UNIT 5

Turing Machine

Turing machine was invented in 1936 by **Alan Turing**. It is an accepting device which accepts Recursive Enumerable Language generated by type 0 grammar.

There are various features of the Turing machine:

1. It has an external memory which remembers arbitrary long sequence of input.
2. It has unlimited memory capability.
3. The model has a facility by which the input at left or right on the tape can be read easily.
4. The machine can produce a certain output based on its input. Sometimes it may be required that the same input has to be used to generate the output. So in this machine, the distinction between input and output has been removed. Thus a common set of alphabets can be used for the Turing machine.

Formal definition of Turing machine

A Turing machine can be defined as a collection of 7 components:

Q: the finite set of states
∑: the finite set of input symbols
T: the tape symbol
q0: the initial state
F: a set of final states
B: a blank symbol used as a end marker for input
δ: a transition or mapping function.

The mapping function shows the mapping from states of finite automata and input symbol on the tape to the next states, external symbols and the direction for moving the tape head. This is known as a triple or a program for turing machine.

1. (q0, a) → (q1, A, R)

That means in q0 state, if we read symbol 'a' then it will go to state q1, replaced a by X and move ahead right(R stands for right).

Example:

Construct TM for the language L ={0^n1^n} where n>=1.

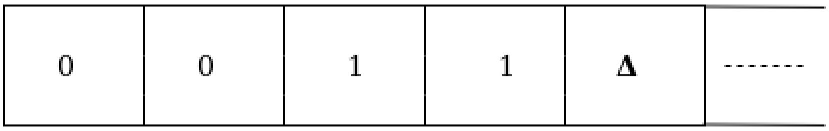
Solution:

We have already solved this problem by PDA. In PDA, we have a stack to remember the previous symbol. The main advantage of the Turing machine is we have a tape head which can be moved forward or backward, and the input tape can be scanned.

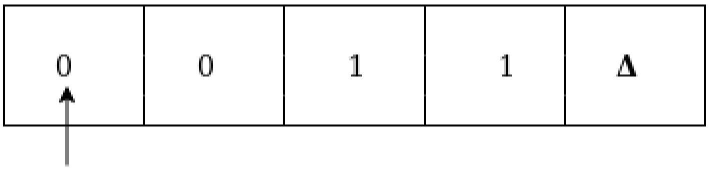
The simple logic which we will apply is read out each '0' mark it by A and then move ahead along with the input tape and find out 1 convert it to B. Now, repeat this process for all a's and b's.

Now we will see how this turing machine work for 0011.

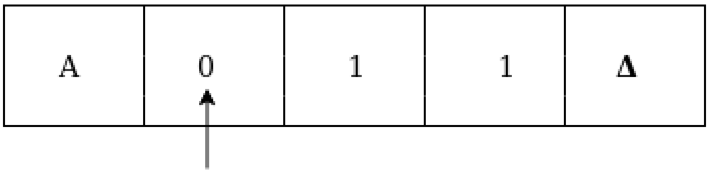
The simulation for 0011 can be shown as below:



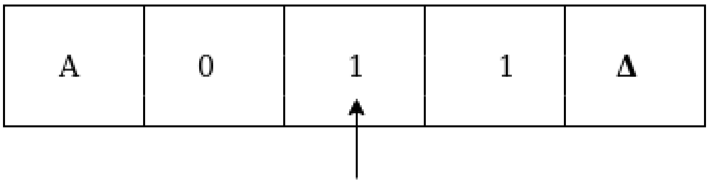
Now, we will see how this turing machine will works for 0011. Initially, state is q0 and head points to 0 as:



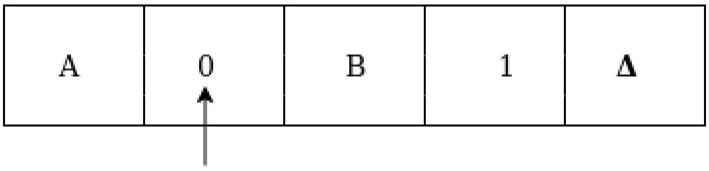
The move will be δ(q0, 0) = δ(q1, A, R) which means it will go to state q1, replaced 0 by A and head will move to the right as:



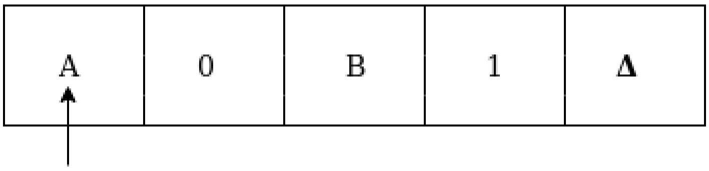
The move will be δ(q1, 0) = δ(q1, 0, R) which means it will not change any symbol, remain in the same state and move to the right as:



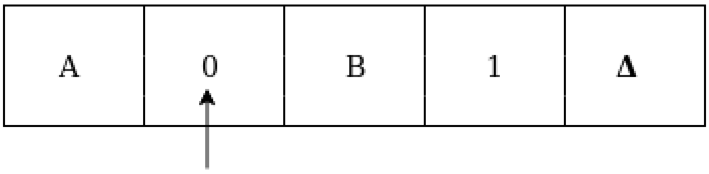
The move will be δ(q1, 1) = δ(q2, B, L) which means it will go to state q2, replaced 1 by B and head will move to left as:



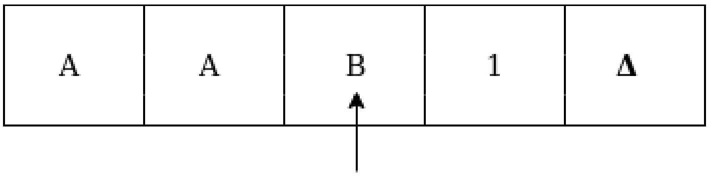
Now move will be δ(q2, 0) = δ(q2, 0, L) which means it will not change any symbol, remain in the same state and move to left as:



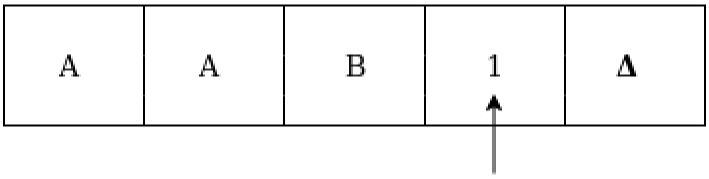
The move will be δ(q2, A) = δ(q0, A, R), it means will go to state q0, replaced A by A and head will move to the right as:



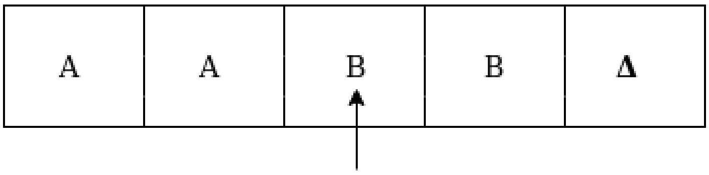
The move will be δ(q0, 0) = δ(q1, A, R) which means it will go to state q1, replaced 0 by A, and head will move to right as:



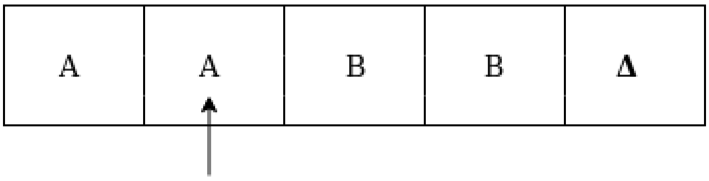
The move will be δ(q1, B) = δ(q1, B, R) which means it will not change any symbol, remain in the same state and move to right as:



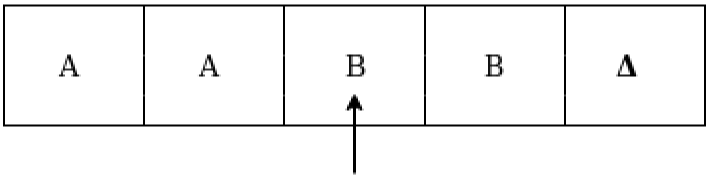
The move will be δ(q1, 1) = δ(q2, B, L) which means it will go to state q2, replaced 1 by B and head will move to left as:



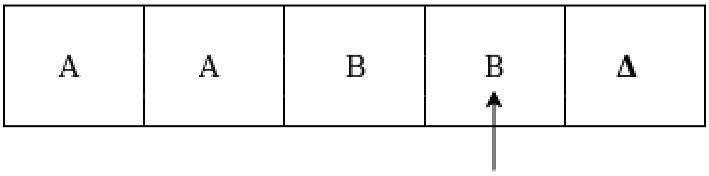
The move δ(q2, B) = (q2, B, L) which means it will not change any symbol, remain in the same state and move to left as:



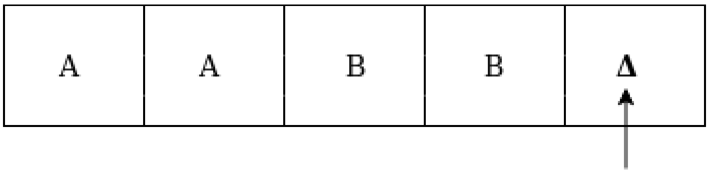
Now immediately before B is A that means all the 0?s are market by A. So we will move right to ensure that no 1 is present. The move will be δ(q2, A) = (q0, A, R) which means it will go to state q0, will not change any symbol, and move to right as:



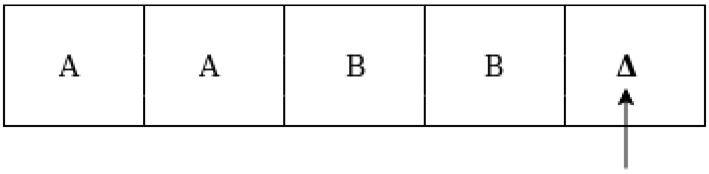
The move δ(q0, B) = (q3, B, R) which means it will go to state q3, will not change any symbol, and move to right as:



The move δ(q3, B) = (q3, B, R) which means it will not change any symbol, remain in the same state and move to right as:



The move δ(q3, Δ) = (q4, Δ, R) which means it will go to state q4 which is the HALT state and HALT state is always an accept state for any TM.

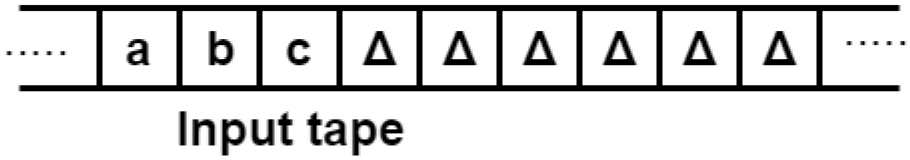


The same TM can be represented by Transition Diagram:

Basic Model of Turing machine

The turning machine can be modelled with the help of the following representation.

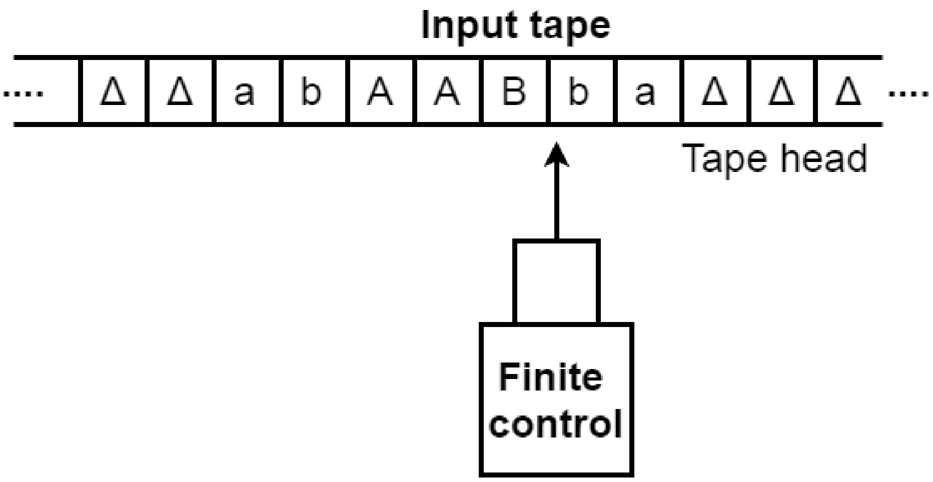
1. The input tape is having an infinite number of cells, each cell containing one input symbol and thus the input string can be placed on tape. The empty tape is filled by blank characters.



2. The finite control and the tape head which is responsible for reading the current input symbol. The tape head can move to left to right.

3. A finite set of states through which machine has to undergo.

4. Finite set of symbols called external symbols which are used in building the logic of turing machine.



Language accepted by Turing machine

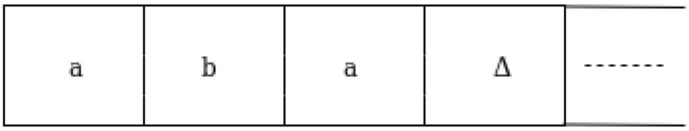
The turing machine accepts all the language even though they are recursively enumerable. Recursive means repeating the same set of rules for any number of times and enumerable means a list of elements. The TM also accepts the computable functions, such as addition, multiplication, subtraction, division, power function, and many more.

Example:

Construct a turing machine which accepts the language of aba over ∑ = {a, b}.

Solution:

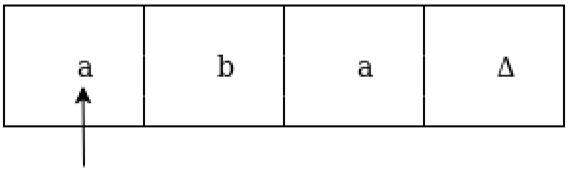
We will assume that on input tape the string 'aba' is placed like this:



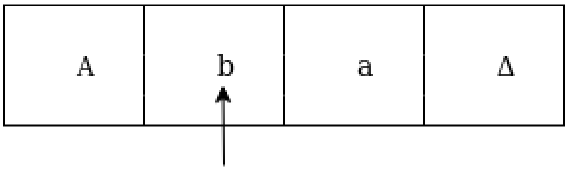
a	b	a	Δ	-------

The tape head will read out the sequence up to the Δ characters. If the tape head is readout 'aba' string then TM will halt after reading Δ.

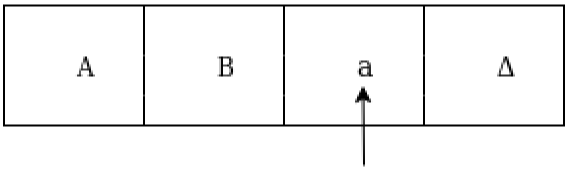
Now, we will see how this turing machine will work for aba. Initially, state is q0 and head points to a as:



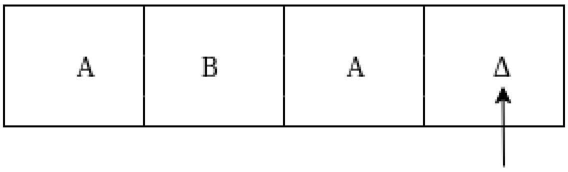
The move will be δ(q0, a) = δ(q1, A, R) which means it will go to state q1, replaced a by A and head will move to right as:



The move will be δ(q1, b) = δ(q2, B, R) which means it will go to state q2, replaced b by B and head will move to right as:



The move will be δ(q2, a) = δ(q3, A, R) which means it will go to state q3, replaced a by A and head will move to right as:

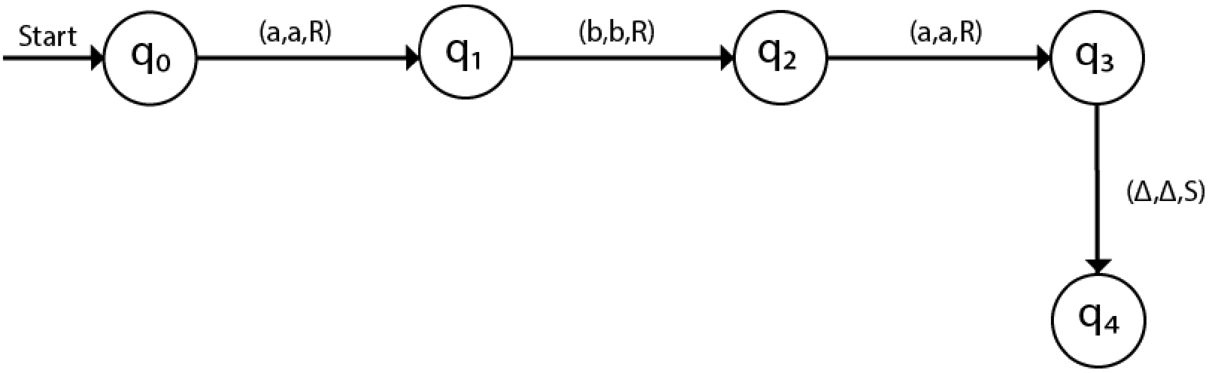


The move δ(q3, Δ) = (q4, Δ, S) which means it will go to state q4 which is the HALT state and HALT state is always an accept state for any TM.

The same TM can be represented by Transition Table:

States	a	b	Δ
q0	(q1, A, R)	–	–
q1	–	(q2, B, R)	–
q2	(q3, A, R)	–	–
q3	–	–	(q4, Δ, S)
q4	–	–	–

The same TM can be represented by Transition Diagram:



Examples of TM

Example 1:

Construct a TM for the language L = $\{0^n1^n2^n\}$ where n≥1

Solution:

L = $\{0^n1^n2^n \mid n\geq1\}$ represents language where we use only 3 character, i.e., 0, 1 and 2. In this, some number of 0's followed by an equal number of 1's and then followed by an equal number of 2's. Any type of string which falls in this category will be accepted by this language.

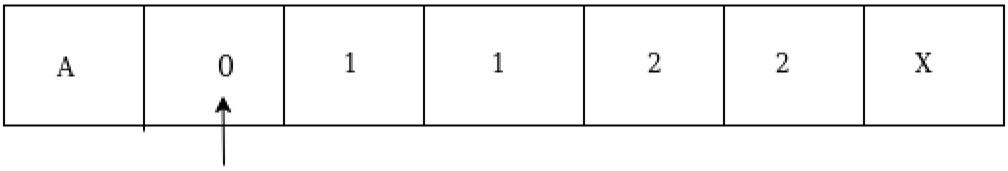
The simulation for 001122 can be shown as below:

0	0	1	1	2	2	X	-------

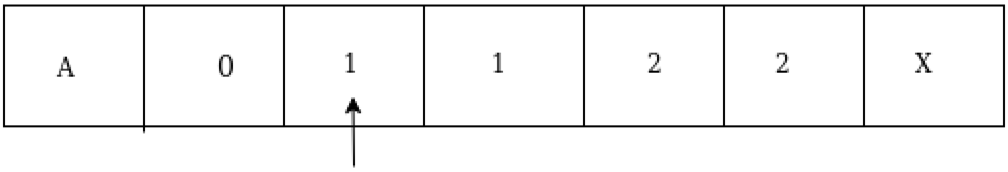
Now, we will see how this Turing machine will work for 001122. Initially, state is q0 and head points to 0 as:

0	0	1	1	2	2	X
↑						

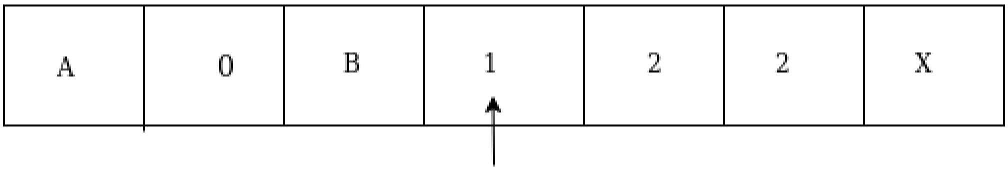
The move will be δ(q0, 0) = δ(q1, A, R) which means it will go to state q1, replaced 0 by A and head will move to the right as:



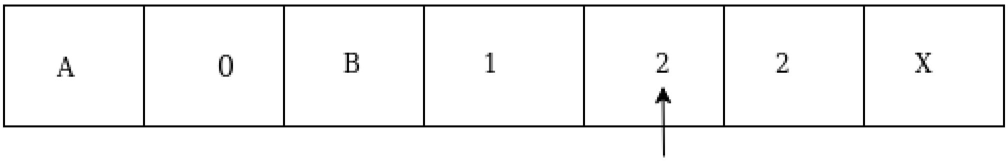
The move will be δ(q1, 0) = δ(q1, 0, R) which means it will not change any symbol, remain in the same state and move to the right as:



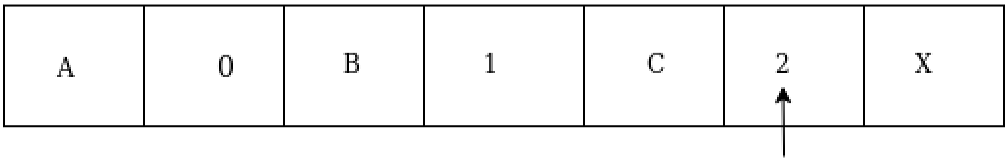
The move will be δ(q1, 1) = δ(q2, B, R) which means it will go to state q2, replaced 1 by B and head will move to right as:



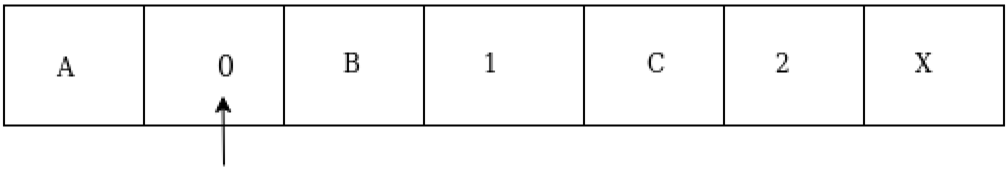
The move will be δ(q2, 1) = δ(q2, 1, R) which means it will not change any symbol, remain in the same state and move to right as:



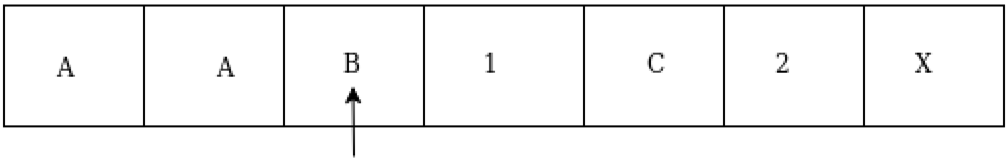
The move will be δ(q2, 2) = δ(q3, C, R) which means it will go to state q3, replaced 2 by C and head will move to right as:



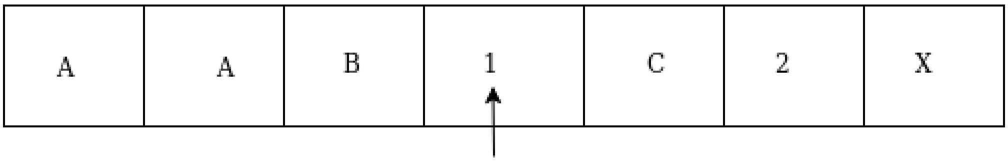
Now move δ(q3, 2) = δ(q3, 2, L) and δ(q3, C) = δ(q3, C, L) and δ(q3, 1) = δ(q3, 1, L) and δ(q3, B) = δ(q3, B, L) and δ(q3, 0) = δ(q3, 0, L), and then move δ(q3, A) = δ(q0, A, R), it means will go to state q0, replaced A by A and head will move to right as:



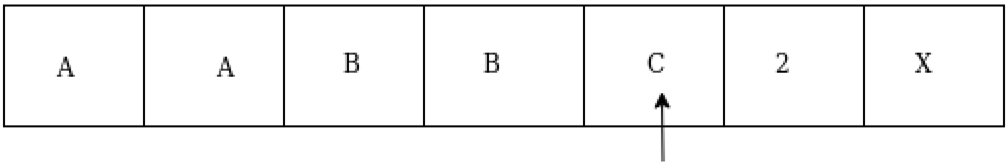
The move will be δ(q0, 0) = δ(q1, A, R) which means it will go to state q1, replaced 0 by A, and head will move to right as:



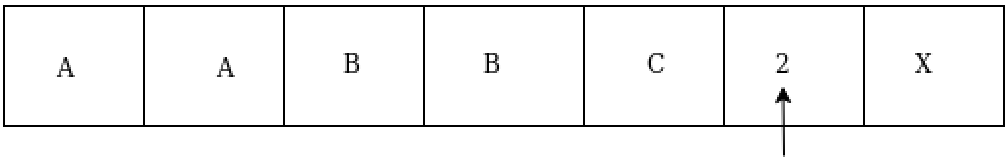
The move will be δ(q1, B) = δ(q1, B, R) which means it will not change any symbol, remain in the same state and move to right as:



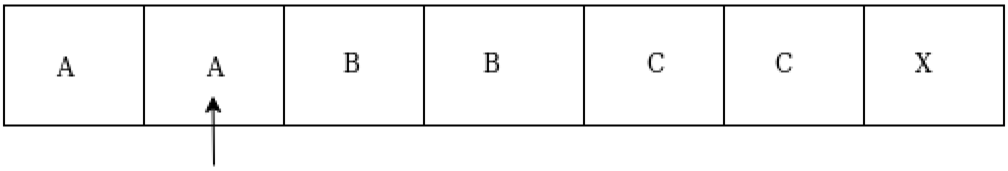
The move will be δ(q1, 1) = δ(q2, B, R) which means it will go to state q2, replaced 1 by B and head will move to right as:



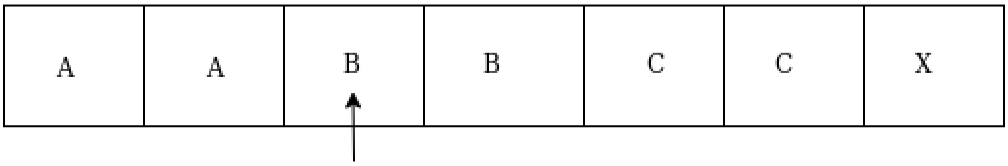
The move will be δ(q2, C) = δ(q2, C, R) which means it will not change any symbol, remain in the same state and move to right as:



The move will be δ(q2, 2) = δ(q3, C, L) which means it will go to state q3, replaced 2 by C and head will move to left until we reached A as:

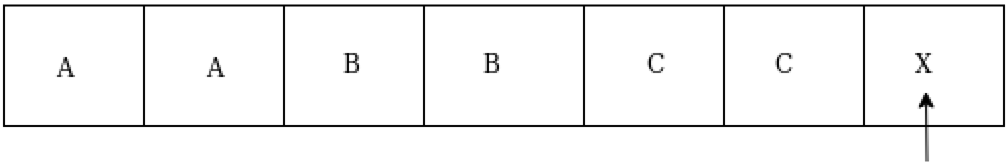


immediately before B is A that means all the 0's are market by A. So we will move right to ensure that no 1 or 2 is present. The move will be δ(q2, B) = (q4, B, R) which means it will go to state q4, will not change any symbol, and move to right as:



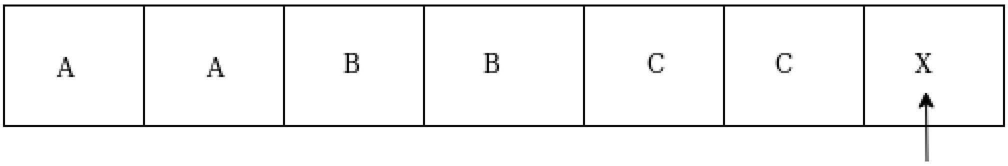
A	A	B	B	C	C	X

The move will be (q4, B) = δ(q4, B, R) and (q4, C) = δ(q4, C, R) which means it will not change any symbol, remain in the same state and move to right as:



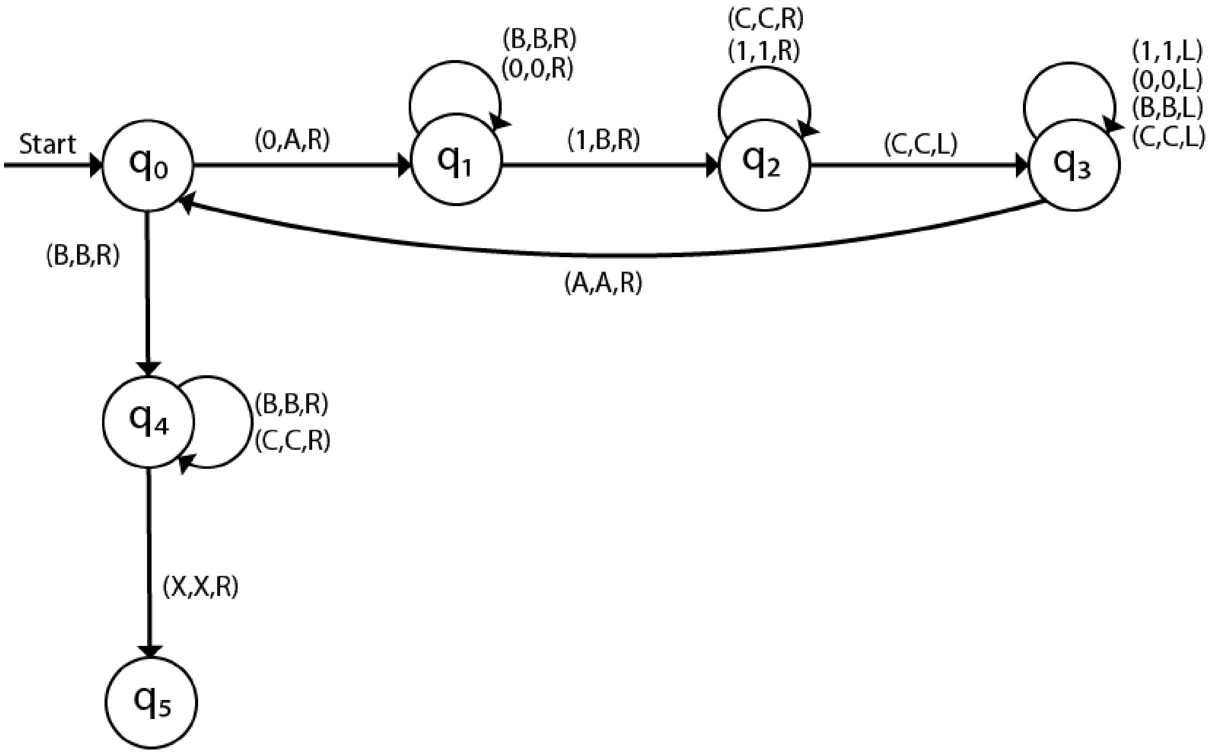
A	A	B	B	C	C	X

The move δ(q4, X) = (q5, X, R) which means it will go to state q5 which is the HALT state and HALT state is always an accept state for any TM.



A	A	B	B	C	C	X

The same TM can be represented by Transition Diagram:



Example 2:

Construct a TM machine for checking the palindrome of the string of even length.

Solution:

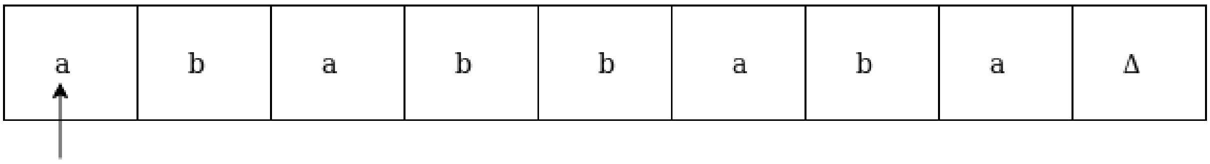
Firstly we read the first symbol from the left and then we compare it with the first symbol from right to check whether it is the same.

Again we compare the second symbol from left with the second symbol from right. We repeat this process for all the symbols. If we found any symbol not matching, we cannot lead the machine to HALT state.

Suppose the string is ababbabaΔ. The simulation for ababbabaΔ can be shown as follows:

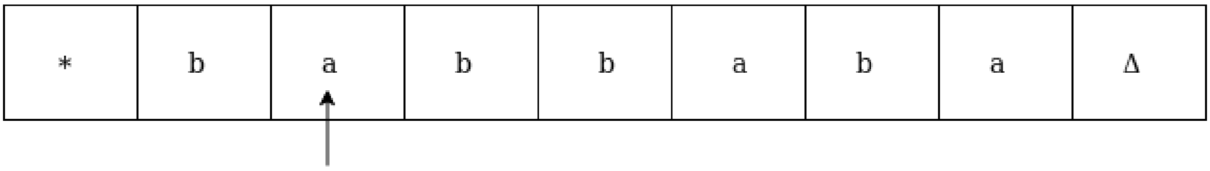
a	b	a	b	b	a	b	a	Δ	---------

Now, we will see how this Turing machine will work for ababbabaΔ. Initially, state is q0 and head points to a as:



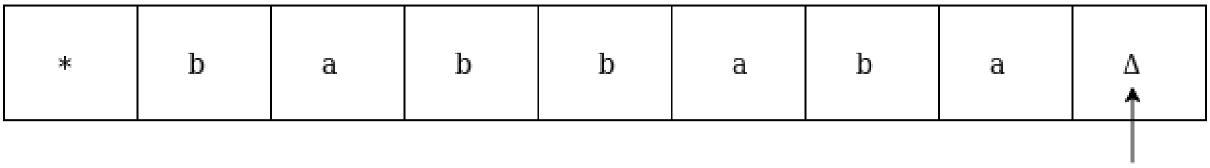
a	b	a	b	b	a	b	a	Δ

We will mark it by * and move to right end in search of a as:



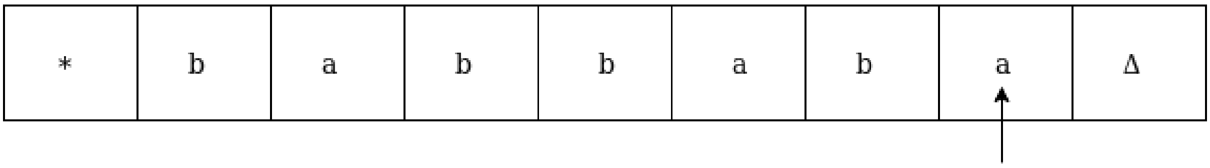
*	b	a	b	b	a	b	a	Δ

We will move right up to Δ as:



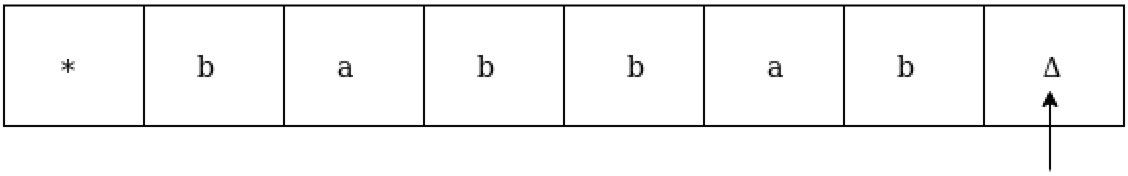
*	b	a	b	b	a	b	a	Δ

We will move left and check if it is a:



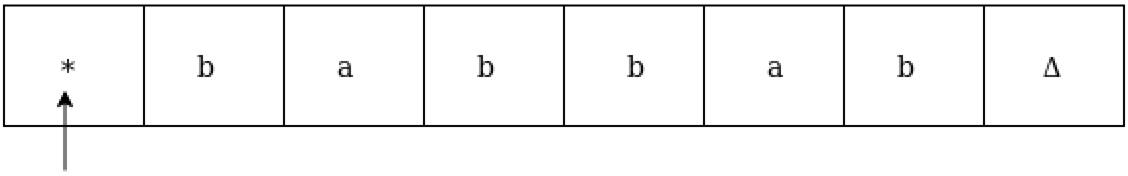
*	b	a	b	b	a	b	a	Δ

It is 'a' so replace it by Δ and move left as:



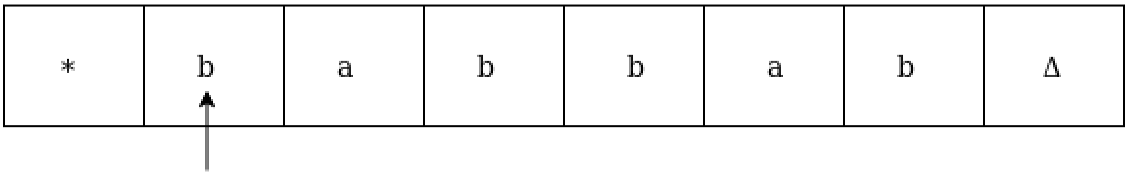
*	b	a	b	b	a	b	Δ

Now move to left up to * as:

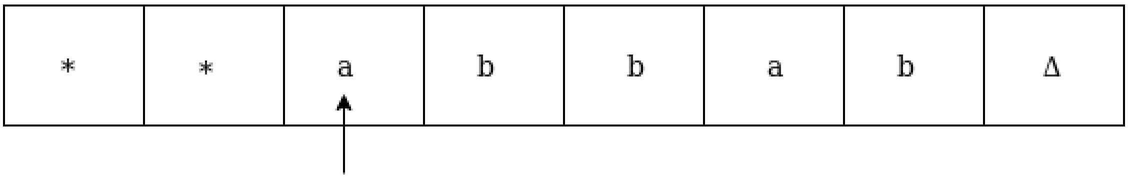


*	b	a	b	b	a	b	Δ

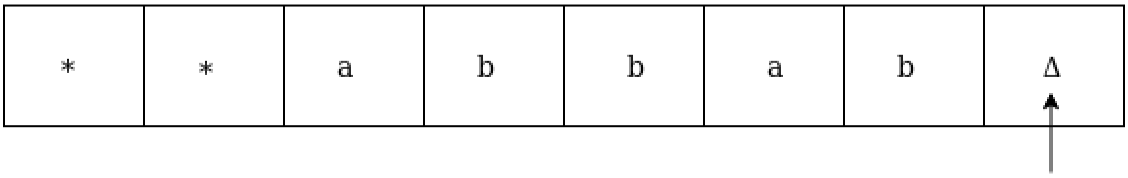
Move right and read it



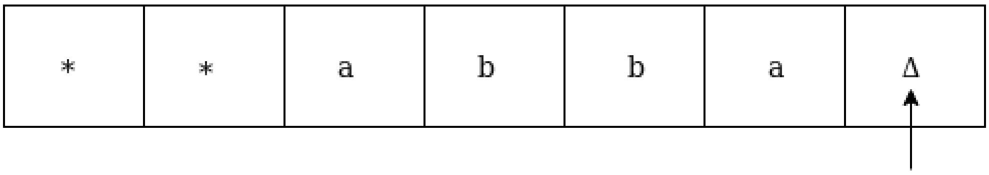
Now convert b by * and move right as:



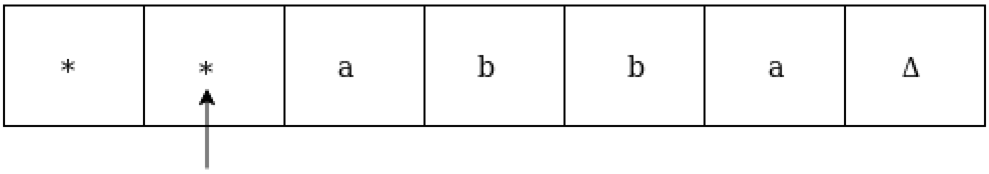
Move right up to Δ in search of b as:



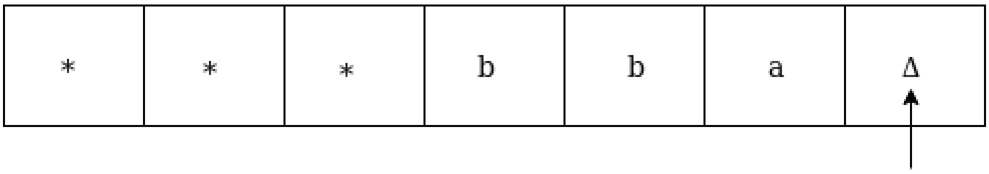
Move left, if the symbol is b then convert it into Δ as:



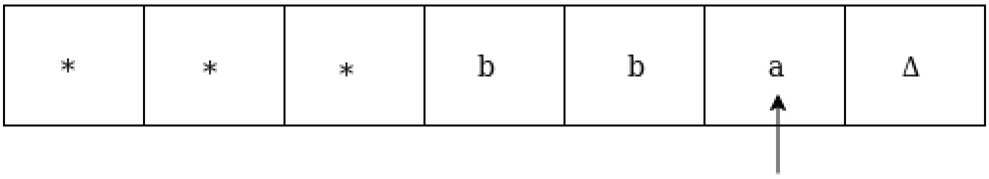
Now move left until * as:



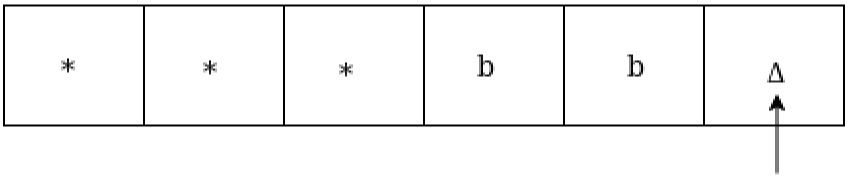
Replace a by * and move right up to Δ as:



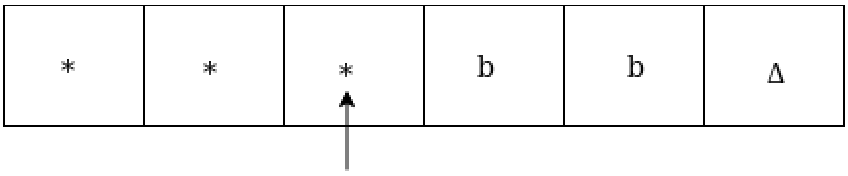
We will move left and check if it is a, then replace it by Δ as:



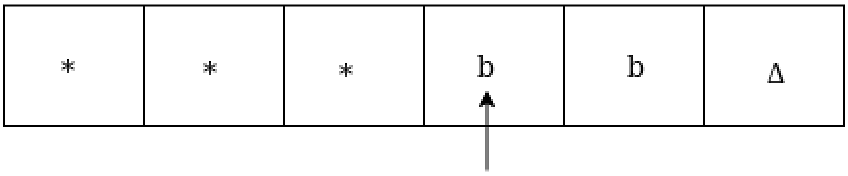
It is 'a' so replace it by Δ as:



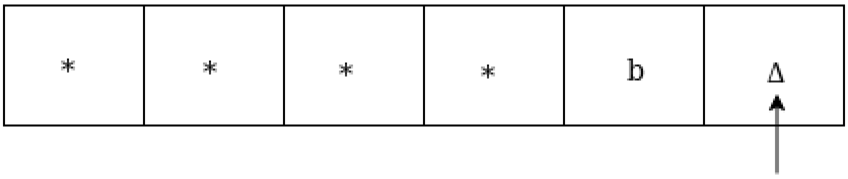
Now move left until *



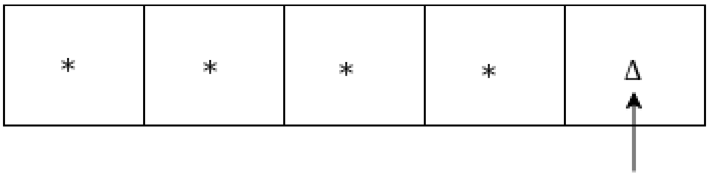
Now move right as:



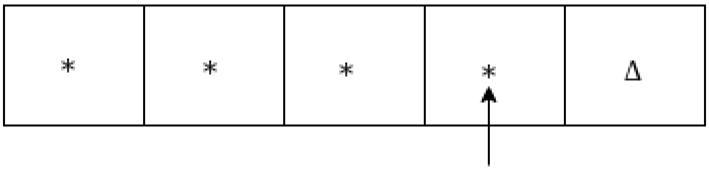
Replace b by * and move right up to Δ as:



Move left, if the left symbol is b, replace it by Δ as:

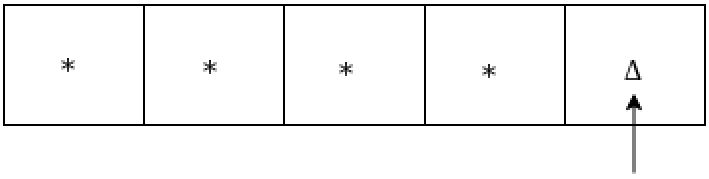


Move left till *



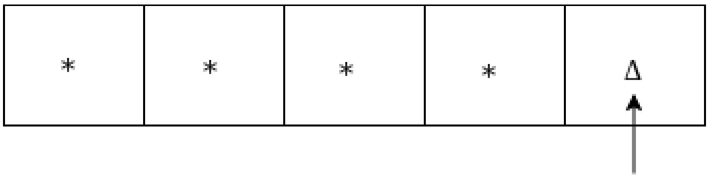
*	*	*	*	Δ

Move right and check whether it is Δ



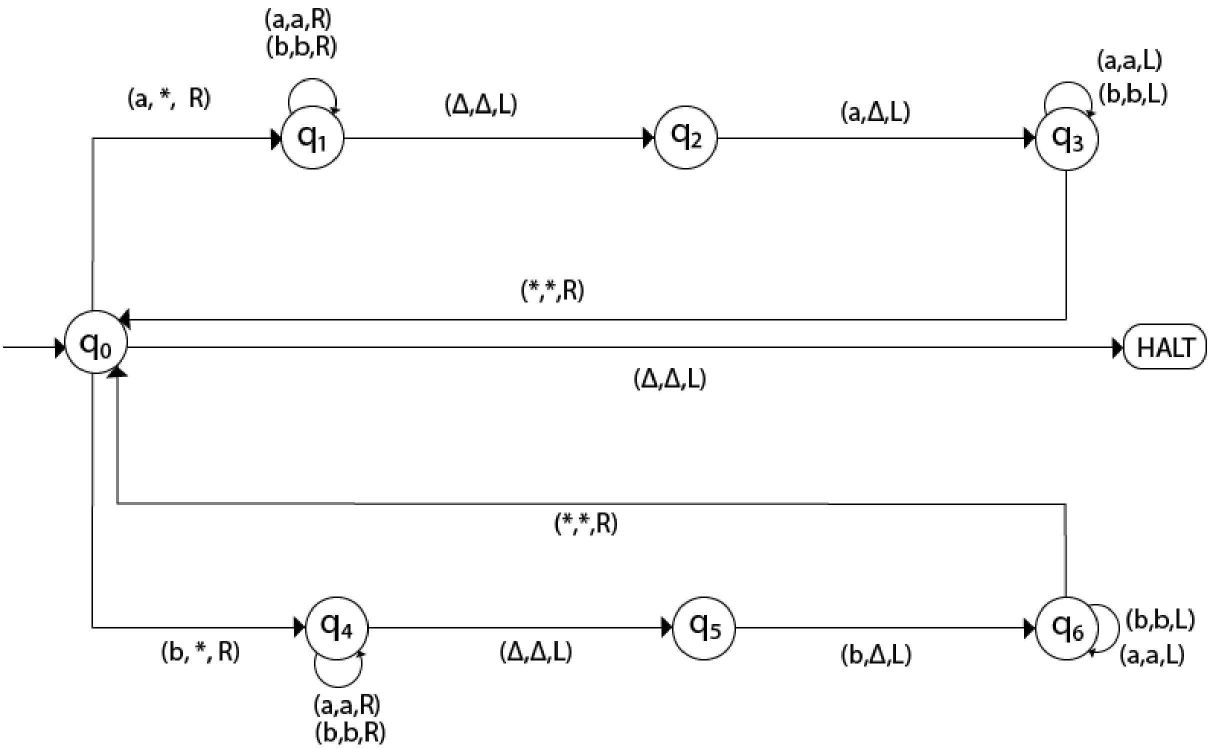
*	*	*	*	Δ

Go to HALT state



*	*	*	*	Δ

The same TM can be represented by Transition Diagram:



Example 3:

Construct a TM machine for checking the palindrome of the string of odd length.

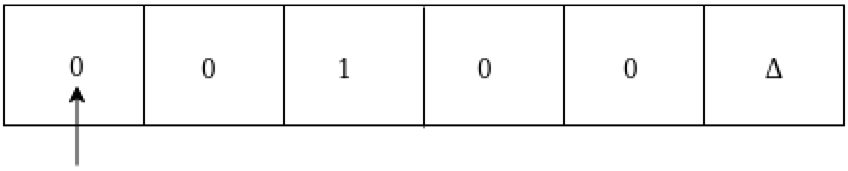
Solution:

Firstly we read the first symbol from left and then we compare it with the first symbol from right to check whether it is the same.

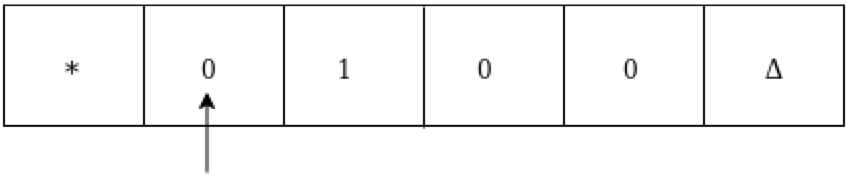
Again we compare the second symbol from left with the second symbol from right. We repeat this process for all the symbols. If we found any symbol not matching, we lead the machine to HALT state.

Suppose the string is 00100Δ. The simulation for 00100Δ can be shown as follows:

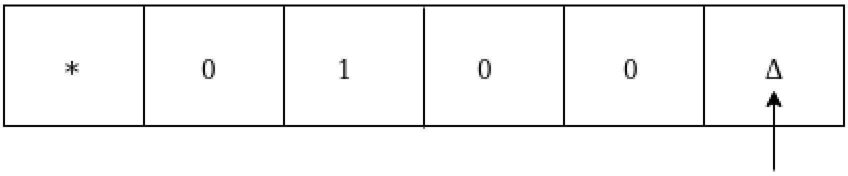
Now, we will see how this Turing machine will work for 00100Δ. Initially, state is q0 and head points to 0 as:



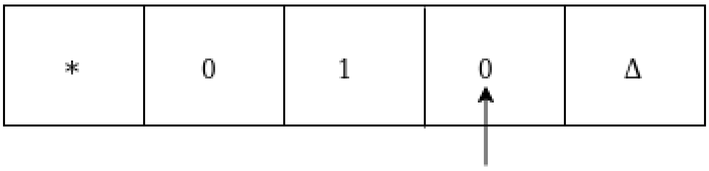
Now replace 0 by * and move right as:



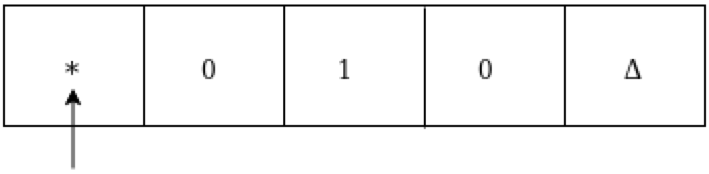
Move right up to Δ as:



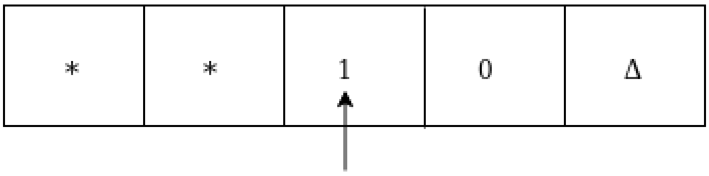
Move left and replace 0 by Δ and move left:



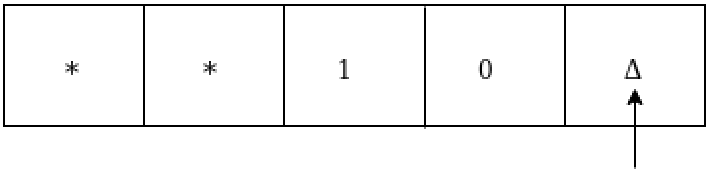
Now move left up to * as:



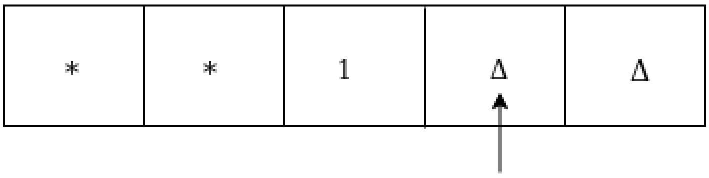
Move right, convert 0 by * and then move right as:



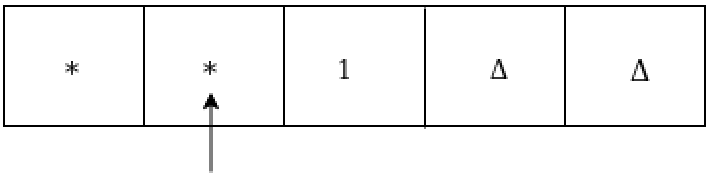
Moved right up to Δ



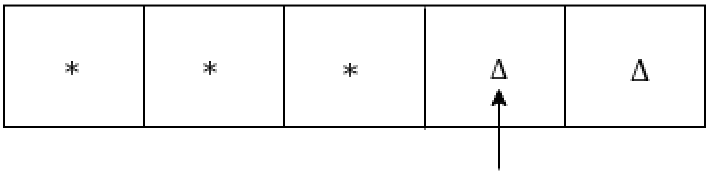
Move left and replace 0 by Δ as:



Move left till * as:



Move right and convert 1 to * as:

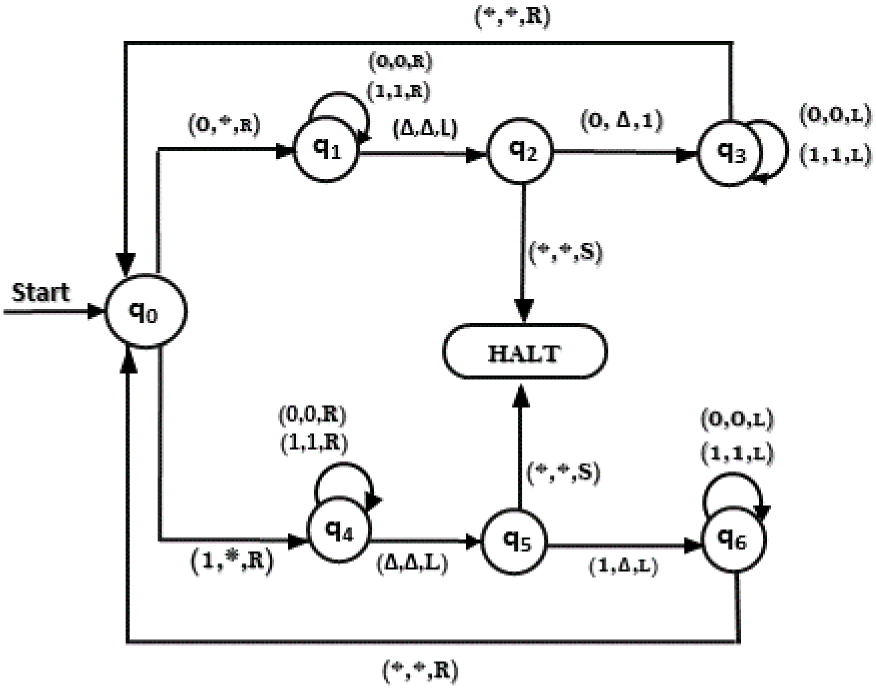


Move left

*	*	*	Δ	Δ

Since it is *, goto HALT state.

The same TM can be represented by Transition Diagram:



Example 4:

Construct TM for the addition function for the unary number system.

Solution:

The unary number is made up of only one character, i.e. The number 5 can be written in unary number system as 11111. In this TM, we are going to perform the addition of two unary numbers.

For example

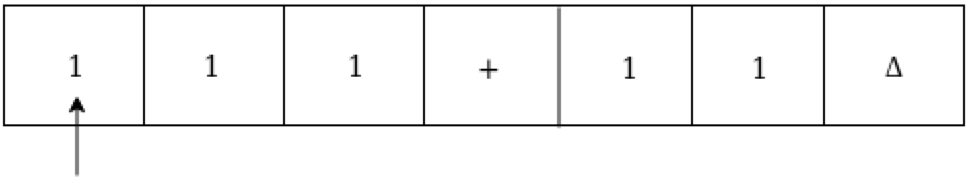
2 + 3
i.e. 11 + 111 = 11111

If you observe this process of addition, you will find the resemblance with string concatenation function.

In this case, we simply replace + by 1 and move ahead right for searching end of the string we will convert last 1 to Δ.

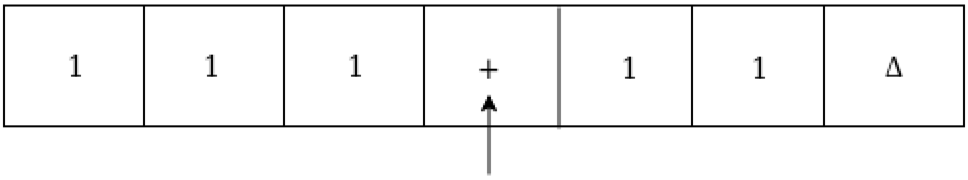
Input: 3+2

The simulation for 111+11Δ can be shown as below:



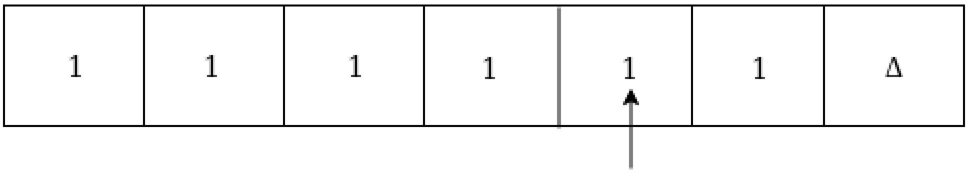
1	1	1	+	1	1	Δ

Move right up to + sign as:



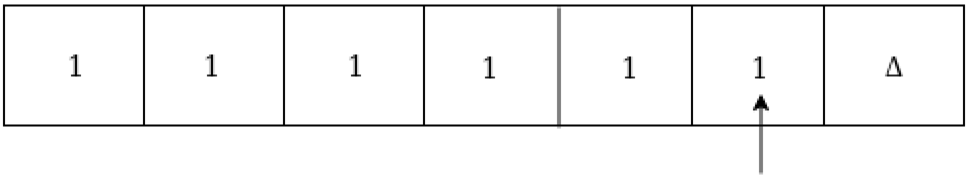
1	1	1	+	1	1	Δ

Convert + to 1 and move right as:



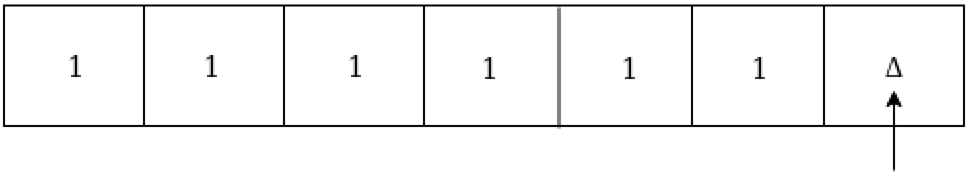
1	1	1	1	1	1	Δ

Now, move right



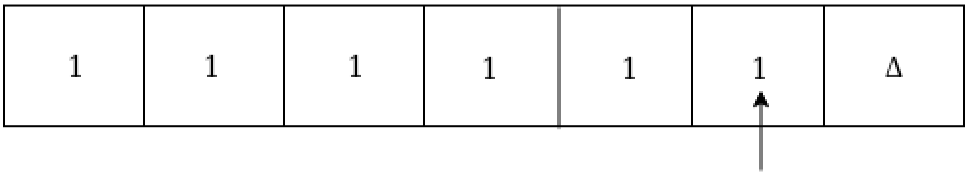
1	1	1	1	1	1	Δ

Again move right



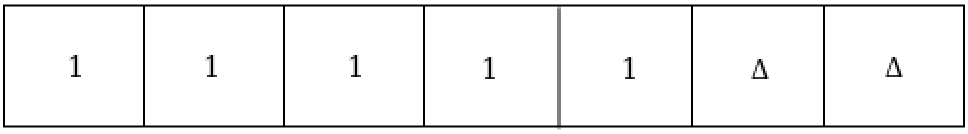
1	1	1	1	1	1	Δ

Now Δ has encountered, so just move left as:



1	1	1	1	1	1	Δ

Convert 1 to Δ

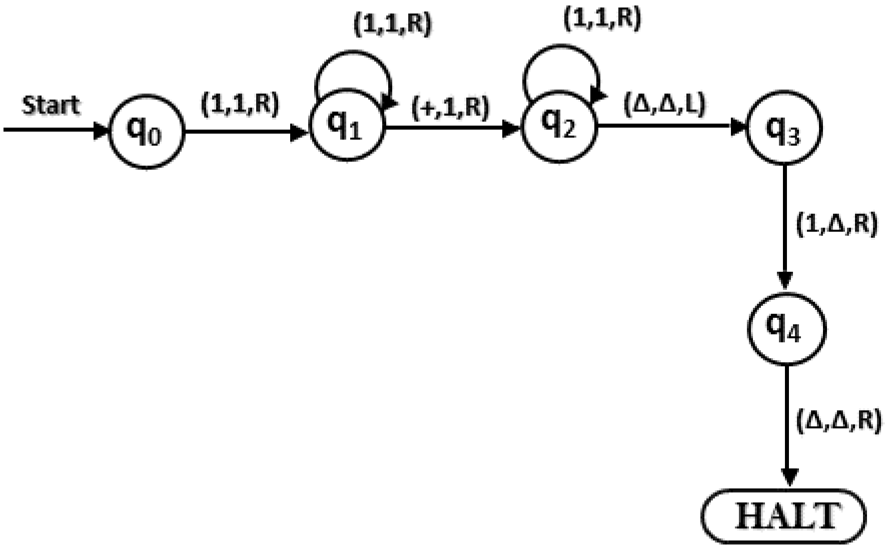


1	1	1	1	1	Δ	Δ

Thus the tape now consists of the addition of two unary numbers.

The TM will look like as follows:

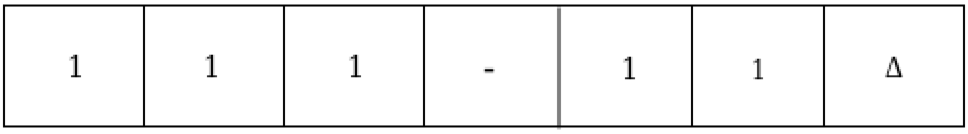
Here, we are implementing the function of f(a + b) = c. We assume a and b both are non zero elements.



Example 5:

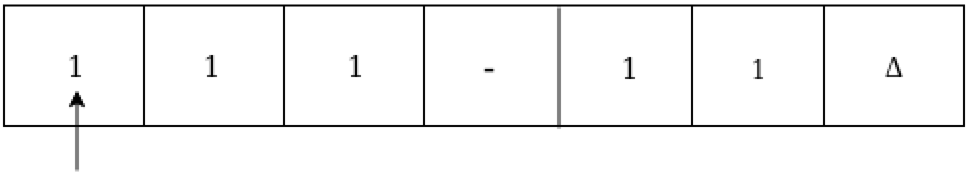
Construct a TM for subtraction of two unary numbers f(a-b) = c where a is always greater than b.

Solution: Here we have certain assumptions as the first number is greater than the second one. Let us assume that a = 3, b = 2, so the input tape will be:



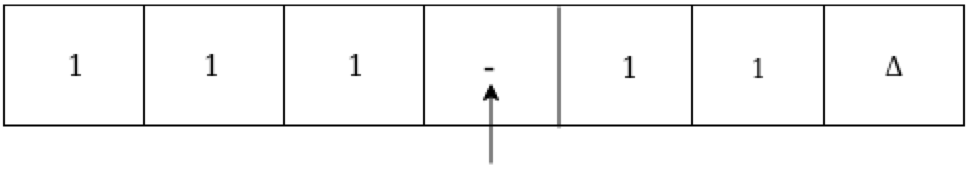
1	1	1	-	1	1	Δ

We will move right to - symbol as perform reduction of a number of 1's from the first number. Let us look at the simulation for understanding the logic:



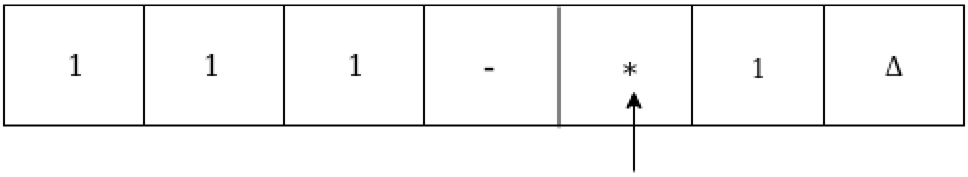
1	1	1	-	1	1	Δ

Move right up to - as:



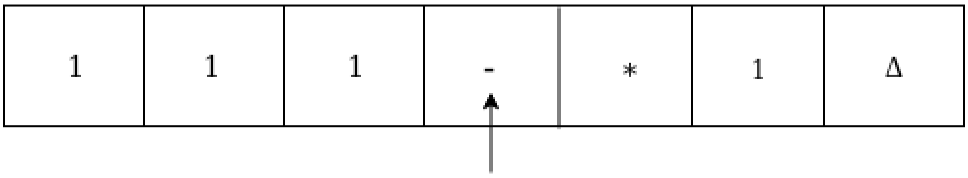
1	1	1	-	1	1	Δ

Move right and convert 1 to * as:



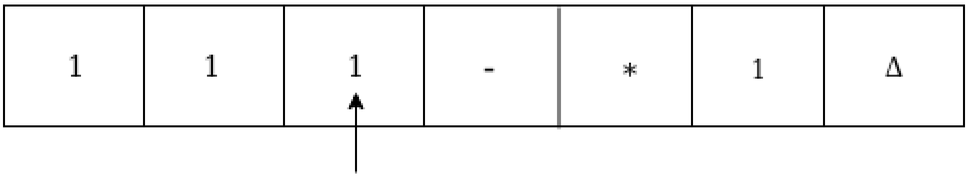
1	1	1	-	*	1	Δ

Now move left



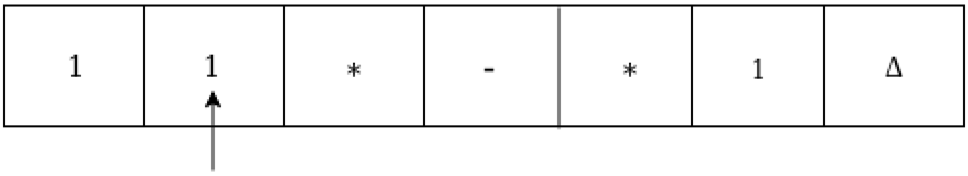
1	1	1	-	*	1	Δ

Again move left

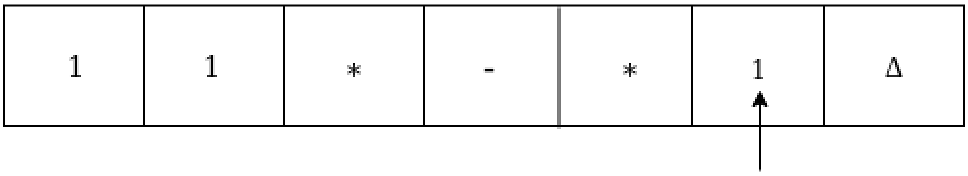


1	1	1	-	*	1	Δ

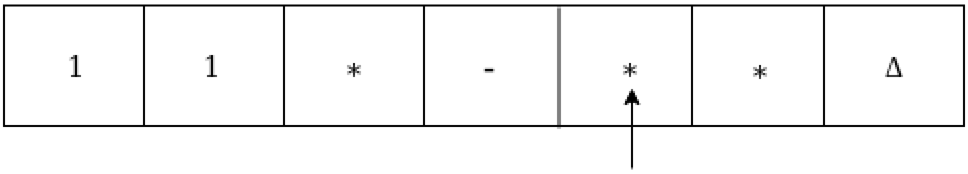
Convert 1 to * and move right-hand



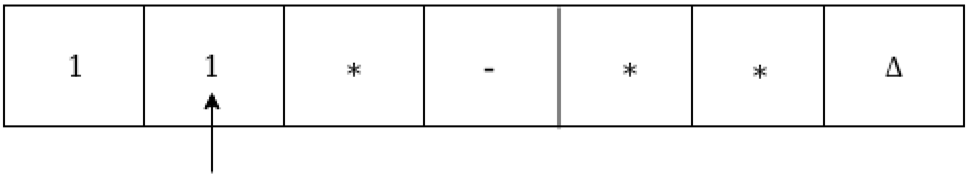
Now move right till 1

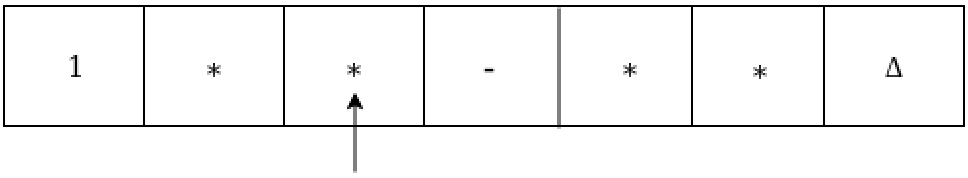


Convert 1 to * and move left

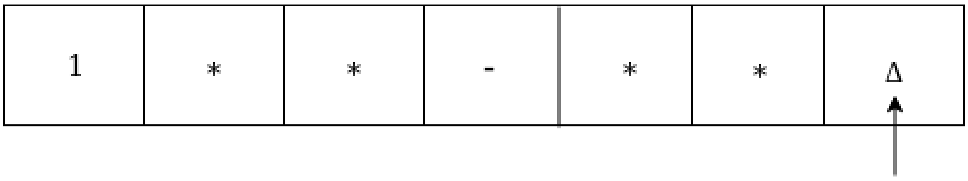


Convert 1 to * and move





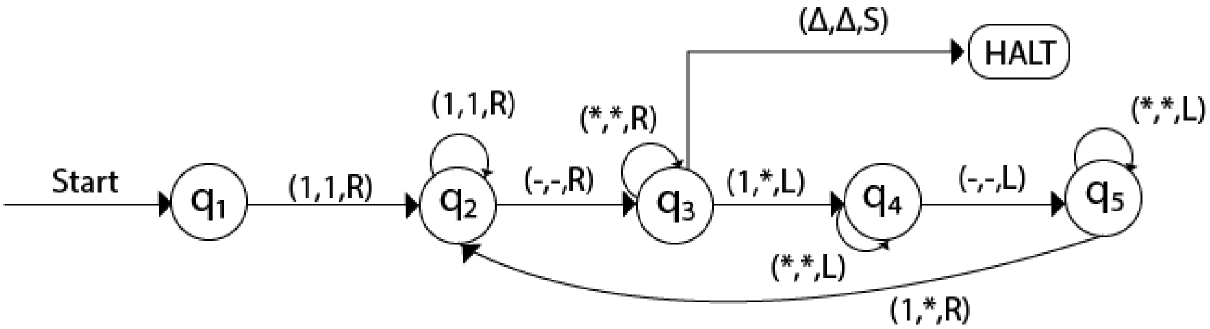
Move right till Δ as:



Now we are in the HALT state.

Thus we get 1 on the input tape as the answer for f(3-2).

The Turing machine will look like this:



Introduction to Undecidability

In the theory of computation, we often come across such problems that are answered either 'yes' or 'no'. The class of problems which can be answered as 'yes' are called solvable or decidable. Otherwise, the class of problems is said to be unsolvable or undecidable.

Undecidability of Universal Languages:

The universal language L_u is a recursively enumerable language and we have to prove that it is undecidable (non-recursive).

Theorem: L_u is RE but not recursive.

Proof:

Consider that language L_u is recursively enumerable language. We will assume that L_u is recursive. Then the complement of L_u that is L`u is also recursive. However, if we have a TM M to accept L`u then we can construct a TM L_d. But L_d the diagonalization language is not RE. Thus our assumption that L_u is recursive is wrong (not RE means not recursive). Hence we can say that L_u is RE but not recursive. The construction of M for L_d is as shown in the following diagram:

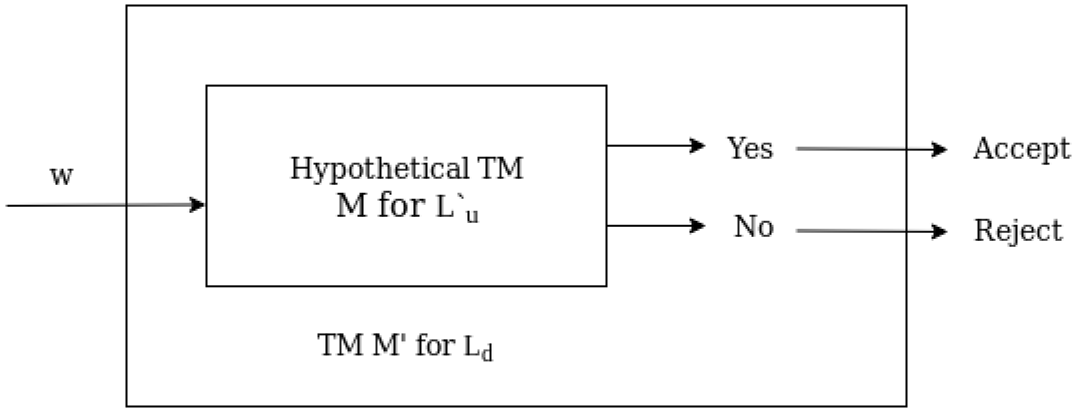


Fig: Construction of M' for L_d

Post Correspondence Problem

In this section, we will discuss the undecidability of string and not of Turing machines. The undecidability of the string is determined with the help of Post's Correspondence Problem (PCP). Let us define the PCP.

"The Post's correspondence problem consists of two lists of string that are of equal length over the input. The two lists are A = $w_1, w_2, w_3, \dots, w_n$ and B = $x_1, x_2, x_3, \dots x_n$ then there exists a non empty set of integers $i_1, i_2, i_3, \dots,$ in such that, $w_1, w_2, w_3, \dots w_n = x_1, x_2, x_3, \dots x_n$"

To solve the post correspondence problem we try all the combinations of $i_1, i_2, i_3, \dots,$ in to find the w1 = x1 then we say that PCP has a solution.

Example 1:

Consider the correspondence system as given below

A = (b, bab^3, ba) and B = (b^3, ba, a). The input set is ∑ = {0, 1}. Find the solution.

Solution:

A solution is 2, 1, 1, 3. That means $w_2w_1w_1w_3 = x_2x_1x_1x_3$

The constructed string from both lists is bab^{3}b^{3}a.

w_2	w_1	w_1	w_3
bab^3	b	b	ba
x_2	x_1	x_1	x_3
ba	b^3	b^3	a

Example 2:

Does PCP with two lists x = (b, a, aba, bb) and y = (ba, ba, ab, b) have a solution?

Solution: Now we have to find out such a sequence that strings formed by x and y are identical. Such a sequence is 1, 2, 1, 3, 3, 4. Hence from x and y list

1	2	1	3	3	4		1	2	1	3	3	4
b	a	b	aba	aba	bb	=	ba	ba	ba	ab	ab	b

Example 3:

Obtain the solution for the following system of posts correspondence problem. A = {100, 0, 1}, B = {1, 100, 00}

Solution: Consider the sequence 1, 3, 2. The string obtained from A = babababb. The string obtained from B = bababbbb. These two strings are not equal. Thus if we try various combination from both the sets to find the unique sequence, we could not get such a sequence. Hence there is no solution for this system.

Example 4:

Obtain the solution for the following system of posts correspondence problem, X = {100, 0, 1}, Y = {1, 100, 00}.

Solution: The solution is 1, 3, 1, 1, 3, 2, 2. The string is

X1X3X1X1X3X2X2 = 100 + 1 + 100 + 100 + 1 + 0 + 0 = 1001100100100
Y1Y3Y1Y1Y3Y2Y2 = 1 + 00 + 1 + 1 + 00 + 100 + 100 = 1001100100100