PL

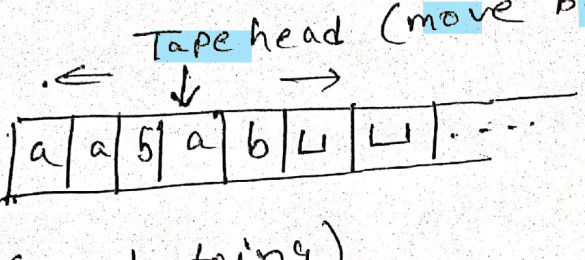
FSM : | a | a | a | a | a |

finite input string

controllen (index) → move forward only

PDA : - input string
- stack

TM :



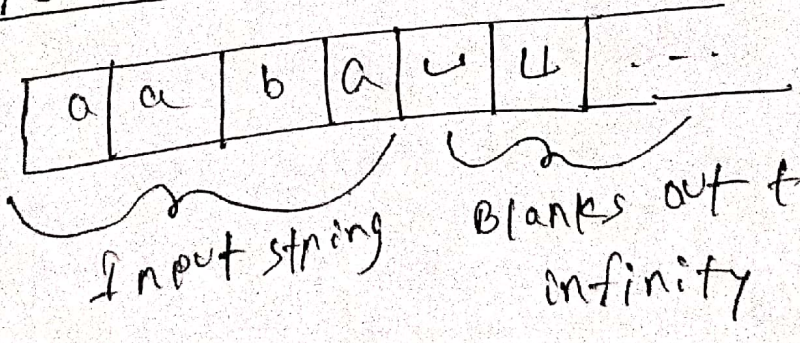
Tape (input string)
- infinite length

Tape alphabets : $\Sigma = \{0, 1, a, b, x, Z_0\}$

Blank : ⊔ $\notin \Sigma$

special symbol to fill the infinite tape

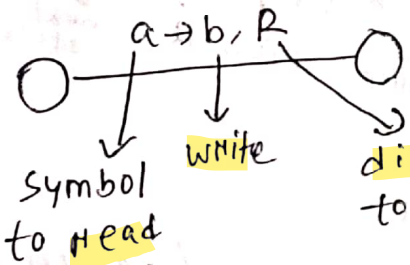
Initial configuration :



Operations:

1) Read / Scan symbol below tape head
2) update / write
3) move 1 step left
4) move 1 step right

* if at 0 index, can't move more left
  - stay at 0
* right - same



* 1→1, L
  same symbol → no update

* initial state
* final " - accept state
  reject state

* computation can
  1) halt and accept
  2) halt and reject
  3) loop (don't halt)

Formal definition: A TM can be defined as a set of 7 tuples.

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

Q — set of states

Σ — set of symbols

Γ — set of tape symbols

δ — transition function

$$Q \times \Sigma \rightarrow \Gamma \times (R/L) \times Q$$

(Q: state of FA; Σ: symbol input; Γ: tape symbol; R/L: right/left; Q: next state)

q_0 — initial state

b — blank symbol

F — set of final state (ac or rej)

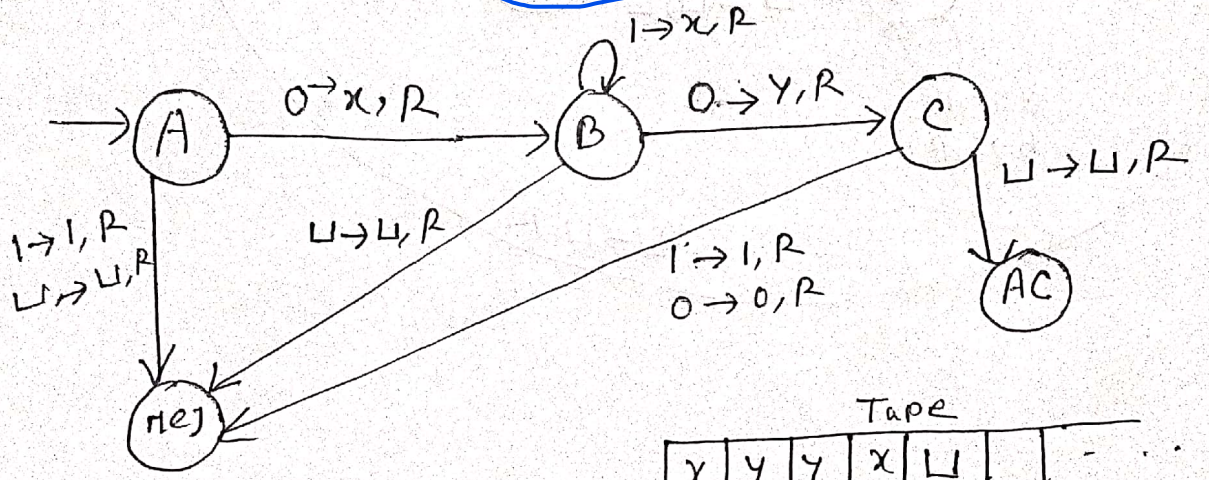
Production rule of TM

$$\delta(q_0, a) \rightarrow (q_1, Y, R)$$

(q_0: input state; a: input symbol; q_1: new state; Y: tape symbol; R: direction)

Recursively Enumerable Language: A language L and Σ is said to be REL if there exists a TM that accepts it.

Ex 1: Design a TM that recognizes the language $L = 01^*0$

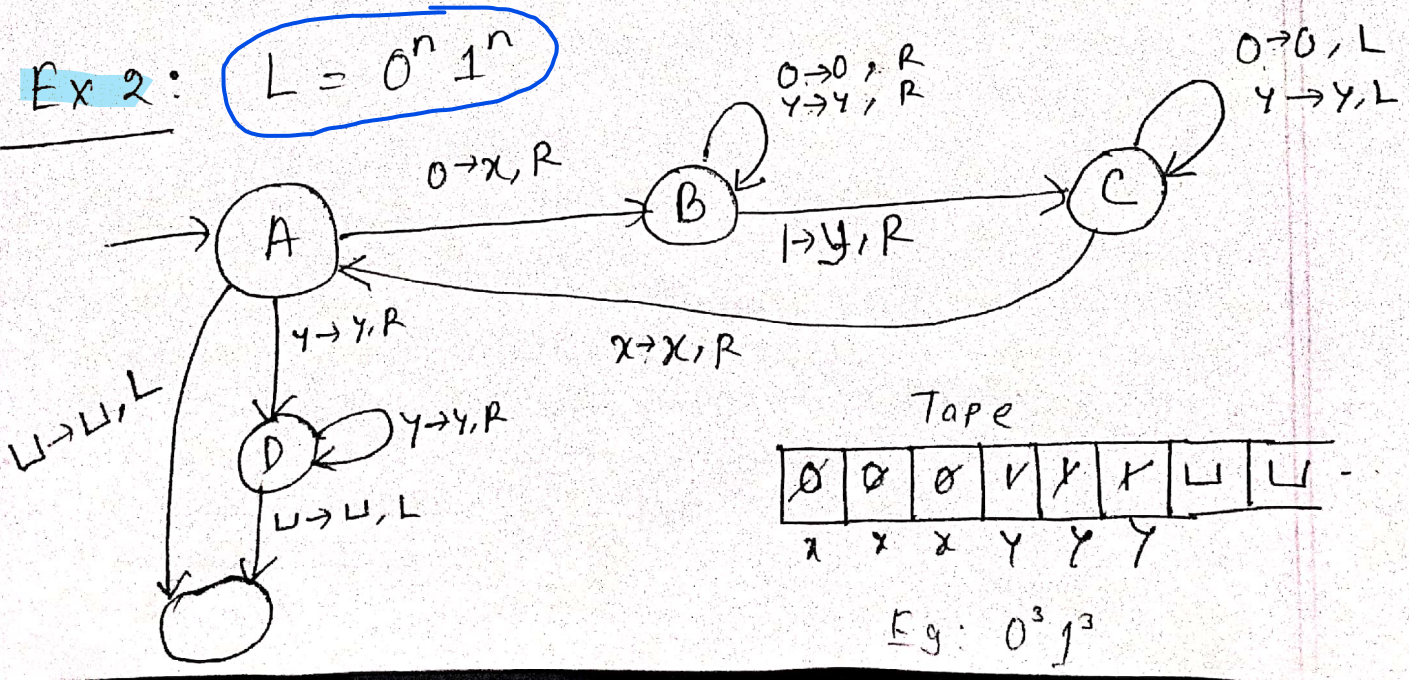


Tape

x	Y	Y	x	⊔		

Eg: 0 1 1 0

Ex 2: $L = 0^n 1^n$



Tape

0	0	0	1	1	1	⊔	⊔
x	x	x	Y	Y	Y		

Eg: $0^3 1^3$

Ex 3: Design a TM for palindrome: