Turing Machine:

TM has infinite size tape and it is used to accept Recursive enumerable lang.

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

⇒ If the string inserted is not in a lang, machine will halt in non-final state.

⇒ If the string inserted is in a lang, machine will halt in final state.

⇒ TM is a mathematical model which consists one infinite length tape is divided into cell on which i/p is given.

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

⇒ A state register stores the state of TM.

⇒ After reading 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 is accepted, otherwise it is rejected.

blank sym

a	b	B	B	B

Formal definition:

A TM can be formally described as 7-tuples

$$(Q, \Sigma, \Gamma, \delta, q_0, b, F)$$

Q is a finite set of states

Σ is the i/p alphabet

Γ is the tape alphabet

δ is the transition function

$$\delta : Q \times x \rightarrow Q \times x \times \{\text{left}, \text{right (or) no move}\}$$

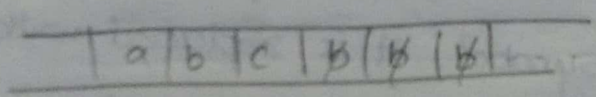
q_0 is the initial state

b is the blank symbol. $b \in \Gamma$

F is the set of final states. $F \subseteq Q$

Basic model of Turing Machine:

1. The i/p tape is having an infinite no. of cells each cell containing one i/p symbol. The empty tape is filled by blank characters.

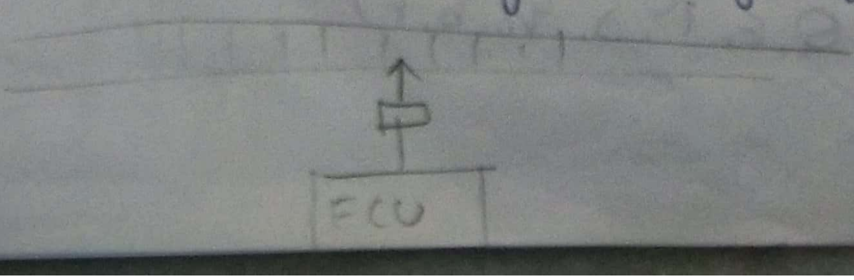


i/p tape

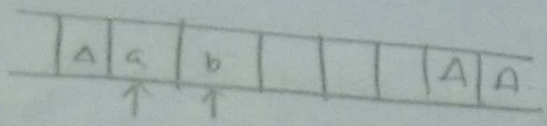
2. The finite control and the tape head which is responsible for reading the current i/p symbol. The tape head can move ~~from~~ to left to right.

3. A finite set of states through which machine has to undergo.

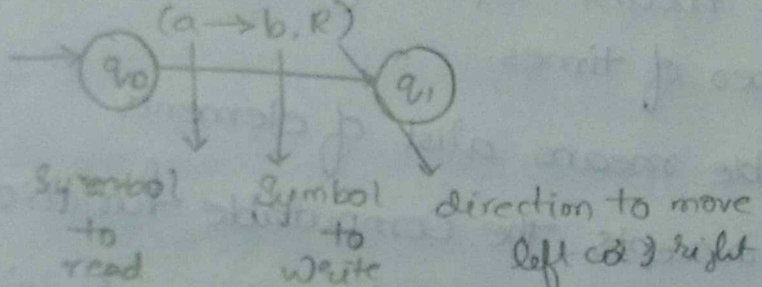
4. Finite set of symbols called external symbols which are used in building the logic of TM



Operations on the tape:



1. Read /scan the symbol on tape head
2. Update /write a symbol to tape head
3. Move the tape head one step left
4. More the tape head one step right.



Turing's Thesis:-

It states that any computation that can be carried out by mechanical means can be performed by some TM.

The arguments for accepting this thesis are:

1. Anything that can be done on existing digital computer can also be done by TM.

2. No one has yet been able to suggest a problem solvable by what we consider an algorithm, for which a TM program cannot be written.

3. The language that is accepted by TM is recursively enumerable language.

A Lang $L \subseteq \Sigma$ is said to RE if there exists a TM that accepts it.

recursively ⇒ replacing same set of rules for any no. of times.

Representation of Language accepted by Turing machine:

The TM accepts all the lang even though they are recursively enumerable

Recursive means repeating same set of rules for any number of times.

Enumerable means a list of elements.

TM also accepts the Computable function, such as addition, subtraction, multiplication, division & many more.

⇒ Construct a TM accepts aba over $\{a,b\}$

⇒ place string 'aba' on i/p tape

a	b	a	$\not{b}$	$\not{b}$	$\not{b}$

If the tape head is read out 'aba' string then TM will halt after reading $\not{b}$.

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

↓ a	b	a	$\not{b}$

q_0

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

A	b	a	$\not b$
	↑		

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

A	B	a	$\not b$

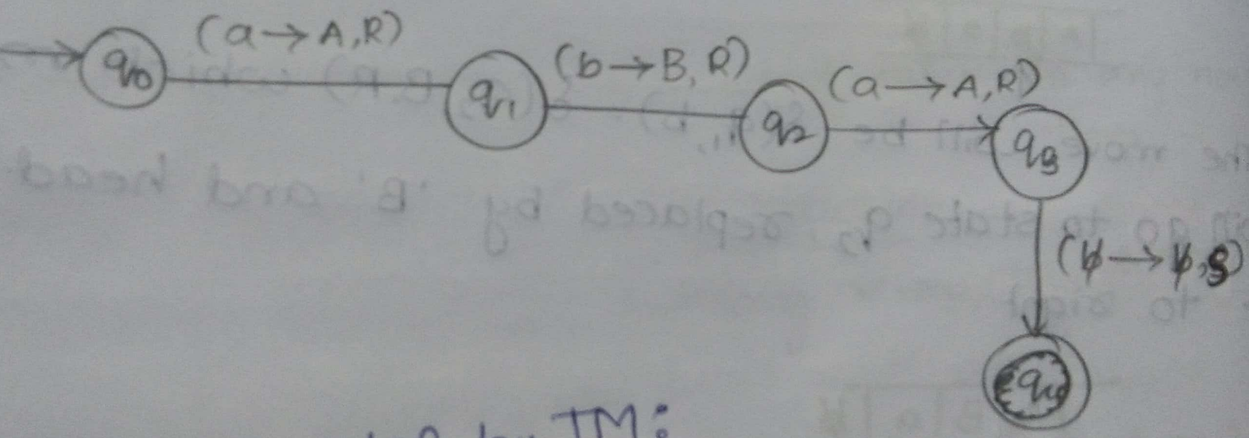
The move $\delta(q_2, a) = \delta(q_3, \not b, R)$ which means it will go to state q_3. replaced 'a' by 'A' & head will move to right as

A	B	A	$\not b$

The move $\delta(q_3, \not b) = (q_4, \not b, S)$ which means it will go to state q_4, which is HALT state which is accept state for any TM.

Transition table

	a	b	$\not b$
q_0	(q_1, A, R)		
q_1		(q_2, B, R)	
q_2	(q_3, A, R)		
q_3			$(q_4, \not b, S)$
q_4	-	-	-

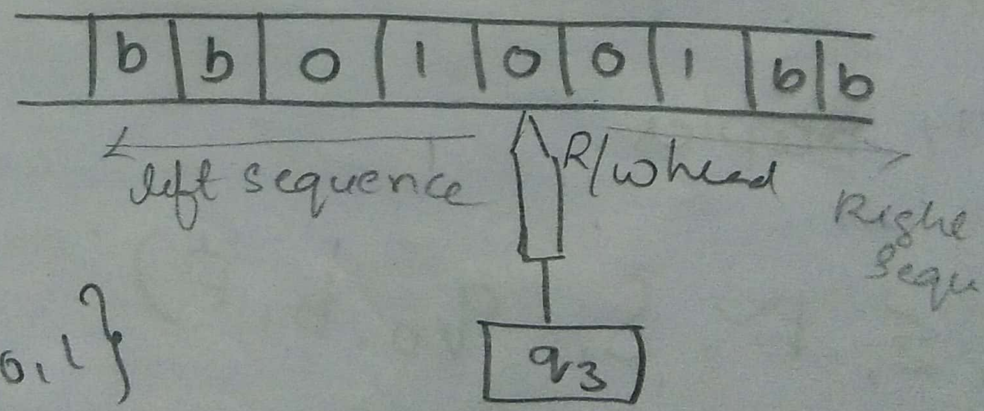


Language accepted by TM:

Representation of TM:

1. Instantaneous Description
2. Transition table
3 Transition Diagram.

Instantaneous Description.



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

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

0 1 q_3 0 0 1

← left sequence of i/p

present state

present symbol under R/w head.

→ Right sequence of i/p

Transition table:

State	0	1	b
$\rightarrow q_1$	$0Rq_1$		$1Lq_2$
q_2	$0Rq_3$	$0Lq_2$	$1Lq_2$
q_3	bRq_3	$1Rq_3$	

Consider an input string 00

⇒ transition starts from left to Right

$$\underset{\uparrow\, q_1}{0}0$$

$$\rightarrow \boxed{q_1 \xrightarrow{0} 0Rq_1}$$

$$00 \vdash 0\underset{\uparrow\, q_1}{0} \vdash 00\underset{\uparrow\, q_1}{b}$$

right no symbol then put blank

$$q_1 \xrightarrow{b} 1Lq_2$$

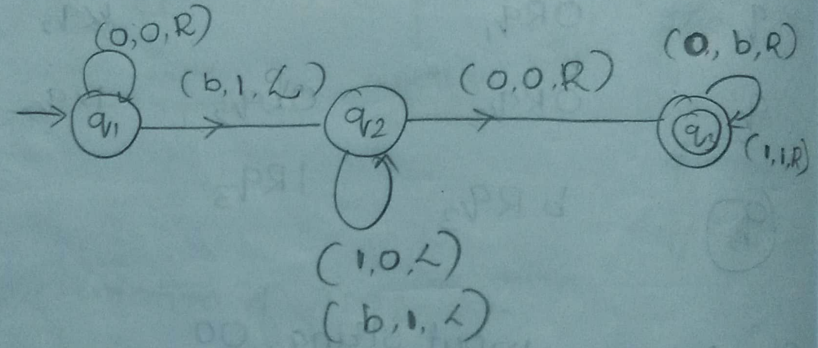
$00b \vdash \underset{\substack{\uparrow \\ q_2}}{001} \vdash 00\underset{\substack{\uparrow \\ q_3}}{1} \vdash 001\underset{\substack{\uparrow \\ q_3}}{b}$

$\boxed{q_2 \xrightarrow{0} 0Rq_3}$

$q_3 \xrightarrow{1} 1Rq_3$

right move no Symbol put blank.

Transition Diagram:



$\overset{q_1}{\downarrow}\!00 \vdash^{(0,0,R)} 0\overset{q_1}{\downarrow}\!0 \vdash^{(0,0,R)} 00\overset{q_1}{\downarrow}\!b \vdash^{(b,1,L)} \underset{\substack{\uparrow \\ q_2}}{0}01$

$\vdash^{(0,0,R)} 00\underset{\substack{\uparrow \\ q_3}}{1} \vdash 001\underset{\substack{\uparrow \\ q_3}}{b}$

reached final State

Language accepted by TM:

Let us consider TM

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

⇒ A string 'w' in language 'L' is said to be accepted by TM

if $q_0 w \vdash^{*} \underbrace{\alpha_1}_{\text{tape symbols}} f \underbrace{\alpha_2}_{\text{tape symbol}}$

$f \in F$

$\alpha_1, \alpha_2 \in \Gamma^{*}$

TM, M does not accept w if machine m halt in non-final state

1. halt, final, Accept
2. halt, nonfinal Reject
3. No halt, any state, Not accepted

Consider the following TM and describe the processing of (a) 011 (b) 0011 (c) 001

tape symbol

PS	0	1	x	y	b
$\rightarrow q_1$	xRq_2	-	-	-	bRq_5
q_2	$0Rq_2$	YLq_3	-	YRq_2	-
q_3	$0Lq_4$	-	xRq_5	YLq_3	-
q_4	$0Lq_4$	-	xRq_1	-	-
q_5	-	-	-	yRq_5	bRq_6
q_6	-	-	-	-	-

$\underset{q_1}{\overset{\downarrow}{0}}11 \vdash x\underset{q_2}{\overset{\downarrow}{1}}1 \vdash \underset{q_3}{x}y1 \vdash x\underset{q_5}{y}1$

$\vdash xy\underset{q_5}{1} \vdash$

$q_5 \rightarrow 1$ (No rule)

then halt

halt, non final state $\Rightarrow$ rejected.

$\underset{q_1}{0}011 \vdash x\underset{q_2}{0}11 \vdash x0\underset{q_2}{1}1 \vdash x\underset{q_3}{0}y1$

$\vdash \underset{q_4}{x}0y1 \vdash x\underset{q_1}{0}y1 \vdash xx\underset{q_2}{y}1$

$\vdash xxy\underset{q_2}{\underset{\uparrow}{1}}$ $\vdash xx\underset{q_3}{\underset{\uparrow}{y}}y$ $\vdash x\underset{q_3}{\underset{\uparrow}{x}}yy$

$\vdash xx\underset{q_5}{\underset{\uparrow}{y}}y$ $\vdash xxy\underset{q_5}{\underset{\uparrow}{y}}$ $\vdash xxyy\overset{\downarrow}{b}\, q_5$

$\vdash xxyyb\underset{q_6}{\underset{\uparrow}{b}}$

halt, final state ⇒ Accepted

001

$\underset{q_1}{\underset{\uparrow}{0}}01$ $\vdash x\underset{q_2}{\underset{\uparrow}{0}}1$ $\vdash x0\underset{q_2}{\underset{\uparrow}{1}}$ $\vdash x\underset{q_3}{\underset{\uparrow}{0}}y$

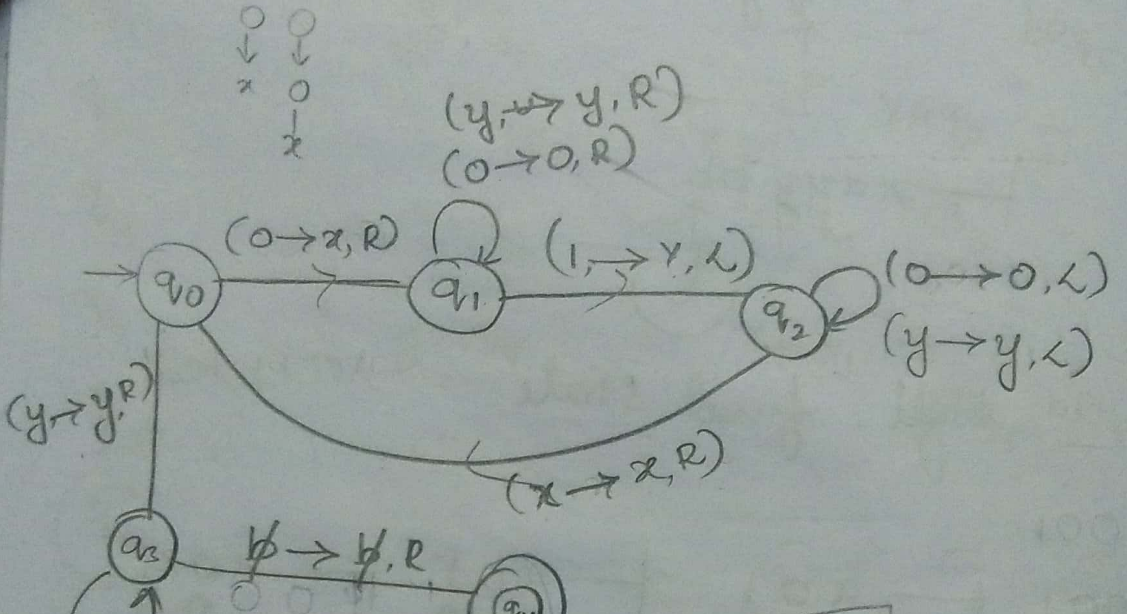
$\vdash \underset{q_4}{\underset{\uparrow}{x}}0y$ $\vdash x\underset{q_1}{\underset{\uparrow}{0}}y$ $\vdash xx\underset{q_2}{\underset{\uparrow}{y}}$

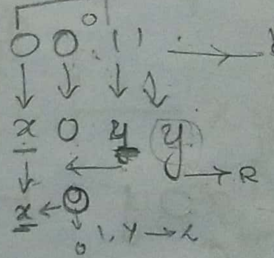
$\vdash xxy\underset{q_2}{\underset{\uparrow}{b}}$ $\boxed{q_2 \to b}$ No move

halt, Non final state ⇒ rejected.

Design a Turing machine that accept language $L = \{0^n 1^n \mid n \geq 1\}$

$L = \{01,\ 0011,\ 000111 \ldots\}$





$\underset{q_0}{0}011$

$0011 \vdash x\underset{q_1}{0}11 \vdash x0\underset{q_1}{1}1 \vdash x\underset{q_2}{0}y1 \vdash \underset{q_2}{x}0y1$

$\vdash x\underset{q_0}{0}y1 \vdash xx\underset{q_1}{y}1 \vdash xxy\underset{q_1}{1}$

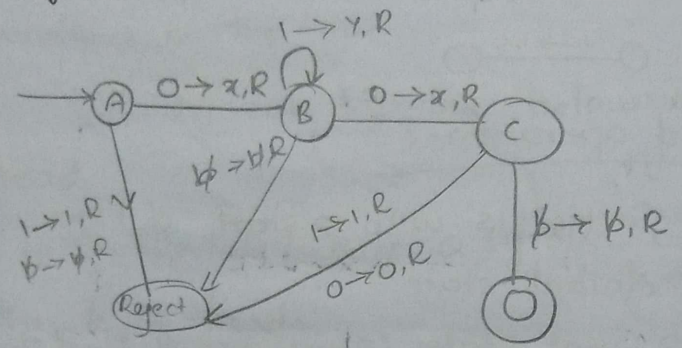
$\vdash xx\underset{q_2}{y}y \vdash x\underset{q_2}{x}yy \vdash xx\underset{q_0}{y}y$

$\vdash x x y \underset{\uparrow q_3}{y}$ $\vdash x x y y \underset{q_3}{b}$ $\vdash x x y y \phi \underset{\uparrow q_4}{\phi}$

halt, q_4 (final state)

$\Rightarrow$ accepted.

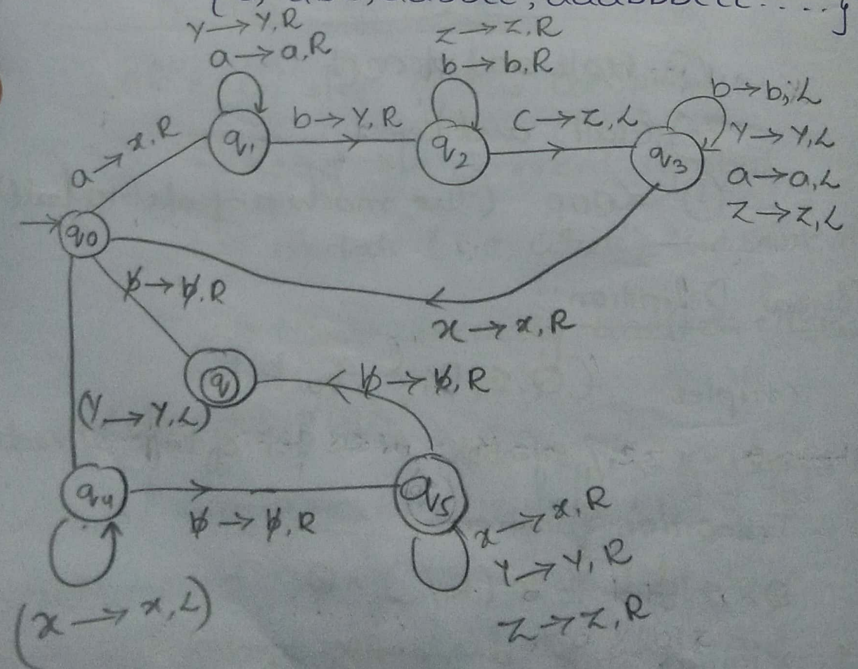
Design a Turing Machine which recognizes the languages $L = 01^*0$



Design a TM that accept language

$L = \{a^n b^n c^n \mid n \geq 0\}$

$L = \{\epsilon, abc, aabbcc, aaabbbccc \ldots\}$



$\underset{q_0}{a}abbcc \vdash x\underset{q_1}{a}bbcc \vdash xa\underset{q_1}{b}bcc$

$\vdash xay\underset{q_2}{b}cc \vdash xayb\underset{q_2}{c}c \vdash xayb\underset{q_3}{z}c$

$\vdash xa\underset{q_3}{y}bzc \vdash x\underset{q_3}{a}ybzc \vdash \underset{q_3}{x}aybzc$

$\vdash x\underset{q_0}{a}ybzc \vdash xx\underset{q_1}{y}bzc \vdash xxy\underset{q_1}{b}zc$

$\vdash xxyy\underset{q_2}{z}c \vdash xxyyz\underset{q_2}{c} \vdash xxyy\underset{q_3}{z}z$

$\vdash xxy\underset{q_3}{y}zz \vdash xx\underset{q_3}{y}yzz \vdash x\underset{q_3}{x}yyzz$

$\vdash xx\underset{q_0}{y}yzz \vdash x\underset{q_4}{x}yyzz \vdash \underset{q_4}{x}xyyzz$

$\vdash \underset{q_4}{b}xxyyzz \vdash \underset{q_5}{x}xyyzz \vdash x\underset{q_5}{x}yyzz$

$\vdash xx\underset{q_5}{y}yzz \vdash xxy\underset{q_5}{y}zz \vdash xxyy\underset{q_5}{z}z \vdash xxyyz\underset{q_5}{z}$

...}

b,L

→y,L

→0,L

→z,L

$\vdash xxyyzzx\,\not{b}$ $\vdash xxyyzz\,\not{b}\not{b}$

($\uparrow q_5$ under the first, $\uparrow q$ under the second)

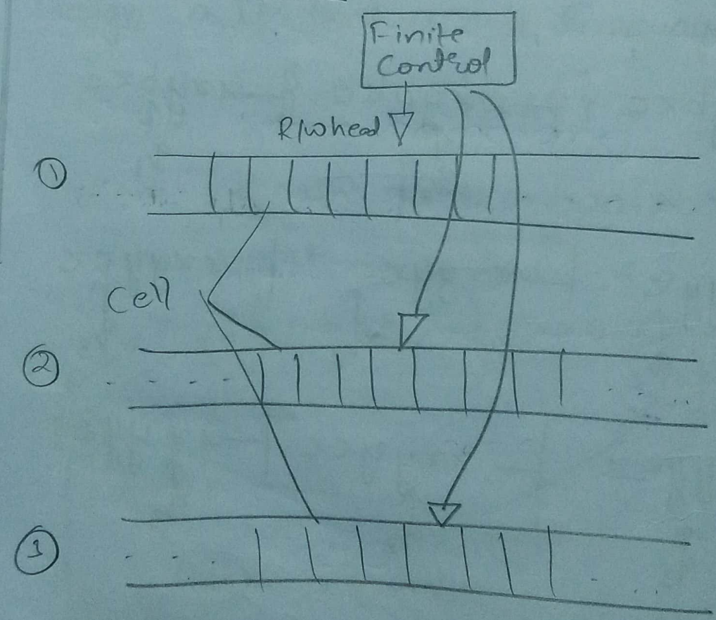
Halt → final state ⇒ accepted

Variants of TM:

29-04-1

1. Multitape TM
2. Non-Deterministic TM

Multitape TM:-



The TM having more than one tape is called Multitape Turing Machine.

Initially

1. Input symbol ε is placed in the first tape.

2. All the often other cell of all tape hold blank "β".

3. Finite control is in initial state.

4. The head of firsttape at left end of the input

5. All the other head arbitrary cells.

6. Each tape head move left (L), Right (R) or stationary (S)

neither left nor right

$\delta : Q \times \Gamma^k \rightarrow Q \times [\Gamma \times L, R, N]^k$

Non-Deterministic TM:

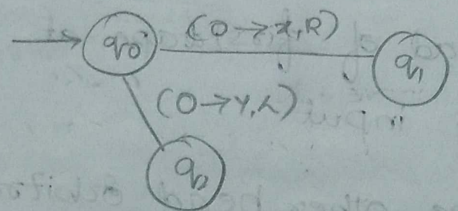
$$TM \begin{cases} DTM \\ NDTM \end{cases}$$

A Non-deterministic TM have different transition function δ, such that for some state q and tape symbol x then

$$\delta(q, x) \rightarrow \{ (q_1, Y_1, D_1)\ (q_2, Y_2, D_2) \ldots (q_k, Y_k, D_k) \}$$

ex: $\delta(q_0, 0) \longrightarrow \{(q_1, Y_1, R)\ (q_2, Y_2, L) \ldots (q_n, Y_n, S)\}$

ex: TM



Design a TM which recognizes palindromes over alphabet $\{a, b\}$, where $|w| \geq 0$

ww^R [even palindrome]

$w = ab$ $w^R = ba$ $w = aba$

abba abaaba

$|w| \Rightarrow$ odd or even

$|ww^R| \Rightarrow$ always even that's why even palindrome

$|w| \geq 0$

$L = \{\epsilon, aa, bb, abba, baab, abbbba, babbab \ldots\ldots\}$

Consider string

aabbbbaa

Scanned with CamScanner

a a b b b b a a

↓

$\not{b}$ a b b b b a a $\not{b}$

⟶ Right upto blank

After blank (end of string) move left and make right side first symbol as blank

$\not{b}$ a b b b b a $\not{b}$

⟵ left

$\not{b} \rightarrow \not{b}, R$

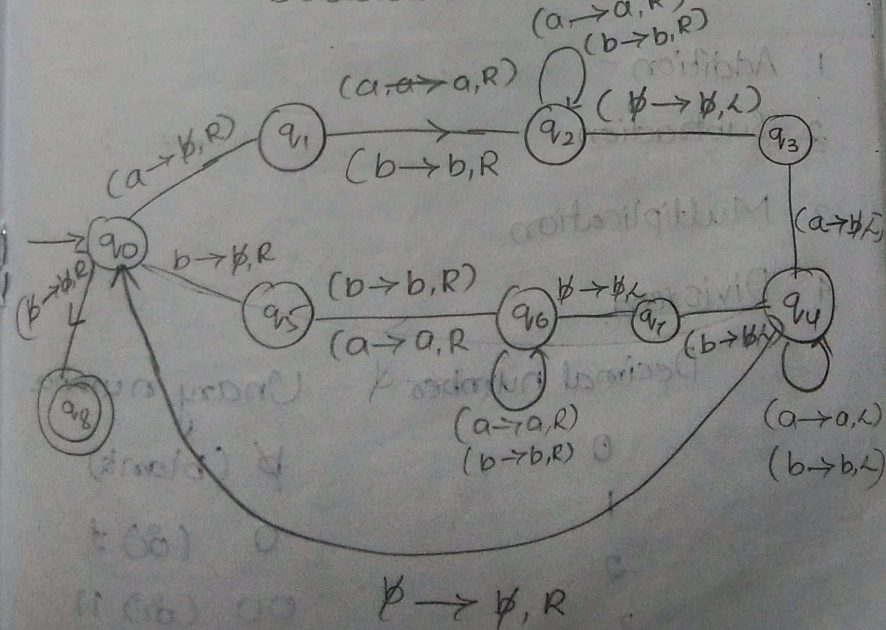
$a \rightarrow \not{b}, R$

$\not{b}$ $\not{b}$ b b b b a $\not{b}$

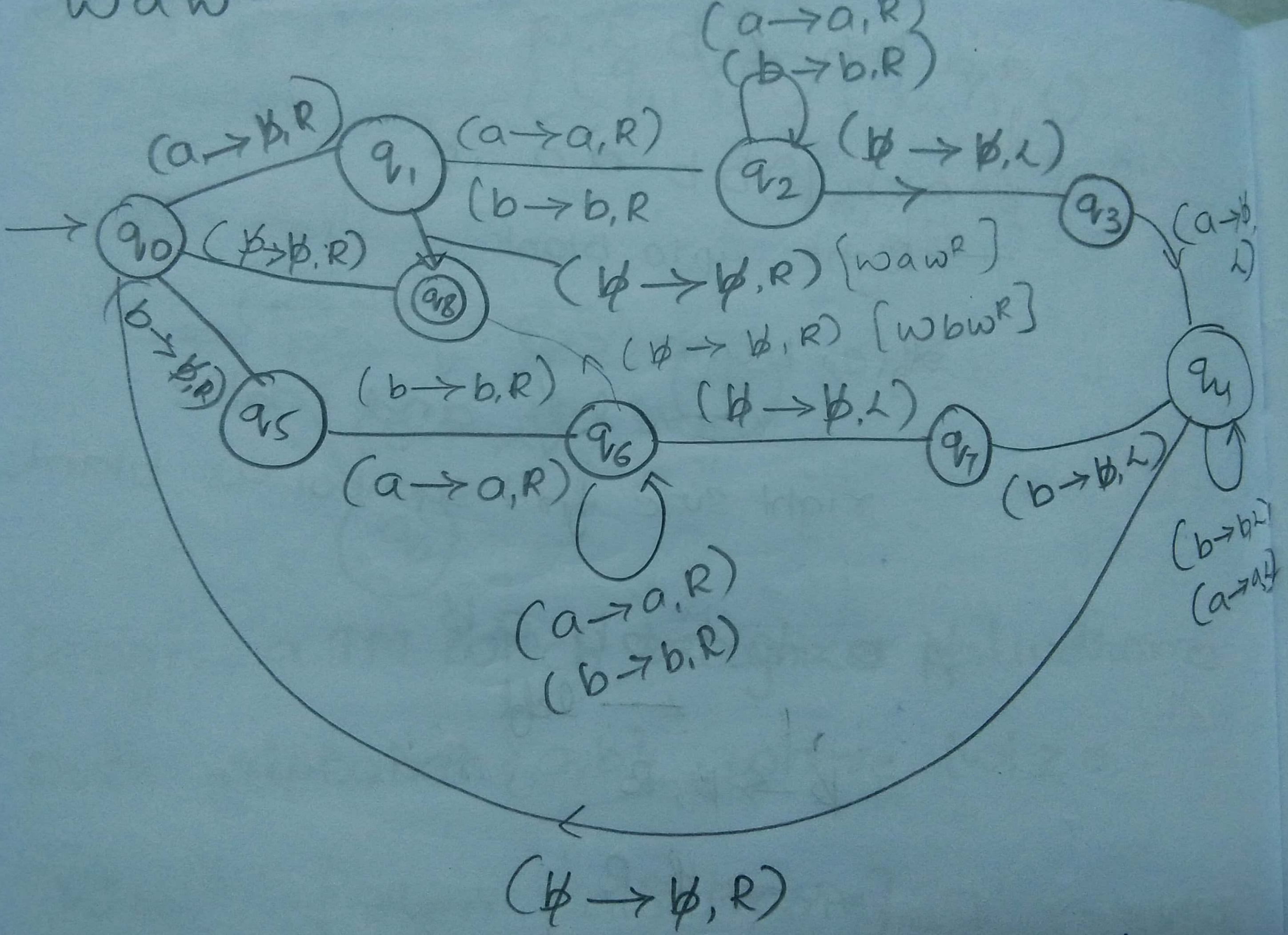
→ right $a \rightarrow \not{b}, R$

another string

b b a a a a b b



waw^R


q0
q1
q2
q3
q4
q5
q6
q7
q8
(a → b̸, R)
(a → a, R)
(b → b, R)
(a → a, R)
(b → b, R)
(b̸ → b̸, L)
(a → b̸, L)
(b̸ → b̸, R)
(b̸ → b̸, R) [waw^R]
(b̸ → b̸, R) [wbw^R]
(b → b̸, R)
(b → b, R)
(a → a, R)
(b̸ → b̸, L)
(a → a, R)
(b → b, R)
(b → b̸, L)
(b → b, L)
(a → a, L)
(b̸ → b̸, R)

TM can be used as Computation

1. Addition
2. Subtraction
3. Multiplication
4. Division.

Decimal number	Unary number
0	$\not{b}$ (blank)
1	0 (or) 1
2	00 (or) 11
3	000 (or) 111

:

5 00000 (or) 11111

n n 0's (or) n 1's

Addition:

Design a TM to Addition of two unary number 5 + 3 = 8

0	0	0	0	0	#	0	0	0

↓ Separator

0 0 0 0 0 0 0 0

↓

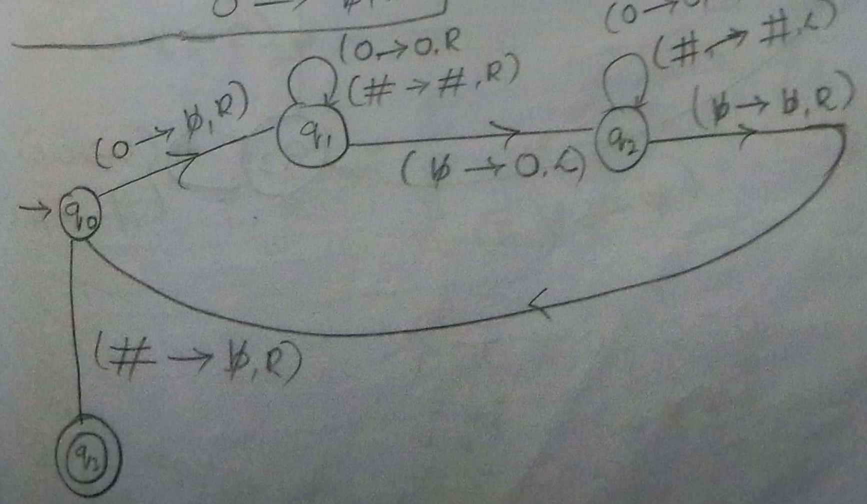
$\not b$ 0 0 0 0 0 0 0 $\not b$

→ Right ↓ 0

left ←

$\not b \rightarrow \not b, R$

$0 \rightarrow \not b, R$



Design a TM to substraction of two unary number 5 - 3 = 2

0	0	0	0	0	#	0	0	0

m>n ✓

m = 5 ; n = 3

ø ø ø 0 0 # ø ø ø

ɓ → ② Right

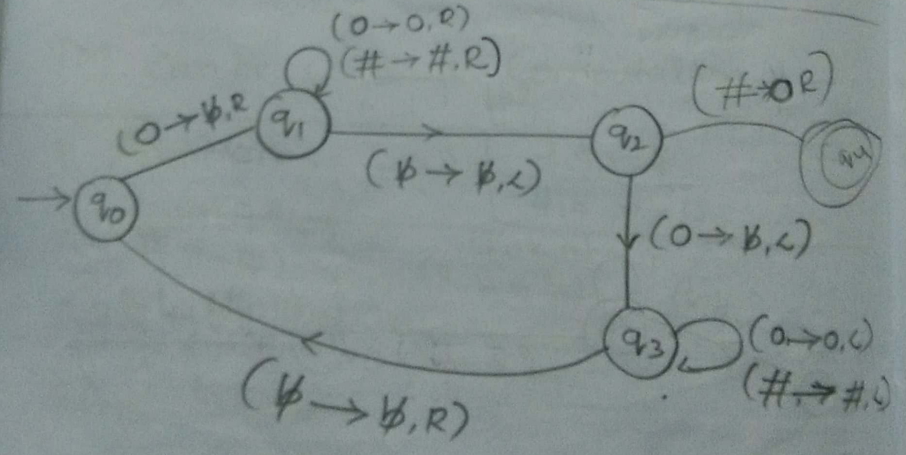
0 ɓ

ɓ → ɓ, L

0 → ɓ, L

left ←

(ɓ → ɓ, R)



0 0 0 0 0 # 0 0 0

ɓ ɓ ɓ ɓ b b b

↳ for this blank no pair then we can recover that '0' (or) make # symbol to '0' then

Design a TM for unary Multiplication

$2 \times 3 = 6$

$00 \times 000 = 000000$

0	0	#	0	0	0	#	b̸	b̸	b̸	b̸	b̸	b̸

b̸ 0 # x 0 0 # b̸

0 0 # 0

left

$(x \rightarrow x, R)$

x 0 # 0 b̸

0 # 0 0

left

x # 0 0 0

left

b̸ 0 # x x x #

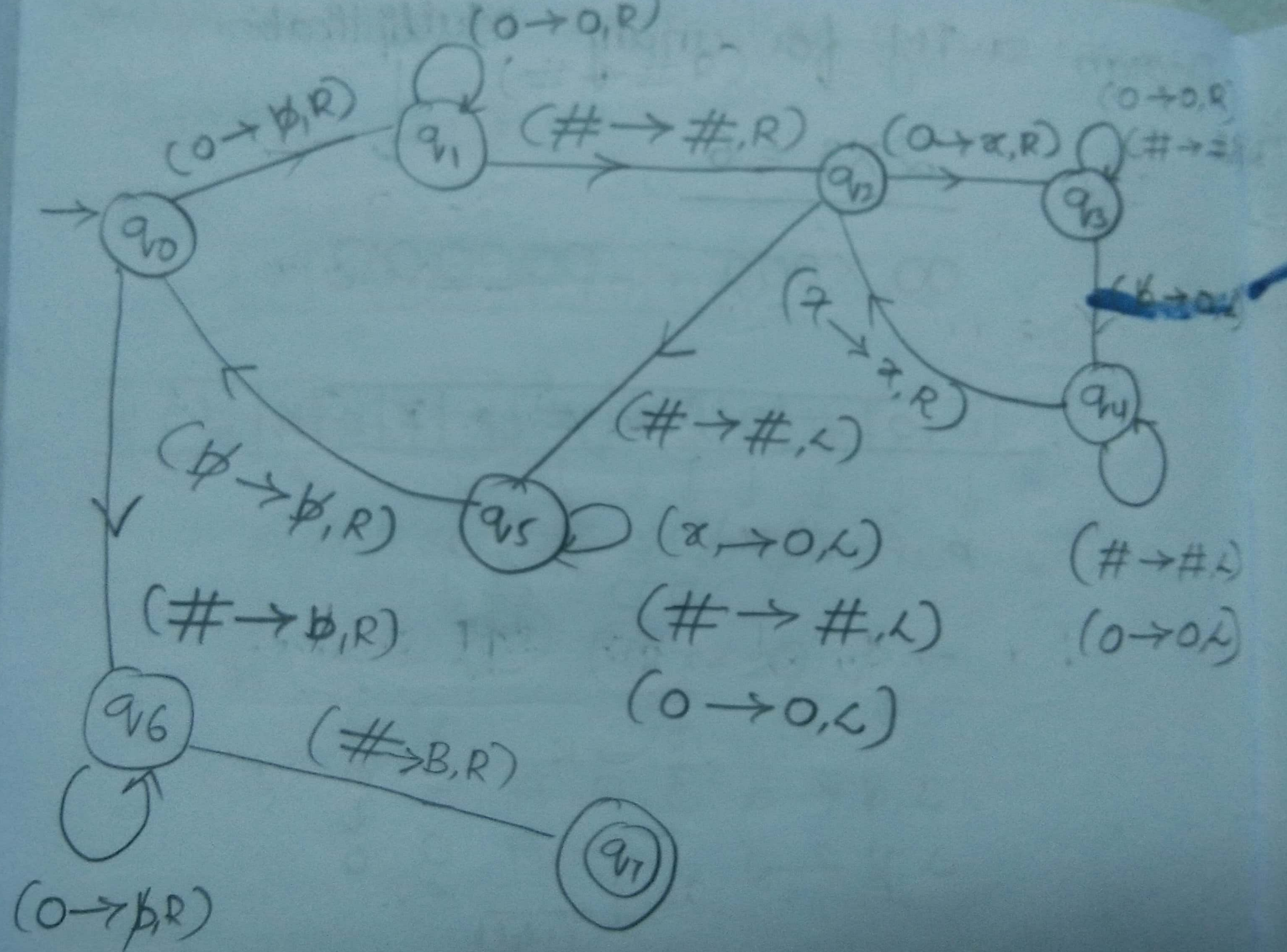
$(x \rightarrow 0, L)$

b̸ 0 # 0 0 0 # 0 0 0

$(b̸ \rightarrow b̸, R)$

b̸ b̸ # 0 0 0 # 000 000

x x x #


(0→0,R)
(0→ɓ,R)
q1
(#→#,R)
q2
(0→x,R)
(0→0,R)
(#→#
q3
q0
(x→x,R)
q4
(#→#,L)
(ɓ→ɓ,R)
q5
(x→0,L)
(#→#,L)
(0→0,L)
(#→#,L)
(0→0,L)
(#→ɓ,R)
q6
(#→B,R)
q7
(0→ɓ,R)

Design a TM for unary Division 30.04

$6/2 = 3$

000000 / 00 = 000

0	0	#	0	0	0	0	0	0	#	b	b	b

1 Scan   Scan   make

2 Sca   000

3 Scan

00 # 000000 # b b b

b 0 # x

(0→b,R) → (0→0,R) (#→#,R) (0→x,L)

(#→#,L) (0→0,L) (b→0,R)

00 # 0 000000 $\not b$ $\not b$ $\not b$

$(0 \to x, R)$ $(\# \to \#, R)$ $(x \to x, R)$ $(0 \to x, L)$

$(x \to x, L)$ $(\# \to \#, L)$ $(\not b \to 0, R)$

$(\#, x, 0 \to \#, x, 0, R)$

$(\not b \to 0, L)$ upto left blank

$(\not b \to \not b, R)$

00 # xx 00 00 # 0 $\not b$ $\not b$

$\not b$ 0 # x x x

0 x # x x x x

00 # xxxx $\longrightarrow$ # 00 $\not b$

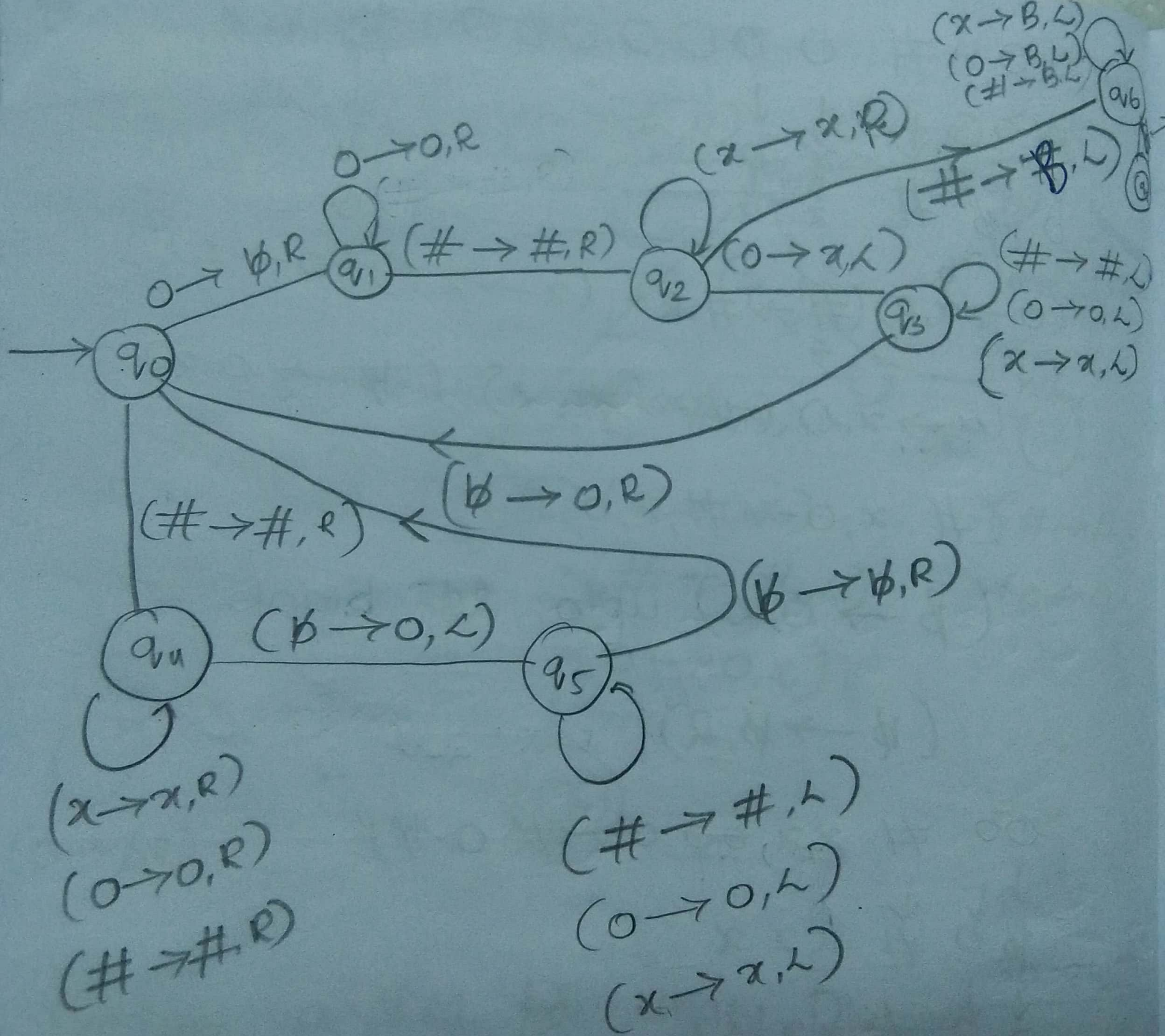
$\not b$ 0 # xxxxx

0 $\not b$ # xxxxxx $\longrightarrow$ # 000

00 # xxxxxx # 000

$\not b$

make it all blank


q0
q1
q2
q3
q4
q5
q6
0 → ɸ, R
0 → 0, R
(# → #, R)
(x → x, R)
(0 → x, L)
(# → B, L)
(x → B, L)
(0 → B, L)
(# → B, L)
→ ɸ, R
(# → #, L)
(0 → 0, L)
(x → x, L)
(ɸ → 0, R)
(# → #, R)
(ɸ → ɸ, R)
(ɸ → 0, L)
(x → x, R)
(0 → 0, R)
(# → #, R)
(# → #, L)
(0 → 0, L)
(x → x, L)

⇒ Desi

b

(# → #, R)

(0 →

(x → x, L)

0	0	#	0	0	0	0	0	0	#	B	B	B
B	0	#	x	0								
B	0	#	x	0								
0	B	#	x	x								
0	0	#	x	x	0	0	0	0	#	0	B	B
B	0	#	x	x	x							
0	B	#	x	x	x	x	0	0	#	0	0	B
0	0	#	x	x	x	x	0	0	#	0	0	B
B	0	#	x	x	x	x	x					
0	B	#	x	x	x	x	x	x				
0	0	#	x	x	x	x	x	x	#	0	0	0
B	0	#	x	x	x	x	x	x	B			
B	B	B	B	B	B	B	B	B	B	0	0	0

→ right

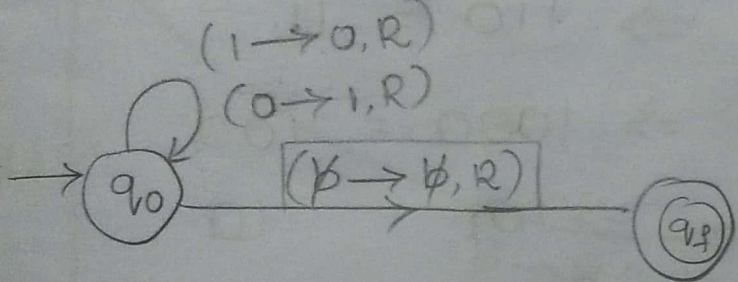
⇒

⇒ Design a TM for 1's complement of binary.

String: 10101

01010 → 1's complement

0 → R, 1 → R, 0 → R, 1 → R, 0 → R, b → R, b → R



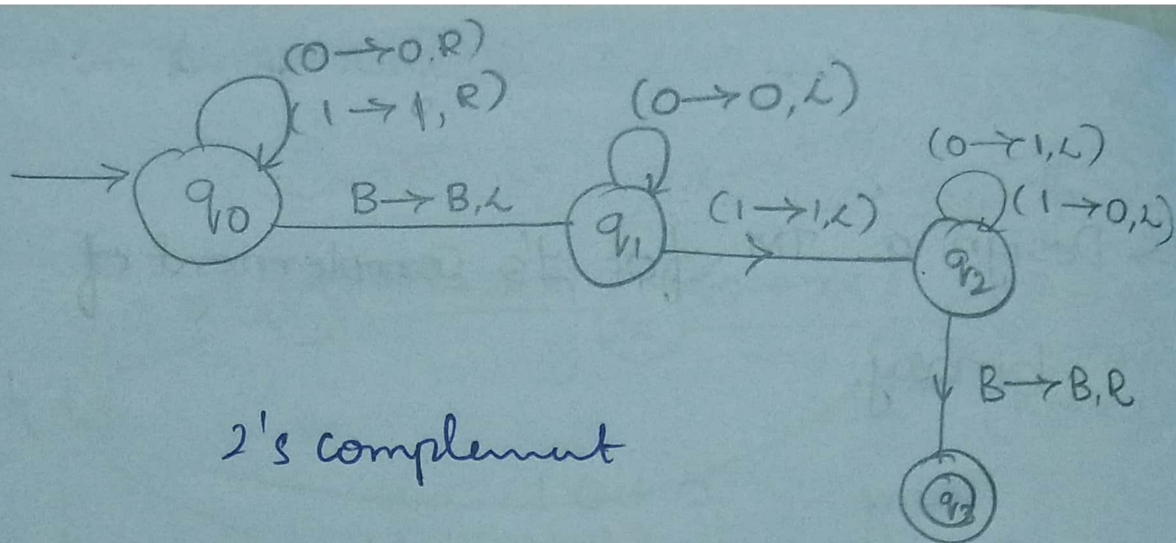
⇒ Design a TM for 2's complement of binary

10101	100010
01010	011101
1	1
01011	011110

⇒ 10101

01011

⇒ whenever 1 appear make it 1 after 0's replace with 1, with 0



2's complement

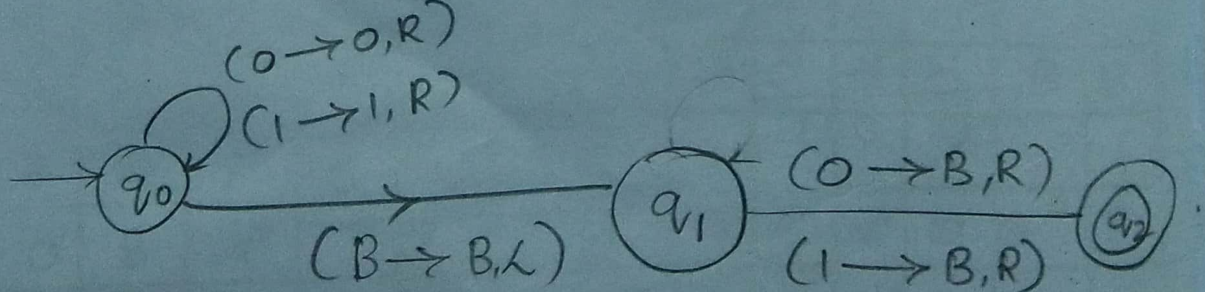
③ Design a TM for finding half of the number.

6 = 3 => 110 = 11

8 = 4 => 1000 = 100

5 = 2 => 101 = 10

=> Here last number becomes blank



Programming Techniques for TM:

Construction of turing machine is a process of writing out the complete set of states and next move function.

This is a totally conceptual phenomenon. The TM can be designed with the help of some conceptual tools.

Those tools are:

1. Storage in finite control
2. Multiple tracks
3. Checking off symbols.
4. Subroutine.

Storage in finite Control:

The model of TM has a finite control. This finite control can be used to hold some amount of information.

The finite automata control stores the Information in pair of elements such as the current state and the current symbols pointed by the tape head.

⇒ This finite control used to know the symbols during any crashes.

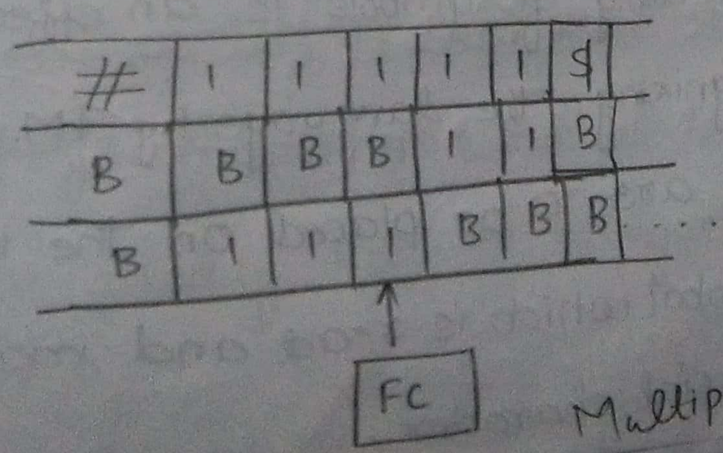
Ex: $\delta\left[[q_0, 0], 1\right] \rightarrow \left[[q_1, 1], \not{b}, R\right]$

⇒ This means that if finite control shows the initial state is q_0 and stores the current symbol '0' if it reads the symbol 1 then the machine goes to next state q_1 & replace that 1 by $\not{b}$ and moves to right.

Multiple Tracks:-

If the input tape is divided into multiple tracks.

Then the input tape will be as follows:



Multiple Tracks

Ex:- The input tape has multiple tracks on the first track the input which is placed and sorrounded by # and $.

The unary number equivalent to 5 is placed on the input tape, on the first track

On the second track unary number 2 is placed.

If we construct a TM which Substract 2 from 5 we get the answer on the third track and that is 3, in unary form.

Thus, the TM is for subtracting two unary numbers with the help of multiple tracks.

Checking off symbols:

Checking off symbols is an effective way of recognizing the language by TM. The Symbols are to be placed on the input tape. The symbol which is read and marked by any special character.

The tape head then can be moved to the right & left.

Ex: Check equal no. of a's & b's

B	a	a	a	b	b	b	B
B	*	a	a	*	b	b	B
	*	*	a	*	*	b	B
	*	*	*	*	*	*	B

In the checking off symbols, each symbol is marked by special character.

In the above example, each symbol is marked with "*" character.

Here "*" is the check off symbol.

It can be #, -, /, + ...

Subroutine:

In the high level languages use of subroutines built the modularity in the program development process.

Ex: Build a TM to recognize the language $0^n 1^n 0^n$

Sol: we already have a Turing machine $0^n 1^n$.

Use this turing machine as subroutine

=> By using this subroutine, we can able to develop a TM for $0^n 1^n 0^n$.

Step1: 00 11 00

$x x \; y y \; 00$

$\underbrace{x x \; y y}_{y^n} \; \underbrace{00}_{0^n}$

Step2: Build a Similar TM to recognize $y^n 0^n$

Step3: Build a larger (or) final turing machine by Combining these 2 Smaller TM together into one larger TM.

Extensions to the Basic Turing Machine:

Infact, there are lots of extensions we can make to our basic turing machine model.

⇒ The enhanced versions of turing machine and later, the languages accepted by a TM is recursive enumerable language.

⇒ These extensions to the standard (or) basic turing machine make it easier to write turing machine program but none of them increase the power

Scanned with CamScanner

of turing machine because every extended machine has an equivalent power to basic machines.

1. Two-way Infinite Tape TM
2. Multi-tape Turing Machine
3. Multi-Head TM
4. k-Dimensional TM
5. Off Line TM
6. Non-Deterministic TM

Two Stack PDA

Two-way Infinite tape Turing Machine:

A TM in which there is an infinite number of sequences of blanks on either side of the tape is said to be two-way infinite tape TM.

The advantage of having this type of TM is that there is no possibility of jumping off the left end of the tape.

B	1	0	1	B

$\uparrow q_0$

Standard TM with boundary on Left-side

B	1	0	1	B	B

Two-way TM with no boundaries.

Formally, a two-way infinite tape TM is represented as 7-tuple

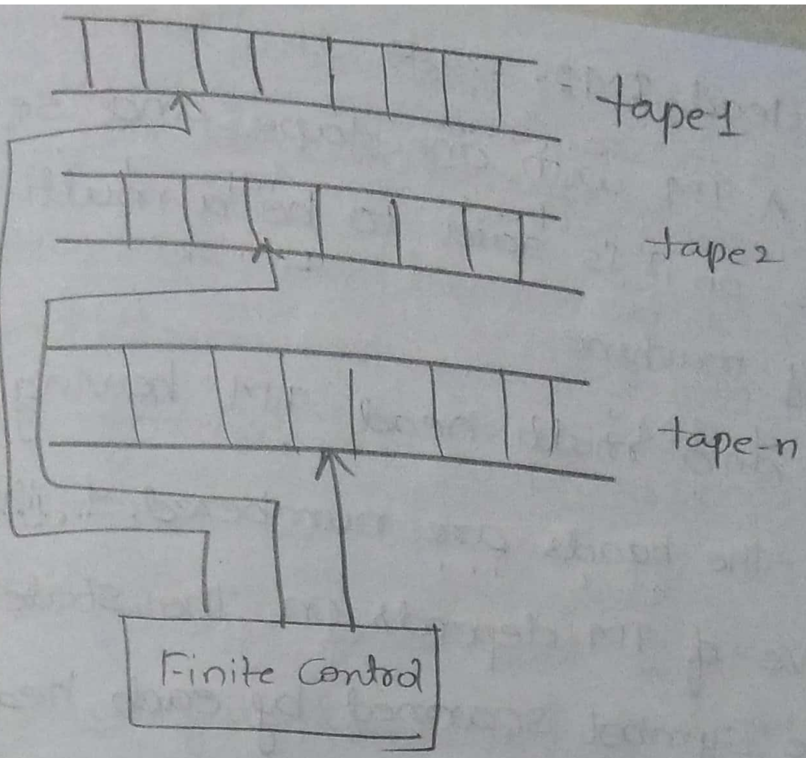
$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

where $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$

Multi Tape Turing Machine:

A turing machine with more than one tape and each tape having it's own independent Read/write head is said to be multi-tape TM.

The language accepted by an n-tape turing machine can be accepted by one-tape turing machine



Multi-tape TM

Formally, a Multitape TM is represented as 7-tuple

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

$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k$

K : k is the number of tapes.

⇒ The advantage of having multi-tape TM is that the design of some functions like copying, reversing and verifying whether a string is palindrome or not can be carried out in much easier way as compared to the design of the corresponding standard TMs.

Multi Head TM:-

A TM with one tape and several heads on it is said to be a multi-head turing machine.

In a multi-head TM having k heads [k ≥ 2 - the heads are numbered 1 through k] a move of TM depends on the state and on the symbol scanned by each head.

In one move, each head can move independently to left or right or remain stationary.

Also, the use of multi ~~tapes~~ heads can sometimes simplify the construction of complex TMs drastically.

K-Dimensional TM:

A TM having its tape with an infinite k-dimensional grid, is said to be a k-dimensional TM.

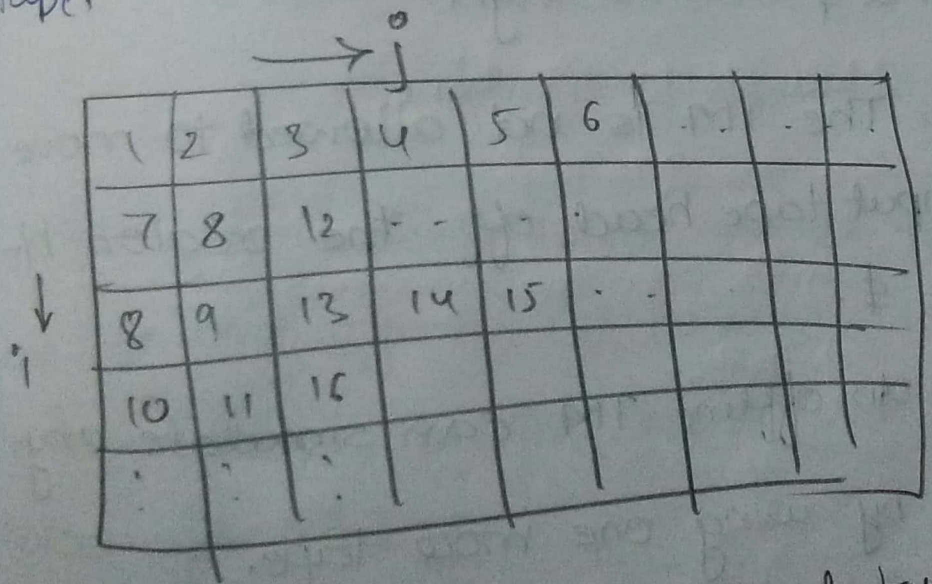
This type of TM has two finite controls

1. Read/write head

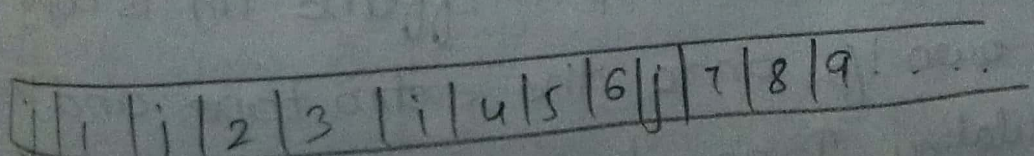
2. Two dimensional tape

⇒ This tape has infinite extension to right and down. It is divided into small squares formed due to corresponding rows and columns.

⇒ TM with one-dimensional tape is equally powerful to that of 2-dimensional tape.



Two-way dimensional tape



⇒ The head of two-dimensional tape moves one square up, down, left & right.

⇒ The k-dimensionals TMs are much more useful than standard TMs for solving sophisticated problems. This type of TM can also be combined with other extensions of TM.

Off-line TM:-

An off-line TM is a multi-tape TM whose input tape is of type read-only. The input is endmarked by a ¢ on the left and $ on the right.

⇒ The TM is not allowed to move the input tape head, off the region b/n ¢ and $.

⇒ An offline TM can simulate any TM, T, by using one more tape.

⇒ The first step in offline TM is copying its own i/p on to the extra tape, and then simulating T as if the extra tape were T's input. The offline TM is useful in the case of limiting the amount of storage space, which is less than the input length.

Non-deterministic TM:-

In a standard TM/deterministic-TM, for the current state (q) and the symbol (s) being scanned by the tape head, the TM performs the unique action in terms of writing a symbol in the cell being scanned and also repositioning to the head to left or right (or) no Move (M) at all.

$$\delta(q,s) = (q_i, s_i, M_i)$$

for $i = 1$

In a Non-deterministic TM, for the current state (q) and symbol (s) being scanned by the tape head, the TM has a finite set of choices for writing a symbol in the cell being scanned and repositining the head to left or right or no Move (M)

$$\delta(q,s) = (q_i, s_i, M_i) \text{ for } i = 1, 2 \ldots$$

→ A Non-deterministic TM

$$\delta : Q \times \Gamma \rightarrow P[Q \times \Gamma \times \{L, R\})$$

General Types of TM:

⇒ A Random Access TM

⇒ Universal TM

⇒ Alternating TM

⇒ Probabilistic TM

⇒ Oracle TM

Restricted Turing Machines:

1. Linearbounded Automaton
2. Multi-stack Machine
3. Counter Machine
4. Limits on the number of states & symbols.