(2) $L = \{a^i b^j c^i d^j \mid i,j \geq 1\}$

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

Turing Machine

Perfect number :- An ie a perfect number if it the sum of its divisors [except for itself]

$6 = 1+2+3$

$28 = 1+2+4+7+14$.

Q) if there is Perfect number > 0.

Algorithm :- which have finiteness property.

Procedure :- which doesn't have finiteness property.

Turing Machine gives only procedure but not Algorithm.

Defn of TM :-

$$(Q, \Sigma, \Gamma, \delta, q_0, \not b, F)$$

$Q \longrightarrow$ finite no. of states

$\Sigma \longrightarrow$ Input alphabet.

$\Gamma \longrightarrow$ Tape alphabet $[\Sigma \subseteq T]$

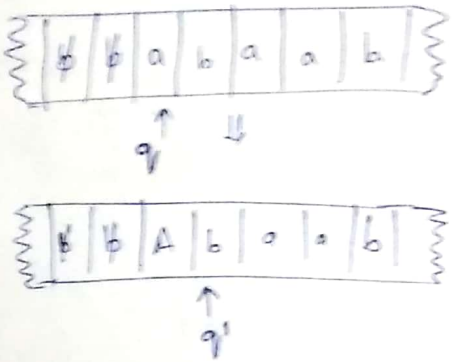
$q_0 \longrightarrow$ Start state.

ϕ → blank symbol.

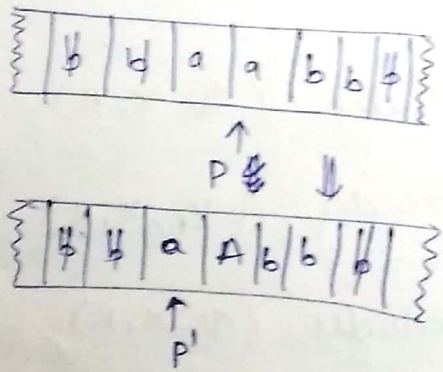
F → set of finite states $\subseteq Q$.

δ : $Q \times \Gamma \rightarrow {}_2^{Q \times \Gamma \times \{L, R\}}$.

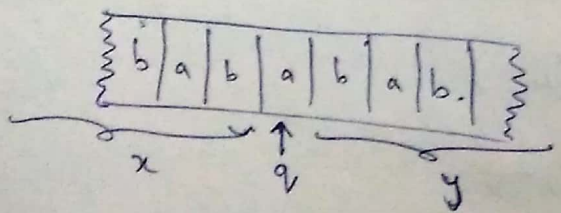
$\delta(q, a) = \{(q', A, R)\}$



$\delta(p, a) = \{(p', A, L)\}$



Instantaneous Discription :-



x → string contained left side of q

y → string contained Right side of q.

ID :- $xqay$.

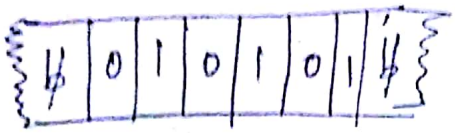
Ex. $abqaba. \Rightarrow ab A q' b a.$ [if q move right rewriting a → A]

Turing Machine can be thought in two ways

(1) Accepting Device

(2) Input output Device

Parity checker



$\delta(q_1, \#) \Rightarrow$ odd.

$\delta(q_0, \#) \Rightarrow$ even.

$\delta(q_0, 0)$ includes (q_0, X, R).

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

$\delta(q_1, 0)$ includes (q_1, X, R)

$\delta(q_1, 1)$ includes (q_0, X, R).

$\delta(q_1, \#)$ includes $(q_{Halt}, 0, -)$

$\delta(q_0, \#)$ includes $(Halt, E, -)$.

Balanced Parenthesis

(( ))( )( ).

Check whether parenthesis are balanced or not.

First search for a closed parenthesis [)], rewrite it with X and check for open parenthesis [move left] and again rewrite it with X. Repeat this step until all the parenthesis are balanced. If they are not balanced then don't accept the string.

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

$\delta(q_0,)) = (q_{01}, X, L)$

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

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

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

$\delta(q_0, \$) = (q_2, \$, L)$

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

$\delta(q_2, () = (Halt, N, -)$

$\delta(q_2, \$) = (Halt, Y, -)$

$\delta(q_1, \$) = (Halt, N, -)$

Example-3:- (Method -1)

unary number. ⇒ binary number.

⇒ b a a b.

where b = 1
a = 0.

Mark alternate 1's as X, see if there are odd 1's
hence, come back to the first ⊔ and mark B or else A.
repeat it.

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

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

$\delta(q_1, \sqcup) = (q_2, \sqcup, L)$

$\delta(q_2, 1/X) = (q_2, 1/X, L) \Rightarrow \delta(q_2, 1/X/A/B) = (q_2, 1/X/A/B, L)$

$\delta(q_2, \sqcup) = (q_4, B, R)$

$\delta(q_0, \sqcup) = (q_3, \sqcup, L)$

$\delta(q_3, 1/X) = (q_3, 1/X, L) \Rightarrow \delta(q_3, 1/X/A/B) = (q_3, 1/X/A/B, L)$

$\delta(q_3, \sqcup) = (q_4, A, R)$

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

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

$\delta(q_4, \sqcup) = (\text{Halt}, \sqcup, -)$

$\delta(q_4, 1) = (q_1, X, R)$

↳ binary counter.

$\delta(q_0, 1) = \{(q_1, X, L)\}$

$\delta(q_1, b) = \{(q_0, B, R)\}$

$\delta(q_0, X/A/B) = \{(q_0, X/A/B, R)\}$.

$\delta(q_1, X) = \{(q_1, X, L)\}$.

$\delta(q_1, B) = \{(q_1, A, L)\}$

~~Anonymous~~.

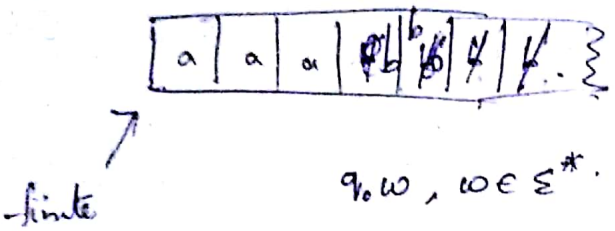
$\delta(q_1, A) = \{(q_0, B, R)\}$.

$\delta(q_0, b) = \{(\text{Halt}, -, -)\}$.

Time complexity of method 1 ≤ Time complexity of method 2

space complexity of method 1 ≥ space complexity of Method 2.

Turing machine as an accepting device :- $(Q, \Sigma, \Gamma, \delta, q_0, \#, F)$



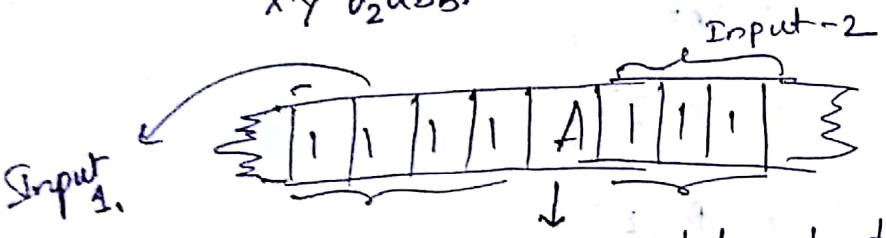
ID :- q_0 aaa bb.

$\alpha_1 q \alpha_2$

$X q_1$ aa b b.

$\alpha_1, \alpha_2 \in \Gamma^*$.

$XY q_2$ abb.



Language Acceptance :-

$L(M) = \{ w \in \Sigma^* / q_0 w \vdash^* \alpha_1 q_f \alpha_2 \}$

where $q_f \in F$

$\alpha_1, \alpha_2 \in \Gamma^*$.

Type-0 Language or recursively enumerable language :-

Set of all language accepted by any Turing machine.

Regular Language — Type-3 Language.

CFL — Type-2 Language

CSL [Context Sensitive lang] - Type-1 Language.

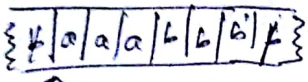
Recursively Enumerable - Type-0 Language.

} Chomskian Hierarchy of Languages.

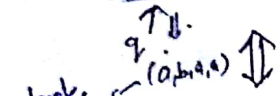
Feature of Turing Machine :-

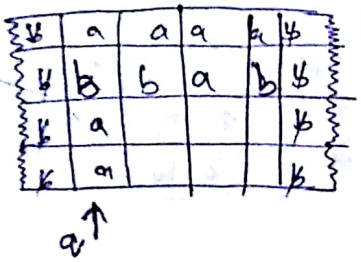
(1) Store up in the finite control (state)

(2) Store up in the tape ~~alphabet~~ using tape alphabet.



→ single track.



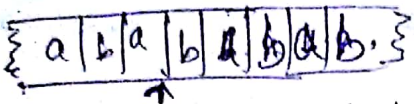


→ multiple track

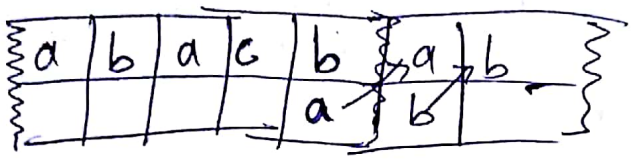
$$\delta(q, a, b, a, a) = (q', A, B, A', A'')$$

Multiple → single ⇒ whatever the input of multiple TM at a time can be accomodated in a n tuple alphabet of single TM.

3) Tracks in a tape doesn't matter.



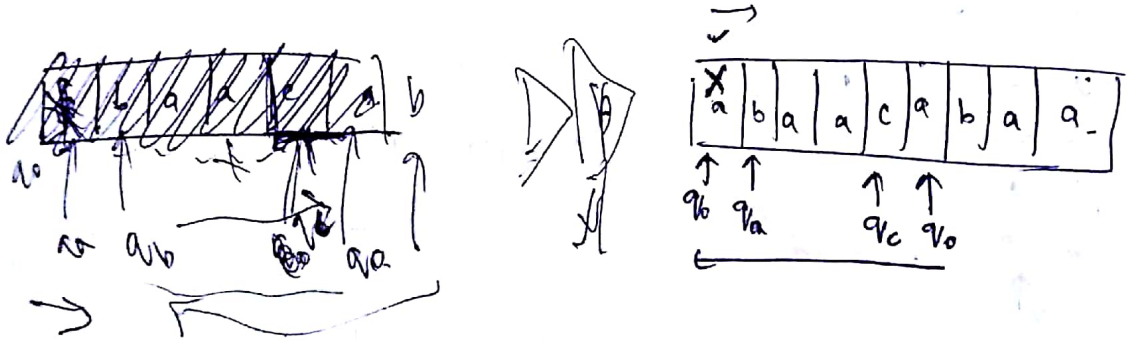
we need to insert c, between a & b, for this we need to traverse along the input and shift every symbol, rather than that we can use Multiple track TM.



(4) Shifting of Symbols.

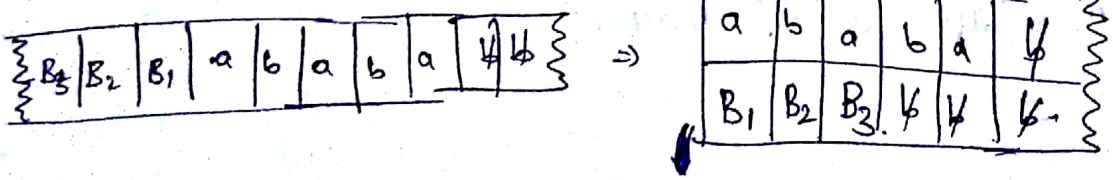
(5) checking of Symbols.

$L = \{wcw \mid w \in \{a,b\}^*\}$

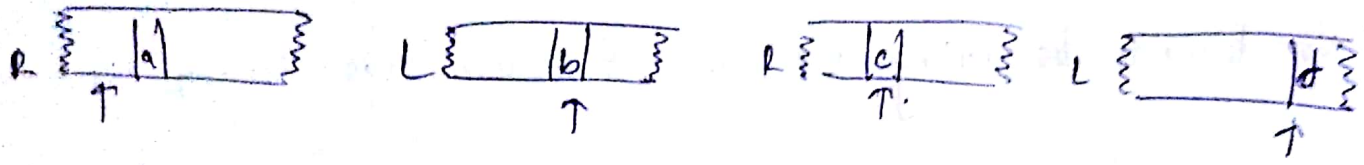


Generalised Turing Machine Models :-

1) 



2) Multitape Turing Machine

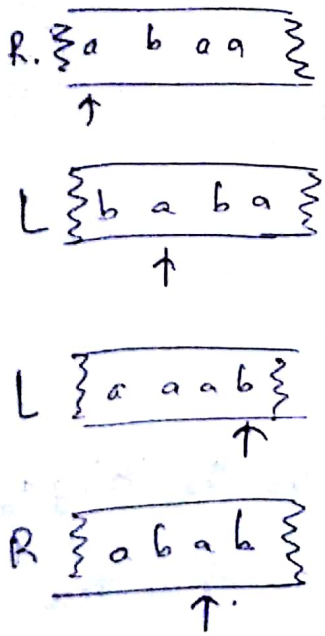


Multiple tapes, Multiple heads.

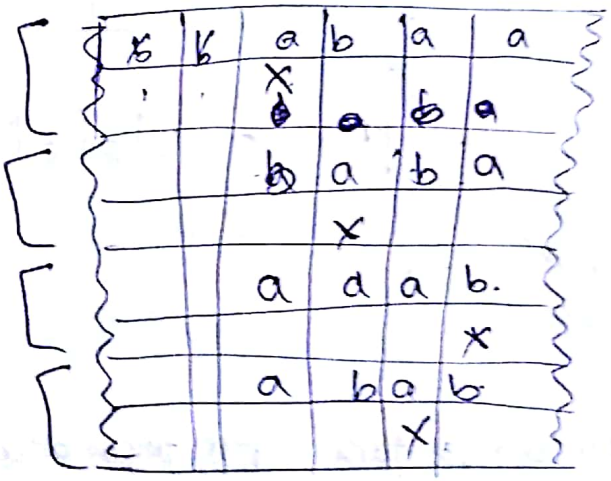
Each head is independent of its movement.

$$\delta(q, a, b, c, d) = (q', A, B, C, D, R, L, R, L)$$

Multiple Tape TM $\Leftrightarrow$ Multitrack TM.



→



$\delta(q, \underline{a}\,\underline{a}\,\underline{b}\,\underline{a}, \$)$

$\delta(q, a, a, b, a) \rightarrow (q', A, A', B, A'', R, L, L, R)$

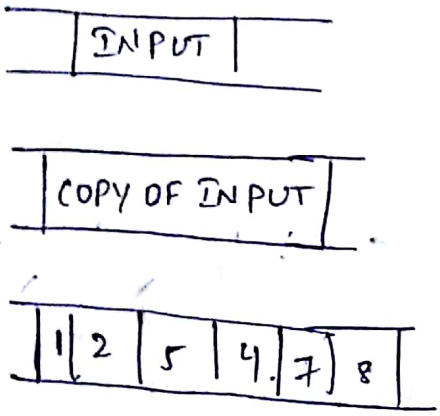
After traversing from left to right, it knows the input, now from right to left it checks for X and replace the symbol desired and the x is moved either left or right

at i^{th} move almost 4^i moves can be done.

(3) Non deterministic Turing machine $\Leftrightarrow$ Deterministic Turing Machine.

$\delta(q,a) = \{(q', B, R), (q'', A, L), (q''', C, L)\}$
(1) (2) (3)

$\delta(P,b) = \{(P', B, R), (P'', A, L)\}$
(1) (2)

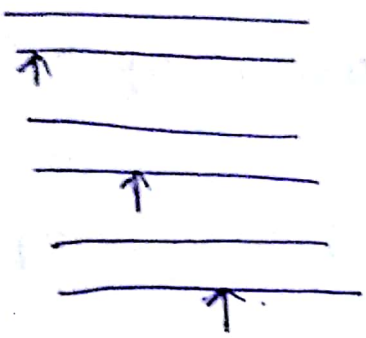


a string containing which element should be selected.

As we proceed, to take input from a copy of it, if the the string of number lead us to final state, hence it fine or else we erase the number string & we will use another one and again make a copy of Input and start over the process again until final state is reached.

The above TM is a multitape $\Leftrightarrow$ Multitrack $\Rightarrow$ single track.

Generative Model of TM:-



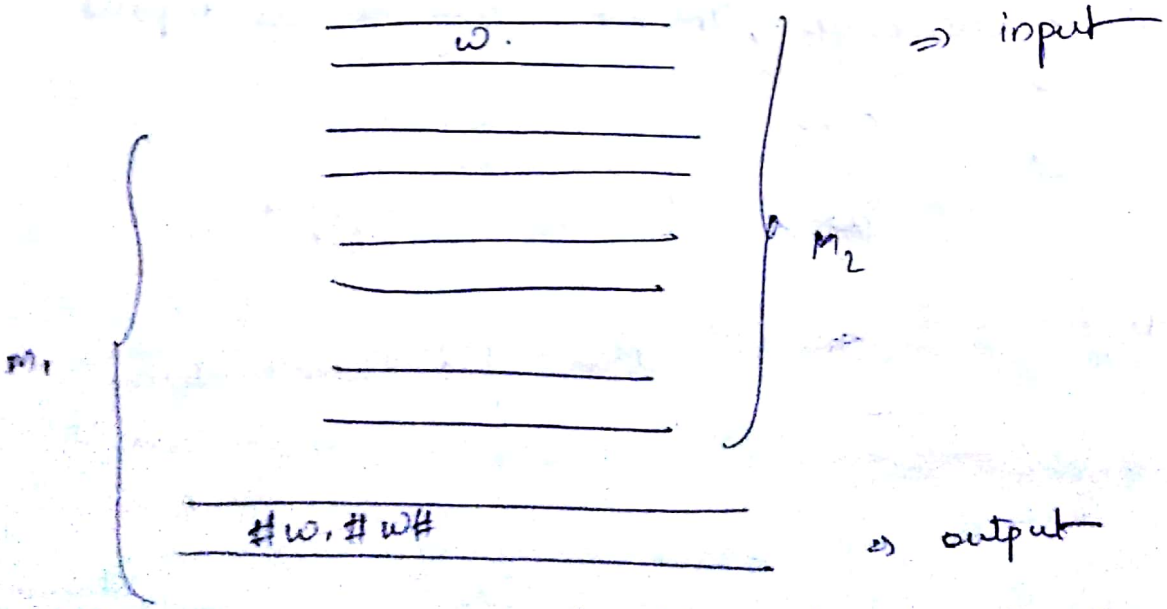
$\#w_1\#w_2\#w_3\#$ ⟶ output tape.

we write all the strings generated by stapes.

1) M_1 $G(M_1) = L$ ⟶ Generative TM

2) M_2 $L(M_2) = L$ ⟶ Accepting TM.

$1 \Rightarrow 2$

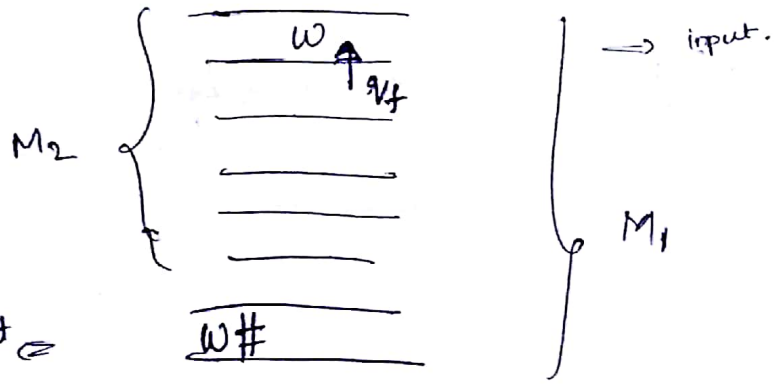


If the generated output string is equal to input string hence both TM's are equivalent.

2 ⇒ 1

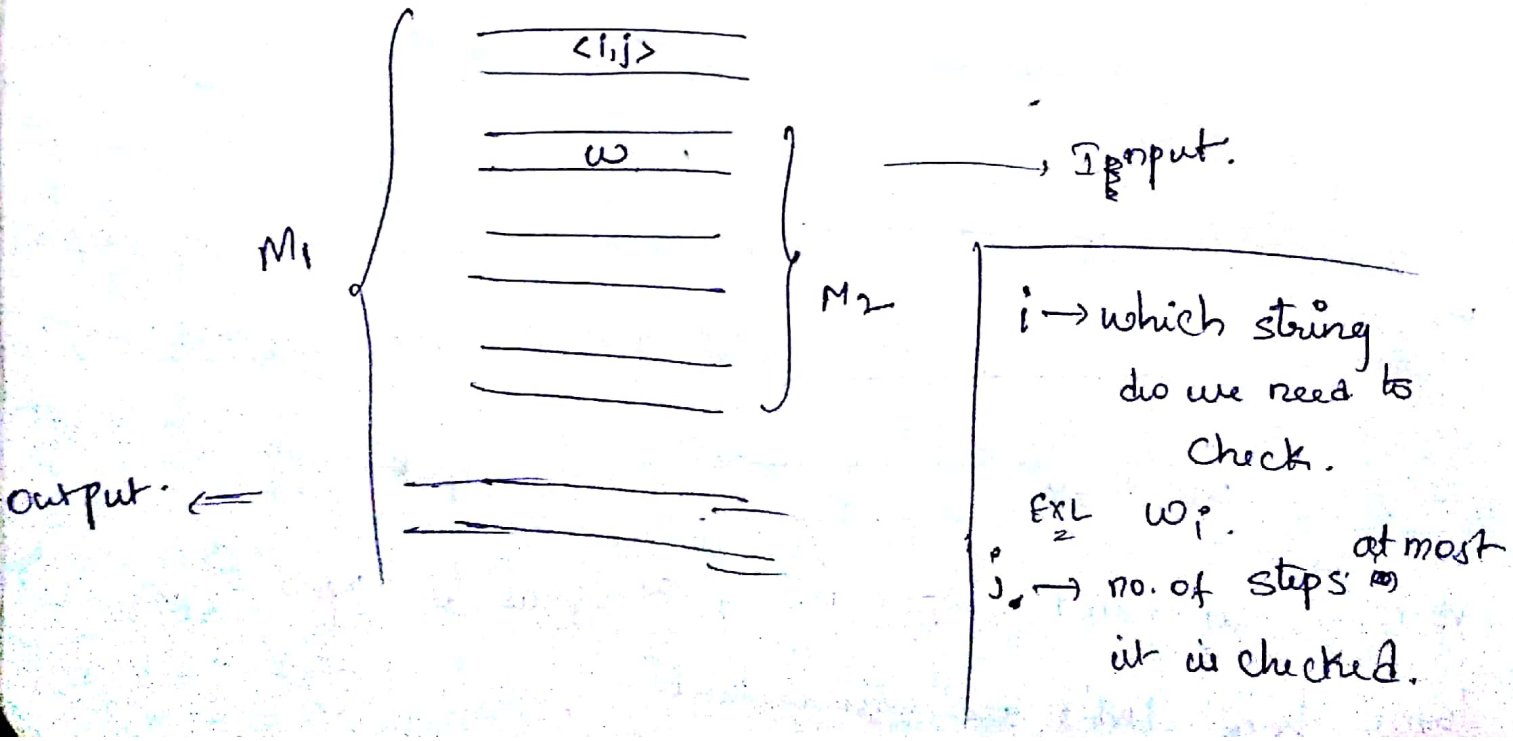
Recursive language :- A Language L is called as recursive if ∃ a TM, M s.t $L(M) = L$ & M halts on all input.

Case 1 :- recursive language, TM halts on all input.

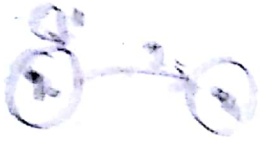


if after reading all input it going to final state, then we write that in output tape or not.

Case 2 :- L is not recursive., TM not halting on all inputs.



we will check whether TM accepts the string within j steps. If yes, then it is written in output tape, or we will generate another number $\langle i', j' \rangle$ and repeat the process.



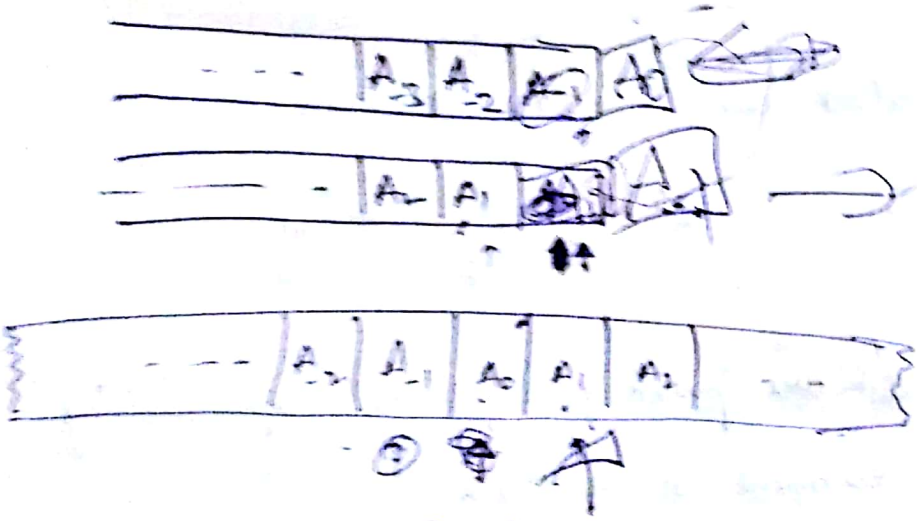
generating unary numbers.

$1 \to 1$
$11 \to 2$
$111 \to 3$

This fn helps in generating random numbers.

Equivalent versions of TM :-

1) 2-stack machine $\Leftrightarrow$ TM. [TM: 2 PD stacks]

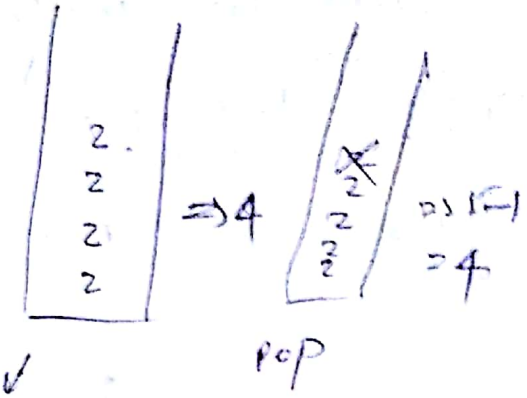


$\forall A_1 \exists q A_1$

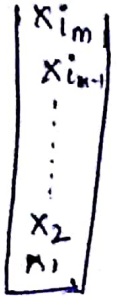
(2) 4-Counter machine $\Leftrightarrow$ TM.

↓

only one alphabet is used.



We will simulate 1 stack by 2 counters.



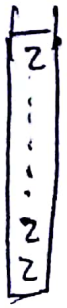
$\Gamma = \{x_1, x_2, \ldots, x_{k-1}\}$.

$i_j \in \{1, 2, \ldots, k-1\}$.

] → stack.

- for representing a stack we need a counter.

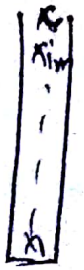
let us take $j = k^{m-1} i_1 + \cdots + k^2 i_{m-2} + k\, i_{m-1} + i_m$.



} j, So j z's are pushed into stack representing all the elements in the stack.

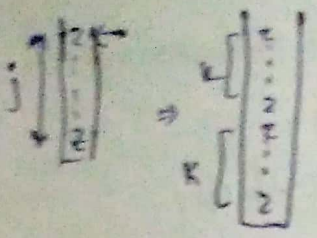
If for pushing & popping operations another counter is required.

If x_r is pushed into stack,



, then the counter 1 has j z's.

first we need to change value of j by $kj+r$ → gives a new value including kr.



each z is poped from counter 1 and pushed k times in counter 2. after each and every z, we push z r times in counter 2.

After end of pushing z's. counter 1 is Null. & counter 2 has $kj+r$

In the same way we pop, by subtracting i_m from j & dividing it by k.

$$j = \frac{j - i_m}{k}.$$

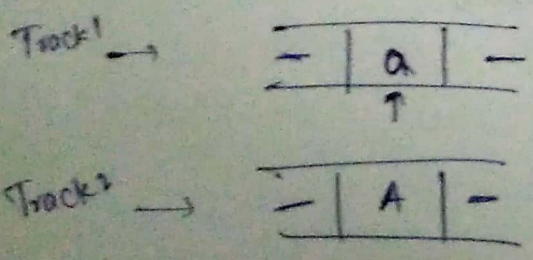
⑧ TM with $\{a,b,\not b\}$ ⟺ TM.

Let a TM with alphabets a_1, a_2, a_3, a_4 can be equivalent to TM with $\{a,b,\not b\}$ if we consider

$a_1 = aa$, $a_2 = ab$, $a_3 = ba$, $a_4 = bb$.

instead of reading a_1, we read aa.

④ TM without rewriting ⟺ TM.

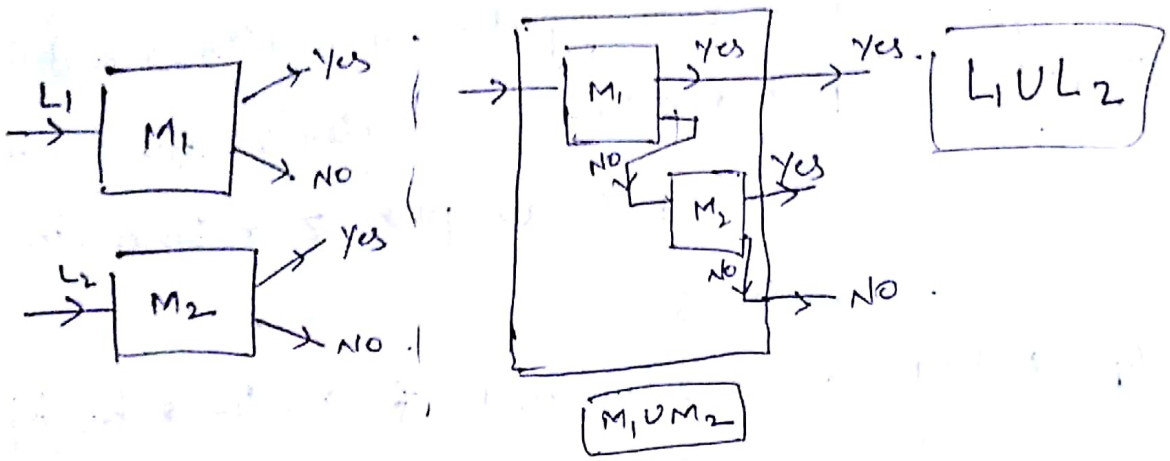


we need to replace a by A but rewriting is not allowed.

Copy left & right of alphabet into Track-2 and in place of a write A.

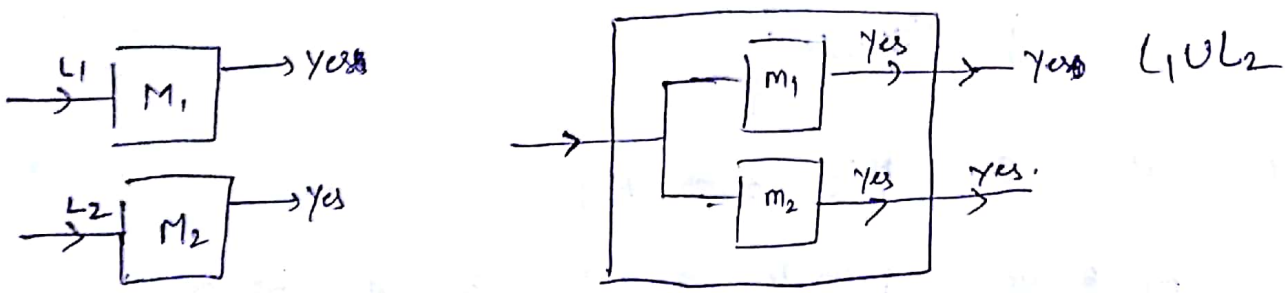
Recursive Sets :-

1) L_1, L_2 are recursive then $L_1 \cup L_2$ is also recursive



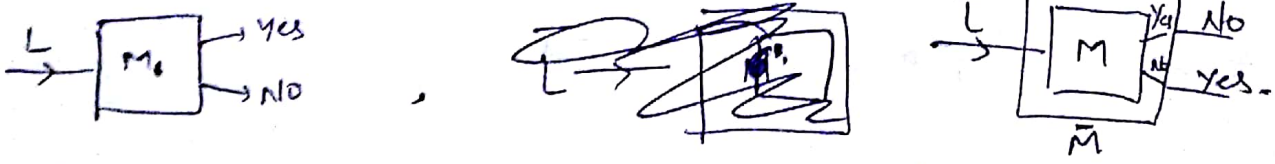
2) L_1, L_2 are Recursively Enumerable (RE). ST $L_1 \cup L_2$ is RE.

Recursive $\subseteq$ RE.

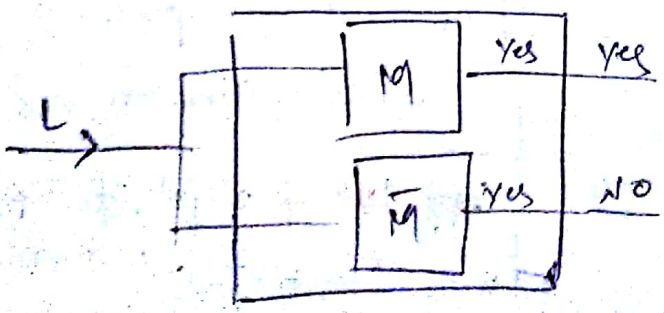


$L_1 \cup L_2$ is also RE.

3) If L is recursive, Prove that $\bar{L}$ is recursive.



4) If L_1 & $\bar{L}_2$ are RE, then L is recursive.



. this machine halts for all input

hence L is recursive.

Problem — Decision
         — General.

Instance of a problem

Sometimes the question itself becomes instance of a prob.

Decision problem

Which have Yes/No type of answer.

Decidable / undecidable

if a problem has finite no. of steps, then it is called decidable.

Returning one problem to another problem

INPUT:- G is a Graph with V_1 as one of the vertex.

Problem:- Does G have a hamiltonian cycle with V_1?

↳ output:- Yes/No

Problem:- find the Hamiltonian path in G starting with vertex V_1.

↳ output:- $V_1, V_2, V_3 \ldots V_n$.

Fermat's conjecture:-

Does ∃ +ve integers ; x, y, z & i , st $x^i + y^i = z^i$, $i > 2$

↳ undecidable

$TM = (K, \Sigma, \Gamma, \delta, q_0, \phi, F)$

$K = \{q_1, q_2, q_3\}$

$\Gamma = \{a_1, a_2\}$

We can ~~do~~ represent ~~state~~ State & alphabet in terms of 0's and 1's.

1 zero for q_1

111 0 11 00 11 000 111 0 11 00 111

delimiter

2 zeros for q_2

2 zero for a_1

for transistions , $\delta(q_i, a_j) = \{(q_n, a_k, L/R)\}$

$11\ 0^i\ 1\ 0^j\ 1\ 0^k\ 1\ 0^l\ 1\ 0\ 1\ 00\ 11.$

L R