Date: 19/12/2022

UNIT-5: TURING MACHINE

→ Turing machine :-
→ Definition
→ Model
→ Representation of Turing Machines
→ Instantaneous descriptions
→ Transition Tables
→ Transition Diagrams.
→ Language of a Turing Machine
→ Design of Turing Machines.
→ Techniques for Turing Machine construction
→ Types of Turing Machines
→ church's thesis
→ Universal Turing Machine
→ Restricted Turing Machine.
→ Decidable and undecidable problems :-
→ NP
→ NP-Hard
→ NP-complete problems.

Introduction to Turing Machine:

Around 1936 there was no computers, Alen Turing proposed a model of abstract machine called Turing Machine. Which could perform any computational process carried out by present day computers.

Turing machine is a machine format of unrestricted grammar, (all types of languages are accepted).

Turing machine is defined as 7-tuple representation

$$M = \{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$$

Where, Q is finite set of states
Σ is finite set of input symbols
Γ is finite set of tape symbols
δ is transitional function
q_0 is starting state
B is Blank symbol
F is Final state.

Transitional function δ mapping is

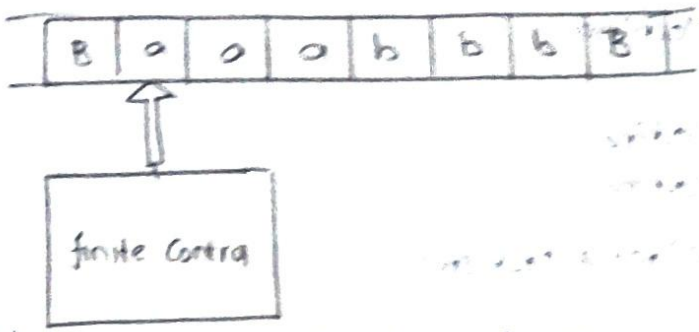
$$(Q \times \Gamma) \longrightarrow (Q \times \Gamma \times \{L, R, H\})$$

It denotes one state by getting one input from the input machine moves to a state. It writes symbol on tape and movement symbol (L, R, H) also their.

Model:

Turing machine consists of an input tape, read-write head and infinite Input tape consists input alphabets with infinite number of bla at the left and right hand side of input symbols.

the read-write head reads an input symbol from the input tape and sends it to finite control.



In finite control, the transition functions are written according to present state and present input, suitable transitional function is executed and perform the following operations.

1. Machine goes into some state
2. Machine writes symbol in input tape where input symbol was scanned.
3. The machine moves reading head to left (or) right (or) halts.

In Turing machine, string is accepted when total string is traversed and the machine halts. (after reaching final state).

Graphical Notation (Transition Diagram):

Turing machine is defined as 7 tuple representation

$$M = \{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$$

where transition function δ consists of two tuples;

Present state and tape symbol (Input symbol) which generates the next state, tape symbol after read and write and movement symbol left (or) right (or) halt.

In Graphical notation, beginning state is represented as → (q_1)

final state is represented as $((q_f))$

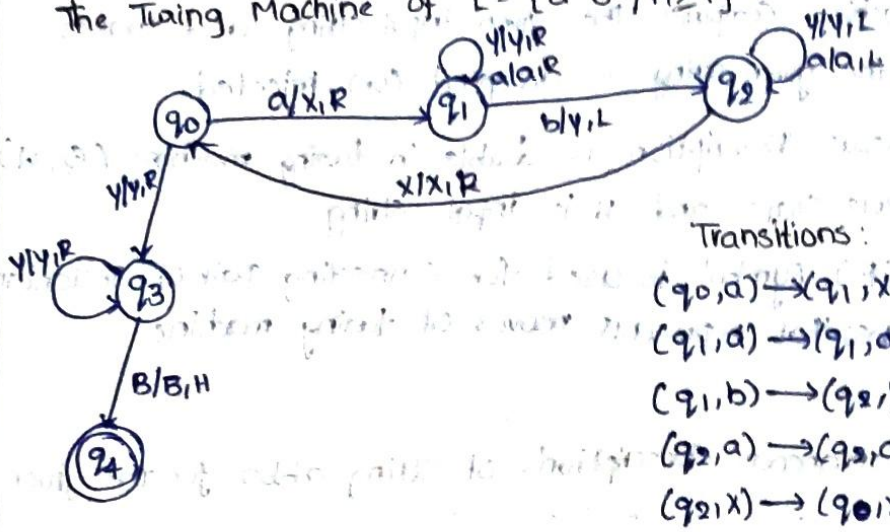
transition can be represented as →

Labels of transition consists of tape symbols before transition

tape symbol after transition and movement symbol are L (or) R (or) H.

Example:

The Turing Machine of $L = \{a^n b^n / n \geq 1\}$



Transitions:

$(q_0, a) \rightarrow (q_1, X, R)$
$(q_1, a) \rightarrow (q_1, a, R)$
$(q_1, b) \rightarrow (q_2, Y, L)$
$(q_2, a) \rightarrow (q_2, a, L)$
$(q_2, X) \rightarrow (q_0, X, R)$
$(q_1, Y) \rightarrow (q_1, Y, R)$
$(q_2, Y) \rightarrow (q_2, Y, R)$
$(q_0, Y) \rightarrow (q_0, Y, R)$
$(q_3, Y) \rightarrow (q_3, Y, R)$
$(q_3, B) \rightarrow (q_4, B, H)$

Tabular Notation (Table of transition):

In tabular representation,
beginning state is represented as $\rightarrow q$.
final state is represented as (q).

Transition can be filled in table based on state and input symbol.

Transition Table for language $L = \{a^n b^n, n \geq 1\}$.

	a	b	X	Y	B
q_0	(q_1, X, R)			(q_3, Y, R)	
q_1	(q_1, a, R)	(q_2, Y, L)		(q_1, Y, R)	
q_2	(q_2, a, L)		(q_0, X, R)	(q_2, Y, R)	
q_3				(q_3, Y, R)	(q_4, B, H)
q_4					

Instantaneous Description:

Instantaneous Description is called Informal notation and explains how turing machine computes the given input string and makes decisions that the given string is accepted (or), rejected.

Instantaneous Description is double in turing machine (Q, H) where Q is current state and H is input string.

Turnstile (⊢) symbol is used for connecting pair of Instantaneous descriptions (IDs) that represents moves of turing machine

Example:

show the Instantaneous Description's of string abba for the given turing machine.

$\delta(q_0, a) \longrightarrow (q_0, X, R)$

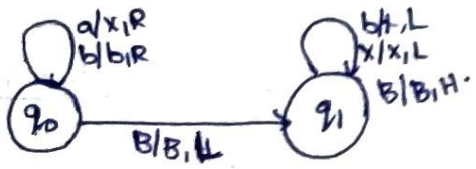
$\delta(q_0, b) \longrightarrow (q_0, b, R)$

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

$\delta(q_0, b) \longrightarrow (q_1, Y, L)$

$\delta(q_1, X) \longrightarrow (q_1, X, L)$

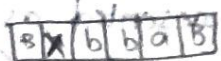
$\delta(q_1, B) \longrightarrow (q_1, B, H)$



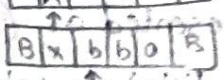
Instantaneous Description of given string

W = abba

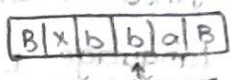
$ID(q_0, abba) \vdash (q_0, \underline{X}bba)$



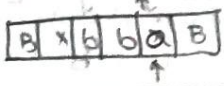
$\vdash (q_0, X\underline{b}ba)$



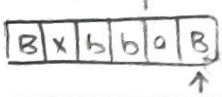
$\vdash (q_0, Xb\underline{b}a)$



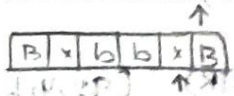
$\vdash (q_0, Xbb\underline{a})$



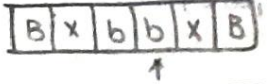
$\vdash (q_0, Xbba\underline{B})$



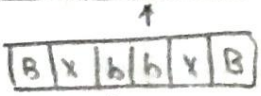
$\vdash (q_0, Xbb\underline{X})$



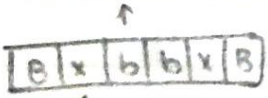
$\vdash (q_1, Xb\underline{b}X)$



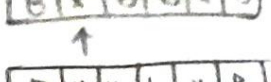
$\vdash (q_1, X\underline{b}bX)$



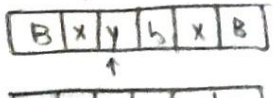
$\vdash (q_1, \underline{X}bbX)$

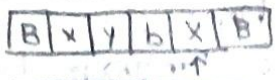


$\vdash (q_1, X\underline{Y}bX)$

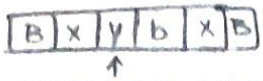


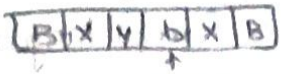
$\vdash (q_1, XY\underline{b}X)$

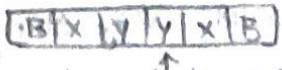


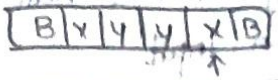
$\vdash (q_1, xyb\underline{x})$ 

$\vdash (q_1, xy\underline{b}x)$ 

$\vdash (q_1, xy\underline{b}x)$ 

$\vdash (q_1, xy\underline{b}x)$ 

$\vdash (q_1, xyy\underline{x})$ 

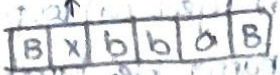
$\vdash (q_1, xyy\underline{x})$ 

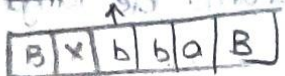
$\vdash (q_1, xyyx\underline{B})$ 

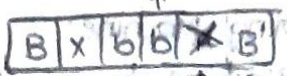
Instantaneous Description of given string

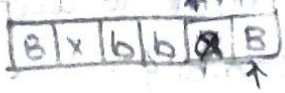
$w = abba$.

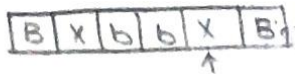
$ID(q_0, abba) \vdash (q_0, \underline{x}bba)$ 

$\vdash (q_0, x\underline{b}ba)$ 

$\vdash (q_0, xb\underline{b}a)$ 

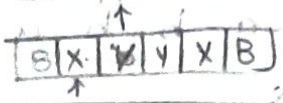
$\vdash (q_0, xbb\underline{a})$ 

$\vdash (q_0, xbb\underline{B})$ 

$\vdash (q_0, xbbx)$ 

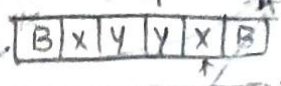
$\vdash (q_0, xby\underline{x})$ 

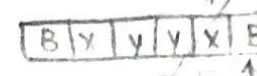
$\vdash (q_1, xyy\underline{x})$ 

$\vdash (q_1, xyy\underline{x})$ 

$\vdash (q_1, xyyx)$ 

$\vdash (q_1, xyyx)$ 

$\vdash (q_1, xyyx\underline{B})$ 

$\vdash (q_1, \underline{B}xyyx)$ 

Languages of Turing Machine:

there are two types of languages for Turing Machine. They are recursive languages and recursive enumerable languages.

Recursive Language:

Recursive language is a formal language for which there exists a turing machine that accepts every string in a language and rejects if it is not in the language. It has halt and accept (or) reject stages are there. for given input

Recursive languages are called as Decidable languages.

Recursive Enumerable Language:

Recursive Enumerable Language is a formal language for which there exists a turing machine that accepts when a string is in language but it may loop forever if the string is not in the language. It has halt and accept (or) reject (or) loop forever (infinite loop) stages are there. Recursive Enumerable languages are called as Recognizable languages.

Construction of Turing Machine:

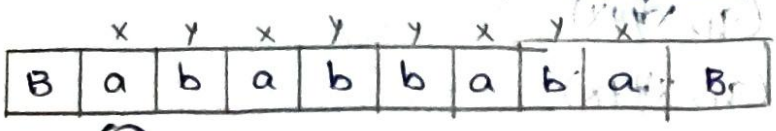
Design Turing Machine to accept the language

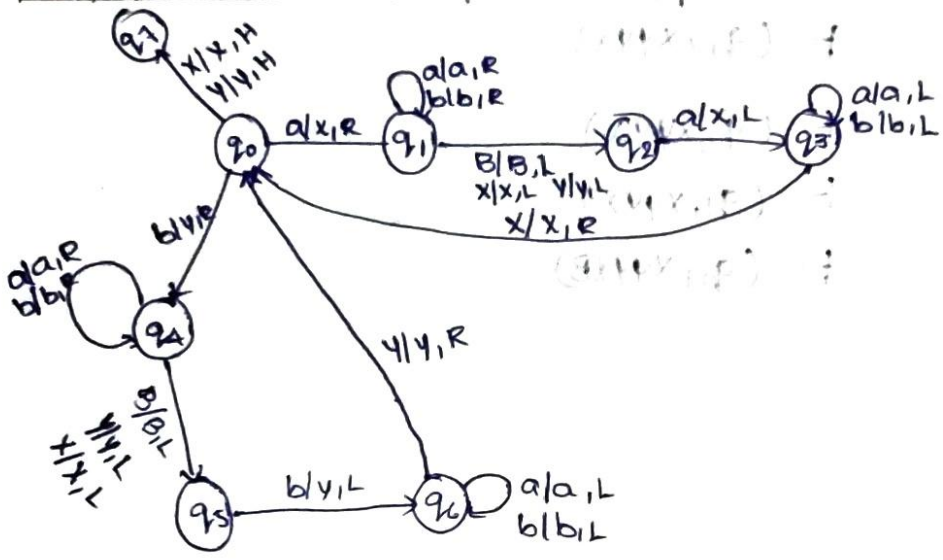
$L = \{WW^R, W \in (a,b)^+\}$ ⟶ Even palindrom

String Acceptance:

aa, aaaa, abba, aabbaa, ababbaba, abbbba, ...,

bb, bbbb, baab, bbaabb, babaabab, baaaab, ...





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

$\delta(q_0, a) \longrightarrow (q_1, a, R)$

$\delta(q_1, b) \longrightarrow (q_1, b, R)$

$\delta(q_1, B) \longrightarrow (q_2, B, L)$

$\delta(q_1, X) \longrightarrow (q_2, X, L)$

$\delta(q_1, Y) \longrightarrow (q_2, Y, L)$

$\delta(q_2, a) \longrightarrow (q_3, X, L)$

$\delta(q_3, a) \longrightarrow (q_3, a, L)$

$\delta(q_3, b) \longrightarrow (q_3, b, L)$

$\delta(q_3, X) \longrightarrow (q_0, X, R)$

$\delta(q_0, b) \longrightarrow (q_4, Y, R)$

$\delta(q_4, a) \longrightarrow (q_4, a, R)$

$\delta(q_4, b) \longrightarrow (q_4, b, R)$

$\delta(q_4, B) \longrightarrow (q_5, B, L)$

$\delta(q_4, X) \longrightarrow (q_5, X, L)$

$\delta(q_4, Y) \longrightarrow (q_5, Y, L)$

$\delta(q_5, b) \longrightarrow (q_6, Y, L)$

$\delta(q_6, a) \longrightarrow (q_6, a, L)$

$\delta(q_6, b) \longrightarrow (q_6, b, L)$

$\delta(q_6, Y) \longrightarrow (q_0, Y, R)$

$\delta(q_0, X) \longrightarrow (q_7, X, H)$

$\delta(q_0, Y) \longrightarrow (q_7, Y, H)$

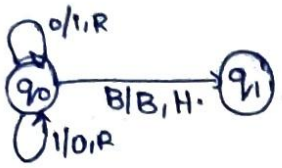
Construction of Turing Machine for 1's compliment of Binary number

1's compliment of given binary number is reciprocal of given number. It means we encounter '1' replace with '0' and '0' replace with '1'.

Example:

Given number = 100110

1's compliment = 011001



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

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

$\delta(q_0,B) \longrightarrow (q_1,B,H)$

Construction of Turing Machine for 2's compliment of Binary number

2's Compliment is nothing but 1's compliment + 1.

Example:

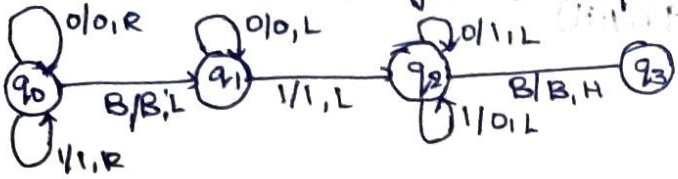
Given number = 100110

1's compliment = 011001

2's compliment = 011010 (1 1)

LSB MSB

By observing from LSB up to encounter first one digits are same after encounter first one all digits are reciprocal.



$\delta(q_0,0) \longrightarrow (q_0,0,R)$

$\delta(q_0,1) \longrightarrow (q_0,1,R)$

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

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

$\delta(q_1,1) \longrightarrow (q_2,1,L)$

$\delta(q_2,0) \longrightarrow (q_2,1,L)$

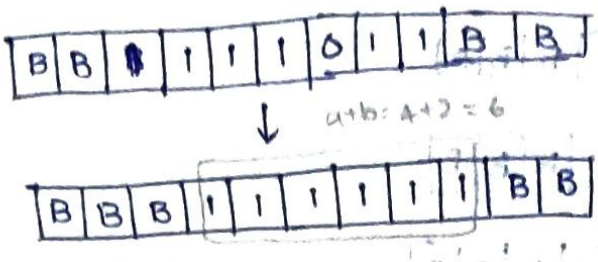
$\delta(q_2,1) \longrightarrow (q_2,0,L)$

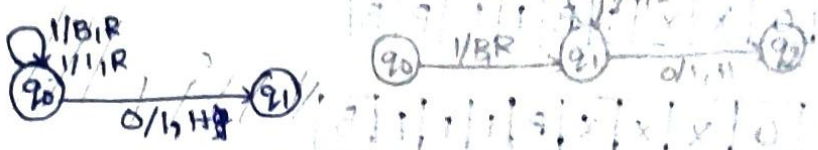
$\delta(q_2,B) \longrightarrow (q_3,B,H)$

Addition:

a = 4 b = 2

1111 0 11





$\delta(q_0, 1) \longrightarrow \delta(q_0, B, R)$

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

$\delta(q_1, 0) \longrightarrow (q_2, 1, H)$

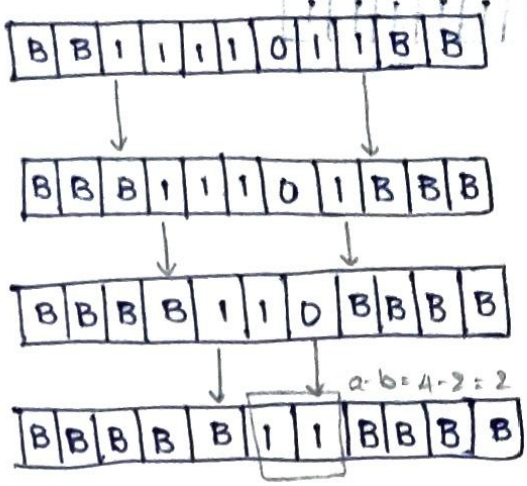
Subtraction:

m > n ; positive value

m ≤ n ⟹ 0

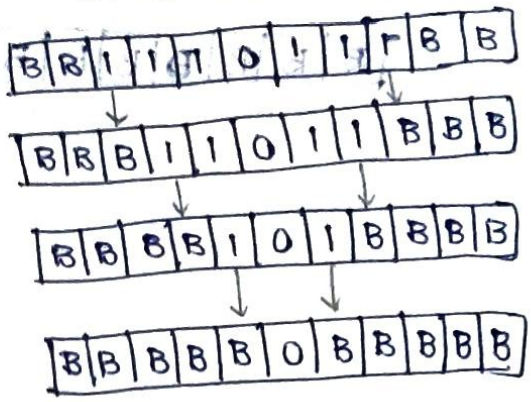
a = 4 b = 2 (positive a>b)

1111 0 11



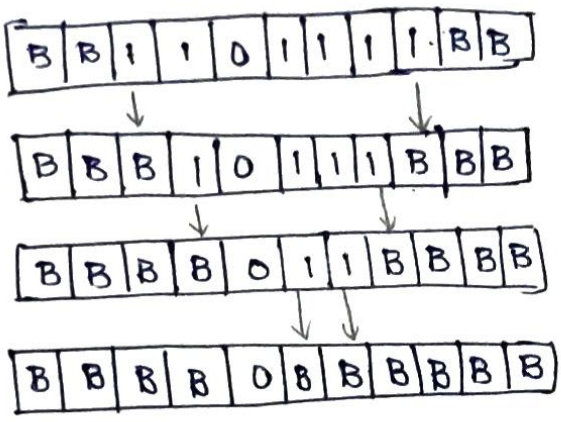
a = 3 b = 3 (a = b)

111 0 111



a = 2 b = 4 (a<b)

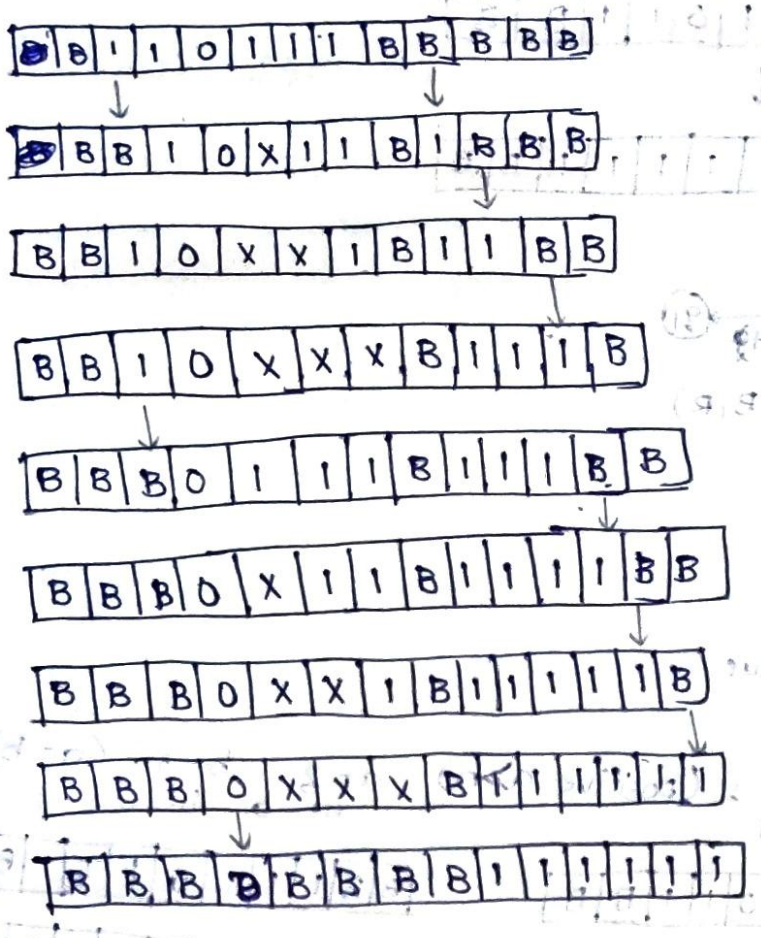
11 0 1111



Multiplication:

a = 2 b = 3

a x b = 6



Variations of the Turing Machine:

The different types of Turing machine are

1. Multi-tape Turing machine
2. Multi-head Turing machine
3. Multi-dimensional Turing machine
4. Non-deterministic Turing machine
5. Enumerator

Multi-tape Turing Machine:

The name suggests that this type of Turing machine has multiple tapes instead of one. each of the tape connected to finite control by read-write head. Transitional function of multi-tape turing machine depends on the present state and input symbol scanned by each of the read-write head from the respective input tape. Turing Machine can.

1. change its state.
2. Write a new symbol on the respective cell of the respective tape from where the inputs were scanned.
3. Move the head to one left (or) one right.

(~~Transition~~)

A multi-tape Turing machine having k tapes ($k \geq 1$) is symbolically represented as

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

Where Q = Finite set of states

Σ = Finite set of input symbols (or) input alphabets

Γ = Finite set of tape symbols

q_0 = Initial state

B = A symbol of Γ called Blank

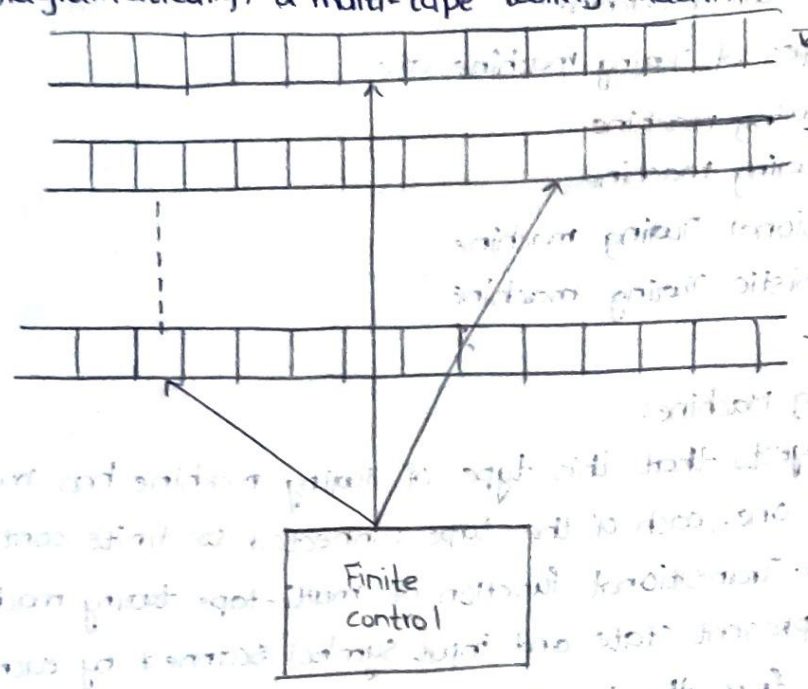
F = Final state

δ = Transitional Function

Where transitional function δ mapping is

$$\delta(Q \times T^N) \longrightarrow Q \times T^N \times (L, R, H).$$

Diagramatically, a multi-tape Turing machine can be denoted as



Multi-head Turing Machine:

The name signifies that this type of Turing machine has multiple heads. All the heads are connected to the finite control.

Transition function δ mapping is

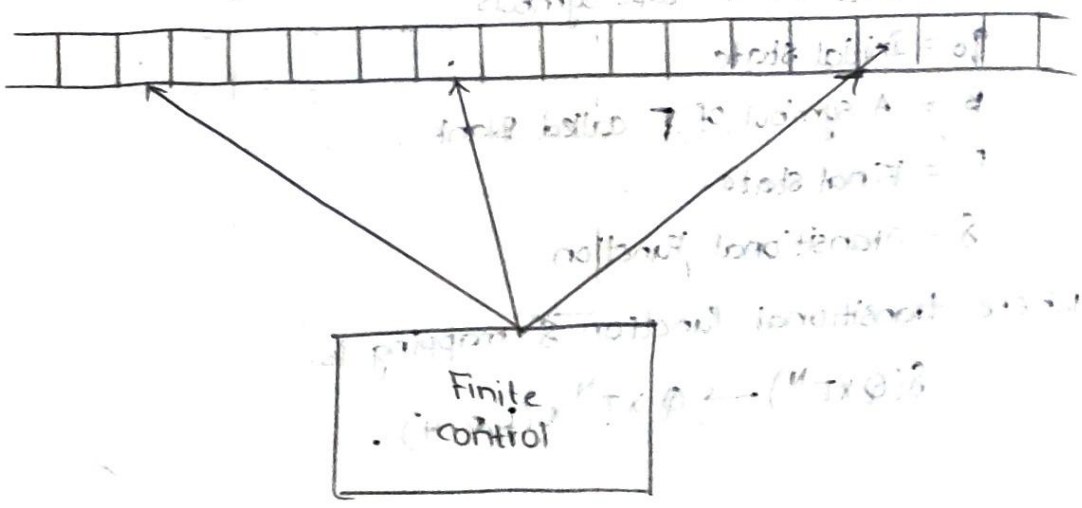
$$\delta(Q, \Gamma_1, \Gamma_2, \Gamma_3) \longrightarrow Q \times \Gamma_1 \times \{L, R, N, H\} \times \Gamma_2 \times \{L, R, N, H\} \ldots$$

Where ~~Γ_1 is the input symbol under head 1~~

~~Γ_1 is the symbol written by replacing Σ_1~~

N means no movement of the head.

A diagrammatical representation of a multi-head Turing machine is given in



We have to be careful about 2 special cases in designing the multi-head turing machine:

1. A situation may arise when more than one heads are scanning a particular cell at the same. But symbol written by one head is different from the symbol written by other hand.
2. Another situation may arise when cell under a head is leftmost cell and transition function instructs the head to go left. This condition is called hanging.

Multi-dimensional Turing Machine:

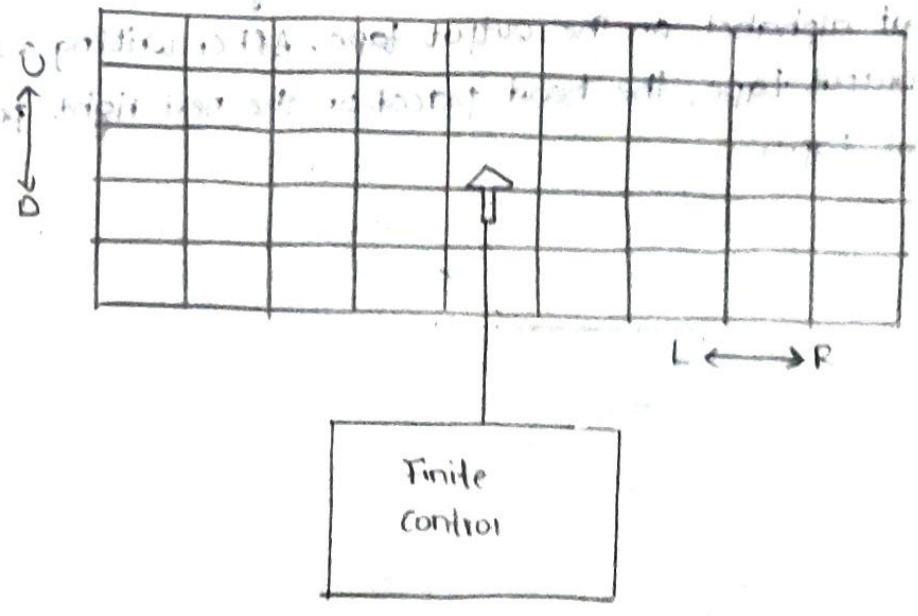
The input tape of two-dimensional turing machine is extended to infinity in both sides but in one direction. If the input tape can extend infinitely in more than one dimension, then the turing machine is called multidimensional turing machine

In case of two-dimensional turing machine, input tape is extended to infinity in x and y direction, in this case, read-write head can move in left, right, up and down direction.

Transitional function δ mapping is

$$\delta(Q \times \Gamma) \longrightarrow Q \times \Gamma \times (L, R, U, D, H)$$

Where L = Left
R = Right
U = Up
D = Down
H = Halt.



Non-deterministic Turing Machine:

We are familiar with non-deterministic while discussing the finite automata and PDA. We see that in non-deterministic finite automata, see that single state and single input there may be more than one move.

We have see different turing machine, but all are determined with a unique triplet combination of next state, tape symbol and move. for present state and a tape symbol.

But in case of non-deterministic turing machine, for present state and input symbol, there may be more than one move.

The transitional function δ mapping

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

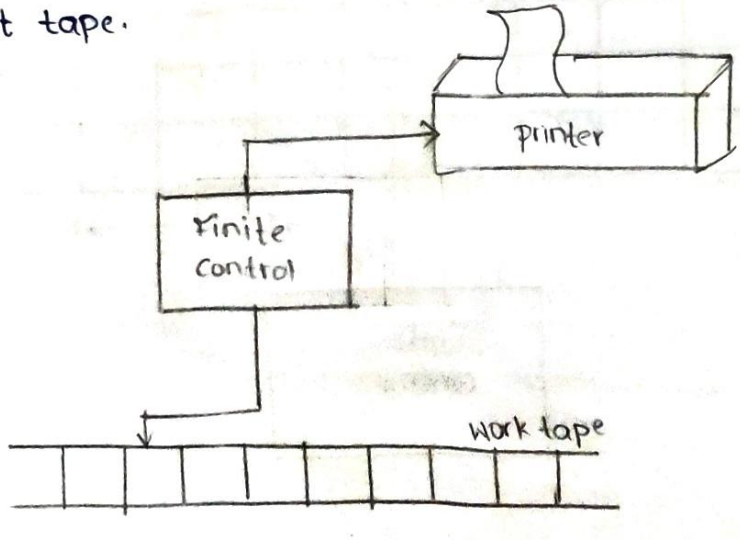
Offline Turing machine:

In offline turing machine, two input tape are connected to the finite control. one input tape for read only and other for write. Initially write tape is empty, if any input is read from read only tape.

Enumerator:

Enumerator is type of turing machine which is attached to a printer. It has a work tape and an output tape. The work tape is write only onee tape.

At each step, the machine chooses a symbol to write from the output alphabet on the output tape. After writing a symbol on the output tape, the head placed on the next right position and input tape.



Enumerator has special state, q_p, entering in which the string w on output is printed then the output tape is erased and head move to left most position.

Transition function δ mapping. is

$$Q \times \Sigma \times T \longrightarrow (Q \times \Sigma \times \{L, R\} \times T \times \{L, R\})$$

Church Thesis:

Church thesis is also known as church Turing thesis Alonzo church, an american mathematician proposed that any machine that can perform computable operations will able to perform all possible algorithms.

Allen Turing, a PhD scholar of church proposed, a machine called Turing machine. It performs all computable operations like present digital computer.

Statement of thesis:

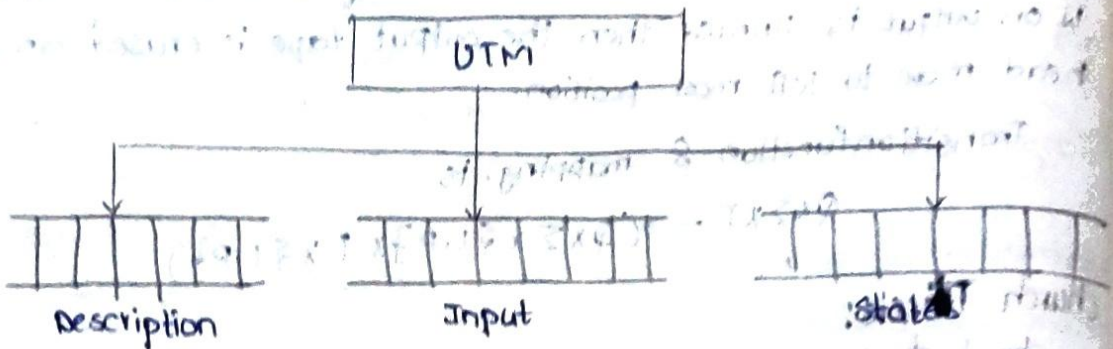
1. Anything that can be done by current digital computer can also be done by Turing machine.
2. Currently there is no problem which can be solved by digital computer and cannot be solved by turing machine.
3. Many mathematical models are suggested but none of them is more powerful than turing machine

Universal Turing Machine (UTM):

Turing machine can perform any computational process carried out by the present day's computer. But turing machine is designed to execute only one program. But real computer computes reprogrammable logics.

Turing machine is called universal turing machine, if it simulates the behaviour of digital computer by accepting input data and the description of turing machine (logic or transitions). So, turing universal turing machine can simulate all turing machines, designed for each separate task.

Universal turing machine is designed with three input tapes. One for description of turing machine (Transitions of turing machine), second tape for input and third tape for states.



In universal turing machine, all input symbols, tape symbols, and movement symbols (or) direction symbols are represented by ones (1's) and each are seperated by zero (0).

Example:

q_0 - 1	L - 1	X - 1	a - 1
q_1 - 11	R - 11	Y - 11	b - 11
q_2 - 111			

$(q_0, a) \longrightarrow (q_1, X, L)$

10 10 11 0 10 1

Restricted Turing machine:

Restricted turing machine while constructing the turing machine impose certain restrictions on the turing machine then such types of turing machines are called as Restricted Turing machine.

Restricted turing machines can be of the following types.

1. Halting turing machine
2. Linear bound Automata
3. Uni directional turing machine
4. Read only turing machine
5. Read only - unidirectional turing machine

Halting Turing machine:

Turing machine is said to be halting turing machine if it always halts for every input string. Halting turing machine accepts recursive languages.

Linear Bound Automata

It behaves as turing machine but the storage space of tape is restricted to only to the length of the input string.

Unidirectional Turing Machine:

The head of the turing machine can move only in one direction, then such type of turing machines is called unidirectional turing machines. Unidirectional turing machine can accepts only regular languages.

Read-only Turing Machine:

It is equivalent to finite automata. It can have read only head which does not have written capability. This type of turing machine is called read-only turing machine.

Read-only turing machines accepts only regular languages.

Read only - Unidirectional Turing Machine:

It is similar to finite automata. It contains read only head and can move only in one direction. This types of turing machines are called read only - unidirectional turing machine:

Thise turing machines accepts regular languages

Decidable and Undecidable problems: (Decidability and Undecidability):

Decidable problems:

If a problem is said to be decidable, if turing machine M can be constructed for the program and M should be halt in finite type time for every input and says yes (or) No.

Example:

1. Equivalence of two DFA's:
2. Finiteness of Regular language.
3. Membership problem of string in Regular language.

Undecidable problems:

If a problem is said to be undecidable, if turing machine M cannot be constructed for the problem. There is no proper algorithm to determine answer for the given input.

Examples:

1. Halting problem of Turing machine.
2. post correspondence problem (pcp)
3. Modified post correspondence problem (Mpcp)
4. Ambuiguity in Context-free grammar

Halting Problem:

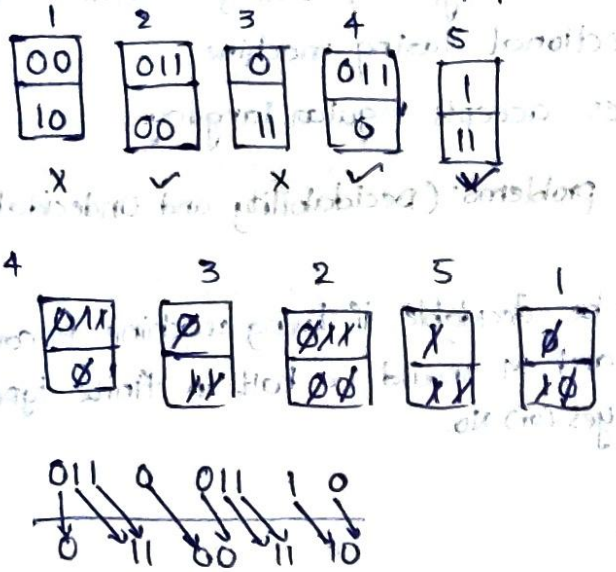
Basically Halting means a program on certain input will accept it and halt (or) reject it and halt (It means terminating).

So, can we have an algorithm that tells the given problem will halt (or) not?

The answer is No, because we cannot design an algorithm which appropriately says that the given problem will halt (or) not.

The only solution for such type of problems is to run the program and check whether it halts (or) not. Because of not designing algorithms for checking the halting problem, it can be categorized as the "Undecidable problem".

Post correspondence problem (pcp):



This pcp has a solution in the series 43251.

PCP:

PCP problem was proposed an American Mathematician Emil Leon post in 1946. This is well known undecidable problem in theory of computation.

PCP consists of set of elements represented by dominos

1. dominos | tiles | coods.

eg: $\left[\frac{ab}{ac}\right]\left[\frac{c}{d}\right]$

Use each dominos many times to find a solution.

2. variables

$X = \{a, b, c\}$

$Y = \{a, c, d\}$

	x	y
1	ab	ac
2	c	d

* Arrange the above representation in a sequence one after another by starting with the same character in top and bottom in repeated manner. Finally the top and bottom string become same then the problem has station other wise no solution.

PCP definition:

Given two sequences of strings $W_1, W_2 \ldots W_n$ and $x_1 x_2 x_3 \ldots x_n$ over Σ. The solution of the problem is to find a non empty sequence of integers $i_1, i_2 \ldots i_k$ such that $W_{i1} - W_{i2} \ldots W_{ik} = x_{i_1} x_{i_2} \ldots x_{i_k}$.

Modified post correspondence problem (MPCP):

If the first substring used in pcp (or) W_1 and x_1 for all instances then the pcp is called modified post correspondence problem.

Given two sequences of string $W_1, W_2 \ldots W_n$ and $x_1, x_2 \ldots x_n$ over alphabet Σ. the solution of the problem is to find a non empty sequence of integers $i_2, i_3 \ldots i_k$ such that $W_{i1} W_{i2} \ldots W_{ik} = x_{i_1} x_{i_2} \ldots x_{i_k}$.

The derivation process of the string accepted by a given grammar can be reduced to MPCP. We know that MPCP means two sequence of strings denoted by 2 sets A and B. The derivation process can be reduced to MPCP with the help of following table.

Top(A)	Bottom(B)	Grammar G
$FS \rightarrow$	F	S: starting symbol ; F: special symbol
a	a	for every symbol $\in \Sigma$
V	V	for every non terminal symbol
ε	$\rightarrow w\varepsilon$	W : is string
y	x	for every production of $x \rightarrow y$
$\rightarrow$	$\rightarrow$	

1. $S \rightarrow aBAb / Aac$, $Ab \rightarrow c$, $Bc \rightarrow acb$ the string aacb convert the derivation to generate string to MPCP.

The grammar with the string is mapped in the following table.

W	A	X	B
1	$FS \rightarrow$	1	F
2	a	2	a
3	b	3	b
4	c	4	c
5	A	5	A
6	B	6	B
7	C	7	C
8	S	8	S

9	E	9	→aaBbc
10	aBAb	10	S
11	Aac	11	S
12	c	12	Ab
13	acb	13	Bc
14	↪	14	↪

S →aBAb →aBc →aacb.

w_1 w_{10}

→ FS → aBAb

$x_1 x_{10}$ x

w_1 w_{10} w_{14} $w_2 w_6 w_{12}$

→ FS → aBAb → aBc

$x_1 x_{10} x_{14}$ — $x_2 x_6 x_{12}$ $x_2 x_6 x_{12}$

w_1 w_{10} w_{14} $w_2 w_6 w_{12}$ $w_2 w_{13} w_4$

→ FS → aBAb → aBc → aaBbc

$x_1 x_{10} x_{14}$ $x_2 x_6 x_{12}$ x_{14} x_2 x_{13} x_4

Reduction of MPCP to PCP:

Given two sequences of string $w_1, w_2 \ldots w_n$ and $x_1, x_2 \ldots x_n$ over alphabet Σ. A sequence of integers $i_2, i_3 \ldots i_k$ exists $w_1 w_{i_2} \ldots w_{i_k} = x_1 x_{i_2} x_{i_3} \ldots x_{i_k}$.

Procedure:

1. Introduction a new symbol *
2. In the first list w_i the new symbol star(*) appears after every symbol. In the second list x_i the new symbol star(*) appears before every symbol.
3. Take the first pair $w_1 x_1$ from the given MPCP and add to the PCP instance another instance pair in which * remains as same. But the extra * is added to the w_i at the beginning.
4. Since the first list has an extra * at the end add another pair $ and *$ to the PCP instance. this is referred as a final pair.

1. convert the following MPCP to an equivalent PCP

	w_i	x_i
1	01	011
2	10	0
3	01	11
4	10	0

This is an MPCP as there is a sequence 1432 which generates a string 011011O

1	A	3	2
0X	1	0Y	10
0XY	0	1X	0

Adding * after every symbol in w_i before every symbol in x and adding an extra star(*) at the beginning of w_i the sequence is

	w_i	x_i
0	*0*1*	*0*1*1
1	0*1*	*0*1*1
2	1*0*	*0
3	0*1*	*1*1
4	1*	*0

Adding extra sequence $ and *$ to the existing sequence as a final pair then pcp becomes

	w_i	x_i
0	*0*1*	*0*1*1
1	1*0*	*0
2	0*1*	*1*1
3	1*	*0
4	$	*$

0 4 3 2 5

4325

Mcp to pcp

This is a pcp as there exist a sequence 14325 which generates the string *0*1*1*0*1*1*0*$

Types of complexity classes:

1. problem is said to be decidable if it has an algorithm to solve it. The problem can be categorised into 2 groups depending on time taken for their execution

Ex: problems whose solution time are bounded by polynomials of small degree

Bubble sort algorithm obtains n numbers in sorted order in polynomial time

2. Second group is made up of problems whose best known algorithm are non-polynomial

Ex: Travelling sales man problem is solved in exponential time

P-problem: (Polynomial)

P stands for deterministic polynomial time. A language L is said to be in class P if there exists a (deterministic) Turing machine in such that M is of time complexity $P(n)$ for some polynomial p and M accepts L.

class P consists of those problem that are solvable in polynomial time by DTM. class P consists of problems which are solved quickly

NP-problem: Non-polynomial

NP stands for non-deterministic polynomial time. the class NP consists of those problems that are solved in polynomial time.

A langue L is in class NP if there is an (non-deterministic) turing machine such that M is of time complexity $p(n)$ for some polynomial p and M accepts L.

Class NP consists of problems for which solution are verified quickly.

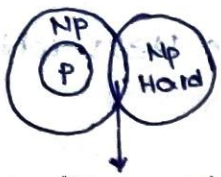
NP-Hard problem:

Problem S is said to be NP-hard if it satisfies the following conditions every other problems $S_i \in NP$ for $i = 1, 2 \ldots n$ there is polynomial time transformation from S_i to S. ie., every problem in NP class polynomial time reducable to.

NP-complete problems:

problem S is said to be NP-complete problem if it satisfies the following conditions.

1. $S \in NP$ and
2. for every other problems, $S_i \in NP$ for some $i = 1, 2 \ldots n$ there is polynomial time transformation from S_i to S i.e, every problem in NP class polynomial time reducible to S.



Techniques of Constructing Turing Machine:

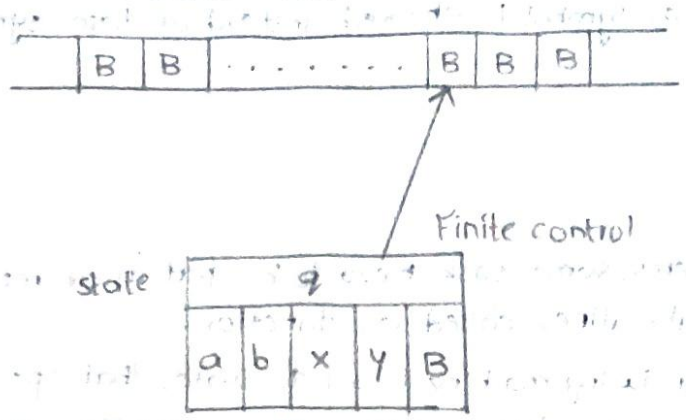
Programming Techniques that used to construct an efficient turing machine that functions as powerful as conventional computer.

* Different techniques that are used to design a turing machine are as follows:

1. Storage in finite control
2. Multiple tracks
3. chiecking offline symbols
4. Sub routines.

Storage in finite control:

In turing machine, finite control contains the finite automata with the state transitions. It represents set of states. But in storage of finite control, we store data along with state.

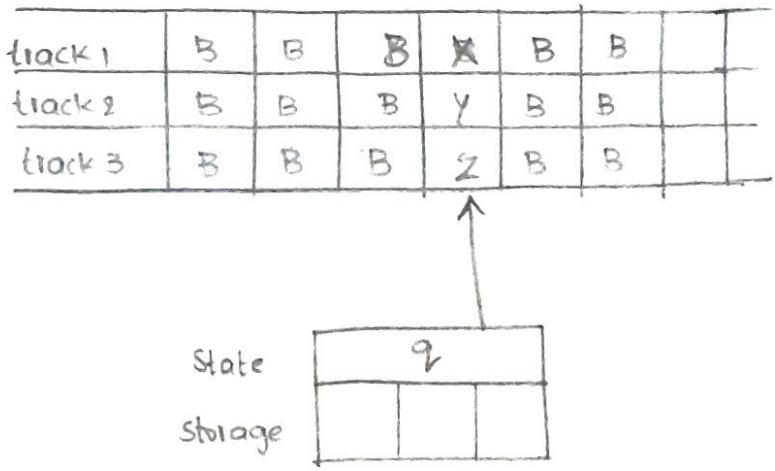


* It is useful to remember the previous input tape on the current state before the transition here state is represent like [q, B]

current state, previous state.

Multiple tracks:

In Turing machine, instead of one input tape extends it to multiple tracks in the input tape each track in input tape contains one symbol.



Here single tape head reads n symbols from n tracks at one step, it accepts recursive enumerable language like normal single track single tape machines.

In transition instead of taking one symbol in input tape, it take one symbol from each track

$(q, a) \longrightarrow$ normal Turing machine

$(q, xyz) \longrightarrow$ Multitrack Turing Machine.

Checking offline symbols.

the Turing machine can be extended by using checking offline symbols this method is used by Turing machine for language that contains repeated strings and compare length of two sub strings.

Examples:

$L = \{WCW / W = \{a,b\}^*\}$

$L = \{WW / W = \{a,b\}^*\}$

$L = \{WW^R / W = \{0,1\}^*\}$

using * to mark as symbol is checked instead of tape symbol x, y.

Examples:

B	a	b	a	c		a	b	a	B

B	*	*	*	c	*	*	*	B

Subroutines:

In some problems, some task need to be performed repeatedly and it can done by subroutines called as function.

Subroutines in Turing machine is set to states that specially perform some tasks. Subroutine has 2 states, one is starting state and another is return state. Return state of subroutine does not have move and it pass control to other set of states which calls subroutine.

Mostly subroutine is used in TMs like multiplication turing machine to multiply two strings. In this multiplication is performing successive addition.

Input:

	B	B	1	1	0	1	1	1	B	B

output:

	1	1	1	1	1	1	B	B	B	