MID TERM

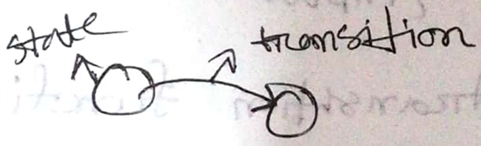
Pre defined

Deterministic Finite Automation (DFA)

Finite Automata: Finite input, Application unlimited

Problem
- No memory.

- একটা state ~~থেকে~~ থেকে আরেকটা state এ transition থাকবে.



- minimum one final state. more than one possible

Terminologies

→ q_1 [initial state]   → [transition]

(O) Final state/accept state

1 (input)   1,0 [1 or 0]

* ε → string empty

Formal Definitation - DFA

Tuple: specific order sequence.

Tuple: ( )

Q : set of state { }

Σ : " " alphabet/input

$\delta : Q \times \Sigma \to Q$ transition function

represent all possible combination

$F \subseteq Q$ [Q এর অনেক প্রকার member F এ থাকবে]

* string Thru machine বানালে wrong হবে. Language থেকে machine বানাতে হবে.

Problem statement

DFA (Formal definition)

A Finite Automaton that contains 5 tuples

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

Q = set of all state. { } [Q is a finite set called the states]

Σ = set of alphabet/input { } [Σ is a finite set called the alphabet]

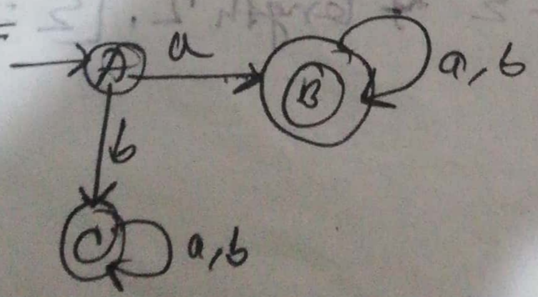
δ = Transition function [$\delta : Q \times \Sigma \rightarrow Q$ is the transition function]

$\delta = Q \times \Sigma \rightarrow Q$

q_0 = initial state [$q_0 \in Q$ is the start state]

F = set of final states. ◎ [$F \subseteq Q$ is the set of accept state]

#

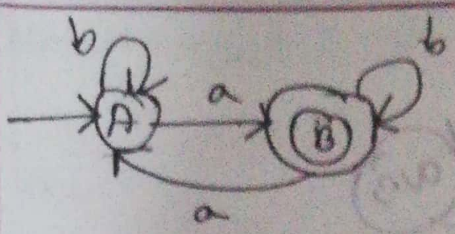


δ	a	b
$\rightarrow$ A	B	C
* B	B	B
C	C	C

F = {B}

CamScanner

Formal Definition



M_1

$M_1 = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{A, B\}$

$\Sigma = \{a, b\}$

$\delta : Q \times \Sigma \to Q$

$q_0 = A$

$F = \{B\}$

δ	a	b
A	B	A
B	A	B

OR

$\delta(A, a) = B$, $\delta(A, b) = A$

$\delta(B, a) = A$, $\delta(B, b) = B$

Transition Table — Transition Function

Formal Definition of DFA computation

$M = (Q, \Sigma, \delta, q_0, F)$

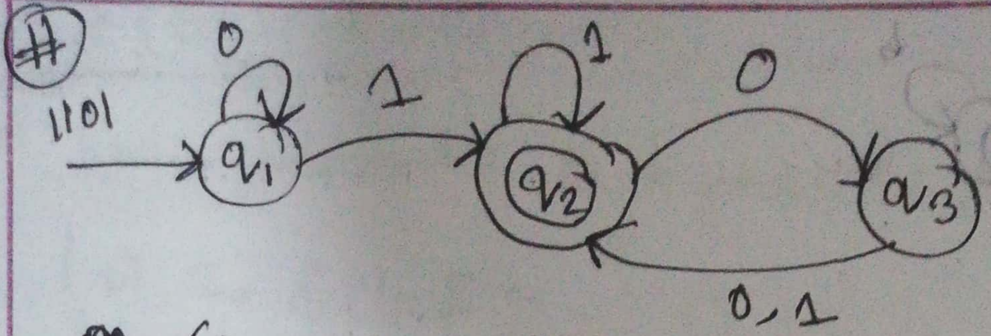
$W = w_1 w_2 \ldots w_n$

Now, $r_0 = q_0$

$\delta(r_i, w_{i+1}) = r_{i+1}$ $[i = 0, 1, 2, \ldots, n-1]$

$r_n \in F$

#



$M = (Q, \Sigma, \delta, q_0, F)$

$Q = \{q_1, q_2, q_3\}$ $\Sigma = \{0, 1\}$, $q_0 = q_1$, $F = \{q_2\}$

$\delta : Q \times \Sigma \to Q$

$\delta(q_1, 0) = q_1$, $\delta(q_1, 1) = q_2$

$\delta(q_2, 0) = q_3$, $\delta(q_2, 1) = q_2$

$\delta(q_3, 0) = q_2$ $\delta(q_3, 1) = q_2$

Simulation

Start in state $r_0 = q_1$ $[r_0 = q_0]$

$\delta(r_0, w_1) = \delta(q_1, 1) = q_2 = r_1$

$\delta(r_1, w_2) = \delta(q_2, 1) = q_2 = r_2$

$\delta(r_2, w_3) = \delta(q_2, 0) = q_3 = r_3$

$\delta(r_3, w_4) = \delta(q_3, 1) = q_2 = r_4 \to r_n \in F$

Accept, as the machine M_1 is an accept state q_2 at the end of the input string.

Construct a DFA which accepts set of all strings over the $\Sigma = \{a, b\}$ where each string starts with an 'a'

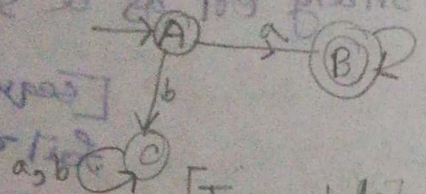
OR

$L(M) = \{ w \mid w$ starts with an 'a'$\}$

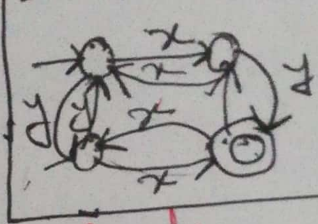
Solⁿ

$\Sigma = \{a, b\}$

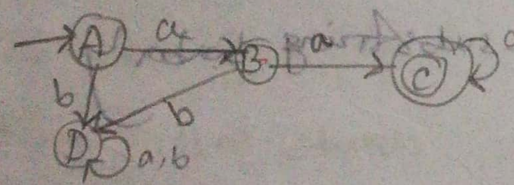
$L = \{a, ab, aa, abb, aaa \ldots\}$



[Trap state]



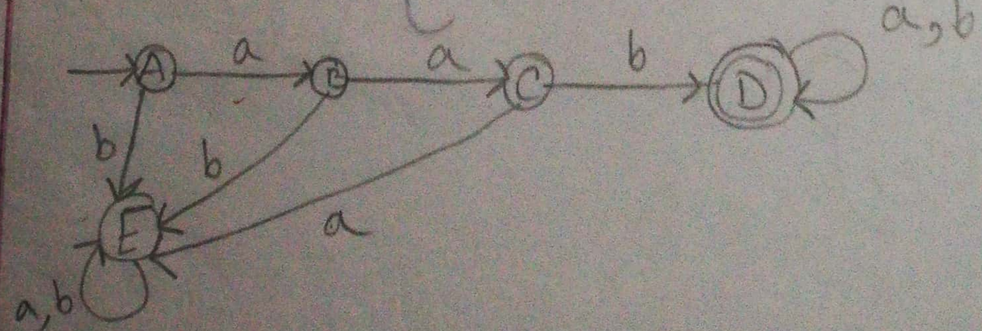
$\Sigma = \{a, b\}$ where each string starts



$\Sigma = \{a, b\}$, $w = aab$

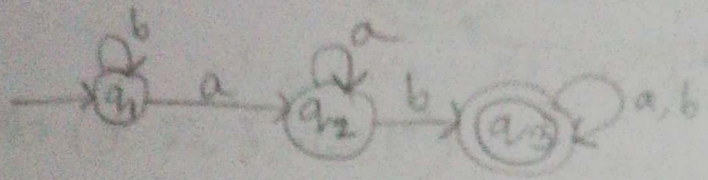
$\Sigma = \{a, b\}$

$L = \{aab, aaba, aabb \ldots\}$



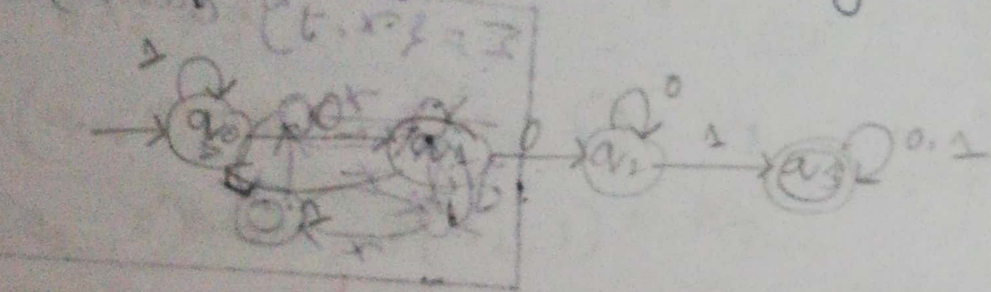
$\Sigma = \{a, b\}$

$L(m) = \{ w \mid w$ contains the string ab as a substring



$\Sigma = \{0, 1\}$

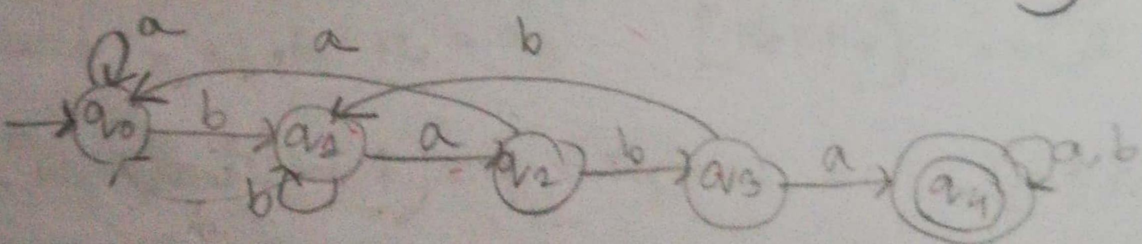
$L(m) = \{ w \mid w$ contains the string 011 as a substring



[Sequence fail করলে আবার শুরু থেকে শুরু করবো]

$\Sigma = \{a, b\}$

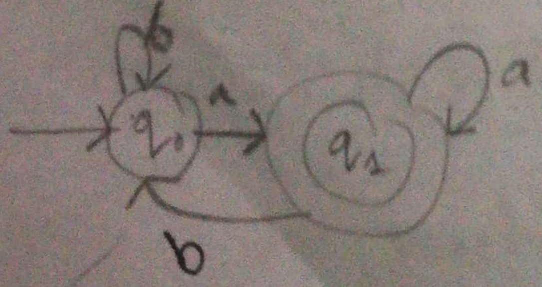
$L(m) = \{ w \mid w$ contain the substring baba $\}$



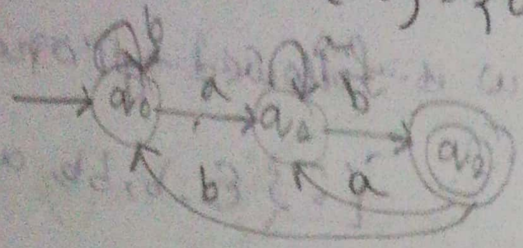
babb

$\Sigma = \{a, b\}$

$L(m) = \{ w \mid w$ ends with an 'a' $\}$

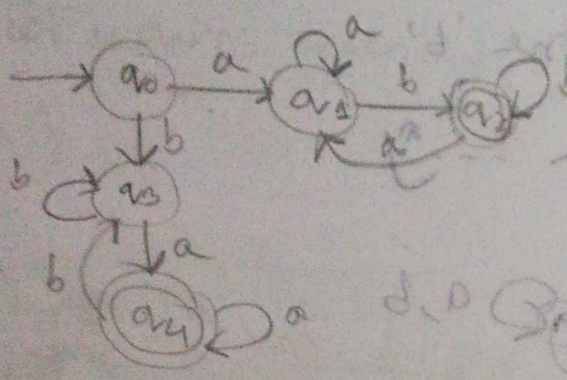


$\Sigma = \{a, b\}$. $L(M) = \{w \mid w$ ends with ab



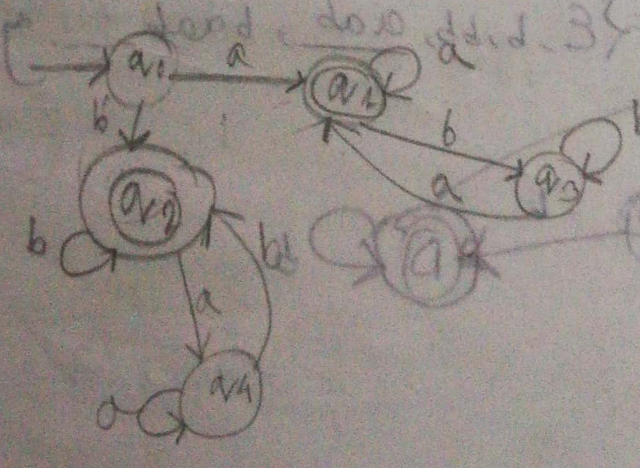
$\Sigma = \{a, b\}$

$L(M) = \{w \mid w$ starts and ends with diff. symbol$\}$



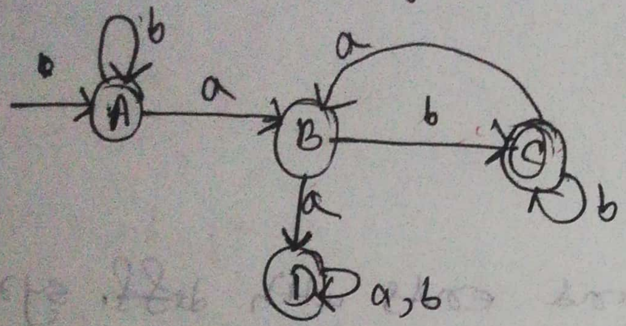
$\Sigma = \{a, b\}$

$L(M) = \{w \mid w$ starts and ends with same symbol$\}$



$\Sigma = \{a, b\}$

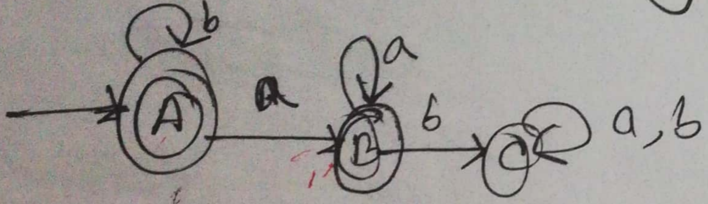
$L(M) = \{w \mid w$ every a in w is followed by one 'b' $\}$



$L = \{\varepsilon, b, bb, ab, bab ..$

Never followed by one 'b'

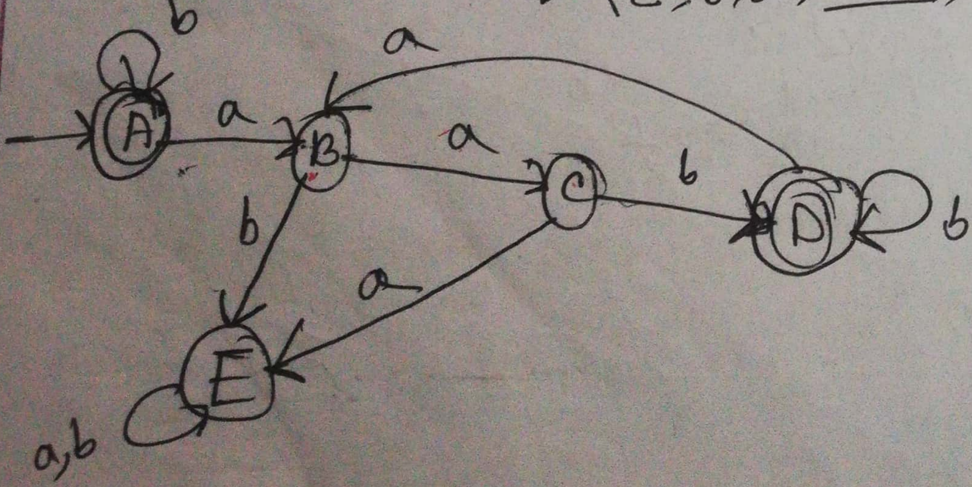
$L = \{\varepsilon, b, b, b, a, aa, ba --- \}$



$\Sigma = \{ab\}$

$L(M) = \{w \mid w$ every a in w is followed by 'ab' $\}$

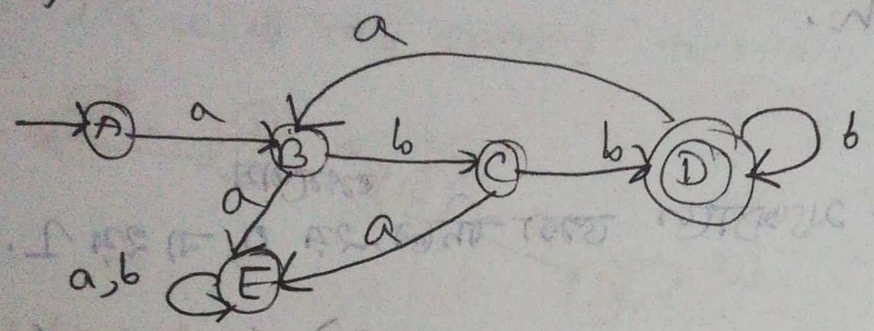
$L = \{\varepsilon, b, bb, \underline{aab}, baab \_\_\_ \}$



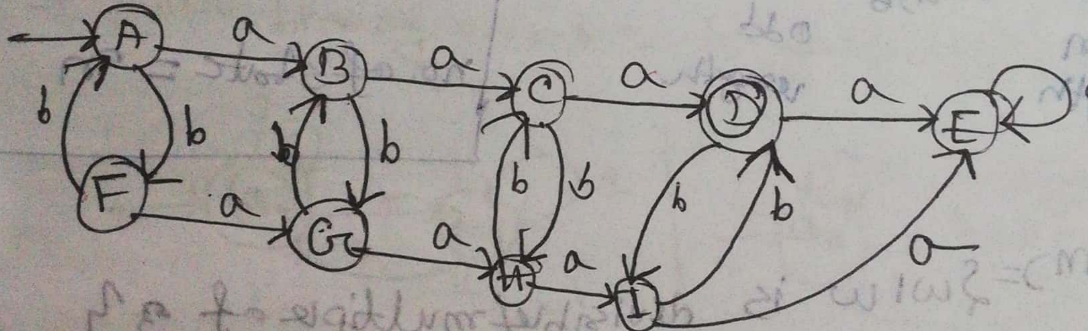
CamScanner

*Sub string এ trap state থাকে না।

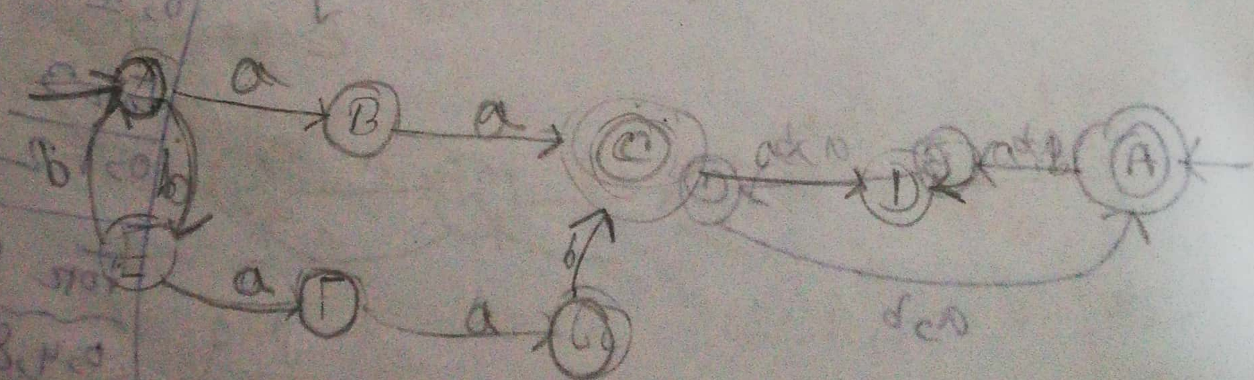
(i) { w | each 'a' in w is followed by at least two b }



(ii) { w | contains even no of 'b' and exactly three 'a' }



(iv) { w | w contains at least two 'b' and exactly two 'a' }



divisible by 2 / multiple of 2 / $|w| \bmod 2 = 0$ / even length.

$\Sigma = \{a, b\}$

*2 দ্বারা কোন সংখ্যাকে ভাগ দিলে ভাগশেষ হয় 0, না হয় 1.

∴ state 2টি

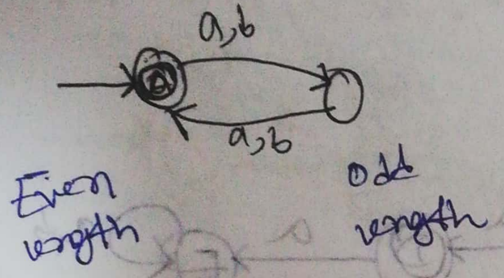
$L = \{\varepsilon, aa, bb, ab, ba \ldots\}$

2)2(1
2
0

2)3(1
2
1

2)5(2
4
1

2)4(2
4
0



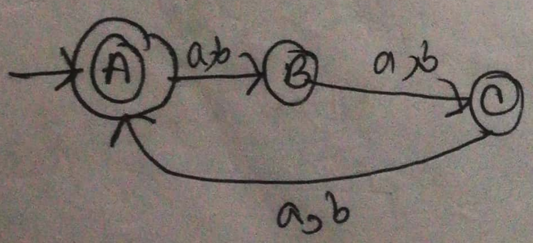
Note: divisible by n

no. of state = n

$L(M) = \{w \mid w$ is divisible/multiple of 3$\}$

$\Sigma = \{a, b\}$

$L = \{\varepsilon, aaa, bbb, aab, bab, bba \ldots\}$



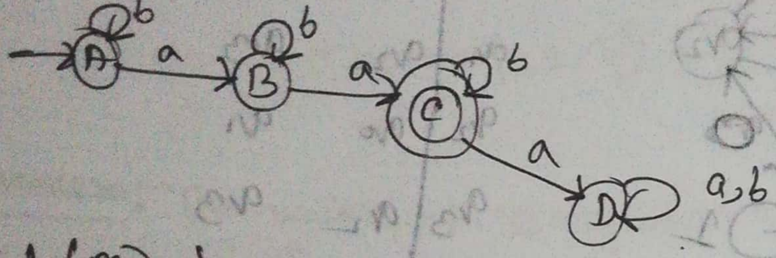
2
0, 1

For 3
0, 1, 2

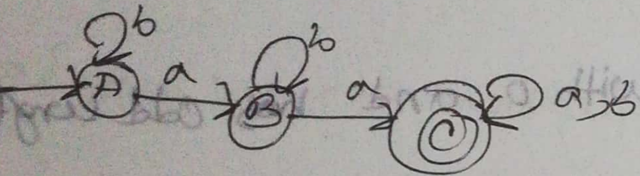
For 4
0, 4, 8, 12

$\Sigma = \{a, b\}$

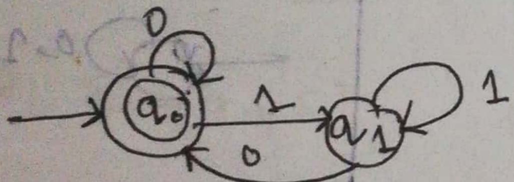
$L(m) = \{ w \mid w$ has exactly two a's



$L(m) = \{ w \mid w$ has at least two a's $\}$



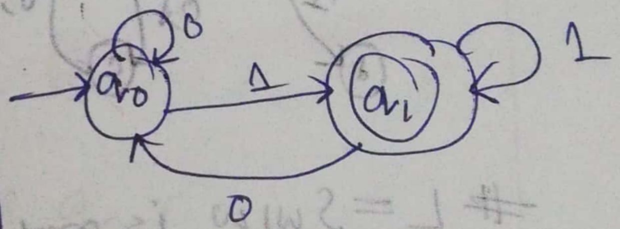
$L = \{ w \mid w$ binary number is divisible by 2 $\}$ $\Sigma = \{0, 1\}$



0=0 1=1

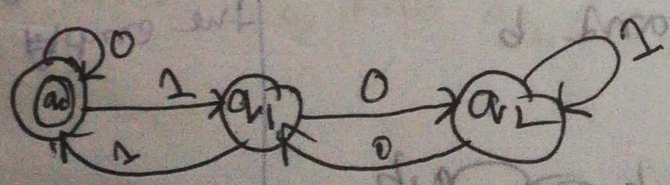
2=10 3=11

4=100 5=101



Not divisible by 2

divisible by 3



0=0 1=1 2=10

3=011 4=100 5=101

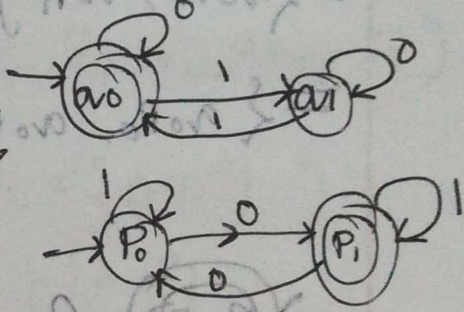
	0	1
$\rightarrow * q_0$	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

Regular Operation

1. Union $A=\{1,2\}$ $B=\{1,3\}$

$\therefore A\cup B=\{1,2,3\}=\{w \mid w\in A \text{ or } w\in B\}$

Theorem: The union of 2 regular language is also a regular language.



$M_1 \rightarrow L_1=\{w \mid w$ has even number of 1's$\}$

$M_2 \rightarrow L_2=\{w \mid w$ .. odd .. .. 0's$\}$

$L_1\cup L_2 = M_1\cup M_2$ $\therefore L=M$

$M_1=(Q_1,\Sigma_1,\delta_1,q_{0_1},F_1)$ $M_2=(Q_2,\Sigma_2,\delta_2,q_{0_2},F_2)$

$M=(Q,\Sigma,\delta,q_0,F)$

$Q=$ ~~Point~~ Pair of state, one from M_1, one from M_2

$=\{(q_1,q_2), q_1\in Q_1, \text{ and } q_2\in Q_2\}$

$=Q_1\times Q_2$

$=\{q_0,q_1\}\times\{P_0,P_1\}$

$=\{q_0P_0, q_0P_1, q_1P_0, q_1P_1\}$

$\Sigma=\Sigma_1\cup\Sigma_2$

$=\{0,1\}$

$q_0 = (q_{01}, q_{02})$

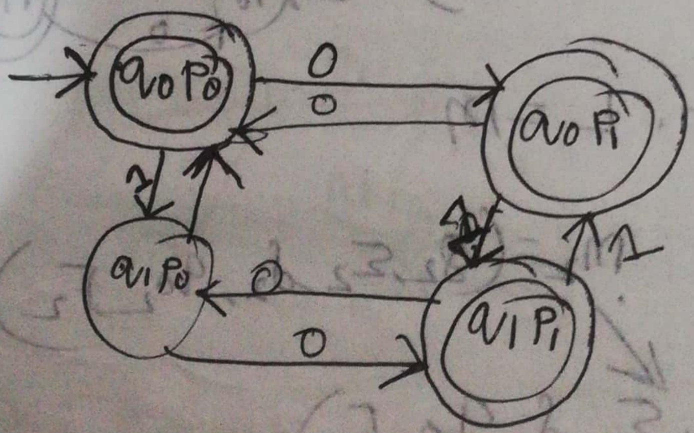
$= q_0 P_0$

$F = (F_1 \times Q_2) \cup (F_2 \times Q_1)$

$= \{q_0\} \times \{P_0, P_1\} \cup \{P_1\} \times \{q_0, q_1\}$

$= \{q_0 P_0, q_0 P_1\} \cup \{q_0 P_1, q_1 P_1\}$

$= \{q_0 P_0, q_0 P_1, q_1 P_1\}$



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

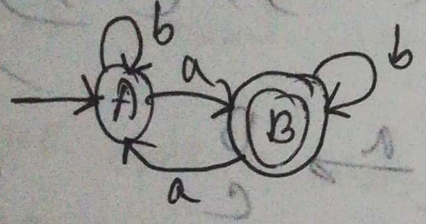
$= (\delta_1(q_1, a), \delta_2(q_2, a))$

$\begin{matrix} q_0 \\ P_0 \end{matrix} \xrightarrow{0} \begin{matrix} q_0 \\ P_1 \end{matrix}$

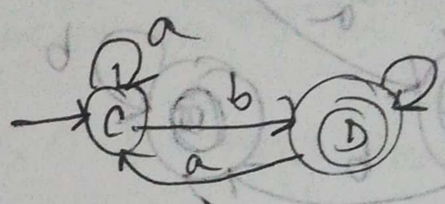
$\begin{matrix} q_0 \\ P_0 \end{matrix} \xrightarrow[1]{1} \begin{matrix} q_1 \\ P_0 \end{matrix}$

$L = \{ w \mid w$ has an odd number of a's or ends with a 'b' $\}$

L_1 L_2



M_1



M_2

$M_1 (Q_1, \Sigma_1, \delta_1, q_{01}, F_1)$

$Q_1 = \{A, B\}$

$\Sigma = \{a, b\}$

$q_0 = A$

$F_1 = \{B\}$

$M = M_1 \cup M_2$

$M_2 (Q_2, \Sigma_2, \delta_2, q_{02}, F_2)$

$Q_2 = \{C, D\}$

$\Sigma = \{a, b\}$

$q_0 = C$

$F_2 = \{D\}$

$Q = Q_1 \times Q_2$

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

$= \{AC, AD, BC, BD\}$

$\Sigma = \{a, b\}$

$q_0 = (q_{01}, q_{02})$

$= AC$

$F = (F_1 \times Q_2) \cup (F_2 \times Q_1)$

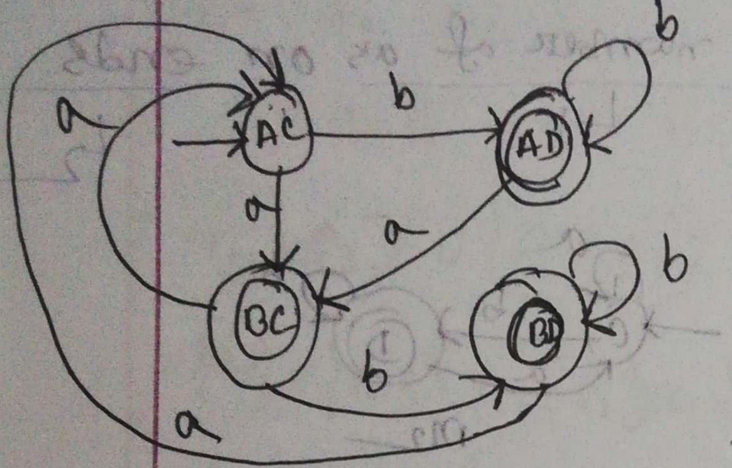
$= \{B\} \times \{C, D\} \cup \{D\} \times \{A, B\}$

$= \{BC, BD\} \cup \{AD, BD\}$

$= \{AD, BC, BD\}$

$BC \xrightarrow{a} AC \quad BD \xrightarrow{a} AC$

$BC \xrightarrow{b} BD \quad BD \xrightarrow{b} BD$



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

$= \delta_1(q_1, a), \delta(q_2, a)$

$\begin{matrix} A \\ C \end{matrix} \xrightarrow{a} \begin{matrix} B \\ C \end{matrix}$

$\begin{matrix} A \\ C \end{matrix} \xrightarrow{b} \begin{matrix} A \\ D \end{matrix}$

$AD \xrightarrow{a} BC$

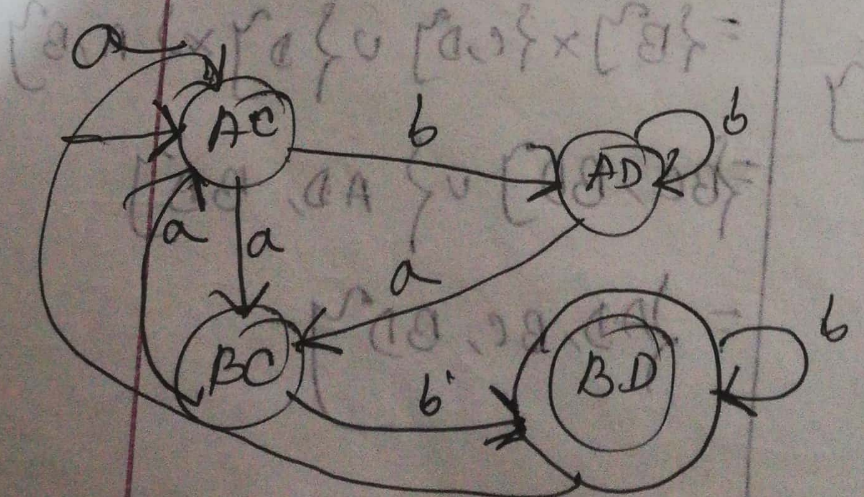
$AD \xrightarrow{b} AD$

Intersection

$F = (F_1 \times Q_2) \cap (F_2 \times Q_1)$

$= F_1 \times F_2$

$L = \{ w \mid w$ has an odd number of a's and ends with a 'b' $\}$



$F = F_1 \times F_2$

$= \{ BD \}$

Q, Σ, δ, q_0

union এর মতো।

NFA

• DFA থেকে transition different হয়।

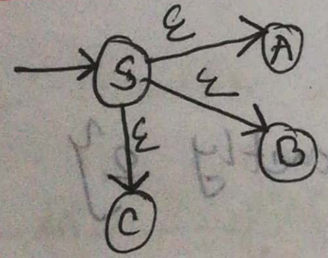
ε → special way এ machine run করবো

• One state to another state: 0 to many transition

* Every DFA is NFA

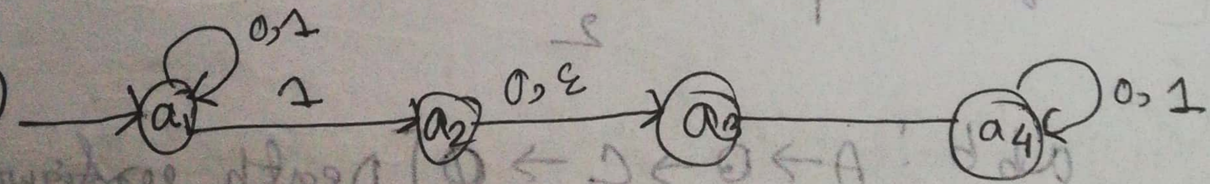
ε → কিছু read না করেও transition করা যায়।

Use of ε

1) 

machine কোন কিছু read করার আগেই outgoing edge কে active করবো, machine branching হয়ে যায়।

[Parallel computing এর সুবিধা দেয়]

2) 

কোনো না কোনো state থেকে কোনো না কোনো কিছু read করার আগেই outgoing ε active হয়।

Formal definition

$\delta : Q \times \Sigma_\varepsilon \rightarrow P(Q)$

$\Sigma_\varepsilon \rightarrow \Sigma \cup \varepsilon$

All possible subset

ε: input signal
না, transition হ
speciality

$$\boxed{\begin{array}{l} \text{DFA}: Q \times \Sigma \rightarrow Q \\ \text{NFA}: Q \times \Sigma \rightarrow P(Q) \end{array}}$$

Design a NFA, $\Sigma = \{a, b\}$

$L = \{w :$ the length of string exactly $2\}$
$= \{aa, ab, ba, bb\}$

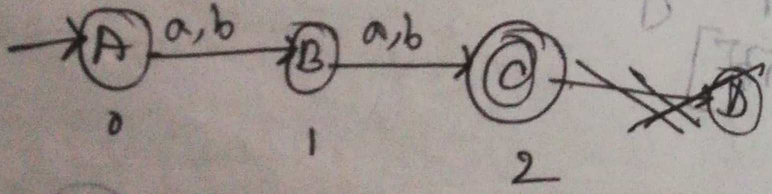
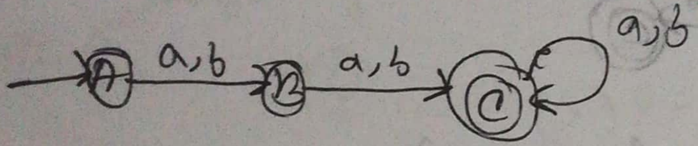


abb : $A \rightarrow B \rightarrow C \rightarrow \phi$ [Death configurati

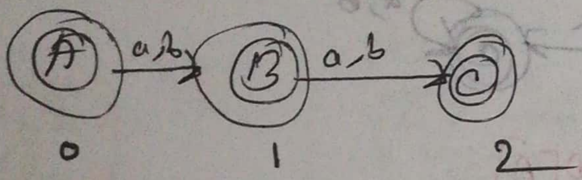
[In DFA: Trap state]

$\Sigma = \{a, b\}$

. at least 2



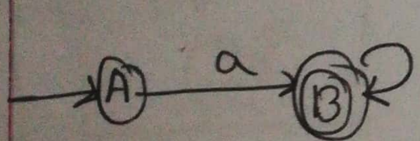
at most 2



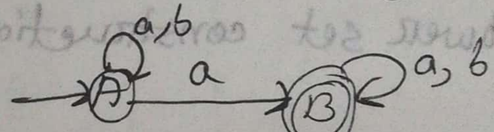
$L = \{\varepsilon, a, b, ab, ba, bb, aa\}$

$\Sigma = \{a, b\}$

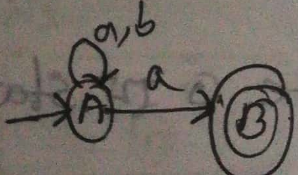
L = {starts with a}



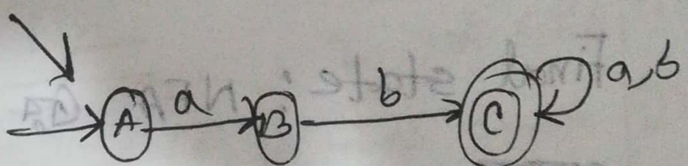
L = {containing a}



L = {ends with a}

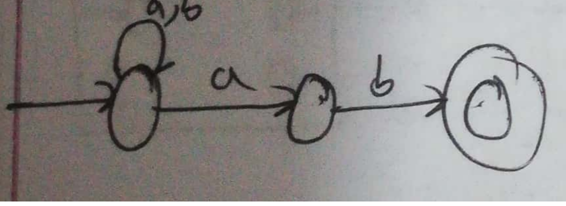


L = {starts with ab}



[containing ab]

L = {ends with ab}



NFA → DFA (Slide)

Power set construction theory

E = Epsilon closer

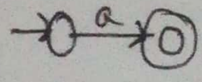
E({b1}) = Epsilon closer of b1

Final state: NFA এর final state এ যে state থাকবে, তার common হবে।

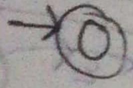
CamScanner

Regular Expression

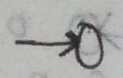
1) R = a ; where a ∈ Σ



2) R = ε → empty string

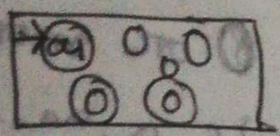


(3) R = φ → empty

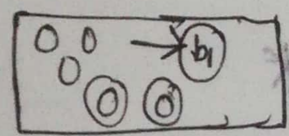


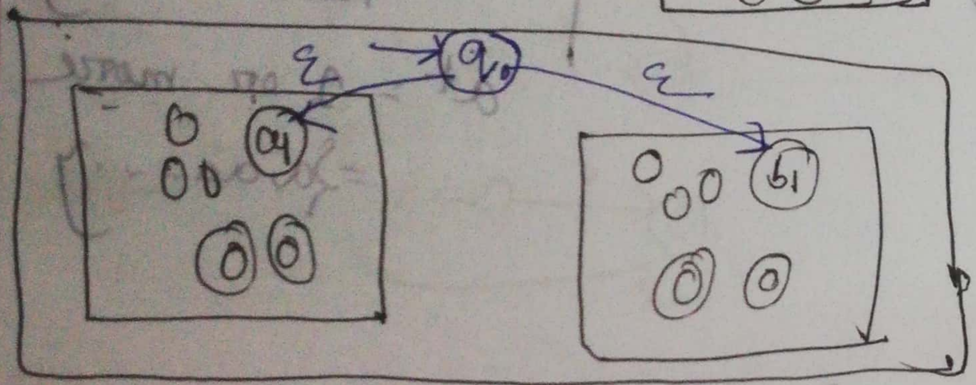
4) $R_1 \cup R_2 = R_1 | R_2$

N1



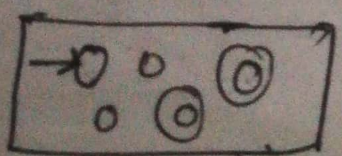
N2



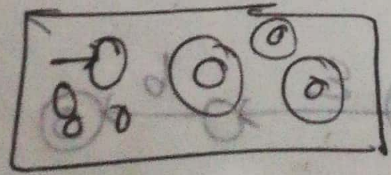


5) $R = R_1 \circ R_2 = R_1 R_2$

N1

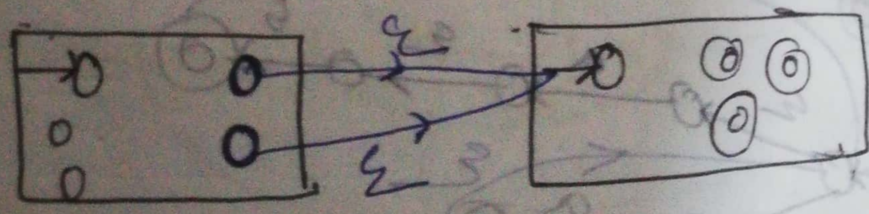


N2



1. 1st NFA এর Final state গুলো non-final হয়ে যাবে
2. Non-final state গুলোকে ε দিয়ে connect করে দিতে হবে।
3. 2nd NFA as it is

$N = N_1 N_2$

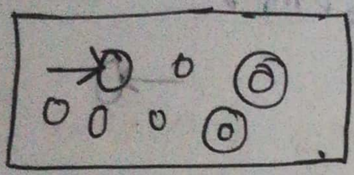


operator precedence

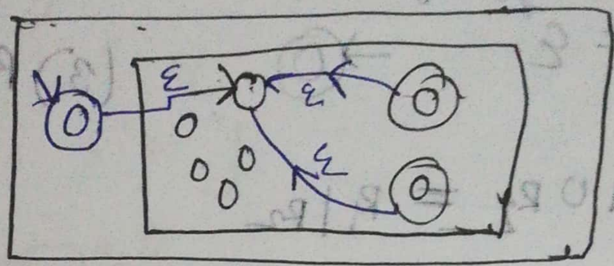
$* > \cdot > \cup$

6) $R = R_1^*$

$R_1 \to N_1$



R_1^*



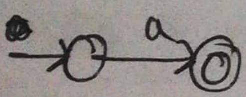
RE to NFA

$(ab \cup a)^*$

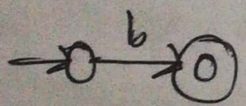
$a^* = 0$ or more $= \{\varepsilon, a, aa, \ldots\}$

$a^+ = 1$ or more $= \{a, aa, \ldots\}$

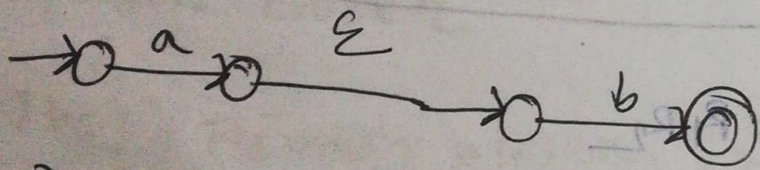
• a



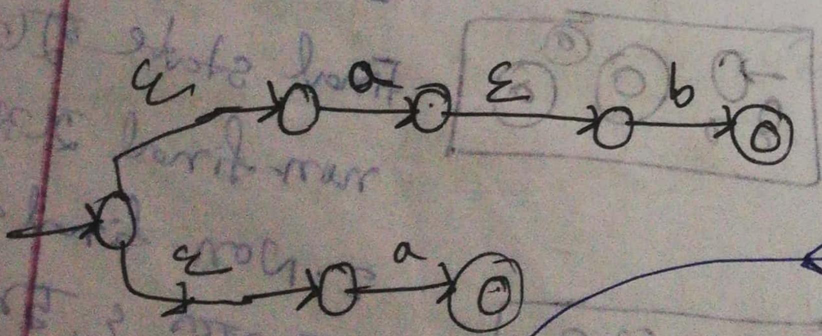
• b



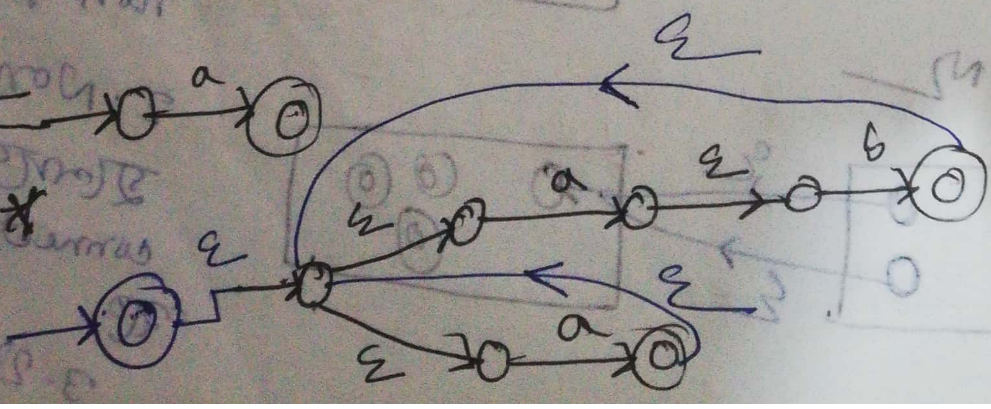
• ab



• $(ab \cup a)$

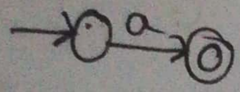


$(ab \cup a)^*$

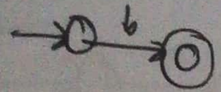


$(a \cup b)^* aba$

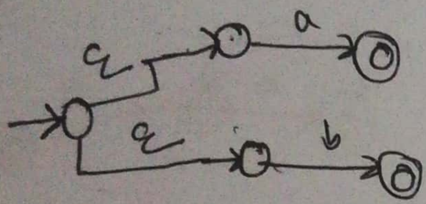
a



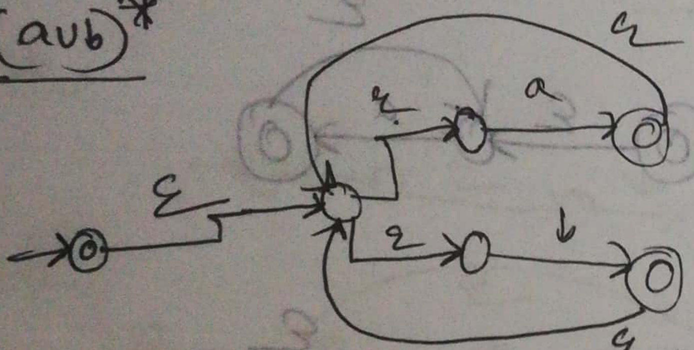
b



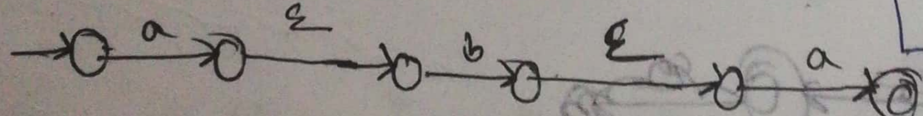
$a \cup b$



$(a \cup b)^*$

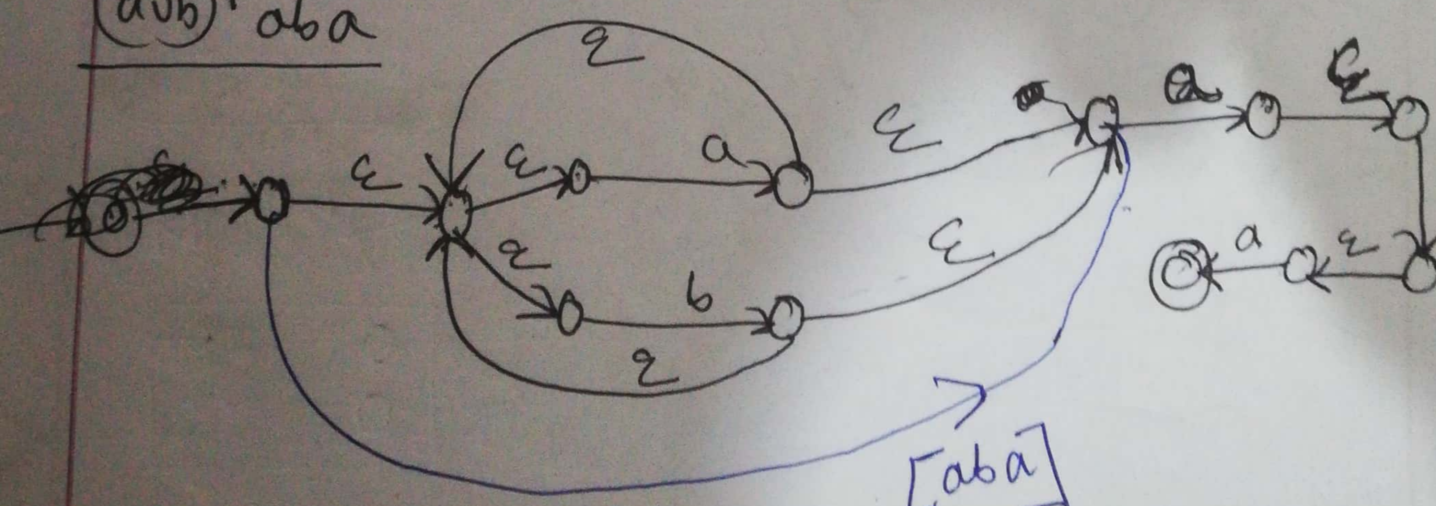


aba



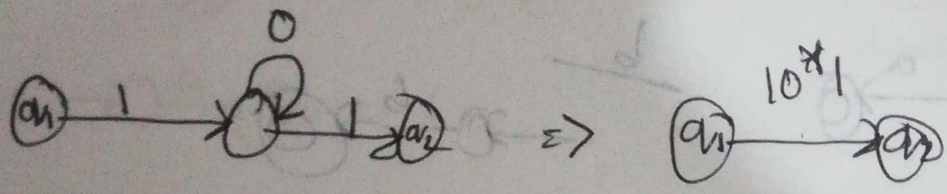
concat... এর সময় শেষের NFA as it is থাকবে, বাকি গুলোর final state non final হয়ে যাবে।

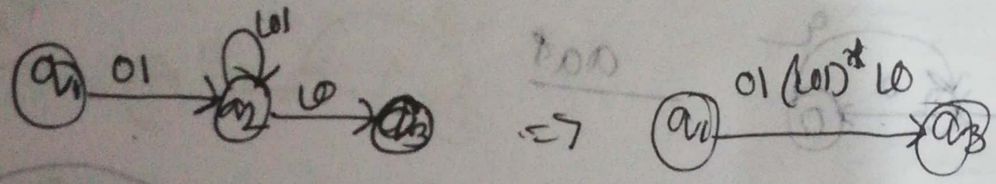
$(a \cup b)^* aba$

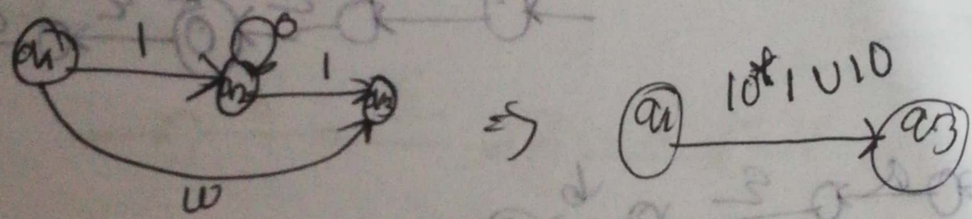


DFA to GNFA to RE

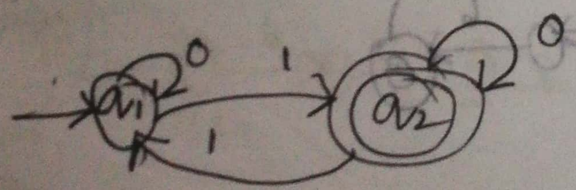
GNFA example



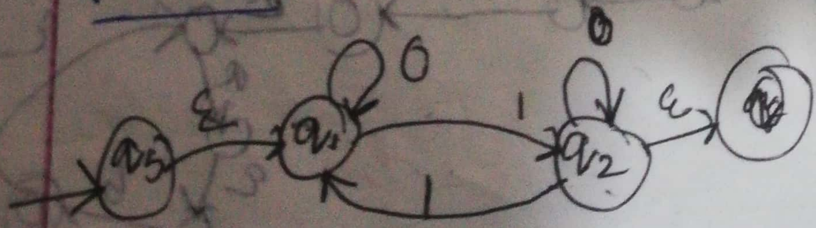




DFA to Regular Expression

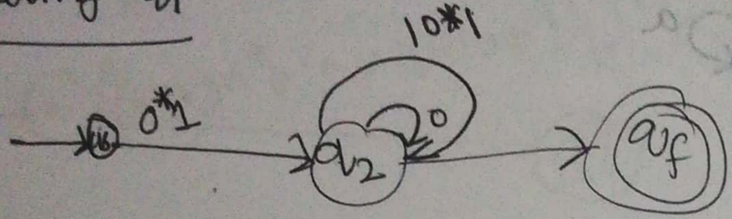


GNFA

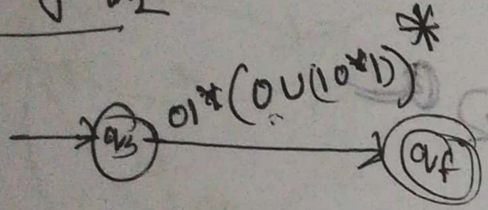


CamScanner

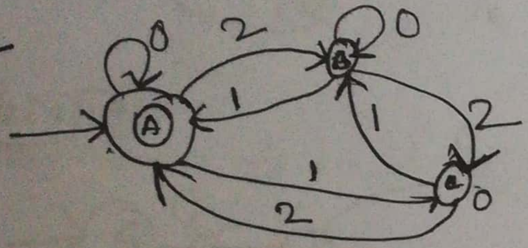
Removing q_1



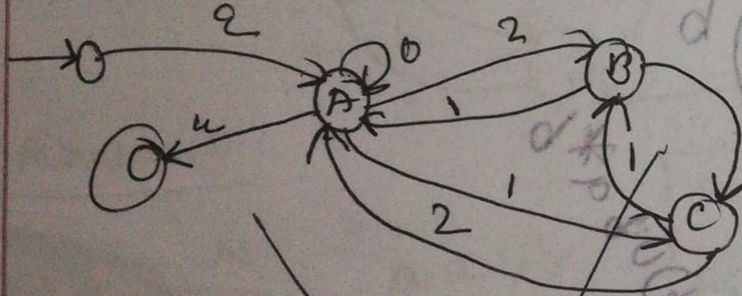
Removing q_2



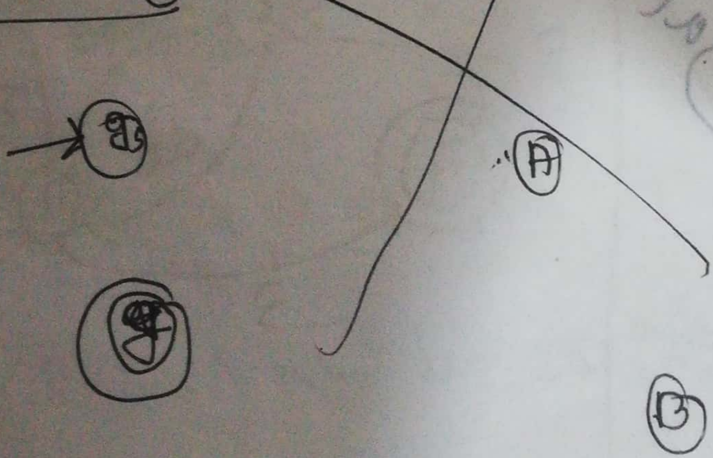
\#



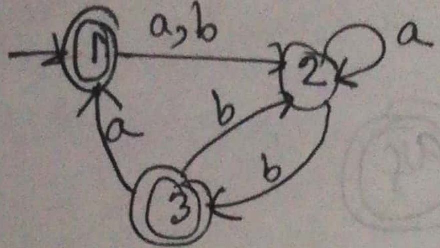
GNFA



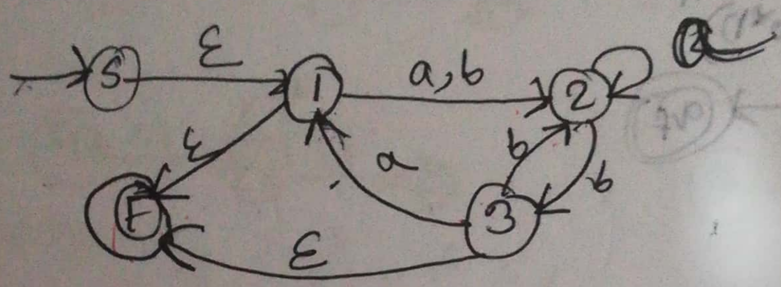
\#Remove C



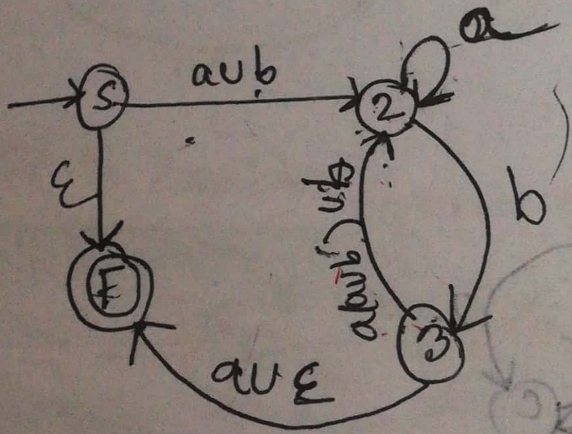
#



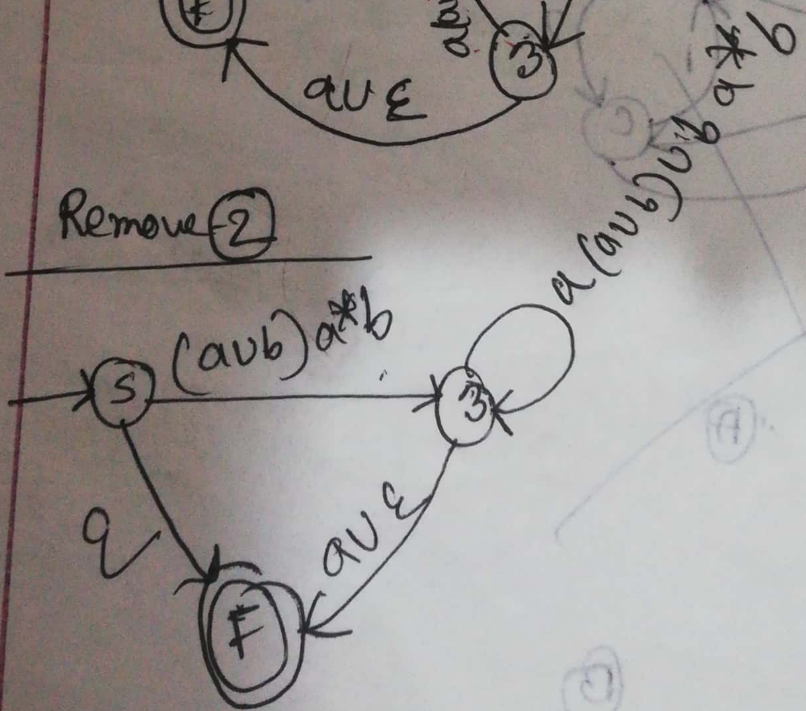
GNFA



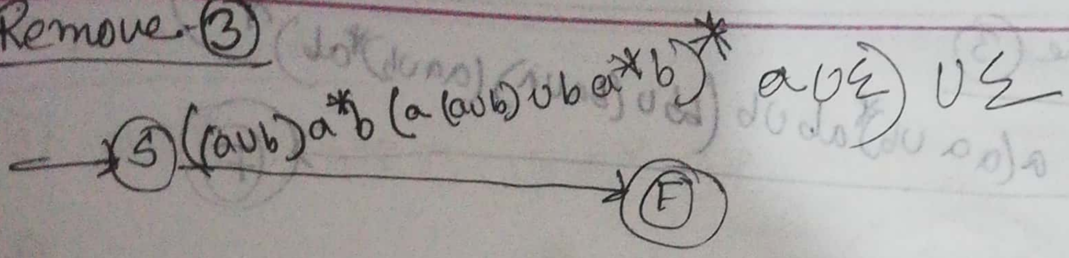
Remove 1

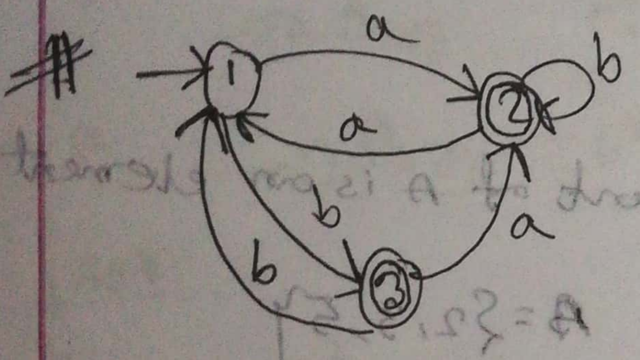


Remove 2

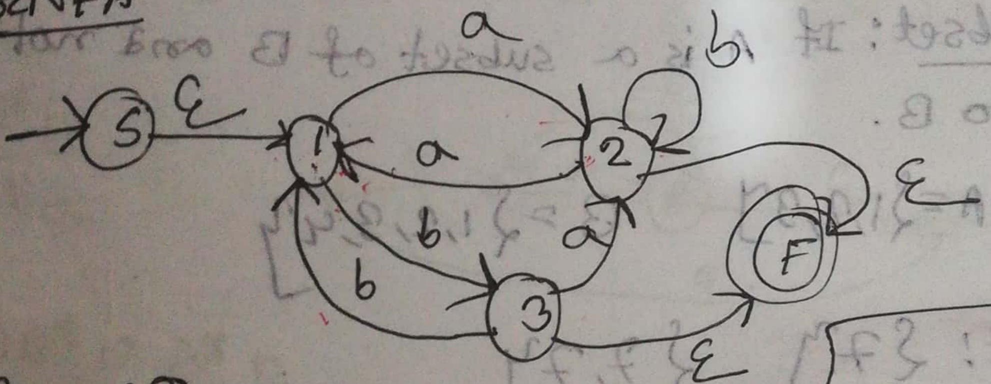


Remove ③

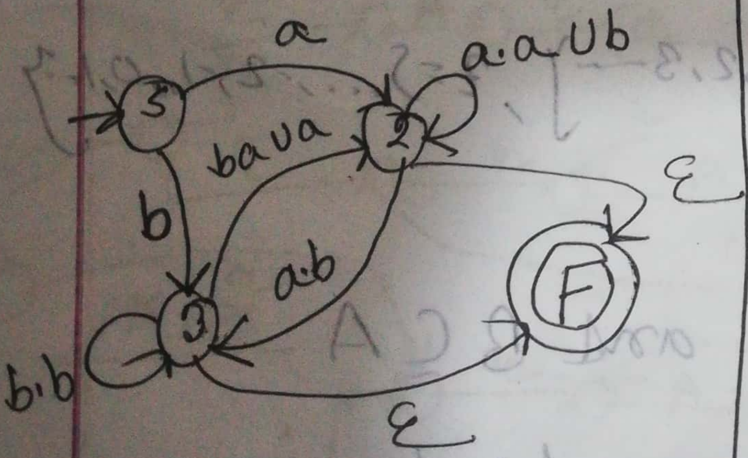




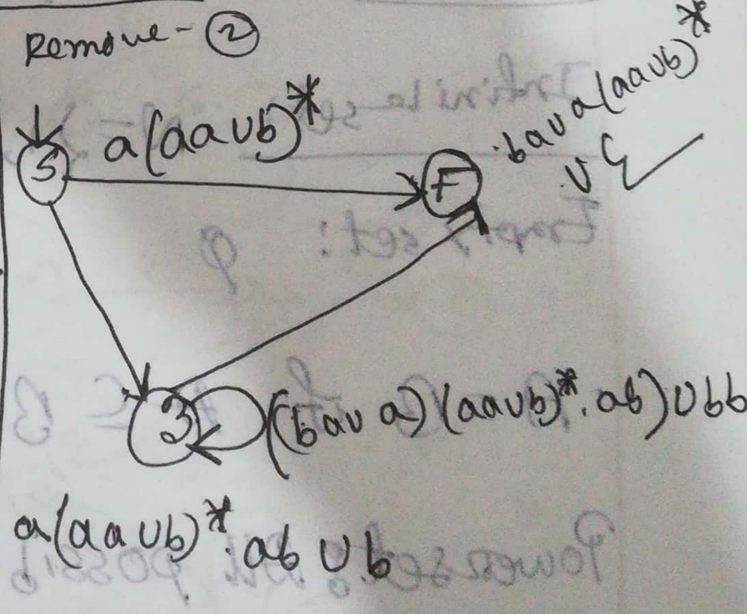
GNFA



Remove ①



Remove - ②



$((ba \cup a(aa \cup b)^* \cup \varepsilon) \cup (a(aa \cup b)^*)$

Remove 3

$\rightarrow (5) \xrightarrow{a(aa \cup b)^* ab \cup b(bb \cup (ba \cup a)(aa \cup b)^* ab)^*} (F)$

Set

subset: $A \subseteq B$, Every element of A is an element of B. $B = \{1,2,3,4,5\}$ $A = \{2,3,5\}$

Proper subset: If A is a subset of B and not equal to B.

$A = \{1,2,3\}$ $B = \{1,2,3,4\}$

multiset: $\{7\}$ $\{7,7\}$

Infinite set: $N = \{1,2,3 \ldots\}$, $Z = \{\ldots, -2, -1, 0, 1$

Empty set: ϕ

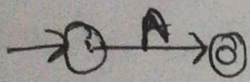
$A = B$ if $A \subseteq B$ and $B \subseteq A$

Powerset: All possible subset including ϕ

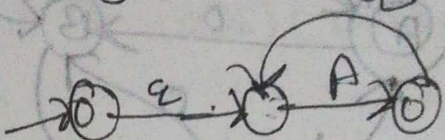
RE to NFA

① $AB^* \cup A^*B \cup A^*B^*$

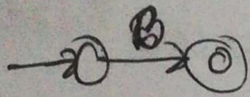
A



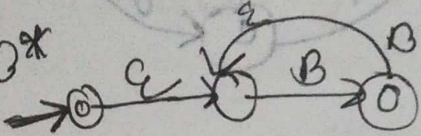
A^*



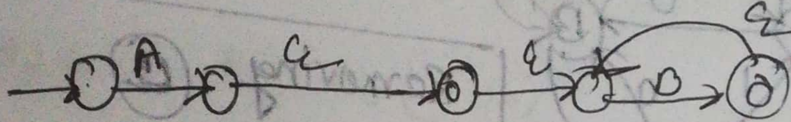
B



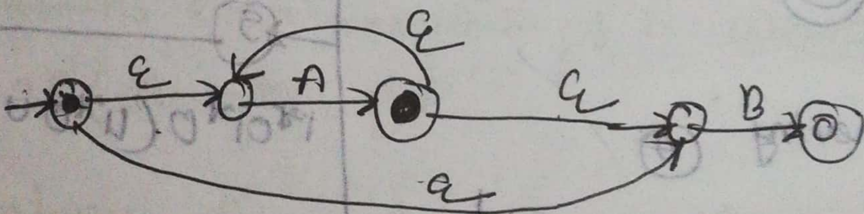
B^*



AB^*

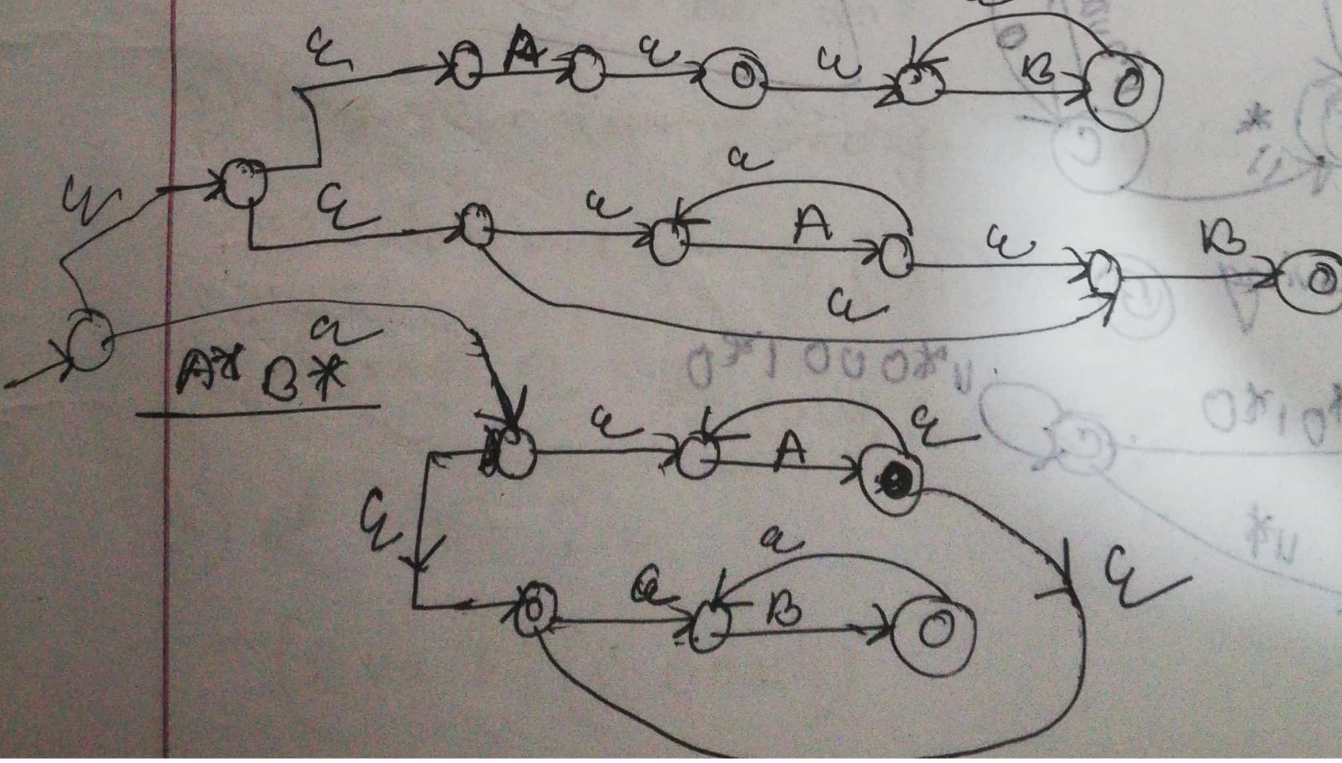


A^*B

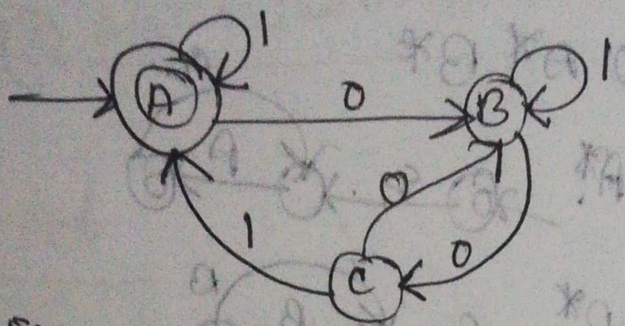


$AB^* \cup A^*B$

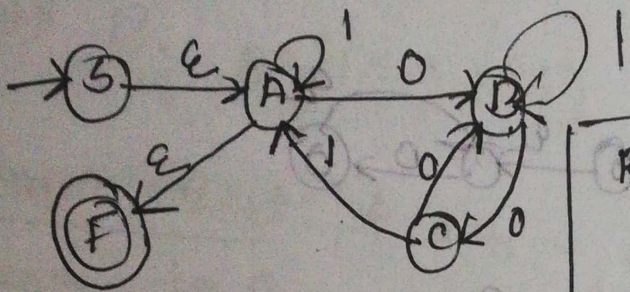
A^*B^*



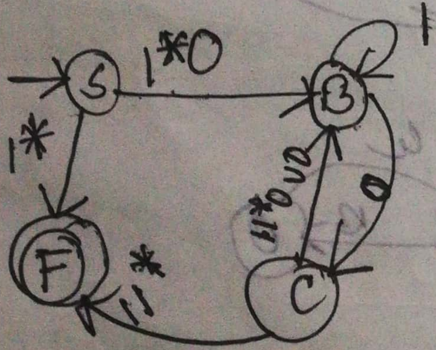
DFA to RE



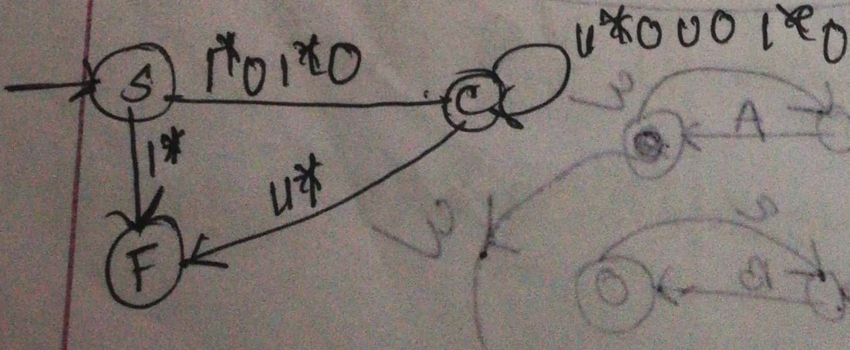
GNFA



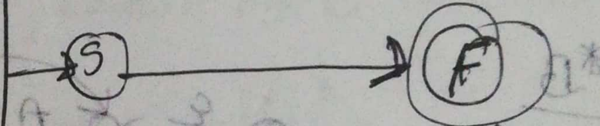
Removing A



Removing B



Removing C



1*01*0(11*0 ∪ 01*0)*.11*

∪ 1*

a ∪ b ∪ aba, aaa, bbb ...

, , , aba, bbb

Design RE for the following language $\Sigma = \{a, b\}$

1) $\{w \mid w$ has odd length$\}$ ⇒ $(a \cup b)((a \cup b)(a \cup b))^*$

2) $\{w \mid$ each 'a' in w is followed by at least two 'b'$\}$

$= b^* abbb^* (abbb^*)^*$

iii) $\{w \mid w$ starts with 'a' and contains at least two b$\}$

~~aa*bb~~ $aa^* ba^* b (a \cup b)^*$

iv) $\{w \mid w$ contains even number of 'b' where $n > 0\}$

$a^* b a^* b a^* (a^* b a^* b a^*)^*$

v) $\{w \mid w$ contains at most two 'a'$\}$

$b^* \cup b^* a b^* \cup b^* a b^* a b^*$

vi) Language accepting strings of length exactly 2.

$aa \cup ab \cup ba \cup bb$ ⇒ $(a \cup b)(a \cup b)$

2) length at least 2 ⇒ $(a \cup b)(a \cup b)(a \cup b)^*$

3) at most 2.

$= (\varepsilon \cup a \cup b)(\varepsilon \cup a \cup b)$

Give the description of ~~the~~ following ~~RE~~

(i) $(a \cup b)^* ababa$ = { w | w ends with ababa substring }

(ii) $ab^* ab^* ab^* ab^*$ = { w | w starts with 'a' and contains four 'a' }

(iii) $(b^* ab^* ab^* ab^*)^*$ = { w | num of 'a' is multiple of three 'a' }

(iv) $a^* \cup a^* b a^* \cup a^* b a^* b a^*$

{ w | w contains at most two 'b' }

FINAL TERM

Context Free Grammar

CFG 4 touples (v, T, P, S)

N / V = variable / non terminal. capital letter

Σ / T = set of terminal [terminal canot be replaced by anything] terminal / consta

R / P = production / rules

S = start variable

→ can be replaced by operator

(→) এর বাম পাশে কখনো terminal থাকে না।

1 (OK) starting variable কিছু উল্লেখ না থাকলে left most

CFG for,

$\{0^n 1^n \mid n \geq 1\}$ $S \to 01 \mid 0S1$

$0^1 1^1 = \{01\}$ $S \to 01$

$0^2 1^2 = \{0011\}$ $S \to 0S1$

$V = \{S\}$

$T = \{0, 1\}$

$S = S$

0^n1^n

$S \to 01 \mid 0S1$

Derivation

$\boxed{0011}$ string

S

0S1

0011

000111

$S \to 0S1$

$S \to 00S11$

$S \to 000111$

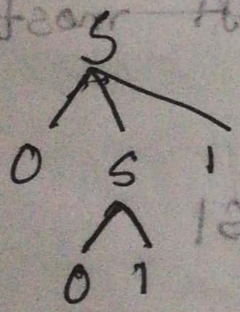
3 Question

- Design
- Derivation
- Parse Tree

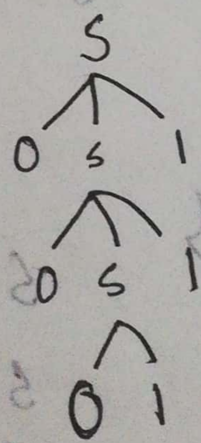
Parse tree

$S \to 01 \mid 0S1$

0011



000111

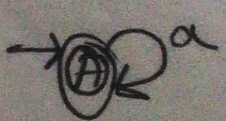


Design CFG

$L = \{a^n \mid n \geq 0\}$ [a^*]

$= \{\varepsilon, a, aa, \ldots\}$

$A \to aA / \varepsilon$



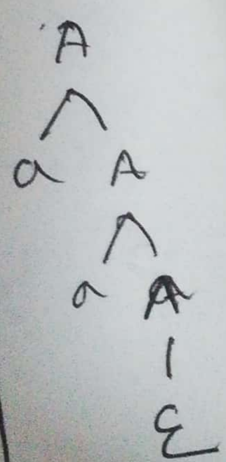
aa

$A \to aA$

aaA

$aa\varepsilon$

aa



$L = \{a^n \mid n \geq 1\}$

$= \{a, aa, \ldots\}$ [at]

$A \to aA \mid a$

$L = \{$set of all strings over $a, b\}$

$(a \cup b)^*$

$A \to aA \mid bA \mid \varepsilon$

set of all strings which length at least 2.

$L = \{aa, ab, ba, bb, aab \ldots\}$

$\underbrace{(a \cup b)(a \cup b)}_{aa/ab/ba/bb}$ [RE]

$\underset{A}{(a \cup b)}\ \underset{A}{(a \cup b)}\ \underset{B}{(a \cup b)^*}$

~~$A \to a + b$~~

~~$B \to aB \mid bB \mid \varepsilon$~~

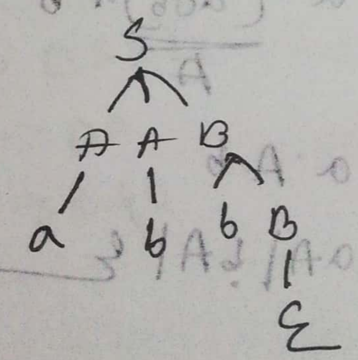
$S \to AAB$

$A \to a \mid b$

$B \to aB \mid bB \mid \varepsilon$

~~Derive~~ Parse tree

abb



set of all strings of at least 3 0's

~~$?0?0?0?$~~

$\underset{A}{(0 \cup 1)^*}\, 0\, \underset{A}{(0 \cup 1)^*}\, 0\, \underset{A}{(0 \cup 1)^*}\, 0\, \underset{A}{(0 \cup 1)^*}$

$S \to A0A0A0A$

$A \to 0A \mid 1A \mid \varepsilon$

1000

$S \to A0A0A0A$

$1A0A0A0A$

$10A0A0A0A$

$100A0A$

100 0A

1000

set of all strings which length at most 2.

$L = \{\varepsilon, a, b, aa, ab, ba, bb\}$

RE → $\underbrace{(a \cup b \cup \varepsilon)}_{A} \; \underbrace{(a \cup b \cup \varepsilon)}_{A}$

S → AA

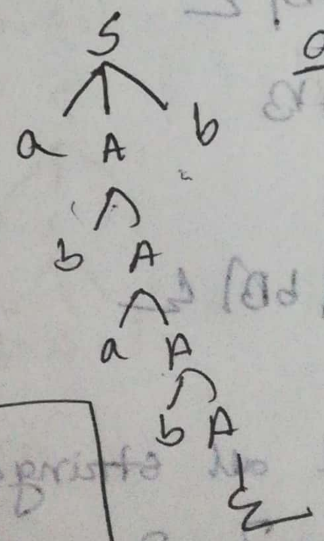
A → a | b | ε

starts with 'a' and ends with b

$a \, \underbrace{(a \cup b)^*}_{A} \, b$

S → aAb

A → aA | bA | ε



abab

starts and end with diff. symbol.

$a \, \underbrace{(a \cup b)^*}_{A} \, b \;\; | \;\; b \, \underbrace{(a \cup b)^*}_{A} \, a$

S → aAb | bAa

A → aA | bA | ε

CS CamScanner

\# $L = \{a^n b^m c^n \mid n, m \geq 1\}$

$S \to aSc \mid aAc$

$A \to bA \mid b$

\# $L = \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$

$S \to aSd \mid aAd$

$A \to bAc \mid bc$

** ## Ambiguity

Q. what is ambiguity grammar? Proof that the ambiguity.

⇒ ~~যদি~~ কোনো string more than one way to derive ~~করা~~ যায় তাকে ambiguity grammar বলে।

Chomsky Normal Form

reduce ambiguity.

- every variable either 2টা variable দ্বারা replace হবে or 1টা terminal দ্বারা replace হবে।
- ε not allowed in grammar. থাকলেও starting variable এর সাথে থাকবে।

$A \to BC$

$A \to a$

conversion

1. add new start

2. ε rule remove

3. Remove unit rule [একটা variable exactly 1টা variable দ্বারা replace হচ্ছে।]

4. Take aditional rule. যারা rule (3) থেকে chomsky এর চেহারা পায় না তাদের জন্য প্রযোজ্য হবে।

i) এমন কোনো variable নেয়া যাবে না, যেটা already left side এ exist করছে। new variable নিতে হবে।

ii) যে rule নেয়া হবে, সেটা যেনো chomsky এর form এ থাকে।

iii) যেগুলো already chomsky form এ আছে সেগুলোর জন্য rule নেয়া যাবে না।

Finally rule গুলোকে grammar এ add করতে হবে।

#

$S \to ASA \mid aB$

$A \to B \mid S$

$B \to b \mid \varepsilon$

Adding new start variable S_0

$S_0 \to S$

$S \to ASA \mid aB$

$A \to B \mid S$

$B \to b \mid \varepsilon$

Remove $B \to \varepsilon$

$S_0 \to S$

$S \to ASA \mid aB \mid a$

$A \to B \mid S \mid \varepsilon$

$B \to b$

Remove $A \to \varepsilon$

$S_0 \to S$

$S \to ASA \mid aB \mid a \mid SA \mid AS \mid S$

$A \to B \mid S$

$B \to b$

Remove $S \to S$

$S_0 \to S$

$S \to ASA \mid aB \mid a \mid SA \mid AS$

$A \to B \mid S$

$B \to b$

Remove $S_0 \to S$

$S_0 \to ASA \mid aB \mid a \mid SA \mid AS$

$S \to ASA \mid aB \mid a \mid SA \mid AS$

$A \to B \mid S$

$B \to b$

Remove $A \to B$

$S_0 \to ASA \mid aB \mid a \mid SA \mid AS$

$S \to ASA \mid aB \mid a \mid SA \mid AS$

$A \to b \mid S$

$B \to b$

Remove $A \to S$

$S_0 \to ASA \mid aB \mid a \mid SA \mid AS$

$S \to$ "

$A \to b \mid ASA \mid aB \mid a \mid SA \mid AS$

$B \to b$

Add rule, $C \to SA$

$S_0 \to AC \mid aB \mid a \mid SA \mid AS$

$S \to AC \mid aB \mid a \mid SA \mid AS$

$A \to b \mid AC \mid aB \mid a \mid SA \mid AS$

$B \to b$

Add rule $C \to SA$

Add rule, $D \to a$

$S_0 \to AC \mid DB \mid a \mid SA \mid AS$

$S \to AC \mid DB \mid a \mid SA \mid AS$

$A \to b \mid AC \mid DB \mid a \mid SA \mid AS$

$B \to b$

$C \to SA$

$D \to a$

$\varepsilon \to$ no input is read $\qquad$ ϕ: end of input

Push Down Automata (PDA)

A push down automata is a way to implement a CFG in a similar way we design DFA for RG.

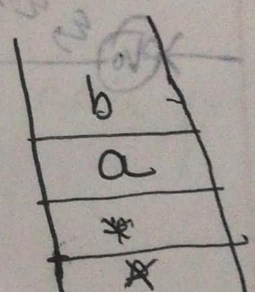
6 Touple $(Q, \Sigma, \Gamma, \delta, q_0, F)$

Γ = Top of the stack pointer [Stack এর কোথায় push, pop করবে]

$\delta : Q \times \Sigma_{\varepsilon} \times \Gamma_{\varepsilon} \to P(Q \times \Gamma_{\varepsilon})$ → new ~~top~~ top of the stack

↓ current top of the stack

	current	new top of the stack
ignore	ε	ε
pop 'a'	a	ε
push 'b'	ε	b
conditional push	a	b

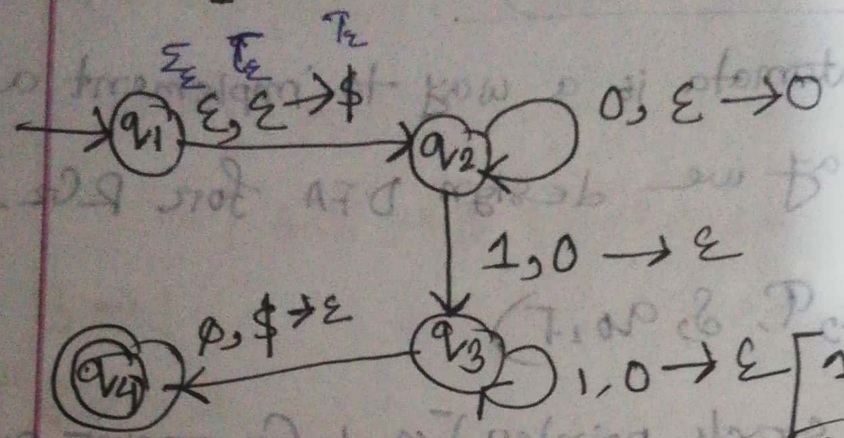


Design

For every PDA, push $ in the first, pop on the last.

class

$0^n 1^n$



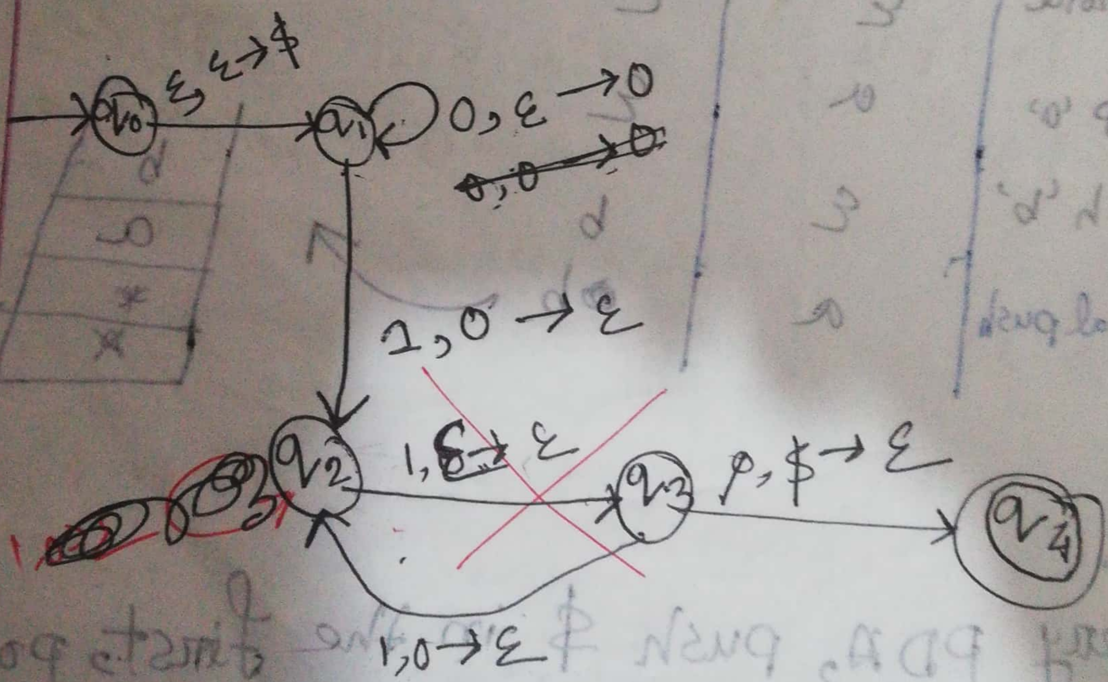
ϕ ; read করার পরেও string থাকলে

[1 read করা হলো, current top of the stack এ 0 ছিলো, তাই pop ~~করা হয়েছে~~

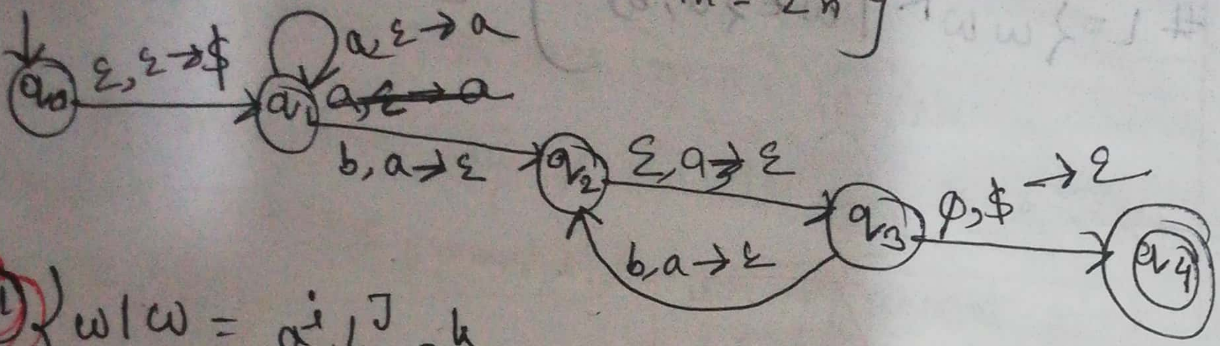
$0^n 1^{2n}$, $n>0$

$= 011, 001111$

0	0	1	1	1	1	ε

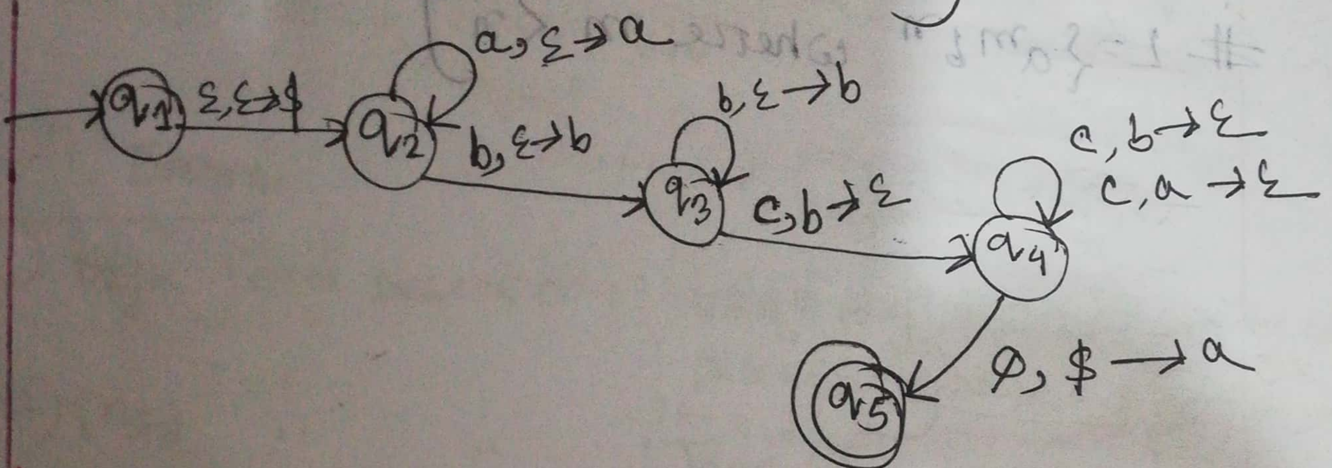


class # $\{w \mid w = a^m b^n$, where $m = 2n\}$



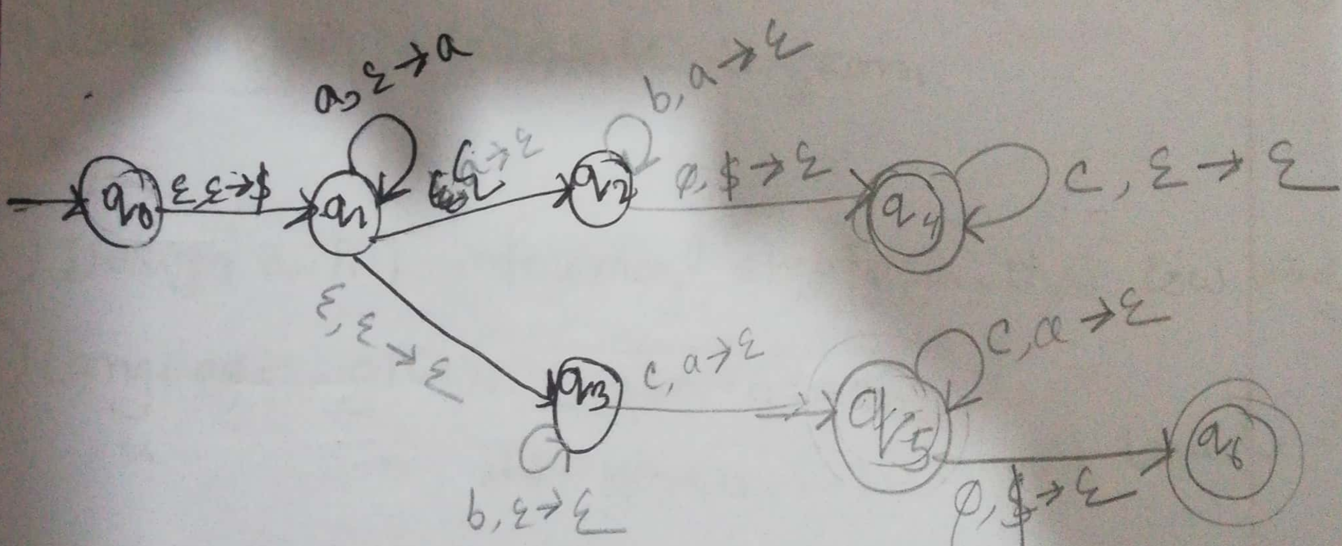
class 11 $\{w \mid w = a^i b^j c^k$, where $i + j = k\}$

P



~~#~~ $\{w \mid w = a^i b^j c^k \mid i, j, k \geq 0$ and $i = j$ or $i = k\}$

$a = b$
$a = c$



Computeability theory

কোনো একটা কাজ করা যাবে কিনা? Yes/No

Turing machine always result generate করবে, positive/ Negative. Turing machine accept / reject state এ গেলে থামবে।

Turing machine: end বলে কিছু নাই।

halting state < Accept state
Reject "

TM Design

i) High level Description : কিভাবে sol^n করা যায় step by step process লিখা। (Algorithm)

ii) Mid " " : Tape কিভাবে utilize করা যায় তা লিখা। কখন left এ যাবে, কখন right এ যাবে, কখন accept হবে, কখন reject হবে।

iii) Low " " : State diagram

Q type

a) Design a TM informal step [High, Mid, Low level]

b) Language given. Write algo

" " , algo given. write mid level...

Nadim Bin Hossain

1. Consider the following Turing machine. In the transition Function, q0 is the Start state, q_{accept} is the Accept State and q_{reject} is the Reject State. In the transition function '#' represents Blank Space. Now simulate this machine with '00000000' string and find whether it is accepted or rejected by the machine.

State	Input	Transition
q0	0	q1, #, R
q0	x	q_{reject}, x, R
q0	#	q_r, #, R
q1	0	q3, x, R
q1	x	q1, x, R
q1	#	q_{accept}, #, R
q2	0	q2, 0, L
q2	x	q2, x, L
q2	#	q1, #, R
q3	0	q4, 0, R
q3	x	q3, x, R
q3	#	q2, #, L
q4	0	q3, x, R
q4	x	q4, x, R
q4	#	q_{reject}, #, R

q_0 0 0 0 0 0 0 0 0 #
q_1 0 0 0 0 0 0 0
x q_3 0 0 0 0 0 0
x 0 q_4 0 0 0 0 0
x 0 x q_3 0 0 0 0 #
x 0 x 0 q_4 0 0 0 #
x 0 x 0 x q_3 0 0 #
x 0 x 0 x 0 q_4 0 #
x 0 x 0 x 0 x q_3 #
x 0 x 0 x 0 q_2 x #
x 0 x 0 x q_2 0 x #
x 0 x 0 q_2 x 0 x #
x 0 x q_2 0 x 0 x #
x 0 q_2 x 0 x 0 x #
x q_2 0 x 0 x 0 x #
q_2 x 0 x 0 x 0 x #
x 0 x 0 x 0 x
q_1 x 0 x 0 x 0 x #
x q_1 0 x 0 x 0 x #
x x q_3 x 0 x 0 x #
x x x q_3 0 x 0 x #
x x x 0 q_4 x 0 x #
x x x 0 x q_4 0 x #
x x x 0 x x q_3 x #
x x x 0 x x x q_3 #
x x x 0 x x q_2 x #
x x x 0 x q_2 x x #
x x x 0 q_2 x x x #

x x x q_2 0 x x x
x x q_2 x 0 x x x
x q_2 x x 0 x x x
q_2 x x x 0 x x x
q_2 # x x x 0 x x x #
q_1 x x x 0 x x x
x q_1 x x 0 x x x
x x q_1 x 0 x x x
x x x q_1 0 x x x
x x x x q_3 x x x
x x x x x q_3 x x
x x x x x x q_3 x
x x x x x x x q_3
x x x x x x q_2 x
x x x x x q_2 x x
x x x x q_2 x x x
x x x q_2 x x x x
x x q_2 x x x x x
x q_2 x x x x x x
q_2 x x x x x x x
q_2 # x x x x x x x #
q_1 x x x x x x x
x q_1 x x x x x x
x x q_1 x x x x x
x x x q_1 x x x x
x x x x q_1 x x x

x x x x x q_1 x x
x x x x x x q_1 x
x x x x x x x q_1
x x x x x x x # q_{acc}

2. Consider a Turing machine with the following transitions:

State	Input	δ(State, Symbol, Move)
Q_0	a	Q_1,#,R
Q_0	#	Q_{accept}, #,R
Q_1	a	Q_1, a, R
Q_1	b	Q_2,x,R
Q1	x	Q_1, x, R
Q_2	a	Q_3,x,R
Q_2	b	Q_2, b, R
Q_2	x	Q_2, x, R
Q_3	a	Q_4,a,L
Q_3	#	Q_6,#,L
Q_4	a	Q_4,a,L
Q_4	b	Q_4,b,L
Q_4	x	Q_4,x,L
Q_4	#	Q_5,#,R
Q_5	a	Q_1,x,R
Q_5	x	Q_5,x,R
Q_6	x	Q_6,x,L
Q_6	#	Q_{accept}, #,R

Here 'Q_0' is the start state, 'Q_{Accept}' is the accept state. Trace the execution of this Turing machine with the string **aabbaa#** as input. Note that '#' represents the blank symbol.

q_0 a a b b a a #
\# q_1 a b b a a #
\# a q_1 b b a a #
\# a x q_2 b a a #
\# a x b q_2 a a #
\# a x b x q_3 a #
\# a x b q_4 x a #
\# a x q_4 b x a #
\# a q_4 x b x a #
\# q_4 a x b x a #

q_4 # a x b x a #
\# q_5 a x b x a #
\# x q_1 x b x a #
\# x x q_1 b x a #
\# x x x q_2 x a #
\# x x x x q_2 a #
\# x x x x x q_3 #
\# x x x x q_6 x #
\# x x x q_6 x x #
\# x x q_6 x x x #

\# x q_6 x x x x #
\# q_6 x x x x x #
q_6 # x x x x x #
\# q_{accept} x x x x x #