Unit- 4.

Turing Machines :-

Introduction :-

→ Alan Turing Introduced a new mathematical model called Turing Machine during the year 1936.

→ TM is a tool for studying the Computability of mathematical functions.

* TM is a model for any possible computations.
  - It is simple, but Logically has all the power of any digital computer.

*. It is a Primitive, abstract Computer. It is used to enhance the Computability of mathematical functions.

- TM is used to define the Languages and to compute the Integer functions.
- TM accepts Recursive lang & Recursively Enumerable Lang.
- TM differs from FA & PDA.
  - FA → finite memory
  - PDA → infinite memory (Stack - LIFO).

TM - has both Infinite memory and no restriction in accessing the i/p. → Infinite tape memory, tape head - move either left/right

*. TM can solve any pbm that a modern Computer program can solve.

The pbms that cannot be solved by a TM cannot be solved by a modern computer pgm.

*. TM is most widely used to define languages and to compute Integer functions.

Model of a TM:-

B → Blank Symbols
$A_1, A_2 \ldots A_n$ → i/p Symbols.

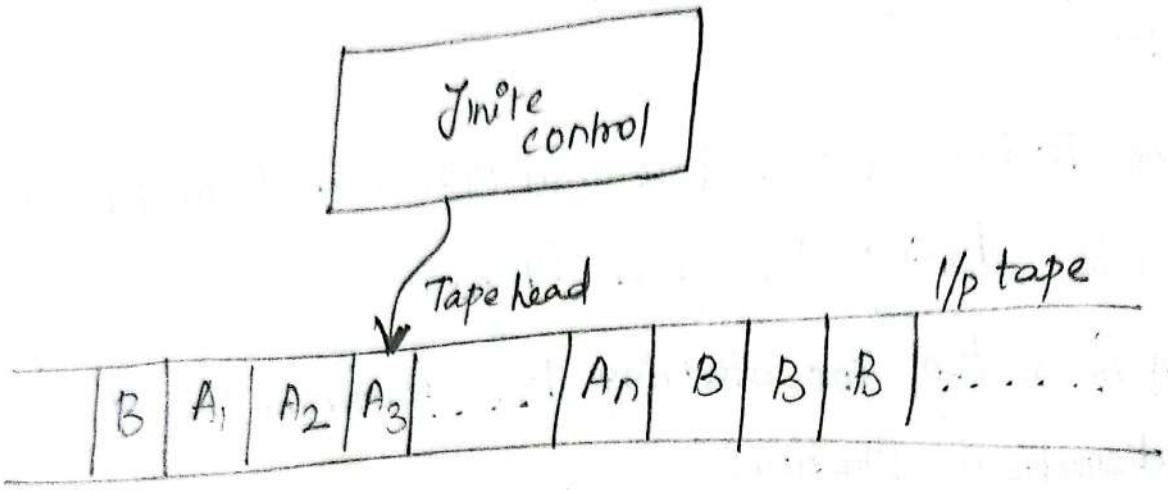


Fig) TM.

→ TM has a finite control and an i/p tape.
→ I/p tape is divided into cells, each cell can hold any one of a symbol.
→ It has a tape head that scans one cell of the tape at a time.

[Initially the tape head is at the leftmost cell that holds the i/p.]

- TM acting as a acceptor.
- TM acting as a transducer.
- TM acting as enumerator.

Formal Definition of a TM :-

TM $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$.

Where,

$Q \rightarrow$ finite set of States.

$\Sigma \rightarrow$ finite set of i/p alphabets.

$\Gamma \rightarrow$ Set of all tape symbols.

$\delta \rightarrow$ Transition fun

$q_0 \rightarrow$ Starting State.

$B \rightarrow$ Blank Symbol.

$F \rightarrow$ Set of Final or accepting States.

δ :-

$\delta(q, X) = (P, A, D)$.

Where $q \rightarrow$ current State in Q.

$X \rightarrow$ Tape symbol.

$P \rightarrow$ Destination State in Q.

$A \rightarrow$ Symbol in Γ written in the cell being scanned.

$D \rightarrow$ Direction of move [left/right].

Processing of Moves in a TM :-

The Single move of a TM depends on the current State of the FC and the tape Symbol present in the i/p tape.

→ The following Changes happen in one move of a TM :-

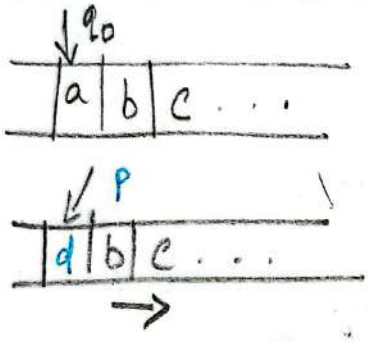
i) Changes the state after consuming an i/p symbol. It may also be in the same state or transfer to any new state.

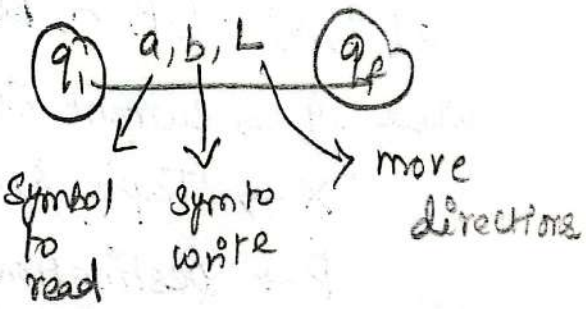
ii) The tape symbol to be replaced for the scanned i/p tape symbol.

iii) Deciding the move of the tape head to left/right of the i/p tape.

iv) Whether to halt the TM or not.

ex: $\delta(q_0, a) = (P, d, R)$.





2 Types of Moves :-

i) Single move $\vdash$.

ii) Sequence of moves or multimoves $\vdash^*$.

Instantaneous Description [ID] :-

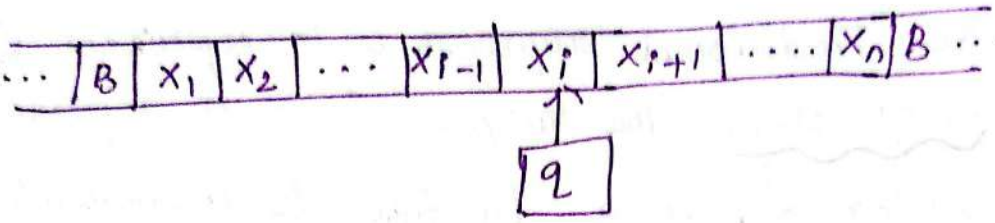
→ Each move of the TM is represented by the ID.

→ ID - describes the current configuration and it can be of following types.

- Accepting configurations
- Rejecting " .

Each move is represented by $\alpha_1 q \alpha_2$ where α_1 and α_2 are the strings from Γ^* and q is the state of TM.

Let us the string $X_1 X_2 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n$ to represent the ID.



eg) $\delta(q, X_i) = (P, Y, \frac{L}{R})$.

Language of a TM :-

*. The set of languages accepted by the Turing machine is the recursively enumerable languages.

*. The i/p string is placed on the tape and the tapehead begins at the leftmost i/p symbol. If the TM enters an accepting state, the input is accepted.

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ be a TM and $L(M)$ is the set of strings, w in Σ^* such that

$$L(M) = \{ w \mid w \text{ in } \Sigma^* \text{ and } q_0 w \vdash^* \alpha_1 P \alpha_2 \text{ for some } P \text{ in } F \text{ and } \alpha_1 \alpha_2 \Sigma \vdash^* \}.$$

Computation :

↳ halt & Accept

→ Halt & Reject

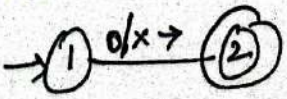
→ Loop → machine fails to halt.

Representation of TM :-

i) ID

ii) Trans. table

iii) Trans. diagram.



Computable Languages and Functions.

"TM as a computer of Integer functions".

→ The TM is also used as a computer of functions from Integers to Integers.

→ The Traditional approach is to represent Integers in unary form. The Integer $i \geq 0$ is represented by the string 0^i. If a fn has k arguments say $i_1, i_2, \ldots i_k$, then these integers are initially placed on the tape separated by 1's as

$0^{i_1} 1 0^{i_2} 1 \ldots 0^{i_k}$.

→ The TM halts with a tape consisting of 0^m for some m when the given function is defined as,

$f(i_1, i_2 \ldots i_k) = m$.

Defn of Recursive Function :-

A fn $f: N \to N$ is said to be a computable fn of k-arguments, if there exists a TM M halts with a tape consisting of 0^m for some m, where

* . $f(i_1, i_2 \ldots i_k) = m$

– If $F(i_1, i_2 \ldots i_k)$ is defined for all $i_1, i_2 \ldots i_k$, then we say, f is a total Recursive fn.

– A fn $f(i_1, i_2 \ldots i_k)$ computed by a TM is called Partial Recursive fn.

Problems :-

1) Design the TM to emplement the fn $f(x) = x+1$.

Solu:

The idea is to design this TM is that if $x=3$, then input tape contains 3 1's (or) 0's in the i/p tape & the results is $3+1 \Rightarrow 4$. (4 1's or 0's).

i/p: 

o/p: 

Steps:

a) The TM is initially in the state q_0 and it reads '1' is the leftmost i/p tape.

b) At the state q_0, when it reads '1' it remains in the same state without changing 1 and just move the tape head to right.

c) When it finds 'B', it enters the final state q_1 and changes B to '1'.

example: $f(x) = x+1$.

$x=3$; $3+1$ $f(x) = 4$.

ID: $(q_0, 111B)$

$\vdash (1q_0 11B)$

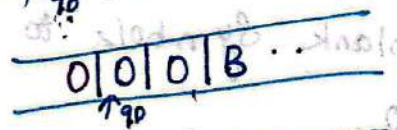
$\vdash (11q_0 1B)$

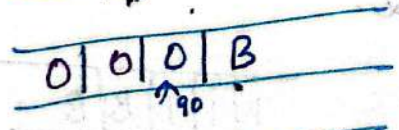
$\vdash (111q_0 B)$

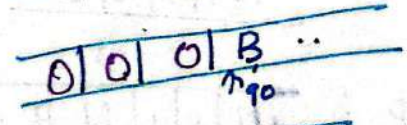
$\vdash (1111, q_1)$.

i) 

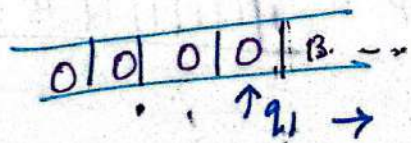
$(q_0, 0) \vdash (q_0, 0, R)$

ii) 

iii) 

iv) 

$(q_0, B) \vdash (q_1, 0, R)$

v) 

So, $M = (\{q_0, q_1\}, \{1\}, \{1, B\}, \delta, q_0, B, \{q_1\})$ (0, B)

δ:

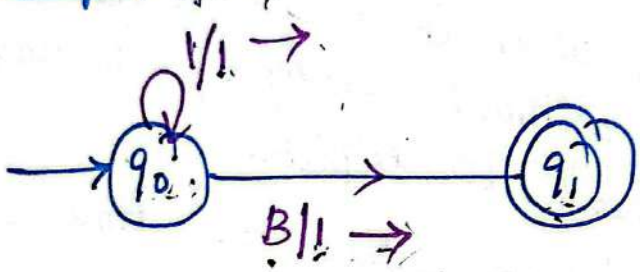
$\delta(q_0, 1) = (q_0, 1, R)$

$\delta(q_0, B) = (q_1, 1, R)$.

T.T:

	1	B
$\rightarrow q_0$	$(q_0, 1, R)$	$(q_1, 1, R)$
$* q_1$	–	–

Trans. diagram: 0/0 →



⇄ 0/- .

2) $f(x) = x + 2$.

Solu

Idea: The idea is to construct this TM if that $x = 3$ then 3 1's are placed in the i/p tape. we have to make the last two blank symbols to 1 to implement this $f(x) = x + 2$.

i/p : 1 1 1 B B ... $x = 3$

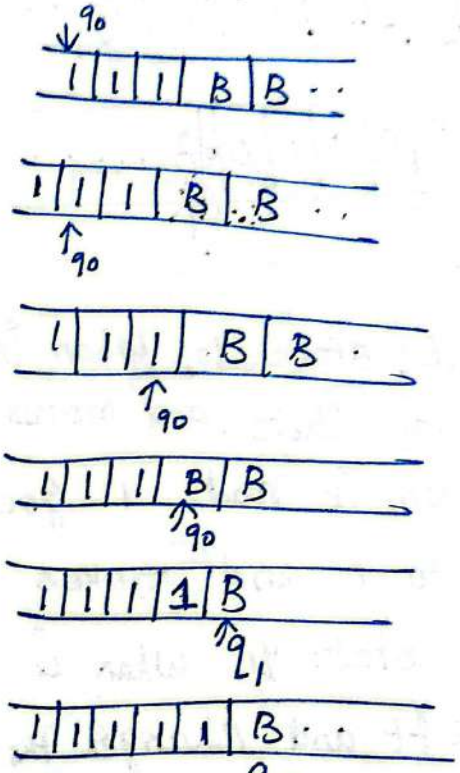
o/p : 1 1 1 1 1 B ... $x = 3 + 2$.

5.

Steps:

1) At state q_0, it reads the Leftmost 1, it skips 1 and searches for the first blank symbol 'B' and moves to right.
2) When it finds B, changes B to 1 and move to right to see the next blank symbol. (q_1)
3) At last reaches the second B, change the state q_2 & moves to right. {accepting state.}

$\delta(q_0, 111BB\ldots)$
$\vdash (1q_0 11BB\ldots)$
$\vdash (11q_0 1BB\ldots)$
$\vdash (111q_0 BB\ldots)$
$\vdash (1111q_1 B\ldots)$
$\vdash (11111q_2 B\ldots)$



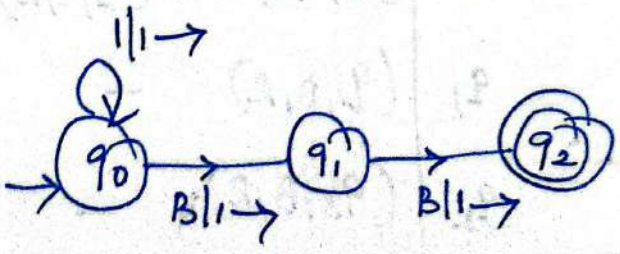
$M = (\{q_0, q_1, q_2\}, \{1\}, \{1, B\}, \delta, q_0, B, \{q_2\})$

δ:

$\delta(q_0, 1) = (q_0, 1, R)$
$\delta(q_0, B) = (q_1, 1, R)$
$\delta(q_1, B) = (q_2, 1, R)$.

TT

	1	B
→ q_0	$(q_0, 1, R)$	$(q_1, 1, R)$
q_1	ϵ —	$(q_2, 1, R)$
* q_2	—	—



3) Construct TM that performs addition operation $f(x,y) = x+y$

or $f(m,n) = m+n$.

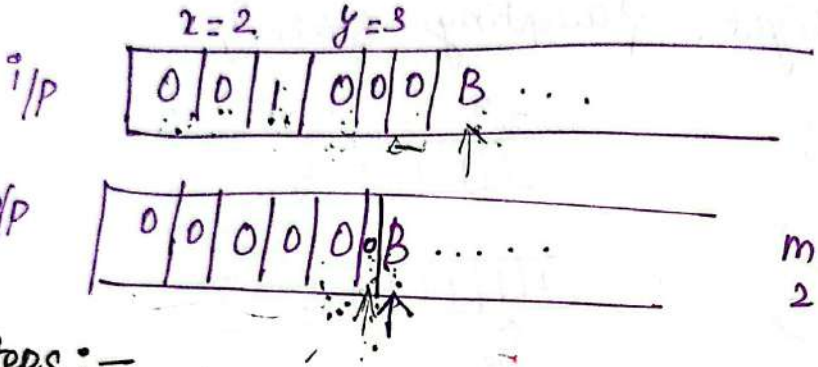
Solv:

$x+y$.

x is represented by 0^x and

y is represented by 0^y.

0^x and 0^y is separated by separator symbol '1'.



$m+n$

$2+3 = 5$.

Steps:-

I.

a) Initial state q_0, when it reads '0' and remains in the same state. and moves right.

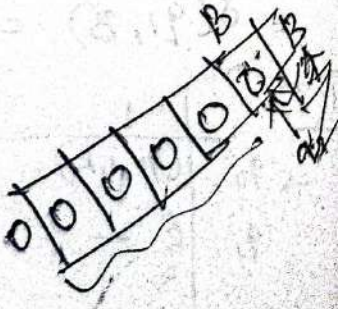
b) When it reads 1 goes to q_1 state and changes 1 to 0 and moves right.

c) At state q_1, when it sees blank symbol, it moves left and changes the state to q_2.

d) At State q_2, when it finds '0', it replaces 0 to B and enters the final state q_3.

II.

	0	1	B
$\rightarrow q_0$	$(q_0, 0, R)$	$(q_1, 0, R)$	–
q_1	$(q_1, 0, R)$	–	(q_2, B, L)
q_2	(q_3, B, R)	–	–
$*q_3$	–	–	–



III. $M = (\{q_0, q_1, q_2, q_3\}, \{0,1\}, \{0,1,B\}, \delta, q_0, B, \{q_3\})$.

δ:

$$\delta(q_0,0) = (q_0,0,R)$$
$$\delta(q_0,1) = (q_1,0,R)$$
$$\delta(q_1,0) = (q_1,0,R)$$
$$\delta(q_1,B) = (q_2,B,L)$$
$$\delta(q_2,0) = (q_3,B,R)$$

IV. ID. example

$i = 2, y = 3$.

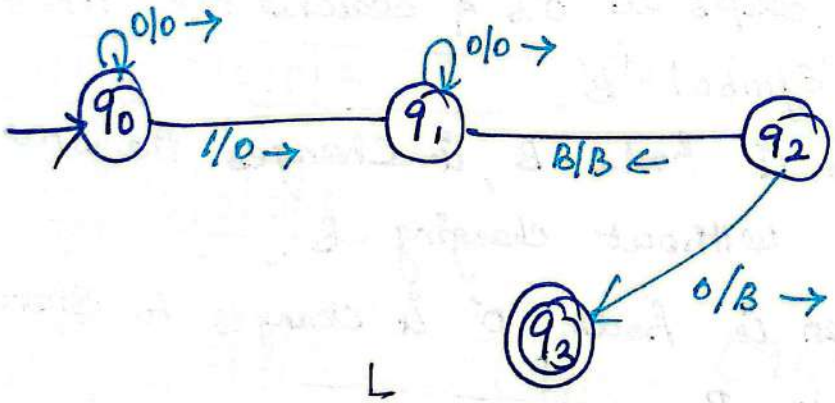
$q_0, 001000 \vdash 0 q_0 01000 \vdash 00 q_0 1000$

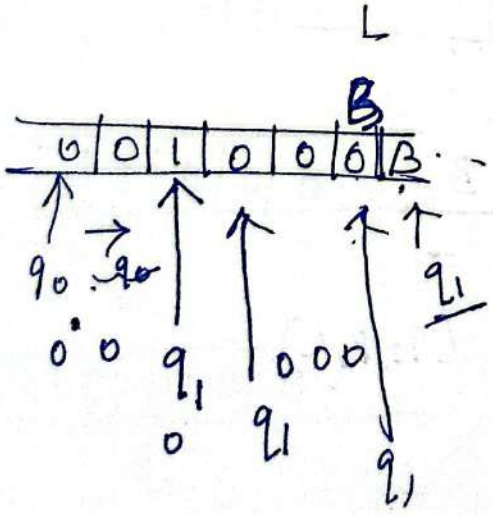
$\vdash 000 q_1 000 \vdash 0000 q_1 00 \vdash 00000 q_1 0$

$\vdash 000000 q_1 B \vdash 00000 q_2 0 B \vdash 00000 B q_3 B \ldots$

accepted.

V.





4) Design a TM to implement the concatenation function

$f(x_1, x_2) = x_1 x_2$.

Solu:- The idea is to design this TM is that to store x_1 and x_2 in the tape. Let $x_1 = 2$ & $x_2 = 3$.

i/p: | 0 | 0 | 1 | 0 | 0 | 0 | B | ...

o/p | 0 | 0 | 0 | 0 | 0 | B | ...

$x_1 = 0^{x_1}$

$x_2 = 0^{x_2}$

$\therefore 0^{x_1} | 0^{x_2}$

$= 0^{x_1+x_2}$

Steps:

→ At initial State q_0, The TM Skill all 0's and Searches for '1' for the Separator Symbol.

→ At State q_0, when it finds 1, it changes '1' to '0' and change to State q_1 & moves to right.

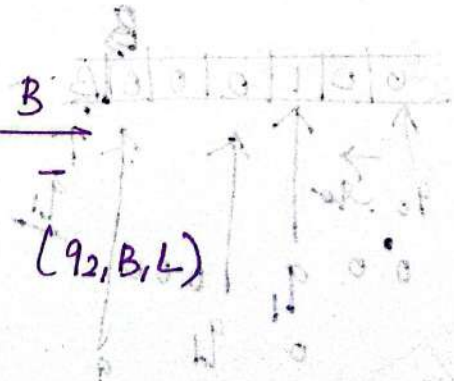
→ At State q_1, it skips all 0's & Searches for first leftmost blank symbol 'B'.

→ At State q_1, when it finds B, it changes the State to q_2 & moves to left without changing B.

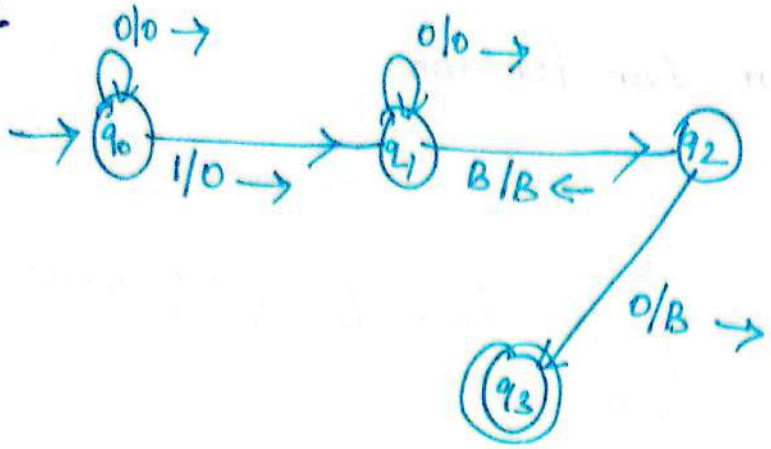
→ At State q_2, when it finds '0' it changes to State q_3 and changes 0 to B.

Step 2:

	0	1	B
→ q_0	$(q_0, 0, R)$	$(q_0, 0, R)$	–
q_1	$(q_1, 0, R)$	–	(q_2, B, L)
q_2	(q_3, B, R)	–	–
* q_3	–	–	–



Step 3



Step 4:

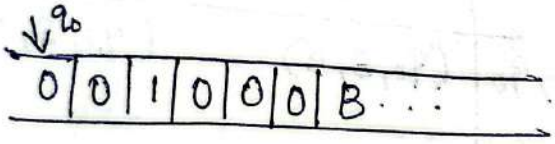
ex: Let $M = (\{q_0, q_1, q_2, q_3\}, \{0,1\}, \{0,1,B\}, \delta, q_0, B, \{q_3\})$

δ: input is 2 + 3, $0^2 0^3$, it is separated by $0^2 1 0^3$.

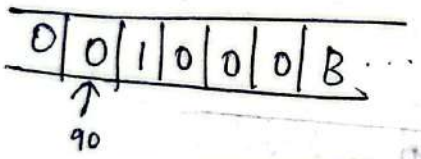
$q_0, 00100 0B \rightarrow 0q_0, 0100 0B \vdash 00q_0 1000 \vdash$

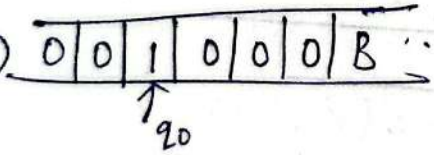
$\vdash 000 q_1 000B \vdash 0000 q_1 1 B \vdash 00000 q_1 B$

$\vdash 000000 q_1 B \vdash 00000 q_2 0 B \vdash 00000 B q_3 B.$

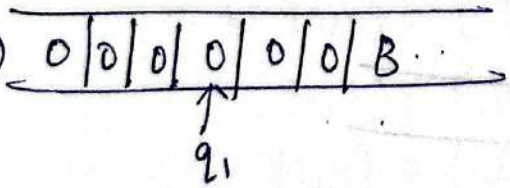
i) 

$(q_0, 0) = (q_0, 0, R)$

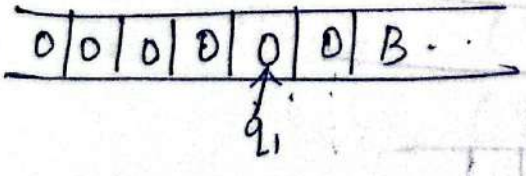
2) 

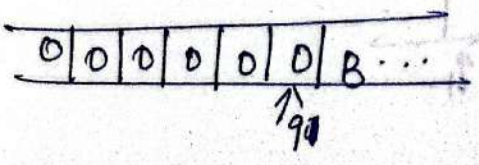
3) 

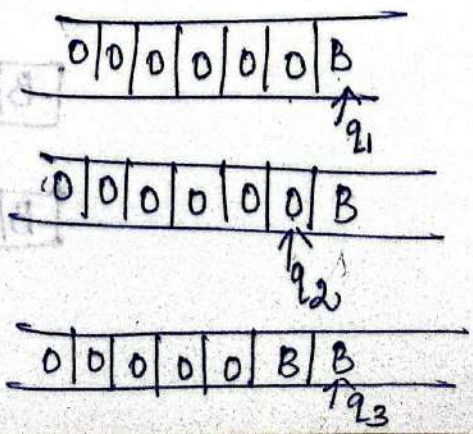
$(q_0, 1) = (q_1, 0, R)$

4) 

$(q_1, 0) = (q_1, 0, R)$

5) 

6) 

7) 

Accepted.

5) Construct a TM for zero function:

$f : N \to N$

$f(x) = 0$.

Solv: The Idea is, change the i/p to blank (B) until the control moves to blank (B).

ex: $x = 3 \Rightarrow 0^x \Rightarrow 0^3$.

i/p: | 0 | 0 | 0 | B ... |

o/p: | B | B | B | B ... |

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

$Q = \{q_0, q_1\}$, $\Sigma = \{0\}$, $\Gamma = \{0, B\}$, $q_0 = \{q_0\}$

$F = \{q_1\}$.

δ:

$\delta(q_0, 0) = (q_0, B, R)$

$\delta(q_0, B) = (q_1, B, R)$.

	0	B
$\to q_0$	(q_0, B, R)	(q_1, B, R)
$*q_1$	—	—

ID:

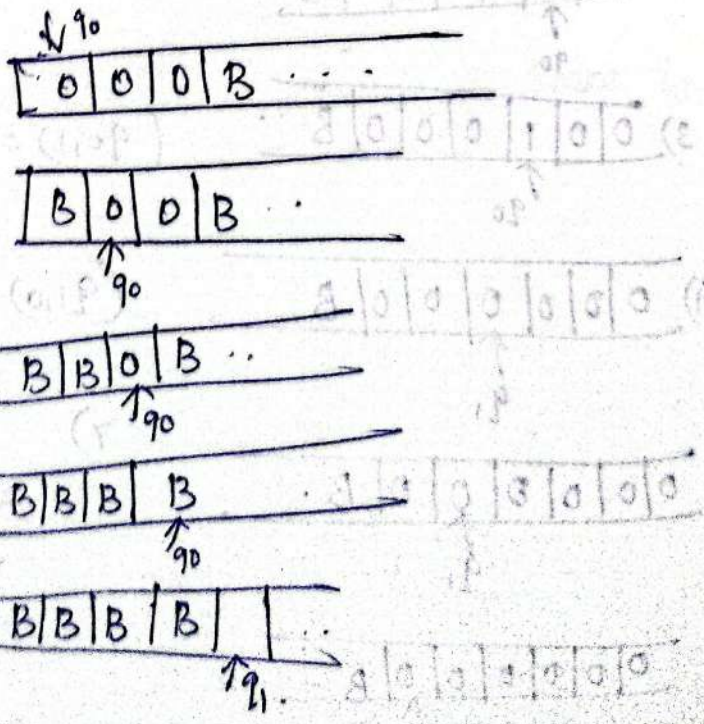
$(q_0, 000B)$

$\vdash (Bq_000B..)$

$\vdash (BBq_00B)$

$\vdash (BBBq_0B)$

$\vdash (BBBBq_1)$.



b) (7)

Design a TM that accept the Lang $L = \{0^n 1^n / n > 1\}$

$w = 0011$

Solv: TM M is in the initial state q_0. At state q_0, it replaces the leftmost symbol 0 by x and changes it to state q_1. At q_1, M searches right for 1's, skipping over 0's and y's. if M find a 1, it changes it to y, entering into state q_2.

From q_2, it searches left for x & moves right to change the state at q_0. At q_0, if Y is encountered, it goes to state q_3 & check that no 1's remain. if y's are followed by a B, State q_4 is entered and accepted.

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, \{q_4\})$.

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

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

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

$q_0 \rightarrow$ initial state & $F \rightarrow$ Final state.

δ:-

$\delta(q_0, 0) = (q_1, X, R)$

$\delta(q_0, Y) = (q_3, Y, R)$

$\delta(q_1, 0) = (q_1, 0, R)$

$\delta(q_1, 1) = (q_2, Y, L)$

$\delta(q_1, Y) = (q_1, Y, R)$

$\delta(q_2, 0) = (q_2, 0, L)$

$\delta(q_2, X) = (q_0, X, R)$

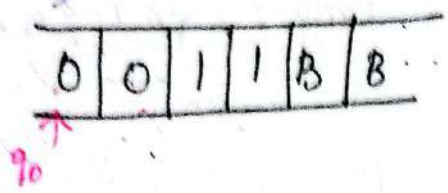
$\delta(q_2, Y) = (q_2, Y, L)$

$\delta(q_3, Y) = (q_3, Y, R)$

$\delta(q_3, Y) = (q_4, B, R)$.

$q_0, 0011B$

0 - x
1 - y.

1) 

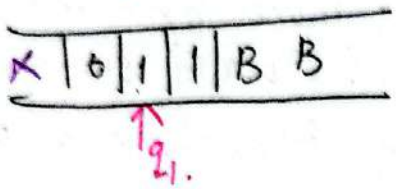
$\delta(q_0, 0) \to (q_1, X, R)$

$\vdash X q_1 0 11$

2) 

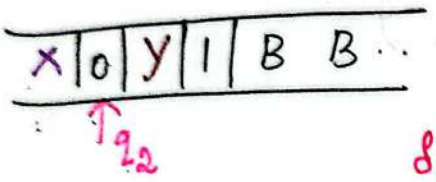
$\delta(q_1, 0) = (q_1, 0, R)$.

$\vdash X 0 q_1 11$

3) 

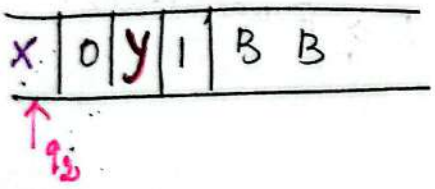
$\delta(q_1, 1) = (q_2, Y, L)$

$\vdash X q_2 0 Y 1$

4) 

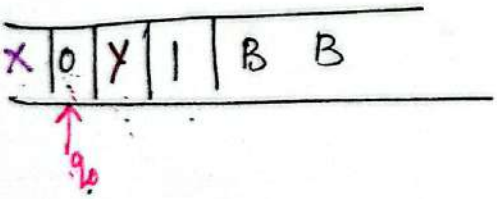
$\delta(q_2, 0) = (q_2, 0, L)$

$\vdash q_2 X 0 Y 1$

5) 

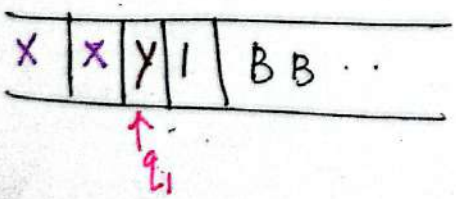
$\delta(q_2, X) = (q_0, X, R)$

$\vdash X q_0 0 Y 1$

6) 

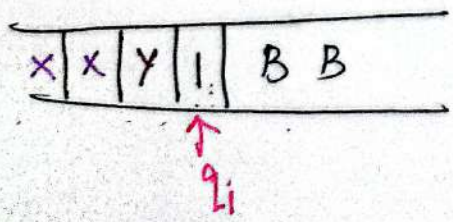
$\delta(q_0, 0) = (q_1, X, R)$

$\vdash X X q_1 Y 1$

7) 

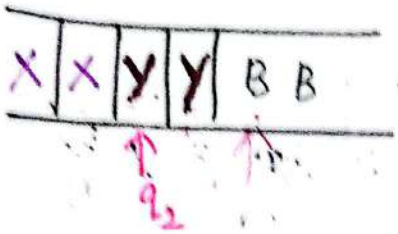
$\delta(q_1, Y) = (q_1, Y, R)$

$\vdash X X Y q_1 1$

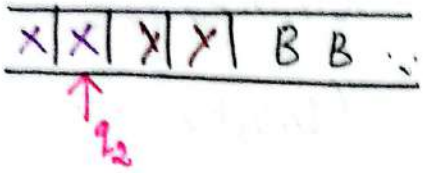
8) 

$\delta(q_1, 1) = (q_2, Y, L)$

$\vdash XXq_2YY$

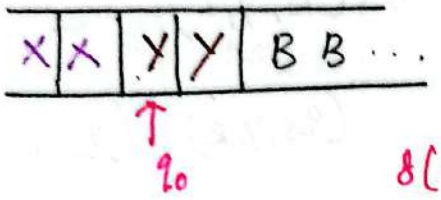
9) 

$\delta(q_2, Y) = (q_2, Y, L)$

10) 

$\vdash Xq_2XYY$

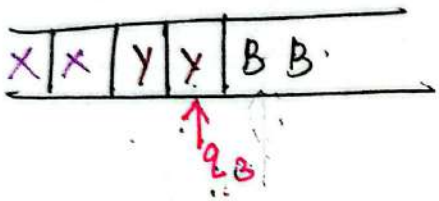
$\delta(q_2, X) = (q_0, X, R)$

11) 

$\vdash XXq_0YY$

$\delta(q_0, Y) = (q_3, Y, R)$

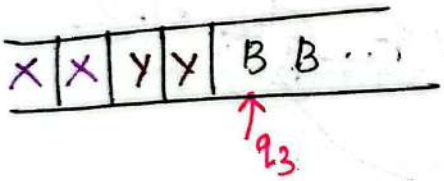
$\vdash XXYq_3Y$

12) 

q_4, R

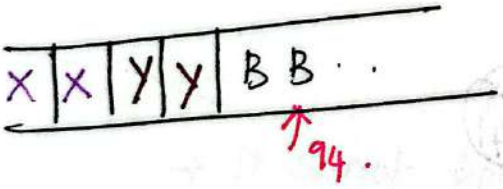
$\delta(q_3, Y) = (q_3, Y, R)$

$\vdash XXYYq_3B$

13) 

$\delta(q_3, B) = (q_4, B, R)$

$\vdash XXYYBq_4.$

14) 

ex: 0011

$q_0 0011 \vdash Xq_1 011 \vdash X0q_1 11 \vdash Xq_2 011 \vdash q_2 X0Y1$

$\vdash q_2 X0Y1 \vdash Xq_0 0Y1 \vdash XXq_1 Y1 \vdash XXYq_1 1 \vdash$

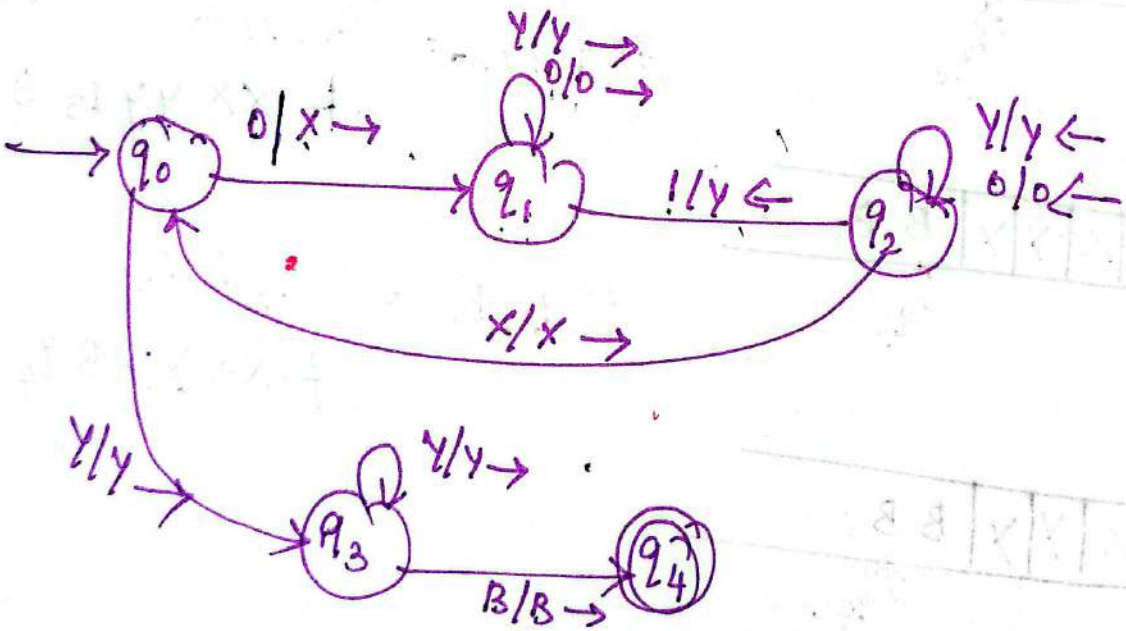
$\vdash XXq_2 YY \vdash Xq_2 XYY \vdash XXq_0 YY \vdash XXYq_3 Y \vdash$

$XXYYq_3 \vdash XXYYq_3 B \vdash XXYYBq_4.$

Trans. Table

state / ip sym	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)
$*q_4$	–	–	–	–	–

Trans. diagram



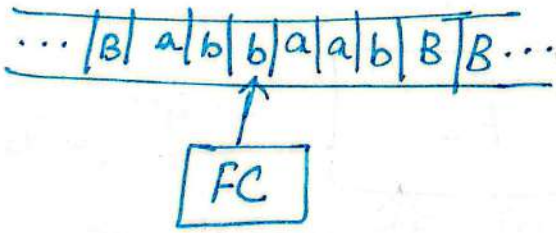
Variations of Turing Machine. (or)
Modifications of Turing Machine.

1) Two-way Infinite Tape.
2) Multitape Turing Machine.
3) Non-deterministic TM
4) Multidimensional TM
5) Multihead TM.
6) offline TM.

1) Two-way Infinite Tape :-

→ In Two way-Infinite tape is infinite in both directions. Any computations can be done by 2-way infinite tape can also be done by Standard TM.

→ extension of Std TM. $M = (Q, \Sigma, T, \delta, q_0, B, F)$.



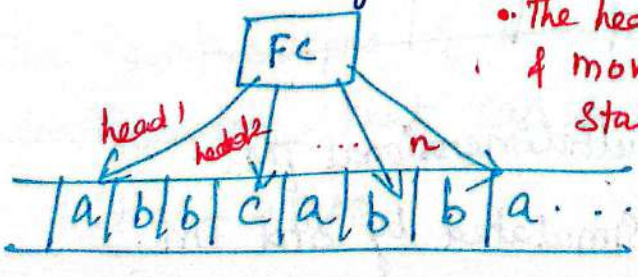
→ It is not powerful than Std TM.

2) Multihead TM :-

- Single tape TM having n-heads reading symbol on the same tape.

→ has a number of heads instead of one.

→ Each head independently r/w symbols and move left/right or keep stationery.



- The heads are numbered 1 through n & move of TM depends upon the state and the symbol scanned by each head.
- In one move the heads may move left/right.

Simulations :-

→ Std TM simulated by Multihead TM.

- Making on head active and ignore remaining head.

→ Multihead TM simulated by Std TM.

- for k-heads using (k+1) tracks if there is ....

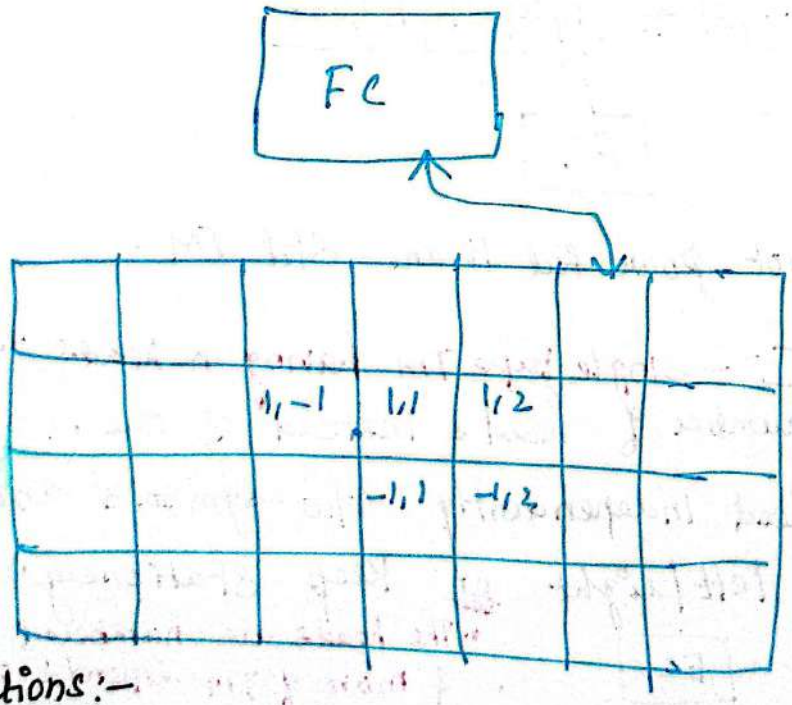
3) Multidimensional TM :-

→ has a Multidimensional tape.

for ex) a two-dimensional TM would read/write on an infinite plane divided into Squares, like a Checkboard.

→ for 2-dimensional TM transaction fnf. defined as

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, U, D\}.$$



2-dimensional address scheme.

Simulations :-

→ Std TM simulated by Multidimensional TM.

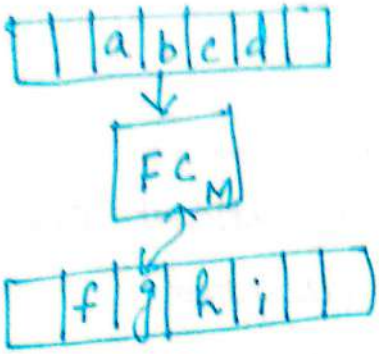
→ Multidimensional TM Simulated by Std TM.

4) OFF-LINE TM:-

→ It has 2 tapes. → one is read-only and contains the input.

→ the other is read-write and is initially blank.

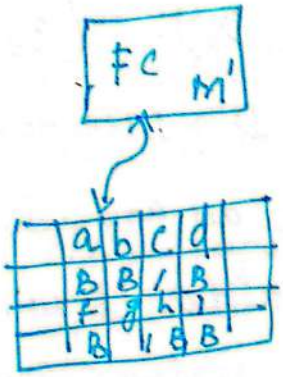
1)



Simulation:-

→ A std TM simulated by off-line TM.

→ An off-line TM simulated by std TM.



5) Non-deterministic TM:-

NTM - Non deterministic TM consists of the finite control and a single one-way finite tape. For each state & i/p more number of moves are available.

→ The transitions are deterministic.

→ The computations of a NTM is a number of configurations that can be reached from the initial state.

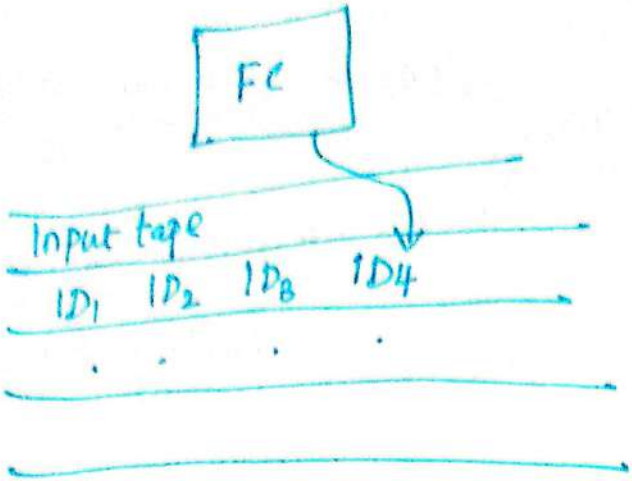


fig) formal NTM model.

→ An i/p is accepted if there is atleast one corrected path, otherwise it is not accepted.

— For a given state & tape symbol scanned by the tape head, TM has a finite no. of choices for the next move..

ii) $\delta(q,x) = \{(q_1, y_1, D_1), (q_2, y_2, D_2) \ldots (q_3, y_3, D_k)\}$

where k - integer & D → Direction.

→ M accepts an i/p w, if there are any sequence of moves that lead from the initial ID with w as input, to an ID with an accepting state.

If M_N is a NTM, then there is a DTM such that

$$L(M_N) = L(M_D).$$

⇒ Any language accepted by a NTM can be accepted by deterministic TM. DTM simulates a NTM.

→ Deterministic multitape TM is used to simulate a NTM.

Given that, the NTM has finite no. of transitions, there are a finite no. of choice i/p strings to generate.

Multitape deterministic TM will always to able to determine if an i/p string is accepted by the NTM.

Theorem :-

Any Lang is accepted by a NTM M_N can be accepted by a Non-deterministic TM M_D.

To prove :

$$L(M_N) = L(M_D).$$

Proof :-

Let M_N be a NTM and $L(M_N)$ be the Lang accepted by M_N. Let w be the string that belongs to $L(M_N)$.
Let M_D be a multitape deterministic Tm to simulate M_N where, M_D has at least three tapes.

- Let the first tape have i/p string w.
- The 2nd tape will be used to generate choice strings in lexicographic order. Finally, the 3rd tape will be used to simulate M_N.
- M_D will start by clearing the 3rd tape, if it is not already blank.
- Then w will be written to the 3rd tape. M_D will then generate the first choice string in lexicographic order on the 2nd tape & then M_D will simulate M_N on the choice string and w.
- If the simulation of M_N fails to reach the accepting halt state, then the next choice i/p will be generated, & the third tape will be cleared and have w rewritten and M_D will begin again by simulating M.
- Thus M_D, deterministically accepts w if and only if there exists a choice input such that M_N accepts w.

Hence, the theorem is proved.

Multitape Turing Machine.

- A multitape TM has a finite control with some finite no. of tapes. Each tape is infinite in both directions. It has its own initial state and some accepting states. Initially,
  - The finite set of i/p symbols is placed on the first tape.
  - All the other cells of all the tapes hold the blank.
  - The ctrl head of the first tape is at the leftend of the i/p.

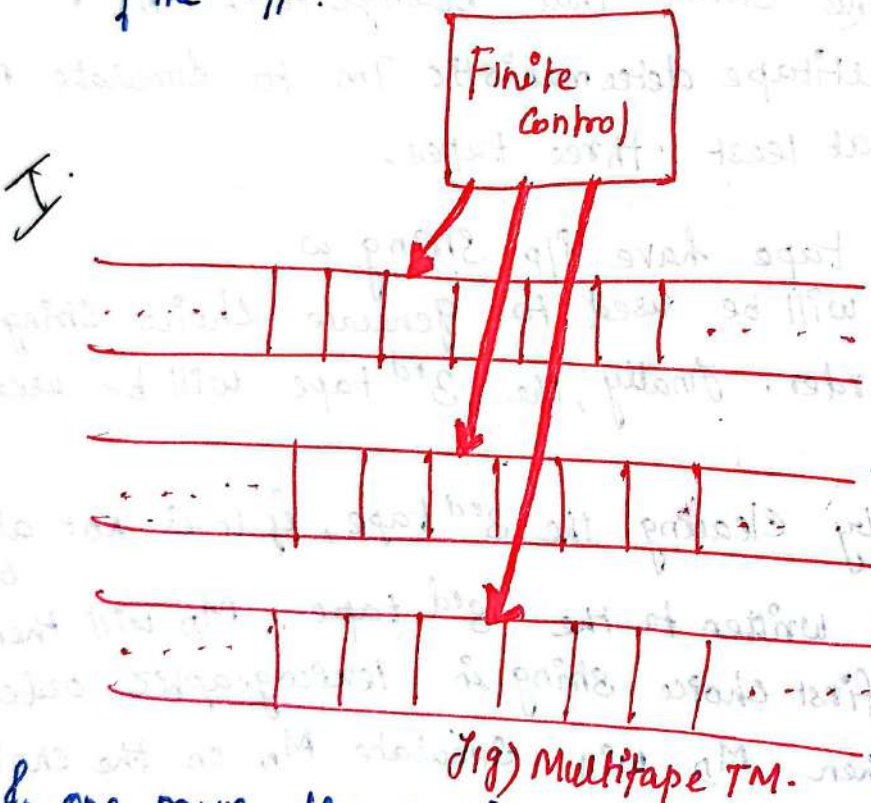


Fig) Multitape TM.

⇒ In one more, the multitape TM can

- Change State
- Print a new symbol on each of the cells scanned by its tape head.
- Move each of its tape heads, independently one cell to the left/right or keep it stationary.

II

Multitape

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

$\delta : Q \times \Gamma^k \to Q \times (\Gamma \times \{L, R, S\})^k$

S → no-shift

k → no. of taps.

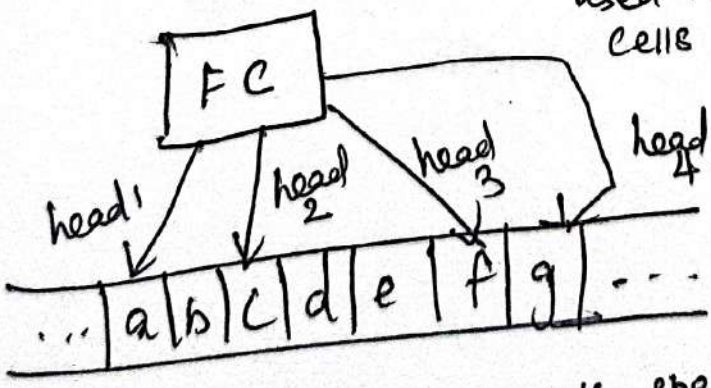
Move – depends on the
- state
- symbol scanned.

IV

Every Lang accepted by multitape TM is recursively enumerable. If one tape TM accepts r.e then multitape TM also accepts r.e.L.

Every multitape TM has an equivalent single-tape TM.

V. In multitape TM, every tape has one or more more tape heads
↓ used to point out the cells on the i/p tape.



→ To increase the speed
→ Improve the traces of TM.

Programming Techniques for TM construction.

→ A TM is also as powerful as a conventional computer. Different Techniques of constructing a TM to meet high-level needs.

1) Storage in the finite control (or) state.
2) Multiple tracks.
3) Subroutines.
4) Checking off 'Symbols'.

1) Storage in the finite control or State :-

*. The Finite control - used to hold a finite amt of intm along with the task of representing a position in the pgm.

*. The state is written as a pair of elements, one for control & other for storing a symbol.

The first elt rep → State,
2nd elt rep → Symbol scanned.

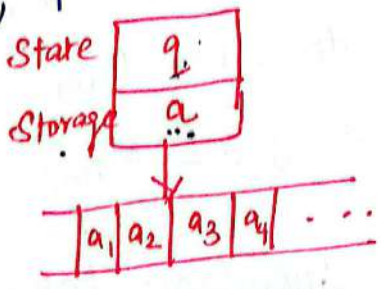


Fig) Storage in finite ctrl.

$\delta(q, a)$

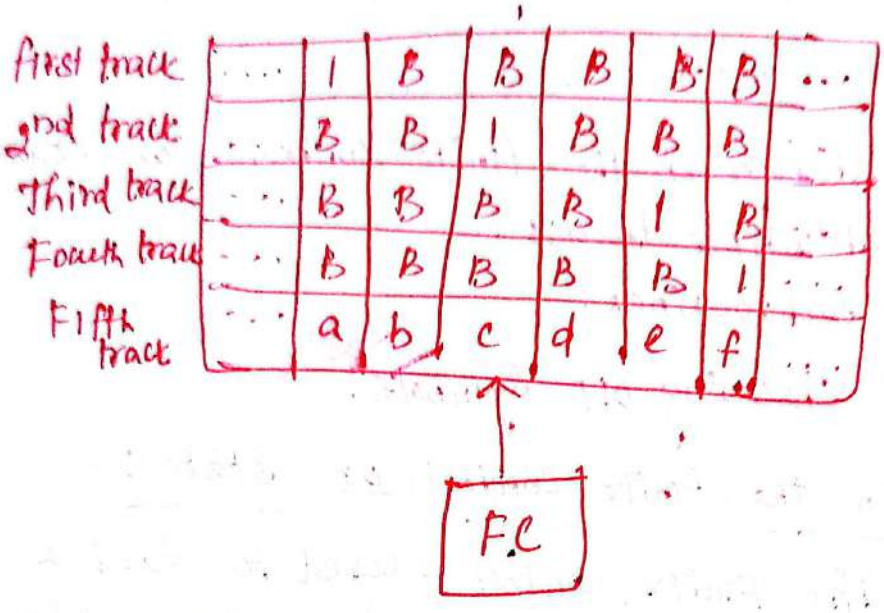
2) Multiple Tracks :-

- It is also possible that a TM's i/p tape can be divided into several tracks. Each track can hold a one symbol, and the tape alphabet of the TM consists of tuples with one component for each track.

- Also called as K-dimensions tape.

where K → number of Tracks.

→ In multi track TM one head is available to perform reads & writes on all tracks as shown in fig.



$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$.

$\delta(q, [a_1, a_2, \cdots]) = (p, [b_1, b_2, b_3, \cdots], L \text{ or } R)$

For every single track TM S_{TM}, there is an equivalent multi-track TM M_{TM} such that

$$L(S_{TM}) = L(M_{TM}).$$

Ex:-

We shall use the techniques storage in the finite control & to design a TM.

$M = (Q, \{0,1\}, \{0,1,B\}, \delta, [q_0, B], B, [q_0, B])$.

→ which takes the leftmost symbol in the i/p & prints it immediately to the right of the i/p.

→ The set of states Q is $\{q_0, q_1, q_2\} \times \{0, 1, B\}$.

that is, the states may be thought of as pairs with two components :-

a) A control portion → q; that remembers what the TM is doing.

b) A data portion → that remembers the first symbol seen, which must be 0 or 1.

The Transition function of M is as follows :-

1) $\delta([q_0, B], a) = ([q_1, a], q, R)$ – Initially, q_0 is the control state, and the data portion of the state is B. The symbol scanned is copied into the 2nd component of the state ; M moves right, entering control q, as it does so.

$\delta(q,a) = (q,a,-)$

2) $\delta([q_1, a], b) = ([q_1, a], b, R)$ – In state q_1, M skips over each non-blank symbol & continue moving right.

3) $\delta([q_1, a], B) = ([q_2, B], q, R)$ – if M reaches the first blank, it copies the symbol to from its finite control to the tape & enters the accepting state (q_2, B).

3) Subroutines :-

→ A pbm with same tasks to be repeated for many no. of times, can be programmed using Subroutines.

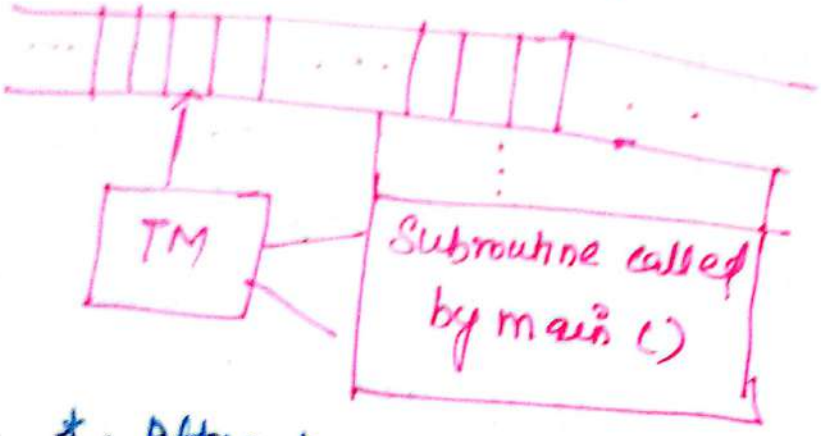
→ A TM with Subroutine is a set of states that perform some useful process.

*. The idea here is to write part of a TM progeams to serve as a Subroutine which has its own initial state & a return state for returning to the calling routine.

– It improves the modular or top-down progeamming design.

Subroutine has 2 states. — Initial State, — return State.

*. When the main fn is executed, the Subroutine is called.



*. The TM reaches the Initial State & follows a series of executions using the Trans. rules of the Subroutines.

*. After processing, the TM reaches the return state that returns back to the main fn after a temp halt.

4) Checking off Symbols :-

- is useful method when a TM recognizes lang. defined by repeated strings and to compare the lengths of the substrings.

$\{ww \mid w \in \Sigma^*\}$ or $\{ww^R \mid w \in \Sigma^*\}$.

- Checking off symbol is implemented in a TM by introducing a new track on the tape with symbols Blank or ✓.
- The three(3) ✓ symbol appear as the tape when symbol below it is taken into consideration for the comparision.

Check for Symbols

Ex: $L = \{wcw \mid w \text{ is } (a+b)^*\}$

aba c aba

1) aba c aba Δ Read a, Replace with *

2) * bac aba Δ

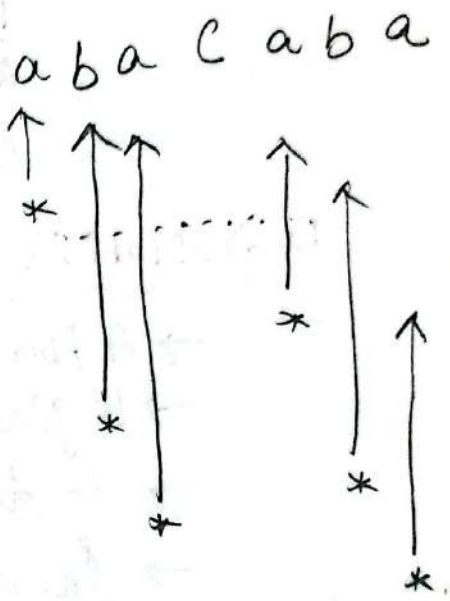
3) * bac * ba Δ

4) * * a c * ba

5) * * a C * * a

6) * * * C * * a

7) * * * c * * *



aba c aba Δ

Subroutines: a = 2, b = 3. Unary.

1	1	#	1	1	1	Δ	Δ		
↑									
X	1	#	1	1	1	Δ			
	↑		↑						
X	1	#	y	1	1	Δ	1		
X	1	#	y	y	1	Δ	1	1	
X	1	#	y	y	y	Δ	1	1	1
X	X	#	y	y	y	Δ	1	1	1

Read 1 – change it into X.

11 # 111 ΔΔ

Partial Solvability.

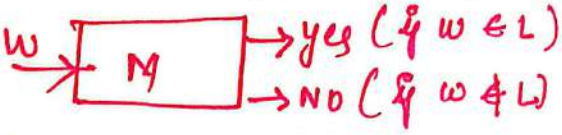
The Pbms solved by the TM are classified into three (3) classes.

1) Decidable.
2) Undecidable.
3) Partially solvable.

1) Decidability :- Solvable / recursive

→ A Pbm is said to be decidable, if there is an algorithm
→ It gives the solution for the given problem.
→ It decides the results either 'yes' or 'no'.
→ In decidable problems we can easily take decisions.

*. A problem P is decidable or solvable if there is a turing Machine T that solves P, such that a T always halts.



for ex:

1) The strings over {a,b} that contains an equal no. of a's & b's.
2) Given NFA accepts the strings w.

2) Undecidability :- / unsolvable

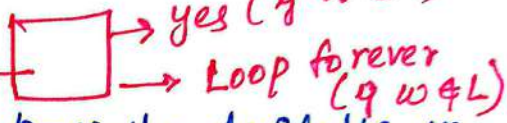
→ A Pbm is said to be Undecidable, if there is no algorithm.
→ It does not give solution for the given problem.
→ We cannot easily determine the solution.
→ It doesnot decide whether the ans is 'yes' or 'no'.

*. A Problem P is Undecidable or Unsolvable if there is a TM T that cannot solve P, such that a T never halts.

All non-trivial pbms about the lang accepted by a TM are undecidable.

for ex:
1. Given a TM M and i/p string w, does TM M halt on i/p w?
2. A TM M that halts with an empty tape is undecidable.

3) Partially Solvable :- Recursively enumerable



→ A pbm P is semi decidable or partially decidable or Partially solvable if there is a TM T that partially solves P, such a T solves all instances of P for which the right answer is "yes", but fails to halt for all instances of P for which the right ans is "no".

for ex: Some pbms involving numbers are :-

- Is this Integer a prime?
- Does this equation have a root blw 0 & 1?
- Is this Integer a perfect Square?

Some of these problems or equations are :-

- Is this program correct?
- How long will this Pgm run?
- Does this pgm contain an infinite loop?
- Is this pgm more efficient than that one?

Partial Solvability.

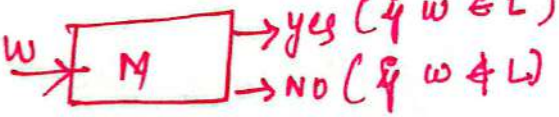
The Pbms solved by the TM are classified into three (3) classes.

1) Decidable.
2) Undecidable.
3) Partially solvable.

1) Decidability :- Solvable / recursive

→ A Pbm is said to be decidable, if there is an algorithm.
→ It gives the solution for the given problem.
→ It decides the results either 'yes' or 'no'.
→ In decidable problems we can easily take decisions.

*. A problem P is decidable or solvable if there is a turing Machine T that solves P, such that a T always halts.



for ex:

1) The strings over {a,b} that contains an equal no. of a's & b's.
2) Given NFA accepts the strings w.

2) Undecidability :- / unsolvable

→ A Pbm is said to be Undecidable, if there is no algorithm.
→ It does not give solution for the given problem.
→ we cannot easily determine the solution.
→ It doesnot decide whether the ans is 'yes' or 'no'.

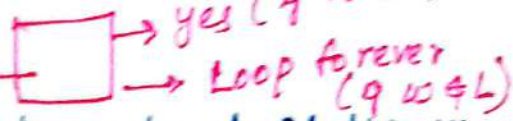
* A Problem P is undecidable or Unsolvable if there is a TM T that cannot solve P, such that a T never halts.

All non-trivial pbms about the lang accepted by a TM are undecidable.

for ex: 1. Given a TM M and i/p string w, does TM M halt on i/p w?

2. A TM M that halts with an empty tape is undecidable.

3) Partially Solvable :- Recursively enumerable



→ A pbm P is semi decidable or partially decidable or Partially solvable if there is a TM T that partially solves P, Such a T solves all instances of P for which the right answer is "yes", but fails to halt for all instances of P for which the right ans is "no".

for ex: Some pbms involving numbers are :-

- Is this Integer a prime?
- Does this equation have a root blw 0 & 1?
- Is this Integer a perfect Square?

Some of these problems or equations are :-

- Is this program correct?
- How long will this Pgm run?
- Does this pgm contain an infinite loop?
- Is this pgm more efficient than that one?

The formal definitions of Solvability for Problem is as follows:

Defn:- A Pbm P is Solvable if and only if there is a TM M_i such that for all "ws". if we can always Give a Pbm by carrying out a computation it is a Solvable Problem.

$$M_i(w) = \begin{cases} 0 & \text{if } P(w) \text{ is false} \\ 1 & \text{if } P(w) \text{ is true.} \end{cases}$$

The Halting Problem.

- Halting is another notation of TM acceptance. if TM halts & enters a state q, scanning a tape symbol x (read) and there is no more move in this situation
  (ie) $\delta(q, x)$ is undefined.

*. The halting Pbm is the Pbm of finding if the Pgm/machine halts or loop forever.

*. The halting Pbm is undecidable over Turing machine.

Description Consider the TM, M & a given string w, the Pbm is to determine whether M halts by either accepting or rejecting w, or run forever.

eg)
```
while (1)
{
    Printf("Halting Pbm");
}
```

→ The above code goes to an infinite loop since the argument of while loop is true forever
→ Thus it doesnot halts.

Hence the Turing machine Pbm is the example of undecidability.

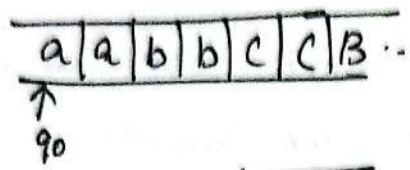
—:

*.

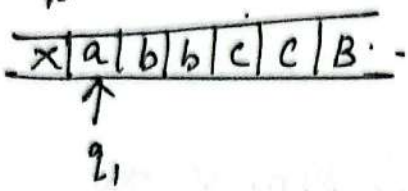
Design a TM for $L = \{ a^n b^n c^n \mid n \geq 1 \}$.

Solu: (I) $a^n \to x$; $b^n \to y$; $c^n \to z$.

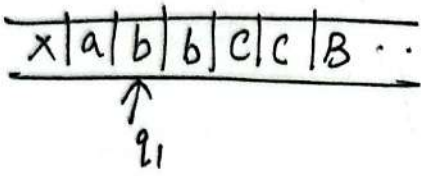
(II)

1) 

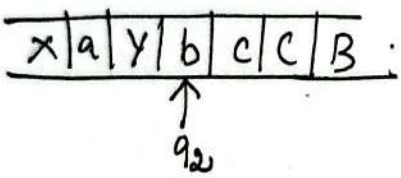
$(q_0, a) = (q_1, x, R)$

2) 

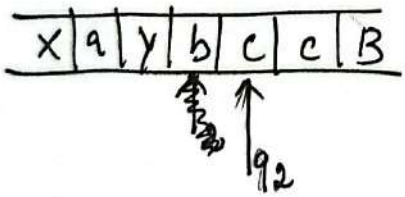
$(q_1, a) = (q_1, a, R)$

3) 

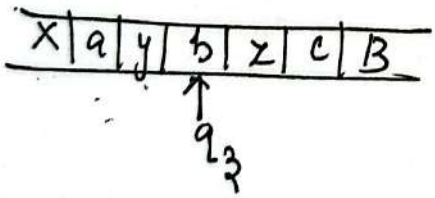
$(q_1, b) = (q_2, Y, R)$

4) 

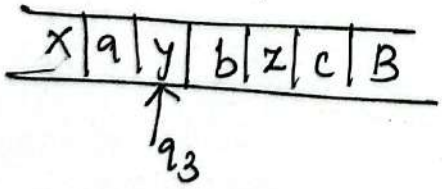
$(q_2, b) = (q_2, b, R)$

5) 

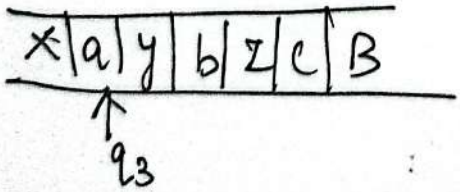
$(q_2, c) = (q_3, Z, L)$

6) 

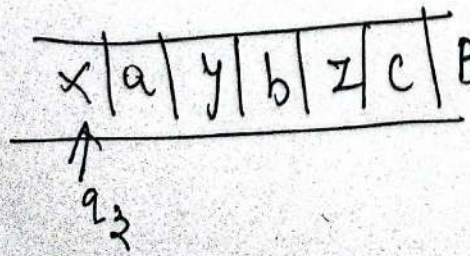
$(q_3, b) = (q_3, b, L)$

7) 

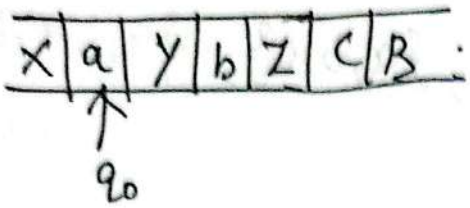
$(q_3, Y) = (q_3, Y, L)$

8) 

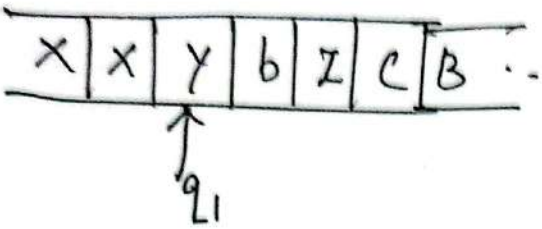
$(q_3, a) = (q_3, a, L)$

9) 

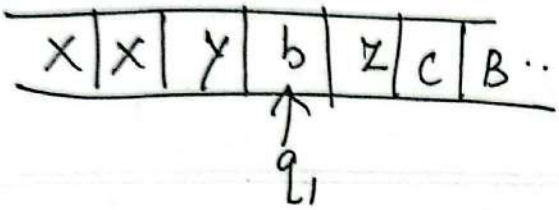
$(q_3, x) = (q_0, x, R)$

10) 

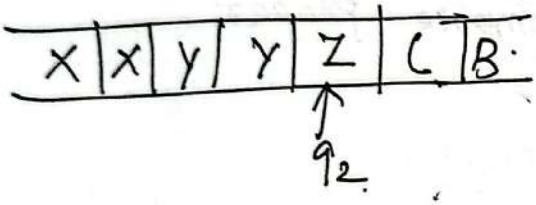
$(q_0, a) = (q_1, X, R)$.

11) 

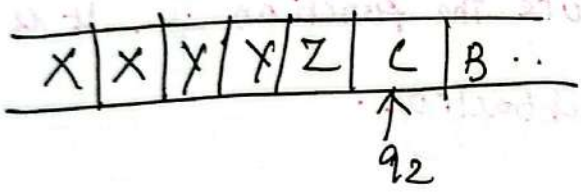
$(q_1, Y) = (q_1, Y, R)$

12) 

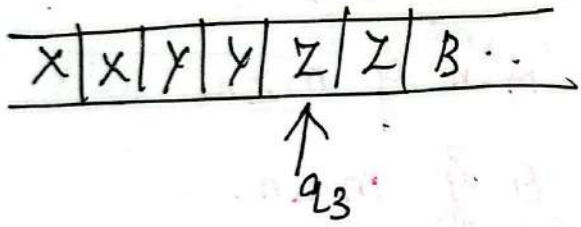
$(q_1, b) = (q_2, Y, R)$.

13) 

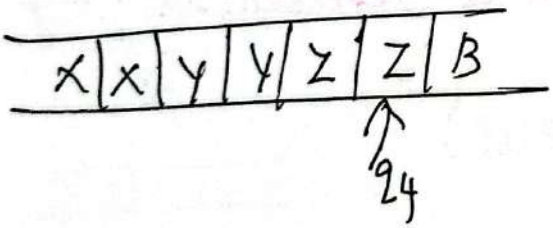
$(q_2, Z) = (q_2, Z, R)$.

14) 

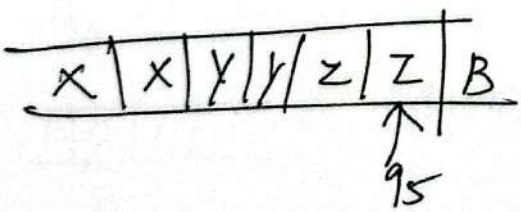
$(q_2, C) = (q_3, Z, L)$

15) 

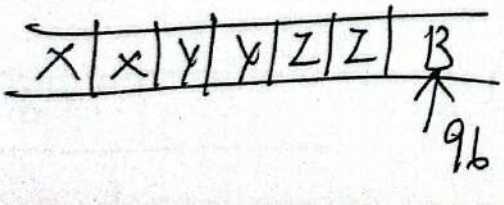
$(q_3, Z) = (q_4, Z, R)$

16) 

$(q_4, Z) = (q_5, Z, R)$

17) 

$(q_5, Z) = (q_6, Z, R)$

18) 

// accepted.

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

IV. δ:

V. Trans. Table:

VI. Trans. diagram:-

Palindrome

---

2) Design a TM to compute proper Subtraction:

Solu: TM might compute the function $\dot{-}$. It is called monus or proper subtraction.

$$m \dot{-} n = \max(m-n, 0)$$

i) $m \dot{-} n$ is $m-n$ if $m \geq n$

$m \dot{-} n$ is 0 if $m < n$.

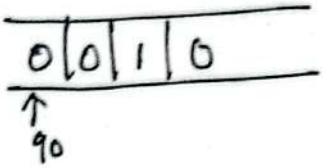
Proper Subtraction $m \dot{-} n$ is represented by $0^m 1 0^n$.

1) * for ex, i/p string 2-1 is represented by $0^2 1 0^1$, $m > n$ ie, $2 > 1$ the result is 1 represented by 0.

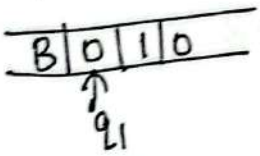
2) Another i/p string 1-2 is represented by $0^n 1 0^2$, $m < n$ ie, 1-2, the result is 0 represented by B.

$0^2 1 0^1$. $2 > 1$. $\Rightarrow 1$. rep by *.

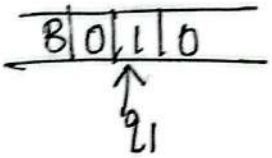
1) ex:

* 1) 

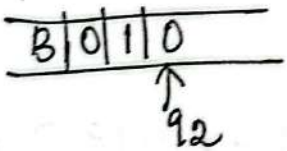
$(q_0, 0) = (q_1, B, R)$

2) 

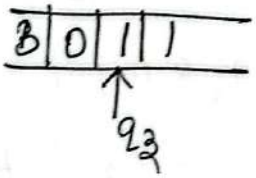
$(q_1, 0) = (q_1, 0, R)$

3) 

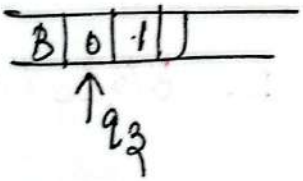
$(q_1, 1) = (q_2, 1, R)$

4) 

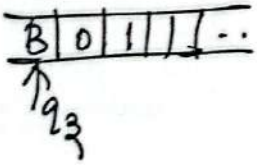
$(q_2, 0) = (q_3, 1, L)$

5) 

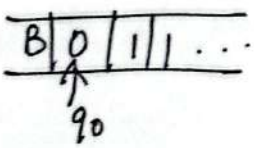
$(q_3, 1) = (q_3, 1, L)$

6) 

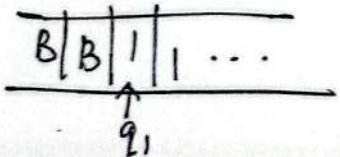
$(q_3, 0) = (q_3, 0, L)$

7) 

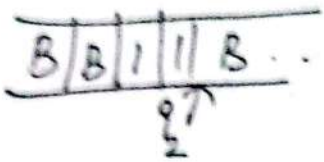
$(q_3, B) = (q_0, B, R)$

8) 

$(q_0, 0) = (q_1, B, R)$

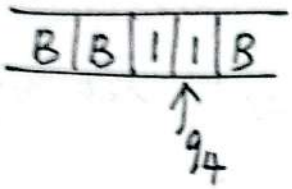
9) 

$(q_1, 1) = (q_2, 1, R)$

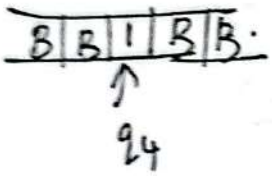


$(q_2, 1) = (q_2, 1, R)$

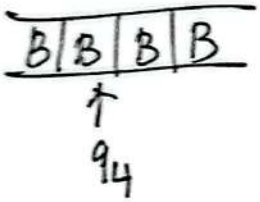
$(q_2, B) = (q_4, B, L)$



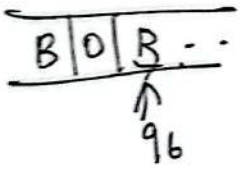
$(q_4, 1) = (q_4, B, L)$



$(q_4, 1) = (q_4, B, L)$



$(q_4, B) \rightarrow (q_6, 0, R)$



$q_0 0010 \vdash B q_1 010 \vdash B 0 1 q_2 0 \vdash$
$B 0 q_3 11 \vdash B q_3 011 \vdash q_3 B 011 \vdash$
$B q_0 011 \vdash B B q_1 11 \vdash B B 1 q_2 1 \vdash$
$B B 11 q_2 \vdash B B 1 q_4 1 \vdash B B q_4 1 \vdash$
$B q_4 \vdash B 0 q_6 .$