TAFL

ONE SHOT Revision

Unit-5

TuringMachines and Recursive Function Theory

AKTU PYQs Covered I for Pdf Notes download app now

Theory of Automata and Formal Languages (BCS402)

SYLLABUS

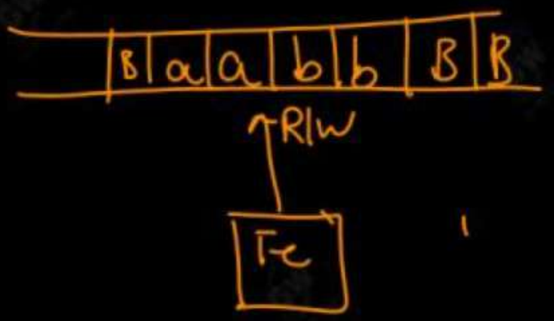
Unit – 5

Turing Machines and Recursive Function Theory : Basic Turing Machine Model, Representation of Turing Machines, Language Acceptability of Turing Machines, Techniques for Turing Machine Construction, Modifications of Turing Machine, Turing Machine as Computer of Integer Functions, Universal Turing machine, Linear Bounded Automata, Church's Thesis, Recursive and Recursively Enumerable language, Halting Problem, Post's Correspondance Problem, Introduction to Recursive Function Theory.

Turing machine

- Turing Machine was invented by Alan Turing in 1936 and it is used to accept Recursive Enumerable Languages (generated by Type-0 Grammar).
- Turing machines are used to study the properties of algorithms and to determine what problems can and cannot be solved by computers.
- A Turing machine is a finite automaton that can read, write, and erase symbols on an infinitely long tape.

- The tape is divided into squares, and each square contains a symbol.
- The Turing machine can only read one symbol at a time, and it uses a set of rules (the transition function) to determine its next action based on the current state and the symbol it is reading.



Derivation

- The Turing machine begins in the start state and performs the actions specified by the transition function until it reaches an accept or reject state. If it reaches an accept state, the computation is considered successful; if it reaches a reject state, the computation is considered unsuccessful

Turing machine

A TM is expressed as a 7-tuple (Q, τ, B, Σ, δ, q0, F) where:

Q is a finite set of states

τ is the tape alphabet (symbols which can be written on Tape)

B is blank symbol (every cell is filled with B except input alphabet initially)

Σ is the input alphabet (symbols which are part of input alphabet)

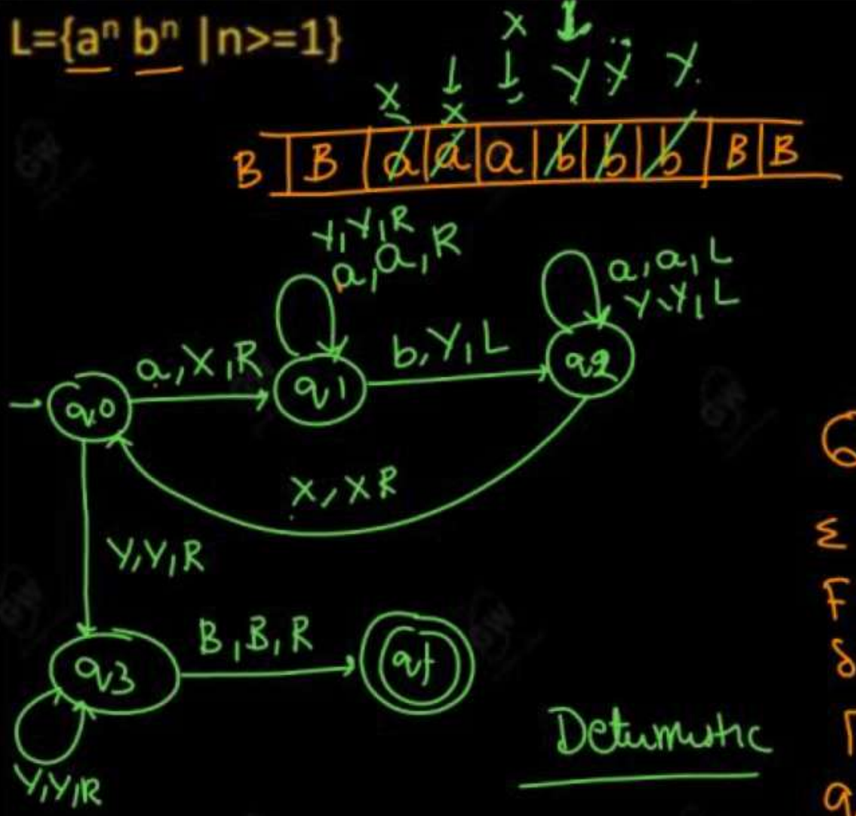
δ is a transition function $Q X \tau \rightarrow Q X \tau X (L/R)$

F is the set of final states. If any state of F is reached, input string is accepted

$q_0 = \{$ Intial State $\}$

Turing machine

$L=\{a^n b^n \mid n>=1\}$



Deterministic

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

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

$F = \{q_f\}$

δ

$\Gamma = \{a, b, X, Y, B\}$

$q_0 = \{q_0\}$

$B = \{B\}$

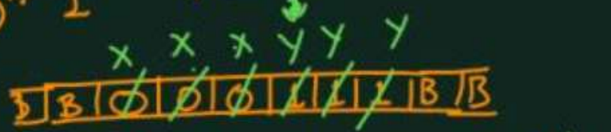
	a	b	X	Y	B
q0	(q_1, X, R)	—	—	(q_3, Y, R)	—
q1	(q_1, a, R)	(q_2, Y, L)	—	(q_1, Y, R)	—
q2	(q_2, a, L)	—	(q_0, X, R)	(q_2, Y, L)	
q3	—	—	—	q_3, Y, R	(q_f, B, R)
qf	—	—	—	—	—

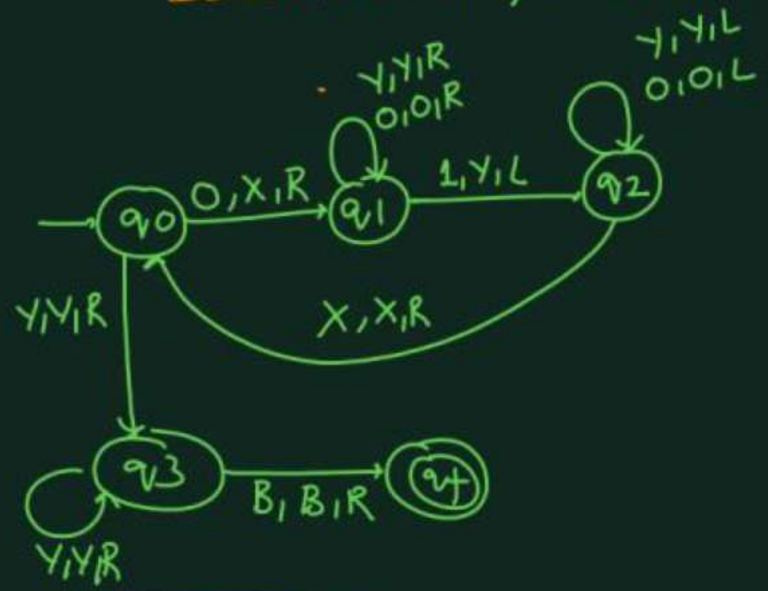
Turing machine

TRANSITION FUNCTION

$\delta(q0, a) = (q1, X, R)$
$\delta(q0, Y) = (q3, Y, R)$
$\delta(q1, a) = (q1, a, R)$
$\delta(q1, Y) = (q1, Y, R)$
$\delta(q1, b) = (q2, Y, L)$
$\delta(q2, a) = (q2, a, L)$
$\delta(q2, Y) = (q2, Y, L)$
$\delta(q2, X) = (q0, X, R)$
$\delta(q3, Y) = (q3, Y, R)$
$\delta(q3, B) = (qf, B, R)$

$0^n 1^n \quad n>=1$



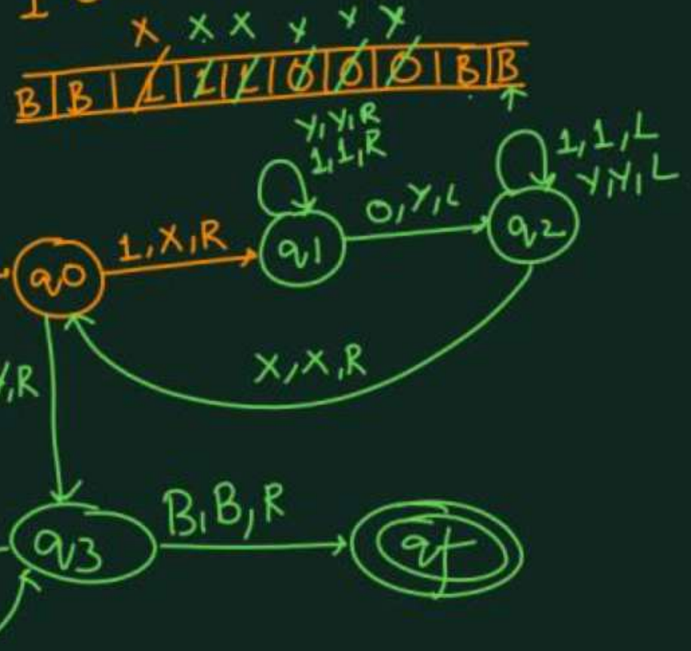


$1^n 0^n \quad n \geq 1$

	0	1	X	Y	B
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_f, B, R
q_f	–	–	–	–	–

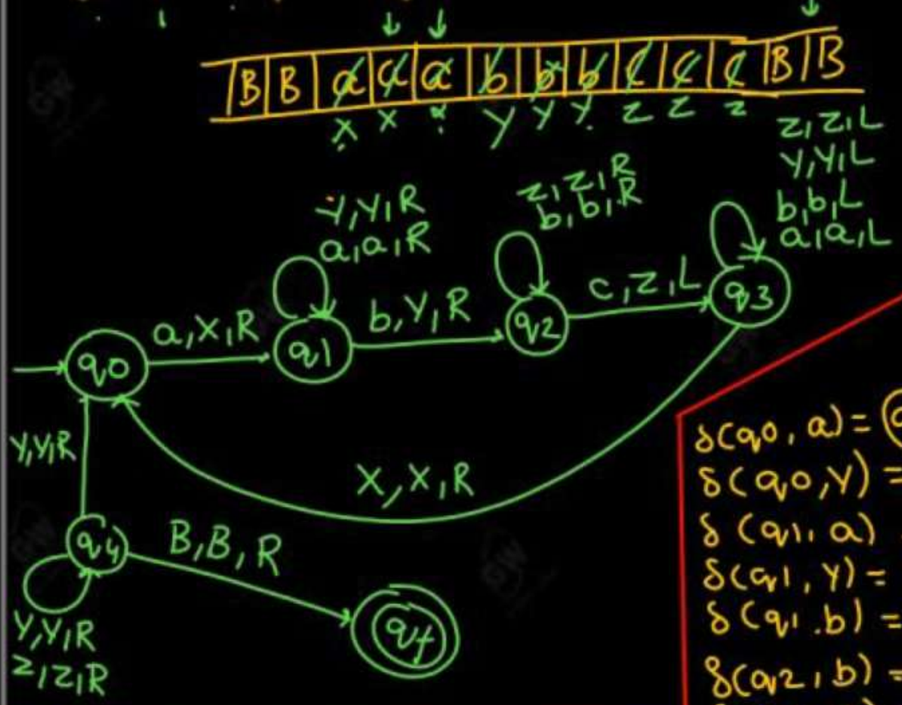
$\delta(q_0, 0) = (q_1, X, R)$
$\delta(q_0, Y) = (q_3, Y, R)$
$\delta(q_1, 0) = (q_1, 0, R)$
$\delta(q_1, 1) = (q_2, Y, L)$
$\delta(q_1, Y) = (q_1, Y, R)$
$\delta(q_2, 0) = (q_2, 0, L)$
$\delta(q_2, X) = (q_0, X, R)$
$\delta(q_2, Y) = (q_2, Y, L)$
$\delta(q_3, Y) = (q_3, Y, R)$
$\delta(q_3, B) = (q_f,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

$1^n 0^n$. $n \geq 1$



Turing machine

$L=\{a^n b^n c^n \mid n>=1\}$



$\delta(q_4, Y) = (q_4, Y, R)$
$\delta(q_4, Z) = (q_4, Z, R)$
$\delta(q_4, B) = (q_f, B, R)$

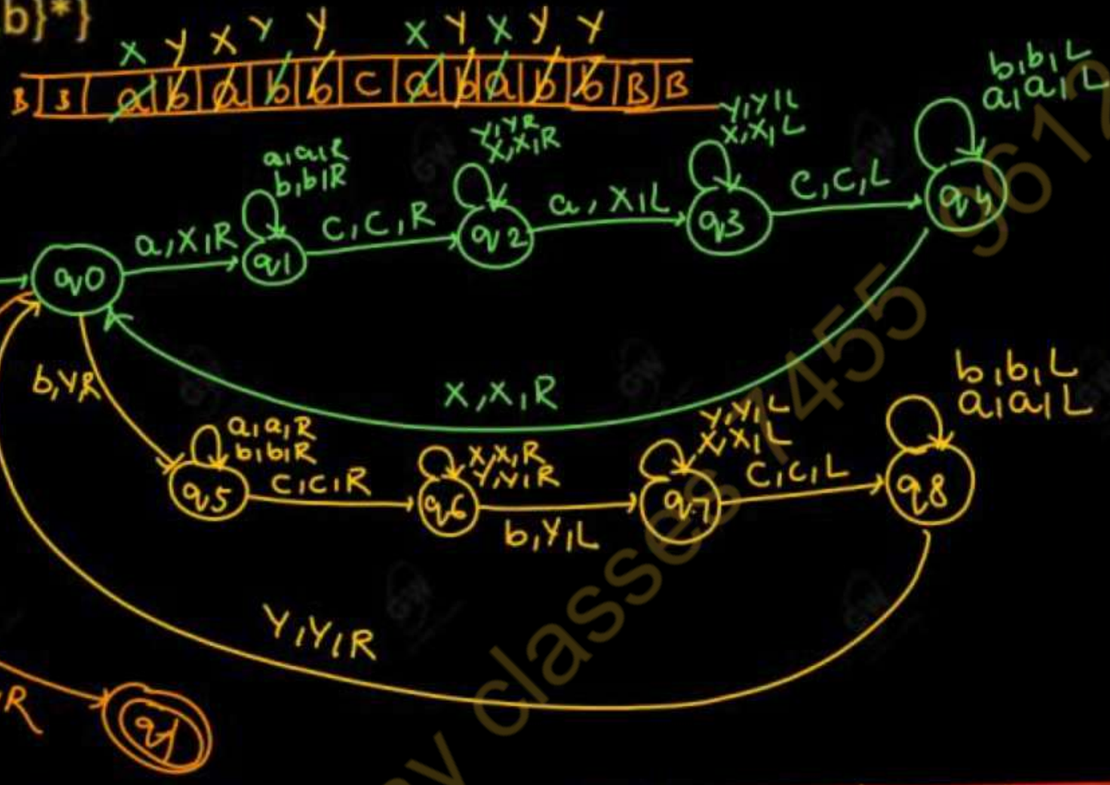
$Q = \{q_0, q_1, q_2, q_3, q_4, q_f\}$
$q_0 = \{q_0\}$ $q_f = \{q_f\}$
$B = \{B\}$ $\Sigma = \{a, b, c\}$ δ
$\Gamma = \{a, b, c, X, Y, Z, B\}$

$\delta(q_0, a) = (q_1, X, R)$
$\delta(q_0, Y) = (q_4, Y, R)$
$\delta(q_1, a) = (q_1, a, R)$
$\delta(q_1, Y) = (q_1, Y, R)$
$\delta(q_1, b) = (q_2, Y, R)$
$\delta(q_2, b) = (q_2, b, R)$
$\delta(q_2, Z) = (q_2, Z, R)$
$\delta(q_2, C) = (q_3, Z, L)$
$\delta(q_3, a) = (q_3, a, L)$
$\delta(q_3, b) = (q_3, b, L)$
$\delta(q_3, Y) = (q_3, Y, L)$
$\delta(q_3, Z) = (q_3, Z, L)$
$\delta(q_3, X) = (q_0, X, R)$

	a	b	c	X	Y	Z	B
q0	q_1, X, R	—	—	—	q_4, Y, R	—	—
q1	q_1, a, R	q_2, Y, R	—	—	q_1, Y, R	—	—
q2	—	q_2, b, R	q_3, Z, L	—	q_2, Z, R	—	—
q3	q_3, a, L	q_3, b, L	—	q_0, X, R	q_3, Y, L	q_3, Z, L	
q4	—	—	—	—	$q_4 Y, R$	q_4, Z, R	q_f, B, R
qf	—	—	—	—	—	—	—

Turing machine

L={WCW W Belongs {a,b}*}



① :

$\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_f\}$

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

$B = \{B\}$

$F = \{q_f\}$

$\Gamma = \{a, b, X, Y, C, B\}$

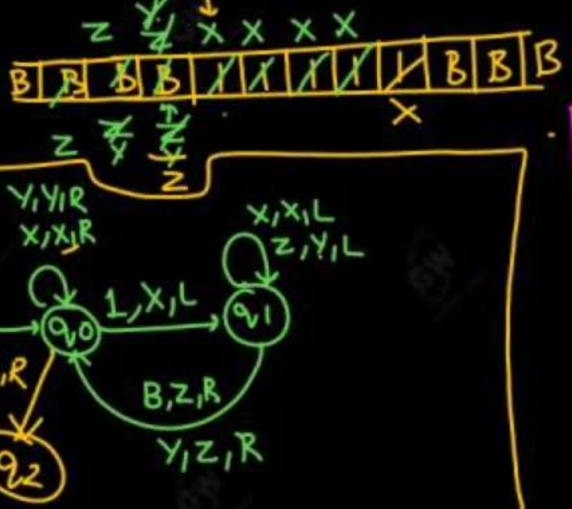
1. $\delta(q_0, a) = (q_1, X, R)$
2. $\delta(q_1, a) = (q_1, a, R)$
3. $\delta(q_1, b) = (q_1, b, R)$
4. $\delta(q_1, C) = (q_2, C, R)$
5. $\delta(q_2, Y) = (q_2, Y, R)$
6. $\delta(q_2, X) = (q_2, X, R)$
7. $\delta(q_2, a) = (q_3, X, L)$
8. $\delta(q_3, X) = (q_3, X, L)$
9. $\delta(q_3, Y) = (q_3, Y, L)$
10. $\delta(q_3, C) = (q_4, C, L)$
11. $\delta(q_4, a) = (q_4, a, L)$
12. $\delta(q_4, b) = (q_4, b, L)$
13. $\delta(q_4, X) = (q_0, X, R)$
14. $\delta(q_0, b) = (q_5, Y, R)$
15. $\delta(q_5, a) = (q_5, a, R)$
16. $\delta(q_5, b) = (q_5, b, R)$
17. $\delta(q_5, C) = (q_6, C, R)$
18. $\delta(q_6, X) = (q_6, X, R)$
19. $\delta(q_6, Y) = (q_8, Y, R)$
20. $\delta(q_6, b) = (q_7, Y, L)$
21. $\delta(q_7, Y) = (q_7, Y, L)$
22. $\delta(q_7, X) = (q_7, X, L)$
23. $\delta(q_7, C) = (q_8, C, L)$
24. $\delta(q_8, b) = (q_8, b, L)$
25. $\delta(q_8, a) = (q_8, a, L)$
26. $\delta(q_8, Y) = (q_0, Y, R)$
27. $\delta(q_6, C) = (q_9, C, R)$
28. $\delta(q_9, Y) = (q_9, Y, R)$
29. $\delta(q_9, X) = (q_9, X, R)$
30. $\delta(q_9, B) = (q_f, B, R)$

Turing machine

	a	b	x	y	B	C
q0	q_1, X, R	q_5, Y, R	—	—	—	(q_9, C, R)
q1	q_1, a, R	q_1, b, R	—	—	—	q_2, C, R
q2	q_3, X, L	—	q_2, X, R	q_2, Y, R	—	
q3	—	—	q_3, X, L	q_3, Y, L	—	q_4, C, L
q4	q_4, a, L	q_4, b, L	q_0, X, R	—	—	
q5	q_5, a, R	q_5, b, R	—	—	—	q_6, C, R
q6	—	q_7, Y, L	q_6, X, R	q_6, Y, R	.	
q7	—	—	(q_7, X, L)	(q_7, Y, L)	—	(q_8, C, L)
q8	(q_8, a, L)	(q_8, b, L)	—	(q_0, Y, R)	—	
q9	—		(q_9, X, R)	(q_9, Y, R)	(q_f, B, R)	
qf	—	—	—	—	—	

TM as a unary to binary convertor

Decimal	Unary	Binary	
1	1	1	Z
2	11	10	ZY
3	111	11	ZZ
4	1111	100	ZYY
5	11111	101	ZYZ



1 - X

0 - Y
1 - Z
} Binary

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

$q_0 = \{q_0\}$

$F = \{\phi\}$

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

δ

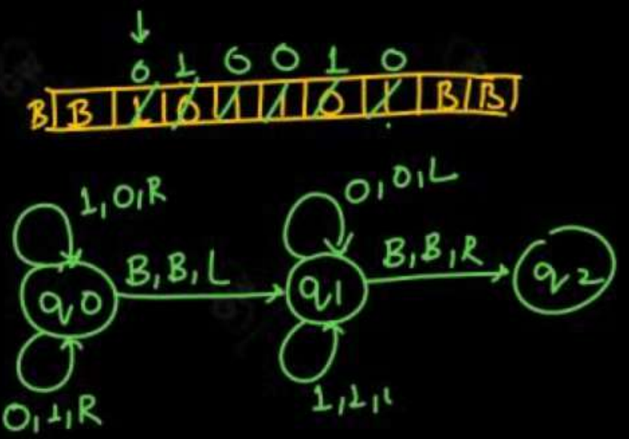
$B = \{B\}$

$\Sigma = \{1\}$

	1	B	X	Y	Z
q_0	(q_1, X, L)	(q_2, B, R)	(q_0, X, R)	(q_0, Y, R)	—
q_1	—	(q_0, Z, R)	(q_1, X, L)	(q_0, Z, R)	(q_1, Y, L)
q_2	—	—	—	—	—

$\delta(q_0, X) = (q_0, X, R)$
$\delta(q_0, Y) = (q_0, Y, R)$
$\delta(q_0, 1) = (q_1, X, L)$
$\delta(q_0, B) = (q_2, B, R)$
$\delta(q_1, Z) = (q_1, Y, L)$
$\delta(q_1, X) = (q_1, X, L)$
$\delta(q_1, B) = (q_0, Z, R)$
$\delta(q_1, Y) = (q_0, Z, R)$

TM for 1s' complement



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

$B = \{B\}$

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

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

$q_0 = \{q_0\}$

$F = \{\}$

δ

	0	1	B
q_0	$(q_0, 0, R)$	$(q_0, 0, R)$	(q_1, B, L)
q_1	$(q_1, 0, L)$	$(q_1, 1, L)$	(q_2, B, R)
q_2	—	—	—

$\delta(q_0, 1) = (q_0, 0, R)$

$\delta(q_0, 0) = (q_0, 1, R)$

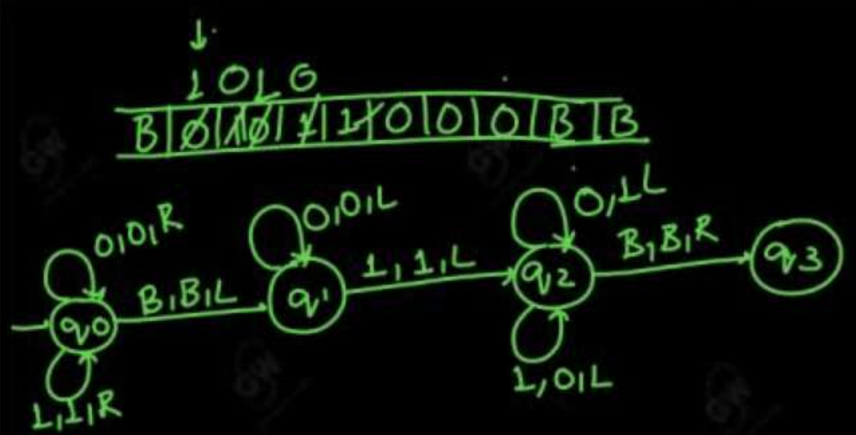
$\delta(q_0, B) = (q_1, B, L)$

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

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

$\delta(q_1, B) = (q_2, B, R)$

TM for 2s' complement



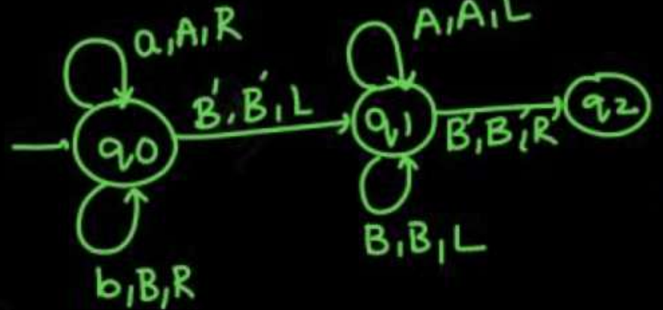
$Q = \{q_0, q_1, q_2, q_3\}$
$q_0 = \{q_0\}$
$F = \{\phi\}$
$\Gamma = \{0, 1, B\}$
$\Sigma = \{0, 1\}$
$B = \{B\}$
δ

	0	1	B
q_0	$(q_0, 0, R)$	$(q_0, 1, R)$	(q_1, B, L)
q_1	$(q_1, 0, L)$	$(q_2, 1, L)$	—
q_2	$(q_2, 1, L)$	$(q_2, 0, L)$	(q_3, B, R)
q_3	—	—	—

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

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

$\{a, b\} \longrightarrow \{A, B\}$ Lower case to upper case conversion



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

$\Sigma = \{a, b\}$ $q_0 = \{q_0\}$ $F = \{\phi\}$

$\Gamma = \{a, b, A, B, B'\}$ $B = \{B'\}$

δ

	a	b	A	B	B'
q_0	q_0, A, R	q_0, B, R	–	–	q_1, B', L
q_1	–	–	q_1, A, L	q_1, B, L	q_2, B', R
q_2	–	–	–	–	–

$\delta(q_0, a) = (q_0, A, R)$

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

$\delta(q_0, B') = (q_1, B', L)$

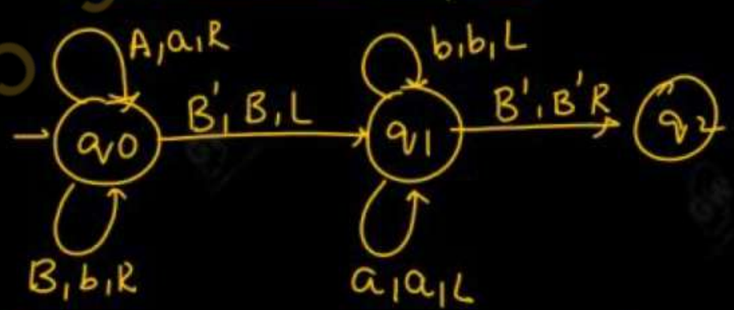
$\delta(q_1, A) = (q_1, A, L)$

$\delta(q_1, B) = (q_1, B, L)$

$\delta(q_1, B') = (q_2, B', R)$

Upper case to lower case

$A, B \longrightarrow a, b$



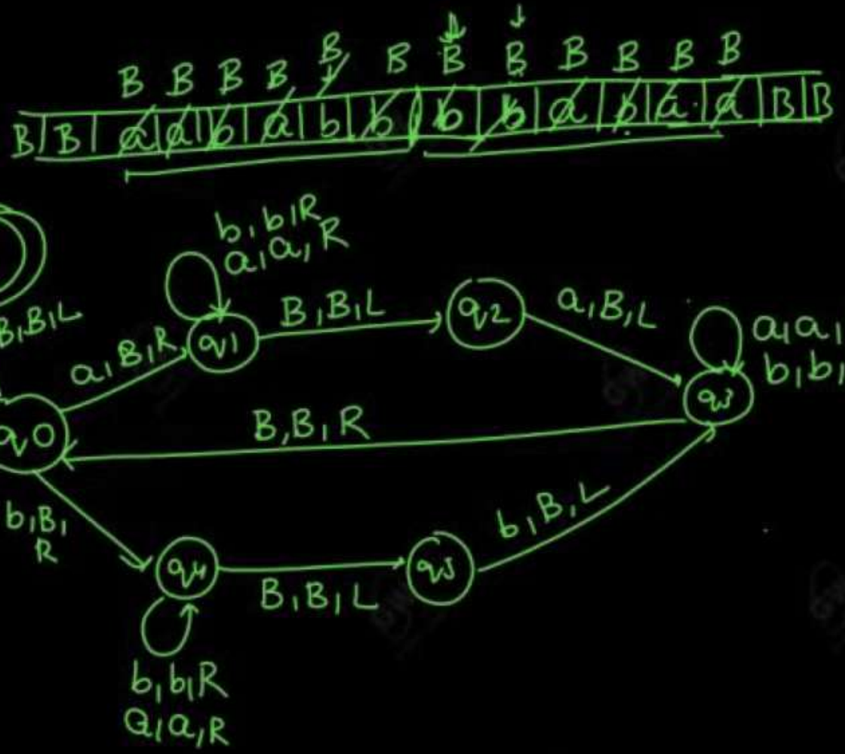
$Q = \{q_0, q_1, q_2\}$ $q_0 = \{q_0\}$

$\Sigma = \{A, B\}$ $F = \{\phi\}$

$\Gamma = \{a, b, A, B, B'\}$

$B = \{B'\}$

Even length palindrome (WW^R) $W \in \{a,b\}^*$ turing machine



	a	b	B
q0	(q1, B, R)	(q4, B, R)	(qf, B, L)
q1	(q1, a, R)	(q1, b, R)	(q2, B, L)
q2	(q3, B, L)	—	—
q3	(q3, a, L)	(q3, b, L)	—
q4	(q4, a, R)	(q4, b, R)	(q5, B, L)
q5		(q3, B, L)	
qf	—	—	—

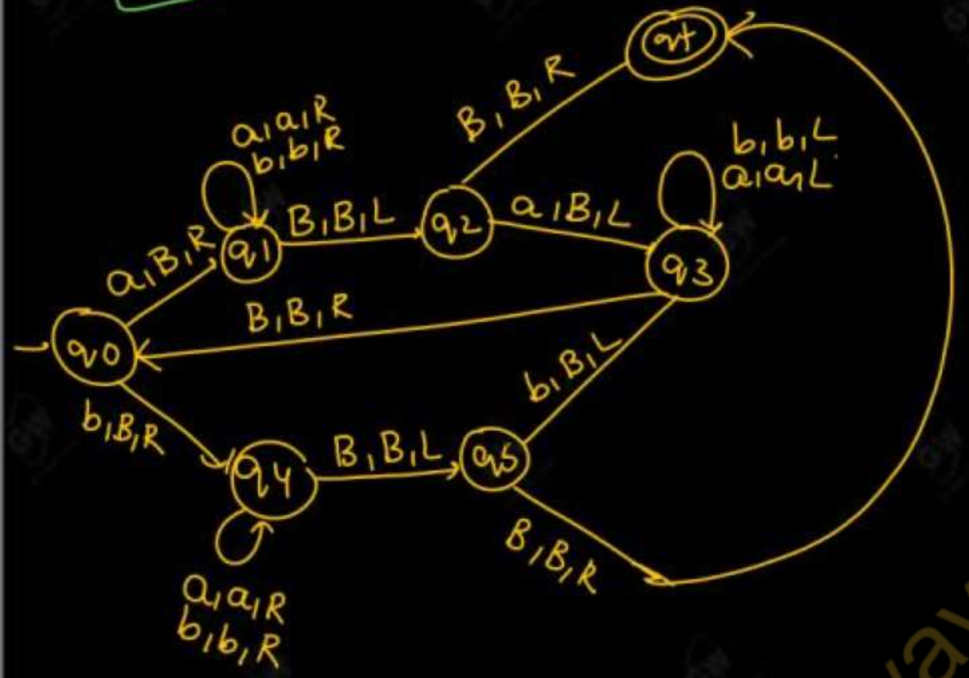
$Q = \{q0, q1, q2, q3, q4, q5, qf\}$
$q0 = \{q0\}$
$F = \{qf\}$
$B = B$
$\Gamma = \{B, a, b\}$
$\Sigma = \{a, b\}$

$\delta(q0, a) = (q1, B, R)$
$\delta(q0, b) = (q4, B, R)$
$\delta(q0, B) = (qf, B, L)$
$\delta(q1, a) = (q1, a, R)$
$\delta(q1, b) = (q1, b, R)$
$\delta(q1, B) = (q2, B, L)$
$\delta(q2, a) = (q3, B, L)$

$\delta(q3, a) = q3, a, L$
$\delta(q3, b) = (q3, b, L)$
$\delta(q4, a) = (q4, a, R$
$\delta(q4, b) = (q4, b, R)$
$\delta(q4, B) = (q5, B, L)$
$\delta(q5, b) = (q3, B, L)$

(Odd length palindrome) Turing machine

waw ✓
wbw ✓
w = {a, b}* ←



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

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

$q_0 = \{q_0\}$

$F = \{q_f\}$

$\Gamma = \{a, b, B\}$

$B = \{B\}$

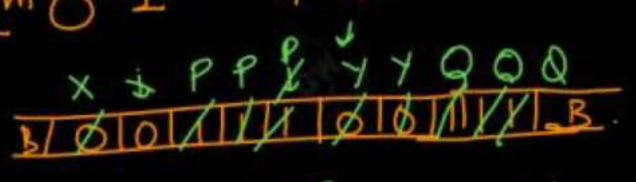
	a	b	B
q_0	(q_1, B, R)	(q_4, B, R)	—
q_1	(q_1, a, R)	(q_1, b, R)	(q_2, B, L)
q_2	(q_3, B, L)	—	q_f, B, R
q_3	(q_3, a, L)	(q_3, b, L)	q_0, B, R
q_4	(q_4, a, R)	(q_4, b, R)	(q_4, B, L)
q_5	—	(q_3, B, L)	(q_f, B, R)
q_f	—	—	—

$\delta(q_0, a) = (q_1, B, R)$
$\delta(q_0, b) = (q_4, B, R)$
$\delta(q_1, a) = (q_1, a, R)$
$\delta(q_1, b) = (q_1, b, R)$
$\delta(q_1, B) = (q_2, B, L)$
$\delta(q_2, B) = (q_f, B, R)$
$\delta(q_2, a)$ (q_3, B, L)

$\delta(q_3, a) = (q_3, a, L)$
$\delta(q_3, b) = (q_3, b, L)$
$\delta(q_3, B) = (q_0, B, R)$
$\delta(q_4, a) = (q_4, a, R)$
$\delta(q_4, b) = (q_4, b, R)$
$\delta(q_4, B) = (q_4, B, L)$

$\delta(q_5, b) =$ (q_3, B, L)
$\delta(q_5, B)$ q_f, B, R

$0^n 1^m 0^n 1^m \quad n, m \geq 1$



$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_f\}$

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

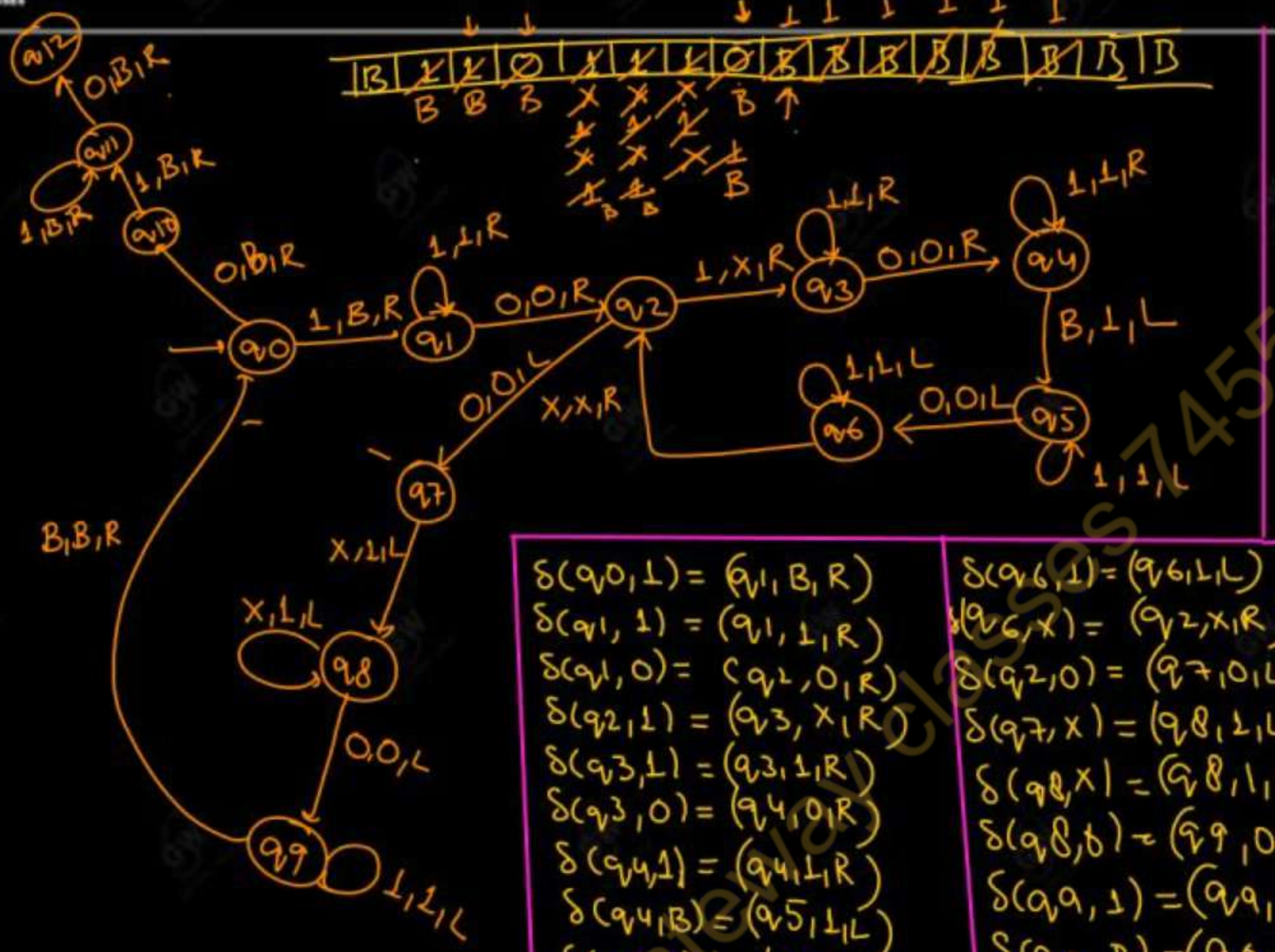
$q_0 = \{q_0\}$

$F = \{q_f\}$

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

$B = \{B\}$

Turing machine for Multiplication



$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}\}$
$q_0 = \{q_0\}$ $F = \{\phi\}$ $\Gamma = \{B, 1, 0, X\}$, δ, $\Sigma = 1$ $\$ = \{B\}$

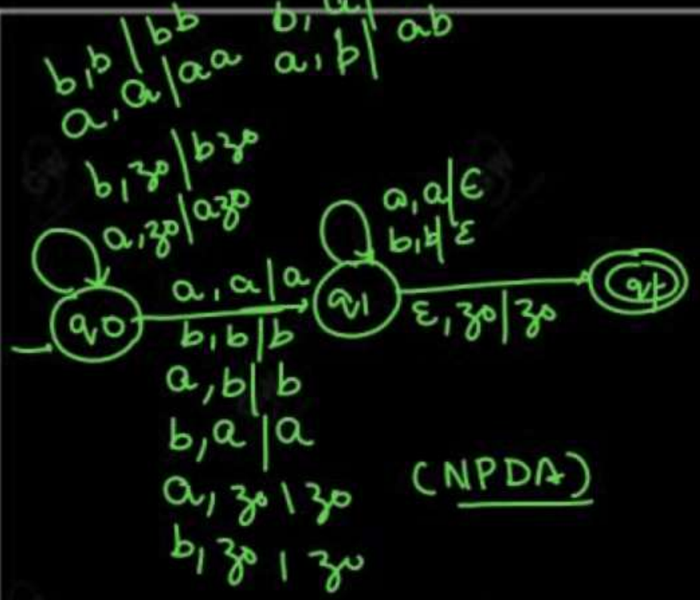
$\delta(q_0, 1) = (q_1, B, R)$
$\delta(q_1, 1) = (q_1, 1, R)$
$\delta(q_1, 0) = (q_2, 0, R)$
$\delta(q_2, 1) = (q_3, X, R)$
$\delta(q_3, 1) = (q_3, 1, R)$
$\delta(q_3, 0) = (q_4, 0, R)$
$\delta(q_4, 1) = (q_4, 1, R)$
$\delta(q_4, B) = (q_5, 1, L)$
$\delta(q_5, 1) = (q_5, 1, L)$
$\delta(q_5, 0) = (q_6, 0, L)$

$\delta(q_6, 1) = (q_6, 1, L)$
$\delta(q_6, X) = (q_2, X, R)$
$\delta(q_2, 0) = (q_7, 0, L)$
$\delta(q_7, X) = (q_8, 1, L)$
$\delta(q_8, X) = (q_8, 1, L)$
$\delta(q_8, B) = (q_9, 0, L)$
$\delta(q_9, 1) = (q_9, 1, L)$
$\delta(q_9, B) = (q_0, B, R)$

$\delta(q_0, 0) = (q_{10}, B, R)$
$\delta(q_{10}, 1) = (q_{11}, B, R)$
$\delta(q_{11}, 1) = (q_{11}, B, R)$
$\delta(q_{11}, 0) = (q_{12}, B, R)$

PDA for odd length palindrome

waw^R ✓ wbw^R ✓ $w \in \{a,b\}^*$

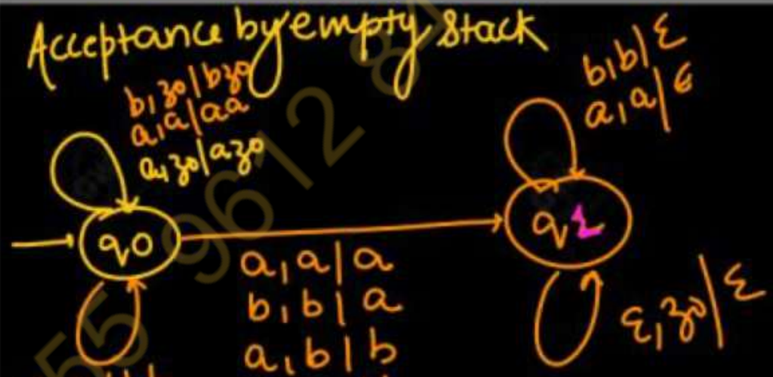


Acceptance by final state

$\delta(q_0, a, z_0) = (q_0, az_0)$ (q_1, z_0) 1
$\delta(q_0, b, z_0) = (q_0, bz_0)$ (q_1, z_0) 2
$\delta(q_0, a, a) = (q_0, aa)$, (q_1, a) 3
$\delta(q_0, b, b) = (q_0, bb)$, (q_1, b) 4
$\delta(q_0, b, a) = (q_0, ba)$, (q_1, a) 5
$\delta(q_0, a, b) = (q_0, ab)$, (q_1, b) 6

$\delta(q_1, a, a) = (q_1, \varepsilon)$ 7
$\delta(q_1, b, b) = (q_1, \varepsilon)$ 8
$\delta(q_1, \varepsilon, z_0) = (q_f, z_0)$ 9

Acceptance by empty stack

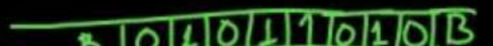


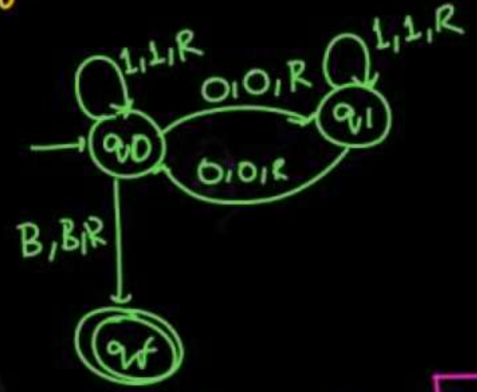
1-8 Same

$\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$

Turing machine for having even zero

Even number of zero



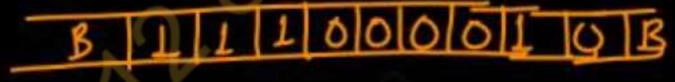


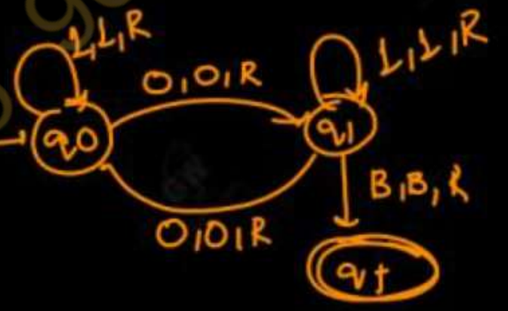
$\delta(q_0, 1) = (q_0, 1, R)$
$\delta(q_0, 0) = (q_1, 0, R)$
$\delta(q_0, B) = (q_f, B, R)$
$\delta(q_1, 1) = (q_1, 1, R)$
$\delta(q_1, 0) = (q_0, 0, R)$

	0	1	B
q_0	$(q_1, 0, R)$	$(q_0, 1, R)$	(q_f, B, R)
q_1		$(q_1, 1, R)$	$(q_0, 0, R)$
q_f	—	—	—

$Q = \{q_0, q_1, q_f\}$
$F = \{q_f\}$ $\Gamma = \{0, 1, B\}$
$q_0 = \{q_0\}$ $\Sigma = \{0, 1\}$
$B = \{B\}$

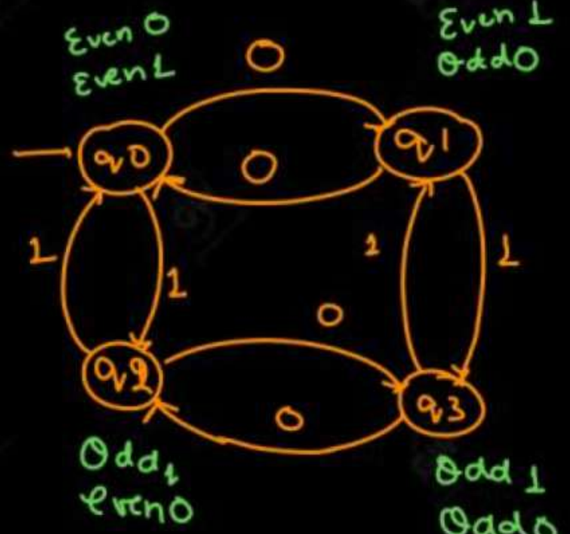
Odd number of zero



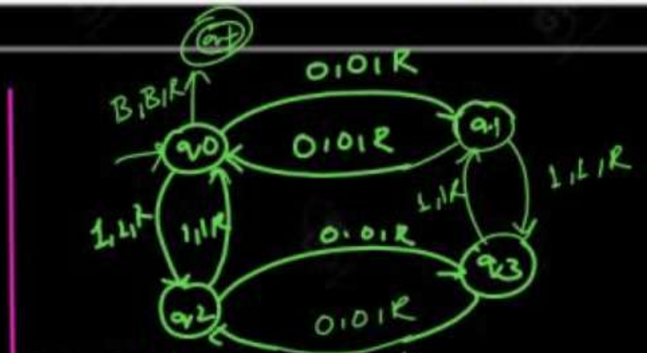


Turing machine for having even zero and even number of 1

(DFA)



$Q = \{q_0, q_1, q_2, q_3\}$
$B = \{B\}$
$\Gamma = \{0, 1, B\}$
$\Sigma = \{0, 1\}$
δ
$F = \{q_f\}$
$q_0 = \{q_0\}$



Even number of zero & 1's

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

Even number of 1's & Odd Zero

1-8 Same
$q\,\delta(q_1,B) = (q_f,B,R)$

Odd number of 1 Even Zero

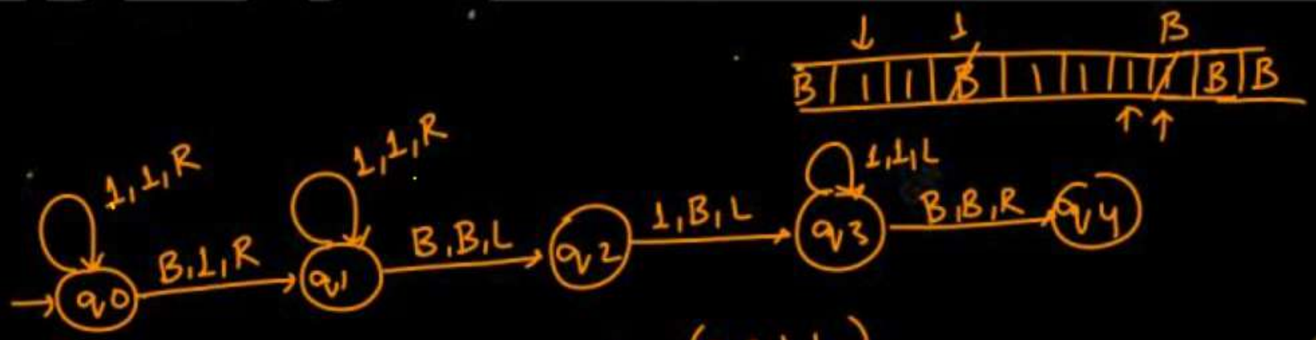
$\delta(q_0,0) = (q_1,0,R)$
$\delta(q_1,0) = (q_0,0,R)$
$\delta(q_0,1) = (q_2,1,R)$
$\delta(q_2,1) = (q_0,1,R)$
$\delta(q_2,0) = (q_3,0,R)$
$\delta(q_3,0) = (q_2,0,R)$
$\delta(q_1,1) = (q_3,1,R)$
$\delta(q_3,1) = (q_1,1,R)$
$\delta(q_2,B) = (q_f,B,R)$

Odd number of Zero & Odd number of 1

1-8 Same
$q\,\delta(q_3,B) = (q_f,B,R)$

	0	1	B
q_0	$q_1,0,R$	$q_2,1,R$	—
q_1	$(q_0,0,R)$	$(q_3,1,R)$	—
q_2	$q_3,0,R)$	$(q_0,1,R)$	—
q_3	$(q_2,0,R)$	$(q_1,1,R)$	(q_f,B,R)
q_f	—	—	—

TM as a adder



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

$\delta(q_3,1) = (q_3,1,L)$
$\delta(q_3,B) = (q_4,B,R)$

$Q = \{q_0, q_1, q_3, q_3, q_4\}$
$q_0 = \{q_0\}$
$\Sigma = \{1\}$
$F = \{\phi\}$
$B = \{B\}$
$\Gamma = \{1, B\}$
δ

TM < 2002-08, 2016-12

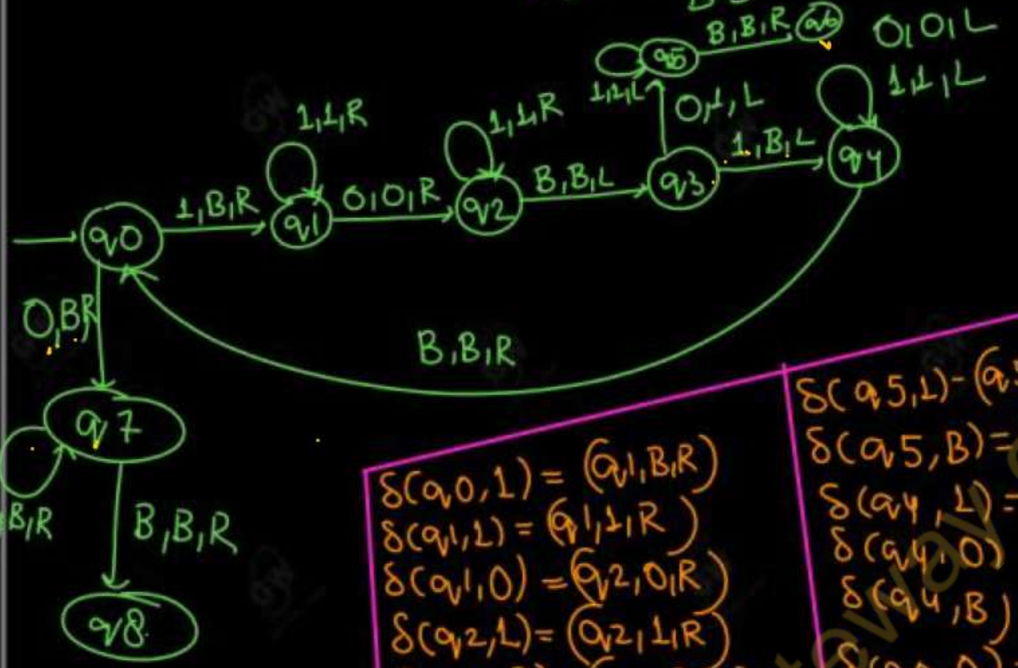
$(x,y) = x+y$

	1	B
q_0	$(q_0,1,R)$	$(q_1,1,R)$
q_1	$(q_1,1,R)$	(q_2,B,L)
q_2	(q_3,B,L)	—
q_3	$(q_3,1,L)$	(q_4,B,R)
q_4	—	—

Turing machine for subtraction

$$f(m,n) = \begin{cases} m-n & \text{if } (m>n) \\ 0 & m \le n \end{cases}$$

q_6 $m>n$
q_7 $m=n$
q_8 $m<n$



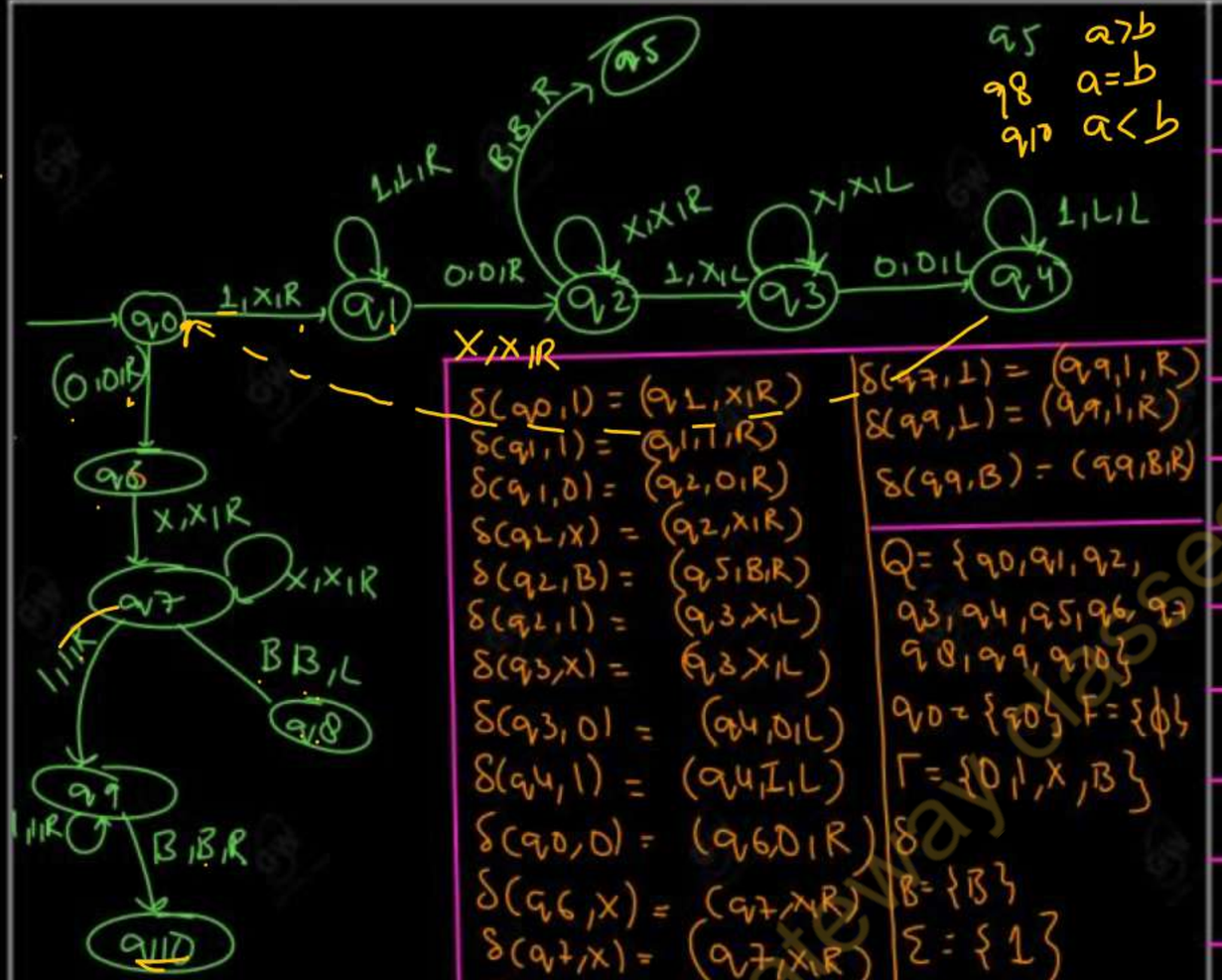
$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$
$q_0 = \{q_0\}$ $F = \{\phi\}$
$\Gamma = \{1, 0, B\}$
$B = \{B\}$
$\Sigma = \{1\}$
δ

$\delta(q_0,1) = (q_1,B,R)$
$\delta(q_1,1) = (q_1,1,R)$
$\delta(q_1,0) = (q_2,0,R)$
$\delta(q_2,1) = (q_2,1,R)$
$\delta(q_2,B) = (q_3,B,L)$
$\delta(q_3,1) = (q_4,B,L)$
$\delta(q_3,0) = (q_5,1,L)$

$\delta(q_5,1) = (q_5,1,L)$
$\delta(q_5,B) = (q_6,B,R)$
$\delta(q_4,1) = (q_4,1,L)$
$\delta(q_4,0) = (q_4,0,L)$
$\delta(q_4,B) = (q_0,B,R)$
$\delta(q_0,0) = (q_7,B,R)$
$\delta(q_7,1) = (q_7,B,R)$
$\delta(q_7,B) = (q_8,B,R)$

	1	B	0
q_0	(q_1,B,R)	—	(q_7,B,R)
q_1	$(q_1,1,R)$	—	$(q_2,0,R)$
q_2	—	(q_3,B,L)	—
q_3	(q_4,B,L)	—	$(q_5,1,L)$
q_4	$(q_4,1,L)$	(q_0,B,R)	$(q_4,0,L)$
q_5	$(q_5,1,L)$	(q_6,B,R)	
q_6	—	—	—
q_7	(q_7,B,R)	(q_8,B,R)	—
q_8	—	—	—

Turing machine as comparator



X, X, R

$\delta(q_0,1) = (q_1, X, R)$
$\delta(q_1,1) = (q_1,1,R)$
$\delta(q_1,0) = (q_2,0,R)$
$\delta(q_2,X) = (q_2,X,R)$
$\delta(q_2,B) = (q_5,B,R)$
$\delta(q_2,1) = (q_3,X,L)$
$\delta(q_3,X) = (q_3,X,L)$
$\delta(q_3,0) = (q_4,0,L)$
$\delta(q_4,1) = (q_4,1,L)$
$\delta(q_4,0) = (q_6,0,R)$
$\delta(q_6,X) = (q_7,X,R)$
$\delta(q_7,X) = (q_7,X,R)$
$\delta(q_7,B) = (q_8,B,L)$

$\delta(q_7,1) = (q_9,1,R)$
$\delta(q_9,1) = (q_9,1,R)$
$\delta(q_9,B) = (q_9,B,R)$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}\}$
$q_0 = \{q_0\}$ $F = \{\phi\}$
$\Gamma = \{0, 1, X, B\}$
δ
$B = \{B\}$
$\Sigma = \{1\}$
$\delta(q_4, X) = (q_0, X, R)$

	1	X	0	B
q_0	(q_1, X, R)	—	$(q_6, 0, R)$	—
q_1	$(q_1, 1, R)$	—	$(q_2, 0, R)$	—
q_2	(q_3, X, L)	(q_2, X, R)	—	(q_5, B, R)
q_3	—	(q_3, X, L)	$(q_4, 0, L)$	—
q_4	$(q_4, 1, L)$	(q_0, X, R)	—	—
q_5	—	—	—	—
q_6	—	(q_7, X, R)	—	—
q_7	$(q_9, 1, R)$	(q_7, X, R)	—	(q_8, B, L)
q_8	—	—	—	—
q_9	$(q_9, 1, R)$	—	—	(q_A, B, R)
q_{10}		—	—	—

Variants of Turing machine

1. Multiple track Turing Machine:

- A k-track Turing machine(for some k>0) has k-tracks and one R/W head that reads and writes all of them one by one.
- A k-track Turing Machine can be simulated by a single track Turing machine

2. Two-way infinite Tape Turing Machine:

- Infinite tape of two-way infinite tape Turing machine is unbounded in both directions left and right

Two-way infinite tape Turing machine can be simulated by one-way infinite Turing machine(standard Turing machine).

3. Multi-tape Turing Machine:

- It has multiple tapes and is controlled by a single head.
- The Multi-tape Turing machine is different from k-track Turing machine but expressive power is the same.
- Multi-tape Turing machine can be simulated by single-tape Turing machine.

Variants of Turing machine

4. Multi-tape Multi-head Turing Machine:

- The multi-tape multi-head Turing machine has multiple tapes and multiple heads
- Each tape is controlled by a separate head
- Multi-Tape Multi-head Turing machine can be simulated by a standard Turing machine.

5. Multi-dimensional Tape Turing Machine:

It has multi-dimensional tape where the head can move in any direction that is left, right, up or down.

Multi dimensional tape Turing machine can be simulated by one-dimensional Turing machine

6. Multi-head Turing Machine:

- A multi-head Turing machine contains two or more heads to read the symbols on the same tape.
- In one step all the heads sense the scanned symbols and move or write independently.
- Multi-head Turing machine can be simulated by a single head Turing machine

Variants of Turing machine

7. Non-deterministic Turing Machine:

- A non-deterministic Turing machine has a single, one-way infinite tape.
- For a given state and input symbol has at least one choice to move (finite number of choices for the next move), each choice has several choices of the path that it might follow for a given input string.
- A non-deterministic Turing machine is equivalent to the deterministic Turing machine.
- $Q \times \tau \dashrightarrow 2^{Q \times \tau \times \{Left\_shift, Right\_shift, \}}$.

8 Turing Machine with Stay option:

- If instead of moving left or right on seeing an input, the head could also stay at one position without moving anywhere i.e.
- $f: Q \times \tau \dashrightarrow Q \times \tau \times \{Left\_shift, Right\_shift, Stay\}$.Still the number of languages accepted by turing machine remains same .

Universal Turing machine

Universal Turing Machine

- It simulates a Turing Machine. Universal Turing Machine can be considered as a subset of all the Turing machines, it can match or surpass other Turing machines including itself.
- Universal Turing Machine is like a single Turing Machine that has a solution to all problems that is computable.
- It contains a Turing Machine description as input along with an input string, runs the Turing Machine on the input and returns a result.

- A Universal Turing Machine, in more specific terms, can imitate the behavior of an arbitrary Turing machine over any collection of input symbols. Therefore, it is possible to create a single machine to calculate any computable sequence.
- The input of a UTM includes:
  - A basic description of the machine
  - The contents of the machine tape
  - The internal state of the machine

Construction of UTM

we assume the following for M:

Q = {q1, q2, ...qn} where q1=initial state and q5=Final State

$q_1 \to 1 \quad \sigma_1 - 1 \quad L - 1$
$q_2 - 11 \quad \sigma_2 - 11 \quad R - 11$
$q_3 - 111 \quad \sigma_3 - 111$

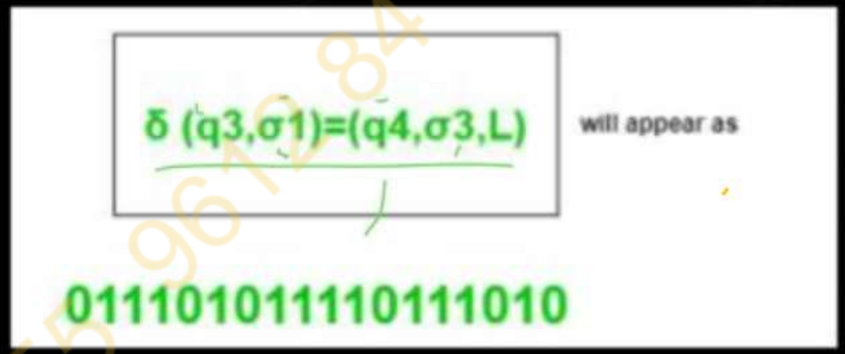
τ = {σ1, σ2,,...σn}

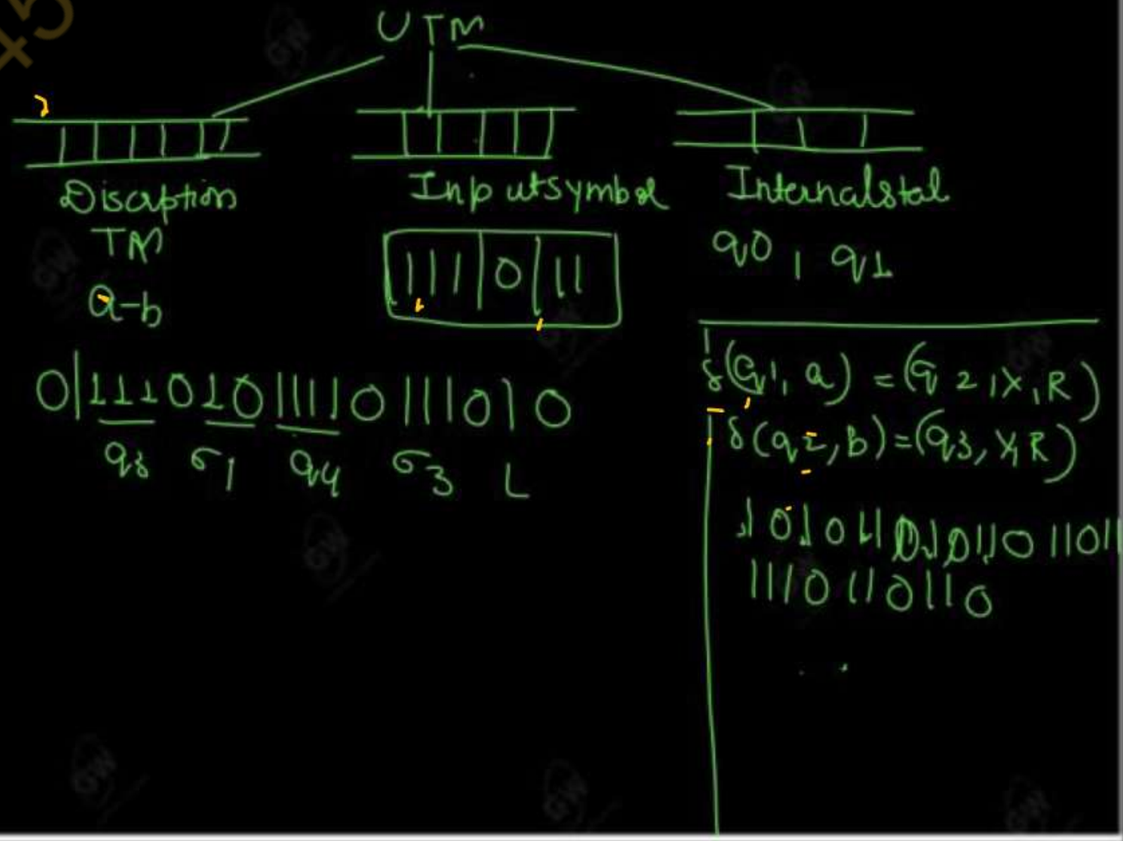
Select an encoding on which q1 is representable by 1, q2 by 11, and so on.

Similarly, σ1 is encoded as 1, σ2 as 11, etc.

Finally, let us represent R/W head directions by 1 for L (Left) and 11 for R(Right).

The symbol 0 will be used as a separator between 1s.





HOW UNIVERSL TURING MACHINE CAN BE CONSIDERED AS MODEL OF DIGITAL COMPUTER

- A Universal Turing Machine (UTM) can be considered a model of a digital computer because it captures the essential principles and capabilities of what a digital computer can do. Here are the key reasons why a UTM serves as a model for digital computers:

1. Universality:

- A UTM can simulate any other Turing machine, just as a digital computer can run any program.
- This means that a UTM can perform any computation that any other Turing machine (or digital computer) can perform, given enough time and memory.

2. Stored Program Concept:

- A UTM has a tape that contains both the instructions (program) and the data. This is similar to how modern digital computers store programs and data in memory.
- The ability to modify the instructions based on data is fundamental to both UTMs and digital computers.

HOW UNIVERSL TURING MACHINE CAN BE CONSIDERED AS MODEL OF DIGITAL COMPUTER

3.Basic Operations:

- The operations of a UTM (reading, writing, moving the tape, and changing states) are analogous to the basic operations of a digital computer (reading from memory, writing to memory, executing instructions, and changing the state of the processor).

4. Sequential Processing: Both a UTM and a digital computer process information sequentially, executing one instruction at a time. This sequential execution is a core characteristic of traditional digital computers

5.Finite Control:

- The UTM has a finite set of states and a transition function, similar to how a digital computer has a finite set of instructions and a control unit that determines the sequence of operations.

6. Computational Completeness:

- The Church-Turing thesis suggests that any function that can be computed by an algorithm can be computed by a Turing machine. Since digital computers are designed to execute algorithms, they are effectively equivalent in computational power to Turing machines.

HOW UNIVERSL TURING MACHINE CAN BE CONSIDERED AS MODEL OF DIGITAL COMPUTER

7. Input and Output:

- UTMs and digital computers both take inputs, process them according to a set of instructions (program), and produce outputs.
- The way a UTM manipulates symbols on a tape parallels how a digital computer manipulates bits in memory.

8.Abstract Model:

- The UTM provides a simplified, abstract model of computation that helps in understanding the theoretical limits of what can be computed.
- This abstraction is useful for analyzing the fundamental capabilities and limitations of digital computers.

Halting Problem

- The Halting Problem is a fundamental concept in computer science and the theory of computation. It involves determining, given a description of an arbitrary computer program and an input, whether the program will eventually halt (i.e., stop executing) or continue to run forever.
- Alan Turing first formulated the problem in 1936, proving that there is no general algorithm that can solve the Halting Problem for all possible program-input pairs. This means that it is impossible to write a program that can determine for every other program and input whether the given program will halt.
- This is an undecidable problem because we cannot have an algorithm which will tell us whether a given program will halt or not in a generalized way i.e by having specific program/algorithm.In general we can't always know that's why we can't have a general algorithm.The best possible way is to run the program and see whether it halts or not.In this way for many programs we can see that it will sometimes loop and always halt.

Halting Problem is undecidable

➢ **Proof by Contradiction –**

Problem statement: Can we design a machine which if given a program can find out if that program will always halt or not halt on a particular input?

Solution: Let us assume that we can design that kind of machine called as HM(P, I) where HM is the machine/program, P is the program and I is the input. On taking input the both arguments the machine HM will tell that the program P either halts or not.

If we can design such a program this allows us to write another program we call this program CM(X) where X is any program(taken as argument) and according to the definition of the program CM(X) shown in the figure.

Halting Problem is undecidable

```
HM ( P,I)
{  Halt

or
May not Halt
}
```

```
CM ( X)
{
   if(HM(X,X)==Halt)
         loop forever;
   else
         return;

}
```

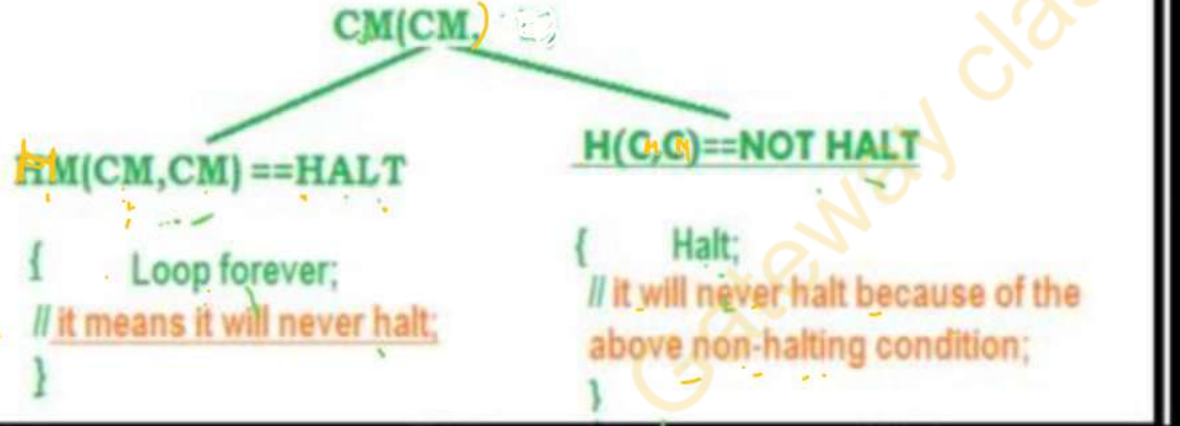
- In the program CM(X) we call the function HM(X), which we have already defined and to HM() we pass the arguments (X, X), according to the definition of HM() it can take two arguments i.e one is program and another is the input.Now in the second program we pass X as a program and X as input to the function HM().We know that the program HM() gives two output either "Halt" or "Not Halt".But in case second program, when HM(X, X) will halt loop body tells to go in loop and when it doesn't halt that means loop, it is asked to return.

Halting Problem is undecidable

```
HM ( P,I)
{  Halt
or
May not Halt
}
```

```
CM ( X)
{
  if ( H ( X,X)  ==HALT)
       loop forever;
  else
       return;
}
```

if we run CM on itself:




```
CM(CM)
{
  if(CM,CM)==Halt)
      loop forever.
else
  return
}
```

→ That why we can. Turing machine halting problem is undecidable

CHRUCH THESIS

- Church's Thesis, also known as the Church-Turing Thesis, is a basic idea in computer science. It says that if you can solve a problem using any method or machine, you can also solve it using a Turing machine, which is a simple mathematical model of a computer.

you can make a Algo.

- Some key points

1.Computing Anything: Imagine you have a problem to solve. If you can write a set of instructions (an algorithm) that a person or any machine can follow to solve the problem, Church's Thesis says that you can also solve this problem using a Turing machine.

2All Methods are Equivalent: There are many ways to describe how to solve problems, like using different programming languages or mathematical systems. Church's Thesis says that all these ways are essentially the same when it comes to what problems they can solve. If one method can solve a problem, they all can.

CHRUCH THESIS

GW GATEWAY CLASSES

3. No Formal Proof: You can't prove Church's Thesis like a math theorem. It's more like a belief based on lots of evidence. Every method we know for solving problems can be simulated by a Turing machine.

4. Limits of Computation: This idea helps us understand the limits of what can be computed. If a problem can't be solved by a Turing machine, it can't be solved by any algorithm or computer, no matter how powerful.

.

(Myhill nerode theorem)

Application

- Test whether a given language is regular or not
- For minimization of DFA

Let L be any language over Σ*

Consider two string X,Y(X,Y ∈ Σ*)are said to be in Class if all possible string(every string)Z (Z∈ Σ*)

Either XY and YZ are in L or both are not

1. A language L divides the set of all possible string into mutual exclusive classes
2. If L is regular , the number of classes created by L is finite
3. if number of classes I is create is finite then L is regular

L={ String ends with 0} Σ={0,1}

$\Sigma = \{0, 01, 110, 111, 1111, 0011, \ldots\}$

L_1 = {ends with zero} Class1

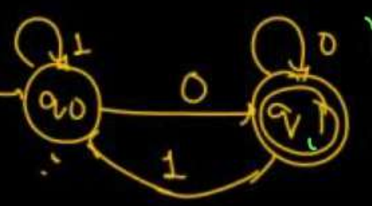
$L_1 = \{0, 10, 0000, \ldots\}$

L_2 = {ends with 1} Class2

$L_2 = \{1, 01, 1111, 10001, \ldots\}$

$X = 0 \in \Sigma^*$
$Y = 11 \in \Sigma^*$
$Z = 000 \in \Sigma^*$

XZ = 0000 belong L_1
YZ = 11000 ∉ belong L_2



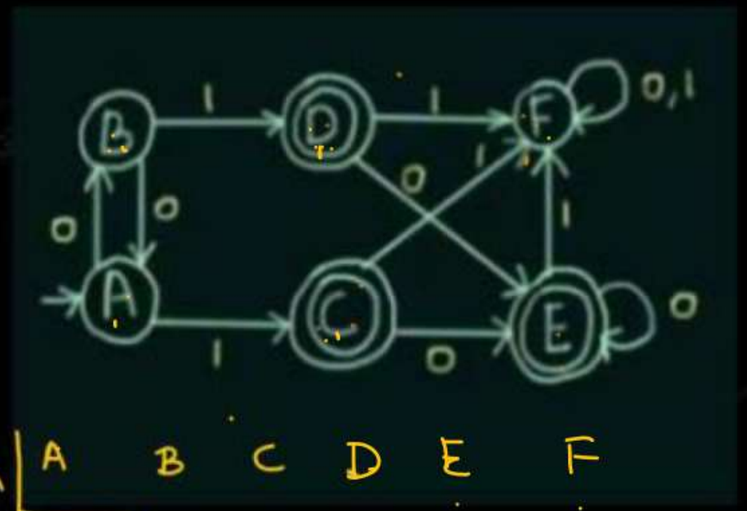
L is Regular

Minimization of DFA using **Myhill nerode theorem**

steps for minimization of DFA

1. Draw a table for all pairs of states(P,Q)
2. Marks all pairs where p$\in$ F and Q $\notin$ F
3. If there are any unmarked pairs (P,Q) such thatδ (P,X) and δ (Q,X) is marked then mark [P,Q] where X is an input symbol Repeat this until no more marking can be made
4. Combine all the unmarked pairs and make them a single state in minimized DFA

Minimization of DFA using **Myhill nerode theorem**



	A	B	C	D	E	F
B						
C	✓	✓				
D	✓	✓				
E	✓	✓				
F	✓	✓	✓	✓	✓	

(A,B)
$\delta(A,0) = B$
$\delta(B,0) = A$
$\delta(A,1) = C$
$\delta(B,1) = D$

(C,D)
$\delta(C,0) = E$
$\delta(D,0) = E$
$\delta(C,1) = F$
$\delta(D,1) = F$

(C,E)
$\delta(C,0) = E$
$\delta(E,0) = E$
$\delta(C,1) = F$
$\delta(E,1) = F$

(D,E)
$\delta(D,0) = E$
$\delta(E,0) = E$
$\delta(D,1) = F$
$\delta(E,1) = F$

(A,F)
$\delta(A,0) = B$
$\delta(F,0) = F$
$\delta(A,1) = C$
$\delta(F,1) = F$

$\delta(B,F)$
$\delta(B,0) = A$
$\delta(F,0) = F$
$\delta(B,1) = D$
$\delta(F,1) = F$

=

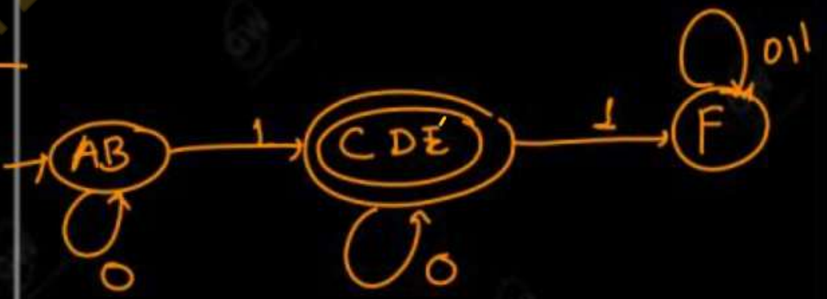
(A,B)
$\delta(A,0) = B$ $\quad \delta(A,1) = C$
$\delta(B,0) = A$ $\quad \delta(D,1) = D$

(C,D)
$\delta(C,0) = E$ $\quad \delta(C,1) = F$
$\delta(D,0) = E$ $\quad \delta(D,1) = F$

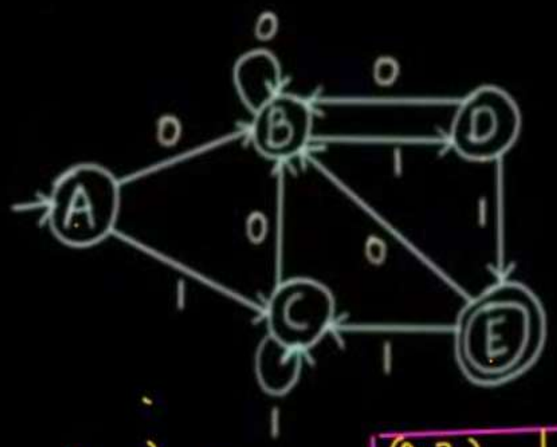
(C,E)
$\delta(C,0) = E$ $\quad \delta(C,1) = F$
$\delta(E,0) = E$ $\quad \delta(E,1) = F$

(D,E)
$\delta(D,0) = E$ $\quad \delta(D,1) = F$
$\delta(E,0) = E$ $\quad \delta(E,1) = F$

(A,B) (C,D) (C,E), (D,E)



Minimization of DFA using **Myhill nerode theorem**



	A	B	C	D	E
A					
B	✓				
C		✓			
D	✓	✓	✓		
E	✓	✓	✓	✓	

(A, B)
$\delta(A,0) = B$ $\quad \delta(A,1) = C$
$\delta(B,0) = B$ $\quad \delta(B,1) = D$

(A, C)
$\delta(A,0) = B$ $\quad \delta(A,1) = C$
$\delta(C,0) = B$ $\quad \delta(C,1) = C$

(A, D)
$\delta(A,0) = B$
$\delta(D,0) = B$
$\delta(A,1) = C$
$\delta(D,1) = E$

(B, C)
$\delta(B,0) = B$
$\delta(C,0) = B$
$\delta(B,1) = D$
$\delta(C,1) = C$

(B, D)
$\delta(B,0) = B$
$\delta(D,0) = B$
$\delta(B,1) = D$
$\delta(D,1) = E$

(C, D)
$\delta(C,0) = B$
$\delta(D,0) = B$
$\delta(C,1) = C$
$\delta(D,1) = E$

II

(A, B)
$\delta(A,0) = B$
$\delta(B,0) = B$
$\delta(A,1) = C$
$\delta(B,1) = D$

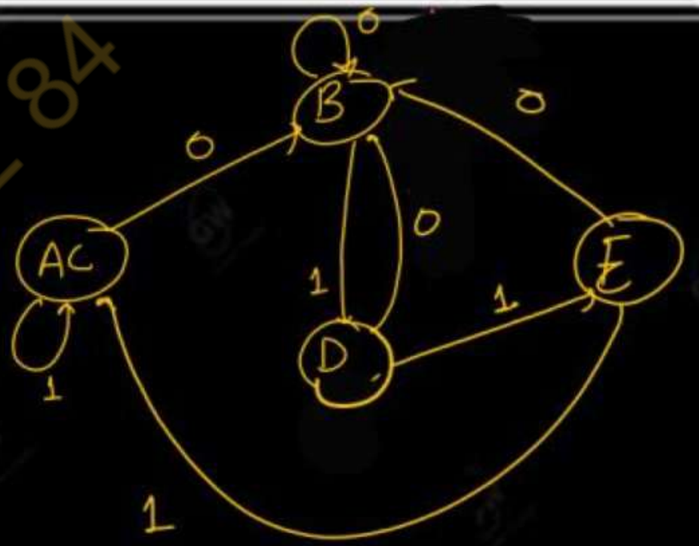
(A, C)
$\delta(A,0) = B$
$\delta(C,0) = B$
$\delta(A,1) = C$
$\delta(C,1) = C$

(B, C)
$\delta(B,0) = B$
$\delta(C,0) = B$
$\delta(B,1) = D$
$\delta(C,1) = C$

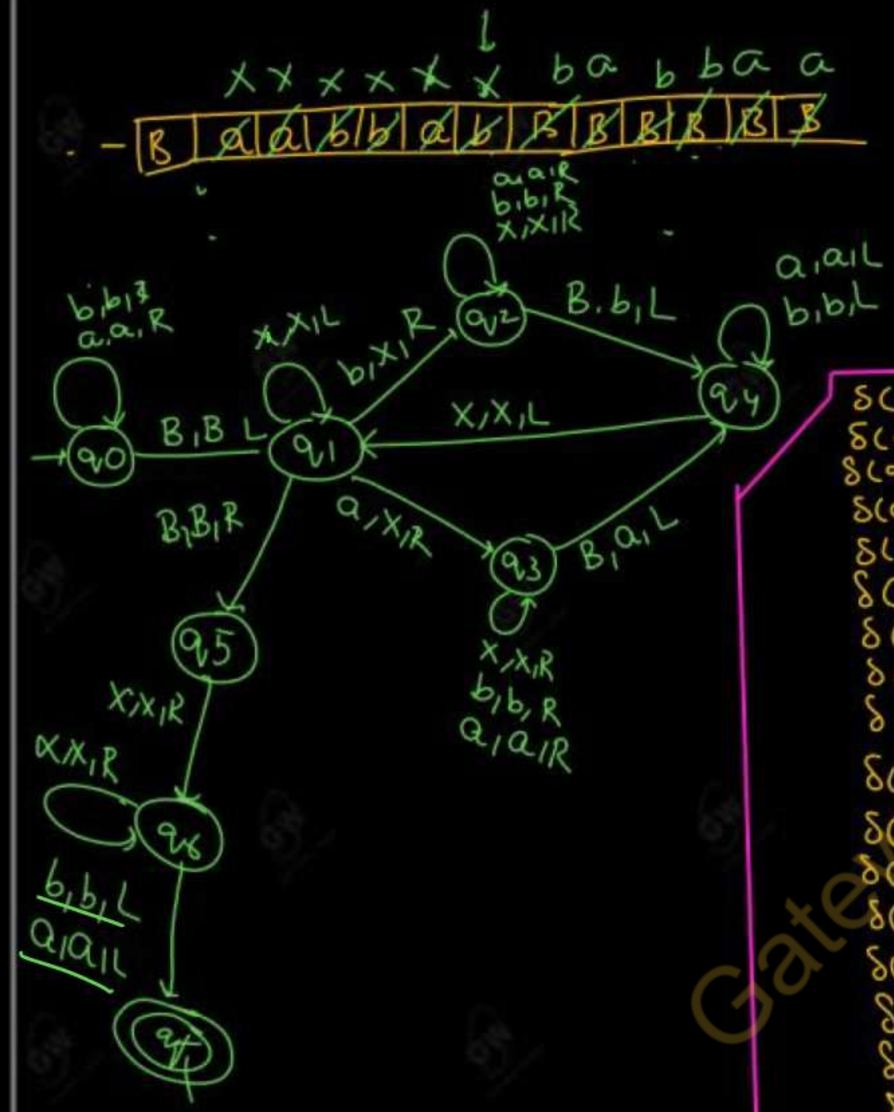
III this

(A C)
$\delta(A,0) = B$
$\delta(C,0) = B$
$\delta(A,1) = C$
$\delta(C,1) = C$

(AC), B, D, E



Turing machine to reverse a string $\Sigma = \{a, b\}$



$\delta(q_1, B) = (q_5, B, R)$
$\delta(q_5, X) = q_6, X, R)$
$\delta(q_6, X) = (q_6, X, R)$
$\delta(q_6, b) = q_f, b, L)$
$\delta(q_6, a)$ $q_f, a, L)$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_f\}$
$T = \{a, b, B, X\}$ $\Sigma = \{a, b\}$
$q_0 = \{q_0\}$ $F = \{q_f\}$ $B = \{B\}$

$\delta(q_0, a) = (q_0, a, R)$
$\delta(q_0, b) = (q_0, b, R)$
$\delta(q_0, B) = (q_1, B, L)$
$\delta(q_1, X) = (q_1, X, L)$
$\delta(q_1, b) = (q_2, X, R)$
$\delta(q_2, a) = (q_2, a, R)$
$\delta(q_2, b) = (q_2, b, R)$
$\delta(q_2, X) = (q_2, X, R)$
$\delta(q_2, B) = (q_4, b, L)$
$\delta(q_1, a) = (q_3, X, R)$
$\delta(q_3, X) = (q_3, X, R)$
$\delta(q_3, b) = (q_3, b, R)$
$\delta(q_3, a) = (q_3, a, R)$
$\delta(q_3, B) = (q_4, a, L)$
$\delta(q_4, a) = (q_4, a, L)$
$\delta(q_4, b) = (q_4, b, L)$
$\delta(q_1, X) = (q_1, X, L)$

	a	b	B	X
q_0	(q_0, a, R)	(q_0, b, R)	$q_1, B, L)$	
q_1	$q_3, X, R)$	$(q_2 X, R)$	$(q_5 B, R)$	(q_1, X, L)
q_2	(q_2, a, R)	(q_2, b, R)	(q_4, b, L)	(q_2, X, R)
q_3	(q_3, a, R)	(q_3, b, R)	(q_4, a, L)	(q_3, X, R)
q_4	(q_4, a, L)	(q_4, b, L)		(q_1, X, L)
q_5				(q_6, X, R)
q_6	$q_f, a, L)$	(q_f, b, L)		
q_f				

Linear bound automata

A linear bounded automaton can be defined as an 8-tuple (Q, τ, ∑, q_0, ML, MR, δ, F) where –

Q is a finite set of states

τ is the tape alphabet { ⌋, ⌊, a, b }

∑ is the input alphabet { a, b }

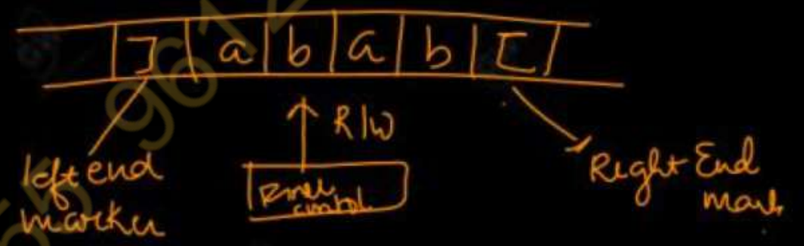
$\mathbf{q_0}$ is the initial state

$\mathbf{M_L}$ is the left end marker { ⌋ }

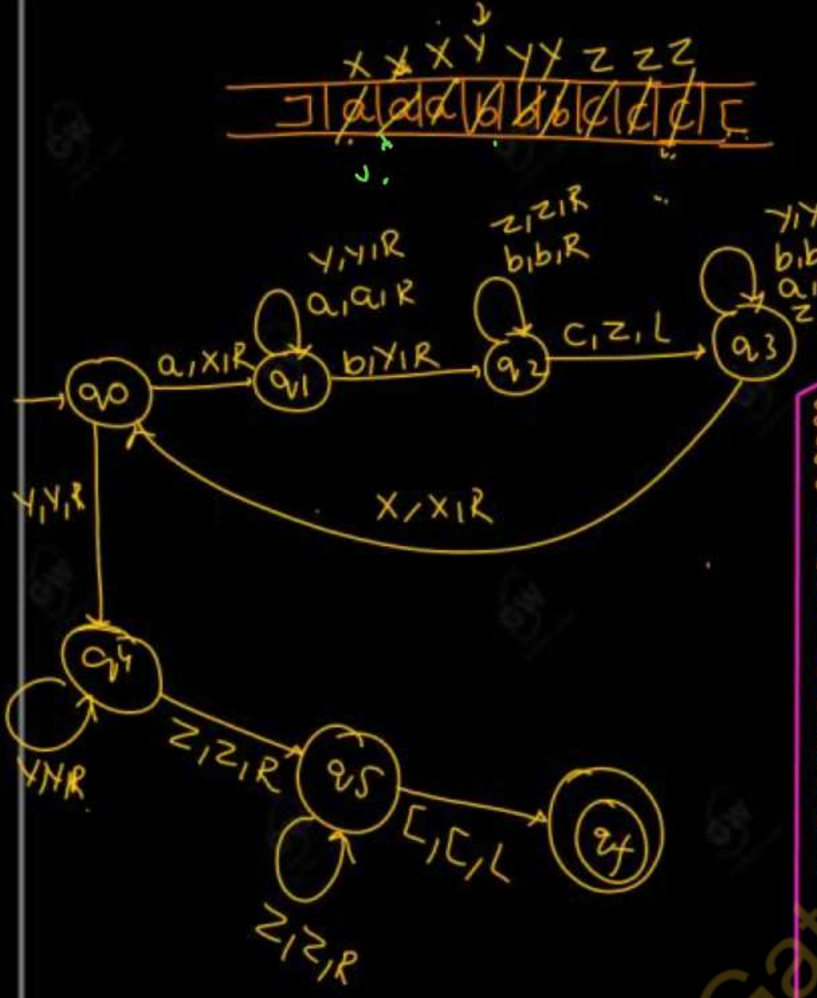
$\mathbf{M_R}$ is the right end marker where $M_R \neq M_L$ { ⌊ }

δ is a transition function Q x τ -> Q x τ x{ leftshift , rightshift }

F is the set of final states



Linear bound automata $a^n b^n c^n$ $n \geq 1$



$\delta(q_0, a) = (q_1, X, R)$
$\delta(q_1, a) = (q_1, a, R)$
$\delta(q_1, Y) = (q_1, Y, R)$
$\delta(q_1, b) = (q_2, Y, R)$
$\delta(q_2, Z) = (q_2, Z, R)$
$\delta(q_2, b) = (q_2, b, R)$
$\delta(q_2, c) = (q_3, Z, L)$
$\delta(q_3, Y) = (q_3, Y, L)$
$\delta(q_3, b) = (q_3, b, L)$
$\delta(q_3, a) = (q_3, a, L)$
$\delta(q_3, X) = (q_0, X, R)$
$\delta(q_0, Y) = (q_4, Y, R)$

$\delta(q_4, Y) = (q_4, Y, R)$ $\quad \delta(q_4, Z) = (q_5, Z, R)$
$\delta(q_5, Z) = (q_5, Z, R)$ $\quad \delta(q_5, \sqsubset) = (q_f, \sqsubset, L)$

	a	b	c	X	Y	Z	⊐	⊏
q_0	(q_1, X, R)	—	—	—	(q_4, Y, R)	—	—	—
q_1	(q_1, a, R)	(q_2, Y, R)	—	—	(q_1, Y, R)	(q_2, Z, R)	—	—
q_2	—	(q_2, b, R)	(q_3, Z, L)	—	—	—	—	—
q_3	(q_3, 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)	—	$(q_f, \sqsubset, L)$
q_f	—	—	—	—	—	—	—	—

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_f\}$ $\quad q_0 = \{q_0\}$
$q_f = \{q_f\}$ $\quad \Gamma = \{\sqsupset, \sqsubset, a, b, c, X, Y, Z\}$
$\Sigma = \{a, b, c\}$ $\quad \delta$
$M_L = \sqsupset$ $\quad M_R = \sqsubset$

Recursive Languages(REC)

- A language is recursive (also called a decidable language) if there is a Turing machine that will always halt and decide whether a given string belongs to the language. In other words, there exists an algorithm that can determine membership in the language in a finite amount of time. Formally, a language L is recursive if there is a Turing machine M such that:
- For every string w in L, M accepts w and halts.
- For every string w not in L, M rejects w and halt

Recursively Enumerable Languages(RE)

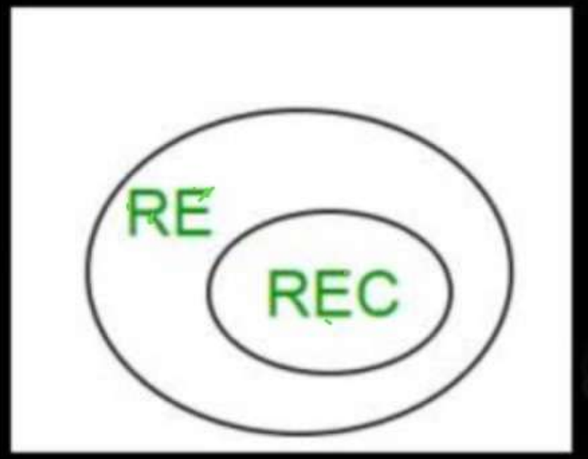
- A language is recursively enumerable (also called a Turing-recognizable language) if there is a Turing machine that will recognize whether a given string belongs to the language. The difference from recursive languages is that the Turing machine might not halt for strings that do not belong to the language. Formally, a language L is recursively enumerable if there is a Turing machine M such that:
- For every string www in L, M accepts w and halts.
- For every string w not in L, M either rejects w and halts, or runs forever without halting

Key Differences

- **Halting:** For recursive languages, the Turing machine always halts, whereas for recursively enumerable languages, the Turing machine might run indefinitely for some strings not in the language.
- **Decidability:** Recursive languages are decidable (there's an algorithm that decides membership), while recursively enumerable languages are not necessarily decidable (there's an algorithm that recognizes membership but might not decide it)

Relationships

Every recursive language is also recursively enumerable, but not every recursively enumerable language is recursive. This means that the set of recursive languages is a subset of the set of recursively enumerable languages



Closure Properties of Recursive Languages

Union: If L1 and **If L2 are two recursive languages, their union L1U L2 will also be recursive because if TM halts for L1 and halts for L2, it will also halt for L1UL2.**

L1= $\{a^nb^nc^n | n>=1\}$ L2= $\{d^me^mf^m | m>=1\}$ L3= L1UL2 = is also recursive.

Concatenation: If L1 and If L2 are two recursive languages, their concatenation L1.L2 will also be recursive. For Example:

L1= $\{a^nb^nc^n | n>=0\}$ L2= $\{d^me^mf^m | m>=0\}$ L3= L1.L2 = $\{a^nb^nc^nd^m\ e^mf^m | m>=0$ and $n>=0\}$ is also recursive.

- **Kleene Closure: If L1is recursive, its kleene closure L1* will also be recursive. For Example:**

L1= $\{a^nb^nc^n | n>=1\}$ L1*= $\{ a^nb^nc^n | | n>=1\}$* is also recursive.

- **Intersection and complement: If L1 and If L2 are two recursive languages, their intersection L1∩L2 will also be recursive. For Example:**

L1= $\{a^nb^nc^nd^n | n>=0$ and $m>=0\}$ L2= $\{a^nb^nc^nd^n | n>=0$ and $m>=0\}$ L3=L1∩L2 = $\{ a^nb^nc^nd^n | n>=0\}$ will be recursive.

also closed under complement

NOTE

As opposed to REC languages, RE languages are not closed under complementation which means complement of RE language need not be RE.

Decidable Problems

A problem is decidable if there exists an algorithm (Turing machine) that can determine the answer (yes or no) for any given input in a finite amount of time. Decidable problems are also called recursive problems.

Examples of Decidable Problems:

Membership Problem for Regular Languages:

Problem: Given a regular language L and a string W, determine if w∈L

Solution: Use a deterministic finite automaton (DFA) to check membership. This can be done in linear time relative to the length of w

Membership Problem for Context-Free Languages:

Problem: Given a context-free language L L and a string www, determine if w∈L

Solution: Use a pushdown automaton (PDA) or the CYK algorithm to check membership.

Undecidable Problems

A problem is undecidable if there is no algorithm (Turing machine) that can provide a solution for all possible inputs. For undecidable problems, there may be some instances where an answer can be determined, but there is no general algorithm that works for all cases.

Examples of Undecidable Problems:

The Halting Problem:

Problem: Given a Turing machine Mand an input w, determine if M halts on w.

Solution: Proven to be undecidable by Alan Turing. No algorithm can determine for all (M,w)pairs whether M will halt on w.

Explain the modification done in finite automata to make it PDA and Turing machine

1. From Finite Automaton to Pushdown Automaton (PDA)

➢ Addition of a Stack:

- A PDA is essentially a finite automaton equipped with an additional stack memory.
- The stack allows the PDA to have potentially infinite memory, which it can use to store and retrieve data dynamically during the computation

➢ transition Function Enhancement:

- The transition function of a finite automaton is extended to incorporate the stack operations.
- Instead of just depending on the current state and input symbol, the transition function now also depends on the current top stack symbol.
- The transition function specifies the next state, the symbol to push onto the stack (if any), and the symbol to pop from the stack (if any).

➢ Acceptance Conditions:

- A PDA can accept input by reaching a designated accepting state or by emptying its stack.

A Pushdown Automata (PDA) can be defined as :

Q is the set of states

∑is the set of input symbols

Γ is the set of pushdown symbols (which can be pushed and popped from stack)

q0 is the initial state

Z is the initial pushdown symbol (which is initially present in stack)

F is the set of final states

δ is a transition function which maps Q x {Σ ∪ ∈} x Γ into Q x Γ*.

2. From Finite Automaton to Turing Machine

➢ Addition of a Tape:

- A Turing machine extends a finite automaton by adding a tape that serves as an infinite memory.
- The tape is divided into cells, each capable of holding a symbol from the tape alphabet.
- The tape has a read/write head that can move left or right across the tape.

➢ Transition Function Enhancement:

- The transition function of a finite automaton is extended to control the read/write head and manipulate the tape.

programming techniques for Turing machine

1. Basic Techniques

- **State Management:**
  - Use states to remember where you are in the computation process
  - Clearly define initial, intermediate, and final (accepting/rejecting) states.
- **Symbol Manipulation:**
  - Define the alphabet and use symbols effectively to represent different parts of the input and the intermediate steps.
  - Utilize a special blank symbol (often denoted as B or _) to mark unused tape cells.

- The transition function specifies the next state, the symbol to write on the tape, and the direction to move the tape head (left or right).

➢ **Acceptance Conditions:**

- A Turing machine accepts input by entering a designated accepting state.
- Unlike PDAs, Turing machines can also reject input by entering a rejecting state

A TM is expressed as a 7-tuple (Q, T, B, Σ, δ, q0, F) where:

Q is a finite set of states

T is the tape alphabet (symbols which can be written on Tape)

B is blank symbol (every cell is filled with B except input alphabet initially)

Σ is the input alphabet (symbols which are part of input alphabet)

δis a transition function which maps Q × T -> Q × T × {L,R}. q0 is the initial state

F is the set of final states. If any state of F is reached, input string is accepted

- **Tape Movements:**
  - Carefully design transitions to move the tape head left (L) or right (R) based on the current state and tape symbol.
  - Use movements to navigate the tape efficiently, avoiding unnecessary back-and-forth.
- **Conditional Branching:**
  - Implement conditional logic by branching to different states based on the current symbol read from the tape.
  - This allows the Turing machine to make decisions and perform different actions.

Advanced Techniques

Subroutines:

- Break down complex tasks into smaller subroutines (groups of states and transitions) that can be reused.
- Use special states to indicate the start and end of a subroutine.

Markers and Flags:

- Use special symbols as markers to track positions on the tape or to remember certain conditions.
- Markers help manage loops and ensure the machine can find its way back to specific positions.

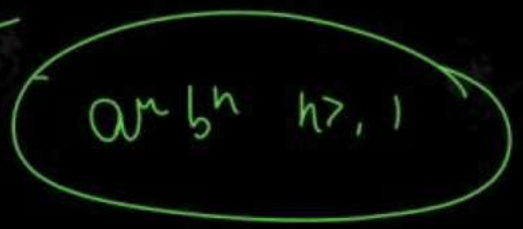
Loops and Iteration:

- Implement loops by creating cycles in the state transitions.
- Use loops for repeated tasks, such as scanning the tape, counting, or copying symbols.

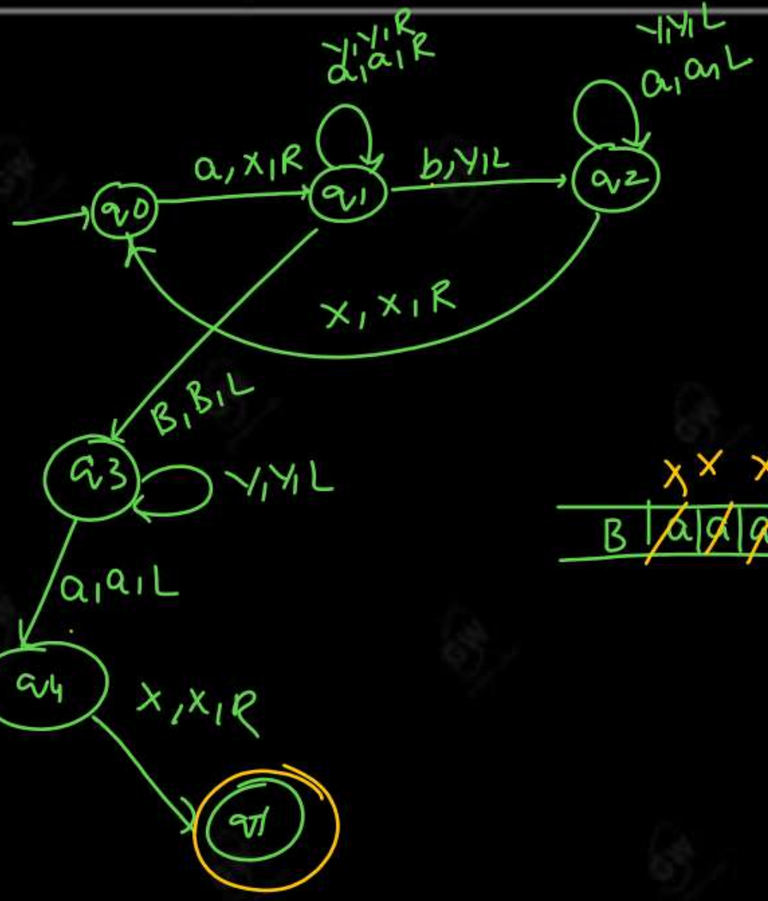
Simulation of Other Machines:

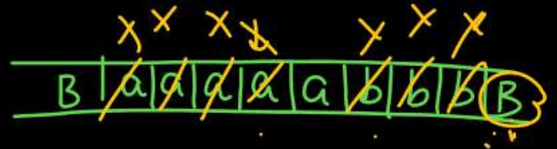
- Design the Turing machine to simulate simpler machines like finite automata or pushdown automata.
- This involves encoding the states and transitions of the simulated machine into the Turing machine's states and tape symbols

Example



Turing machine $a^{n+2}\ b^n$ n>0 0r n>=1





$\delta(q_0, a) = (q_1, X, R)$
$\delta(q_1, a) = (q_1, a, R)$
$\delta(q_1, Y) = (q_1, Y, R)$

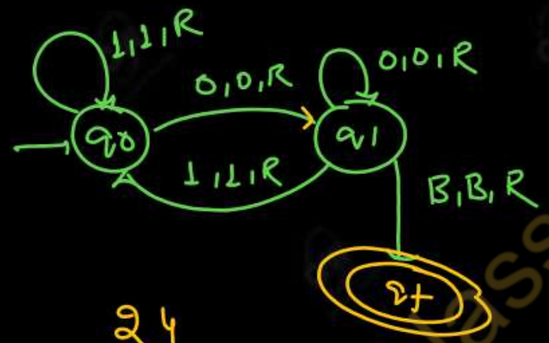
	a	b	X	Y	B
q_0	$q_1 X R$				
q_1	$q_1 a R$			(q_1, Y, R)	
⋮					

Turing machine that accept language of even number written in binary

0,2,4,6,8,10,12

1	1	0	0	0	B

	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0



16

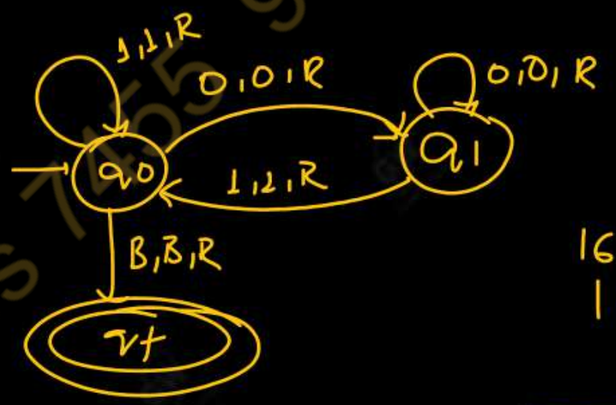
16	8	4	2	1	
1	0	0	0	0	- 16
1	1	0	0	0	- 24
1	1	1	0	0	- 28
1	1	1	1	0	- 30

24

16	8	4	2	1
1	1	0	0	0

01011 → 11

TM Accept odd number written in Binary



16	8	4	2	1	
1	0	1	0	1	21

1	0	1	0	1	B

Recursive Function

A Initial function:

There are three initial functions-

1 Zero function

It is define as $Z(x)=0$

$Z(50)=0$ $Z(70)=0$

2 Successor function

It is defined as

$S(x)=x+1$

$S(6)=6+1=7$

$S(70)=70+1=71$

3 projection function

$P_1^n(x_1, x_2 \ldots x_n) = x_i$

$P_2^3 = (7,9,12) = 9$

$P_4^6 = (2,5,12,15,18,8) = 15$

B composition function

A composite function is define when one function is substituted to another function

let $f(x)=3x+2$

$g(x)=x+5$

then $f(g(x)) = f(x+5)$

$3(x+5)+2$

$3x+17$

$g(f(x)) = g(3x+2)$

$3x+2+5$

$3x+7$

Recursive Function

Recursive function

A function that return itself again and again

Primitive recursive function

A function is said to be primitive recursive function If it can be obtained from initial function through finite number of composite and recursive state

Partial recursive function

A function is said to be partial recursive function, if it is defined for some of its argument

subtraction of two positive number m and n

F(m,n)=m-n m>=n

M<n is not defined

Ackerman's function

It is a function with two argument , each of which can be assigned any non negative integer value

It is nota primitive recursive function but a recursive function

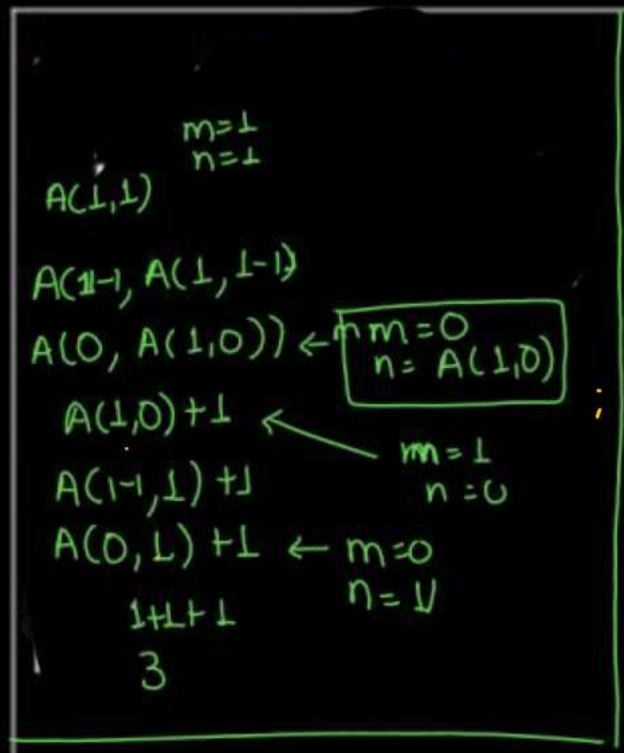
A(m,n)

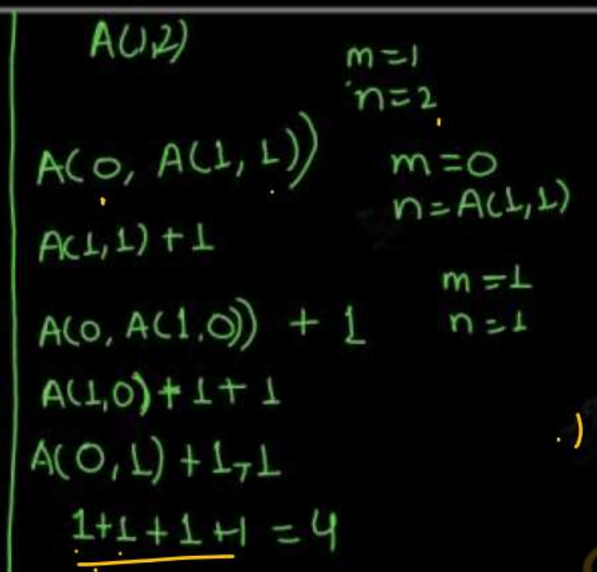
If m=0,n ≠0 then A(0,n)= n+1

If m≠0 , n=0 then A(m,0)= A(m-1,1)

If m≠0 , n≠0 then A(n,m)=A(m-1,A(m,n-1))

calculate A(1,1) and A(1,2) using AKERMAN FUNCTION





Ackerman's function

It is a function with two argument , each of which can be assigned any non negative integer value

It is nota primitive recursive function but a recursive function

A(m,n)

If m=0,n ≠0 then A(0,n)= n+1

If m≠0 , n=0 then A(m,0)= A(m-1,1)

If m≠0 , n≠0 then A(m,n)=A(m-1,A(m,n-1))

Q The complement of Recursive language is also Recursive

Let L be a recursive language

Let M be a turing machine that halts on all i/p & accept L

L' be the complement of L & M' is the turing machine which accept L'

M accepting state is the non-accepting of M' & M' accepting state is the non-accepting state of M

So M' all on all input & accept L'

halt

So we can L' is also Recursive language.

Strong church thesis

Church-Turing Thesis

- The basic Church-Turing Thesis says:
- If you can solve a problem using any step-by-step method (algorithm), then a Turing machine can also solve that problem.
- A Turing machine is a simple abstract computer that can simulate any algorithmic process.

Strong Church-Turing Thesis

- The Strong Church-Turing Thesis goes a step further:
- Not only can a Turing machine solve any problem that can be solved by a step-by-step method, but it can also do so efficiently (without taking too much extra time or resources) compared to any other kind of computer you can think of.
- This includes all reasonable kinds of computers, even ones that might be based on physical processes we haven't fully discovered yet.

Strong church thesis

- In other words, the Strong Church-Turing Thesis suggests that Turing machines are not just capable of solving any problem that can be algorithmically solved, but they can do it about as well as any other machine, including future computers. This idea is central to understanding what makes classical computers so powerful and is a fundamental concept in computer science.

PCP (post correspondence problem)

It is undecidable and unsolvable problem

It was introduced by Emil post 1946

Given two list of string A and B of equal length

$A=w_1w_2w_3....w_n$

$B=x_1x_2x_3x_4....x_n$

The problem is to determine if there is a sequence of one or more integer $i_1, I_2..... I_n$ such that

$Wi_1, wi_2, wi_3.....wi_n=xi_1, xi_2, xi_3.....xi_n$ (wi,xi)

is called corresponding pair

Note:- Tractable & non-tractable

↓ Decidable ↓ Non-decidable

Index	A	B
i	wi	xi
1	1	111
2	10111	10
3	10	0

Does PCP instance has solution

Step 1 w_2 x_2

10111 10

Step 2 w_2w_1 x_2x_1

101111 10111

Step 3: $w_2w_1w_1$ $x_2x_1x_1$

1011111 10111111

Step 4 $w_2w_1w_1w_3$ $x_2x_1x_1x_3$

1011111110 1011111110

2113

PCP has solution

PCP (post correspondence problem)

Doe pcp instance ha solution

A={110,0011,0110} w_1 w_2 w_3

B={110 110,00,110}

x_1 x_2 x_3

2, 3, 1

Step 1: w_2 x_2

0011 00

Step 2: $w_2 w_3$ $x_2 x_3$

00110110 00110

Step 3: $w_2 w_3 w_1$ $x_2 x_3 x_1$

00110110110 00110110110

Modified post correspondence problem

It is also a unsolvable problem

The modified PCP has one additional requirement

PCP , first pair in the solution is the first pair in the List , Remaining sequence in the solution can be any order

The problem is to determine if there is a sequence of one or more integer $i_1, I_2 \ldots\ldots I_n$ such that

$Wi_1, wi_2, wi_3 \ldots\ldots\ldots wi_n = xi_1, xi_2, xi_3 \ldots\ldots\ldots xi_n$ (wi,xi)

is called corresponding pair

in

	List A	LIST B
1	10	10
2	11	011
3	110	11

Step1 w_1 = 10, x_1 = 10

Step2: $w_1 w_3$ = 10 110, $x_1 x_3$ = 1011

Step3 $w_1 w_3 w_2$ = 10110 11, $x_1 x_3 x_2$ = 10 11 011

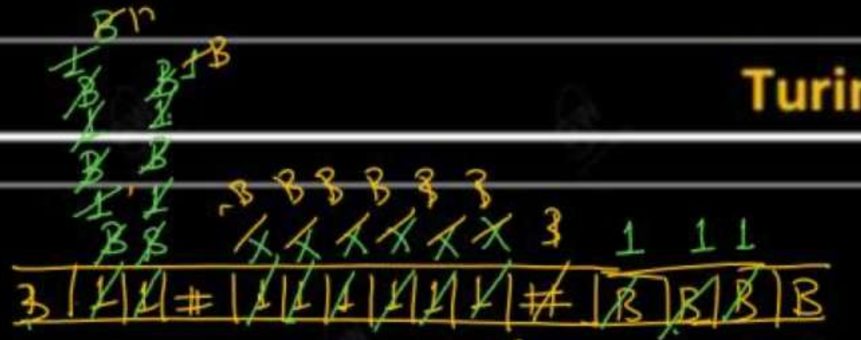
132

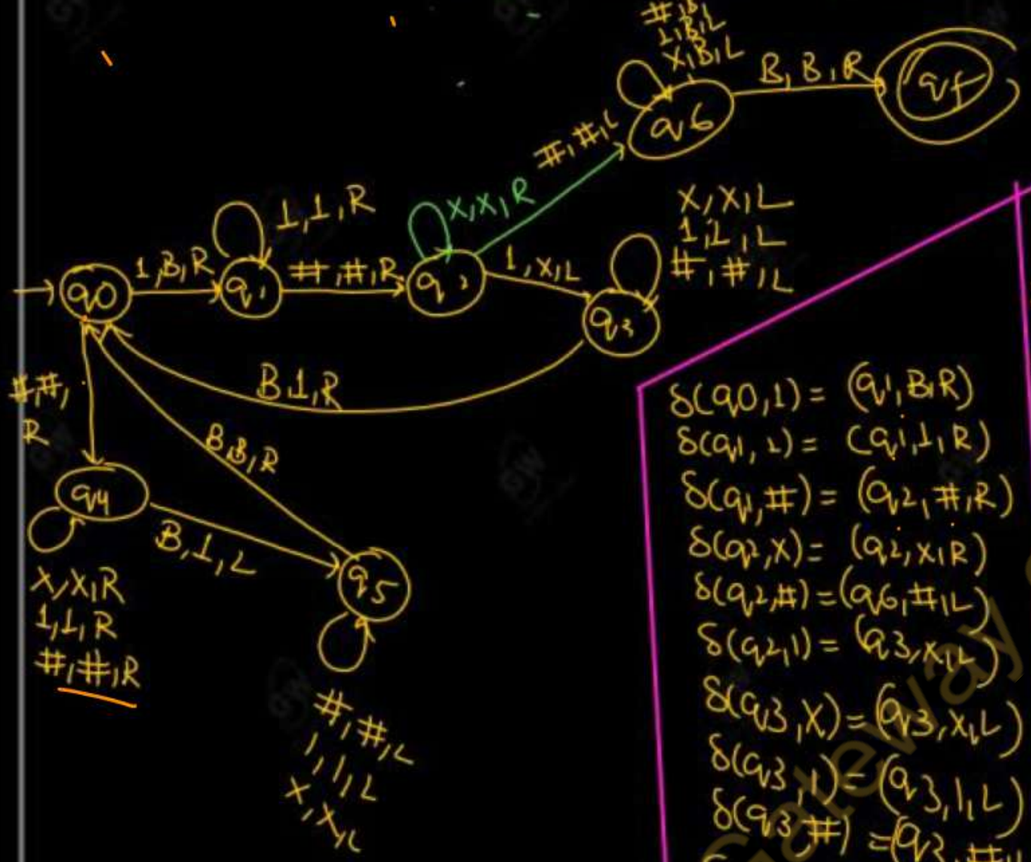
Instantaneous Id of Turing machine

Instantaneous description(ID) in the case of TM is the information in respect of-

- Contents of all cells of the tape, starting from the right most cell up to the last cell, containing a non-blank symbol and containing all cells upto the cell being Scanned
- The cell currently being scanned by the machine
- The sate of the machine

Turing machine for division





$\delta(q_0,1) = (q_1,B,R)$
$\delta(q_1,1) = (q_1,1,R)$
$\delta(q_1,\#) = (q_2,\#,R)$
$\delta(q_2,X) = (q_2,X,R)$
$\delta(q_2,\#) = (q_6,\#,L)$
$\delta(q_2,1) = (q_3,X,L)$
$\delta(q_3,X) = (q_3,X,L)$
$\delta(q_3,1) = (q_3,1,L)$
$\delta(q_3,\#) = (q_3,\#,L)$
$\delta(q_3,B) = q_0,1,R)$

$Q =$
$\{q_0,q_1,q_2,$
$q_3,q_4,$
$q_5,q_6,q_f\}$
$q_0 = \{q_0$
$F = \{q_f\}$
$\Gamma = \{1,X,B,\#\}$
$B = B$
δ

$\delta(q_0,\#) = (q_4,\#,R)$
$\delta(q_4,X) = (q_4,X,R)$
$\delta(q_4,1) = (q_4,1,R)$
$\delta(q_4,\#) = (q_4,\#,R)$
$\delta(q_4,B) = (q_5,1,L)$

$\delta(q_5,\#) = (q_5,\#,L)$
$\delta(q_5,1) = (q_5,1,L)$
$\delta(q_5,X) = (q_5,X,L)$
$\delta(q_5,B) = (q_0,B,R)$

$\delta(q_6,\#) = (q_6,B,L)$
$\delta(q_6,1) = (q_6,B,L)$
$\delta(q_6,X) = (q_6,B,L)$
$\delta(q_6,B) = (q_f,B,R)$

	X	B	1	#
q_0			(q_1,B,R)	$q_4,\#,R$
q_1			$(q_1,1,R)$	$(q_2,\#,R)$
q_2	(q_2,X,R)		(q_3,X,L)	$q_6,\#,L$
q_3	(q_3,X,L)	$(q_0,1,R)$	$(q_3,1,L)$	$q_3,\#,L$
q_4	(q_4,X,R)	$(q_5,1,L)$	$(q_4,1,R)$	$q_4,\#,R$
q_5	(q_5,X,L)	(q_0,B,R)	$(q_5,1,L)$	$(q_5,\#,L)$
q_6	(q_6,B,L)	q_f,B_R	(q_6,B,L)	(q_6,B,L)
q_f	—	—	—	—