TURING MACHINE

Standard Turing Machine

Turing Machine can be defined as $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$

$Q \Rightarrow$ finite set of states

$\Sigma \Rightarrow$ finite set of i/p alphabets

$\Gamma \Rightarrow$ finite set of tape alphabets

$\delta \Rightarrow \quad \delta : (Q \times \Gamma) \longrightarrow (Q \times \Gamma \times \{L, R\})$

$q_0 \in Q \Rightarrow$ Initial state

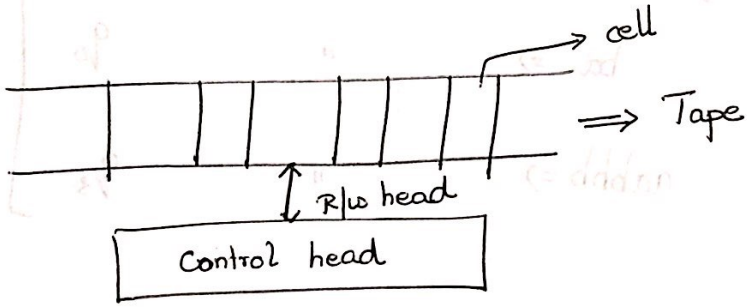
$B \in \Gamma \Rightarrow$ is used to represent blank

$F \subseteq Q \Rightarrow$ set of final states.

partially deterministic

Ex:- $\Sigma = \{a, b\} \quad \Gamma = \{a, b, X, Y, B\}$

Σ is the subset of Γ.

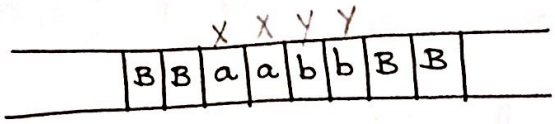


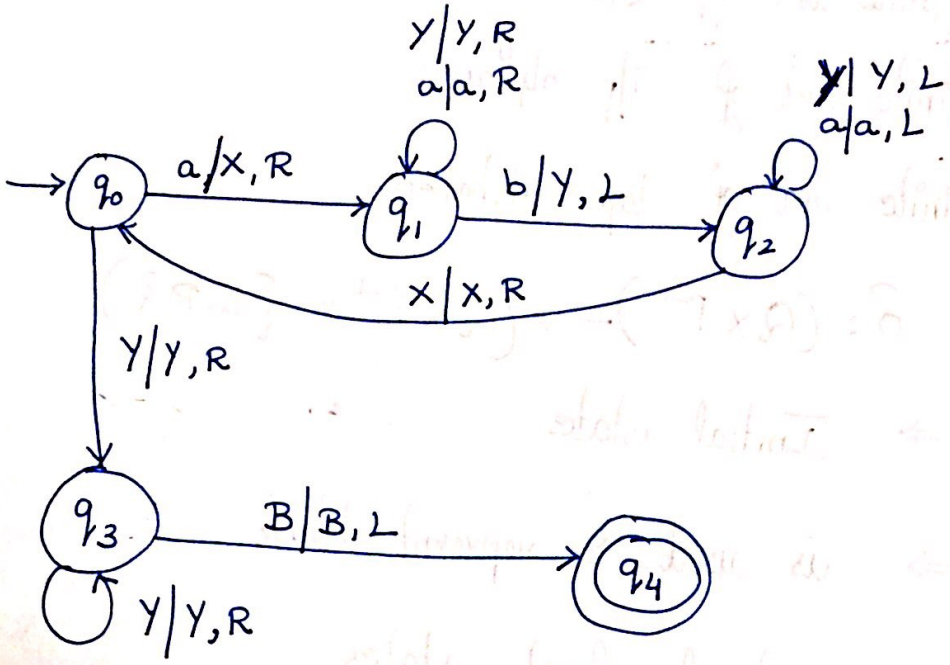
Summary:

1. Tape is unbounded, so any number of left and right moves possible.
2. It is deterministic, i.e., atmost one move for each configuration.
3. No special i/p or o/p file.

Examples for Turing Machine:

① Language $L_1 = \{ a^n b^n \mid n \geq 1 \}$





For the string aabb ⇒ TM will halt at the final state q_4.

For the string aaabb ⇒ TM will halt at q_1

ba ⇒ " q_0

aabbb ⇒ " q_3

} dead configuration (no moves after this)

Equivalent Transition Function: for $L = \{a^n b^n\}$

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

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

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

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

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

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

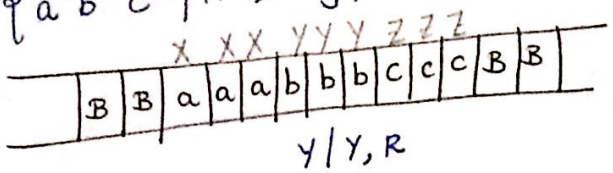
$$\delta(q_2, Y) = (q_2, Y, L)$$

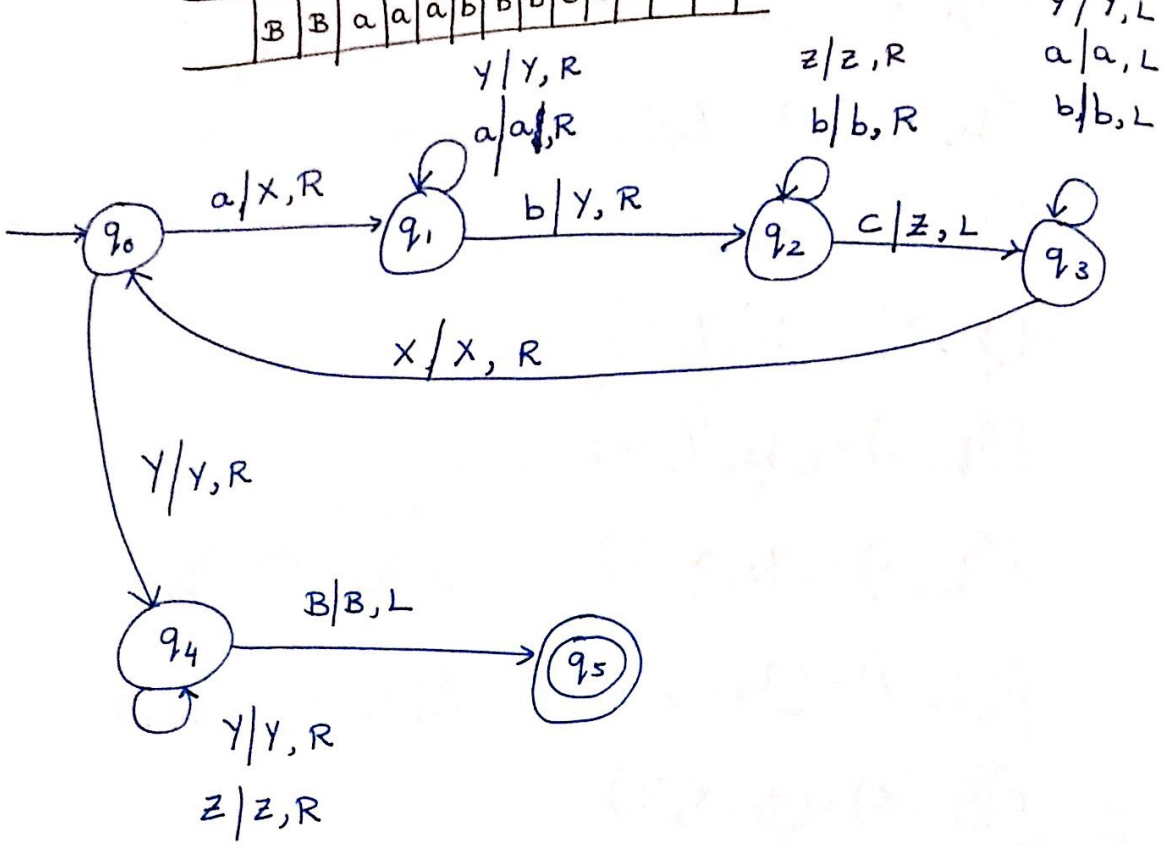
$$\delta(q_2, X) = (q_0, X, R)$$

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

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

② $L = \{a^n b^n c^n \mid n \geq 1\}$.





String	Halts at
aabbbcc	q_4
aabbccc	q_4
bbaacc	q_0
aaccbb	q_1

Automata

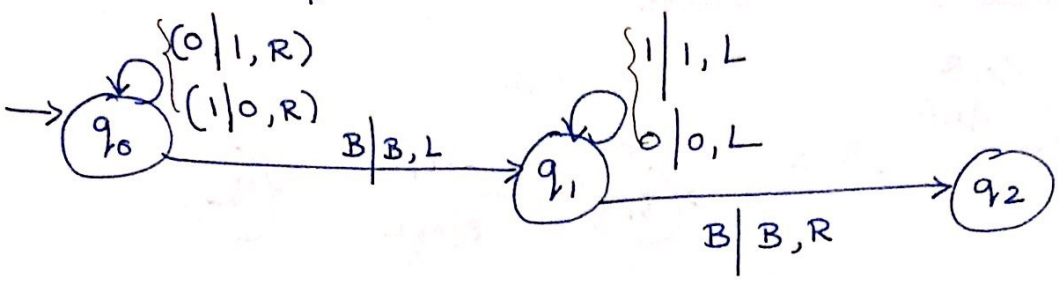
Acceptor — accepts i/p; o/p → Yes (accept) / No (reject)

Transducer — converts i/p to o/p.

③. TM to find 1's complement of a binary number.

Ex: i/p => 1100

o/p => 0011.

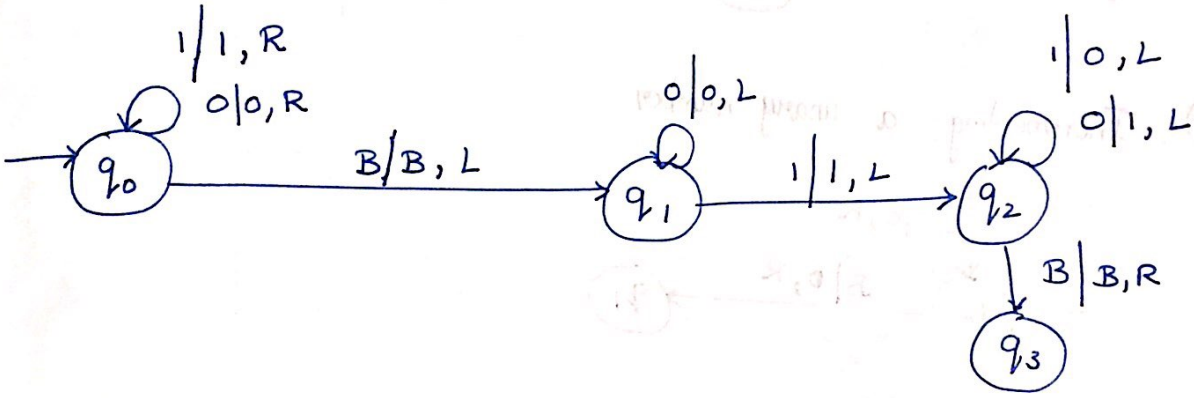


③. TM to find 2's complement of a binary number.

[Assume no overflow i/p => n bits o/p => n bits].

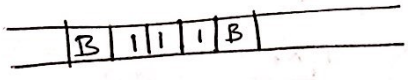
i/p₁ => 0 11 1 0 0 0 i/p₂ => 0 0 1 1 0 0

2's complement => 1 0 0 1 0 0 0 2's c => 1 1 0 1 0 0

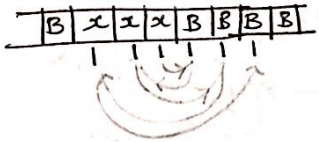


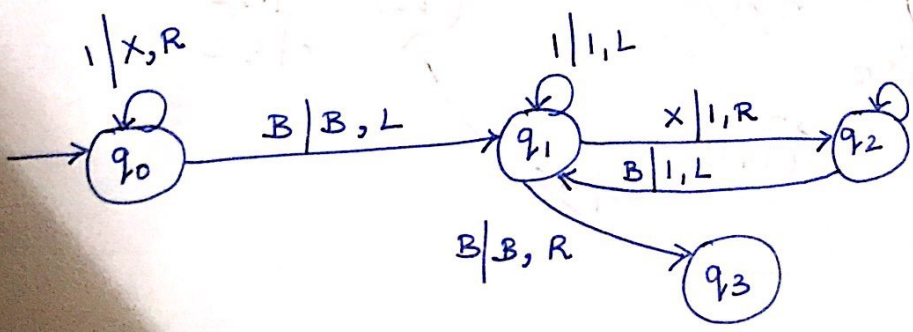
④. TM used for copying.

i/p 11 => o/p 1111



i/p 111 => o/p 111 111

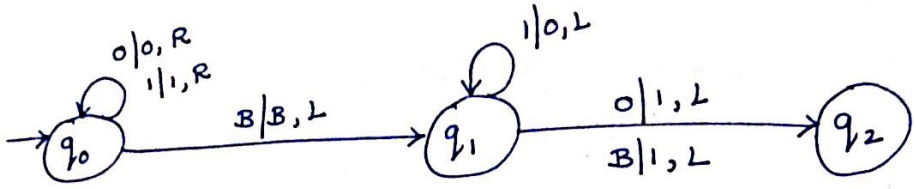




⑤. Construct a TM for incrementing a given binary

1100 + 1 = 1101

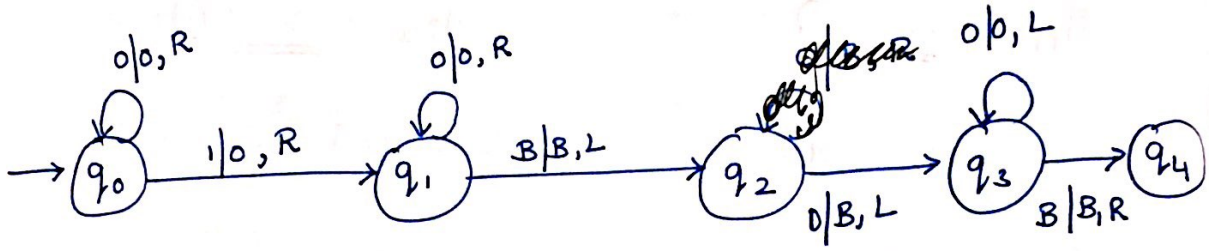
1101 + 1 = 1010



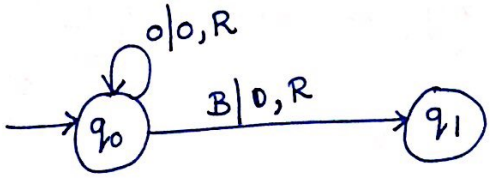
⑥ TM to add 2 integers.

i/p : $0^m 1 0^n$

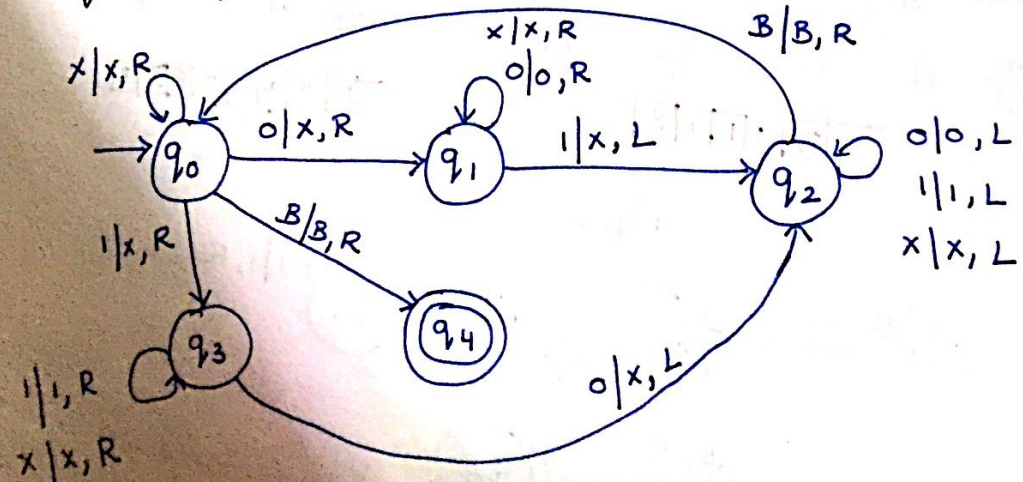
o/p : 0^{m+n}



⑦. Incrementing a unary number.



⑧. Equal no. of 0's & 1's.



⑨. Multiply

⑨. Multiply 2 unary numbers.

i/p : 0^3 0^5

o/p : $0^{3\times5}$

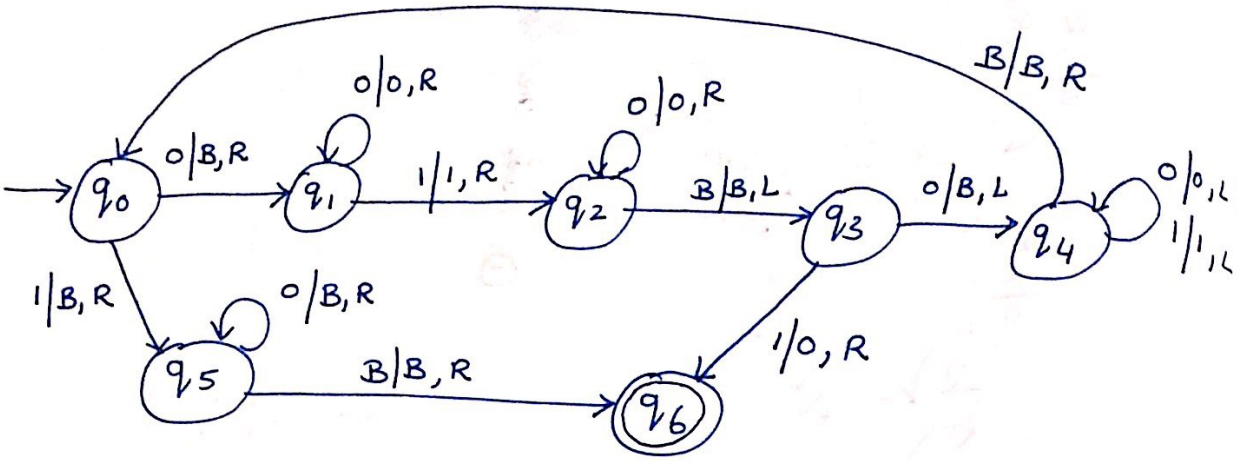
① Mark the separator.

i/p ⇒ ∴ B 0 0 0 1 0 0 0 B ...

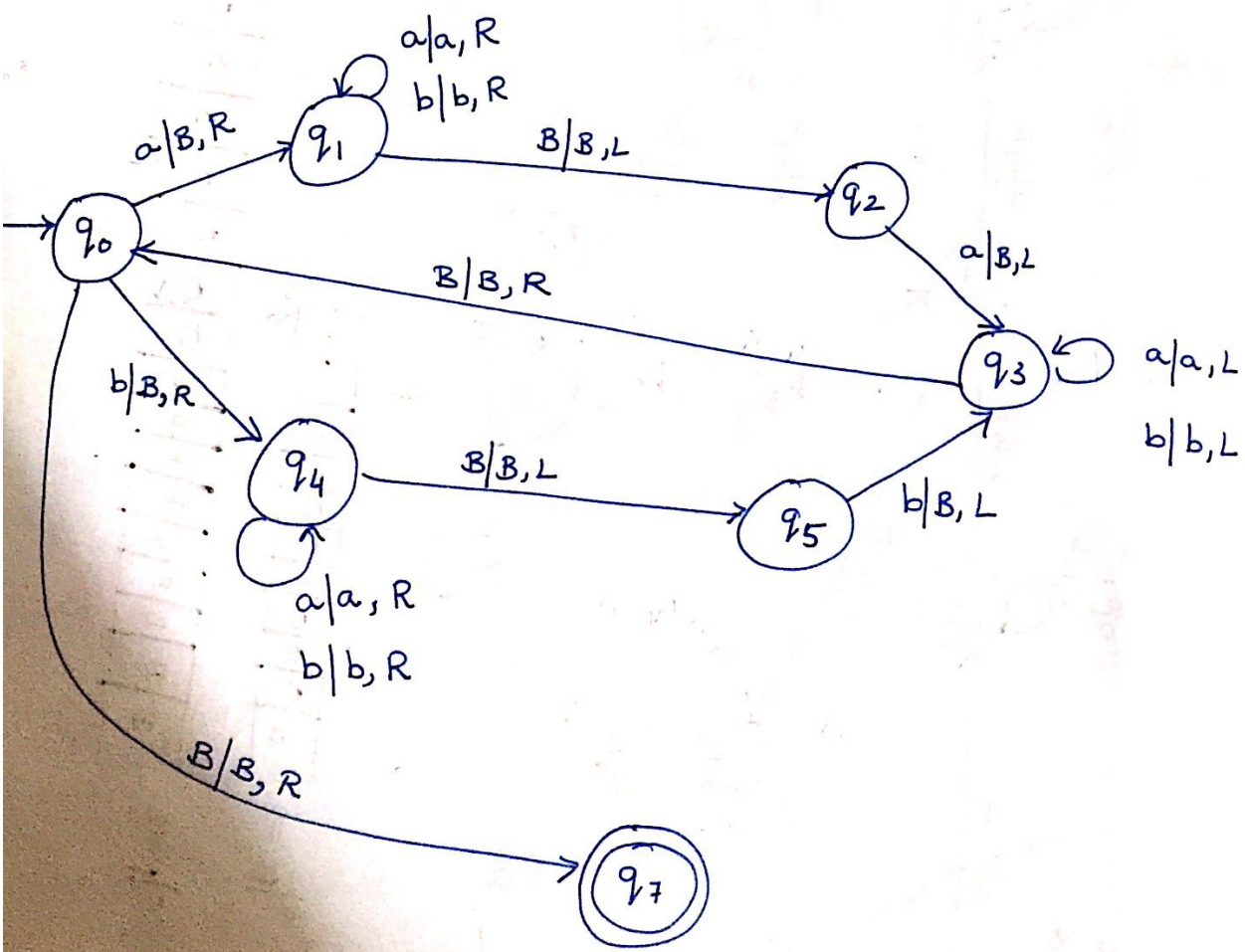
After processing; o/p ⇒ ∴ B B B B B B B 0 0 0 0 0 0 0 0 B ...

(10). Design a TM to subtract two non-negative integers. Function is precisely defined below.

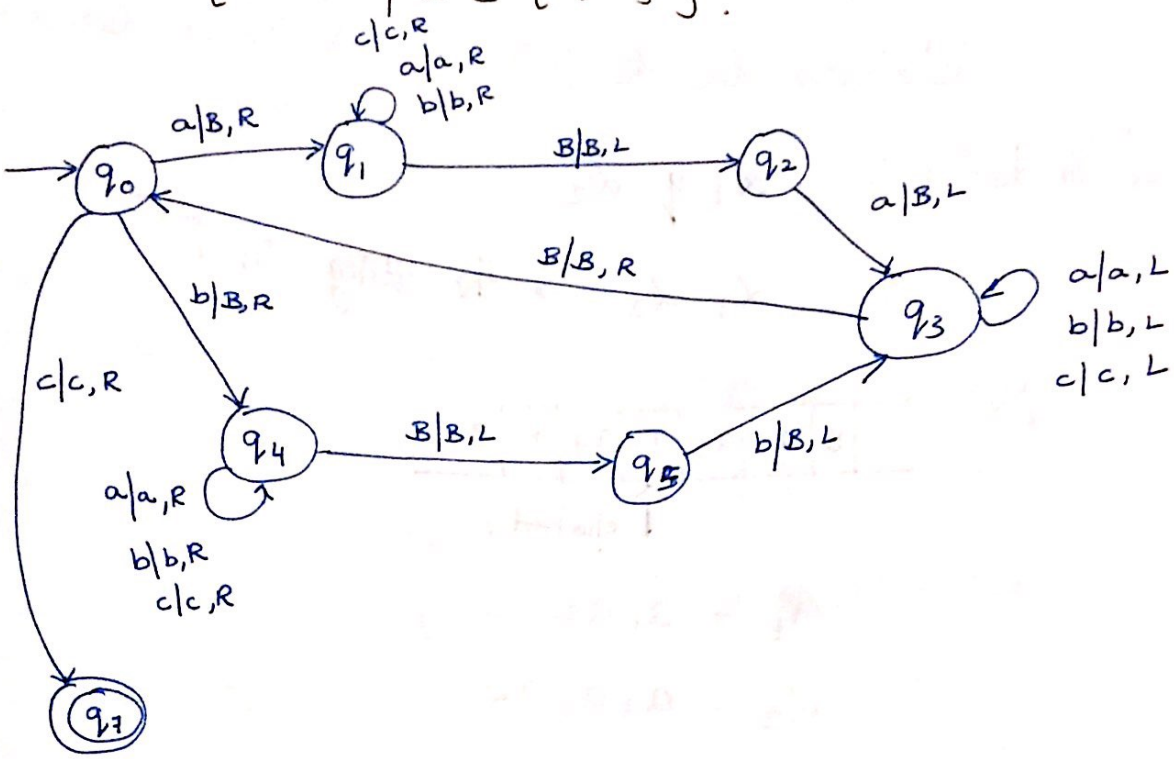
$$f = m-n \quad \text{if } m \geq n$$
$$= 0 \quad \text{otherwise.}$$



. $L = \{ ww^R \mid w \in \{a,b\} \}$



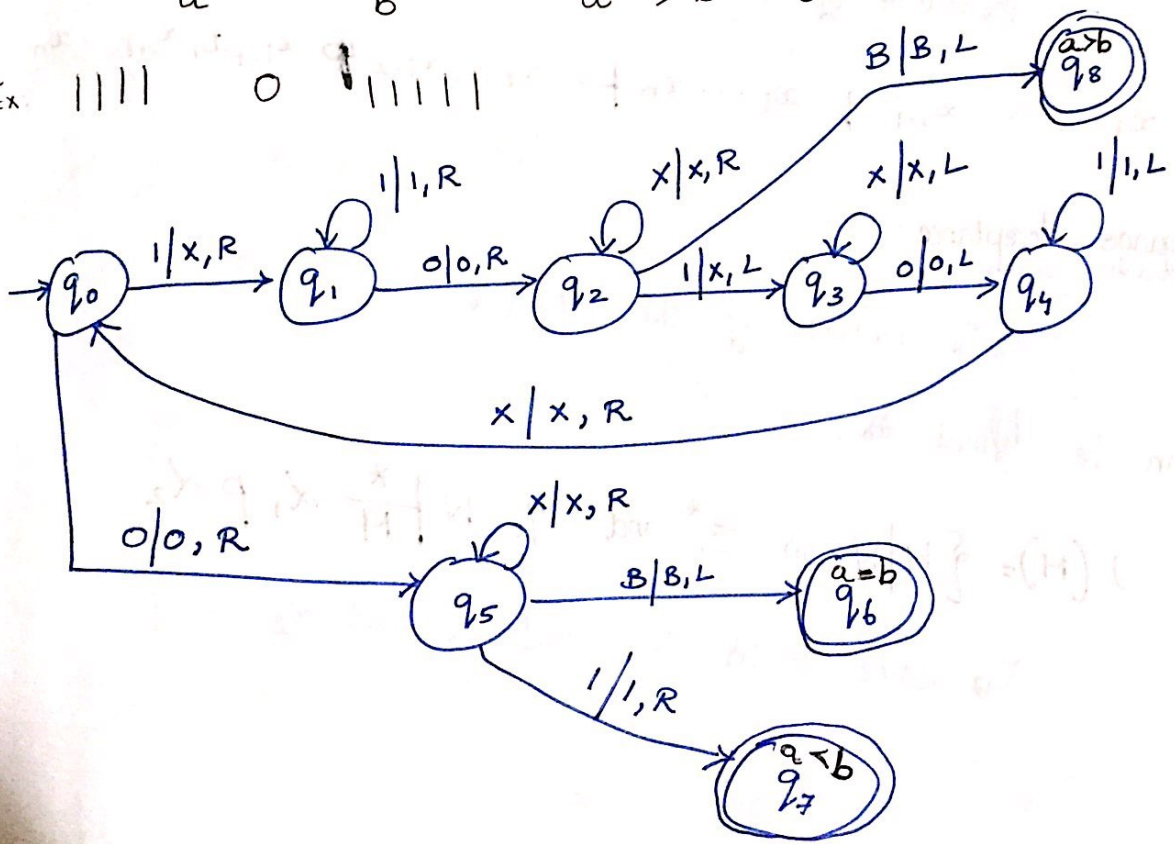
(12) $L = \{ wcw^R \mid w \in \{a, b\} \}$.



(13) Comparator:

a b $a > b$ or $a < b$ or $a = b$

Ex |||| 0 |||||



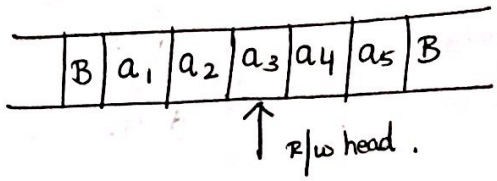
TM can perform any mathematical function.

Instantaneous Description (ID):

Instantaneous description (ID) of the Turing Machine M can be denoted by $\alpha_1 \, q \, \alpha_2$

where.. $\alpha_1 \, \alpha_2 \rightarrow$ the string in Γ^*

Ex:-



$\alpha_1 = a_1 \, a_2$

$\alpha_2 = a_3 \, a_4 \, a_5$

Moves in a TM:

The change of ID is represented by:

$$x_1 \, x_2 \cdots x_{i-1} \, q \, x_i \cdots x_n \vdash x_1 \cdots x_{i-2} \, p \, x_{i-1} \, x_i \, x_{i+1} \cdots x_n$$

Language acceptance:

The language of the TM $(Q, \Sigma, \Gamma, \delta, B, F)$ M can be defined as...

$$L(M) = \{ W \mid W \text{ in } \Sigma^* \text{ and } q_0 \, W \overset{*}{\underset{M}{\vdash}} \alpha_1 \, p \, \alpha_2$$

For some P in F, and α_1 and α_2 in Γ^*

Transi

Transiti

State
q_0
q_1
q_2
q_3
q_4
q_5

Transition Function for the comparator:

$\delta(q_0, 1) = (q_1, X, R)$

$\delta(q_0, 0) = (q_5, \blacksquare, R)$

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

$\delta(q_1, 0) = (q_2, 0, R)$

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

$\delta(q_2, B) = (q_8, B, L)$

$\delta(q_2, 1) = (q_3, X, L)$

$\delta(q_3, X) = (q_3, X, L)$

$\delta(q_3, 0) = (q_4, 0, L)$

$\delta(q_4, 1) = (q_4, 1, L)$

$\delta(q_4, X) = (q_0, X, R)$

$\delta(q_5, X) = (q_5, X, R)$

$\delta(q_5, B) = (q_6, B, L)$

$\delta(q_5, 1) = (q_7, 1, R)$

Transition Table:

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

Computation M (comparator) on input 11011:

$q_0 11011 \vdash X q_1 1011$

$\vdash X1 q_1 011$

$\vdash X10 q_2 11$

$\vdash X1 q_3 0X1$

$\vdash X q_4 10X1$

$\vdash B q_4 X10X1$

$\vdash X q_0 10X1$

$\vdash XX q_1 0X1$

$\vdash XX0 q_2 X1$

$\vdash XX0X q_2 1$

$\vdash XX0 q_3 XX$

$\vdash XX q_3 0XX$

$\vdash X q_4 X0XX$

$\vdash XX q_0 0XX$

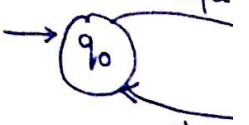
$\vdash XX0 q_5 XX$

$\vdash XX0X q_5 X$

$\vdash XX0XX q_5 B$

$\vdash XX0X q_6 XB$

Non-halting TM:



Turing Thesis: 19

Any co

"mechanical means

Some

definition of

(a). Anything th

also be do

(b). No one

by what

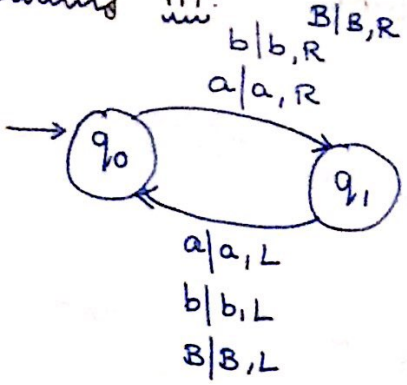
turing mac

(c). Alternati

computati

than the

Non-halting TM:



≅ infinite loop

Turing Thesis: 1930's till date nobody was able to prove his work is wrong.

Any computation that can be carried out by "mechanical means" can be performed by some turing machine.

Some arguments why turing thesis is accepted as definition of mechanical computation or computer.

(a). Anything that can be done by existing digital computer can also be done by TM.

(b). No one has yet been able to suggest a problem, solvable by what we intuitively consider an algorithm, for which a turing machine program cannot be written.

(c). Alternative models have been proposed for mechanical computation, but none of them are more powerful than the turing machine model.

Some modifications to std. Turing Machine:

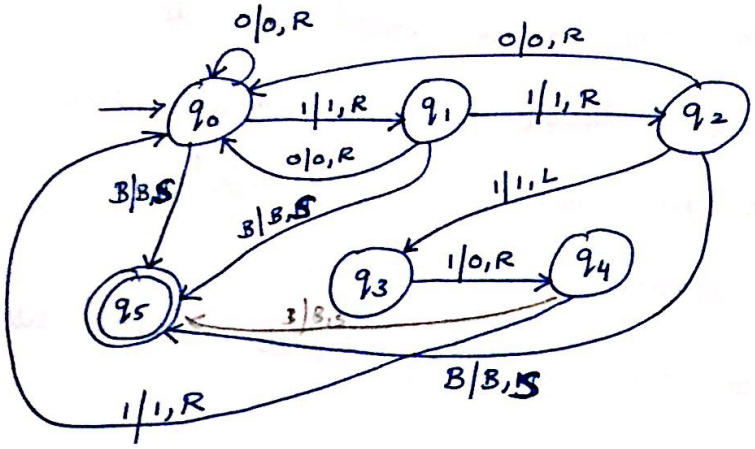
① Turing machine with stay option:

TM with stay option can be defined as...

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

where $Q, \Sigma, q_0, F, B, \Gamma$ have same meaning as std. TM.

But $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R, S\}$.



Ex: TM with stay option for converting each occurance of "111" by "101".

② Turing machine with semi-infinite tape:

This TM will have a left boundary.

This machine is identical to std. TM, except that no left move is permitted when the read-write head is at the boundary.

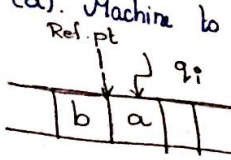
Simulation:

The simulating machine M has a tape with 2 tracks.

Upper track: keep the information to the right of ref. pt on M

Lower track: contains the left part of M's tape in reverse order.

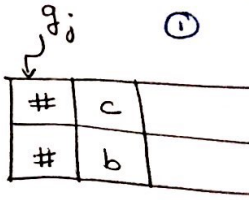
(a). Machine to



Transition: ① $\delta(q_i$

q_i, q_j

P_j



③. Off-Line Turing

⊛ $(Q, \Sigma,$

⊛ Addition

given a

⊛. I/P

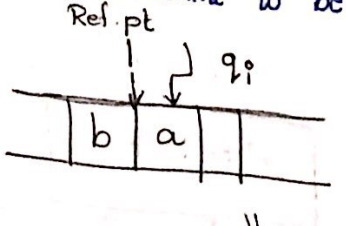
⊛. If m

i/p fi

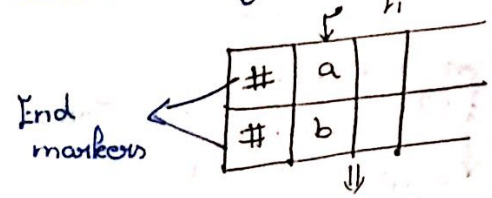
⊛. After

Remain

(a). Machine to be simulated.



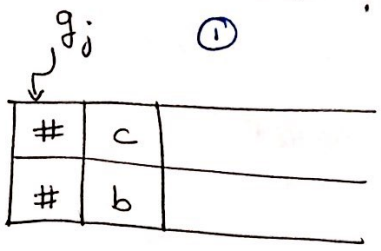
(b). Simulating machine.

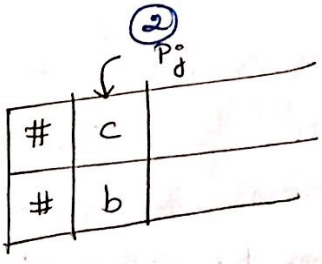


Transition: ① $\delta(q_i, a) = (q_j, c, L)$ ① $\delta(q_i, (a,b)) = (q_j, (c,b), L$

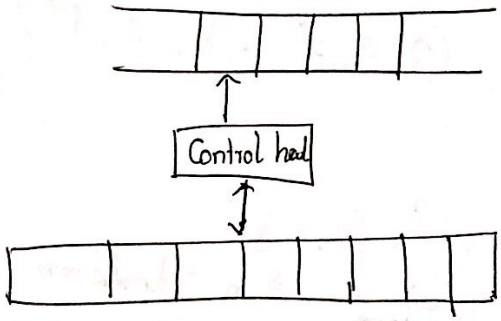
$q_i, q_j \in$ UT (Upper track)
$P_j \in$ LT (Lower ")

② $\delta(q_j, (\#, \#)) = (P_j, (\#, \#), R$





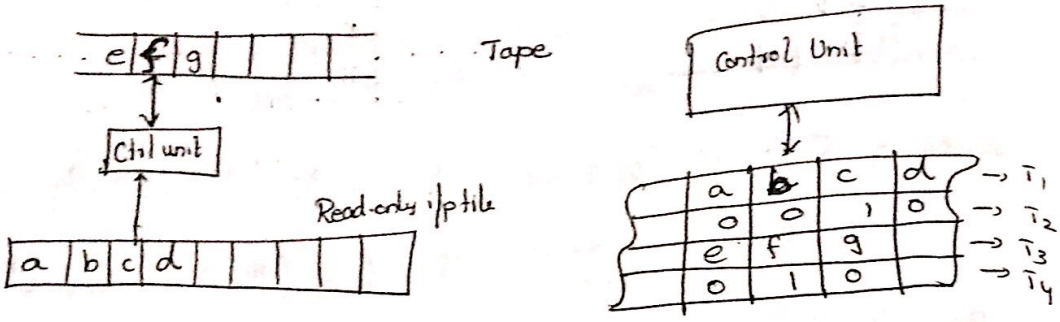
③. Off-Line Turing Machine:



⇒ Read only i/p file

- ⓐ $(Q, \Sigma, \delta, q_0, F, B, \Gamma)$
- ⓐ Additional to the def. of std. TM read only file is given as i/p.
- ⓐ. I/p contents in the file cannot be modified.
- ⓐ. If modifications are needed then copy the contents from i/p file to i/p tape then do the modifications.
- ⓐ. After process, the i/p contents in the file should remain same.

Simulation of off-line TM



T_1 => has the i/p

T_2 => marks the position at which the i/p is read.

T_3 => represents the tape of M

T_4 => shows the position of M's read-write head.

④. Multitape TM:

It is defined as $(Q, \Sigma, \delta, q_0, F, B, \Gamma)$

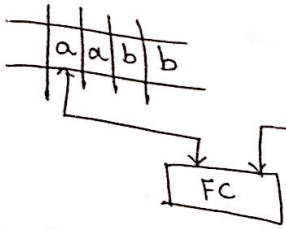
where .,.

$$\delta : \underbrace{Q}_{\text{state}} \times \underbrace{\Gamma^n}_{\substack{\text{read} \\ n \\ \text{tape symbols}}} \longrightarrow \underbrace{Q}_{\text{state}} \times \Gamma^n \times \{L, R\}^n$$

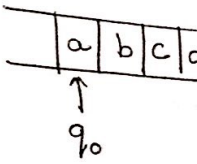
⇓ write on n tape ind.

⇓ n independent moves

Same Language multitape TM.,



⑤. Jumping TM:



Ex:

Jumping

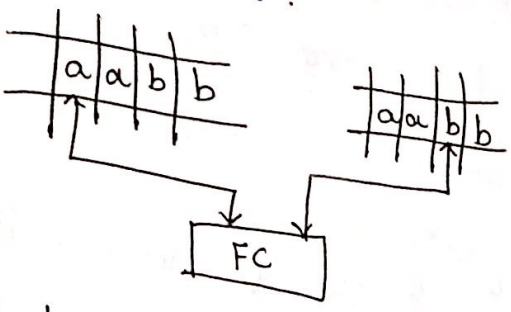
where .,.

δ: Q

⑥. Non-erasing

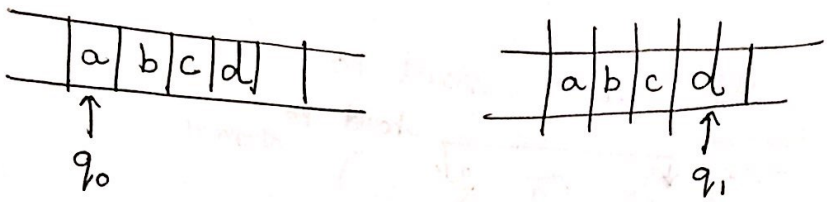
Users are

Same Language if it is implemented with multitape TM.,.



for copying = $O(n)$
comparing = $O^+(n)$

⑤. Jumping TM:



Ex: $\delta(q_0, a) = (q_1, a, R, 3)$

Jumping TM is defined as $(Q, \Sigma, \delta, q_0, F, B, \Gamma)$

where .,.

$$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\} \times \{n\}$$

↓ no. of cells.

⑥. Non-erasing TM:

Users are Not supposed to change the input symbol to blank.

It is defined as $(Q, \Sigma, \delta, q_0, F, B, \Gamma)$ where changing input symbol to blank is forbidden.

To indicate blank user can use diff. symbols.

$\delta(q_0, a) = (q_1, B, R)$ ⇒ Invalid transition in non-erasing TM.

⑦. Always writing TM:

It is defined as $(Q, \Sigma, \delta, q_0, F, B, \Gamma)$ with the restriction that for reading every input the machine should write someother symbol.

Ex $\left.\begin{array}{l}\delta(q_0, a) = (q_1, a, L) \\ \delta(q_1, b) = (q_1, b, R)\end{array}\right\}$ Invalid transistions in always writing TM

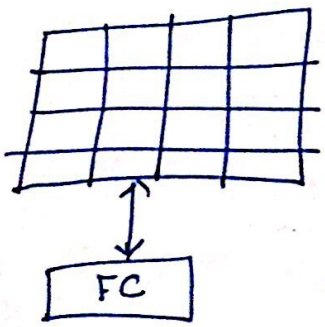
All transistions should be like..,

$\delta(q_0, \underline{a}) = (q_1, \underline{X}, L)$ — should be changed.

⑧. Multidimensional TM:

It is defined as $(Q, \Sigma, \delta, q_0, F, B, \Gamma)$

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



=> 2-dimensional.

$\delta(q_0, a) = (q_1, X, U)$.

For simulation 2-track tape can be used.

a				b			
1	#	2	#	1	0	#	

=> cell contents
=> associated address.

⑨. Multihead TM

It

δ: Q

Ex

δ(q

⑩. Automata

⑪. TM with

⑫. Multitape

⑬. Non-dete

δ:

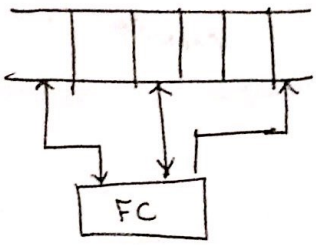
Ex:

(9) Multihead TM:

It can be defined as $(Q, \Sigma, \delta, q_0, F, B, \Gamma)$

$$\delta: Q \times \Gamma^n \longrightarrow Q \times \Gamma^n \times \{L, R\}^n$$

- Γ^n: n symbols on single tape
- Γ^n: writes on single tape
- $\{L, R\}^n$: movement on single tape



Ex

$$\delta(q_0, [a, b, c]) = (q_1, [X, Y, X], [L, L, R])$$

(10). Automata with a Queue:

(11). TM with only 3 states $\Rightarrow$ can design a Universal TM.

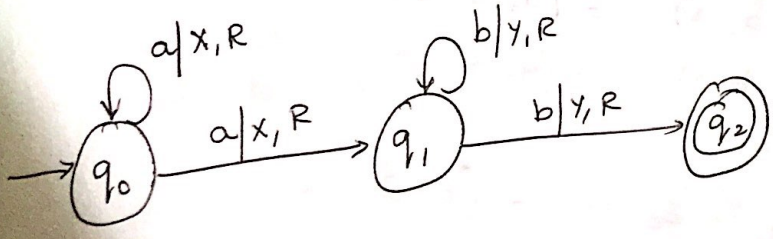
(12). Multitape TM with stay option and atmost 2 states.

(13). Non deterministic TM:

It can be defined as $(Q, \Sigma, \delta, q_0, F, B, \Gamma)$

$$\delta: Q \times \Gamma \longrightarrow 2^{Q \times \Gamma \times \{L, R\}}$$

Ex: $L = \{a^n b^m \mid n, m \geq 1\}$

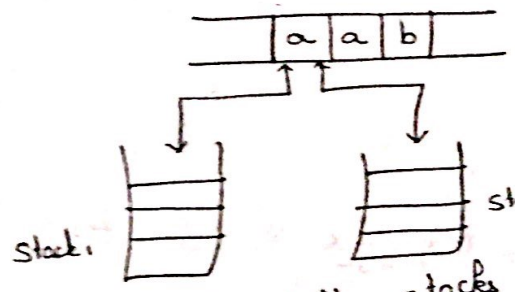


(14) A NPDA with 2 independent stacks: (DPDA with multi-stack machine)

It can be defined as $(Q, \Sigma, \delta, q_0, F, Z_0, \Gamma)$

$$\delta : Q \times (\Sigma \cup (\epsilon)) \times \Gamma \times \Gamma \longrightarrow 2^{Q \times \Gamma^* \times \Gamma^*}$$

Ex: Construct a 2 stack pushdown automata, which creates ww^R where $w \in (a,b)^*$



① Push w in both stacks using following transition.

$\delta(q_0, a, Z_0, Z_0) = (q_0, aZ_0, aZ_0)$

$\delta(q_0, b, Z_0, Z_0) = (q_0, bZ_0, bZ_0)$

$\delta(q_0, a, a, a) = (q_0, aa, aa)$

$\delta(q_0, b, b, b) = (q_0, bb, bb)$

$\delta(q_0, a, b, b) = (q_0, ab, ab)$

$\delta(q_0, b, a, a) = (q_0, ba, ba)$

② Pop each symbol from stack 1 and push to stack 2.

$\delta(q_0, \epsilon, a, a) = (q_0, \epsilon, aa)$

$\delta(q_0, \epsilon, a, b) = (q_0, \epsilon, ab)$

$\delta(q_0, \epsilon, b, a) = (q_0, \epsilon, ba)$

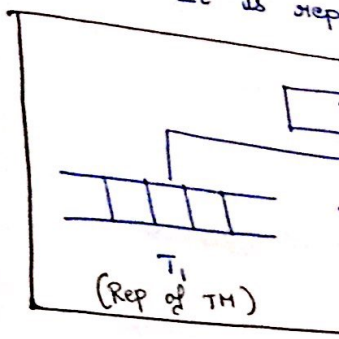
$\delta(q_0, \epsilon, b, b) = (q_0, \epsilon, bb)$

③ $\delta(q_0, \epsilon, Z_0, a) = (f, Z_0, a)$

$\delta(q_0, \epsilon, Z_0, b) = (f, Z_0, b)$

Universal Turing M

* It will ha
* It is rep



Design repo

for eg

Tape T_1 in

some TM

Tape T_2 in

some TM

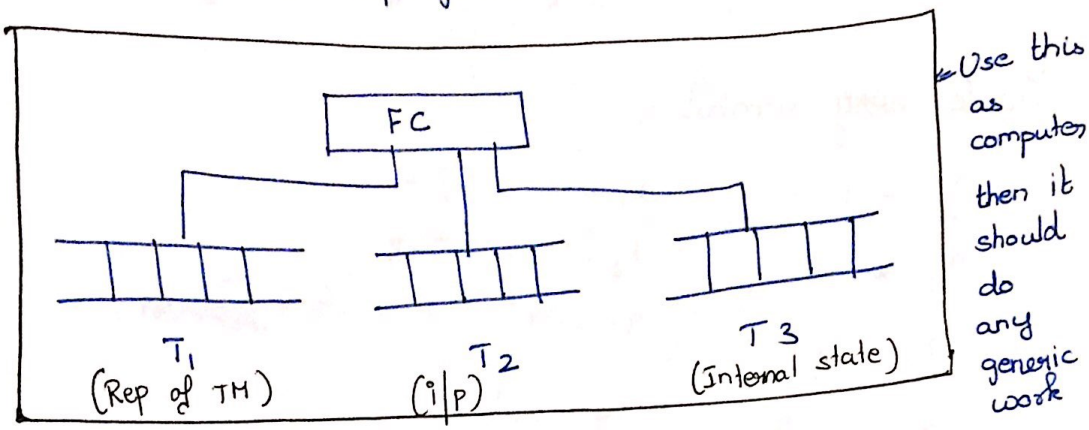
Tape T_3 in

Eg: δ

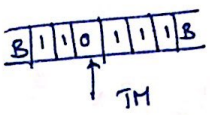
Scanned by CamScanner

Universal Turing Machine:

* It will have multitape TM.
* It is reprogrammable.



Design seperate TM

for eg:.. 

} std TM.

Tape T_1 in UTM contains the representation of some TM (eg)

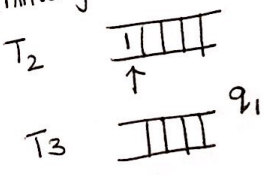
Tape T_2 in UTM contains the original i/p given to some TM (eg)

Tape T_3 in UTM contains the internal states (initially starts with start state)

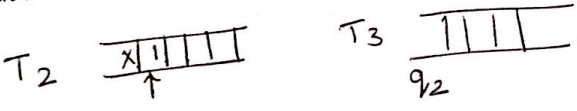
Eg: $\delta(q_1, 1) = (q_2, X, R)$

T_1

initially



after this transition



Without loss of generality..

Set of states $Q = (q_1, q_2, q_3, q_4, \ldots)$

$\Gamma = (a_1, a_2, a_3, a_4, \ldots)$

} General TM

Encode every symbol..

q_1 - 1	a_1 - 1	L - 1
q_2 - 11	a_2 - 11	R - 11
q_3 - 111	a_3 - 111	"0" as seperator
q_4 - 1111	a_4 - 1111	
⋮	⋮	

Ex: $\delta(q_1, a_1) = (q_2, a_2, R)$

0 1 0 1 0 1 1 0 1 1 0 1 1 0

=> Entire TM can be encoded as 0's and 1's

Note:

$\Sigma = \{0, 1\}$ for a TM

① Every TM can be represented as a string of 0's & 1's

② Not every string of 0's & 1's might be a turing machine

Recursively

Consider Σ^*

No membership

Recursive Lan

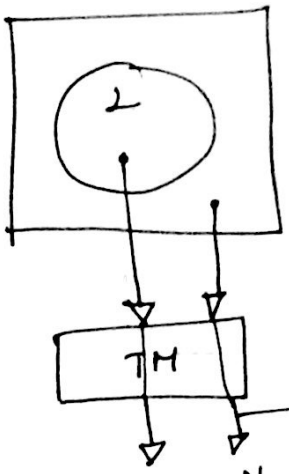
Theorem: I

both recu

Recursively Enumerable Languages

Σ^* => set of strings possible.

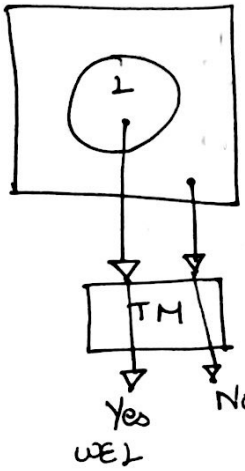
Consider Σ^* then



(ω ∈ L) (ω ∉ L)

No membership algorithm.

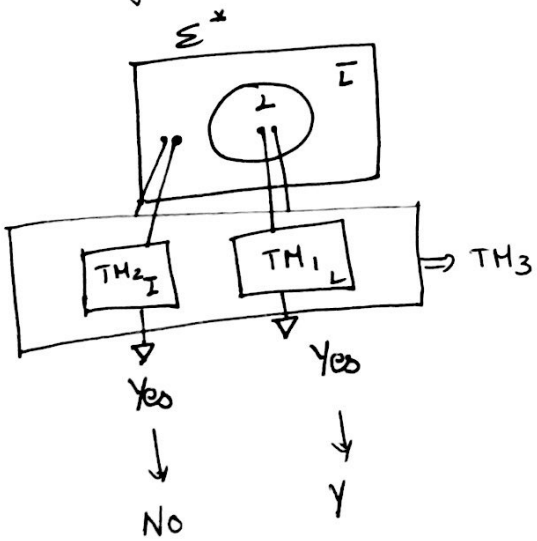
Recursive Language:



ω ∈ L ω ∉ L

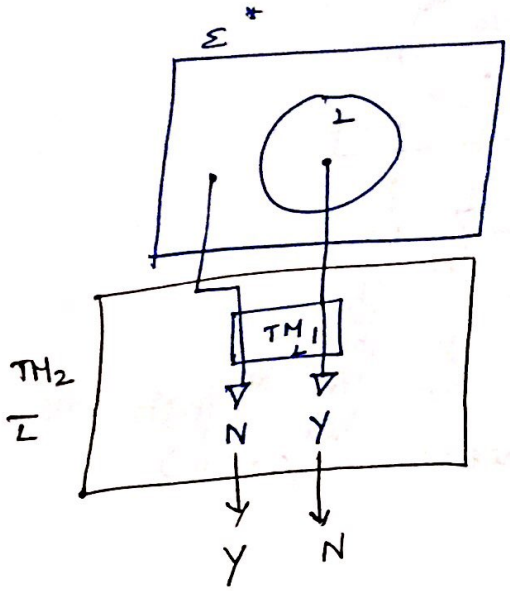
Membership algorithm exist.

Theorem: If a language L and its complement $\bar{L}$ are both recursively enumerable then both languages are recursive.



TM_3 always halts.

If L is recursive, then $\bar{L}$ is also recursive and consequently both are recursively enumerable.



* The complement of a recursive language is recursive
* The union of 2 recursive languages is recursive.
* The union of 2 recursively enumerable language is recursively enumerable.
* If L is a recursive language then $\Sigma^* - L$ is recursive.

COUNTABILITY:

$F < C < U$

Sets < Finite, Infinite < Countable, Uncountable

Countable:

A set 'S' is said to be countable, if all the elements of the set can be put in one to one correspondence with the set of natural numbers.

Uncountable:

A set is uncountable, if it is infinite and not countable

Set of all even nos

$i \in E = \{0, 2, 4, 6, 8, 10, 12, \ldots\}$

$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$ → one to one correspondence

$(i+1) \in N = 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7, \ldots$

} Ex. for countable

Set of all real nos ⇒ Ex. for uncountable

Alternative definition of countability:

A set is said to be countable if there exists an enumeration method using which all the elements of the set can be generated and for any particular element, it takes only finite number of steps to generate it. The finite number of steps taken to generate an element can be used as its index and hence a mapping into natural numbers.

Examples,

① Set of all even

for

② Set of all

③ set of

the nos

④ "P|q",

S =

Examples:

① Set of all even numbers

```
for (i=0 to ∞)
{
    generate (2i);
}
```

⟸ ①. enumeration method

0, 2, 4, 6, ...
↓ ↓ ↓ ↓
1 2 3 4

⟸ ②. for every element there is an index

② Set of all odd numbers.

```
for (i=0 to ∞)
{
    generate ('2i'+1)
}
```

⟸ ①. enumeration method.

1, 3, 5, 7, 9, 11, ...
↓ ↓ ↓ ↓ ↓ ↓
1 2 3 4 5 6

⟸ ②. for every element there is an index

③. set of all real nos

Using enumeration we cannot generate all the nos

0 — 1 ⇒ 0.1
0.01
0.001
⋮

∴ Uncountable.

④. "P/q", P, q ∈ Z_+ ; Set of all quotients are countable.

↓ No. of this form are countable or not.

S = { 1/1, 1/2, 2/1, 1/3, 1/4, 3/4, ... }

1/1, 1/2, 1/3, 1/4, 1/5, 1/6, ...
2/1 cannot be generated enumeration not possible for all nos

② ③ ④ ⑤

1/1, 1/2, 2/1, 1/3, 2/2, 3/1, 1/4, 2/3, 3/2, 4/1 . . . . .

index 1 2 3 4 5 6 7 8 9 10 . . . .

⑤. Set of all strings over any finite alphabet are countable.

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

$\Sigma^* = \{\epsilon, a, b, aa, ab, \ldots\}$

Proper order:

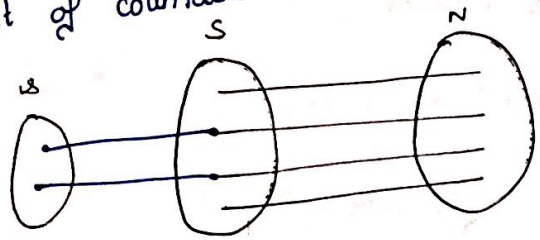
increasing order of string lengths ⇒ 0, 1, 2, 3, 4

ϵ , a, b , aa, ab, ba, bb , aaa .... bbb, ...

use dictionary ordering

1 2 3 4 5 6 .....

Every subset of countable set is either finite or countable



By composition

S is also countable.

$L \subseteq \Sigma^*$

L: countable, Σ^*: countable

⑥. Set of all TM are countable.

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

(*1) $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, \ldots\}$ countable.

(*2) Every TM can be encoded as a string of 0's & 1's.

(*3) Set of all TM, $S \subseteq \Sigma^*$.

(*4) Every subset of countable set is finite or countable.

∴ Set of all TM are countable

⑦. Implications

⇒ Recursively

⇒ Recursive

⇒ CSL are

⇒ CFL are

⇒ Reg. Lang.

∴ TM

LBA

PDA

FA

(X) Set

There

⑧. Diagnoliza

are uncou

Σ

vector.

{a, b,

{aa,

{ϵ, b

⑦. Implications of the fact that the TM are countable.

⇒ Recursively Enumerable languages are countable.

⇒ Recursive languages are countable.

⇒ CSL are countable

⇒ CFL are countable.

⇒ Reg. lang. are countable.

∴ TM are countable
LBA are countable
PDA are countable
FA are countable

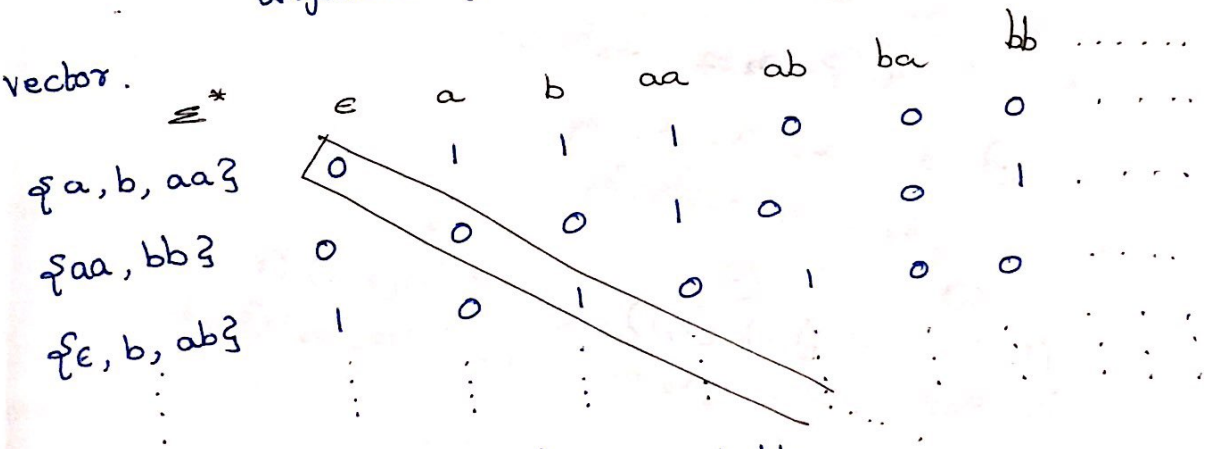
(X) Set of all languages possible ⇒ uncountable.
There are some languages which are not countable.

⑧. Diagnolization method to prove that set of all languages are uncountable.

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

Σ^* is countable, 2^{Σ^*} is uncountable.

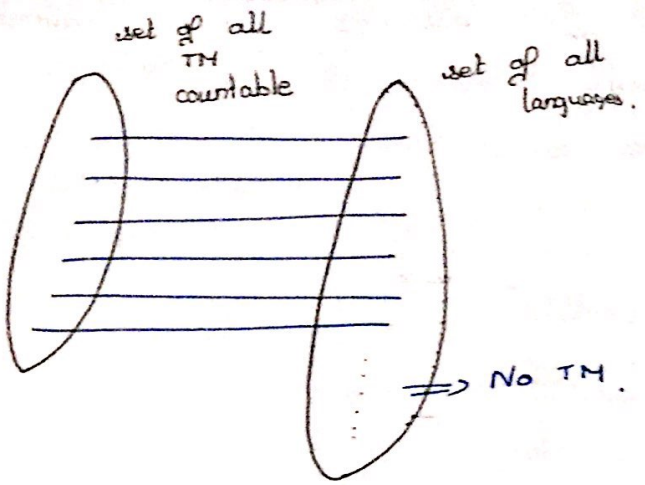
Any subset of 2^{Σ^*} can be represented in vector.



① Assume 2^{Σ^*} is countable
∴ it will have 1-1 corres. ℓ indices.

② Consider diag elements
ℓ = 001.... $\bar{\ell}$ = 110.....

③ By contradiction of the proof.



To prove countable => enumeration

To prove uncountable => diagnolisation.

① If S_1 and S_2 are countable sets, then $S_1 \cup S_2$ is countable and $S_1 \times S_2$ is countable.

② The cartesian product of finite no. of countable sets is countable.

③ The set of all languages that are not recursively enumerable is uncountable.

Ex: ① S_1 : a_1, a_2, a_3, a_4 → Enum of S_1 one step.

S_2 : b_1, b_2, b_3, b_4 → Enum of S_2 one step.

a_1b_1, a_2b_2, a_3b_3 ....

② S_1 : a_1, a_2, a_3, a_4

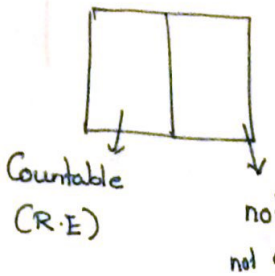
S_2 : b_1, b_2, b_3, b_4.

② (1, 1) → (a_1, b_1)

③ (1, 2) (2, 1) → $(a_1b_2)(a_2b_1)$

④ (1, 3) (2, 2) (3, 1) → $(a_1, b_3)(a_2, b_2)(a_3, b_3)$

set of all lang. 2^{Σ^*}



But

RE is countable

Assume $\overline{RE}$ is countable

$\therefore$ $RE \cup \overline{RE}$ is countable

$\Downarrow$

2^{Σ^*} is not countable

$\therefore$ $\overline{RE}$ is not countable

Computability and Decidability:

- Algorithm is a TM which will halt.
- Only halting TM are considered as algorithms.

TM are countable.

$\therefore$ HTM are also countable.

$\therefore$ Algorithm are also countable.

- There are some problem for which algorithm does not exist.

Computability

$f(n) = n^2 + 1$

if n is given as i/p then f(n) will be calculated

if there exist an alg or TM (halts) for given function then it is computable.

TM is able to solve the function for all i/ps from the domain and it will halt then such function is called computable.

Decidability

Problem:

Stmt will be true or false.

Ex: If 'n' a prime no.

D = set of all natural

Single instance is always decidable.

Ex: A given grammar 'G' is ambiguous. } Undecidable

D: set of all CFGs.

If Domain is "G_1" ⟹ Decidable

(= single grammar)

Given any particular instance of a problem, is always decidable.

2). Reducability:

If there is a problem P_1 which is converted to P_2 using some conversion algorithm and there exists an algorithm to solve P_2, it is as good as having alg. for P_1.

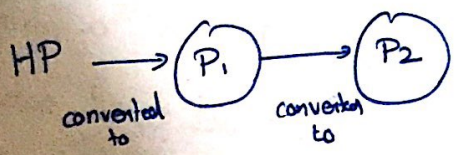
∴ If P_2 is decidable then P_1 is decidable.

3).

If the problem P_1 is already proven that it is undecidable and there exists an alg. to solve P_2 then P_2 must be undecidable.

∴

Ex: Halting Problem ⇒ Undecidable



P_1 & P_2 are undecidable.

TM Halting Problem

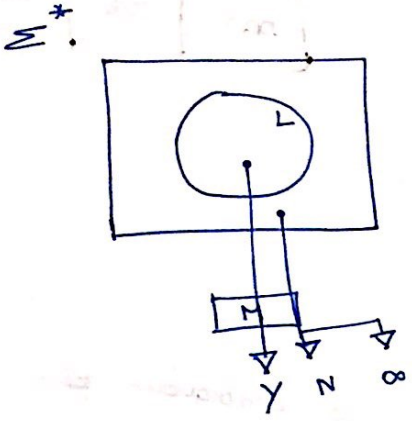
Given the description of a TM 'm' and an input 'w', does 'm' when started with 'w' as its input eventually halts?

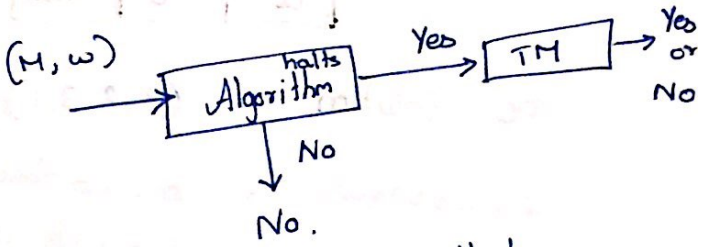
Theorem: If the halting problem were decidable, then every Recur. Enumerable language would be recursive.

Consequently, the halting problem is undecidable.

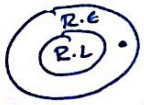
· Let L be Recu. Enu. Lang.

TM be M





We already know that



∴ Halting problem is undecidable.

Undecidable problems based on TM halting problem:

* The state entry problem is given a TM, a state $q \in Q$ and $w \in \Sigma^+$, decide whether or not the state 'q' is ever entered when M is applied to 'w'. This is undecidable.
* Given a TM M, whether or not M halts if started with a blank tape. This is undecidable.
* Almost any problem related to Recursively Enumerable language is undecidable.

Post correspondence problem

PC pbm is undecidable

Given a sequence of 'n' strings on some alphabet Σ. say $A = \omega_1, \omega_2, \ldots \omega_n$ and $B = v_1 v_2 \ldots v_n$ we say there exists a PC solution for pair (A,B) if there is a non empty sequence of integers $i, j \ldots k$ such that $\omega_i \omega_j \ldots \omega_k = v_i v_j \ldots v_k$

PC problem is to device an algorithm that will tell us for any (A,B), whether or not there exist a PC solution.

① Ex A =

ω_1	ω_2	ω_3
a	ab	bba

B =

v_1	v_2	v_3
baa	aa	bb

PC Solution is (3,2,3,1)

$\omega_3 \omega_2 \omega_3 \omega_1$ = bbaabbbaa

$v_3 v_2 v_3 v_1$ = bbaabbbaa

PC problem can be related to ambiguous pbm.

∴ for ex.,.

$\omega_3 \omega_2 \omega_3 \omega_1$ = bbaabbbaa

$v_3 v_2 v_3 v_1$ = bbaabbbaa

} ⇒ strings derived in 2 diff ways

Variables

② Let $\Sigma = \{0, 1\}$. Let X and Y be lists of three strings each, defined as follows:

	List X	List Y
i	ω_i	v_i
1	1	111
2	10111	10
3	10	0

PCP has a solution,

PC solution is $(2, 1, 1, 3)$

$w_2 w_1 w_1 w_3 = 10111 | | 10$

$v_2 v_1 v_1 v_3 = 10 | | | 1 | | 10$

③. Let $\Sigma = \{0, 1\}$. Let X and Y be lists of 3 strings as follows:

	List X	List Y
i	w_i	v_i
1	10	101
2	011	11
3	101	011

PC solution cannot start with 2 & 3

because $w_2 = 011$ & $w_3 = 101$

$v_2 = 11$ $w_3 = 011$

Now consider ①.

1 should be combined with 3 only.

because $w_1 \Rightarrow 10$ $v_1 \Rightarrow 101$

$w_3 \Rightarrow 101$ $v_3 \Rightarrow 011$

In this case $w_1 w_3 \Rightarrow 10101$

$v_1 v_3 \Rightarrow 10101 1$

1 variable extra this continue

∴ There is no PC solution.

Decidability:

	Reg lan	DCFL	CFL	CSL	Rec lang	REL
①. Does $w \in L$? (membership pbm)	D	D	D	D	D	UD
②. Is $L = \phi$? (Emptiness pbm)	D	D	D	UD	UD	UD
③. Is $L = \Sigma^*$? (Completeness pbm)	D	UD	UD	UD	UD	UD
④. Is $L_1 = L_2$? (Equality pbm)	D	UD	UD	UD	UD	UD
⑤. Is $L_1 \subseteq L_2$? (subset pbm)	D	UD	UD	UD	UD	UD
⑥. $L_1 \cap L_2 = \phi$	D	UD	UD	UD	UD	UD
⑦. Is L is finite or not? (finiteness)	D	D	D	UD	UD	UD
⑧. Is complement of 'L' a language of same type or not?	D	D	UD	D	D	UD
⑨. Is intersection of 2 languages of same type	D	UD	UD	UD	UD	UD
⑩. Is L regular language	D	D	UD	UD	UD	UD

D => decidable UD => Undecidable

Chomsky Hierarchy

	Languages	Grammar	Machine	Example
Type 0	Recursively Enumerable & Recursive	Unrestricted	Turing machine	Any compu function
Type 1	CSL	CSG	LBA	$a^n b^n c^$
Type 2	CFL	CFG	PDA	$a^n b^n$
Type 3	RL	Reg.G	FA	a^n $n \geq$

Type 0:

$$\alpha \to \beta$$

$$\alpha, \beta \in (V \cup T)^*$$

Type 1:

$$\alpha \to \beta$$

$$\alpha, \beta \in (V \cup T)^* \ \&$$

$$|\alpha| \leq |\beta|$$

Type 2:

$$\alpha \to \beta$$

$$\alpha \in V \quad \& \quad \beta \in (V \cup T)^*$$

Type 3:

$$\alpha \to a\beta / a$$

$$\alpha, \beta \in V \quad \& \quad a \in T \cup \{\epsilon\}$$

Linear Bounded Automaton:

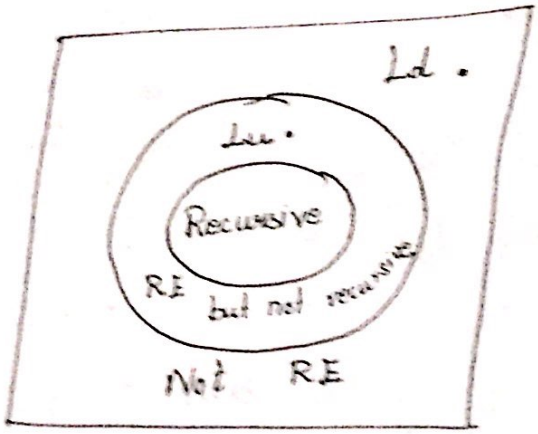
A linear bounded automaton is a non-deterministic TM $M = (Q, \Sigma, \delta, q_0, F, \Gamma, B)$ subject to the restriction that Σ must contain two special symbols [and] such that $\delta(q_i, [\,)$ can contain only elements of the form $(q_j, [, R)$ and $\delta(q_i,])$ can contain only elements of the form $(q_j,], L)$.

Context Sensitive Languages:

A string is accepted by a linear bounded automata if there is a possible sequence of moves

$$q_0[w] \vdash^* [x_1 q_f x_2]$$

for some $q_f \in F$, $x_1, x_2 \in \Gamma^*$. The language accepted by the lba is the set of all such accepted strings.



Language Ld:

This language is the set of strings of 0's and 1's that when interpreted as a TM, are not in the language of that TM. The language Ld is a good example of a language that is not RE; i.e., no TM accepts it.

Universal Language: (Undecidable)

The language Lu consists of strings that are interpreted as a TM followed by an input for that TM. The string is in Lu if the TM accepts that input. Lu is a good example of a language that is RE but not recursive.

Counter Machines:

The counter machine is a restricted multi-stack machine. Each stack in the machine is replaced by a counter. Each counter holds a non-negative number.

The move of a machine depends on the current state, current i/p symbol and if one of the value of the counter is zero.

The machine can do the following activities in one move:

(a). It can change the state.

(b). It can add or subtract 1 from the counters. But, if the counter is zero, subtracting 1 from that counter is not possible.

Languages accepted by counter machine:

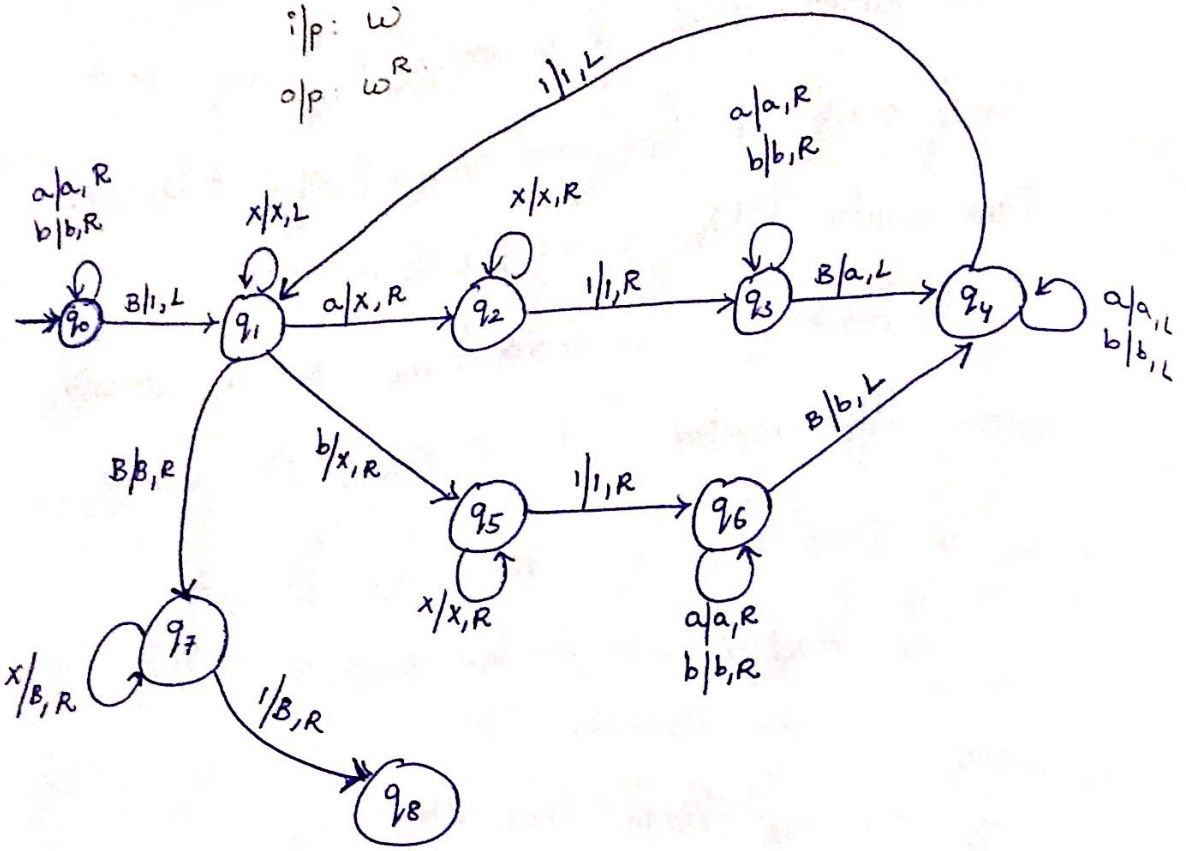
(1). The languages accepted by counter machines are recursively enumerable i.e., we can obtain an equivalent TM to accept those languages which are accepted by counter machines.

(2). A language accepted by only one counter machine is CFL.

① TM for reversing a string.

i/p : w

o/p : w^R



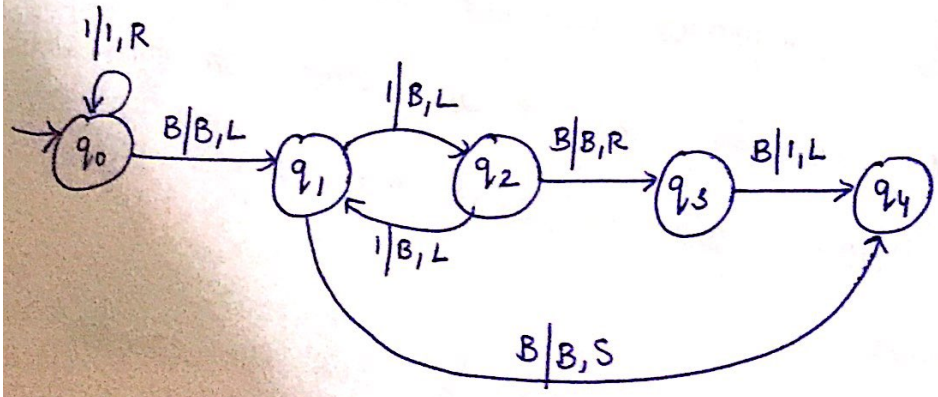
② TM for n mod 2

i/p : n

o/p : 1 for n → odd

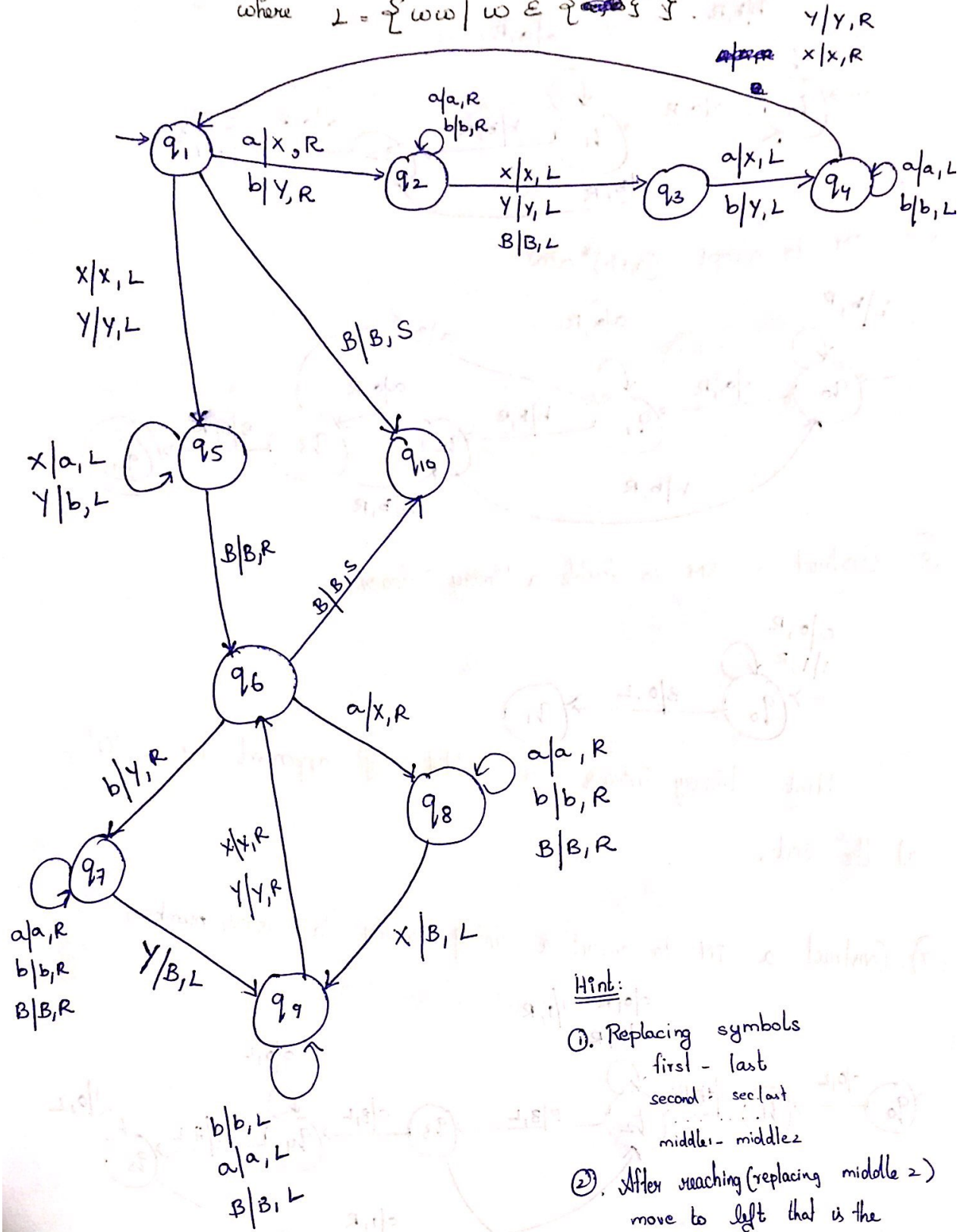
0 for n → even

⇒ (B in TM)



③. TM as a language acceptor

where $L = \{ww \mid w \in \{a,b\}^*\}$.



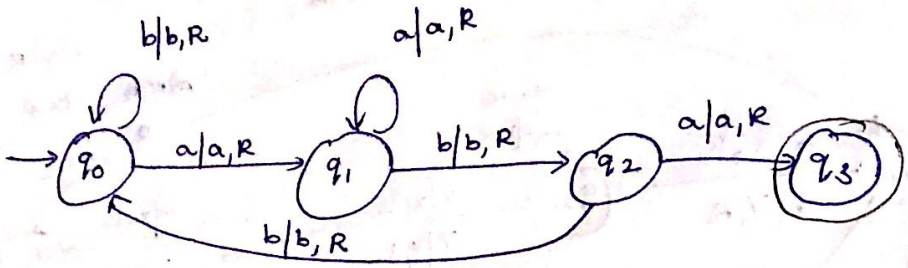
Hint:

① Replacing symbols
first - last
second - second last
middle1 - middle2

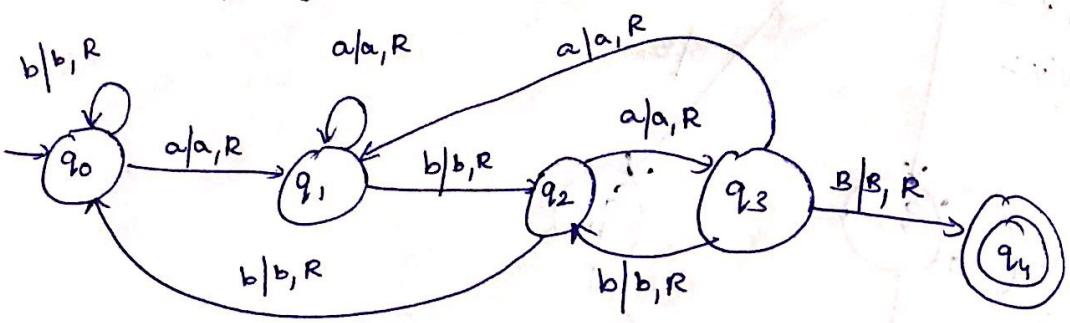
② After reaching (replacing middle 2) move to left that is the last symbol of w

③. Change 1st w symbols & cross check with 2nd w symbols.

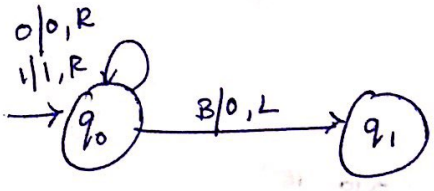
(4). TM to accept $(a+b)^* aba (a+b)^*$



(5). TM to accept $(a+b)^* aba$

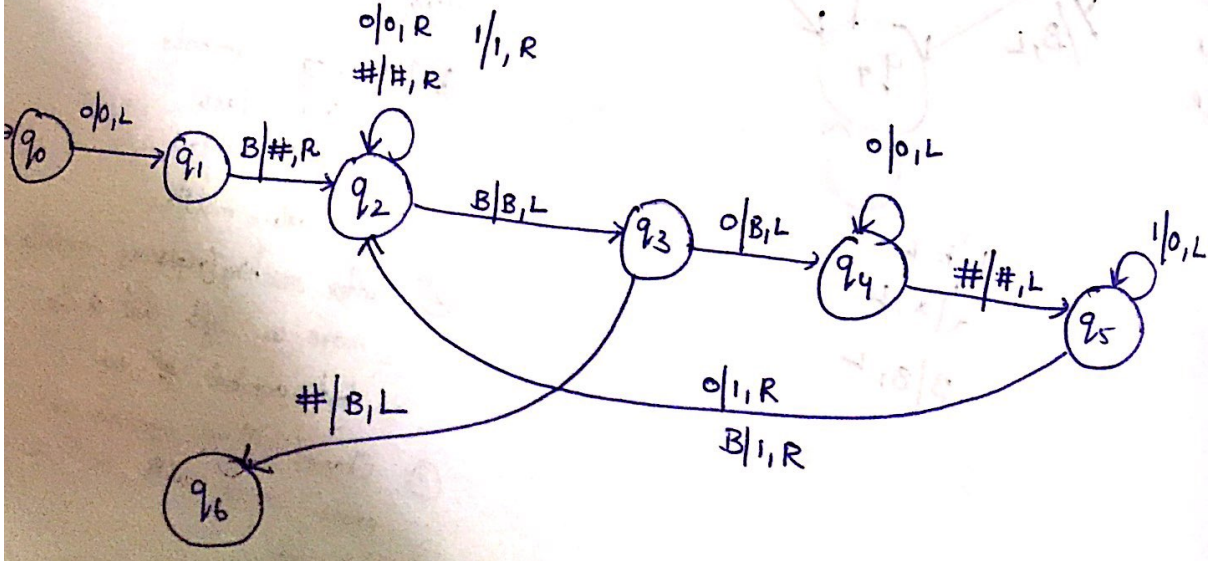


(6). Construct a TM to double a binary integer.



Hint: binary integer gets doubled if appended by a '0' at the end.

(7). Construct a TM to convert a unary number to binary number.



⑧. Construct a TM to compute 2^n (in binary representation)

i/p : n in binary

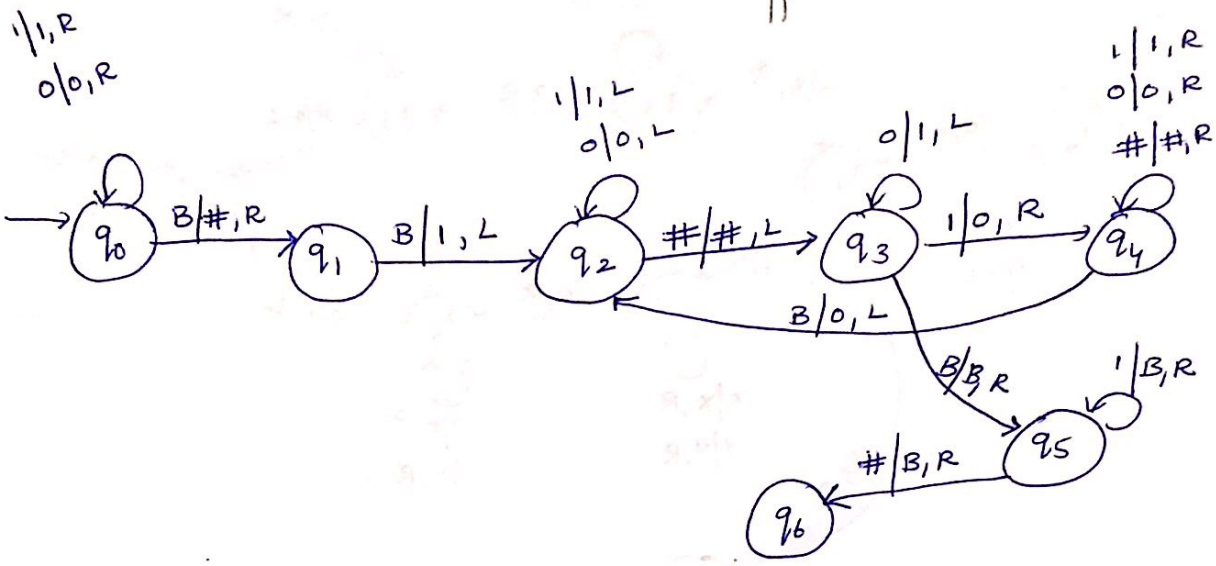
o/p : 2^n in binary.

Ex ① i/p : 10 $\Rightarrow$ (2)

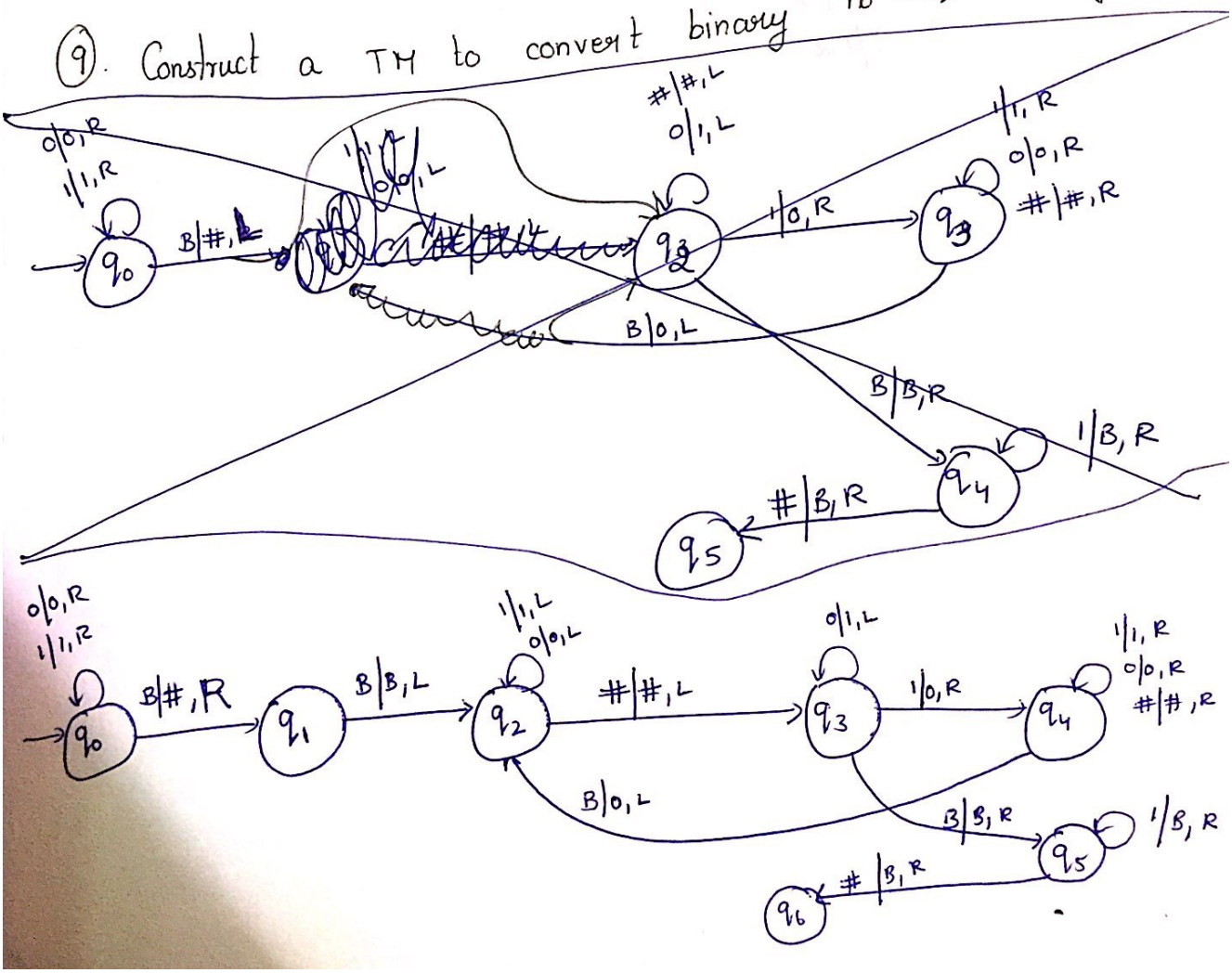
o/p : 100 $\Rightarrow$ (2^2)

② i/p : 100 $\Rightarrow$ (4)

o/p : 10000 $\Rightarrow$ (2^4)



⑨. Construct a TM to convert binary no $\longrightarrow$ unary.



(10) Design a TM such that $(q_0, BwB) \vdash^* (q_f, BwBw^rB)$ where q_0 is the initial state, q_f is the final state, B is blank and $w \in \{0,1\}^*$.