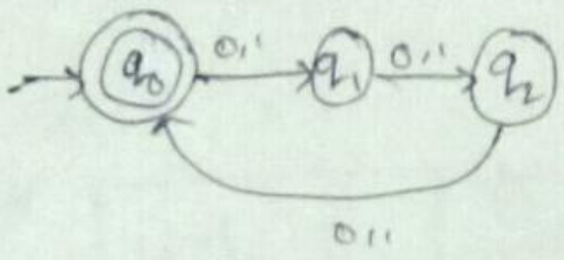
29/1/...

Theory of Computation :

Q:

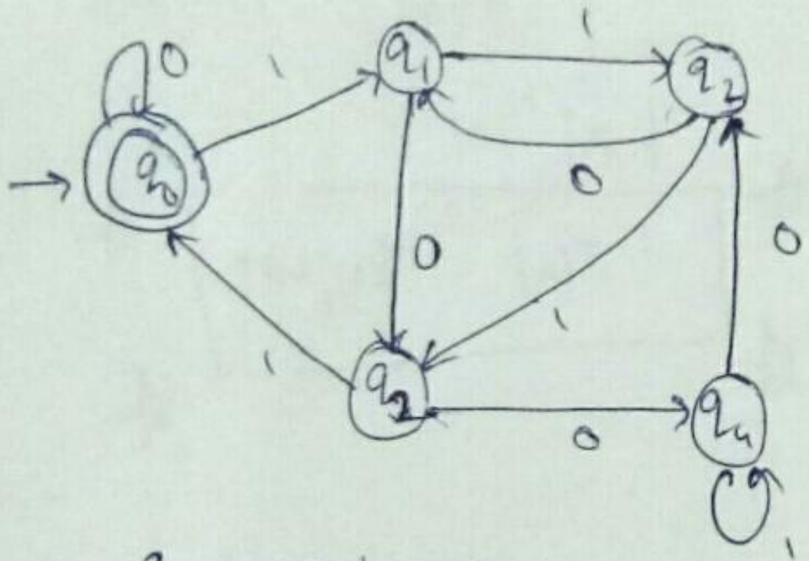
* Design a DFA to recogonite set of all strings whose length is exactly divisible by 3 over alphabet $\Sigma = \{0,1\}$



$L = \{\varepsilon, 000, 111, 010, 000111000, 011100 \ldots\}$

Q: Design a DFA to recogonize set of all strings over $\{0,1\}$, here the binary numbers are interpreted as decimal integers. The machine should accept all the strings who decimal value is divisible by 5.

$L = \{0, 101, 1010, 1111, 10100, 11001, 11110, 100011$



q_0 - mods - 0

q_1 - mods - 1

q_2 - mods - 2

q_3 - mods - 3

q_4 - mods - 4

q_1 : 10 mods - 2 - q_2

11 mods - 3 - q_3

q_2 : 100 mods - 4 - q_4

101 mods - 0 - q_0

q_3 : } silly

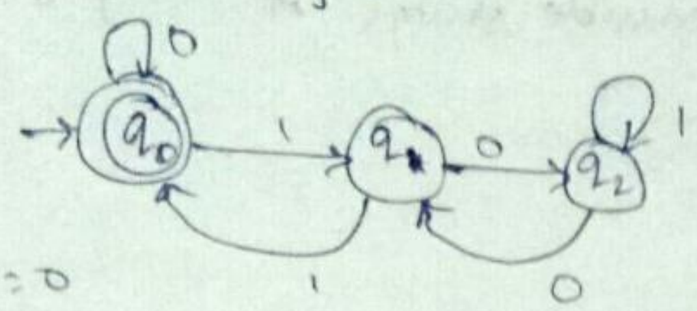
q_4 : }

imp

	0	1
$\rightarrow q_0^*$	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

Pattern

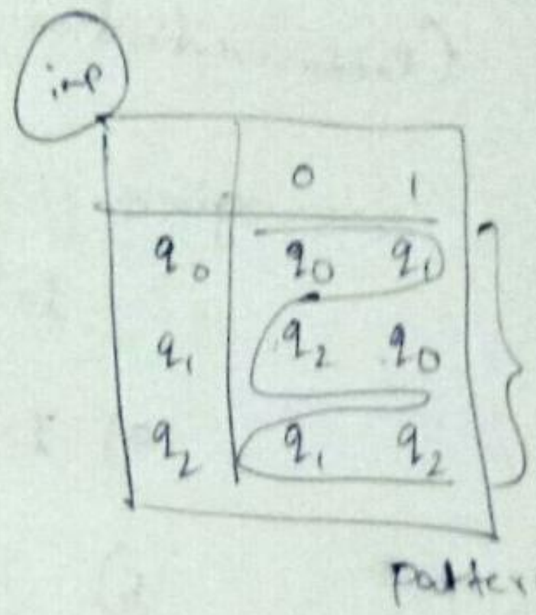
silly for divisible by 3



q_0 : 0 mod 3 = 0
q_1 : 1 mod 3 = 1
q_2 : 2 mod 3 = 2

q_1 : 10 mod 3 = 2 $\Rightarrow q_2$
11 mod 3 = 0 $\Rightarrow q_0$

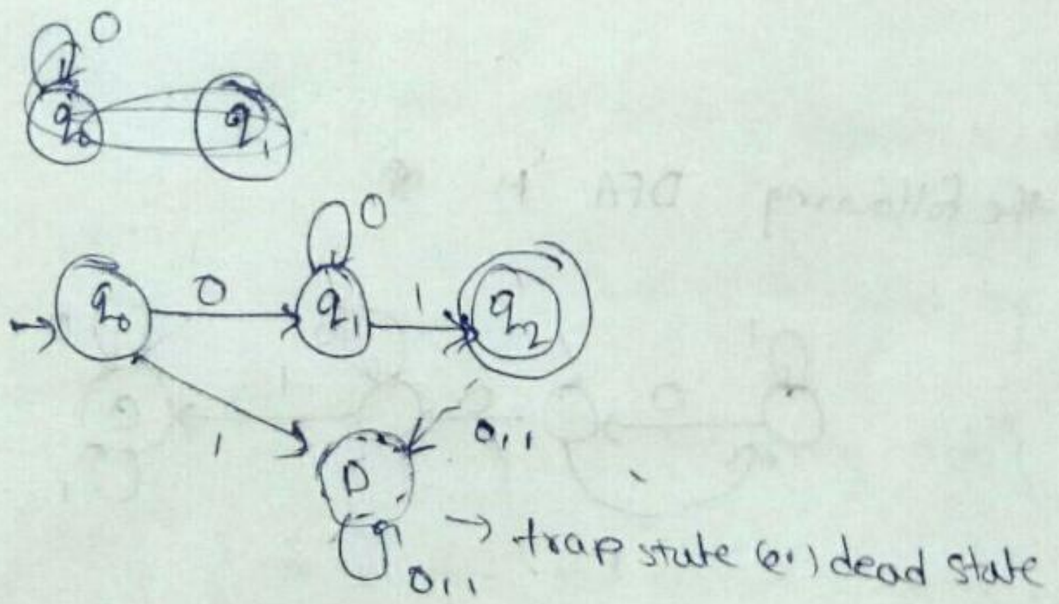
q_2 : 101 mod 3 = 2 $\Rightarrow q_2$
100 mod 3 = 2 $\Rightarrow q_1$



Q: Construct at DFA for the lang

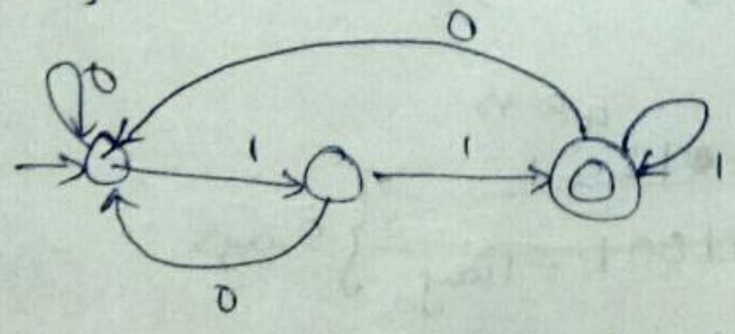
$L = \{0^n 1 / n > 0\}$

$L = \{01, 001, 0001 ..\}$



Previous Questions:

Consider the following DFA



The M accepts all strings over $\{0,1\}$

a) begin with 0, end with 1
b) ending 1
c) substring 11
d) ending 11

for this type of questions list possible strings of long given and option

(Random verification may go wrong)

given

$L = \{11, 111, 1011, 1011, 011 \ldots\}$

a) $L = \{01 \ldots\}$ ✗

b) $\{1, \ldots\}$ ✗

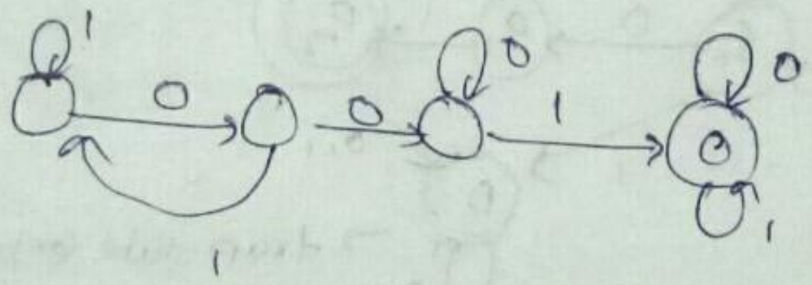
c) $\{11, 011, 110\}$ ✓ ✓ ✗

d) $\{11, 011, 111, 0011, 01011 \ldots\}$ ✓ ✓ ✓ ✓ ✓

∴ ending with 11.

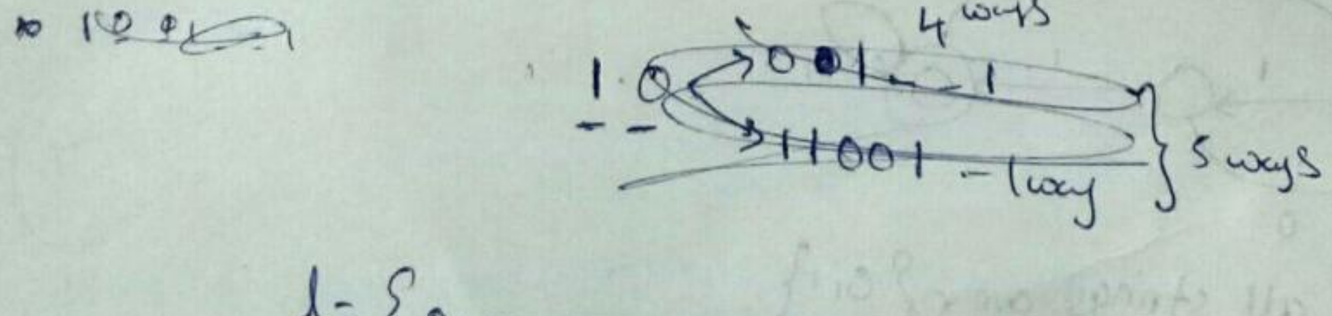
(2003)

Consider the following DFA 'M'



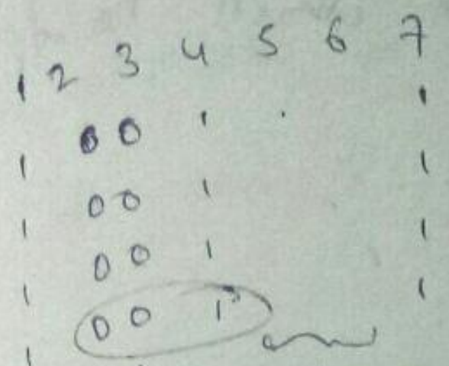
Let S denote set of 7 bit binary strings in which the 1st, 4th and last bits are 1. The no of strings in S that are accepted by M is

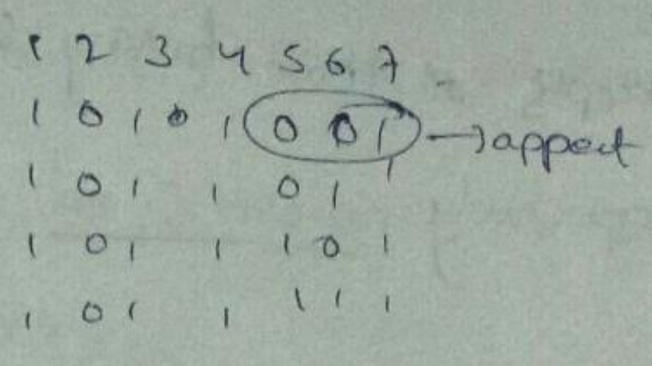
a) 1 b) 5 c) 7 d) 8



$L = \{\underline{001}, 1\underline{001}, 01\underline{001}, 010\underline{010} \ldots\}$

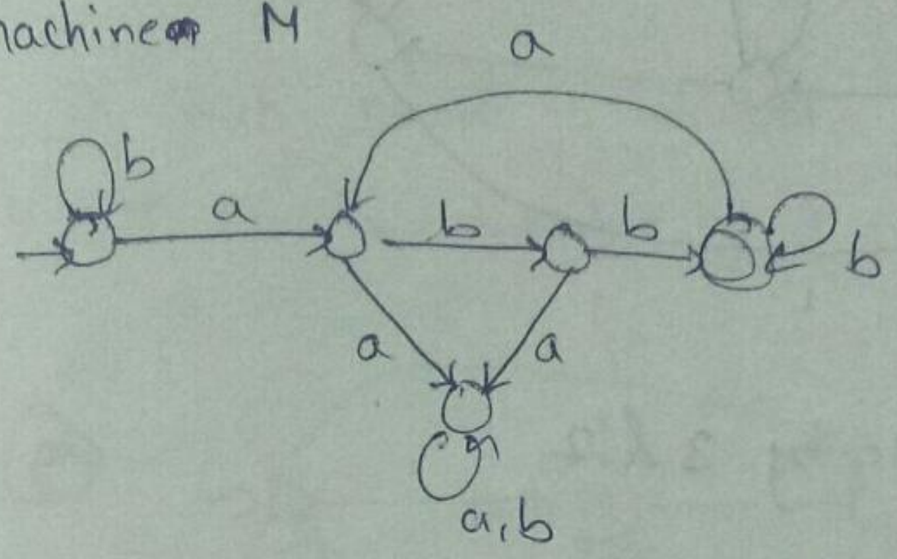
Observing above automata, it accepts ~~sets~~ strings with substring '001'.





$4 + 1 + 1 + 1 = 7$

Ex: Consider the machine M



The lang recogonized by M is

a) $\{ w \in \{a,b\}^* \mid$ every a in w is followed by exactly 2 b's $\}$

b) $\{ w \in \{a,b\}^* \mid$ every a in w is followed by atleast 2 b's $\}$

c) $\{ w \in \{a,b\}^* \mid w$ contains the substring abb $\}$

d) $\{ w \in \{a,b\}^* \mid$ w does not contain aa as substring $\}$

for given

$L = \{$ abb, babb, abbb, bubbb, abbabb,

for (a)

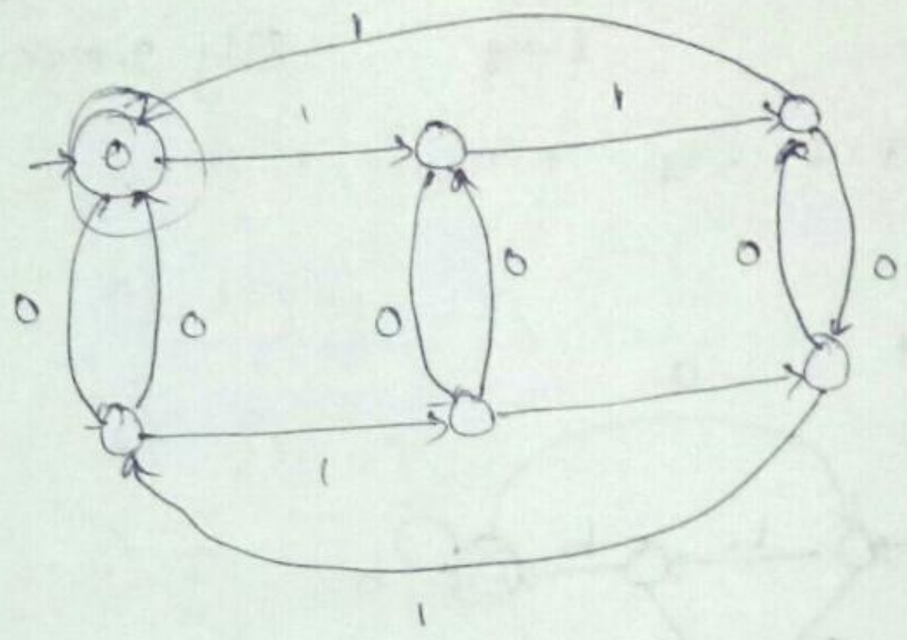
$L = \{$ abb, abbb ... $\}$ ✗

(b) $L = \{$ abb, babb, bbabb, babbbabb, abb ... $\}$ ✓

(c) $L = \{$ abb, aabb .. $\}$ ✗

(d) $L = \{ \epsilon, a, b, ab, aba \ldots \}$ ✗

Q: The following FSM accepts all those binary strings in which no of 1's & 0's are respectively _____



a) divisible by 3 & 2

b) odd & even

c) even & odd

d) divisible by 2 & 3

for given

$L = \{00, 111, 11100, 00111, 10011, 10110, 1001001 \ldots\}$

for (a) $L = \{00, 111, 11100, 11001, 10011, 01110 \ldots\}$ ✓

(b) $L = \{1_{x} \ldots\}$ ✗

(c) $L = \{11_{x} \ldots\}$ ✗

(d) $L = \{11_{*}, 000_{*} \ldots\}$ ✗

If no of 1's mod 3 = 1 & no of 0's divisible by 2 the in above fig we take

☆

Note:

for 1's divisible by 3 & 0's divisible by 2

remainder combinations are

(0,0) (1,0) (2,0)

(0,1) (1,1) (2,1) i.e., 6 states

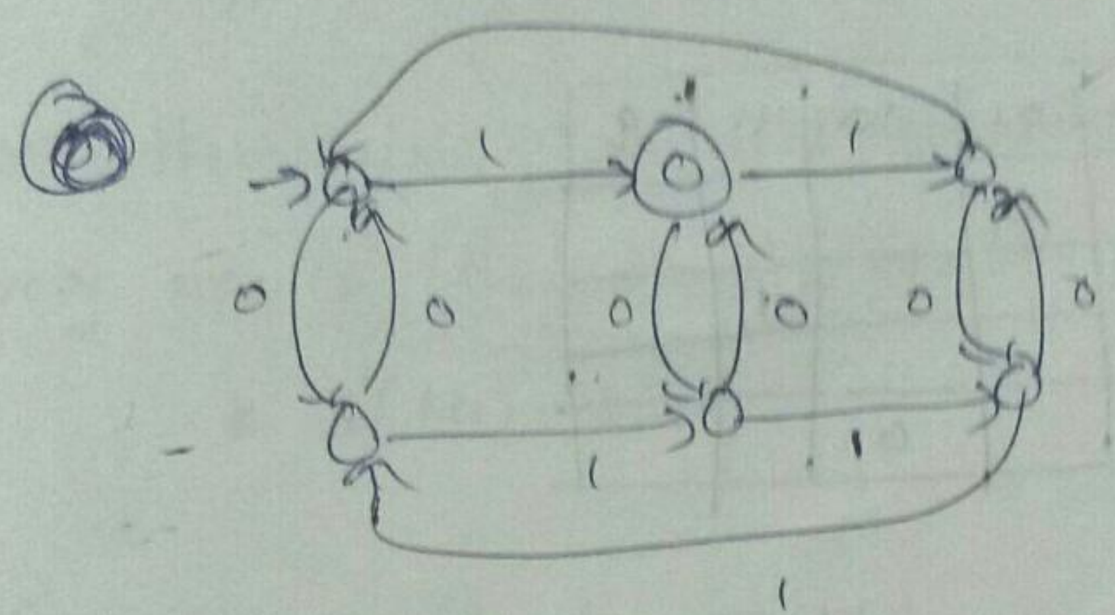
i.e., 3 × 2 = 6 states

→ The automata in which accepts string containing 1's divisible by 5 and 0's divisible by 3 requires 15 states

i.e., $5 \times 3 = 15$

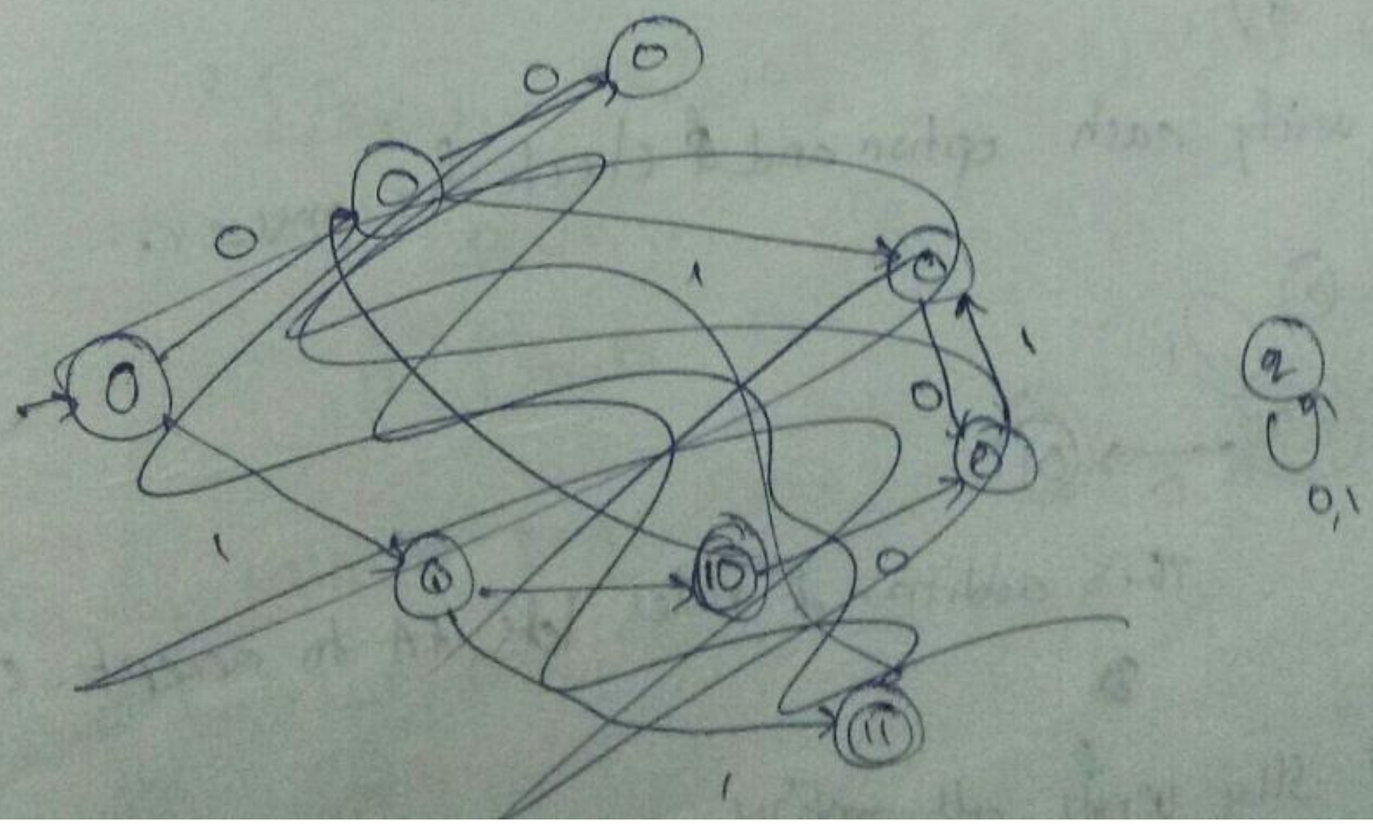
(0,0) (1,0) (2,0) . . .
(0,4) (1,1) (2,1) . . .
(0,2) (1,2) (2,2) . . .

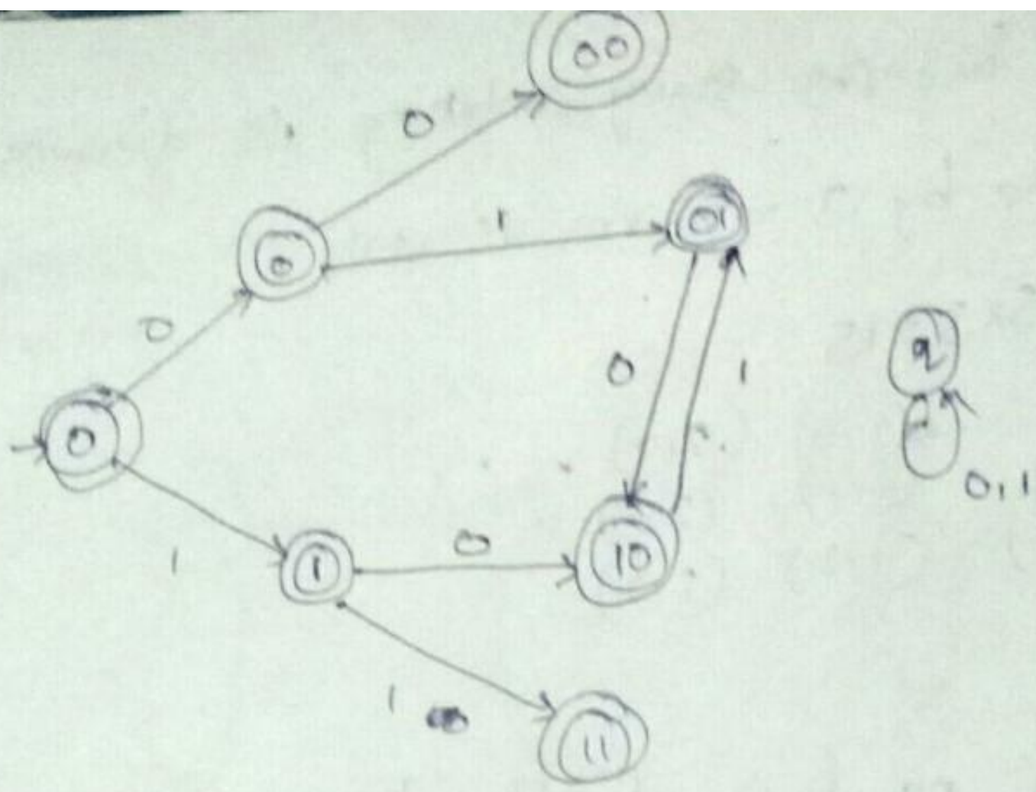
→ Design DFA in which no of 1's divisble by 3 gives remainder 1 and no of 0's is divisible by 2



(2012)

Q) Consider the set of strings on {0,1} in which every substring of 3 symbols has atmost two 0's. For eg 00110 and 011001 are in the language. But 100010 is not in the lang. All the strings of length less than 3 are also in the lang. Consider below automata





Find missing transitions

a)

	00	01	10	11	q
00	1	0			
01				1	
10	0				
11			0		

b)

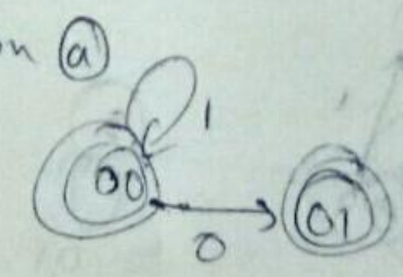
	00	01	10	11	q
00					
01		0			1
10		1		0	
11		0		1	

~~Here we~~ c)

d)

Here verify each option and check for answer.

In option (a)



This addition of rules lets FA to accept 000 X.

(b) Silly verify all options.

NFA to DFA conversion:

→ An NFA is 5 tuple system

$$M = \{Q, \Sigma, \delta, q_0, f\}$$

where δ: transition function

$$Q \times \Sigma \to 2^Q$$

i.e., for each i/p symbol there may exist more than one transition.

The NFA mechanism is very helpful to describe certain non-deterministic system's behaviour but it is very complex to code.

for every NFA there exists DFA that simulates behaviour of NFA

i.e., for every NFA 'M' there exists equivalent DFA 'M₁' such that

$$L(M_1) = L(M)$$

Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA

we construct an equivalent DFA M_1

$$M_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$$

where $Q_1 = 2^Q$ i.e., power set of Q

$F_1 \in Q_1$ but contains element F

δ_1 defined as

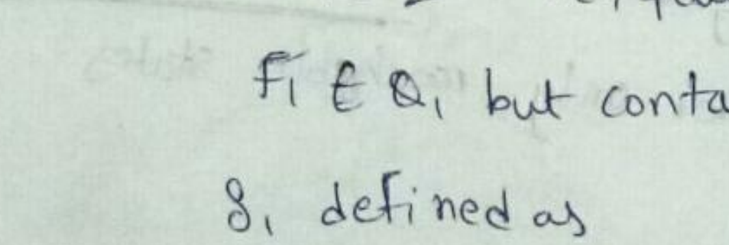
$$\delta_1\{\{q_0, q_1, \dots q_n\}, a\} = \delta(q_0, a) \cup \delta(q_1, a) \cup \dots \delta(q_n, a)$$

$$= P_0 \cup P_1 \cup \dots \cup P_n$$

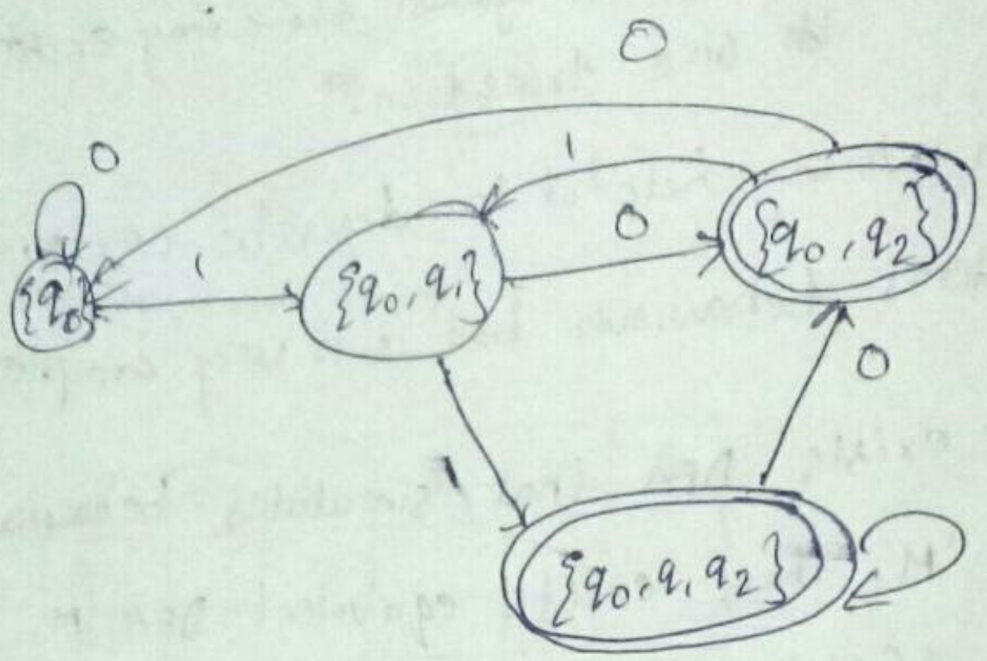
$$= [P_0, P_1 \dots P_n]$$

Eg: Convert the following NFA to DFA

NFA in which last but one symbol is 1.



	0	1
→ $\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1, q_2\}$	$\{q_0\}$	$\{q_0, q_1, q_2\}$
	$\{q_0, q_2\}$	



⇒ In the process of conversion ~~for~~ from NFA to DFA

$$Q_1 = 2^Q$$

which means Q_1 contains 2^n states.

But this may also contain unreachable states.

So we consider only reachable states

→ Convert below NFA to DFA

	0	1
→ P	P, q	P
q	r	r
r	-	s
*s	s	s

DFA →

	0	1
→ P	P, q	P
P, q	P, q, r	P, r
P, q, r	P, q, r	P, r, s
P, r	P, q	P, s
P, r, s	P, q, s	P, s
P, s	P, q, s	P, s
P, q, s	P, q, r, s	P, r, s
P, q, r, s	P, q, r, s	P, r, s

→ Convert NFA to DFA

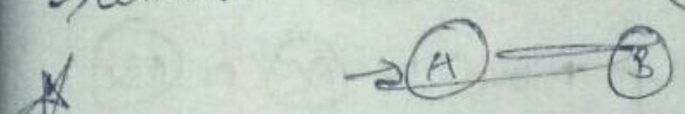
	0	1
→ P	q, r	q
q	r	s
r*	q	s
s*	s	r

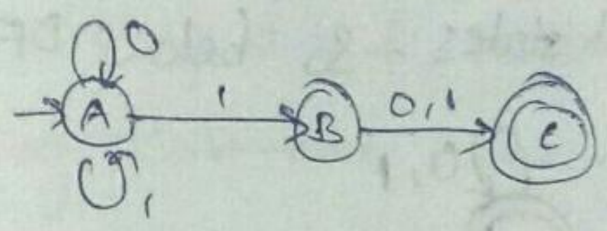
to DFA ⟶

	0	1
P	q, r	q
(q, s)	q, r	s
q	r	s
(s)	s	r
(r)	q	s

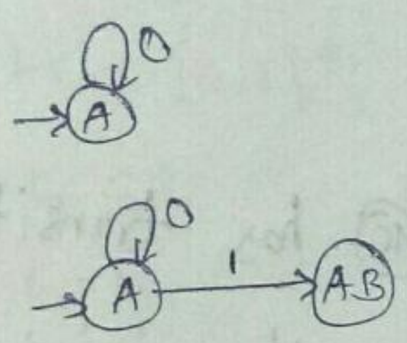
~~→ Convert NFA to DFA~~

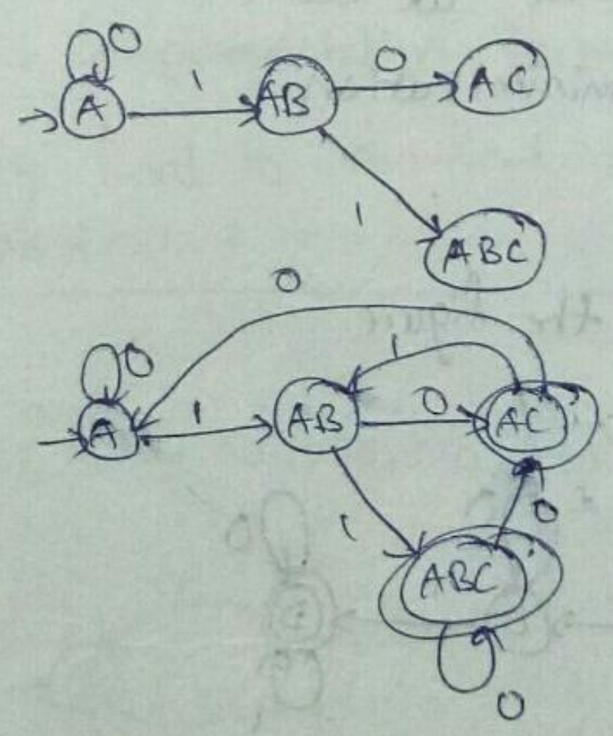
→ Convert NFA to DFA (Direct diagram to diagram conversion)





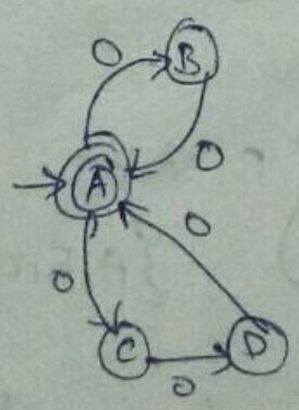
↓ direct diagram to diagram conversion

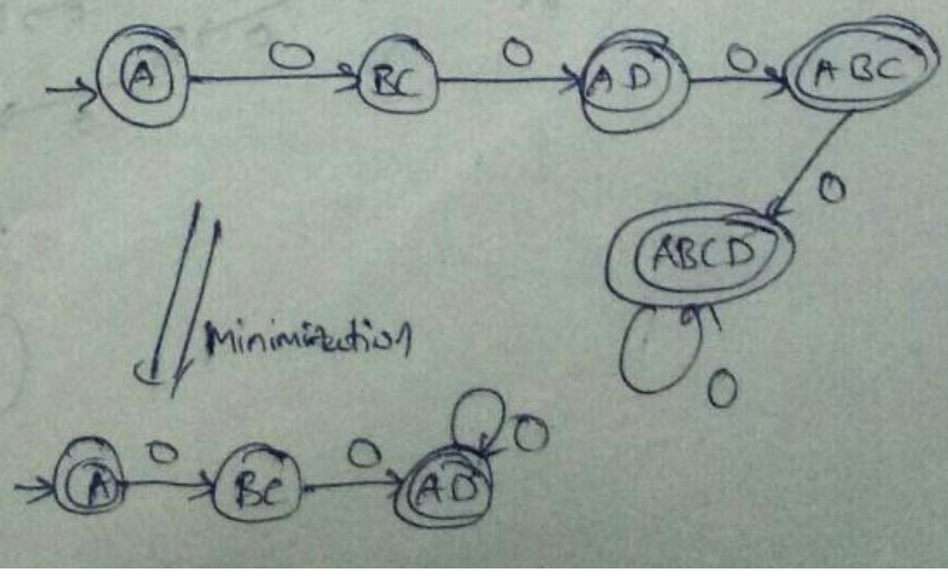




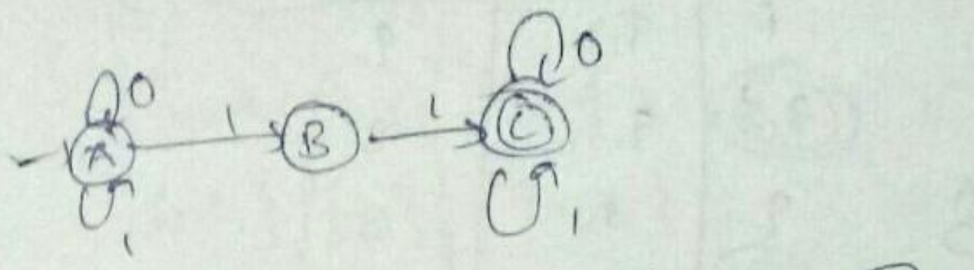
→ Convert NFA to DFA

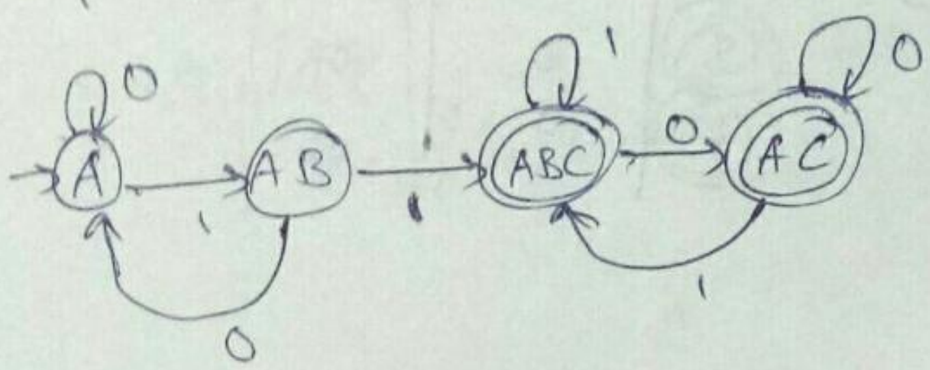
(1)



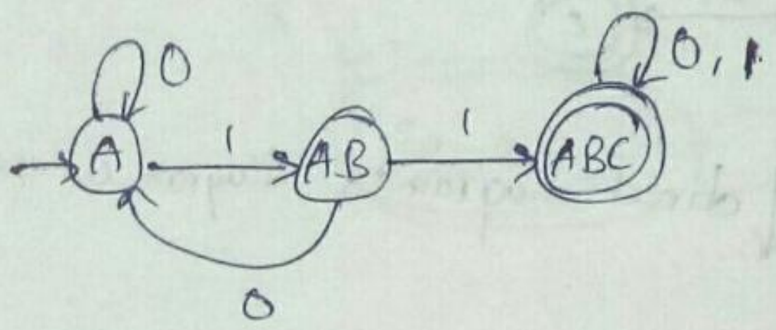


→ Convert NFA to DFA





In above fig once we reach state (ABC) No matter what i/p we read, we only move to (AC) or (ABC) which are final states. So below DFA is minimized

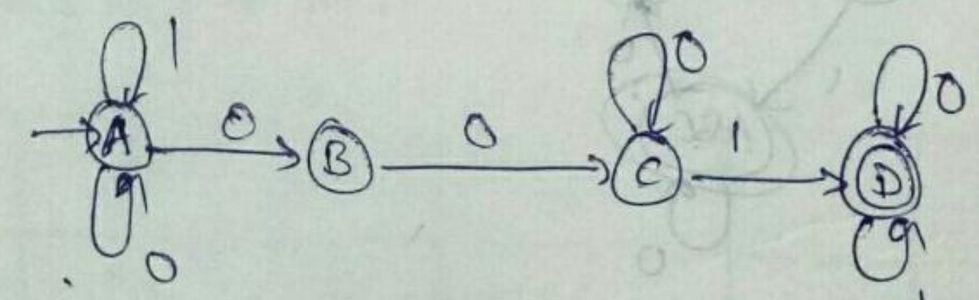


★ NOTE:

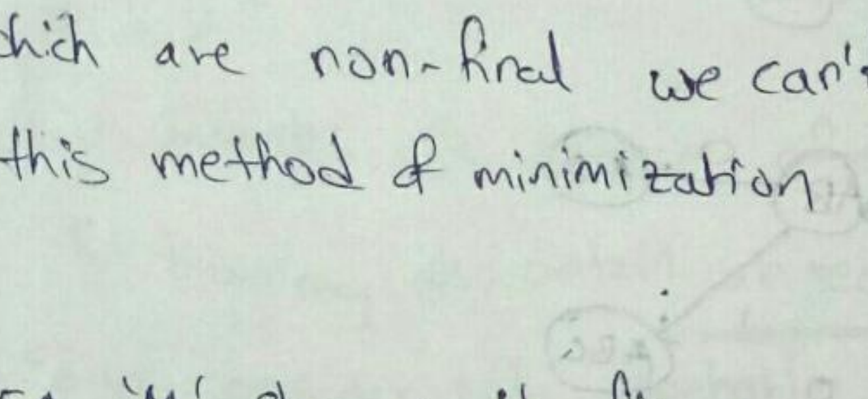
If either of states (ABC) or (AC) has transition to (A) (AB) or (AB) which are non-final we can't minimize the DFA in this method of minimization

Q:

Consider the NFA 'M' shown in the figure



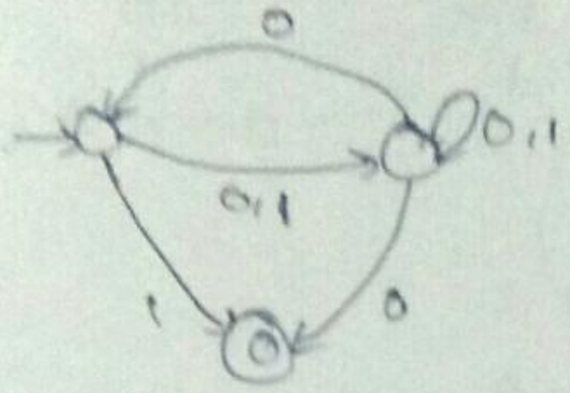
Find the no of reachable states for i/p string $w = 11100000$



(Draw tree for this kind of sums)

8. Consider below NFA 'M'

(1)



Let the lang accepted by M be L

Let L_1 be the lang accepted by NFA 'M₁', obtained by changing the ~~acc~~ accepting state of M to non-accepting state and by changing non-accepting state of M to accepting state. Which of the following ~~state men~~ statement is true?

a) $L_1 = \{0,1\}^* - L$

b) $L_1 = \{0,1\}^*$

c) $L_1 \subseteq L$

d) $L_1 = L$

This is complementation technique of DFA ~~of NFA~~ in which changing final to non-final and non-final to final does not work for NFA

given

$L = \{1, 10, 100, 000, 010, 100 \ldots\}$

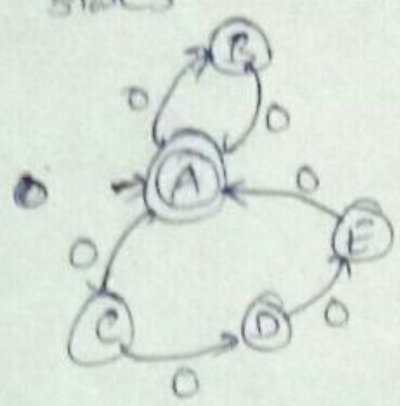
Now

If we observe the above NFA accepts any string of any length

$L = \{\varepsilon, 0, 1, 00, 01, 10, 11 \ldots\}$

$\therefore L = \{0,1\}^*$

Find min no of states when below DFA is converted to DFA.

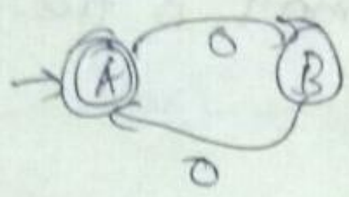


Method:1:

Observing above DFA it is clear that it accepts string containing $2x+4y$ number of zeroes, $x, y \geq 0$

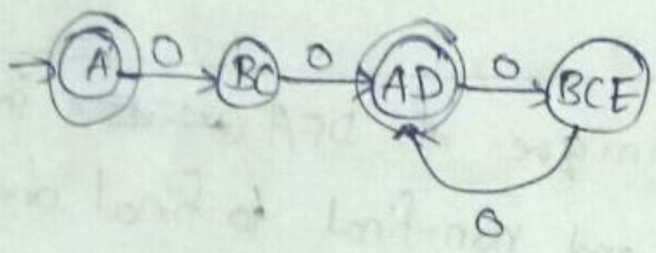
So we can say no of zeroes is divisible by 2

Hence DFA is



Method2:

direct conversion to DFA



Here direct minimization is not possible as it was in previous cases

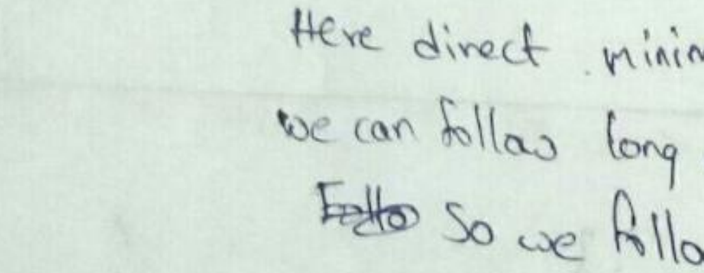
So we follow (Method 1)

ϵ-NFA

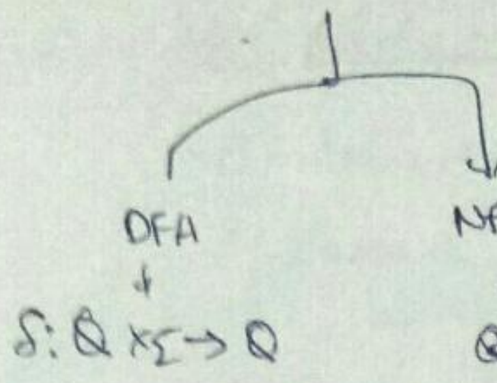
→ permits ϵ-transitions

→ ϵ-NFA 5tuples $M = (Q, \Sigma, \delta, q_0, F)$

$\delta: Q \times \Sigma \cup \epsilon \rightarrow 2^Q$



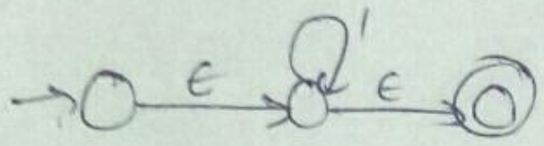
	0	1	2	ϵ
A	A	-	-	B
B	-	B	-	C
C	-	-	C	-



↳ ϵ can't be in Σ so we represent it seperately.

→ lang accepted by NFA shown above can be described as any no of zeroes followed by any no of 1's follow by any no of 2's

Eg:



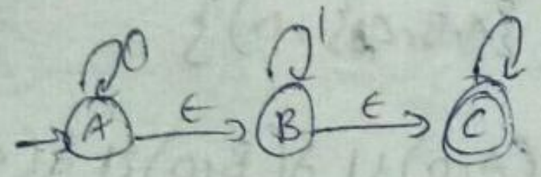
$L = \{\epsilon, 1, 11, 111, 1111, \ldots\}$

i.e, any no of 1's.

ϵ-Closure:

→ ϵ-closure of state q (ϵ-closure(q)) is the set of states P such that every state in the p is reachable from q through ϵ.

Consider below ϵNFA



ϵ-closure (A) = {A}

= {A} ∪ {B}

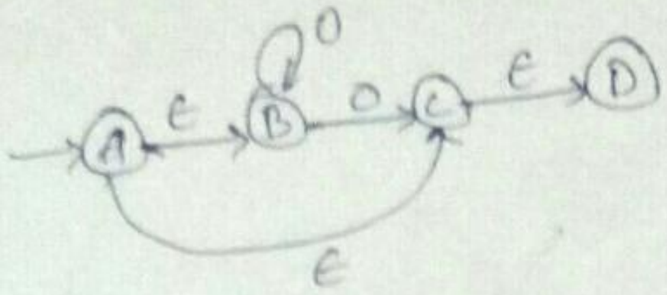
= {A} ∪ {B} ∪ {C}

ϵ-closure(A) = {A, B, C}

ϵ-closure(B) = {B, C}

ϵ-closure(C) = {C}

5:



ε-closure(A) = {A, B, C, D}

ε-closure(B) = {B}

ε-closure(C) = {C, D}

ε-closure(D) = {D}

→ ★ $\hat{\delta}(A, \epsilon) = \epsilon\text{-closure}(A)$ ★

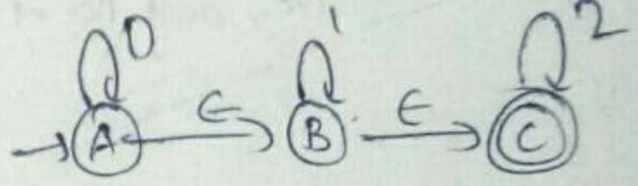
$\hat{\delta}(0,1)$ is known as extended transition function

$\hat{\delta}(A, \epsilon) = \{A, B, C, D\}$

Similarly $\hat{\delta}(B, \epsilon) = \{B\}$

$\hat{\delta}(C, \epsilon) = \{C, D\}$

$\hat{\delta}(D, \epsilon) = \{D\}$



→ ★ $\hat{\delta}(q, a) = \epsilon\text{-closure}\left(\delta(\hat{\delta}(q, \epsilon), a)\right)$ ★

5:

$\hat{\delta}(A, 0) = \epsilon\text{-closure}\left(\delta(\hat{\delta}(A, \epsilon), 0)\right)$

$= \epsilon\text{-closure}\left(\delta(\{A, B, C\}, 0)\right)$

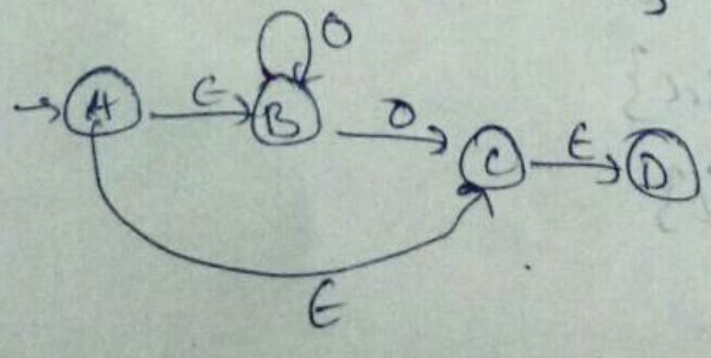
$= \epsilon\text{-Closeure}\left(\delta(A, 0) \cup \delta(B, 0) \cup \delta(C, 0) \cup \delta(D, 0)\right)$

$= \epsilon\text{-closure}(A \cup \phi \cup \phi)$

$= \epsilon\text{-closure}(A)$

$= \{A, B, C\}$

Consider



$\hat{\delta}(A,0) =$ ~~$\delta(\epsilon\hat{\delta}(A,0),0$~~

$= \epsilon\text{-closure}(\delta(\hat{\delta}(A,\epsilon),0))$

$= \epsilon\text{-closure}(\delta(\{A,B,C,D\},0)\}$

$= \epsilon\text{-closure}(\delta(A,0) \cup \delta(B,0) \cup \delta(C,0) \cup \delta(D,0))$

$= \epsilon\text{-closure}(\phi \cup \{B,C\} \cup \phi \cup \phi)$

$= \epsilon\text{-closure}(\{B,C\})$

$= \epsilon\text{-closure}\{B) \cup \epsilon\text{-closure}(C)$

$= \{B\} \cup \{C,D\}$

$= \{B,C,D\}$

→ Let 'w' be a string.

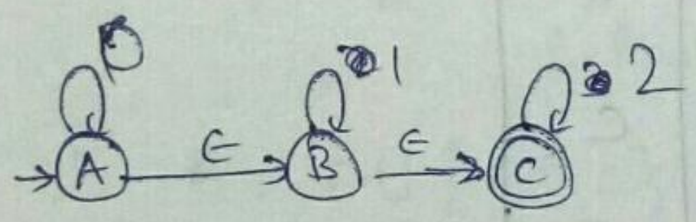
Let 'a' be last symbol of w and x be starting part

$w = xa$.

~~$\hat{\delta}(q_0,w) = \epsilon\text{closure}(\delta(\hat{\delta}(q_0,x),a))$~~

$\hat{\delta}(q_0,\omega) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0,x),a))$

⇒ Consider



$\hat{\delta}(B,12)$

$\hat{\delta}(B,1) = \epsilon\text{-closure}(\delta(\hat{\delta}(B,\epsilon),1))$

$= \epsilon\text{-clo}(\delta(\{B,C\},1))$

$= \epsilon\text{-clo}(B)$

$= \{B,C\}$

$\hat{\delta}(B,12) = \epsilon\text{-closure}(\delta(\hat{\delta}(B,1),2))$

$= \epsilon\text{-close}(\delta(\{B,C\},2)$

$= \epsilon\text{-clo}(C) = \{C\}$

ε-NFA to NFA :

In notes there is a method to convert ε-NFA to DFA directly

For every NFA with ε, we can construct an NFA without ε such that both will perform same job. In other words if L is accepted by NFA with ε, then L is also accepted by NFA without ε.

Proof :

Let $M = (Q, \Sigma, \delta, q_0, F)$ be ε-NFA

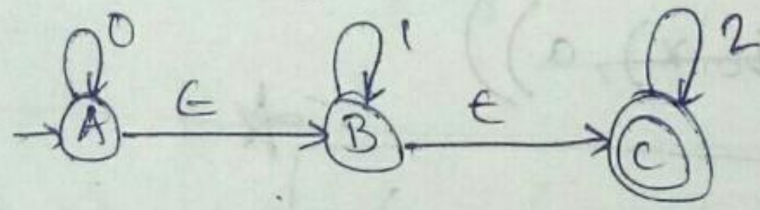
- we can construct an equivalent NFA M_1 to M as

$M_1 = (Q, \Sigma, \delta_1, q_0, F_1)$

δ : $\hat{\delta}(q, a)$ $\forall q \in Q$ & $a \in \Sigma$

$$F_1 = \begin{cases} F \cup \epsilon\text{-closure}(q_0) & \text{if } \epsilon\text{-closure}(q_0) \text{ contains element } F \\ F, & \text{otherwise} \end{cases}$$

→ Convert the following ε-NFA to NFA



	0	1	2	ε
→A	A	-	-	B
B	-	B	-	C
C	-	-	C	-

↓ ε-NFA to NFA

	0	1	2
→A*	{A, B, C}	{B, C}	{C}
B*	ϕ	{B, C}	{C}
C*	ϕ	ϕ	{C}

For each entry [q, a] find $\hat{\delta}(q, a)$

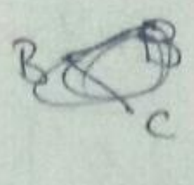
$\hat{\delta}(A,0) = \epsilon\text{-clo}(\delta(\hat{\delta}(A,\epsilon),0))$

$= \epsilon\text{-clo}(\delta(A,0) \cup \delta(B,0) \cup \delta(C,0))$

$= \epsilon\text{-clo}(A)$

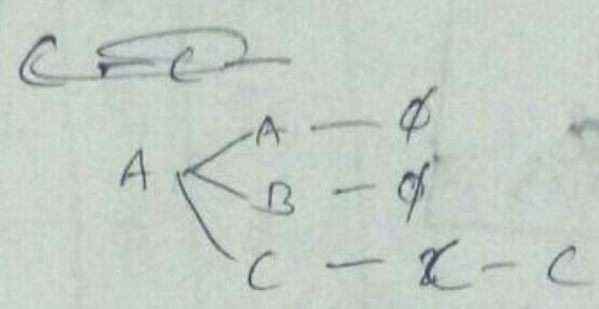
$= \{A, B, C\}$

$\hat{\delta}(A,1) =$



	e-clos	δ	e-closure
A	A	ϕ	ϕ
	B	B	B, C
	C	ϕ	ϕ

$\hat{\delta}(A,2) =$



$\hat{\delta}(B,0) = B \begin{cases} B - \phi - \phi \\ C - \phi - \phi \end{cases}$

$\hat{\delta}(B,1) = B \begin{cases} B - B - B - B, C \\ C - \phi - \phi \end{cases}$

Computing final states

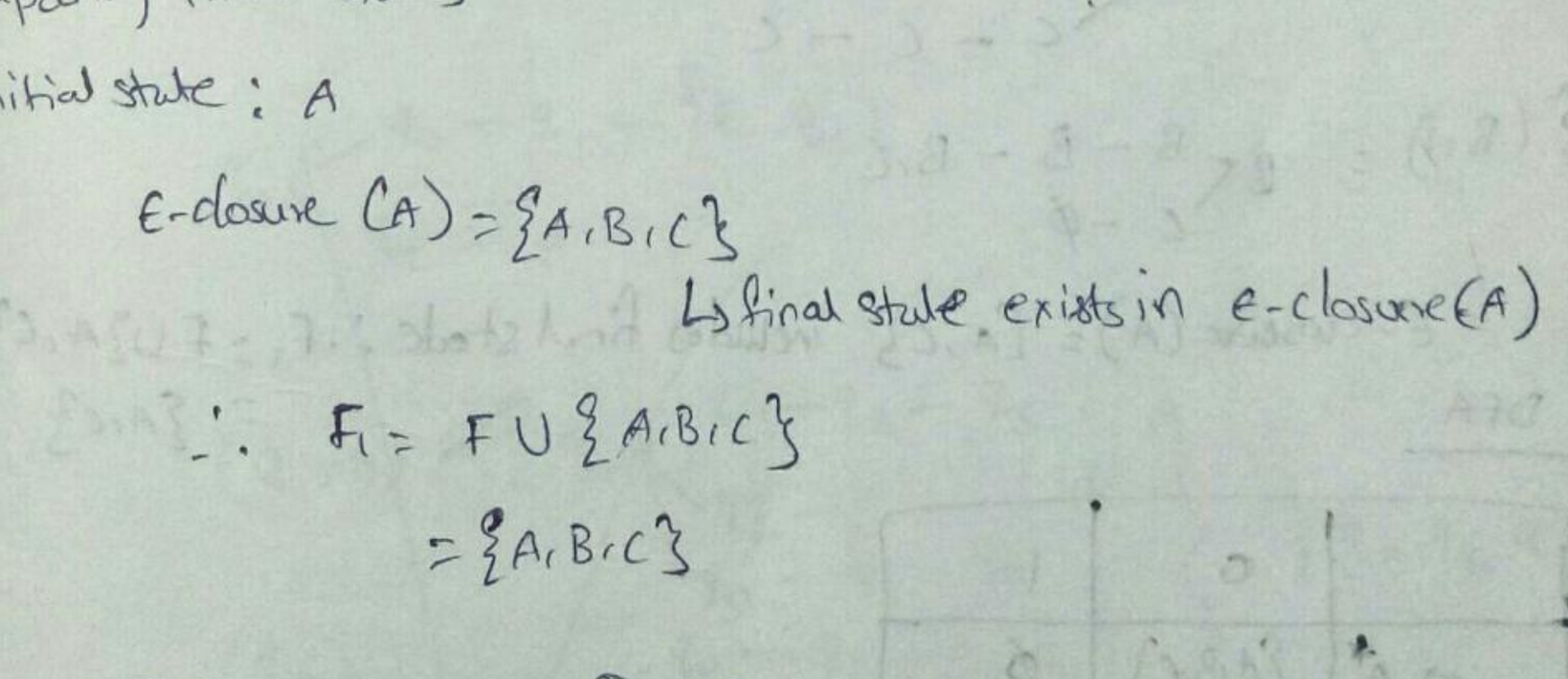
Initial state : A

$\epsilon\text{-closure}(A) = \{A, B, C\}$

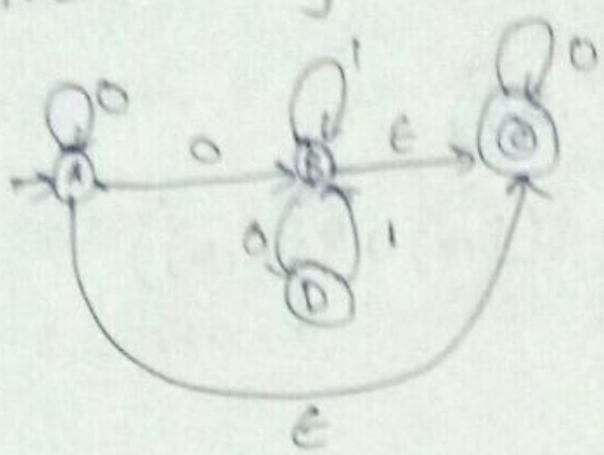
↳ final state exists in ϵ-closure(A)

$\therefore F_1 = F \cup \{A, B, C\}$

$= \{A, B, C\}$



→ Consider the following ε-NFA



	0	1
→A*	{A,B,C}	ϕ
B	{C,D}	{B,C}
C*	{C}	ϕ
D	ϕ	ϕ {B,C}

$\hat{\delta}(A,0) = A \begin{cases} A - A,B - A,B,C \\ C - C - C \end{cases}$

$\hat{\delta}(A,1) = A \begin{cases} A - \phi \\ C - \phi \end{cases}$

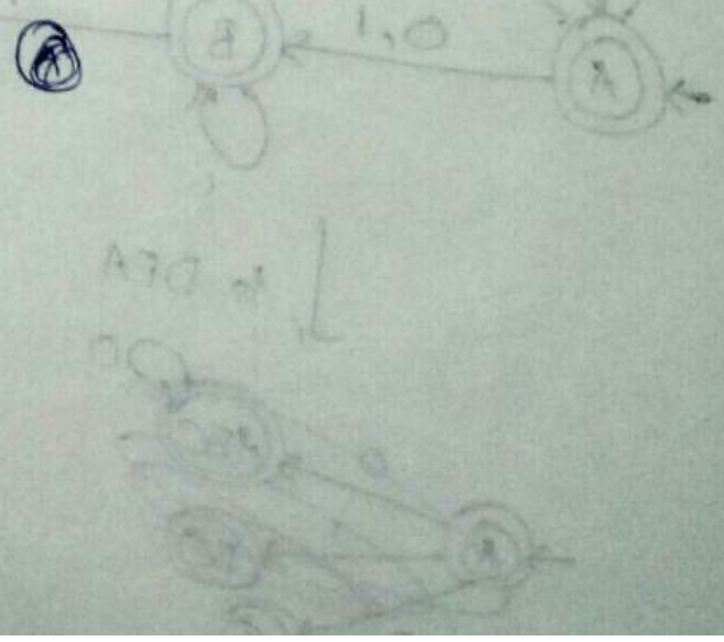
$\hat{\delta}(B,0) = B \begin{cases} B - D - D \\ C - C - C \end{cases}$

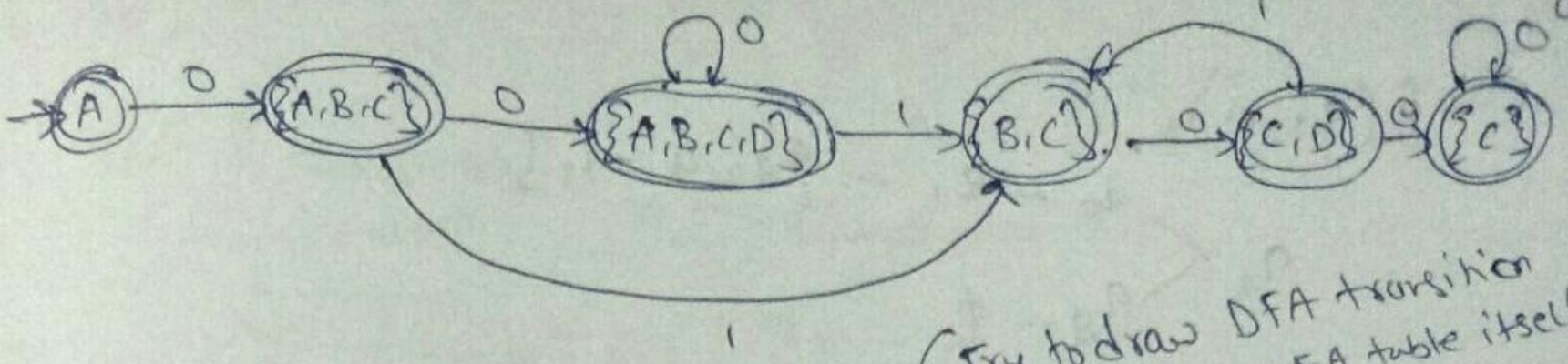
$\hat{\delta}(B,1) = B \begin{cases} B - B - B,C \\ C - \phi \end{cases}$

ε-closure(A) = {A,C} includes final state ∴ $F_1 = F \cup \{A,C\}$
$= \{A,C\}$

To DFA

	0	1
→A*	{A,B,C}	ϕ
{A,B,C}*	{A,B,C,D}	{B,C}
{A,B,C,D}*	{A,B,C,D}	{B,C}
{B,C}*	{C,D}	{B,C}
{C,D}* {C}*	{C} {C}	{B,C} ϕ





(Try to draw DFA transition diagram from NFA table itself)

Q: Let 'δ' denote the transition function, $\hat{\delta}$ denote the extended transition function of ϵ-NFA below

	ϵ	a	b
q_0	q_2	q_1	q_0
q_1	q_2	q_2	q_3
q_2	q_0	ϕ	ϕ
q_3	ϕ	ϕ	q_2

In this type of questions Calculate closures first so that we make no mistakes

then $\hat{\delta}(q_2, aba) = ?$

a) ϕ b) $\{q_0, q_1, q_2\}$ c) $\{q_0, q_1, q_3\}$ d) $\{q_0, q_2, q_3\}$

$q_2 \begin{cases} q_0 - q_1 - \{q_1, q_2\} \\ q_2 - \phi \end{cases}$

$(q_1, b) \quad q_1 \begin{cases} q_1 - q_3 - q_3 \\ q_2 - \phi \\ q_0 - q_0 - q_0 \\ q_0 - q_0 - \{q_0, q_2\} \end{cases} \Big\} \{q_0, q_2, q_3\}$

$(q_2, b) \quad q_2 \begin{cases} \\ q_2 - \phi \end{cases}$

$\{q_0, a\} \quad q_0 \begin{cases} q_0 - q_1 - \{q_1, q_2\} \\ q_2 - \phi \end{cases}$

$\hat{\delta}\{q_2, q_3\} - \{q_1, q_2\}$

$\{q_3, a\} - q_3 - q_3 - \phi$

$\hat{\delta}(q_2, a)$

$q_2 \begin{cases} q_0 - q_1 - \{q_0, q_1, q_2\} \\ q_2 - \phi \end{cases}$

$\hat{\delta}(q_2, ab)$

$q_0 \begin{cases} q_0 - q_0 - \{q_0, q_2\} \\ q_2 - \phi \end{cases}$

$q_1 \begin{cases} q_0 - q_0 - \{q_0, q_2\} \\ q_1 - q_3 - \phi \\ q_2 - \phi - \phi \end{cases}$

$q_2 \begin{cases} q_0 - q_0 - \{q_0, q_2\} \\ q_2 - \phi \end{cases}$

$\Rightarrow \{q_0, q_2\}$

$\hat{\delta}(q_2, aba)$

$q_0 \begin{cases} q_0 - q_1 - \{q_0, q_1, q_2\} \\ q_2 - \phi \end{cases}$

$q_2 \begin{cases} q_0 - q_1 - \{q_0, q_1, q_2\} \\ q_2 - \phi \end{cases}$

$\Rightarrow \{q_0, q_1, q_2\}$

$\therefore \hat{\delta}(q_2, aba) = \{q_0, q_1, q_2\}$

(ii) find $\hat{\delta}(q_1, abb)$

$\hat{\delta}(q_1, a)$

$q_1 \begin{cases} q_0 - q_1 - \{q_0, q_1, q_2\} \\ q_1 - q_2 - \{q_1, q_2\} \\ q_2 - \phi \end{cases}$

$\Rightarrow \{q_0, q_1, q_2\}$

$\hat{\delta}(q_1, ab)$

~~q_0~~ ~~ϕ~~ ~~Bound ab~~

from above

~~it is $\{q_0, q_2\}$~~

$q_0 - q_0 - \{q_0, q_1, q_2\}$

$q_0 \begin{cases} q_1 - \phi \\ q_2 - \phi \end{cases}$

$q_1 \begin{cases} q_0 - q_0 - \{q_0, q_2\} \\ q_1 - q_3 - \phi \\ q_2 - \phi \end{cases}$

$q_2 \begin{cases} q_0 - q_0 - \{q_0, q_2\} \\ q_2 - \phi \end{cases}$

$\hat{\delta}(q_1, abb)$

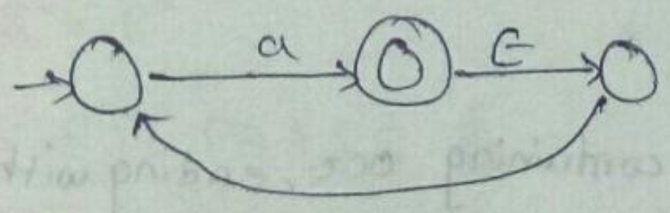
$q_0 \begin{cases} q_0 - q_0 \\ q_2 - \phi \end{cases}$

$q_0 \begin{cases} q_0 - q_0 - \{q_0, q_2\} \\ q_2 - \phi \end{cases}$

$q_2 \begin{cases} q_0 - q_0 - \{q_0, q_2\} \\ q_2 - \phi \end{cases}$

$\therefore \hat{\delta}(q_1, abb) = \{q_0, q_2\}$

Q: What complementation of lang accepted by below NFA

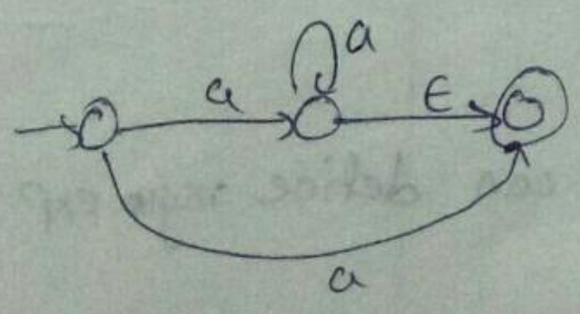


a) ϕ b) $\{\epsilon\}$ c) a^* d) $\{a, \epsilon\}$

Observing above NFA we can say it accepts strings containing one or more no of a's. $L = \{a, aa, aaa \ldots\}$

$\therefore$ complementation = $\{\epsilon\}$

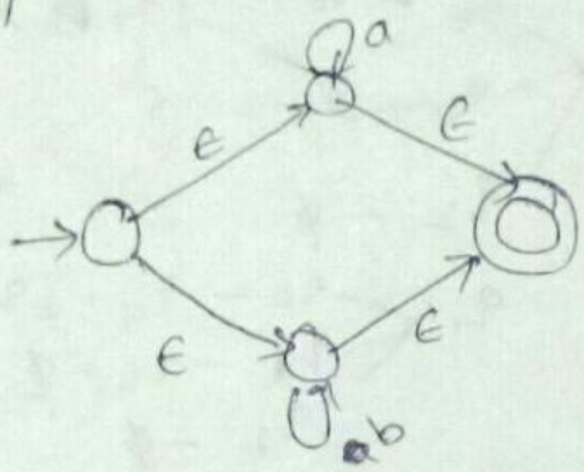
Q: Find complementation of below ϵ-NFA



$L = \{a, aa, aaa \ldots\}$

$\bar{L} = \{\epsilon\}$

Q: find lang accepted by ϵ-NFA below



a) $\{a,b\}^*$ b) a^* (or) b^* c) $\{a,b\}^+$ d) None

↓

$a^* + b^*$

05/01/20

Regular Sets & Regular Expressions:

→ The languages which are accepted by FA are called regular sets or regular languages.

Eg: Set of all strings over $\{0,1\}$ containing 000, ending with 00, even no of 0's etc.

Regular Expressions:

→ It is possible to express all regular languages in mathematical expressions using the operations

(i) union (+ or |)

(ii) Concatenation (.)

(iii) Kleen closure (*)

→ Let Σ be the input alphabet. we can define regex exp over Σ recursively as follows

(i) $\emptyset$ is regex denotes empty language

(ii) ϵ is regex denotes empty string.

(iii) 'a' is a regex such that $\forall a \in \Sigma$

(iv) Let r_1, r_2 be two regex, then

$r_1 + r_2$, $r_1 r_2$, r_1^* are also regex
(union) (concate) (kleen closure)

(v) The expressions which are obtained by using the rules 1 to 4 will also form regular expressions

Identities for regular expressions:

1. $\epsilon R = R\epsilon = R$
2. $\phi R = R\phi = \phi$
3. $\phi + R = R + \phi = R$
4. $R + R = R$
5. $R \cdot R = R^2$
6. $(R^*)^* = R^*$
7. $RR^* = R^*R = R^+$
8. $\epsilon + RR^* = RR^* + \epsilon = R^*$
9. $\epsilon + R^+ = R^*$
10. $P(Q+R) = PQ + PR$
11. $(Q+R)P = QP + RP$
12. $(P+Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$
13. $(\epsilon + R)^* = (R + \epsilon)^* = R^*$

Union is commutative but concatenation is not
i.e., $P+Q = Q+P$, $PQ \neq QP$

Both Union & concatenation are associative
i.e., $(P+Q)+R = P+(Q+R)$
$(PQ)R = P(QR)$

Regular set	Regular Exp
$L = \{\}$	$r = \phi$
$L = \{\epsilon\}$	$r = \epsilon$
$L = \{a\}$	$r = a$
$L = \{ab\} \cup \{ba\}$	$r = ab + ba$ (or) $ab \mid ba$
$L = \{aa, ba, bb\}$	$r = aa + ba + bb$

→ $L = \{aa, ab, ba, bb\}$

→ $L = \{a,b\}^* - \{a,b\} \cup \{aa, ba\}$

→ $L = \{a\}^*$

→ $L = \{a,b\}^*$

→ $L = \{a,b\}^* ab$

$r_1 = (a+b)(a+b) = (a+b)^2$

$(a+b) + (aa+ba)$

a^*

$(a+b)^*$

$(a+b)^* ab$

→ Give regexp over $\{0,1\}$ for following languages

(i) All strings ending with 00 → $(0+1)^* 00$

(ii) All string containing 000 → $(0+1)^* 000 (0+1)^*$

(iii) All strings begin with '1' and end with '0' → $1(0+1)^* 0$

(iv) Contains exactly 2 0's → $1^* 0 1^* 0 1^*$

(v) Contains atmost 2 0's → $1^* + 1^* 0 1^* + 1^* 0 1^* 0 1^*$

~~$\overset{2}{=} 1^*(0+1)\,1^*(0+1)\,1^*$~~

(or)

$1^*(\epsilon+0+1)\,1^*(\epsilon+0+1)\,1^*$

(vi) containing atleast 2 zeroes : $(0+1)^* 0 (0+1)^* 0 (0+1)^*$

(or)

$1^* 0 1^* 0 (0+1)^*$

(or)

$(0+1)^* 0 1^* 0 1^*$

(or)

$1^* 0 (0+1)^* 0 1^*$

(vii) strings of length 3 : $(0+1)^3$

(viii) strings of length 3 or more : $(0+1)^3 (0+1)^*$

(ix) length 3 or less : $\epsilon + (0+1) + (0+1)^2 + (0+1)^3 \overset{=}{=} (\epsilon+0+1)^3$

(x) containing even no of zeroes : $(1^*01^*01^*)^*1^*$

$\sim (1^*01^*0)^*1^*$

$\sim 1^*(01^*01^*)^*$

(xi) containing odd no of zeroes : ~~$(1^*01^*0)^*0$~~

$1^*0(1^*01^*0)^*1^*$

$1^*(01^*01^*)01^*$

(or)

~~$1^*0^*0 \quad (01^*01^*)^*0$~~

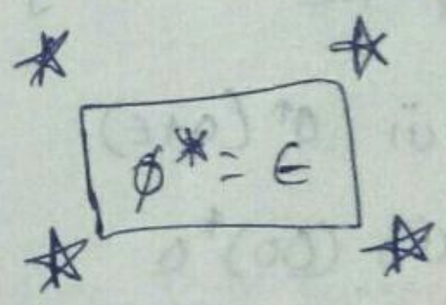
(xii) even 0's & even 1's

$$[00+11+(01+10)(00+11)^*(01+10)]^*$$

→ Consider the following regular expressions

$l_1 = a$

$l_2 = \{\}$



find $l_1 l_2^* + l_1$

$l_2^* = \epsilon$

$\Rightarrow l_1 l_2^* + l_1 = a\, l_1 = a$

→ $1^* + 00^*1^* = \underline{\quad ? \quad}$

a) 0^*1^* b) 1^*0^* c) $(0+1)^*$ d) None

$1^* + 00^*1^* = (\epsilon + 00^*)1^* = (\epsilon + 0^+)1^* = 0^*1^*$

→ Let $l_1 = a$ $l_2 = \phi$ find $l_1^* l_2^* + l_2^*$

$(l_1^* + \epsilon) l_2^* = l_1^* l_2^* = a^* \phi^* = a^* \epsilon = a^*$

→ Consider the following languages. give regular expressions.

1. $\{a^{2i} \mid i>0\}$ ⟶ $(aa)^+$

2. $\{a^{2i+1} \mid i\geq 0\}$ ⟶ $a(aa)^*$

3. $\{a^n b^m \mid n,m\geq 0\}$ ⟶ $a^* b^*$

4. $\{a^{n+1} b^{m+2} \mid n,m>0\}$ ⟶ ~~$aa^* bbb^*$~~ $aaa^* bbbb^*$

5. $\{a^{2n} b^{3m} c^{4k} \mid n,m,k\geq 0\}$ ⟶ $(aa)^*(bbb)^*(cccc)^*$

6. $\{a^{2n+1} b^{3m+2} c^{4k+4} \mid n,m,k\geq 0\}$ → $a(aa)^* bb(bbb)^* cccc(cccc)^*$

7. $\{a^n b^m \mid n=m\geq 0\}$

This is not a regular language and hence we can't write regex

→ Consider the following regular expressions

(i) $0^*(0+\epsilon)$

(ii) $(00)^*0$

(iii) $(00)^*$

(iv) 0^*

which of the above are equal

a) i & iii b) ii & iii c) i & iv d) None

$0^*(0+\epsilon) = 0^*0 + 0^*$

$= 0^+ + 0^* = 0^*$

★ $(00)^*(0+\epsilon) = (00)^*0 + (00)^*$
(odd) (even) $= 0^*$

2013

» Consider the following regexp

$$a^*b^*(ba)^*a^+$$

abbab

ba ab bb

find min length of string which do not present in above lang.

0	1	2	3
ϵ	a	aa	aaa
	b	ab	aab
		ba	aba
		bb	abb
			baa
			bab X

bb

∴ bab ⇒ 3

» ~~The regex~~ Q: In the above language how many strings of length 4 can't be produced?

abab
bbab
baab
babb

» The regexp denoting set of all strings not containing two consecutive 0's is given by

a) $(1+01)^*(\epsilon+0)$

b) $(0+10)^*(\epsilon+1)$

c) $(1+01)^*$

d) $(\epsilon+0)(101)^*(\epsilon+0)$

$L = \{\epsilon, 0, 101, 010, 110, 011, 111, 1111, \ldots\}$

a => $\{\epsilon, 0, 1, 101, 011, 110 \ldots\}$.

b => 100 belongs X

c => 0 does not belong X

d => 00 belong x

Q:
⇒ Which one of the following regexp over {0,1} denotes the set of all strings not containing 100 as substring

a) $0^*(1+0)^*$

b) 0^*1010^*

c) $0^*1^*01^*$

d) $0^*(10+1)^*$

req $l = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, \ldots\}$

a => $\{\epsilon, 0, 1, 0100\}$ ✗

b => $\{\underset{\times}{\epsilon}, \ldots\}$

c => $\{\underset{\times}{\epsilon}, \ldots\}$

d => $\{\epsilon, 0, 1, 00, 01, 10, 11, 000 \ldots\}$ ✓
100 ✗

∴ $0^*(10+1)^*$

Regular Exp to Finite Automata :

- For every regex r, we can construct an equivalent FA M such that $l(M) = l(r)$ i.e., lang accepted by M is equivalent to lang generated by r.

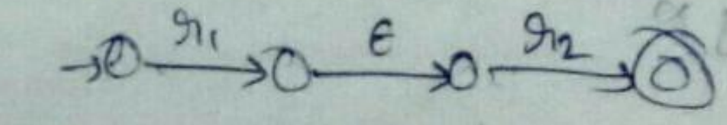
Synthesis Method :

Decomposition :

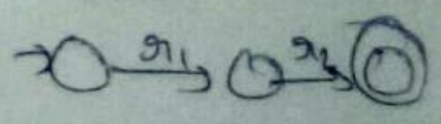
$l = \phi$ =>

$l = \epsilon$ =>

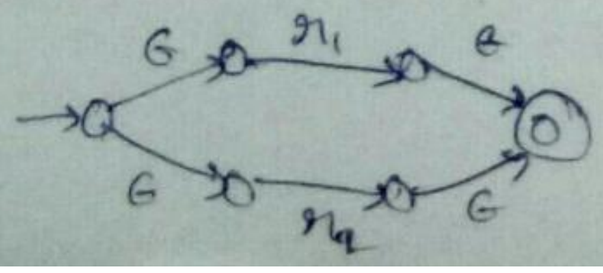
$l = r_1 r_2$ =>



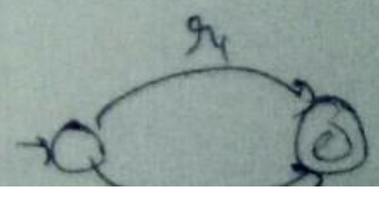
=>



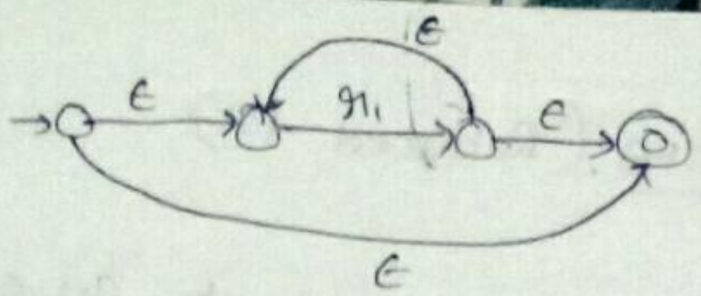
$l = r_1 + r_2$ =>



=>



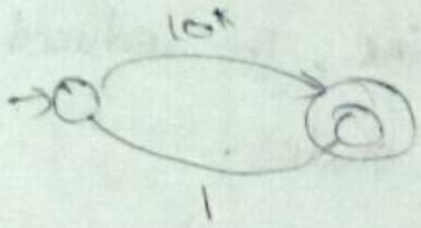
$r = r_1^*$ ⇒



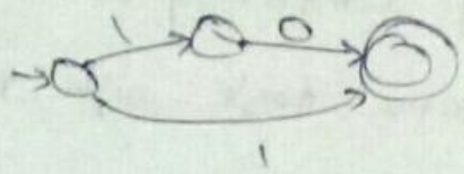
⇒

123

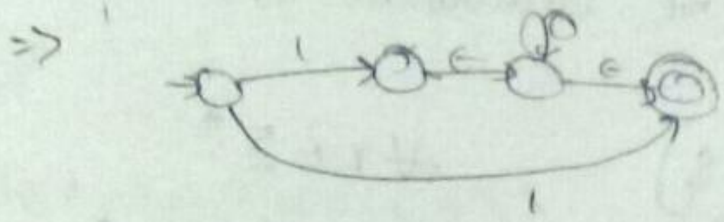
→ let $r = 10^* + 1$



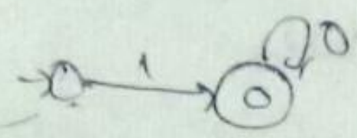
⇒



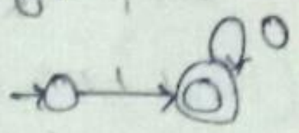
⇒



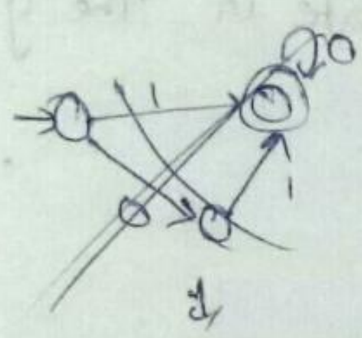
⇒



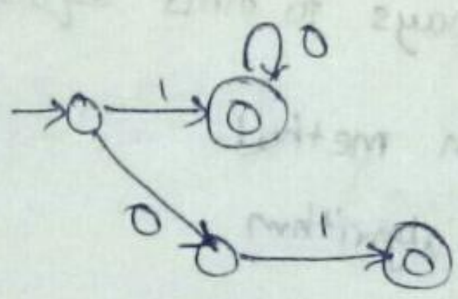
$r = 10^* + 1$
$= 1(0^* + \epsilon)$
$= 10^*$



→ $r = 10^* + 01$



(This is wrong)



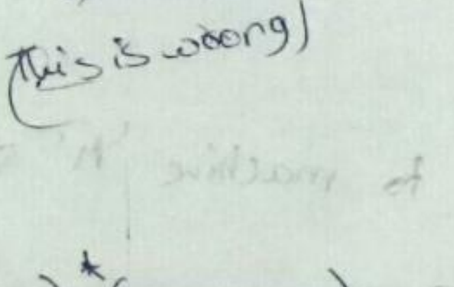
→ $r = (1+01+001)^*(\epsilon+0+00)$

The lang description for above lang is

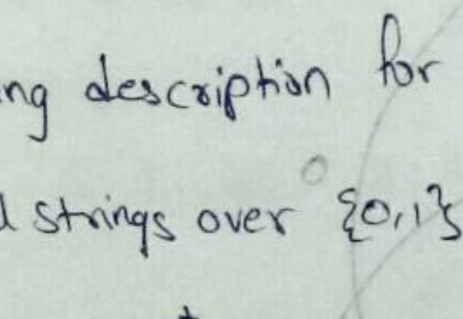
all strings over $\{0,1\}^*$ which do not more than 2 consecutive 0's.

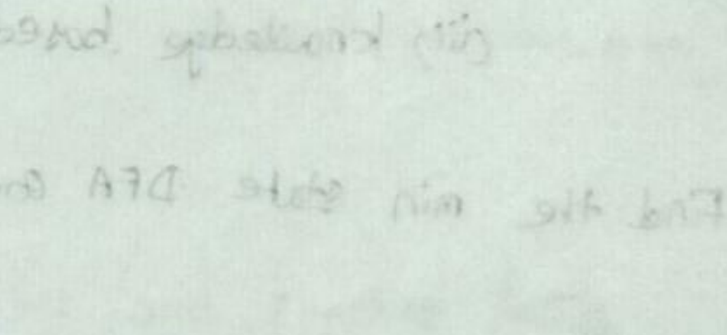
So we can draw complementation of FA of substring 000

for $(1+01+001)^*$

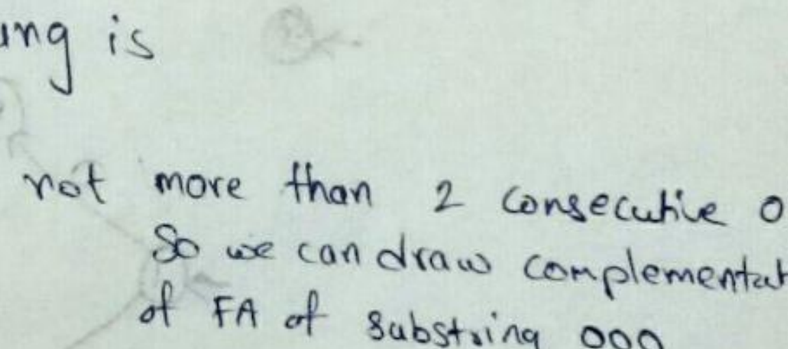


for $(1+01+001)^*(\epsilon+0+00)$





If complements



Minimum State DFA (Reduced FA)

→ For every FA 'M' we can construct an equivalent minimum state DFA that performs same task as M does i.e., the main objective of minimization of DFA is reducing complexity of actual machine. The reduced one will also perform the same task with min no of states.

→ Two states P & Q are said to be equivalent iff

$$\left(\delta(P,x), \delta(Q,x)\right) = (R,S) \quad , \quad \forall x \in \Sigma^*$$

where R & S are two final states

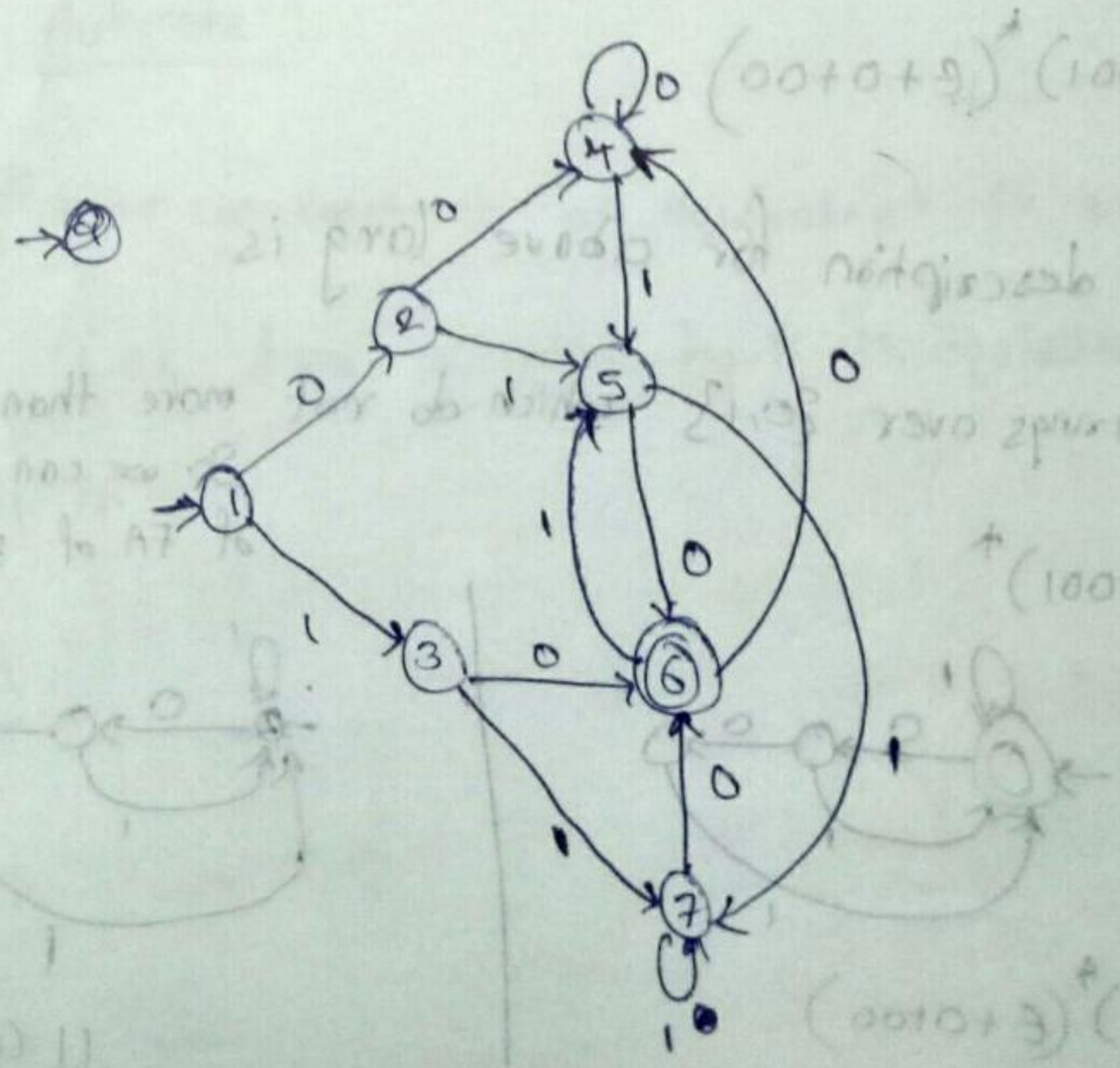
(or)

R & S are two non-final states

There are multiple ways to find equivalent state in the given DFA

(i) Tabulation method
(ii) partition algorithm
(iii) knowledge based

Eg: Find the min state DFA corresponding to machine 'M' shown below



	0	1
1	2	3
2	4	5
3	6	7
4	4	5
5	6	7
6*	4	5
7	6	7

Partition Alg:

$P_0 = \{1,2,3,4,5,6,7\}$ (All states)

Partition $P_1 = \{\{NF\}, \{F\}\}$ (Divide into final & non-final)

$P_1 = \{\{1,2,3,4,5,7\}\{6\}\}$

Partition $P_2 = \{\{1,2,4\}, \{3,5,7\}, \{6\}\}$

(Partition should continued until we obtain a partition similar to previous one)

Partion $P_3 = \{\{1,2,4\}, \{3,5,7\}, \{6\}\}$

(1) (3) (6)

while partition take two states from one set and check

Let these states be q_1, q_2

Now find

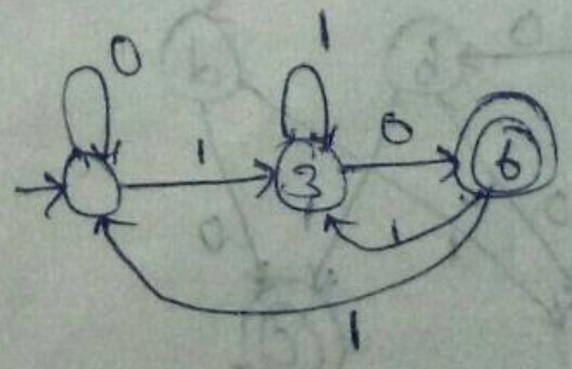
$\delta(q_1, a)$ $\delta(q_2, a)$

$\forall a \in \Sigma$

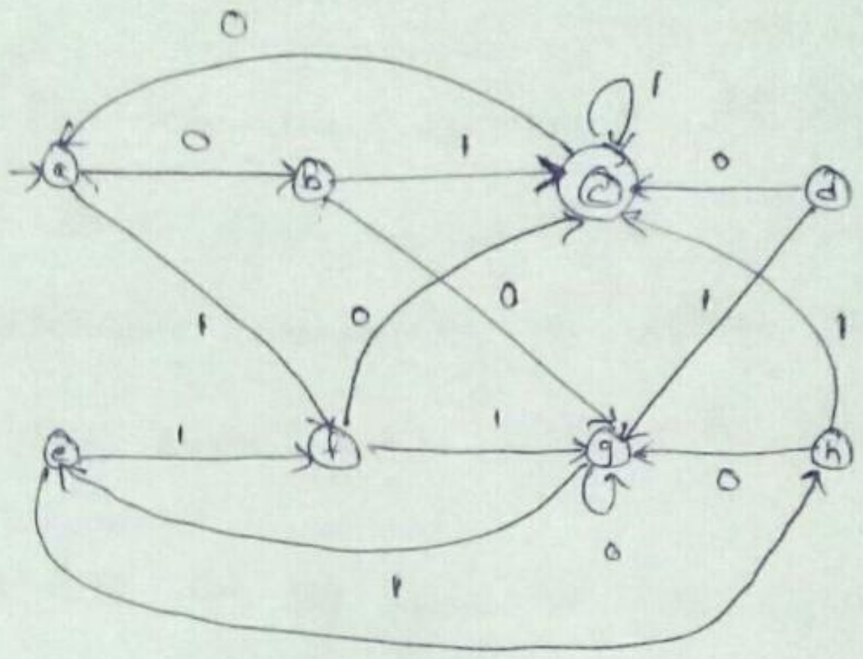
Now if both $\delta(q_1,a)$ & $\delta(q_2,a)$ produces results from same set then put q_1, q_2 in same set.

Otherwise divide them.

	0	1
→1	1	3
3	6	3
6*	1	3



→ Minimize the following DFA



	0	1
→a	b	f
b	g	c
c*	a	c
d	c	g
e	h	f
f	c	g
g	g	e
h	g	c

Tabulation method:

Check for ~~set~~ same entries of 0 & 1 in two rows, Now merge them and draw a new table. Now repeat the process until there are no two same entries

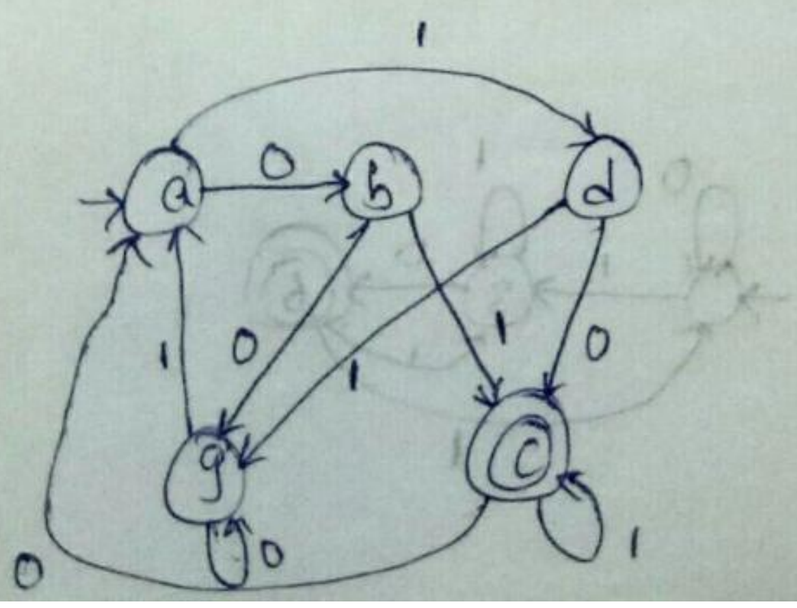
$P_1 = \{ \{a,b,d,e,f,g,h\}, \{c\} \}$

$P_2 = \{ \{a,e,g\}, \{b,h\}, \{d,f\}, \{c\} \}$

$P_2 = \{ \{a,e,\}, \{g\}, \{b,h\}, \{d,f\}, \{c\} \}$

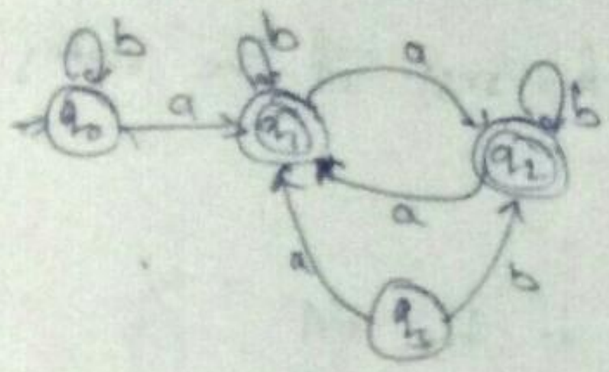
$P_3 = \{ \{a,e\}, \{g\}, \{b,h\}, \{d,f\}, \{c\} \}$

a g b d c



	0	1
→a	b	d
b	g	c
c*	a	c
d	c	g
g	g	a

→ Minimize



(Since q_3 is unreachable remove it)

	a	b
→q_0	q_1	q_0
q_1*	q_2	q_1
q_2*	q_1	q_2
q_3	q_1	q_2

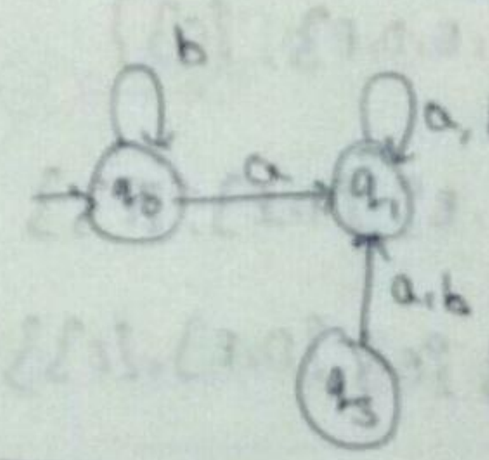
In this problem we can't find any identical entries and hence ~~we~~ using tabulation method is difficult ~~difficult~~ difficult. So we prefer partition method

$P_1 = \{\{q_0, q_3\}, \{q_1, q_2\}\}$

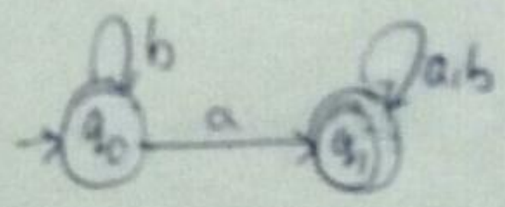
$P_2 = \{\{q_0\}, \{q_3\}, \{q_1, q_2\}\}$

$P_3 = \{\{q_0\}, \{q_3\}, \{q_1, q_2\}\}$

	a	b
q_0	q_1	q_0
q_1	q_1	q_1
q_3	q_1	q_1



Here q_3 is unreachable state, so remove it too



This step can be performed even before applying actual partition algorithm

☆

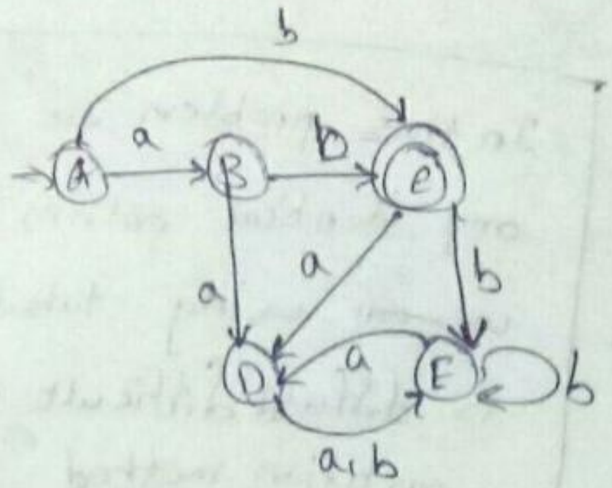
→ The minimization of FA procedures applicable for DFA only

i.e, if given machine is NFA, it should be transformed into DFA before application of any minimization procedures.

→ We have to ensure that the DFA machine is free from unreachable states.

→ If there is any unreachable state in DFA, it should be ignored and procedure should be applied for rest of the states

Eg: Find min state DFA corresponding to the FA



	a	b
→A	B	C
B	D	C
C*	D	E
D	E	E
E	D	E

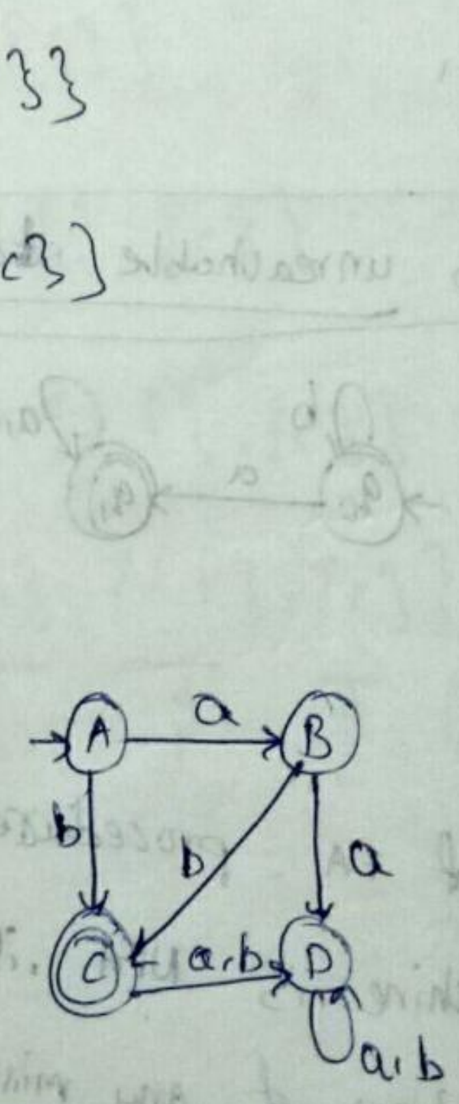
$P_1 = \{ \{A, B, D, E\}, \{C\} \}$

$P_2 = \{ \{A, B\}, \{D, E\}, \{C\} \}$

$P_3 = \{ \{A\} \{B\} \{D, E\}, \{C\} \}$

$P_4 = \{ \{A\} \{B\} \{D, E\} \{C\} \}$

	a	b
→A	B	C
B	D	C
C*	D	D
D	D	D

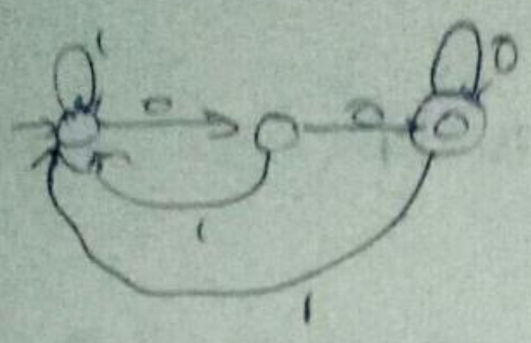


→ Finite language always contains a dead state.

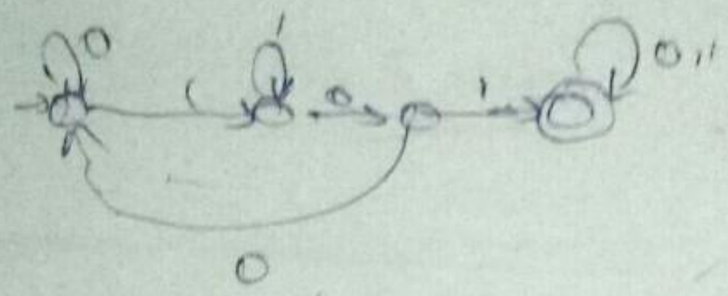
→ Infinite language may or may not have a dead state

Q: The min state DFA to recognize set of all strings over {0,1} such that every string end with 00 has 3 no of states.

Sol

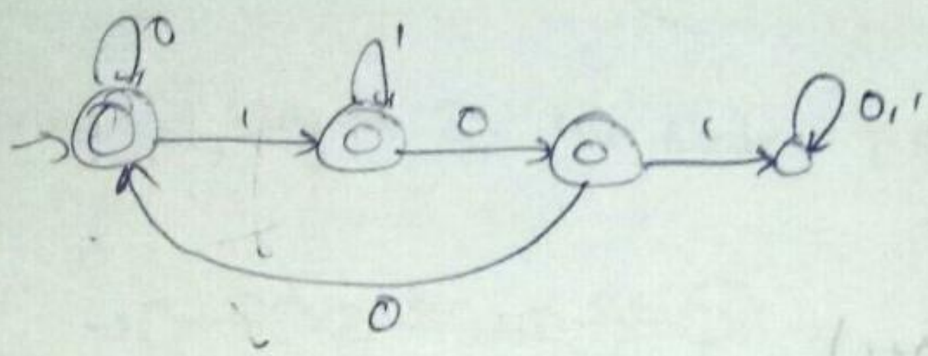


Q: Determine minstate DFA to recognize set of all strings containing 101 substring



∴ 4 states

for do not contain 101



∴ 4 states (including dead state too)

→ The min state DFA to recognize set of all strings over Σ, ending with some certain string of length 'n' has n+1 states

→ The no of states in DFA to recognize the language set of all strings over Σ contain a substring x such that |x|=n has n+1 states

→ The min state DFA to accept language L has n states, the DFA for L' also have same no of states

→ The min state DFA to recoganise set of all strings over {0,1} such that second symbol from right end is one.

Sol:

2nd symbol from right end is 1

$n = 2$

no of states $= 2^n = 2^2 = 4$

no of final state $= 2^{n-1} = 2$

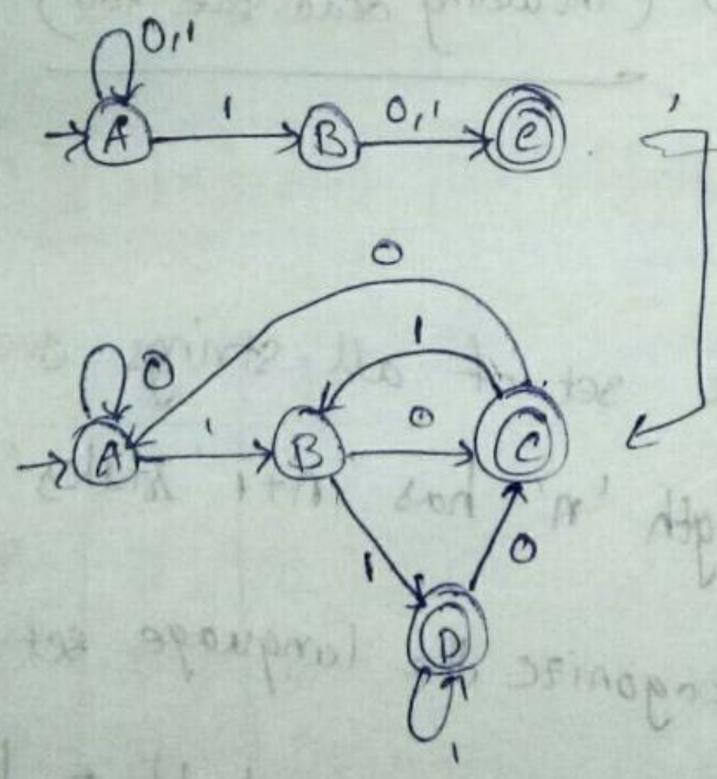
Transition table:

	0	1
→ q_0	q_0	q_1
q_1	q_2	q_3
q_2^*	q_0	q_1
q_3^*	q_2	q_3

∴ 4 states

Method 2:

Regex: $(0+1)^* 1 (0+1)$

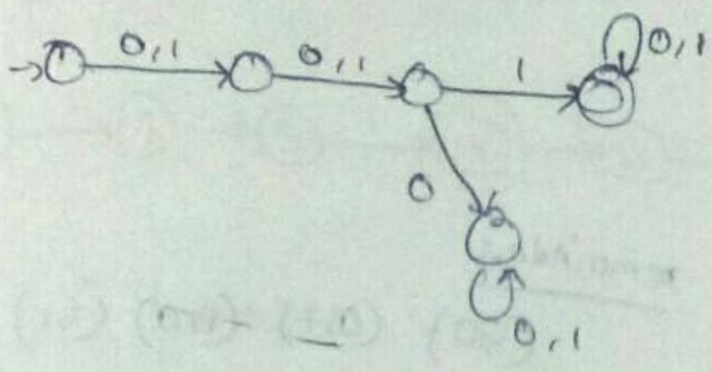


→ Min no of states in DFA containing nth symbol from right end is contains 2^n states defined over {0,1}

→ Set of all strings over $\{0,1\}$ has 3rd symbol from left end as 1 has ____ states.

Soln:

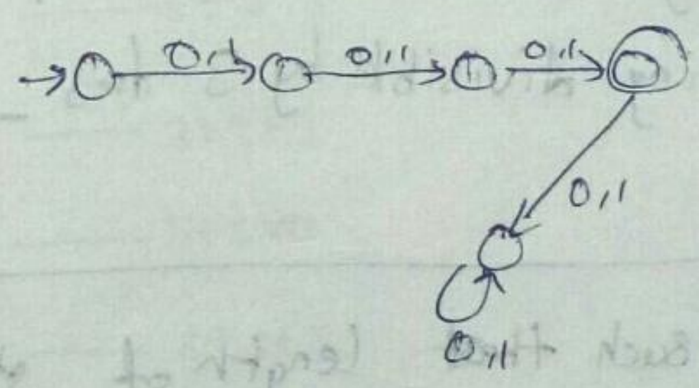
Regex : $(0+1)(0+1)1(0+1)^*$



∴ 5 states

→ The min state DFA to recogonize ove $\{0,1\}$ such that n^{th} symbol from left end is 1 is $n+2$

→ Set of all string of length 3. Min state DFA = 5 states



Regex : $(0+1)(0+1)(0+1)$

Remember that finite language always contains a dead state

The min state DFA to recogonize set of all strings of length n over alphabet $\{0,1\}$ has $n+2$ states

The min state DFA to recogonize set of all strings over $\{0,1\}$ (or any Σ) of length n or less has $n+2$ states

→ The min state DFA to recogonize set of all strings over {0,1} such that length of string is 2^n or more has $n+1$ states

→ The min state DFA to recogonize set of all strings over {0,1} such that no of 0's is divisible by 2 and no of 1's is divisible by 2 has 4 no of states

∴ Sol:

2×2 = 4 — remainders: (0,0) (0,1) (1,0) (1,1)

∴ 4 states

0's by 2 < 0, 1 }
0's by 2 < 0, 1 } 4 combinations

→ The min state DFA to recogonize set of all strings over {0,1} such that no of 0's is divisible by m & no of 1's is divisible by n has m×n no of states.

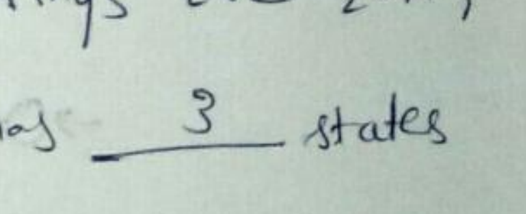
→ The min state DFA to recogonize set of all strings over {0,1} such that length of the string divisible by 3 has 3 states

→ set of all strings over Σ such that length of string is divisible by n. This lang's min DFA has n states

Problems:

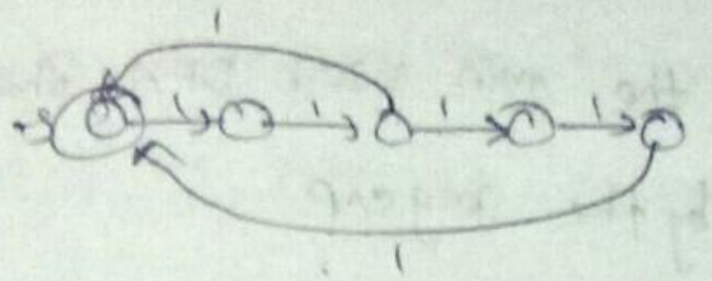
(2006) Consider the regular lang

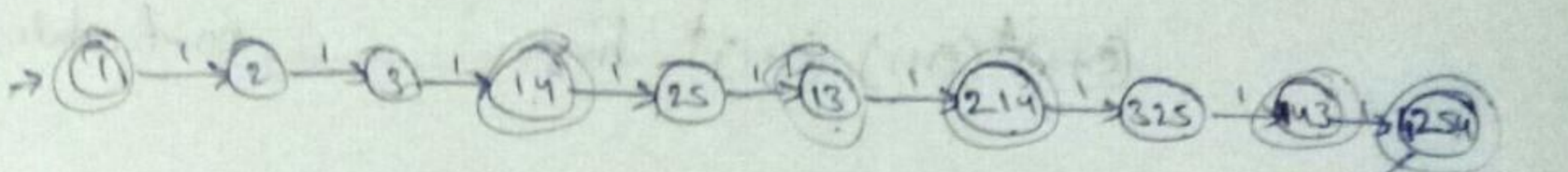
$$l = (111 + 1111)^*$$



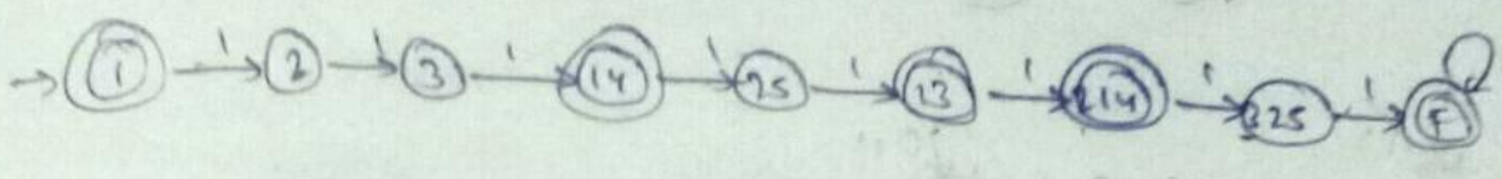
The min no of states in a DFA accepting this lang has __ states

a) 3 b) 5 c) 8 d) 9





Minimization



∴ 9 states

Method 2:

$1 \longrightarrow x$

$11 \longrightarrow x$

$111 \longrightarrow 3$

$1111 \longrightarrow x$

$11111 \longrightarrow 5$

$1^6 \longrightarrow 3+3$

$1^7 \longrightarrow x$

$1^8 \longrightarrow 3+5$

$1^9 \longrightarrow 3+3+3$

$1^{10} \longrightarrow 5+5$

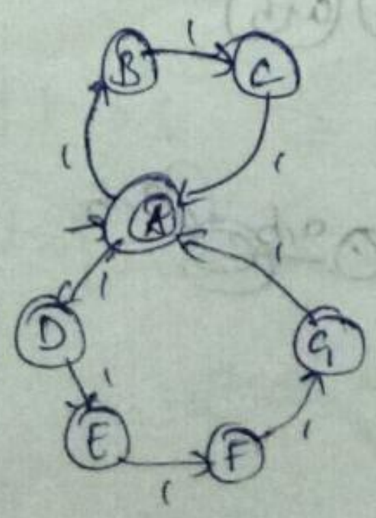
$1^{11} \longrightarrow 3+3+5$

i.e., More than 8 ones (or) 8 ones

DFA accepts any length

∴ we can draw it with 9 states

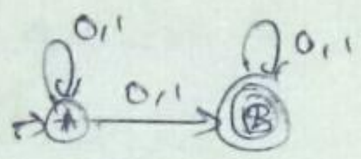
PFA can also be drawn as



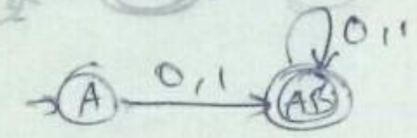
(2016) Consider the no of states in the min sized DFA that accepts the language defined by the regexp

$(0+1)^*(0+1)(0+1)^*$ has _____ no of states

NFA:



DFA:



(2017) 2M Consider the lang l given by the regexp $(a+b)^*b(a+b)^*$ over $\{a,b\}$

The smallest no of states needed in DFA accepting l

Sol:

This DFA which accepts strings whose 2nd symbol from right end is b

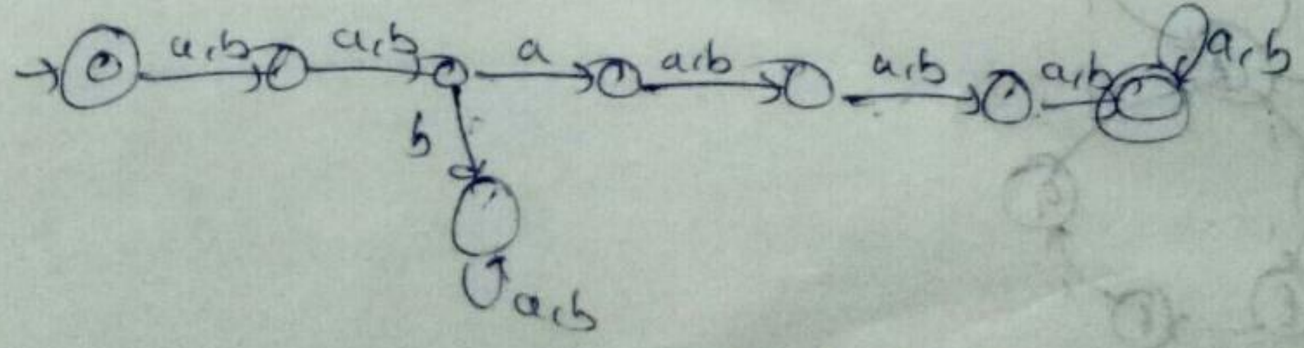
here $n=2$

$2^2 = 4$ states.

(2017) The min possible state of DFA that accept regular lang

$l = \{\omega_1 a \omega_2 \mid \omega_1, \omega_2 \in \{a,b\}^*$, length of ω_1 is 2 and length of ω_2 is greater than or equal to 3$\}$

$(a+b)(a+b)\,a\,(a+b)(a+b)(a+b)(a+b)^*$

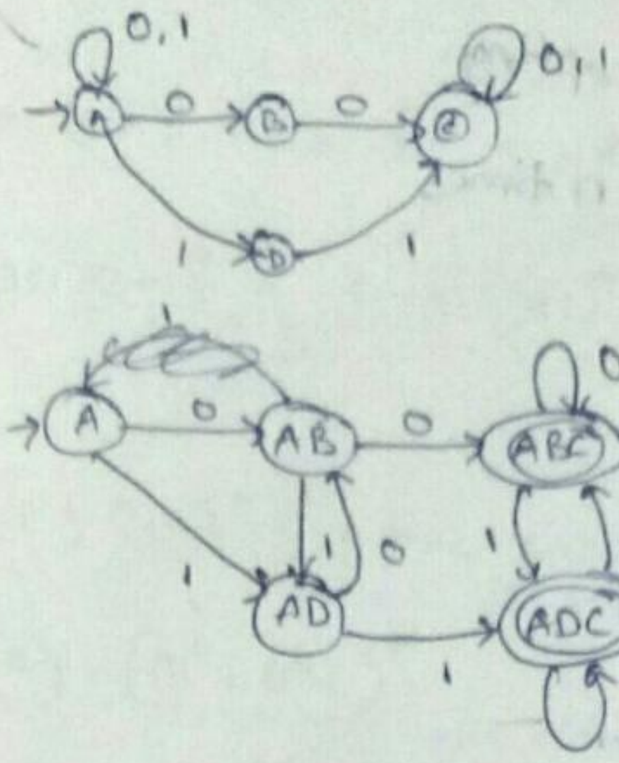


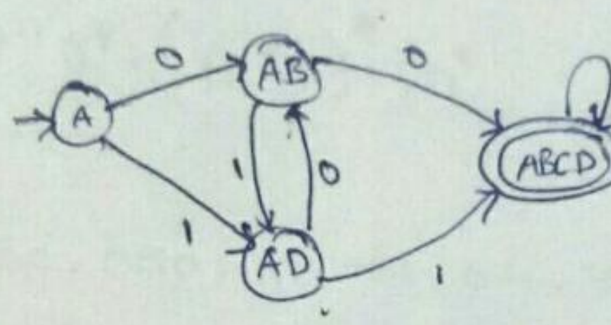
∴ 8 states

12/01/20

Q: The min state DFA to recogonize set of all strings over {0,1} containing the substring 00 (or) 11 has _____ no of states

$(0+1)^*(00+11)(0+1)^*$





(2011) (N) Definition of the language L with alphabet $\{a\}$ is given as follows

$L = \{a^{nk} \mid k>0\}$ and n is positive integer constant.

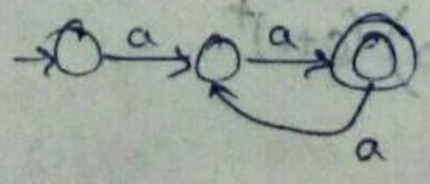
What is min no of states needed in DFA to recogonise L?

a) $k+1$ b) $n+1$ c) 2^{n+1} d) 2^{k+1}

Put $n=2$

$\{a^{2k} \mid k>0\}$

$L = \{aa, aaaa \ldots\}$



i.e., $n+1$ states

→ Let l be lang representing $\Sigma^* 0011 \Sigma^*$ where $\Sigma = \{0,1\}$

Find $\uparrow^{min}$ no of states in DFA representing $\bar{l}$.

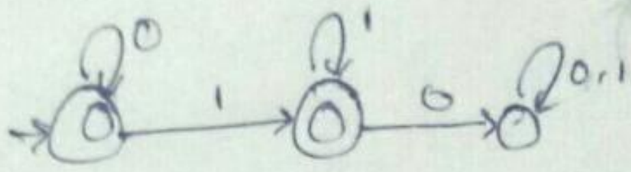
5 states

→ No of states in min DFA accepting

$(0+1)(0+1) \cdots$ n times

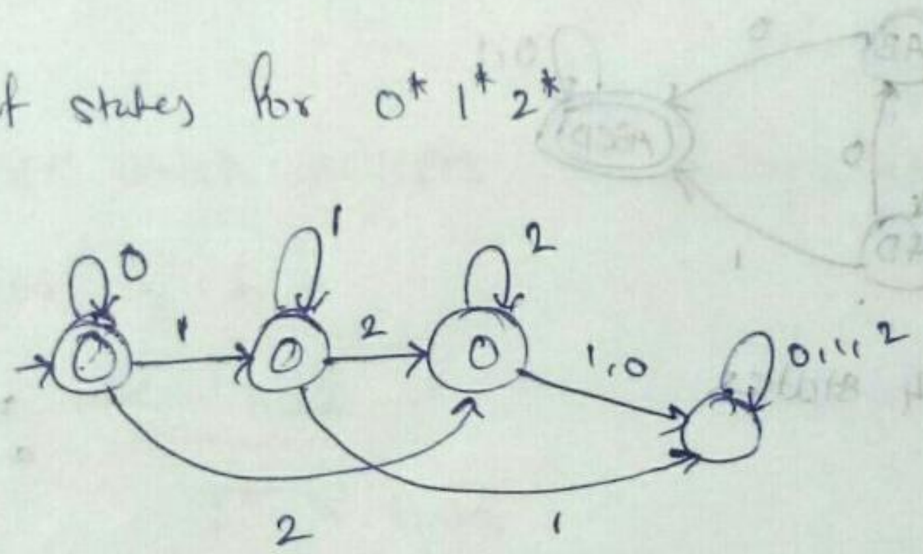
Ans: n+2 states.

→ Min no of states for $0^* 1^*$



3 stats

→ Min no of states for $0^* 1^* 2^*$



∴ 4 states

(3)

→ Min no of states in $a^* b^* c^* \cdots z^*$

(2)

26+1 = 27 states

$a_1^* a_2^* a_3^* \cdots a_n^*$

n+1 states

→ which one of the following regexp represent the lang set of all binary strings having 2 consecutive 0's and 2 consecutive 1's

a) $(0+1)^*(00+11)(0+1)^* + (0+1)^*1100(0+1)^*$

✓b) $(0+1)^*(00(0+1)^*11 + 11(0+1)^*00)(0+1)^*$

c) $(0+1)^* 00 (0+1)^* + (0+1)^* 11 (0+1)^*$

d) $00(0+1)^* 11 + 11(0+1)^* 00$

req lang

$(0+1)^* 00 (0+1)^* 11 (0+1)^* + (0+1)^* 11 (0+1)^* 00 (0+1)^*$

$(0+1)^* [00(0+1)^* 11 (0+1)^* + 11 (0+1)^* 00 (0+1)^*]$

$(0+1)^* [00(0+1)^* 11 + 11(0+1)^* 00] (0+1)^*$

→ Let $L = \{ w \in \{0,1\}^* / w$ has even no of 1's $\}$

a) $(0^*10^*1)^*$

b) $0^*(10^*10^*)^*$

c) $0^*(10^*1)^* 0^*$

d) $0^*1(10^*1)^* 10^*$

$L = \{\epsilon, 0, 00, 000, 11, 101, 011, 110, \ldots\}$

a $\Rightarrow \{\epsilon, 11 \ldots\}$ 00 ∉

b $\Rightarrow \{\epsilon, 0, 00, 000, 11, 101, 011, 110 \ldots\}$ ✓ 11011 ∈

c $\Rightarrow \{\epsilon, 0, 00, 000, 11, 101, 011, 110, \ldots\}$ 11011 ∉

d $\Rightarrow \{\frac{\epsilon}{\times} \ldots\}$

(2) → Let $r = 1(1+0)^*$

(2) $s = 11^*0$

$t = 1^*0$

be 3 regular expressions. which of the following is true?

(i) $L(s) \subseteq L(r)$ and $L(s) \not\subseteq L(t)$

(ii) $L(r) \subseteq L(s)$ and $L(s) \subseteq L(t)$

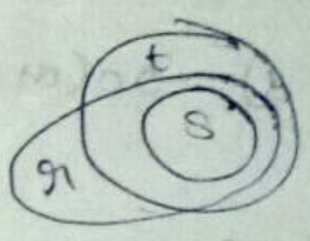
(iii) $L(t) \subseteq L(s)$ and $L(s) \subseteq L(r)$

(iv) $L(t) \subseteq L(s)$ and $L(s) \subseteq L(r)$

$r_1 = 1(1+0)^*$ $= \{1, 10, 110 \ldots\}$

$s = 11^*0$ $= \{10, 110, 1110 \ldots\}$

$t = 1^*0$ $= \{0, 10, 110, 1110 \ldots\}$



$\therefore L(s) \subseteq L(t)$, $L(s) \subseteq L(r_1)$ but $L(t) \notin L(r_1)$

→ Consider the following reg exp

$$r_1 = (a+b^*)^* \, b \, (a+b)^*$$

How many strings are there in $l(r_1)$ of length 3 or less.

Sol:

The lang is containing substring b

i.e., containing atleast one 'b'.

length 0	length 1	length 2	length 3
ϵ	1	4 - 1 (aa)	8 - 1 (aaa)
0	1	3	7

$= 11$

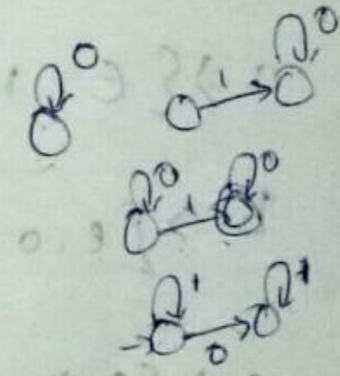
→ The reg exp $0^*(10^*)^*$ denote the same set as

a) $(1^*0)^*1^*$

b) $0 + (0+10)^*$

c) $(0+1)^* 10 (0+1)^*$

d) no of the above



$l = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, \ldots\}$

$a = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, \ldots\}$

$b = \{\epsilon, 0, 1, \ldots\}$

$c = \{1, 111\}$

Ans: a

Let S & T be languages over $\{a,b\}$ represented by the reg exp $(a+b^*)^*$ and $(a+b)^*$ respectively which of the following is true

a) $S \subset T$

b) $T \subset S$

✓ c) $S = T$

d) $S \cap T = \phi$

$$(a+b)^* = (a^*+b^*)^* = (a^*b^*)^* = (a^*+b)^* = (a+b^*)^*$$

Transformation of Finite Automata to Regular Expression

Methods:

(i) Arden's theorem

(ii) kleen's theorem

(iii) State Elimination method

Arden's Theorem:

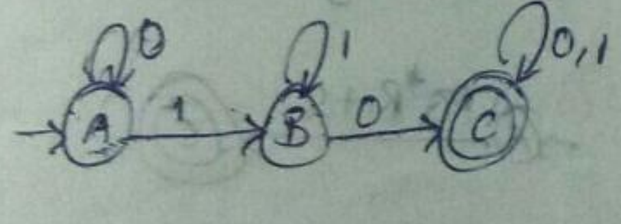
Let P, Q & R be 3 reg exp and $Q \neq \phi$. The expression

$R = Q + RP$, have one and only solution

$R = QP^*$

and P does not contain ϵ

Consider the following FA, M, Determine the reg exp corresponding to it



$0^*11^*0(0+1)^*$

Using Arden's theorem

$A = A0 + \epsilon$

$B = A1 + B1$

$C = B0 + C0 + C1$

> Work all incoming edges of a state including state from which edge is coming and label. For initial state include ϵ.

Now we need to obtain expression at C.

$A = A0 + \epsilon$

$A = \epsilon + A0$

i.e., $R = Q + RP$

$R = QP^*$

$\Rightarrow A = \epsilon 0^* = 0^*$

Now

$B = A1 + B1$

$B = 0^*1 + B1$

i.e., $R = Q + RP$

$R = QP^*$

$B = 0^*11^*$

Now

$C = B0 + C0 + C1$

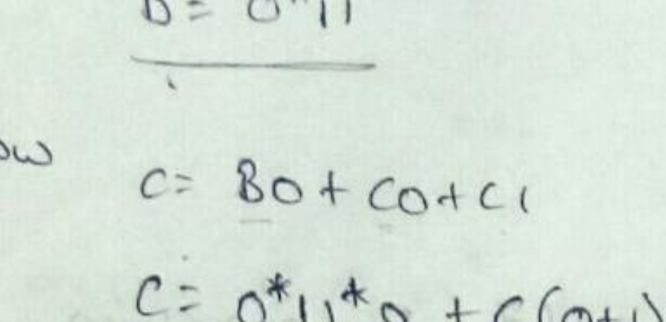
$C = \underset{R}{0^*11^*} \underset{Q}{0} + \underset{R}{C}\underset{P}{(0+1)}$

$C = 0^*11^*0(0+1)^*$

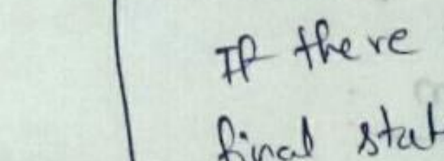
> Solving equations we find the expression at a final state which is the required reg exp. If there are more than one final states then union of expressions at those final states will be the required reg exp

State Elimination Method:

Rule 1:

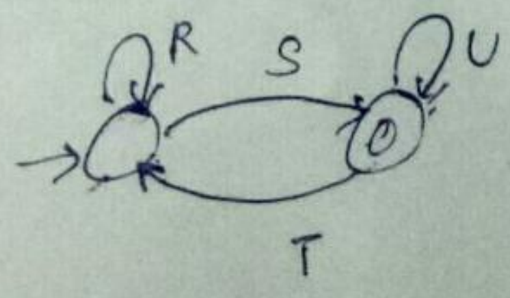


eliminating B



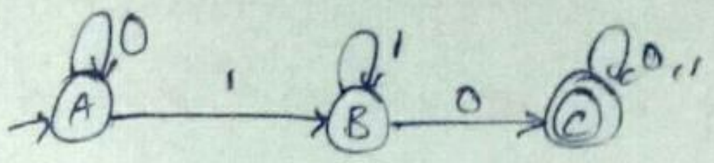
$\therefore r = PQ^*R + S$

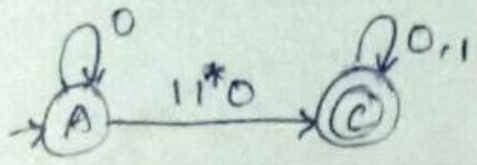
Rule 2:



Final expression: $(R + SU^*T)^*SU^*$ (or) $R^*S(U + TR^*S)^*$ } Analyze

Eg:

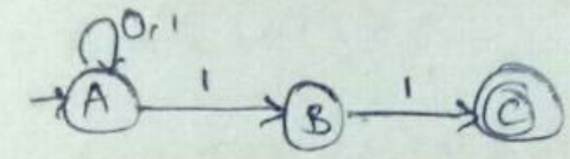


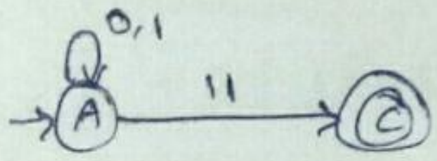


$$r = (0 + \underbrace{11^*0}_{S}\ \underbrace{(0+1)^*}_{R}\ \underbrace{\phi}_{T})^*\ 11^*0\,(0+1)^* \quad \text{[from rule 2]}$$

$$r = 0^*11^*0(0+1)^*$$

Eg: Find the regexp of FA 'M' shown in fig





$(0+1)^*11$

$A = A0 + A1 + \epsilon$

$B = A1$

$C = B1$

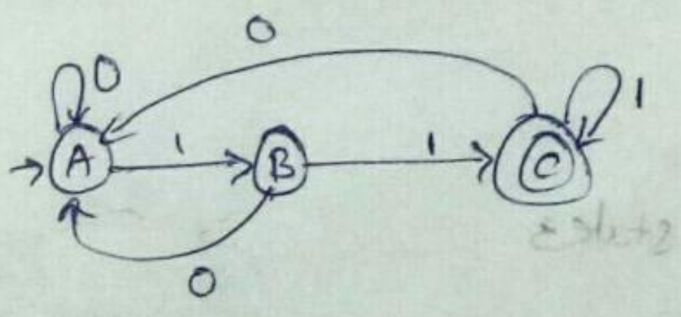
Now $A = \epsilon + A(0+1)$

$A = \epsilon(0+1)^* = (0+1)^*$

$B = A1 = (0+1)^*1$

$C = B1 = (0+1)^*11$

Eg: Find reg Exp



~~$A = A0 +$~~

$A = A0 + C0 + B0 + \epsilon$ ↳①

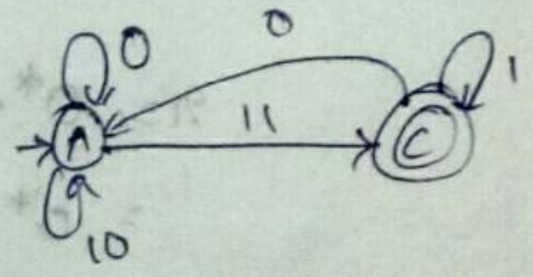
$B = A1$ — ②

$C = B1 + C1$ — ③

$C = B1 + C1$

$C = A11 + C1$ (from ②)

$C = A111^*$ — ④



$((0+10)+111^*0)^*111^*$

$= (0+10+111^*0)^*111^*$

Substitute ④ & ② in ①

$$A = A0 + A10 + A111^*0 + \epsilon$$

$$A = \epsilon + A(0 + 10 + 111^*0)$$

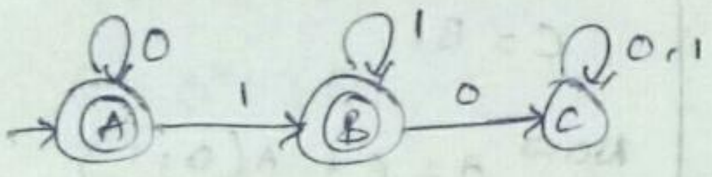
$$A = \epsilon(0 + 10 + 111^*0)^*$$

$$A = (0 + 10 + 111^*0)^*$$

$$C = A111^*$$

$$C = (0 + 10 + 111^*0)^* 111^*$$

Eg: ~~firing~~ Find reg exp



$$A = A0 + \epsilon$$

$$B = A1 + B1$$

$$C = B0 + C0 + C1$$

$$A = \epsilon + A0$$

$$A = \epsilon 0^* = 0^*$$

$$B = 0^*1 + B1$$

$$B = 0^*11^*$$

Since Both A & B are final states

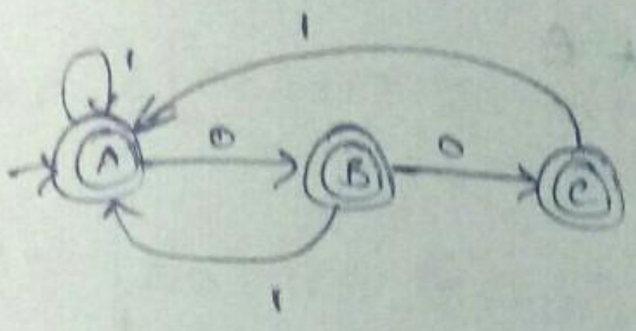
$$\eta = 0^* + 0^*11^*$$

$$= 0^*(\epsilon + 11^*)$$

$$= 0^*(\epsilon + 1^+)$$

$$= 0^*1^+$$

Q. Find the reg exp for DFA



$A = A1 + B1 + C1 + \epsilon$

$B = A0$

$C = B0$

$C = B0 = A00$

$\Rightarrow A = A1 + B1 + C1 + \epsilon$

$A = \epsilon + A1 + A01 + A001$

$A = \epsilon + A(1 + 01 + 001)$

$A = \epsilon(1 + 01 + 001)^*$

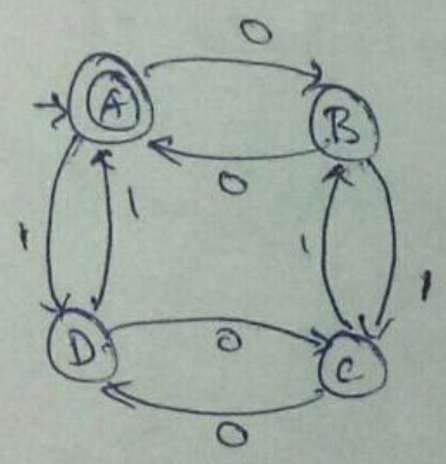
$C = A00$

$= (1 + 01 + 001)^* 00$

$B = A0 = (1 + 01 + 001)^* 0$

$A + B + C = (1 + 01 + 001)^* (\epsilon + 0 + 00)$

Q.



$A = B0 + D1 + \epsilon$

$B = A0 + C1$

$C = B1 + D0$

$D = A1 + C0 \Rightarrow D = (B0 + D1)1 + (B1 + D0)0$

$D = (B01 + B10) + D(11 + 00)$

$D = (B01 + B10 + 1)(11 + 00)^*$

$A = B0 + (B01 + B10 + 1)(00+11)^* 1 + \epsilon$

$A = B[0 + (01+10+1)(00+11)^* 1] + \epsilon$

$C = B1 + D0$

$= B1 + (B01 + B10 + 1)(00+11)^* 0$

$= B[1 + (01+10+1)(00+11)^* 0]$

$B = A0 + C1$

$= (B[0 + (01+10+1)(00+11)^* 1] + \epsilon)0 + B[1 + (01+10+1)(00+11)^* 0]1$

$= B00 + B[01 + 1 + (01+10+1)(00+11)^* (1+0)]$

$C = B1 + D0$

$= A01 + C11 + A10 + C00$

$C = A(01+10) + C(00+11)$

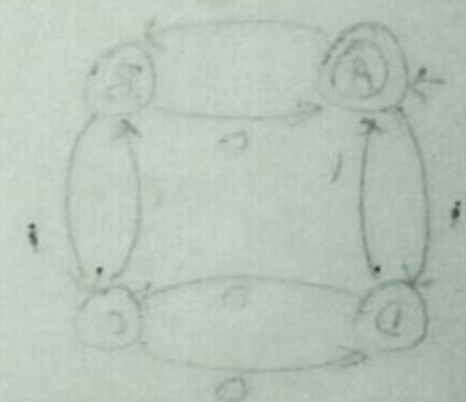
$C = A(01+10)(00+11)^*$

We need the expression for A.
So we change all the expression into terms of A

Now $B = A0 + C1$

$= A0 + A(01+10)(00+11)^* 1$

$= A[0 + (01+10)(00+11)^* 1]$



Now $D = A1 + C0$

$= A1 + A(01+10)(00+11)^* 0$

$= A[1 + (01+10)(00+11)^* 0]$

$A = \epsilon + B0 + D1$

$= \epsilon + A[0 + (01+10)(00+11)^* 1]0 + A[1 + (01+10)(00+11)^* 0]1$

$= \epsilon + A[00 + (01+10)(00+11)^* 10] + A[11 + (01+10)(00+11)^* 01]$

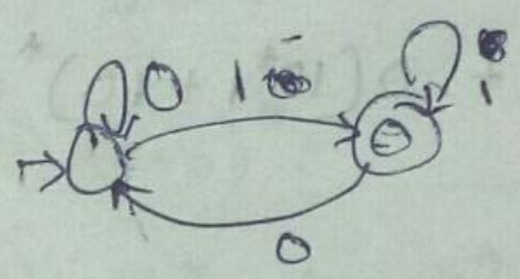
$$A = \epsilon + A[00+11+(01+10)(00+11)^*(10+01)]$$

$$A = \epsilon[00+11+(01+10)(00+11)^*(10+01)]^*$$

$$A = [00+11+(01+10)(00+11)^*(10+01)]^*$$

Problems:

(2014) which of the reg exp given below represent the following DFA



I. $0^*1(1+00^*1)^*$

II. $0^*1^*10+11^*0^*1$

III. $(0+1)^*1$

a) I & II b) I & III c) II & III d) I, II, III

$R = (0+10^*0)^*10^*$

$L = \{1, 01, 11, 111, 001, 011, 101 \ldots\}$ ending with 1

I = $0^*1(1+00^*1)^*$ clearly directly represents DFA

II = $(0^*1^*+11^*0^*)1$

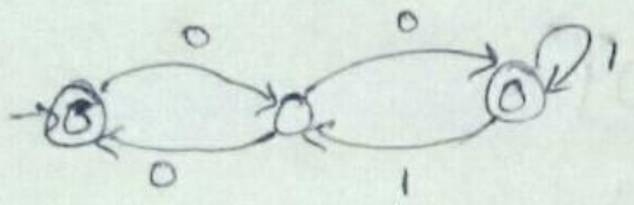
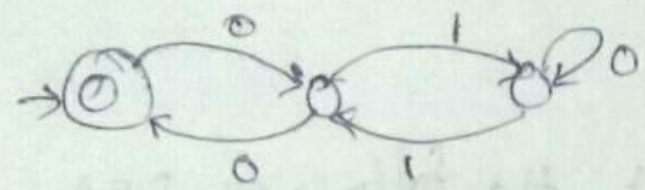
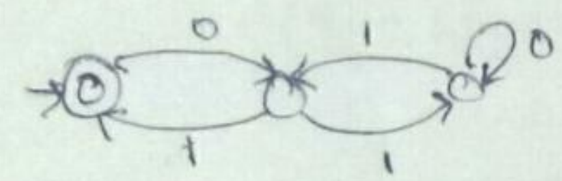
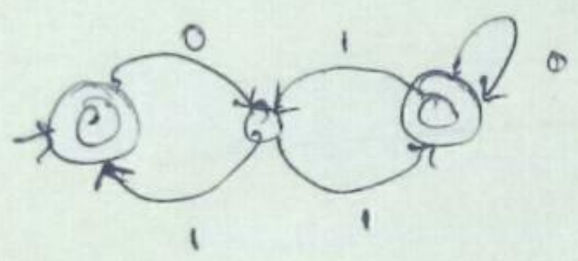
1011 is not accepted by II

III = $(0+1)^*1$ this is ending 1 clearly

∴ I & III

(2008) Match the following NFAs with the reg exp

(2)

NFA	Reg Exp
P) 	1. $\epsilon + 0(01^*1+00)^*01^*$
Q) 	2. $\epsilon + 0(10^*1+00)^*0$
R) 	3. $\epsilon + 0(10^*1+10)^*1$
S) 	4. $\epsilon + 0(10^*1+10)^*10^*$

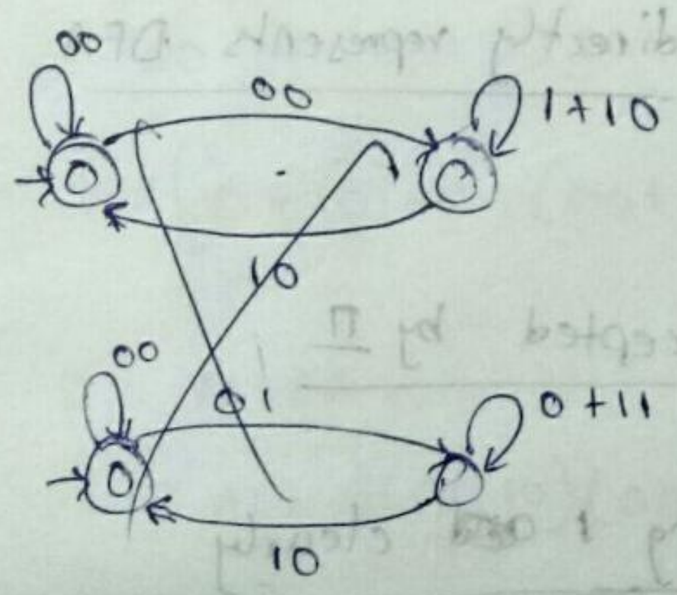
P Q R S

a) 2 1 3 4

b) 1 3 2 4

✓ c) 1 2 3 4

d) 3 2 1 4

P) 

Q) $\epsilon + [00 + 01(0+11)^*10]^*$

$\epsilon + [0(0+1(0+11)^*10)]^*$

P → 001

-in (2) it always ends with 0

∴ eliminate opt (a)

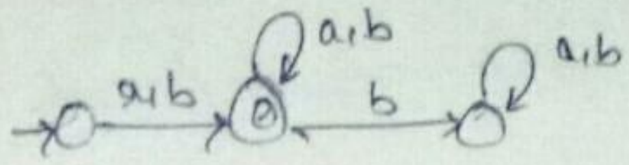
P → 00 in (3) ends with 1

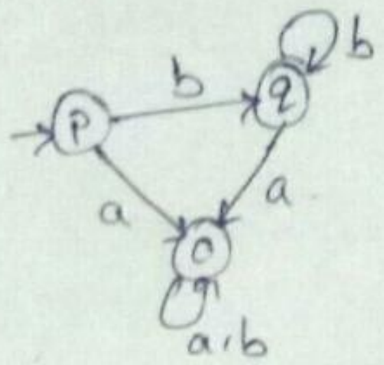
∴ eliminate opt (d) ⇒ P=1

Now Q ends with 0 or ε

Now R ends with 1

∴ C ✓

b) 

c) 

d) None;

from fig $r_1 = a(a+b)^* + ba(a+b)^* = (a+ba)(a+b)^*$

b) => accepts string starting with bb

e) also accepts few string staring with bb

$\therefore a$

6. Q: which of the following reg exp accept set of all strings not containing 100 as substring

a) $0^*(1+0)^*$

b) 0^*1010^*

c) $0^*1^*01^*$

d) $0^*(10+1)^*$

$L = \{\epsilon, 0, 1, 00, 01, 10, 11, 001, 010, 011 \ldots\}$

a) => $\{\epsilon, 0, \underset{\times}{1}, 00, \underset{\times}{01} \ldots\}$ ×

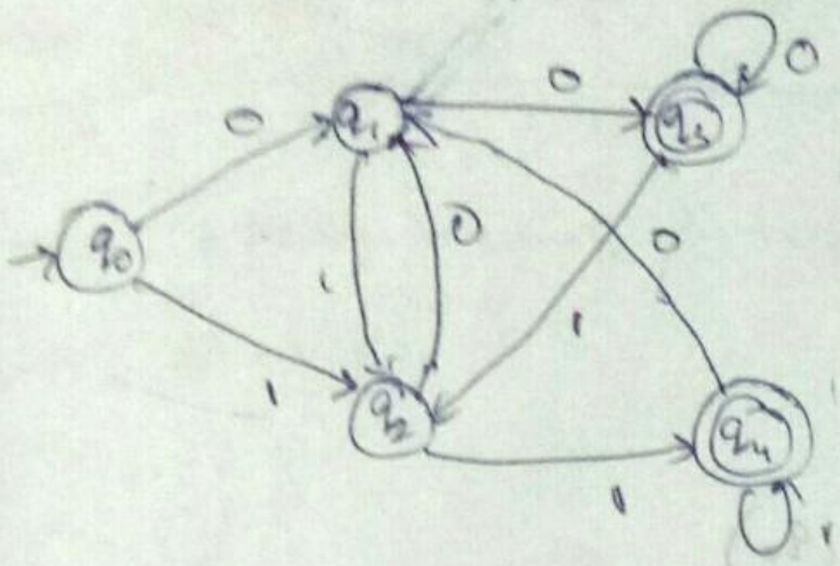
b) => $\{\epsilon_{\times} \ldots\}$ ×

c) => $\{\underset{\times}{\epsilon} \ldots\}$ ×

d) => $\{\epsilon, 0, 1, 00, 01, 10, 11, 001, 010, 011 \ldots\}$ ✓

Material problems

(10)



(17) question means that number divisible by 29099

∴ 29099 no of states

(24)

~~(a+b)* ab~~

FSM with Outputs:

An FSM is 6 tuple system

$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$

where Q = finite set of states

Σ = i/p alphabet

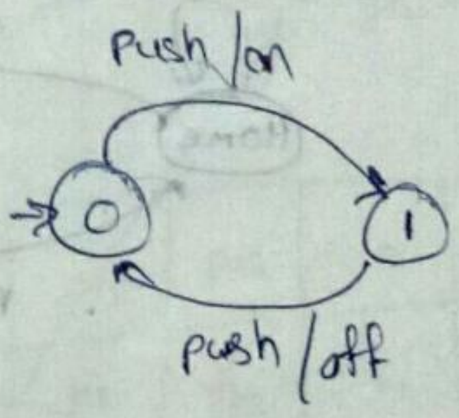
Δ = o/p alphabet

δ = transition function

λ = o/p function

q_0 = initial state

Behavior of digital computer switch



$Q = \{0, 1\}$

$\Sigma = \{push\}$

$\Delta = \{on, off\}$

δ: $\delta(0, push) = 1$

$\delta(1, push) = 0$

λ: $\lambda(0, push) = on$

$\lambda(1, push) = off.$

$q_0 : 0$

This FSM with o/p is classified into two types

1. Mealy machine
2. Moore machine

Mealy machine:

It is a 6 tuple system

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

where the o/p function $\lambda : Q \times \Sigma \rightarrow \Delta$

i.e., o/p Mealy machine depends on both presents state and i/p.

Moore Machine:

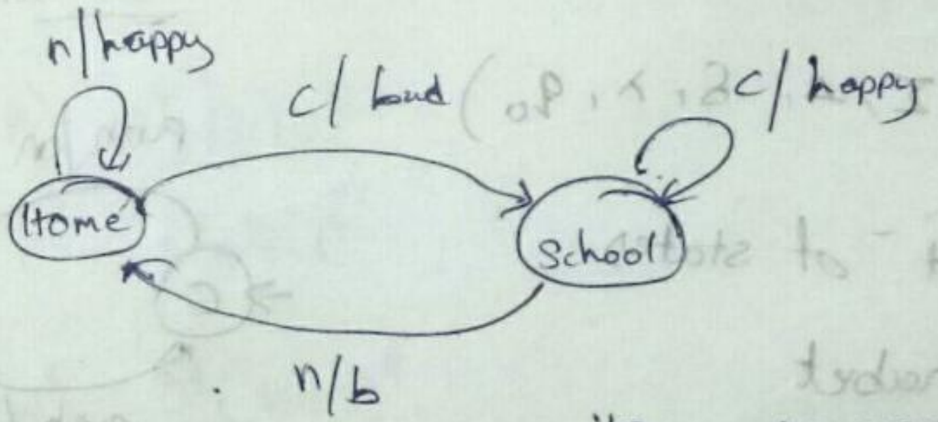
→ 6 tuple system

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

o/p function $\lambda : Q \rightarrow \Delta$

i.e., o/p of the machine depends only on current state.

Eg of Mealy:



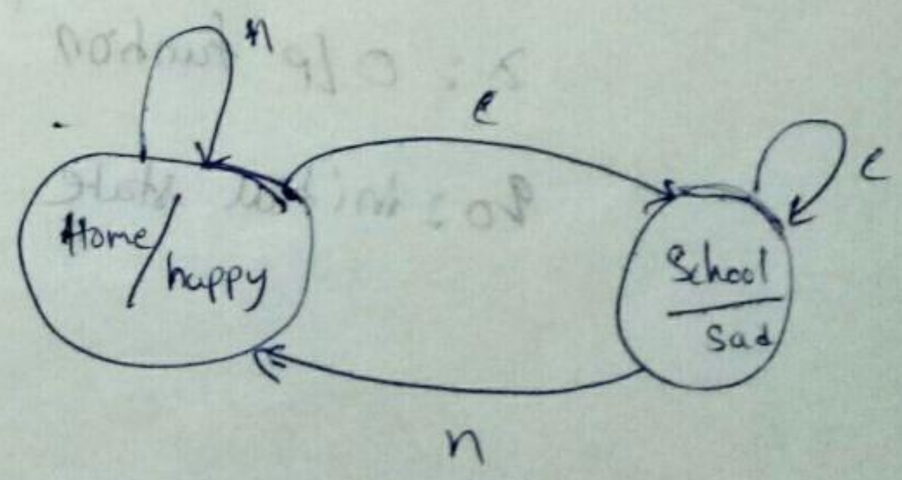
	n	c
→H	H,h	S,b
S	H,b	S,h

i/p: w = ccnn

b h b h

Eg of Moore

o/p	ps	n	c
h	→H	H	S
b	S	H	S



i/p:

w = nccn

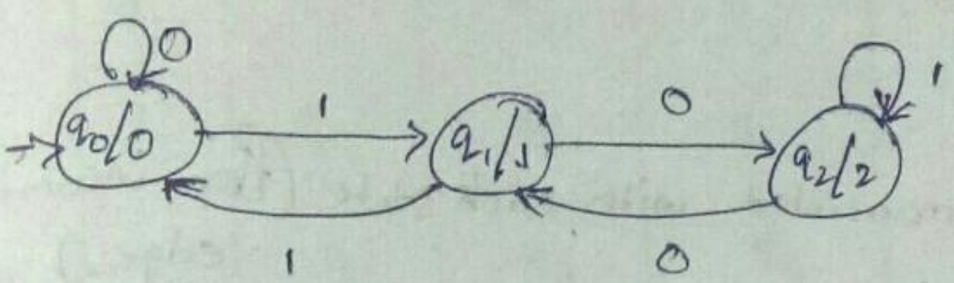
H → H → S → S → H

h h b b h

→ For mealy machine if length of i/p is n then length of o/p is n
→ For moore machine if length of i/p is n then length of o/p is n+1

→ Design Moore & Mealy machines to recognize residue of mod 3 numbers

Eg: for the binary integer i/p

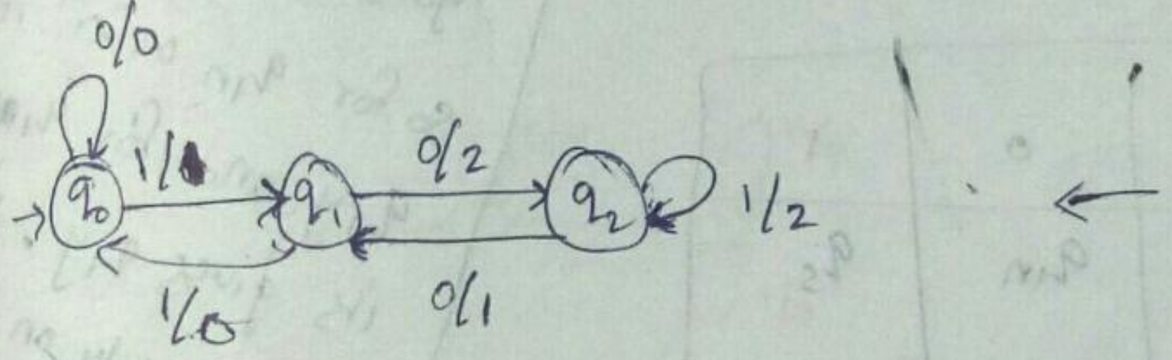


O/P	PS	0	1
0	→q_0	q_0	q_1
1	q_1	q_2	q_0
2	q_2	q_1	q_2

1010 → q_1 i.e., 1 (remainder)

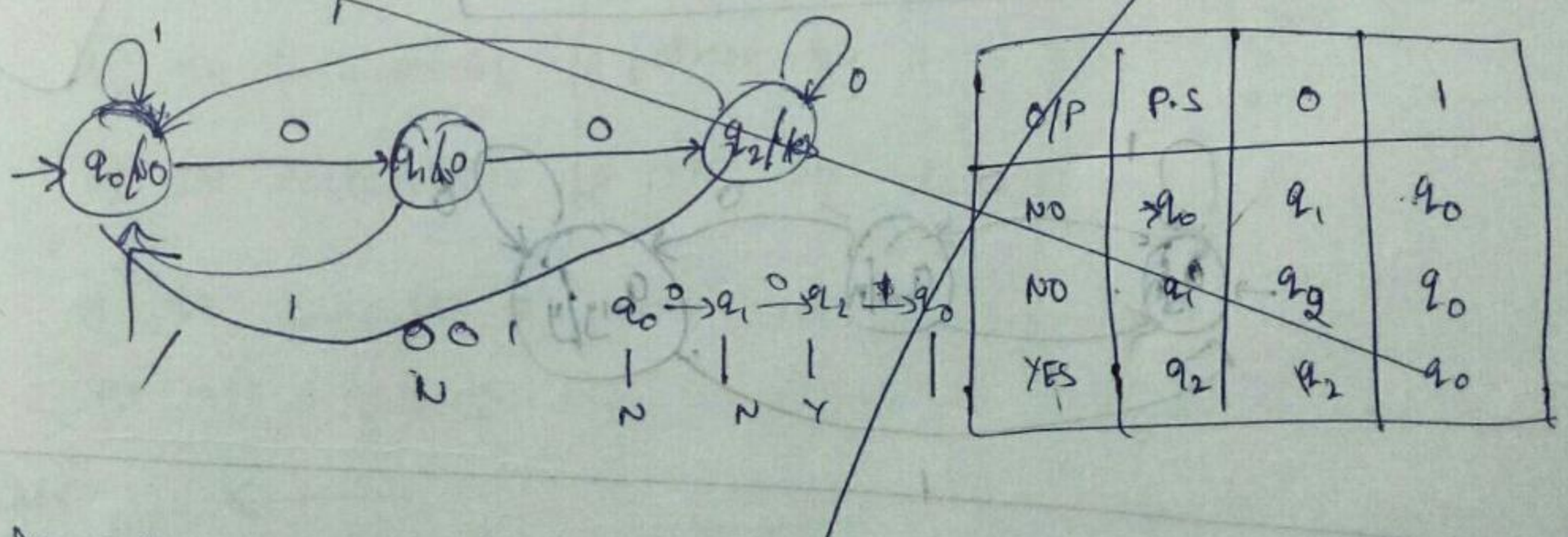
⇓ Moore to Mealy

Mealy to Moore: Moore to Mealy:



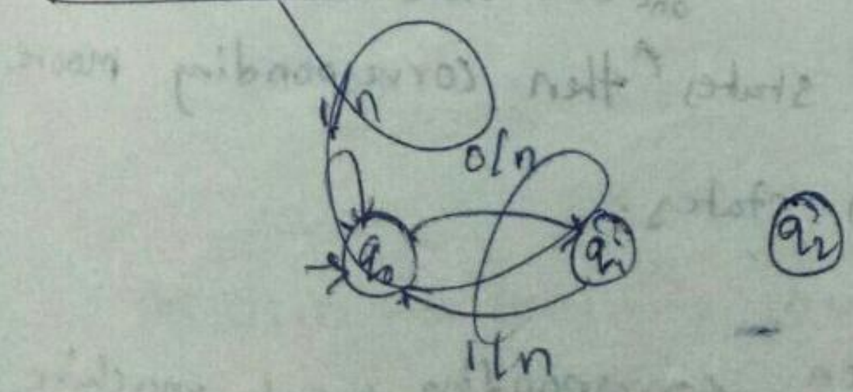
PS	0	1
→q_0	q_0,0	q_1,1
q_1	q_2,2	q_0,0
q_2	q_1,1	q_2,2

Q: Design Mealy & Moore machines to recognize set of all strings ending with 00



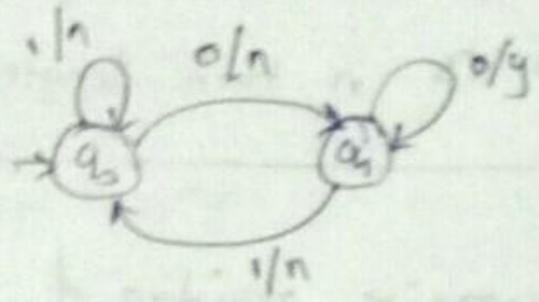
O/P	P.S	0	1
NO	→q_0	q_1	q_0
NO	q_1	q_2	q_0
YES	q_2	q_2	q_0

Mealy to Moore



5: Design Mealy & Moore for strings ending with 00.

(2)



	0	1
$\rightarrow q_s$	q_1, n	q_s, n
q_1	q_1, y	q_s, n

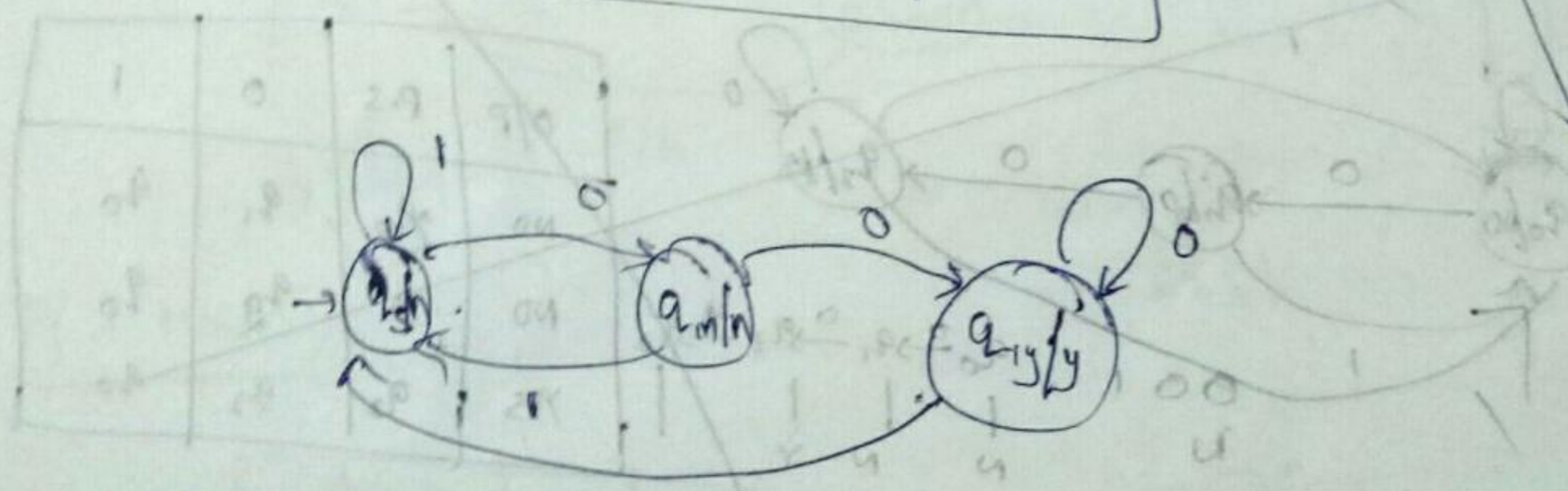
✱ Mealy to Moore:

Step 1: Identify o/p associated with each state (from incoming edges)

$q_s - n$	$q_s - n$
$q_1 <^{n}_{y}$	$q_{1n} - n$ $q_{1y} - y$

o/p	P.S	0	1
n	$\rightarrow q_s$	q_{1n}	q_s
n	q_{1n}	q_{1y}	q_s
y	q_{1y}	q_{1y}	q_s

Here for q_1 in mealy on 0 it goes to q_1 giving o/p y so for q_{1n} on '0' it giving q_{1y} and for q_{1y} on 0 its gives q_{1y}. for q_1 in mealy on 1 it goes to q_s giving o/p n. So it goes to q_s



Note:

→ If Mealy machine has n no of states and each state associated with m o/p then corresponding Moore machine can have at most n x m states.

→ If moore machine has n states then corresponding mealy machine will have exactly n states

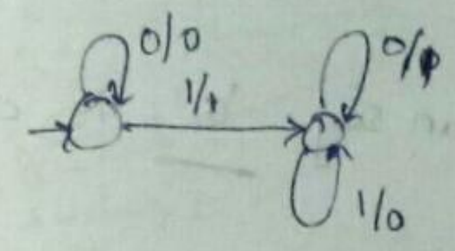
9: Design a ~~more~~ moore machine to recogonize residue of mod 5 number over the ~~i/p~~ ternary i/p alphabet

o/p	P.S	0	1	2
0	$\rightarrow q_0$	q_0	q_1	q_2
1	q_1	q_3	q_4	q_0
2	q_2	q_1	q_2	q_3
3	q_3	q_4	q_0	q_1
4	q_4	q_2	q_3	q_4

Same pattern can be use in this case too

Problems:

8: Consider the following FSM shown in the figure



It accepts the binary string from least significant bit positions and o/p the result in same order. which of the following is true

a) It computes 1's complement of

b) It increaments i/p pattern by 1.

c) It decreaments i/p patter by 1

d) It computes 2's complements.

Consider

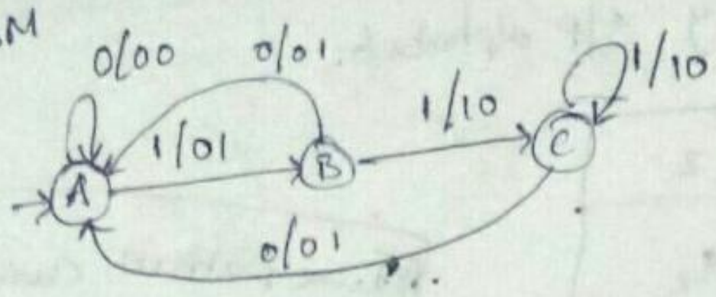
i/p: 0 1 1 0

o/p: 1 0 1 0

0110 $\xrightarrow{1's}$ 1001 $\xrightarrow{2's}$ 1010

∴ 2's complement.

Q: The FSM



(N)

Each edge label is x/y, x stands for 1 bit i/p and y stands for 2 bit o/p

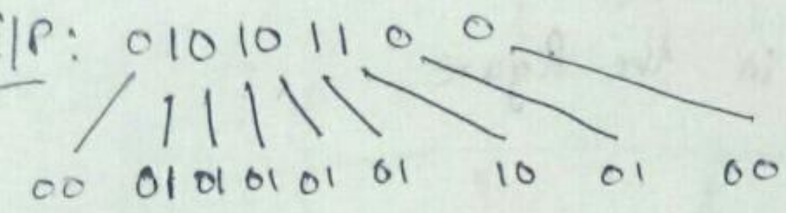
a) outputs the sum of the present and previous bits of the i/p

b) outputs 01 whenever i/p sequence is 11

c) outputs 00 whenever i/p sequence contains 10

d) none

i/p: 0 1 0 1 0 1 1 0 0



From above i/p & o/p we can say 'a' is correct answer

Q: The FSM in following state table has single i/p 'x' & single o/p 'z'

(N)

	x=1	x=0
A	D,0	B,0
B	B,1	C,1
C	B,0	D,1
D	B,1	C,0

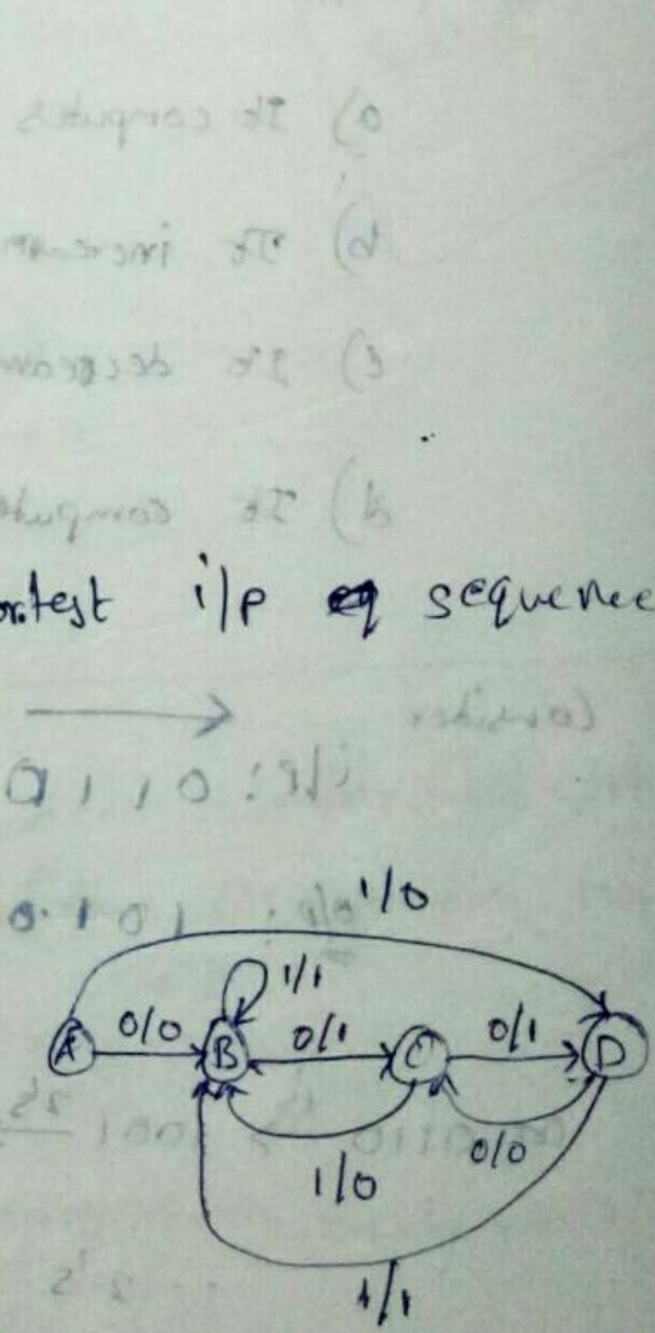
If the initial state is unknown, then shortest i/p sequence to reach final state C is ____

a) 01 b) 10 c) 101 d) 110

for A

$A \xrightarrow{1} D \xrightarrow{0} C$

∴ 10



Q: Determine min length of the string to reach the state D with ~~0/p~~ from the initial state A: (from previous question)

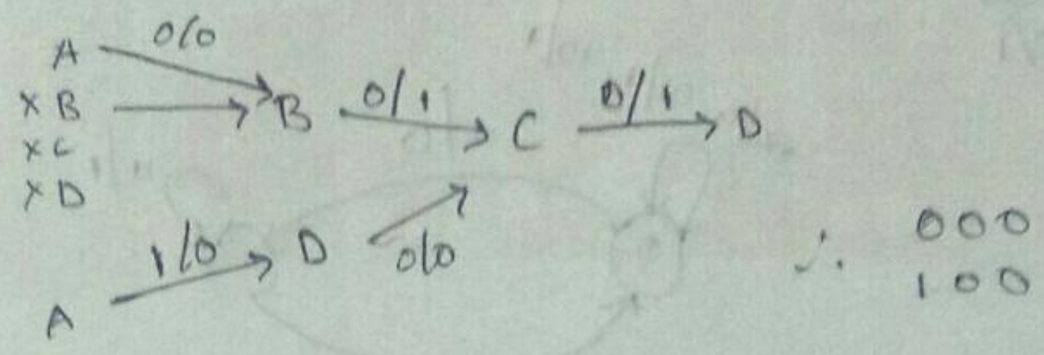
(N)

Sol:

Ans: 000

Q: No of strings of length 3 to reach state D with o/p '1' starting at A.

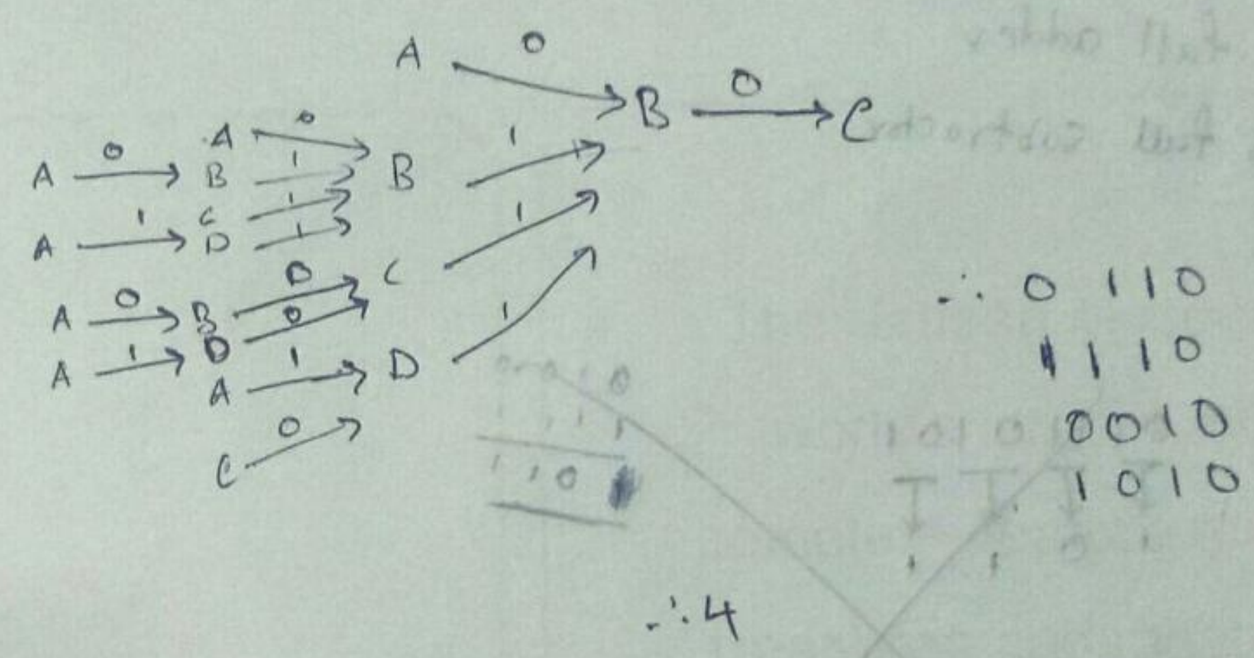
(N)



∴ 000
100

Q: No of strings of length 4 to reach state C with o/p '1' from A

(N)



∴ 0 110
1110
0010
1010

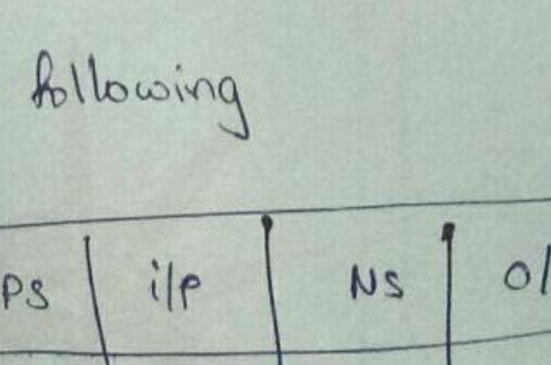
∴ 4

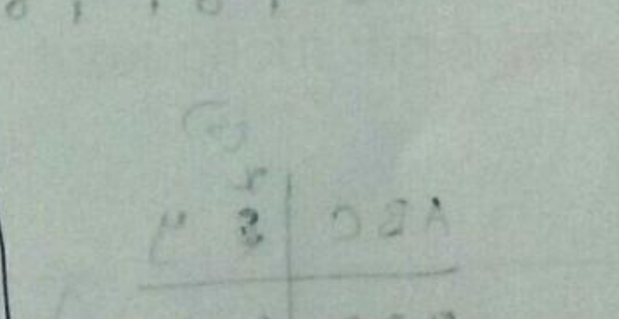
Q: Consider the following

(N)

PS	i/p	NS	o/p
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

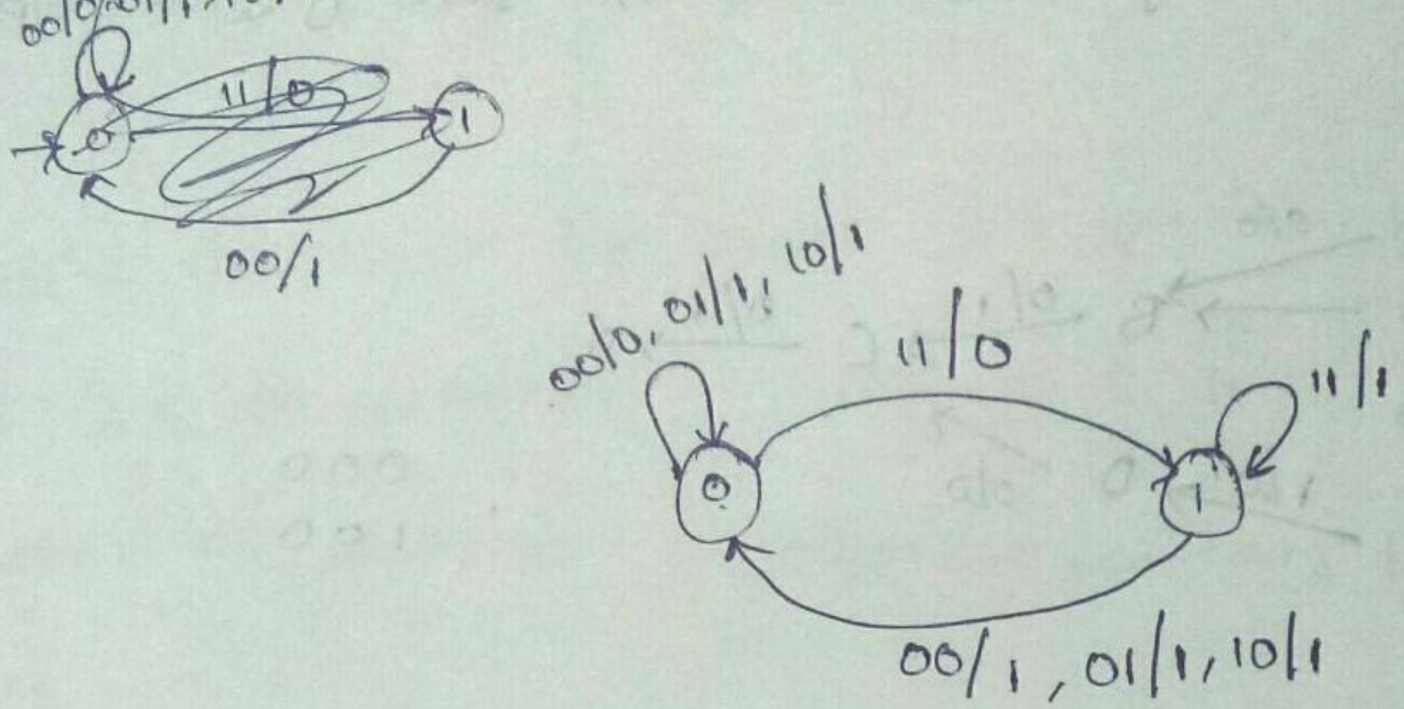
If initial state of the machine is 0 How many strings of length 3 are existed to reach 1 with o/p '1'





∴ 010
100

Q: Consider the following FSM shown in the figure. Here each edge label is x/y where x is a 3bit number obtained by combining node label followed by edge labels, y is a 2 bit number obtained by combining edge label followed by node label. which of the following is true.



a) M is half adder

b) M is full adder

c) M is full subtractor

✓d) None