

Initial state =  $q_0$

Final state =  $q_4$

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

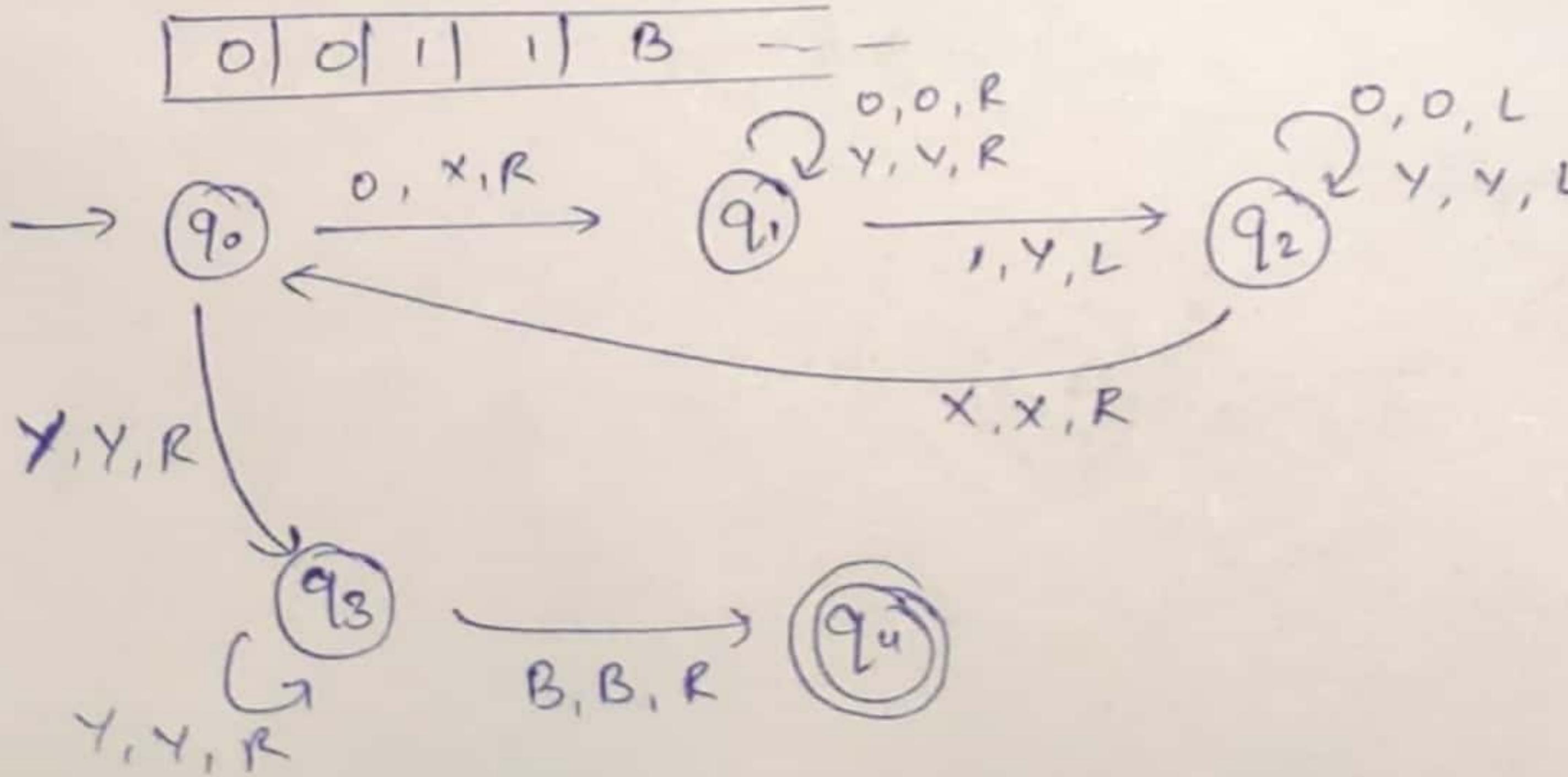
$$\Sigma = \{1, \#\}$$

$$T = \{0, 1, \#, X, Y\}$$

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 =  $q_4$

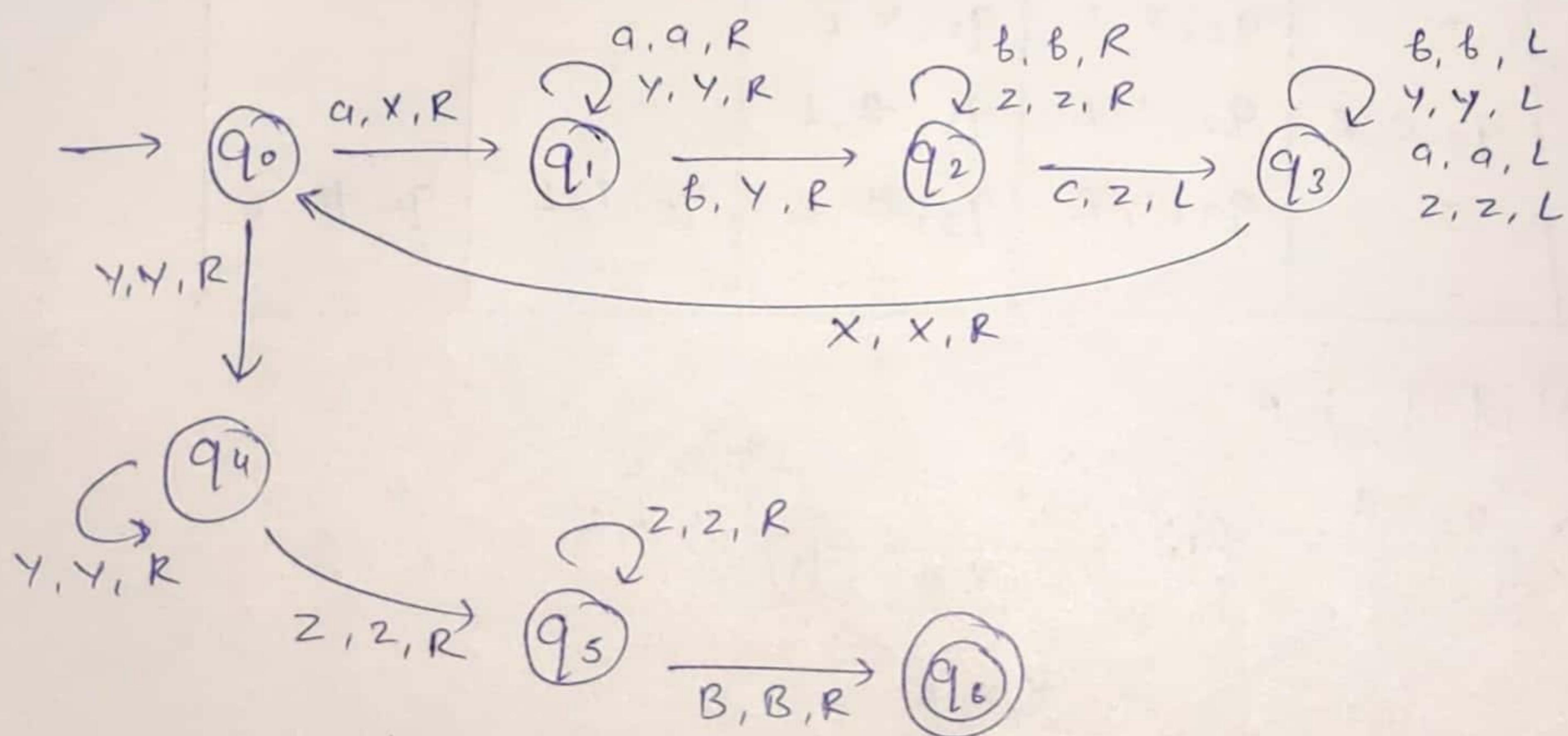
$$\Sigma = \{0, 1\}$$

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

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

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 =  $q_6$

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

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

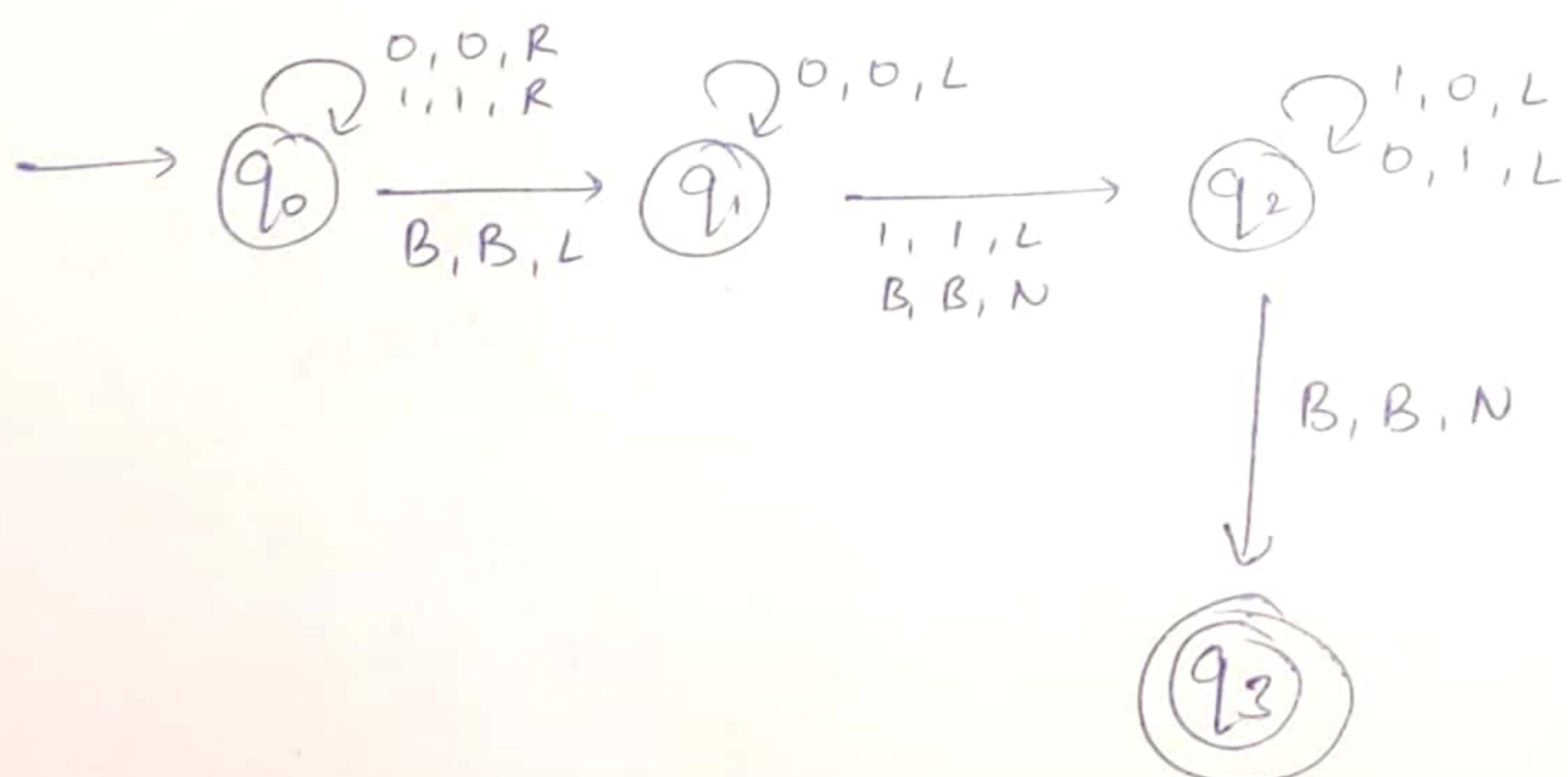
Transition Table :-

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

4)

0|0|0|0|0|0| - -

0 0 0 0 1



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

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

Initial state =  $q_0$

Final state =  $q_3$

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

5.) as  $L(G) = \emptyset$

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

6.)  $L(G) = \Sigma^*$

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

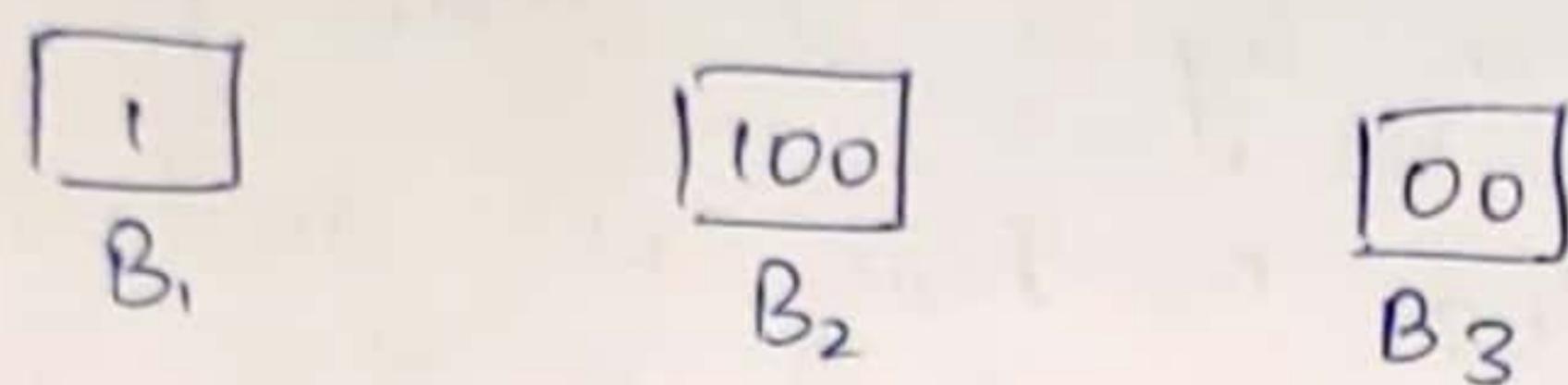
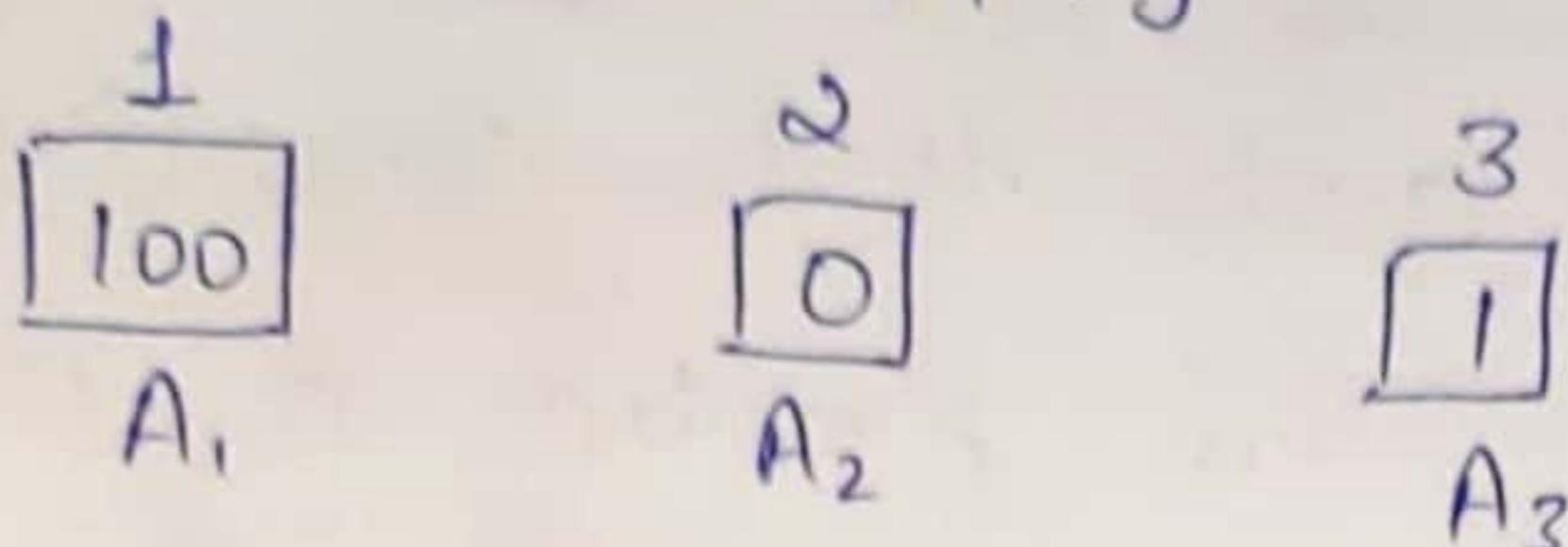
d)  $L(A) \supseteq L(N)$

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

All steps are algorithmic.

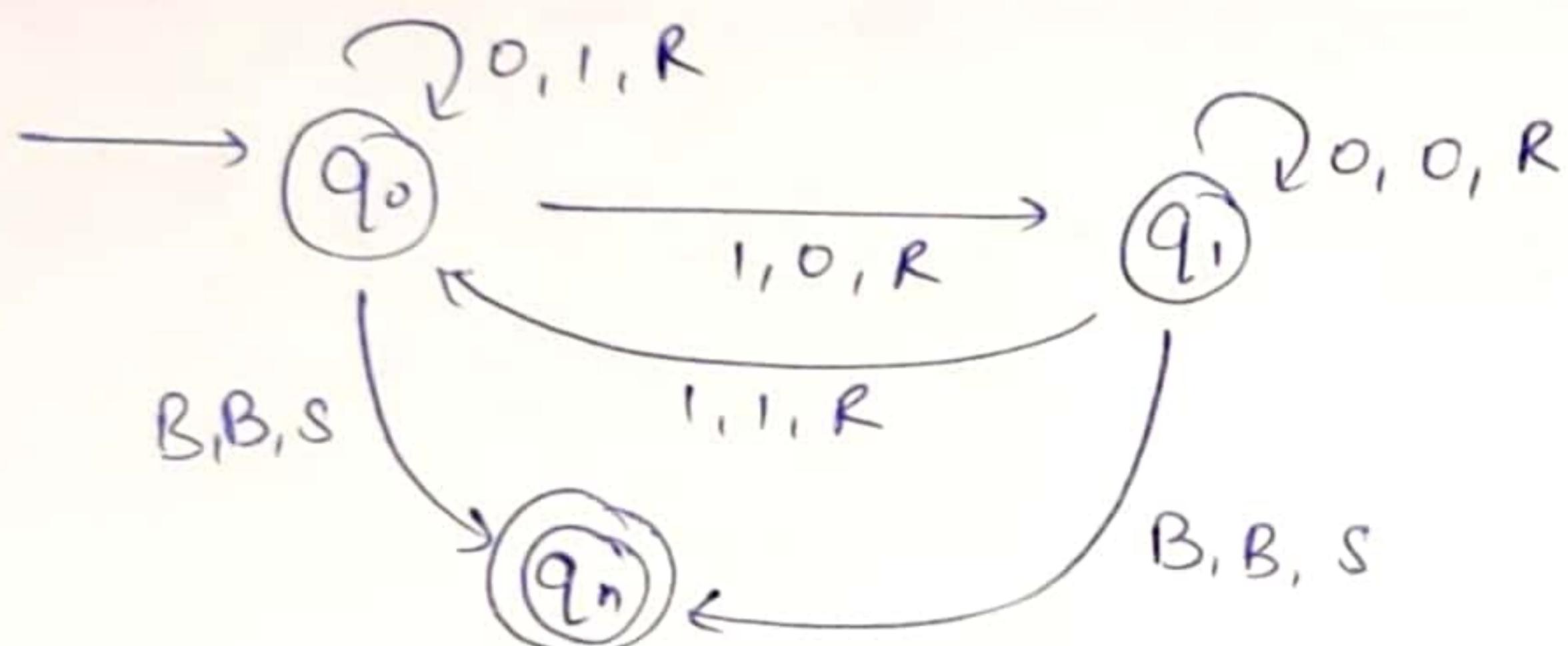
6) PCP

$$A = \{100, 0, 1\} \quad B = \{1, 100, 00\}$$



$A$	1 100	2 0	2 0	3 1	
$B$	1 1	100	100	00 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 strings 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 and only if ( $M$ ) accepts ( $w$ ). Since no algorithm can decide halting problem, no algorithm can decide the PCP.

$$\text{Example: } u = (b, bab^3, ba)$$

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

$b$	$bab^3$	$ba$
$u_1$	$u_2$	$u_3$

$b^3$	$ba$	$a$
$y_1$	$y_2$	$y_3$

	2	1	1	3
$u$	babbb	b	b	ba
$y$	ba	bbb	bbb	a

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

Solution exists.

9.) Let  $L_u = \langle M, w \rangle$ ,  $M$  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  $\langle M', w \rangle \in L_u$ .

Since Halt is undecidable,  $L_u$  cannot be recursive.

Define the diagonal language

$L_d = \langle M \rangle / M$  does not accept  $\langle M \rangle$

The complement of

$K = \langle M \rangle / M$  accepts  $\langle M \rangle$

$K$  is r.e.  $K$  is not recursive.

If  $L_d \supseteq \overline{K}$  were then both  $K$  and  $\overline{K}$  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.

$H(\varrho_H z, w)$  halts and tells us whether m/c  $H$  halts or input  $w$ .

- Construction of D: Construct a new m/c D, that takes an input  $\lambda H y$  and uses H as a sub routine.
- D asks H: "Does M halts on its own description  $\lambda H y$ ?"
- D is programmed to do the opposite of what H does.
- If H says M halts on  $\lambda H y$ , then D loops
  - If H says M loops on  $\lambda H y$ , then D halts.

Contradiction: When run D on its own description  $\lambda D y$ ,

- if D halts on  $\lambda D y$ : D must loop (contradiction)
- if D loops on  $\lambda D y$ , D must halt (contradiction)

Hence, this halting problem is insoluble.

$$13) \quad \begin{array}{c} \boxed{b} \\ A_1 \end{array} \quad \begin{array}{c} \boxed{bab^3} \\ A_2 \end{array} \quad \begin{array}{c} \boxed{ba} \\ A_3 \end{array}$$

$$\begin{array}{c} \boxed{b^3} \\ B_1 \end{array} \quad \begin{array}{c} \boxed{ba} \\ B_2 \end{array} \quad \begin{array}{c} \boxed{a} \\ B_3 \end{array}$$

$$\begin{array}{c|c|c|c} & 2 & 1 & 1 & 3 \\ A & bab^3 & b & b & ba \\ B & ba & b^3 & b^3 & a \end{array}$$

$$m = 4$$

Hence, Solution exists for this PCP.

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

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  $x \in L_1$ ,
- $M_2(u) = \text{accept}$  iff  $u \in L_2$

$L_1 \cup L_2$  - build decider  $M_0$  for input  $u$ :

- 1.) Run on  $x$ . If it accepts, accept.
- 2.) Otherwise run  $M_2$  on  $u$ . If it accept, accept.
- 3.) If both reject, reject.

Termination:  $M_1$  and  $M_2$  always halt, so  $M_U$  always halts.

Correctness: If  $x \in L_1 \cup L_2$ , then at least one of  $M_1, M_2$  accepts, and  $M_U$  accepts, if  $x \notin L_1 \cup L_2$  both reject, so  $M_U$  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 submachines 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 6 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:  
 → 1 → top = 1, bottom = 111 (prefix mismatch)  
 → 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 7 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  $\rightarrow$  sol<sup>n</sup> must start with tile 1.

Date.....

- 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.]

Ques19 Design a Turing machine for Reversing the given string  
aabbaa

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 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  $L$ .

∴ 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 sol<sup>n</sup> exists for this set.]