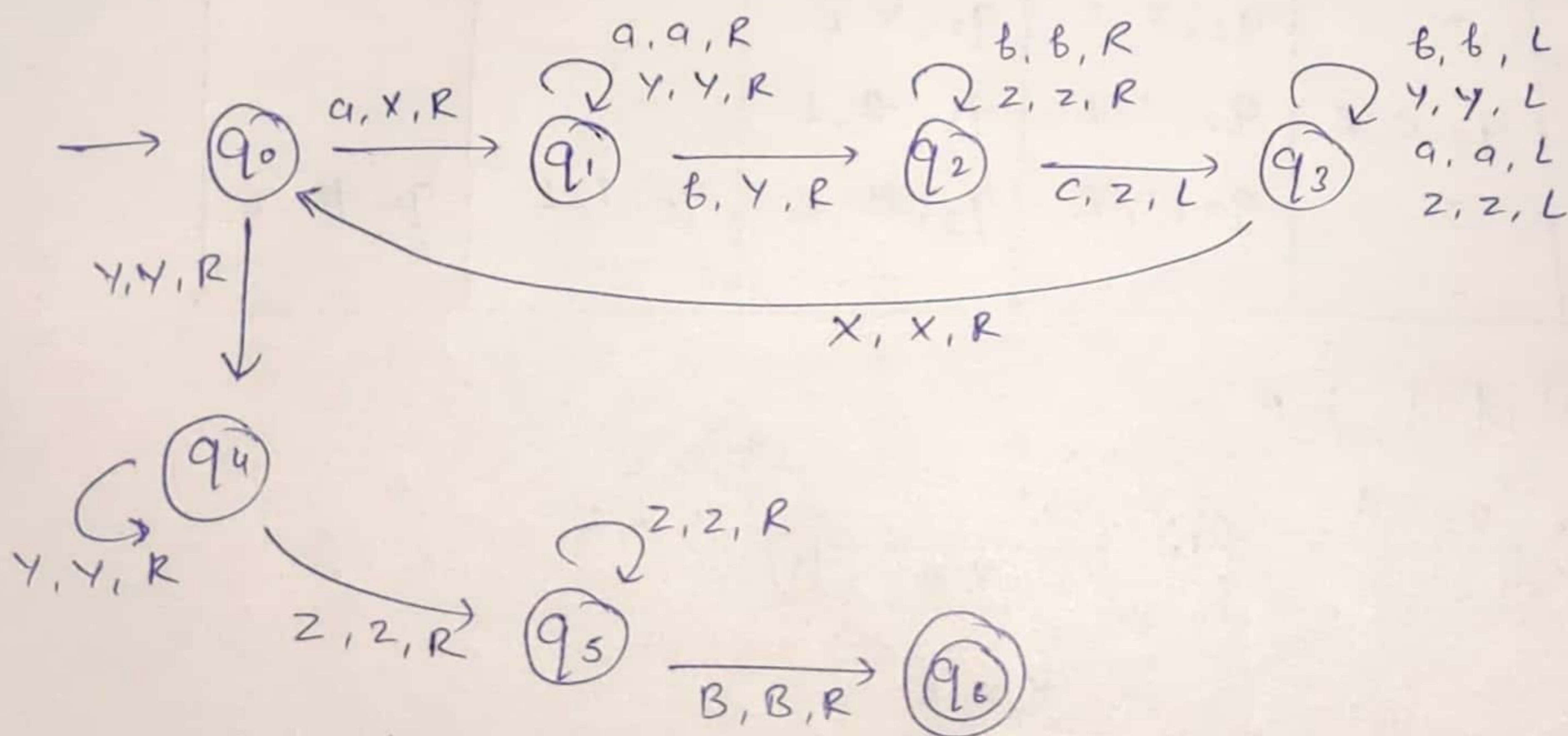
State	0	1	x	y	B
$\rightarrow q_0$	q_1, x, R	—	—	q_3, y, R	—
q_1	$q_1, 0, R$	q_2, x, L	—	q_1, y, R	—
q_2	$q_2, 0, L$	—	q_0, x, R	q_2, y, L	—
q_3	—	—	—	q_3, y, R	q_4, B, R
$*q_4$	—	—	—	—	—

3.)

$$a^n b^n c^n \mid n \geq 1$$

if $n=2$

a | a | b | b | c | c | B |



$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$

Initial state = q_0

Final state = q_6

$\Gamma = \{a, b, c, x, y, z, B\}$

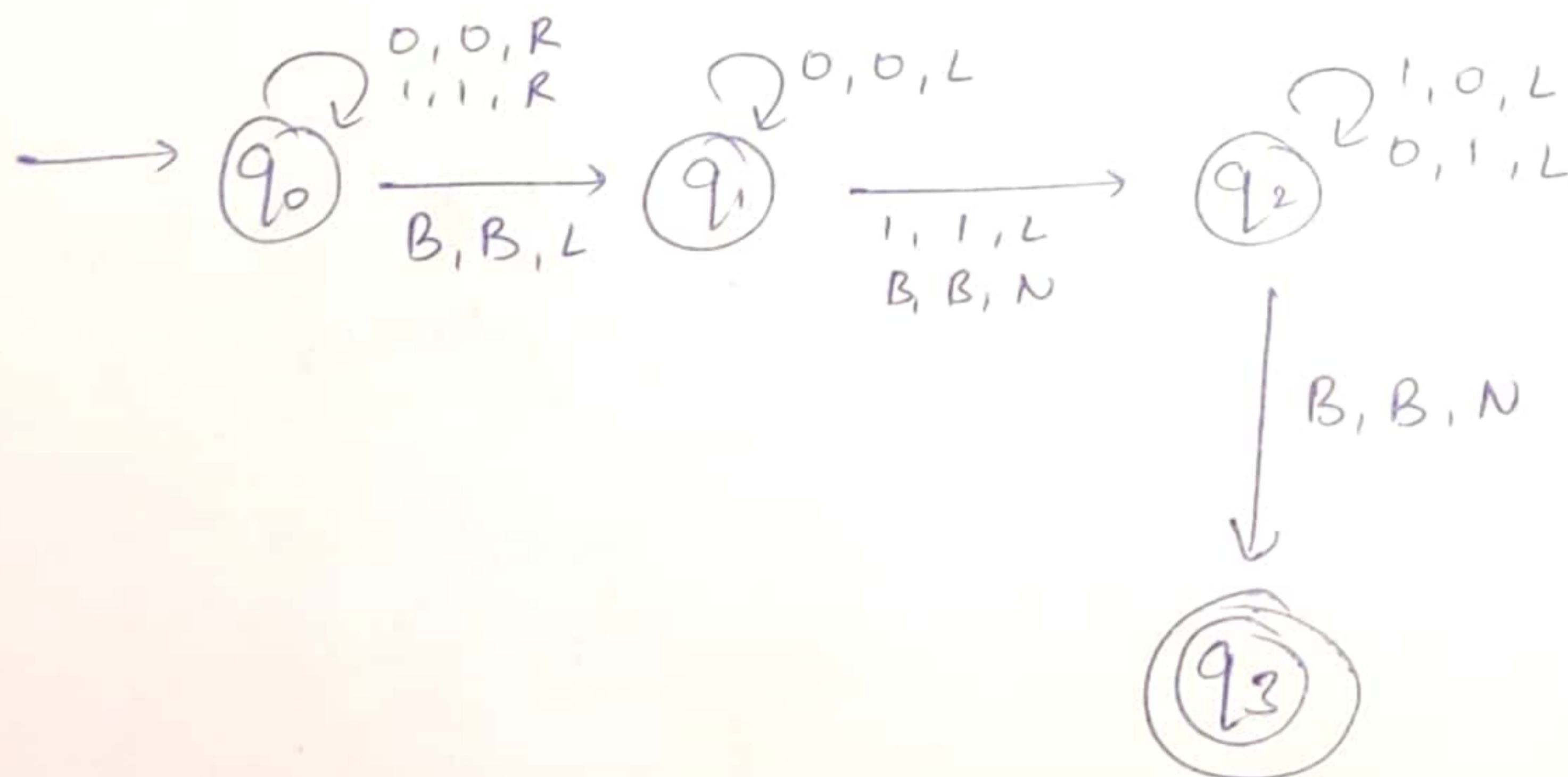
$\Sigma = \{a, b, c\}$

Transition Table :-

State	a	b	c	x	y	z	B
$\rightarrow q_0$	q_1, x, R	—	—	—	q_4, y, R	—	—
q_1	q_1, a, R	q_1, y, R	—	—	q_1, y, R	—	—
q_2	—	q_2, b, R	q_3, z, L	—	—	q_2, z, R	—
q_3	q_2, a, L	q_3, b, L	—	q_0, x, R	q_3, y, L	q_3, z, L	—
q_4	—	—	—	—	q_4, y, R	q_5, z, R	—
q_5	—	—	—	—	—	q_5, z, R	q_6, B, R
q_6	—	—	—	—	—	—	—

4.)

0 0 0 0 0 1 — —



$$Q = \{q_0, q_1, q_2, q_3\}$$

$$\Sigma = \{0, 1\}$$

$$\text{Initial state} = q_0$$

$$\text{Final state} = q_3$$

$$\Gamma = \{0, 1, R, L, B, N\}$$

5.) a) $L(G) = \phi$

Decidable - we can check if any variable generates terminals using a finite marking algorithm. If start symbol is not generating, language is empty.

b) $L(G) = \Sigma^*$

Undecidable - universality of CFG is a known undecidable problem proven via reduction from PCP. No algorithm can decide it for all CFGs.

c) $L(M)$ is regular

Undecidable - by Rice's Theorem, regularity is a non-terminal property of Turing machine languages. Hence, no Turing M/C can decide it.

d) $L(A) = L(N)$

Decidable - convert NFA to DFA and test DFA equivalent via symmetric difference emptiness.

All steps are algorithmic.

6) PCP

$$A = \{100, 0, 1\}$$

$$B = \{1, 100, 00\}$$

1
100

A_1

2
0

A_2

3
1

A_3

1

B_1

100

B_2

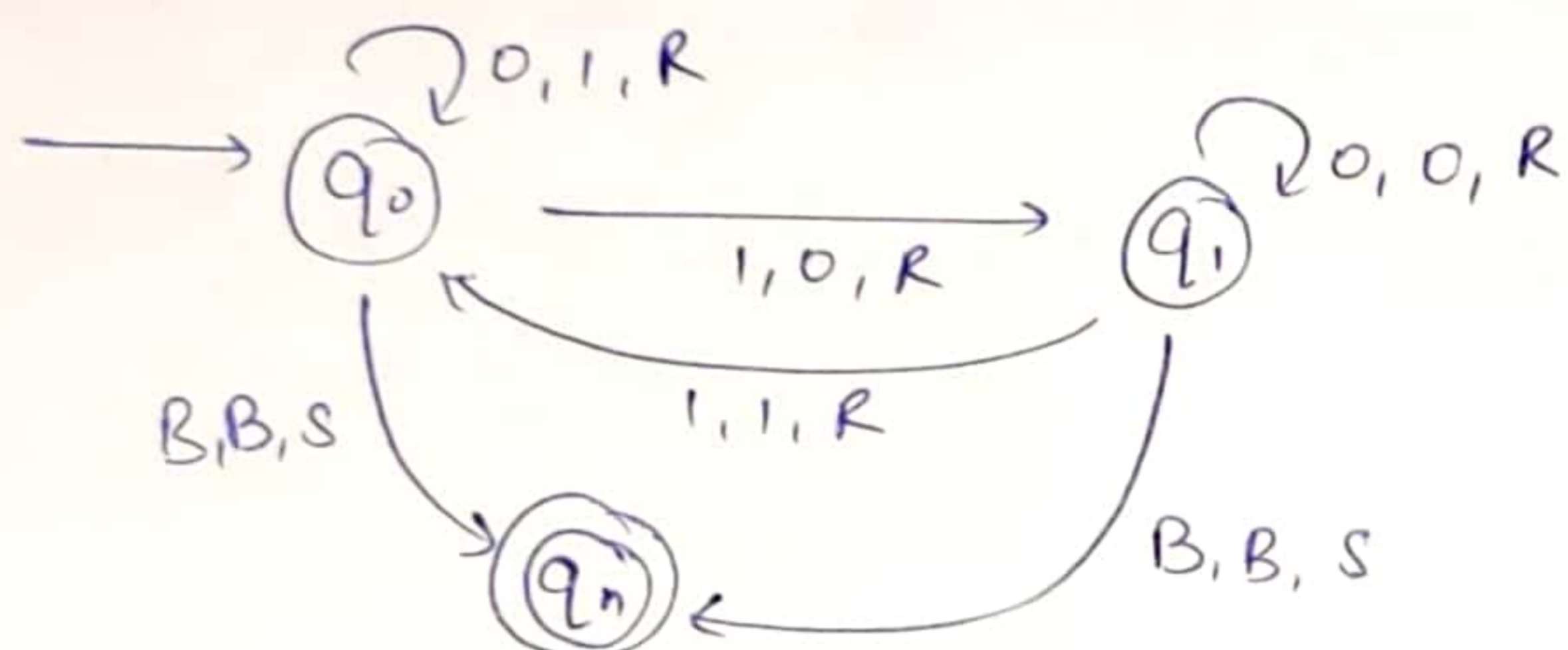
00

B_3

A	1	2	23	3
	100	0	0	1
B	1	100	100	00

Solution doesn't exist.

7)



$$Q = \{q_0, q_1, q_n\}$$

$$\Sigma = \{0, 1\}$$

$$\Gamma = \{0, 1, B\}$$

$$I.S = q_0$$

$$F.S = q_n$$

Transition State ..

State	0	1	B
$\rightarrow q_0$	$q_0, 1, R$	$q_1, 0, R$	q_n, B, S
q_1	$q_1, 0, R$	$q_0, 1, R$	q_n, B, S
q_n	—	—	—

8.) The Post Correspondence Problem (PCP) asks whether, for two lists of strings (list A, list B) a sequence of indices can be found such that concatenation of string from list B. The problem is undecidable, the undecidability is proven by a reduction from the Halting problem. A Turing machine (M) and its input can be systematically converted into a PCP instance has a solution if only if (M) accepts (w). Since no algorithm can decide Halting problem, no algorithm can decide the PCP.

Example: $x = (b, bab^3, ba)$

$y = (b^3, ba, a)$

x_1 x_2 x_3

b	bab ³	ba
---	------------------	----

y_1 y_2 y_3

b ³	ba	a
----------------	----	---

	2	1	1	3
x	babbb	b	b	ba
y	ba	bbb	bbb	a

All a's and b's match
 $m = 4$

Solution exists.

9) Let $L_u = \{ \langle M, w \rangle \mid M \text{ accepts } w \}$.

L_u is r.e.: Simulate M on w by detailing if M accepts the simulator halts and accepts $\langle M, w \rangle$.

If L_u were recursive we should decide the halting problem on input $\langle M, w \rangle$ build ' M ' that on any input simulates M on w and accepts if that simulator halts and accepts if $\langle M', u \rangle \in L_u$.

Since Halt is undecidable, L_u cannot be recursive.

Define the diagonal language

$L_d = \{ \langle M \rangle \mid M \text{ does not accept } \langle M \rangle \}$

The complement of

$K = \{ \langle M \rangle \mid M \text{ accepts } \langle M \rangle \}$

K is r.e. K is not recursive.

If $L_d = \bar{K}$ were then both K and \bar{K} would be r.e. so K would be decidable - contradiction.

Hence L_d is neither r.e. nor recursive.

10) Repeated ques.

12) By Contradiction method,

Assumption: Assume a Turing m/c H exists solves the halting problem.

$H(\langle M \rangle, w)$ halts and tells us whether.

m/c H halts on input w .

Construction of D: Construct a new m/c D, that takes an input $\langle M \rangle$ and uses H as a sub routine.

D asks H: "Does M halt on its own description $\langle M \rangle$?"

D is programmed to do the opposite of what H does.

→ If H says M halts on $\langle M \rangle$, then D loops

→ If H says M loops on $\langle M \rangle$, then D halts.

Contradiction: when run D on its own description $\langle D \rangle$,

→ if D halts on $\langle D \rangle$: D must loop (contradiction)

→ If D loops on $\langle D \rangle$: D must halt (contradiction)

Hence, this halting problem is insoluble.

13) \boxed{b} $\boxed{bab^3}$ \boxed{ba}
 A_1 A_2 A_3

$\boxed{b^3}$ \boxed{ba} \boxed{a}
 B_1 B_2 B_3

	2	1	1	3
A	bab^3	b	b	ba
B	ba	b^3	b^3	a

$m = 4$

Hence, Solution exists for this PCP.

14.) let $L \neq \emptyset = \langle M \rangle \mid L \in M \rangle \neq \emptyset$

Construct a semi-decision procedure S for input $\langle M \rangle$.

Algorithm: enumerate all strings w_1, w_2, \dots for stage $t = 1, 2, \dots$ simulate M for t steps on each w_1, \dots, w_t .
If any simulation accepts, halt and accept $\langle M \rangle$.

→ If $L(M) \neq \emptyset$ there is some w_k accepted in T steps, at stage $t = \max(k, T)$ the simulation finds the acceptance and S accepts.

→ If $L(M) = \emptyset$ no simulation ever accepts, so S never accepts.

Thus, S semi-decides $L \neq \emptyset$, so $L \neq \emptyset$ is recursively enumerable.

15.) Let L_1, L_2 be recursive (decidable). So there exist deciders (total Turing machine) M_1 and M_2 that always halt and satisfy:

→ $M_1(x) = \text{accept}$ iff $x \in L_1$,

→ $M_2(x) = \text{accept}$ iff $x \in L_2$

$L_1 \cup L_2$ - build decider M_3 for input x :

1) Run on x . If it accepts, accept.

2) Otherwise run M_2 on x . If it accepts, accept.

3) If both reject, reject.

Termination:- M_1 and M_2 always halt, so M_0 always halts.

Correctiveness: If $x \in L_1 \cup L_2$ then atleast one of M_1, M_2 accepts and M_0 accepts, if $x \notin L_1 \cup L_2$ both reject, so M_0 rejects. Thus $L_1 \cup L_2$ is recursive.

Intersection: $L_1 \cap L_2$ - build decider M_n for input x :

- 1) Run M_1 on x . If it rejects, reject.
- 2) Otherwise run M_2 on x . If it accepts, accept, else reject.

Termination: Same reason - both sub machines halt, so M_n halts.

Correctness: M_n accepts exactly when both M_1 and M_2 accept i.e exactly when $x \in L_1 \cap L_2$.
So $L_1 \cap L_2$ is recursive

Ques 16 Define Post correspondence problem. Let $\Sigma = \{0, 1\}$. Let A and B be the lists of three strings, each defined as

	List A	List B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

Does this PCP have a solution?

- Tile 3 has $A = 10$ and $B = 0$, so it cannot start (top begins with 1, bottom with 0 \rightarrow mismatch).
- Tile 2 ends with 1 on top but 0 on bottom, so it cannot end any matching sequence.
- Trying possible combination starting with tile 1:
 $\rightarrow 1 \rightarrow$ top = 1, bottom = 111 (prefix mismatch)
 $\rightarrow 1, 1$ or $1, 3$ or $1, 2, 3$ etc, also fail because prefixes diverge.

No sequence makes A-side = B-side

This PCP has no solution.

Ques 17 Does a PCP solution exist for the following set?

$(10, 101), (01, 100), (0, 10), (100, 0), (1, 010)$

- The first tile must have the same starting symbol on top and bottom. Only tile 1 $(10, 101)$ qualifies \rightarrow solⁿ must start with tile 1.

- But tile 1 already creates mismatch:

Top = 10

Bottom = 101

Bottom has an extra 1 which no other tile can remove or balance without causing further prefix mismatch.

- Trying possible short sequences starting with tile 1 (like 1-3, 1-4, 1-1-3, etc) all fail because the prefixes never match.

This PCP instance has no solution.

Ques 19 Design a Turing machine for Reversing the given string {abb}.

TM marks each symbol from left ($a \rightarrow x, b \rightarrow y$), moves to a separator #, writes the same symbol at the end, returns, and repeats. After all symbols are marked, it erases the original part & #.

Key Transitions:

1. Create #:

$(q_0, a/b \rightarrow R), (q_0, B \rightarrow \text{write } \#, L \rightarrow q_1)$

2. Mark next symbol:

$(q_1, a \rightarrow x, R, q_2), (q_1, b \rightarrow y, R, q_2)$

3. Append after #:

Move to #, then write a or b to its right.

4. Cleanup:

Erase $x, y, \#$ and halt.

O/P for input abb

Appending order = a, b, b \rightarrow final reversed string:

bba

Ques 21 Show that any PSPACE-hard language is also NP-hard.

Proof:

1. $NP \subseteq PSPACE$

Every NP problem can be solved using polynomial space (because a nondeterministic polynomial-time TM uses only polynomial space).

2. Definition of PSPACE-hard

A language L is ~~space~~ PSPACE-hard if every language in PSPACE reduces to L using a polynomial-time reduction.

3. Since $NP \subseteq PSPACE$, every NP language is also in PSPACE.

4. Therefore, if all PSPACE languages reduce to L , then in particular all NP languages also reduce to L .

\therefore Every NP problem reduces to a PSPACE-hard problem, every PSPACE-hard language is automatically NP-hard.
Hence proved.

Ques 25 Check the following is PCP or not.

I	A	B
1	00	0
2	001	11
3	1000	011

Try to build matching strings:

- Using tile 1: $A = 00$, $B = 0 \rightarrow$ mismatch.
- Extend with tile 1 again: $A = 0000$, $B = 00 \rightarrow$ still

Date.....

mismatch.

- Trying combinations with tiles 2 or 3 always leaves extra unmatched 0's / 1's on one side.
- No sequence can make the A-side and B-side equal.

No PCP solⁿ exists for this set.