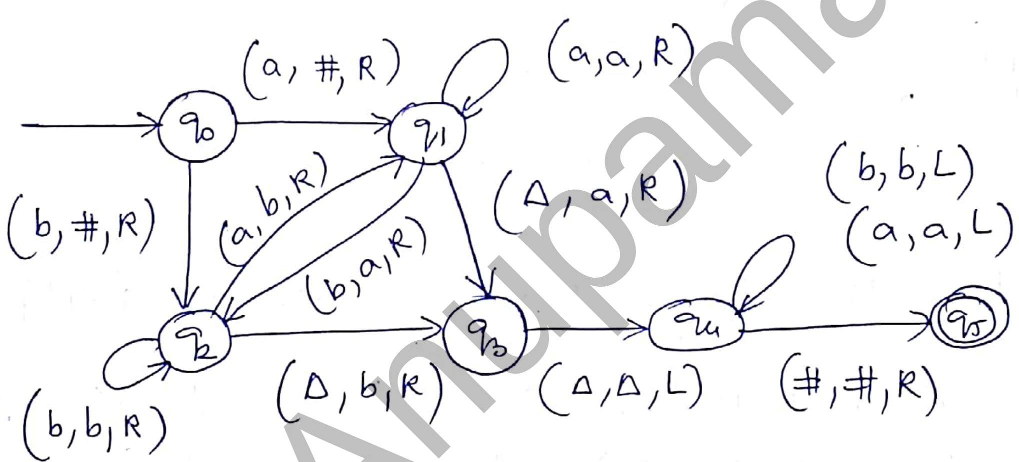
Turing Machines. ①

1. TMachine to shift right an entire string by one cell. and insert a special character '#' as the first character.

Eg: I/p ababbaΔ

O/p #ababbaΔ

logic: This is used to add a special left end marker so complicated logics become simple. At the end the head is at the 1st character after # i.e a.



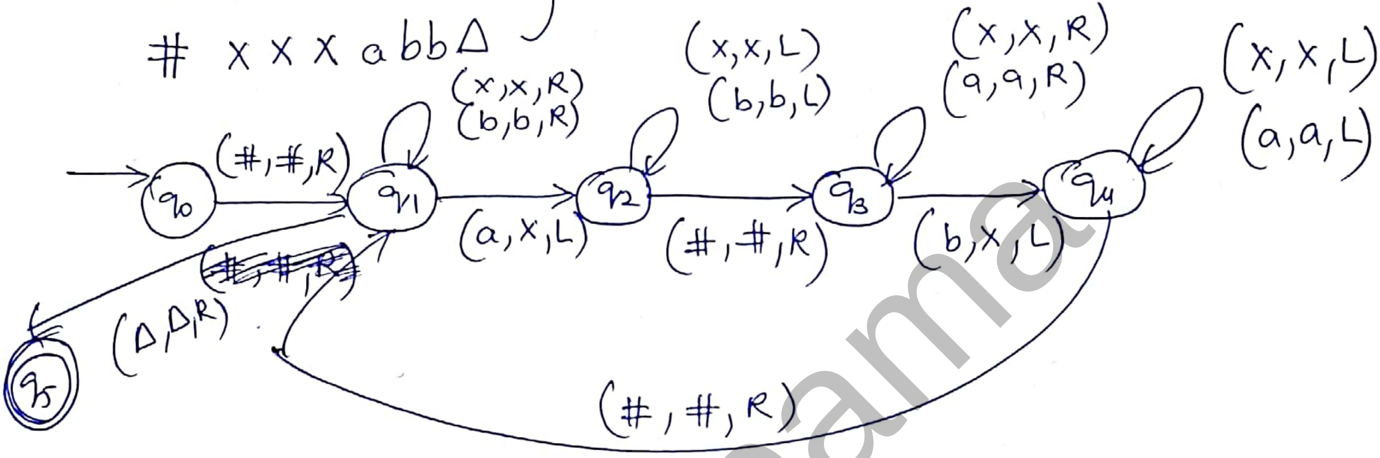
2. Equal no. of a's and b's / $N_a(x) = N_b(x)$

The solution to the above problem is simplified if we assume that the entire string is shifted right by one cell and a special character like # acts as the left end marker. So we first shift right.

logic: We start with a # and move right till an 'a' comes. We mark it with an X and move left till #. On seeing a '#' we leave it as a # and move right, till a 'b' comes. We mark it with an X and move left till #.

We move right till the next 'a' is seen and continue the process and halt when there are no more a's and b's & till a blank is seen.

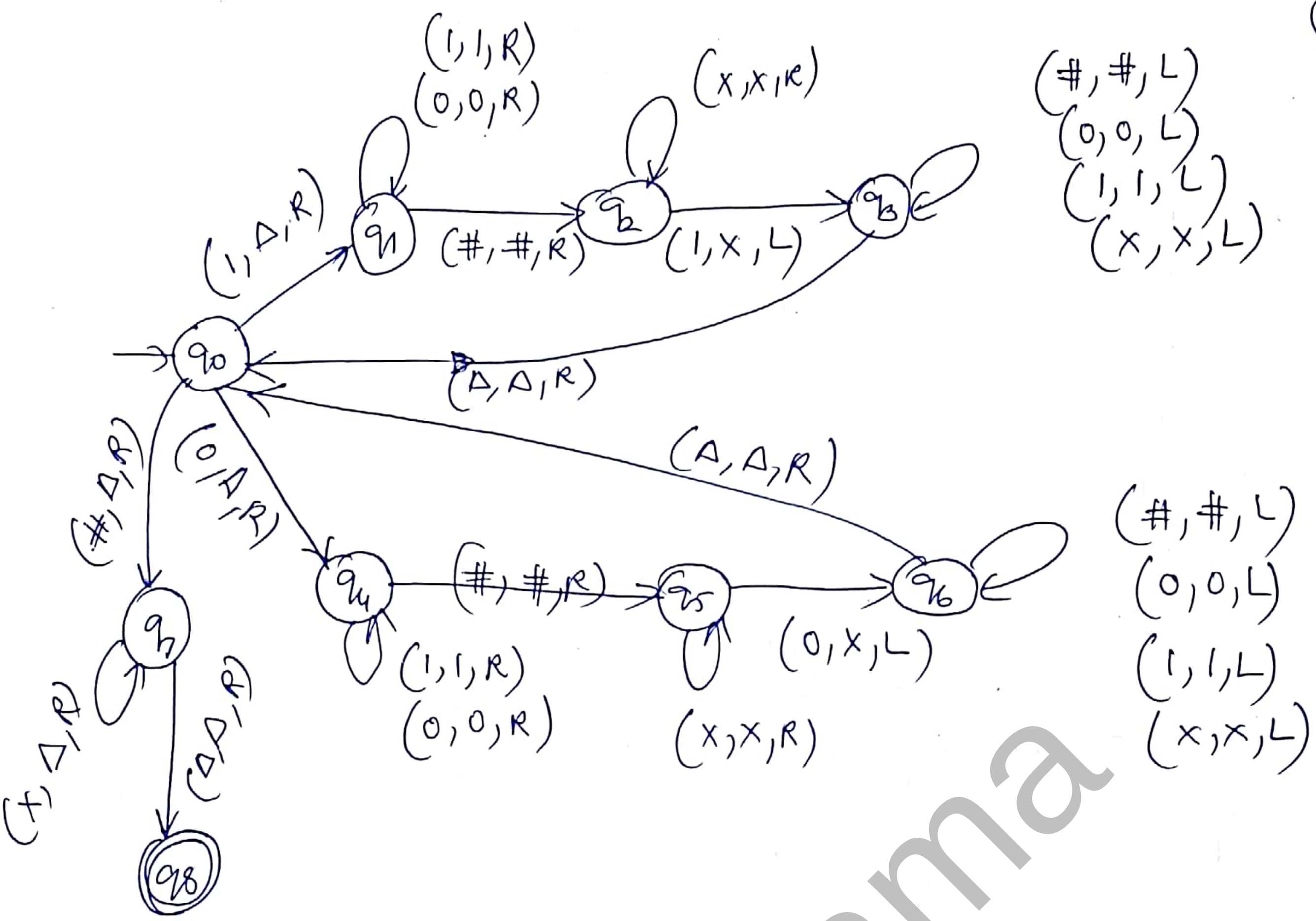
abaabbΔ
xbaabbΔ
XX aabbΔ
XXX abbΔ
XXXaXbΔ
XXXXXbΔ
XXXXXXΔ



3. Tm to accept w#w, where w∈{0,1}

Eg: 1001 # 1001Δ
Δ001 # X001Δ
ΔΔ01 # XX01Δ
ΔΔΔ1 # XXX1Δ
ΔΔΔΔ # XXX1Δ
ΔΔΔΔΔ XXXXΔ
ΔΔΔΔΔ ΔΔΔΔΔ

logic: We make the first character a Δ and move right and cross over #, and make the first character after # an X. Move left till blank. We make the second character a Δ and continue the process. When there are no more 0's and 1's before #, we make the # a Δ move right and make all X's a Δ and halt.

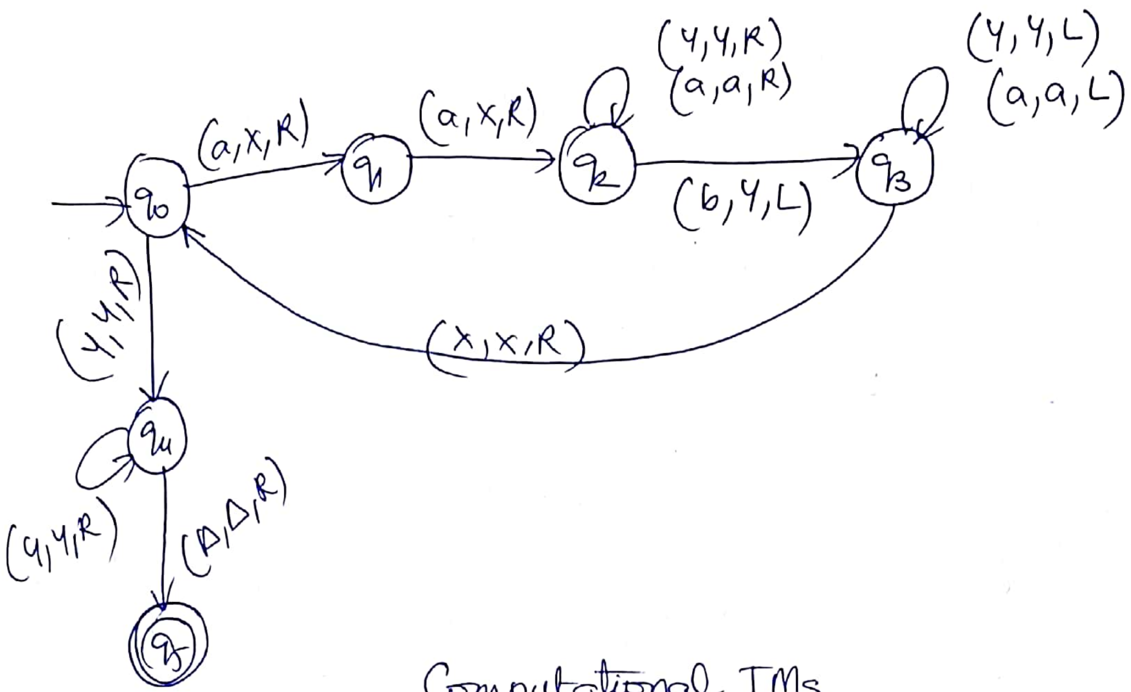


4. $a^{2n} b^n \quad n \geq 1$

Eg: aaaabbΔ

On seeing an 'a' make it an X and move right. On seeing second 'a' make it an X and move right. On any no. of a's leave as a and move right. On seeing a b, make it a Y and move left. On seeing any no. of a's leave it as a 'a' and move left. On seeing an X leave it as X and move right. Repeat the process and break when no more a's are seen. Then cross over all the Y's till blank & Halt.

aaaabbΔ
XaaabbΔ
XXaaYbΔ
XXXaYbΔ
XXXXYYΔ

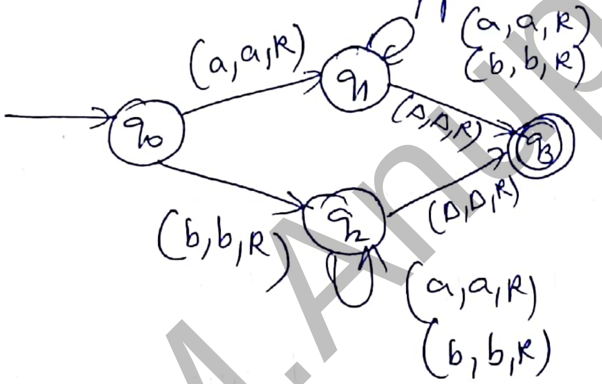


Computational TMs

1. TM for identity function. i.e $f(x) = x$ $\Sigma = \{a, b\}$

I/p : a a Δ   I/p : a b a Δ

O/p : a a Δ   O/p : a b a Δ

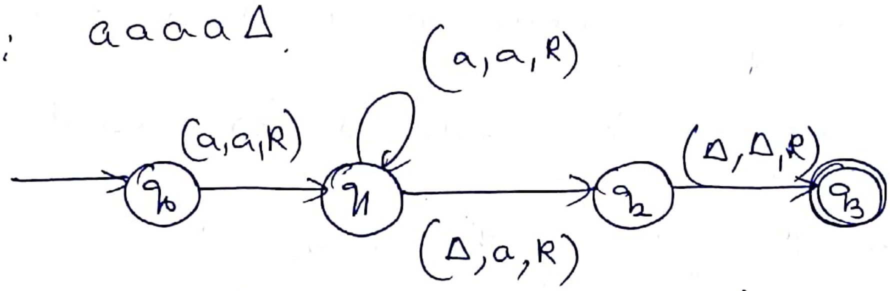


On seeing an a, or b leave it as the same character and move rt till Δ.

2. Successor function. $f(x) = x+1$ $\Sigma = \{a\} \mid \{b\} \mid \{0\} \mid \{1\}$

I/p : aaaΔ

O/p : aaaaΔ.



On seeing an 'a' leave it as an 'a' and move rt.

On seeing any no. of a's leave as a's and move rt.

On seeing a Δ, make it as an 'a' and halt ons

3. TM for 2n.

Eg: I/p : aaa Δ (3 a's)

O/p : AAAAAA Δ (6 A's)

logic: On seeing "a" make it A and move right. On seeing any no. of a's move right leaving them as a's till Δ. On seeing Δ, make it a B and move left till A. Move rt making the second 'a' an A and repeat the process. When no more a's are seen make all B's as A's and move rt and halt on Δ.

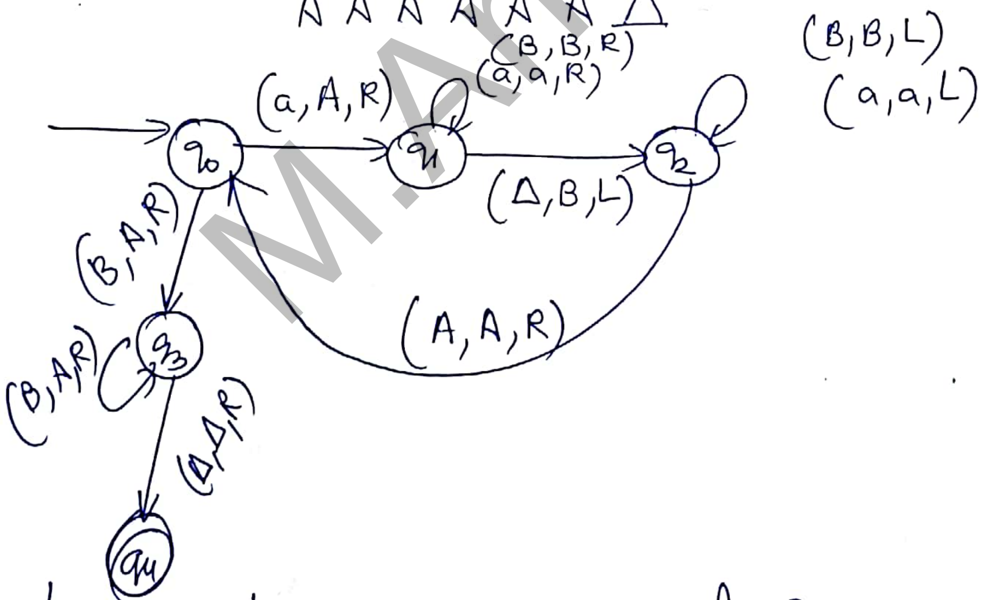
a a a Δ Δ Δ Δ

A a a B Δ Δ Δ

A A a B B Δ Δ

A A A B B B Δ

A A A A A A Δ



4. $\log_2 n$ where n is a power of 2.

Eg: let n = 8, $\log_2 n = \log_2 2^3 = 3$ is the answer.

When n is converted to binary we get: n = 8 = 1000

i.e no. of 0's present after 1 in 1000 i.e 3 is the answer

1000Δ

Δ000Δ No. of 0's on the tape is the answer.

logic: We replace the 1st 1 with a Δ and no. of 0's present on the tape is the answer.

Eg: $n = 16$

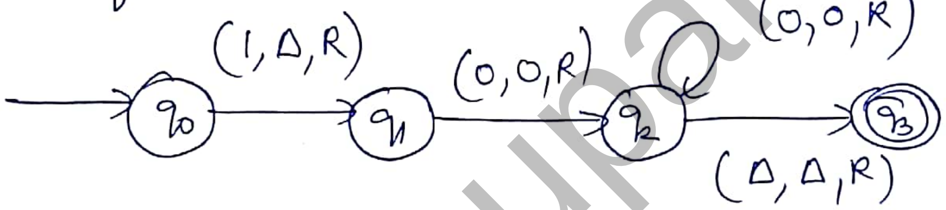
$\log_2 2^4 = 4$

n is binary is 10000Δ

Δ0000Δ

4 is the answer since 4 0's are on the tape.

The tape can be attached to a counter which counts the no. of 0's.



~~E~~ <u>Reasons for TM not accepting input:</u> (3 m)

~~A TM~~ or <u>TM as an Acceptor.</u>

A TM halts when it no longer has any other moves. If it halts in a final state it accepts the input otherwise it rejects the input. There are 4 cases of rejecting the input string.

1. The TM halts in a non final state.
2. The TM never halts (stops) i.e it goes into an infinite loop.
3. When no transition is specified for the current configuration.
4. The head is at the left end of the tape and ~~we are~~ it is asked to move left.

When none of these 4 cases occur TM is an Acceptor.

Church's Hypothesis (3m)

It is also called church - turing thesis. According to church's hypothesis which can be identified by humans can be computed by a turing machine. Hence T.M is believed to be the ultimate computing Machine. In other words no computational procedure will be considered an algorithm unless it can be represented as a T.M.

The assumption that the notation of computable fns that can be identified with a class of partial recursive fns is known as CH.

T.M to multiply two unary no's separated by a delimiter 1 (Copy fn)

7

Eg: I/P is given as $0^m 1 0^n 1$

$m = 2, n = 4$

I/P 0010000 1

O/P $\underbrace{BB00000000BB}_{m \times n}$

Let x & y be two unary nos such that x has m no of zeros & y has n no of zeros separated by one & return of tape as $0^m 1 0^n 1$. The product of $m \times n$ is stored on the tape after the one & o/p on the tape is $m \times n$ zeros. The main logic is that we copy second set of zeros (n), m times on to right repeatedly m n

0010000 1 Δ Δ B means Δ

B01 x0001 0 Δ

B01 x x001 0 0 Δ

B01 x x x01 0 0 0 Δ B01 x x x 0 1 0 0 0 0 Δ

B01 x x x x1 0 0 0 0 Δ

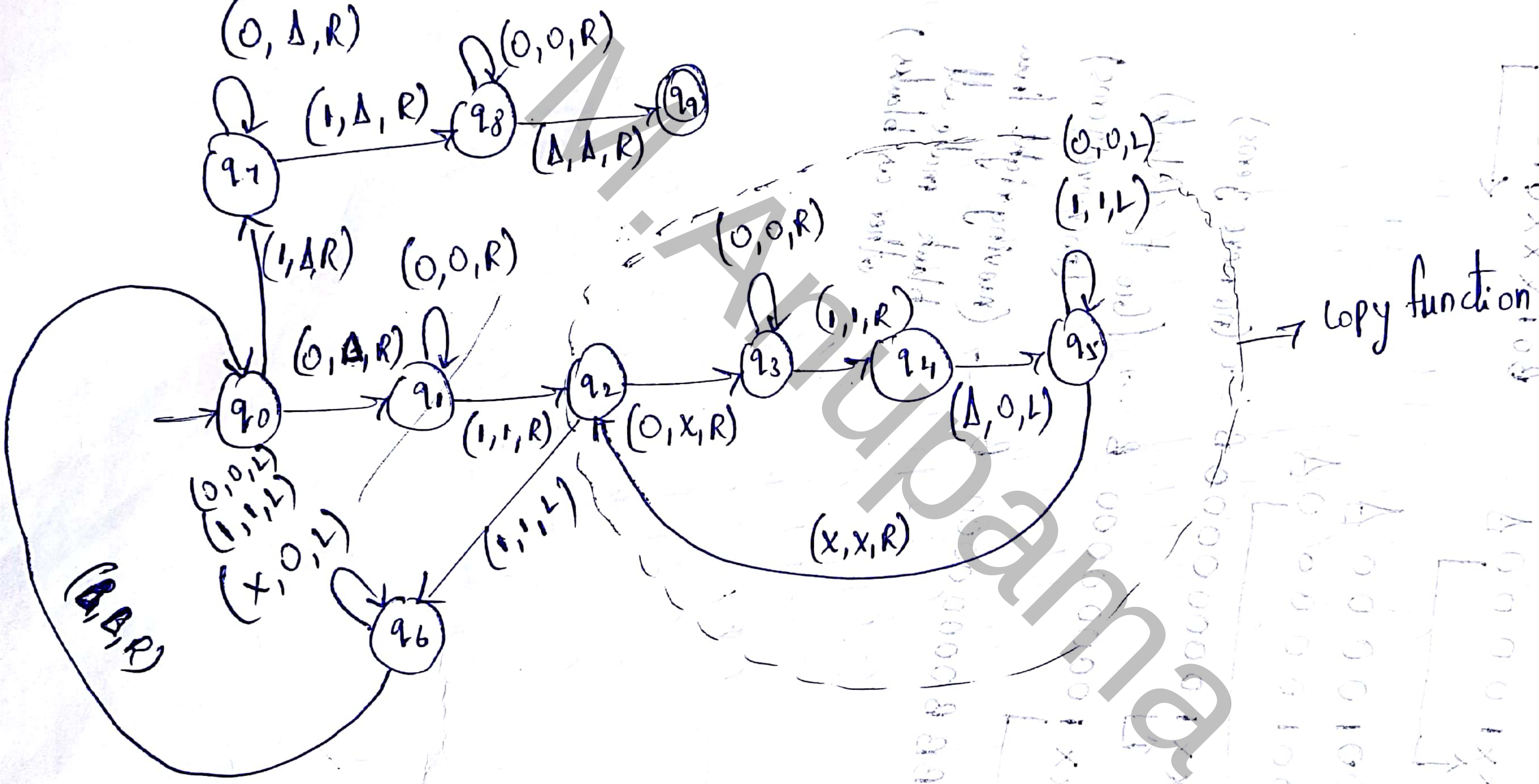
B010 0 0 01 0 0 0 0 Δ

BB1 x 0 0 0 1 0 0 0 0 0 Δ

BB1 x x x x 1 0 0 0 0 0 0 0 0 B → (no more zeros)

BB1 x x x x 1 0 0 0 0 0 0 0 0 B → (Go to Blank)
(no more zeros)

BBBBBBBB 0 0 0 0 0 0 0 0 B → (moving right from blank make all other as blanks)


(0, Δ, R)
(1, Δ, R)
(0, 0, R)
(Δ, Δ, R)
q7
q8
q9
(1, Δ, R)
(0, 0, R)
(0, Δ, R)
q0
q1
q2
(1, 1, R)
(0, X, R)
(0, 0, R)
q3
(1, 1, R)
q4
q5
(Δ, 0, L)
(0, 0, L)
(1, 1, L)
(X, X, R)
(0, 0, L)
(1, 1, L)
(X, 0, L)
(1, 1, L)
q6
(B, B, R)
Copy function

T.M as a Transducer (bitquestion)

A T.M is made to act as a transducer when it is treated as follows:-

We treat the entire non blank portion of the initial tape as I/P & we treat the entire non blank portion of the tape when machine reaches the final state as o/p.

Eg: $0^m 1 0^n 1$

I/P B 0010 0001 B

o/p BB 0000000 BB

o/p

T.M for Substraction (8)

Given m-n if m>n. ans is m-n

If m≤n ans is zero

Consider the I/P string $a^4 b a^2 \Delta$ m>n

m=4 n=2

aaaab a a Δ

A a a a b a a Δ

A a a a b a Δ Δ

A a a Δ b a Δ Δ

A a a Δ b Δ Δ Δ

A a a Δ b Δ Δ

A a Δ Δ b Δ Δ

a a Δ Δ Δ Δ Δ

4-2=2 a's

$m \le n$

$m=2 \quad n=4$

a a b a a a a Δ
Δ a b a a a a Δ
Δ a b a a a Δ Δ
Δ Δ b a a a Δ Δ
Δ Δ b a a Δ Δ Δ
Δ Δ Δ Δ Δ Δ Δ Δ

The first set of a's are completed, second set of a's are remaining. So when we move to the left of b there are no more small a's, so move left till a Δ ~~move~~ make it a blank move right make 'b' a blank move right make a's as blank & halt on Δ.