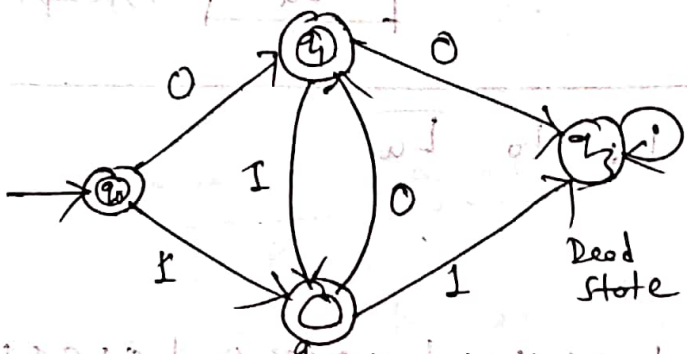
2020-21

2005001 – Anik Saha

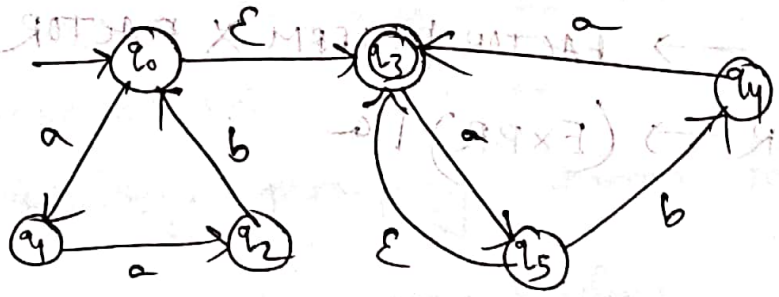
1.(a) Cryptography looks for hard problems so that encrypted information is hard to decode.

RSA uses public key $n = p \times q$ where p, q are two large primes that are private keys. Factoring is a very hard problem. And thus, getting the private key from public key is computationally impractical when p, q are large enough.

(b)



(c)



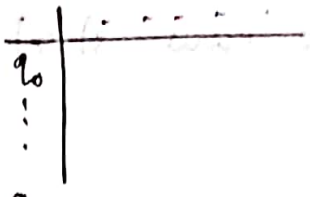
(d)

$S \longrightarrow SS \mid BAB$

$A \longrightarrow aaaA \mid \varepsilon$

$B \longrightarrow bB \mid \varepsilon$

2. (a) No, the number of rows is always $|Q|$ in δ columns always $|\Sigma|$, it does not depend on $|F|$



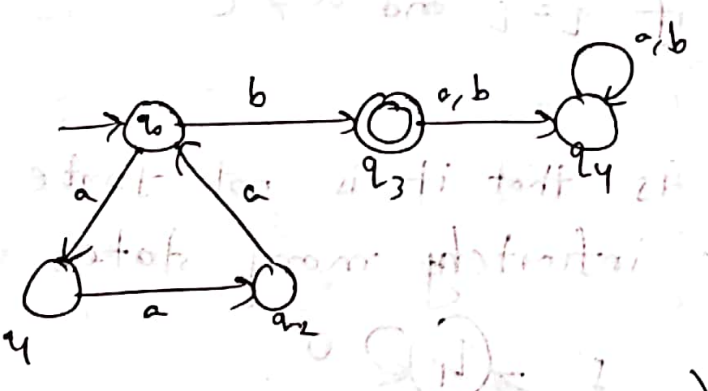
(b) : For DFA:

$\delta : Q \times \Sigma \to Q$ where $\delta(q, a) = p$, implies that the DFA goes from state q to p upon reading a

for NFA:

$\delta : Q \times \Sigma \to \mathcal{P}(Q)$ where $\delta(q, a)$ = set of states NFA can go into after reading a from state q

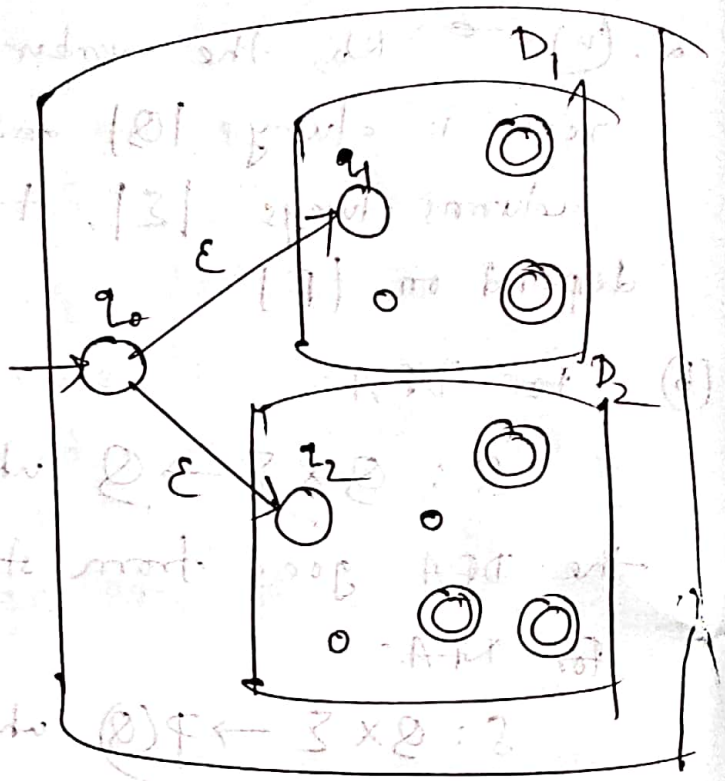
(c) 5 states needed



(d) Let $D_1 = (Q_1, \Sigma_1, \delta_1, q_1, f_1)$

$D_2 = (Q_2, \Sigma, \delta_2, q_2, f_2)$

Let the new start state be q_0



If new DFA

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

$Q = Q_1 \cup Q_2 \cup \{q_0\}$

$\Sigma = \Sigma$

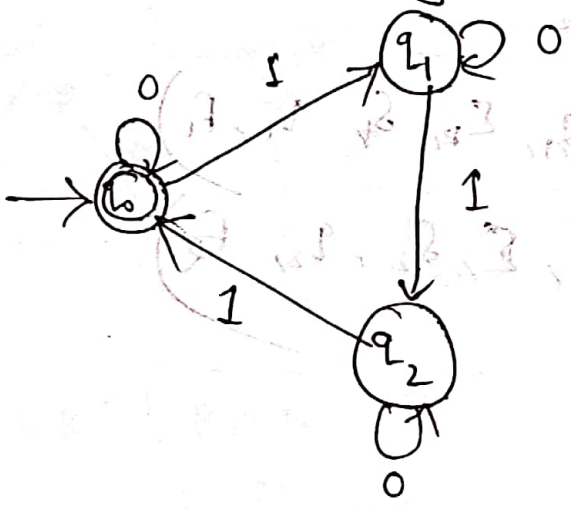
$q_0 = q_0$

$F = F_1 \cup F_2$

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & \text{if } q \in Q_1 \\ \delta_2(q, a) & \text{if } q \in Q_2 \\ \{q_1, q_2\} & \text{if } q = q_0 \text{ and } a = \varepsilon \\ \emptyset & \text{if } q = q_0 \text{ and } a \neq \varepsilon \end{cases}$$

3. (a) The flaw is that it is not finite. A DFA cannot have infinitely many states.

Design:



(b) ~~let p~~ Assume L_n is regular.

let the pumping length be p.

let $w = 0^p 1^{2p} 0^p$, $|w| = 4p \geq p$

Now, if $w = xyz$, such that $|xy| \leq p$, $|y| > 0$, then

$y = 0^k$ for $k > 0$, Now, then

$$xy^0z = 0^{p-k} 1^{2p} 0^p$$

Now, $p - k \neq p$ as $k \neq 0$

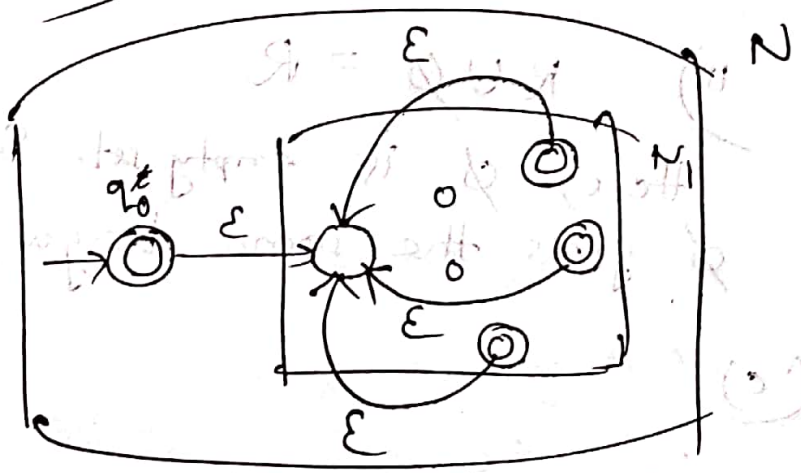
So, $xy^0z \notin L_n$

So, L_n is not regular.

(c) let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

let, $L(N) = L_1^*$,

$N = (Q, \Sigma, \delta, q_0, F)$.



Now, ~~Q =~~ if new start state is q_0'

$Q = Q_1 \cup \{q_0'\}$

$q_0 = q_0'$, $F_2 = F_1 \cup \{q_0\}$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & \text{if } q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_1\} & \text{if } q \in F_1 \text{ and } a = \varepsilon \\ \{q_1\} & \text{if } q = q_0,\ a = \varepsilon \\ \emptyset & \text{if } q = q_0 \text{ and } a \neq \varepsilon \end{cases}$$

4. (a) This string is said to be ambiguously derived.

And this grammar is ambiguous.

(b) i) $\emptyset^* = \{\varepsilon\}$

Here, if we take no element from an empty set and concatenate, we get empty set.

So, $\emptyset^* = \{\varepsilon\}$

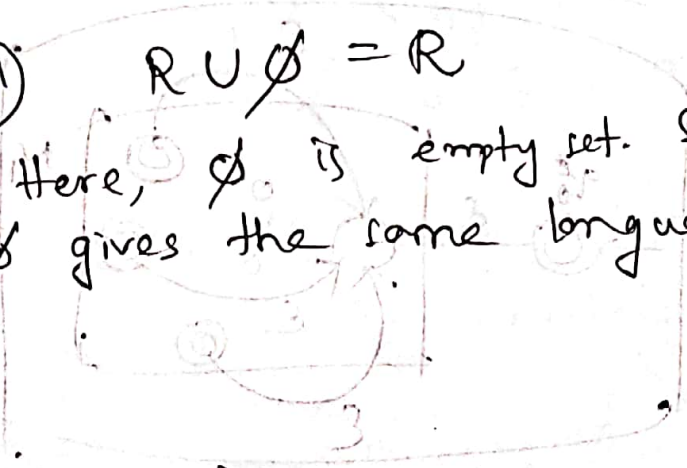
ii) $A \circ \emptyset = \emptyset$

Here, there is no string in $\emptyset$ to be concatenated with the strings present in A.

iii) $R \cup \emptyset = R$

Here, $\emptyset$ is empty set. So, taking union with $\emptyset$ gives the same language back.

(c)



Conversion:

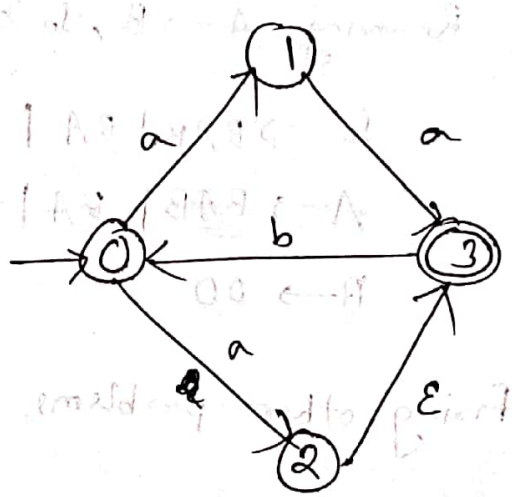
$E(0) = \{0\}$

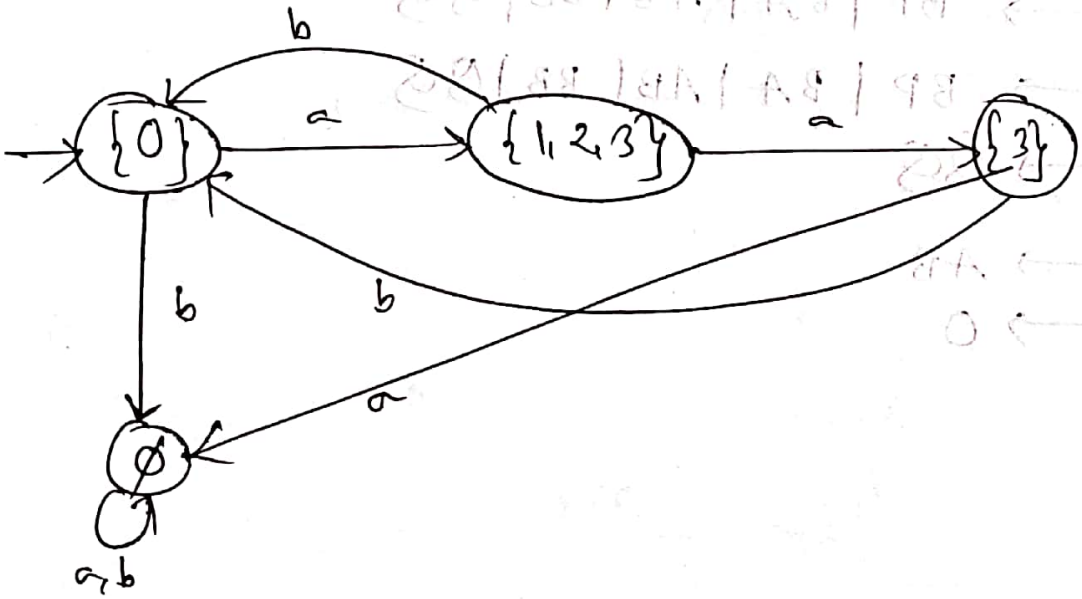
$E(1) = \{1\}$

$E(2) = \{2, 3\}$

$E(3) = \{3\}$

~~$E(4) =$~~





(d) $A \rightarrow BAB \mid B \mid \varepsilon$

$B \rightarrow 00 \mid \varepsilon$

1. A appears in RHS. Now,

$S_0 \rightarrow A$

$A \rightarrow BAB \mid B \mid \varepsilon$

$B \rightarrow 00 \mid \varepsilon$

2. Removing $B \rightarrow \varepsilon$

$S_0 \rightarrow A$

$A \rightarrow BAB \mid B \mid \varepsilon \mid BA \mid AB \mid A$

$B \rightarrow 00$

3. Removing $A \rightarrow \varepsilon$

$S_0 \rightarrow A$

$A \rightarrow BAB \mid B \mid BA \mid AB \mid \cancel{A} \mid BB$

$B \rightarrow 00$

4. Removing $A \to B$, $S_0 \to A$

$S_0 \to BAB \mid BA \mid AB \mid BB \mid 00$

$A \to BAB \mid BA \mid AB \mid BB \mid 00$

$B \to 00$

5. fixing other problems,

$S_0 \to BP \mid BA \mid AB \mid BB \mid QQ$

$A \to BP \mid BA \mid AB \mid BB \mid QQ$

$B \to QQ$

$P \to AB$

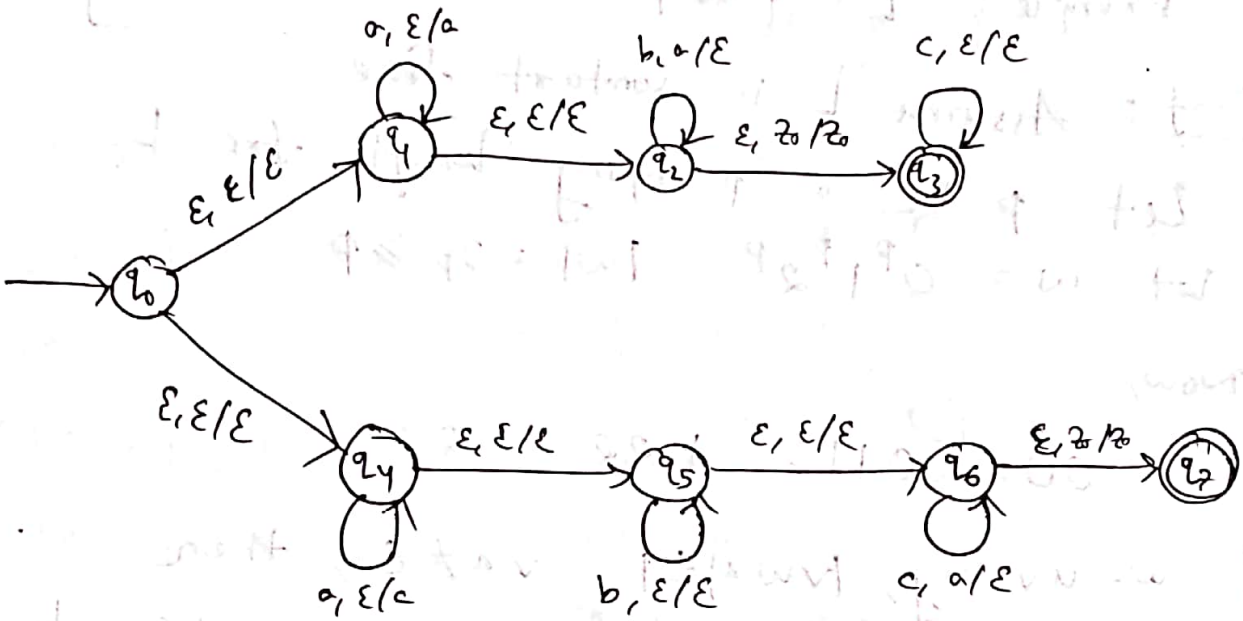
$Q \to 0$

5.

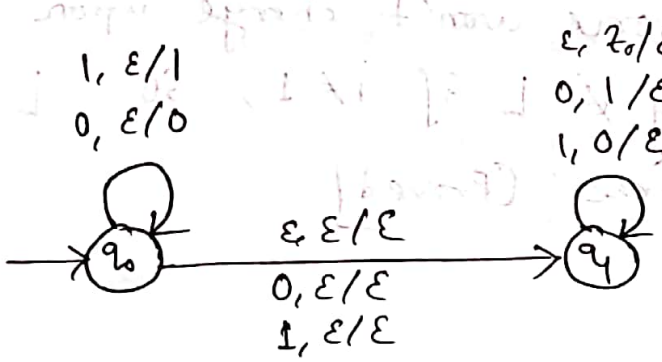
A, S			
A, S,	A, B, S		
A, S	A, B, S	A,	
A, Z	A, Y, S	A, Z	A, Z
0	1	0	0

So, yes $0100 \in L(G)$

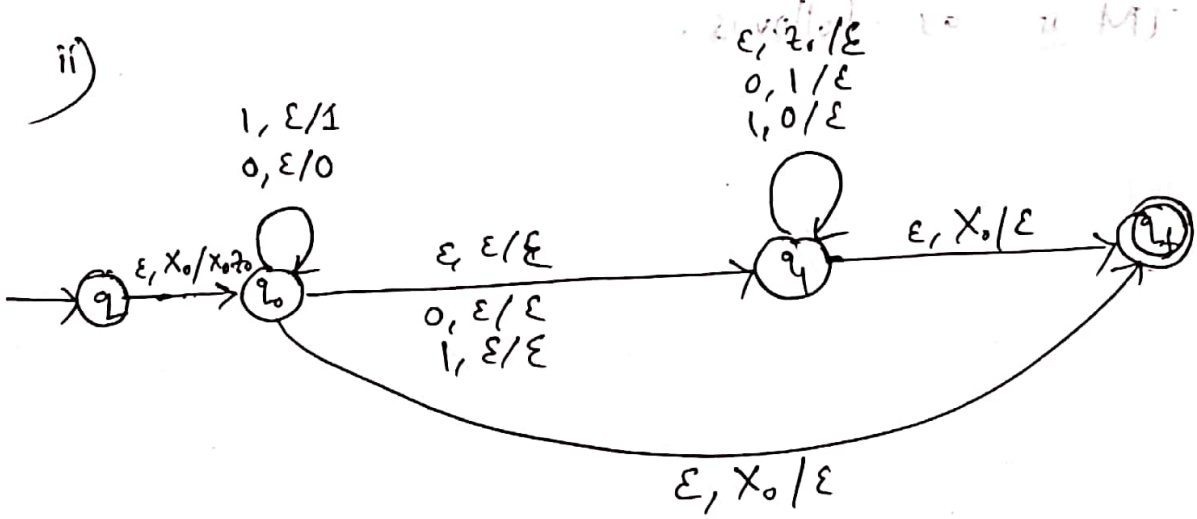
6.


a, ε/a
b, ε/b
c, ε/ε
ε, ε/ε
ε, z0/z0
ε, ε/ε
a/ε
b/ε
c/ε
a/a
b/b
c/c
ε/ε
ε, z0/z0
ε/ε
ε/ε
ε/ε
ε, z0/z0
a/ε
b/ε
c/ε

7. i)


1, ε/1
0, ε/0
ε, z0/z0
0, 1/ε
1, 0/ε
ε/ε
0, ε/0
1, ε/1
0/0
1/1
ε/z0
0/1
1/0

ii)


1, ε/1
0, ε/0
ε, X0/X0z0
ε, z0/z0
ε/ε
0, ε/0
1, ε/1
ε, X0/ε
0/0
1/1
ε/z0
0/1
1/0
ε/X0
ε, X0/ε

8. Example: $L = \{ w \mid w = 0^n 1^n 2^n, n \geq 0 \}$

Proof: Assume L is context free.

Let p be a pumping length for L.

let $w = 0^p 1^p 2^p$, $|w| = 3p \geq p$

Now,

00...011...122...2

if $w = uvwxy$, $|vwx| \leq p$, $vx \neq \varepsilon$, then

it is clear that, vwx can at most contain two of 0, 1, 2. Thus, the number of occurrences of the remaining one won't change upon pumping and it $\Rightarrow uv^i wx^i y \notin L$ if $i \neq 1$, So, L is not context free. [Proved]

9.

The TM is as follows:

Rules:

B 1 1 0 1 0 c 1 1 _ _

$\delta([q_0, B], [B, a]) = ([q_1, a], [*, a], R)$

$\delta([q_1, a], [B, 0/1]) = ([q_1, a], [B, 0/1], R)$

$\delta([q_1, a], [B, c]) = ([q_2, a], [B, c], R)$

$\delta([q_2, a], [B, 0/1]) = ([q_2, a], [B, 0/1], R)$

$\delta([q_2, a], [B, B]) = ([q_3, B], [B, a], L)$

$\delta([q_3, B], [B, 0/1/c]) = ([q_3, B], [B, 0/1/c], L)$

$\delta([q_3, B], [*, 0/1]) = ([q_0, B], [*, 0/1], R)$

$\delta([q_0, B], [B, c]) = ([q_4, B], [B, c], R)$

$\delta([q_4, B], [B, 0]) = ([q_4, B], [B, 0], R)$

$\delta([q_4, B], [B, 1]) = ([q_5, B], [B, 1], R)$

$\delta([q_5, B], [B, 0]) = ([q_5, B], [B, 1], R)$

$\delta([q_5, B], [B, 1]) = ([q_5, B], [B, 0], R)$

10. i) False.

Since an NDTM allows branching, it may need exponential amount of moves $Exp(n)$, rather than $P(n)$ to simulate it in a DTM.

ii) True

If the DTM is multi-tape, it can do so in $O(n^3)$, if it is single tape, it can do so in $O(n^6)$. Both are polynomials of n.

11. Example of ~~not RE~~ = L_d = Diagonalization Language

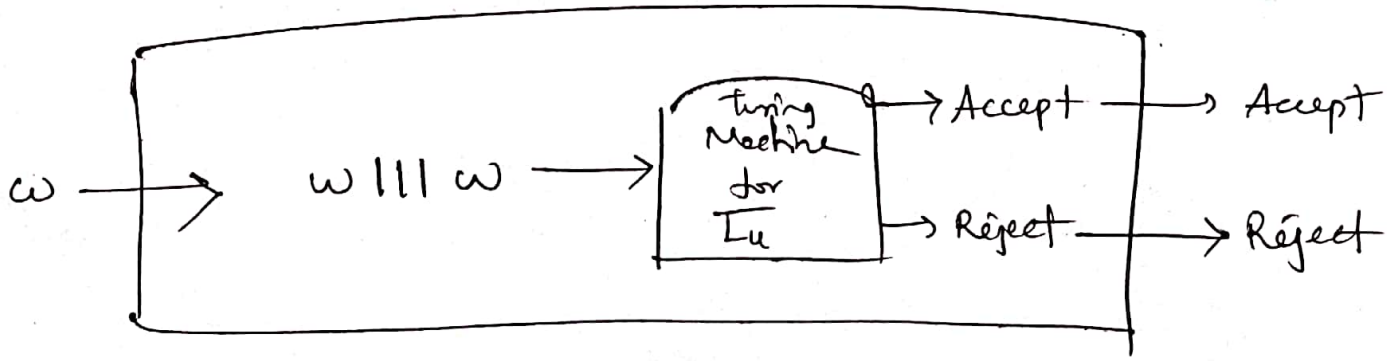
$L_d = \{ w_i \mid M_i \text{ does not accept } w_i \}$

In other words, L_d contains those strings, which are not accepted by a TM of the same encoding as w.

Proof that L_u is not Recursive!

Assume L_u is recursive. Then $\overline{L_u}$ is recursive.

Now, we can reduce L_d to $\overline{L_u}$,



But then L_1 would be RE. We know L_d is not RE

So, this is a contradiction.

So, L_u is not recursive. [Proved]

12. Cook's theorem : SAT is NP—complete.

Notations : $s(q, j)$, T_{ij}, $c(i, j, t)$, $u(i, j)$

Rules :

1. for $0 \le j \le p(n)$ $\bigvee_{q \in Q} s(q, j)$

$$s(q, j) \longrightarrow \bigwedge_{q_1 \in Q - \{q\}} \overline{s(q_1, j)}$$

2. for $0 \le j \le p(n)$ $\bigvee_{i=0}^{p(n)} T_{ij}$

And $T_{kj} \longrightarrow \bigwedge_{i \neq k} \overline{T_{ij}}$

3. for $0 \le i, j \le p(n)$ $\bigvee_{t \in T} c(i, j, t)$

$$c(i, j, t) \longrightarrow \bigwedge_{t_1 \in T - \{t\}} \overline{c(i, j, t_1)}$$

4. $u(i, j) \Longrightarrow \wedge\, c(i, j, t) \rightarrow c(i, j+1, t)$

5. if $\delta(q,t) = \{(q_1, t_1, d_1), \ldots\}$

$$s(q,j) \wedge \cancel{c(i,j,t)} \wedge I_{ij} \wedge c(i,j,t) \longrightarrow$$

$$\bigwedge_{k \neq i} u(k,j) \wedge \Big[\big(s(q_1, j+1) \wedge \cancel{\in} I_{\phi(1)(j+1)} \wedge c(i, j+1, t_1) \big) \vee (\ldots) \ldots \Big]$$

$\phi(k)$ = next position of head

6. $s(q_0, 0) \wedge I_{00} \wedge c(0,0,w_1) \wedge c(1,0,w_2) \wedge \ldots$

$$c(n-1, 0, w_n) \wedge c(n, 0, B) \wedge \ldots c(p(n), 0, B)$$

7. $s(q_a, p(n))$

Given, $\delta(q,t) = \{(q_1, t_1, L), (q_2, t_2, R)\}$

Corresponding Formula

$$s(q,j) \wedge I_{ij} \wedge c(i,j,t) \longrightarrow$$

$$\bigwedge_{k \neq i} u(k,j) \wedge \Big[\big(s(q_1, j+1) \wedge I_{i-1, j+1} \wedge c(i, j+1, t_1) \big) \vee \big(s(q_2, j+1) \wedge I_{i+1, j+1} \wedge c(i, j+1, t_2) \big) \Big]$$

Ans)

13. PSPACE = : A language L is in PSPACE if L can be decided by some polynomial space deterministic Turing Machine.

NPSPACE : A language L is in NPSPACE if L can be decided by some polynomial space nondeterministic Turing Machine.

Prof :

It is easy to see that, PSPACE $\subseteq$ NPSPACE because a DTM is just an NDTM without choices.

Now, proving the other way requires Savitch's theorem.

By Savitch theorem, what an NDTM can do in $f(n)$ space can be simulated in a DTM in $f^2(n)$ space.

It works by a reachability finder DFC algorithm where the total space is shown to be $O(f^2(n))$

So, NPSPACE $\subseteq$ PSPACE

$\therefore$ PSPACE = NPSPACE (Proved)

2019-20

2005001
Anik

1. (a) No, the proof sketch is not acceptable. Because it inherently assumes that the pumping length p must be a prime number.

(b) One such scenario where a computationally harder problem is more preferable is the field of cryptography.

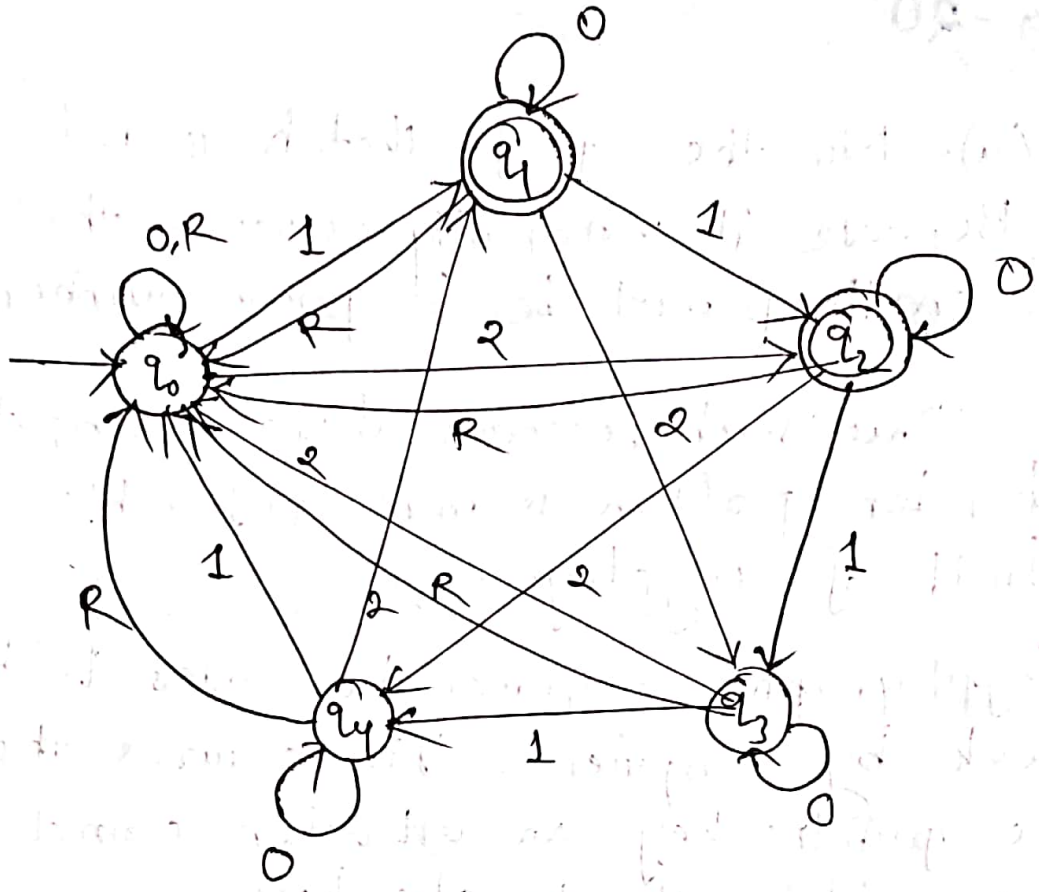
Cryptography requires the codes to be hard to break by computers. This ensures that, given the public key, an attacker cannot easily decrypt it to obtain the private keys.

For instance, factoring problem is a very hard problem that can not yet be solved in polynomial time. In RSA scheme chooses two large primes p and q as private keys and $n = pq$ as the public key.

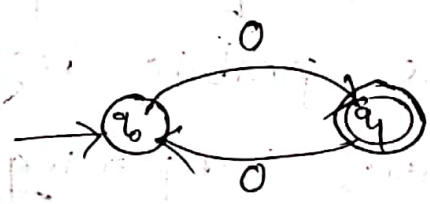
Thus, computing n, given p, q is very easy. However, doing the reverse is computationally very intensive when p and q are chosen sufficiently long enough.

Thus cryptography is a valid example.

(c)



2.(a) A state in a finite automaton serves as a memory element which remembers a particular configuration interpreted from the input received so far.



for e instance, the DFA here recognizes $L = \{ w \mid w \in \{0\}^* \text{ and } |w| = \text{odd} \}$

Here, q_0 remembers that so for, the input received is of even length and q_1 remembers that it is odd.

(b) i) $\phi^* \cup a^* \cup b^* \cup (a \cup b)^*$

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

Here, $(a \cup b)^*$ is the set of all strings over a, b.

So, other expressions are covered inside it.

So, equivalent expression $= (a \cup b)^*$

ii) $\left((a^* b^*)^* (b^* a^*)^*\right)^*$

Here, $(a^* b^*)^*$ denotes the set of strings that can be broken down into substrings containing some a's at start and some b's afterword.

It is clear that every string over $\{a, b\}$ can be broken down in this way.

So, $(a^* b^*)^* = \Sigma$

Now, $\left(\Sigma (b^* a^*)^*\right)^* = \Sigma^* = \Sigma = (a \cup b)^*$

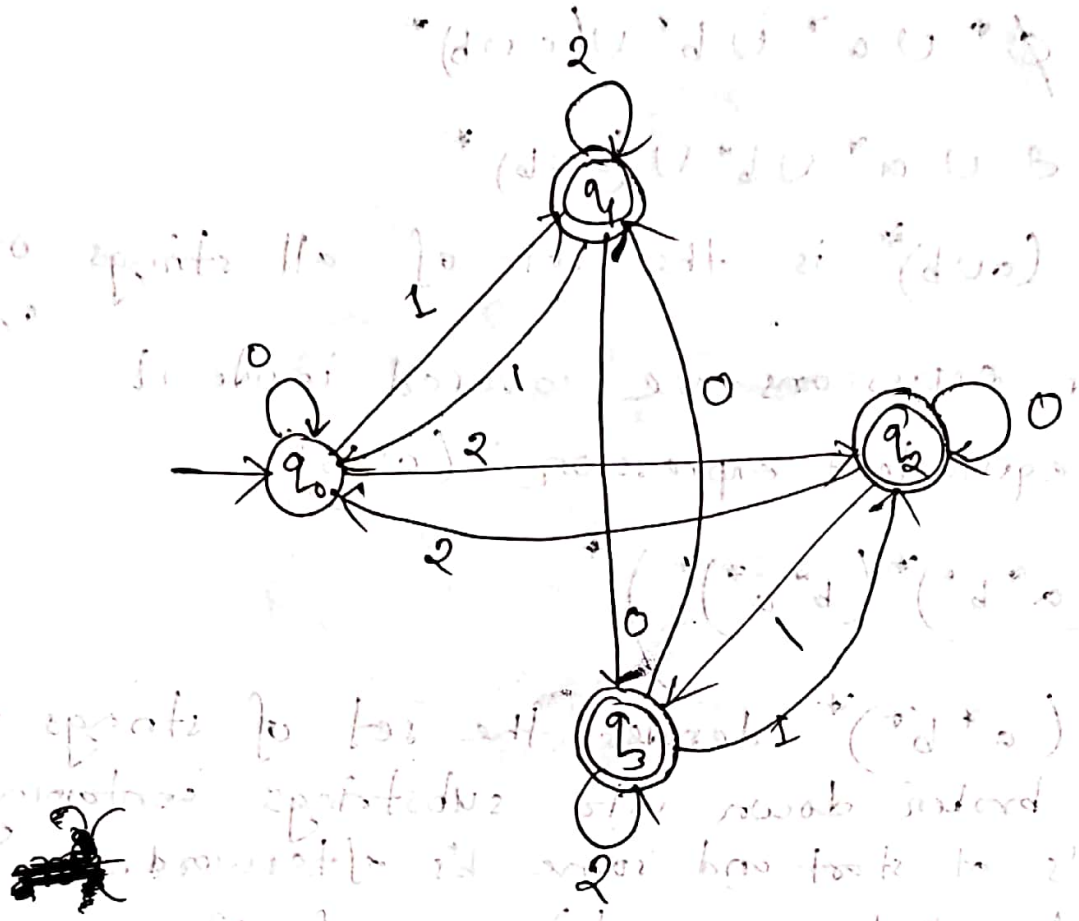
iii) $(a^* b)^* \cup (b^* a)^*$

Here, $(a^* b)^*$ = set of strings ending with b

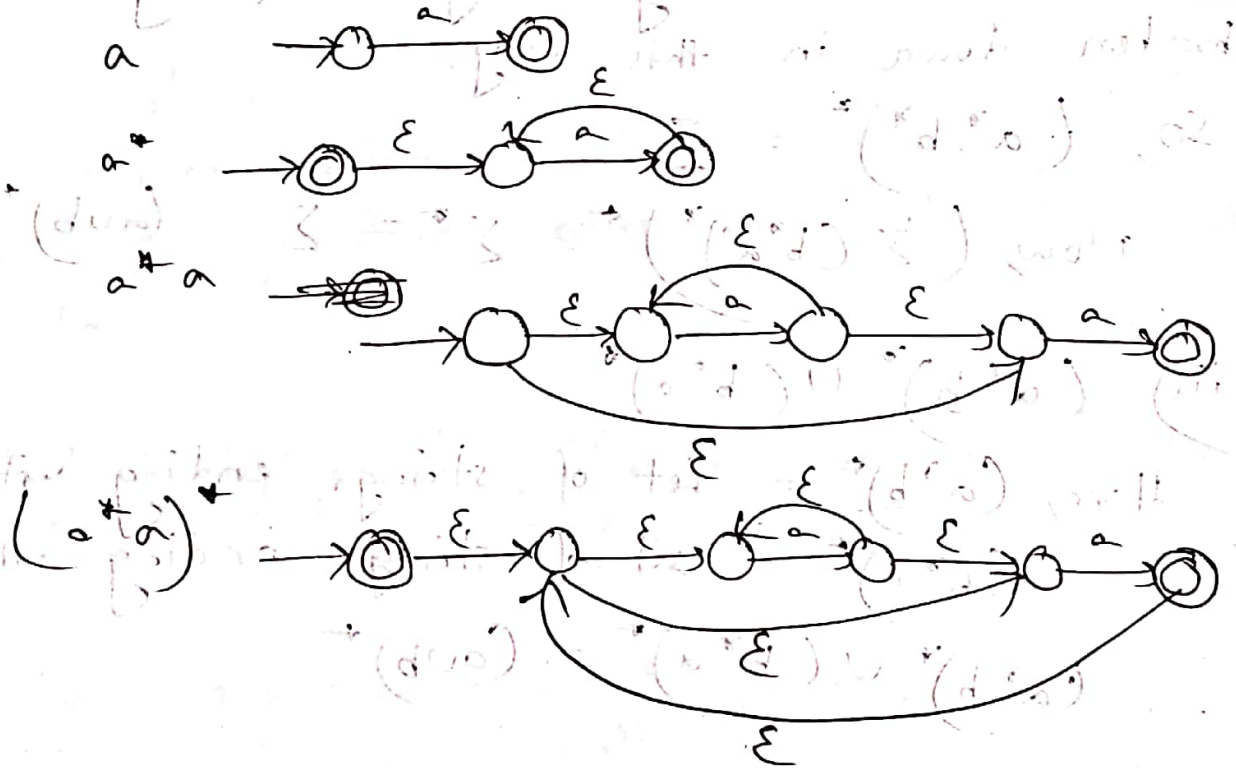
$(b^* a)^*$ = set of strings ending with a

$\therefore (a^* b)^* \cup (b^* a)^* = (a \cup b)^*$

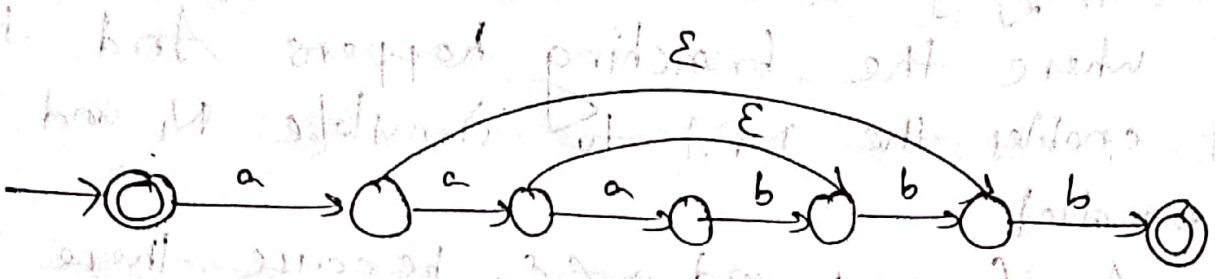
(c)



3. (a)



(6)



(c) Given $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

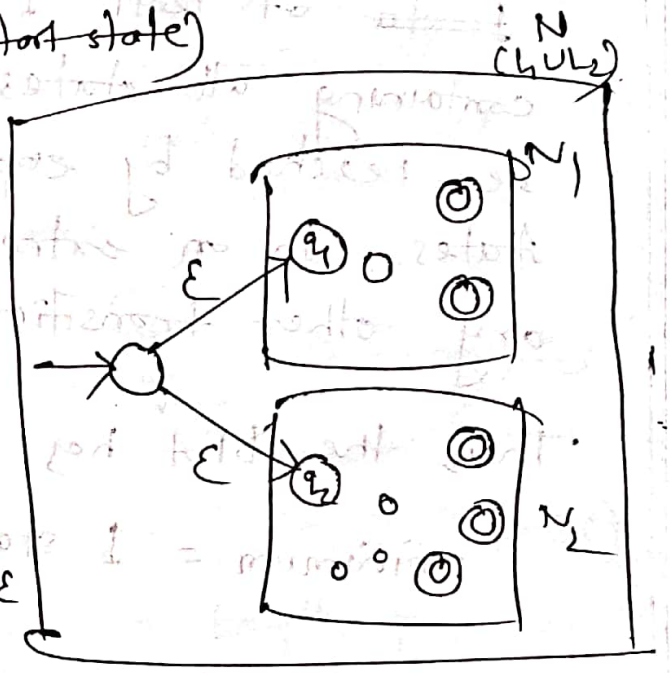
Let $N = (Q, \Sigma, \delta, q_0, F)$

where $Q = Q_1 \cup Q_2 \cup \{q_0\}$

$\Sigma = \Sigma$

$q_0 = q_0$ (new start state)

$F = F_1 \cup F_2$



$$\delta(q, a) = \begin{cases} \delta_1(q, a) \text{ if } q \in Q_1 \\ \delta_2(q, a) \text{ if } q \in Q_2 \\ \{q_1, q_2\} \text{ if } q = q_0 \text{ and } a = \varepsilon \\ \emptyset \text{ if } q = q_0 \text{ and } a \neq \varepsilon \end{cases}$$

Parts i) $\delta_1(q, a)$ if $q \in Q_1$ — This is true because when $q \in Q_1$, we simply follow the transitions of N_1

ii) Similar for $\delta_2(q, a)$ if $q \in Q_2$ for N_2

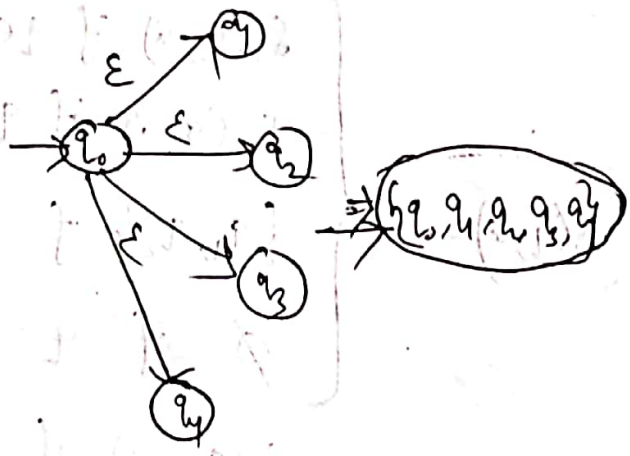
iii) $\{q_1, q_2\}$ if $q = q_0$, $a = \varepsilon$, because this is where the branching happens. And this part enables the NFA to simulate N_1 and N_2 in parallel.

iv) $\emptyset$ if $q = q_0$ and $a \neq \varepsilon$, because there is no other required transitions from the new start state.

4.(a) Since an NFA with five states has a start state, it's equivalent DFA must have at least 1 state, the new start state containing all states of original NFA that can be reached by empty transitions from start states. In an extreme case, there may not be any other transitions.

This the DFA has

minimum = 1 state



Maximum = 2^n states = 32 states
(all subsets)

(b) i) $n(b)$ is divisible by three

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

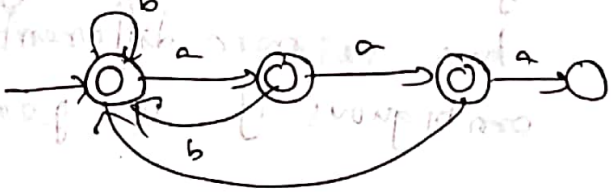
ii) no more than three a's

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

iii) exactly one occurrence of substring 'aaa'

$$\underbrace{(b \cup ab \cup aab)^*}_{\text{ends with b and no 'aaa'}} aaa \underbrace{(b \cup ba \cup baa)^*}_{\text{starts with b and no 'aaa'}}$$

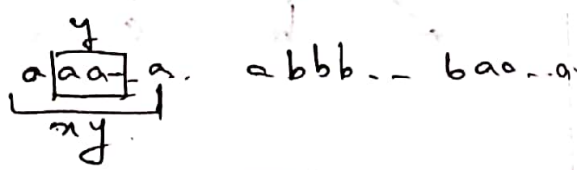
Help.: no aaa



(c) Proof: Assume the language $L = \{a^{2n} b^{3n} a^n \mid n \geq 0\}$ is regular. Then there is a pumping length p.

Let a string $w = a^{2p} b^{3p} a^p \in L$ and $|w| = 6p > p$

Now, take any possible division of $w = xyz$ where ~~$|y| \leq p$ and~~ $y \neq \varepsilon$, $|xy| \leq p$, it is clear that y will consist entirely of a's



Now, clearly, since $|y| \geq 1$,

$xy^2z = xz$ will not be of the form, $a^{2p-m} b^{2p} a^p$ where $m > 0$

But this fails to be w, as $(2p - m \neq 2 * p)$

So, L is not regular. [Proved]

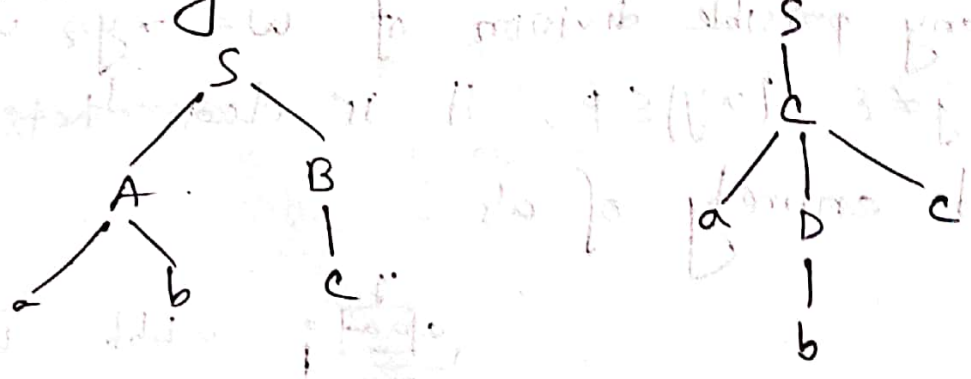
5. i) $S \rightarrow SS \mid (S) \mid \varepsilon$

ii) $S \rightarrow 0S1 \mid 1S0 \mid SS \mid \varepsilon$

6. Ambiguous grammar: A string is said to be generated ambiguously by a grammar if it has two or more different parse trees. A grammar G is ambiguous if it generates some string ambiguously.

Inherently Ambiguous Language: If no unambiguous grammar can be designed for a language it is inherently ambiguous.

The string 'abc' has two parse trees



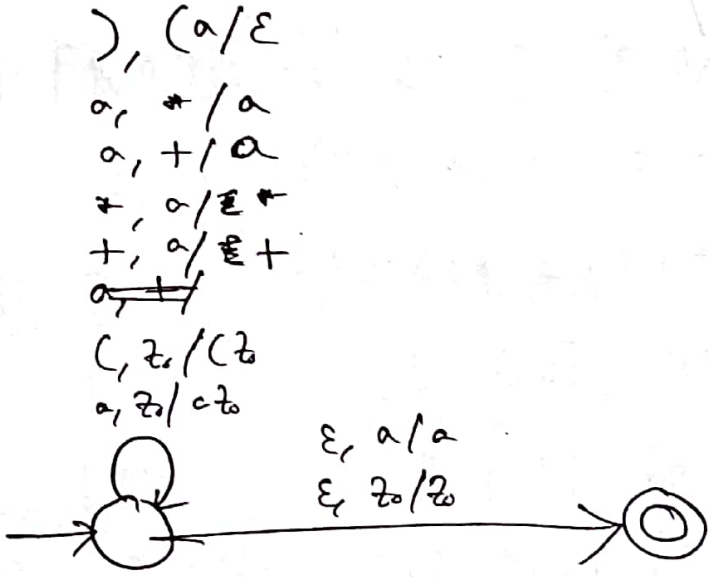
7.

A, S, B				
A, S, B	A, C,			
A, B	A, C	A, C,		
B, A	A, S, C	A, C	A, S, B	
A, Z	A, Y	A, Y	A, Z	A, Z
0	1	1	0	0

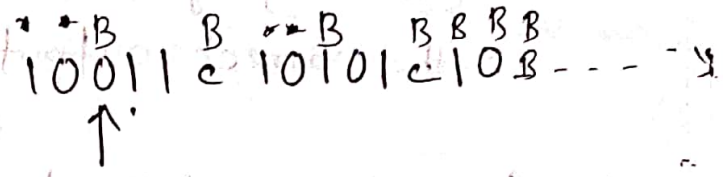
So, 01100 is in L

Informally → L = strings of length more than 1, that start and end with the same character

8.



9. Assume leftmost bit = LSB



$\delta([q_0, B, B], [B, a]) = ([q_1, a, B], [*, a], R)$; $a = 0/1$

$\delta([q_1, a, B], [B, 0/1]) = ([q_1, a, B], [B, 0/1], R)$

$\delta([q_1, a, B], [B, c]) = ([q_2, a, B], [B, c], R)$

$\delta([q_2, a, B], [*, 0/1]) = ([q_2, a, B], [*, 0/1], R)$

$\delta([q_2, a, B], [B, b]) = ([q_3, a, b], [*, b], R)$

$\delta([q_3, a, b], [B, 0/1]) = ([q_3, a, b], [B, 0/1], R)$; $b = 0/1$

$\delta([q_3, a, b], [B, B]) = ([q_4, B, B], [B, a \wedge b], L)$;

here $a \wedge b$ is the = a AND b

$\delta([q_4, B, B], [B, 0/1]) = ([q_4, B, B], [B, 0/1], L)$

$\delta([q_4, B, B], [B, c]) = ([q_5, B, B], [B, c], L)$

$\delta([q_5, B, B], [*/B, 0/1]) = ([q_5, B, B], [*/B, 0/1], L)$

$\delta([q_5, B, B], [B, c]) = ([q_6, B, B], [B, c], L)$

$\delta([q_6, B, B], [B, 0/1]) = ([q_6, B, B], [B, 0/1], L)$

$\delta([q_6, B, B], [*, 0/1]) = ([q_0, B, B], [*, 0/1], R)$

No other rule is needed since when $[q_0, B, B]$ gets $[B, c]$, it halts automatically with correct output already written

10. (a) The proof has two directions. Both are proven below:—

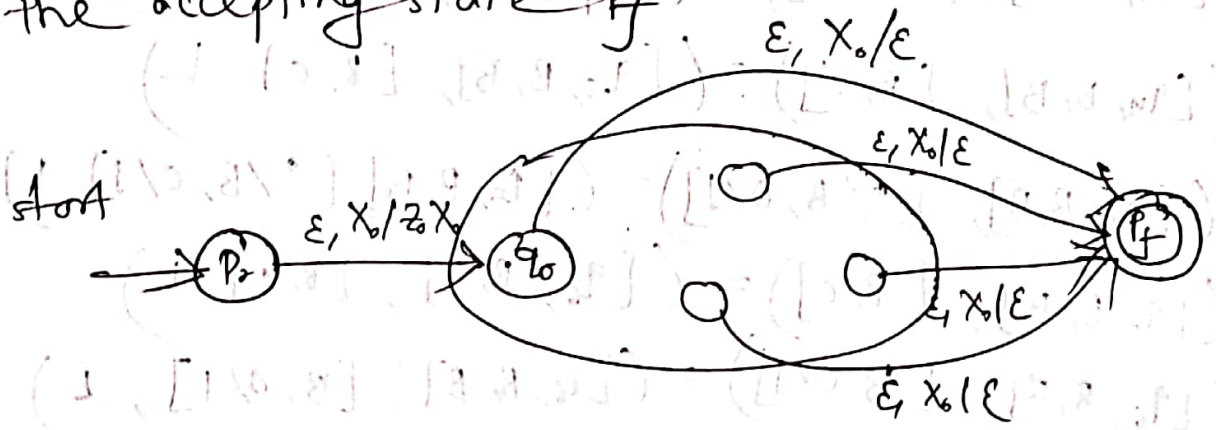
From Empty stack to final state:

Let $L = L(N(P_N))$ for some PDA $P_N = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where P_N is a PDA with empty stack.

Another PDA P_f is constructed that accepts by final states.

— We need a new start symbol X_0 and a new start state p_0 to solely push the symbol Z_0 onto the stack and transfer to the original start state q_0

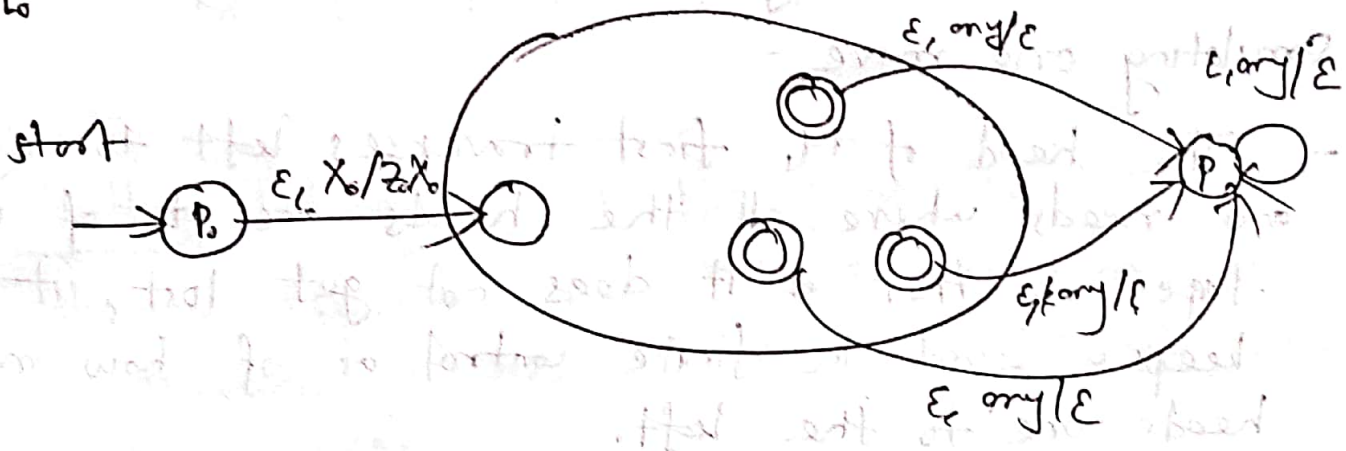
— We also need a new state p_f which is the accepting state of P_f and from every state, whenever it sees X_0 on top of the stack, it goes to the accepting state p_f.



from final state to empty stack.

Let P_F be a PDA, $P_F = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, another $L = L(P_F)$ Now another PDA with empty stack can be made such that, $L = N(P_N)$

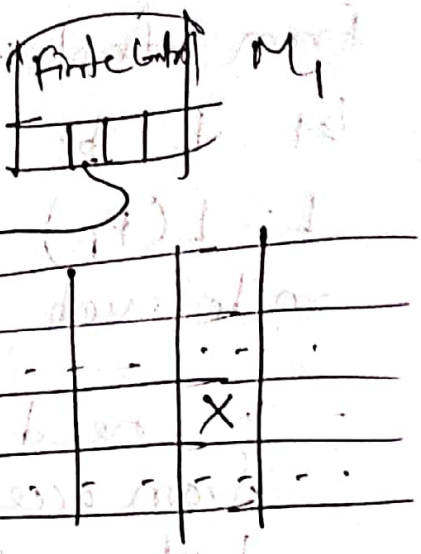
- We need a new state p and add ε transitions from accepting states of P_F to p. p pops its stack content without reading any input symbol
- So that, the stack is not emptied accidentally, we add a new start state p_0 and new start symbol X_0. p_0 pushes Z_0 onto stack and transfers to q_0



(b) Let M be a TM with k tapes. It can be simulated with a TM with one tape having 2k tracks and storage in state.

Construction:

Let a TM M_1 with one tape have 2k tracks.



Half of these tracks hold the tape contents of M. For instance, in figure, tracks 2 and 4 holds the content of tape 1 and 2 of M respectively. The other tracks hold a single head marker that indicates at which position the corresponding head of tape is currently at.

Simulating one move -

- The head of M_1 first traverses left to right and reads where all the heads are at of each tape. So that it does not get lost, it keeps a count in finite control as of how many heads are to the left.
- Once the head of M_1 has read all head positions and scanned the input symbols of that position of corresponding ~~stat~~ tape, now as it also knows the current state of M, from finite control, it knows, what move M will make.
- Accordingly, it again traverses the heads left to right, moves the head marker left or right

and changes the symbol on previous position
— Thus one move is simulated.

∴ M needs $O(n)$ moves, M_1 needs $O(n^2)$ moves.

11. (a) ~~Regal~~ $\{w \mid w \in \{0,1\}^* \text{ and } |w| = \text{even}\}$

(b) $\{0^n 1^n \mid n \geq 0\}$

(c) $\{0^n 1^n 2^n \mid n \geq 0\}$ (d) Universal language L_u

(e) Diagonalization Language L_d

$L_d = \{w_i \mid M_i \text{ does not accept } w_i\}$

~~(e)~~

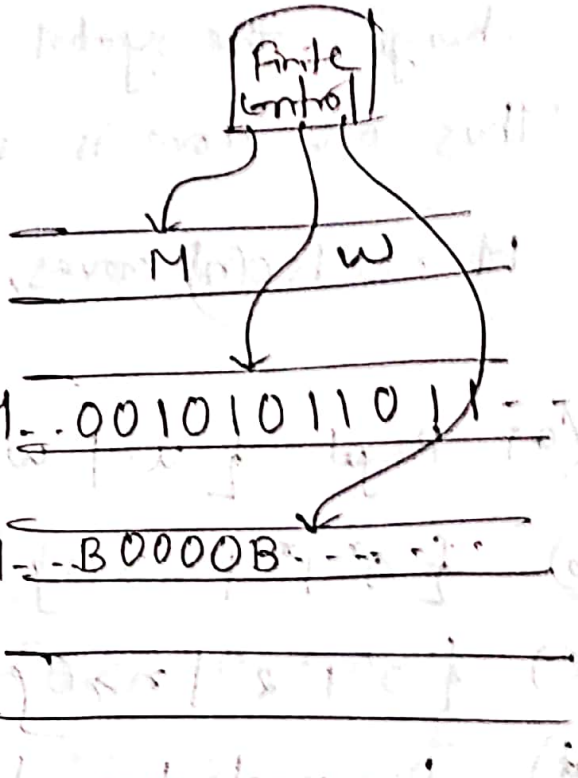
12. Halting Problem: Let $H(M)$ be the set of inputs w, for which the TM M halts on w. Halting problem is the set of pairs (M, w) such that $w \in H(M)$.

Proof:

First proving that Halting problem is RE.

We can construct a Turing machine that simulates the machine M's actions on w.

It is a multitape Turing machine with a finite control, input tape containing M and w. M means the binary coding of M here.



Then there are three other tapes, tape of M, state of M, scratch tape.

This Turing machine M_H, first checks whether M is valid, if not halts. Otherwise initializes its second tape with input and moves heads on tapes and acts according to coding of M.

If M_H halts, M_H halts and accepts w.
If M does not halt, M_H keeps running.

Not Recursive: Let's assume, Halting problem D recursive.

Then we can reduce L_u to the halting problem.

Determining whether M halts on input w, if it does, we could simulate M on w further to check if it halt in an accepting state. If yes accept (M,w) otherwise reject. And if M does not halt on w, reject.

Thus, L_u is reduced to halting problem

But L_u is not recursive. So, Halting problem is not recursive.

$\therefore$ L_H is RE but not recursive.

13. (a) By Cook's Theorem, SAT is known to be an NP-complete problem. So, for every problem $X \in NP$, $X \leq_p SAT$

Now, if SAT can be solved in polytime, fore every $X \in NP$, X can be solved in polytime.

So, $P = NP$. So, True.

(b) False. it is the other way.

$P \subseteq NP \subseteq PSPACE$

$\therefore$ PSPACE is a superset of NP

(c) True. $PSPACE = NPSPACE$

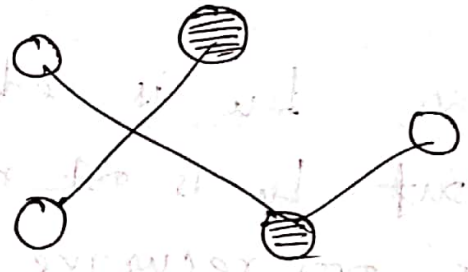
As, $NSPACE(f(n)) \subseteq SPACE(f^2(n))$

So, $PSPACE = NPSPACE$

Jan 2020

1.(a) Dominating set problem asks for a subset of vertices in a Graph G, subset $D \subseteq V$, such that for every $v \in V$, either, $v \in D$ or v has a neighbour in D. so that $|D| \leq k$

Certificates:



1. Given an instance of dominating set, or the set D,
it can be checked in $O(|D|+|E|)$ whether it is valid.
2. Traverse through all edges which are adjacent to the vertices of D and keep track of the number of distinct vertices visited. If it reaches $|V|$, return true
3. This is a polytime certificate. So, DS is NP.

(b) i) He chose a particular string for which the conditions is fulfilled. But another string can be chosen that violates conditions.

For instance, take $w = 0^p 1 0^p 1$, $|w| = 2p+2 \geq p$
But for any $w = xyz$, $|xy| \leq p$, $y \neq \varepsilon$ it is clear that $y = 0^q$ for some $q > 0$,

So, $xy^0z = 0^{p-2}10^p1$,

This does not belong to L for any q.

So, L_1 is not regular.

iii) Here, the choice of y is invalid.

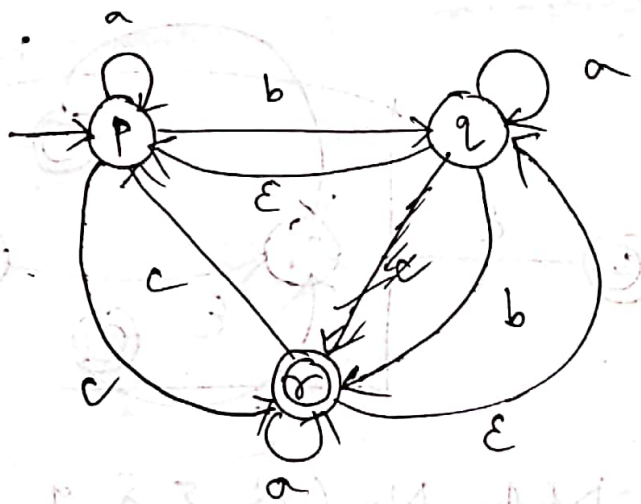
$w = \underbrace{\overbrace{000}^{x}\overbrace{\cdots 0}^{y}}_{0^p}\underbrace{00\cdots 0}_{0^{2p}}\underbrace{1\cdots 1}_{1^p}$

Here, $y \neq \varepsilon, |xy| \leq p$, thus, since $w = 0^{3p}1^p$, y must consist entirely of zeros.

So, $xy^iz = 0^{3p+i|y|}1^p$, But as $|y| > 0$,

$3p + |y| \neq 3p$ So, L is not regular.

(c)



All strings = every string w that contains more than one b or has at least a c ~~ends with c a*~~

So, $L = \Sigma^* b \Sigma^* b \Sigma^* \cup \Sigma^* c \Sigma^*$

2.(a) $L = \{ w \mid$ the last two characters of w are equal $\}$

(b) i) $\emptyset^* \emptyset^* \emptyset^* \emptyset^* = \{\varepsilon\}$

ii) $\emptyset^* \cup \{\varepsilon\} = \{\varepsilon\}$

iii) $(\emptyset^* \cup 1)\{0,1\}$

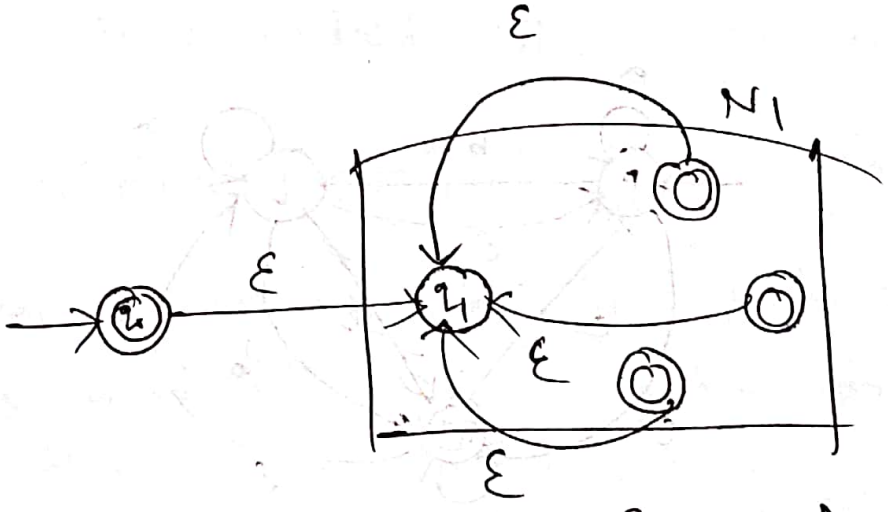
$= \{\varepsilon, 1\}\{0,1\} = \{0, 1, 10, 11\}$

iv) $(11)^+ \emptyset (00)^* = \emptyset$

v) $(\emptyset^* \cup 0)\{0, 1, 11\} = \{0, \varepsilon\}\{0, 1, 11\}$

$= \{00, 01, 011, 0, 1, 11\}$

(b) $N_1 = (Q_1, \Sigma, \delta_1, q_1, f_1)$



Here, for new, NFA, $N = (Q, \Sigma, \delta, q_0, F)$, q_0 is a new start state and has empty transition to q_1.

Now, i) when $q \in Q_1$, $q \notin F_1$, this means the non-accepting states of N_1. They behave like before. $\delta(q, a) = \delta_1(q, a)$ for $q \in Q$, $q \notin F_1$

ii) when $q \in Q_1$, $q \in F_1$, they almost behave like before, but when symbol is ε, they also go to q_1

$$\therefore \delta(q, a) = \begin{cases} \delta_1(q, a) & \text{for } q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q, a) \cup \{q_1\} & \text{for } q \in F_1 \text{ and } a = \varepsilon \end{cases}$$

iii) And, from new start state only on ε, it goes to q_1

$$\therefore \delta(q, a) = \begin{cases} \{q_1\} & \text{if } q = q_0, \ a = \varepsilon \\ \emptyset & \text{if } q = q_0, \ a \neq \varepsilon \end{cases}$$

4. (a) $(1+\varepsilon)(0.0^*1)^*0^*$

= strings ~~contain~~ not containing two consecutive 1's

(b) The converted DFA is as follows, ...

Here, calculating $E(q_i)$.

$E(q_0) = \{q_0, q_1, q_2, q_4, q_7\}$

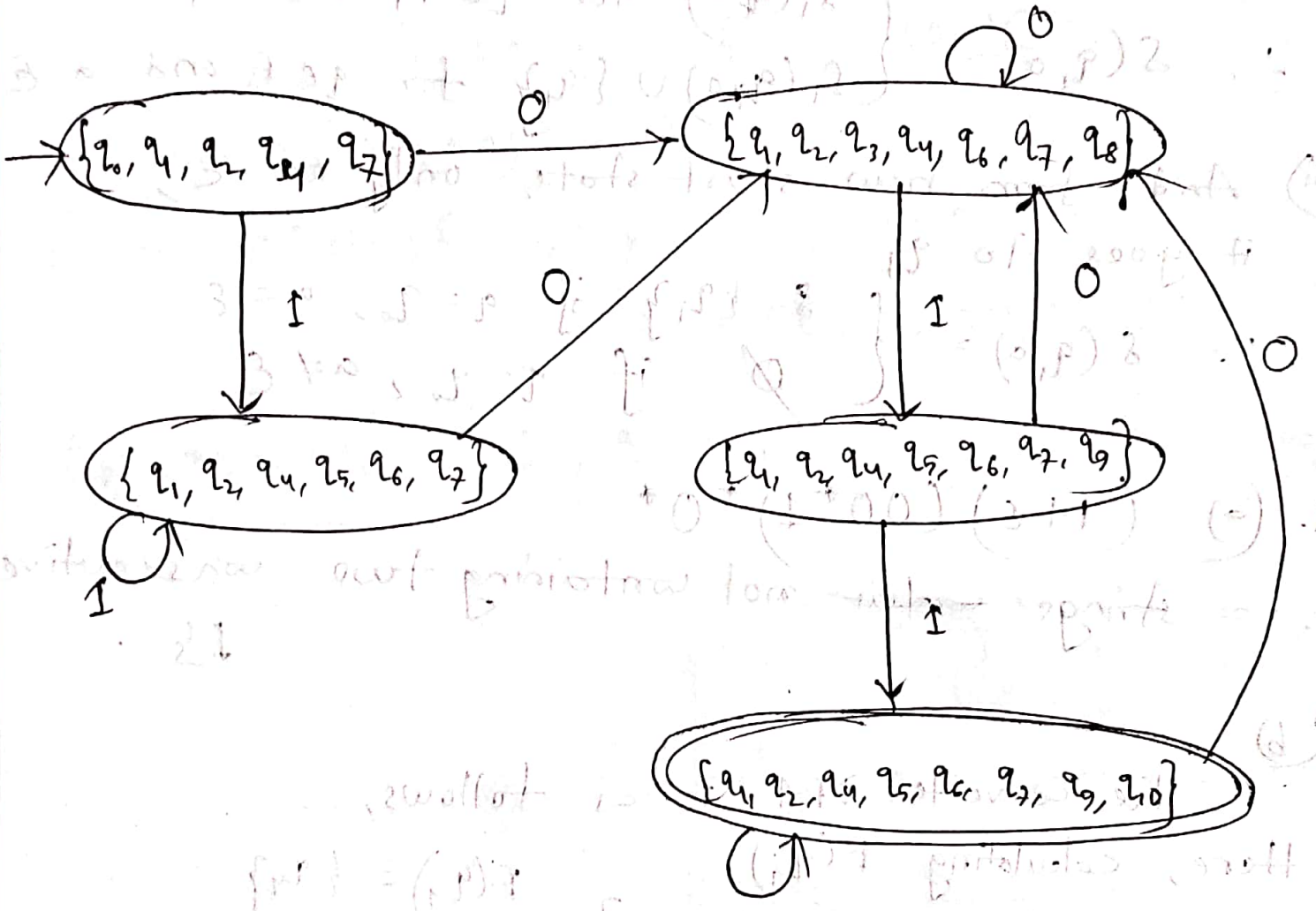
$E(q_1) = \{q_1, q_2, q_4\}$

$E(q_2) = \{q_2\}$

$E(q_3) = \{q_3, q_6, q_1, q_2, q_4, q_7\}$

$E(q_4) = \{q_4\}$

$E(q_5) = \{q_5, q_6, q_1, q_2, q_4, q_7\}$


{q0, q1, q2, q4, q7}
{q1, q2, q3, q4, q6, q7, q8}
{q1, q2, q4, q5, q6, q7}
{q1, q2, q4, q5, q6, q7, q9}
{q1, q2, q4, q5, q6, q7, q9, q10}
0
1
0
1
0
0
1
1
0

5. (a) $S \rightarrow POP \mid P1POP$

$P \rightarrow 0P \mid 1P \mid \varepsilon$

(b) $S \rightarrow SS \mid 0S1 \mid 1S0 \mid \varepsilon$

6. $S \rightarrow aPbcQd \mid aTd$

$P \rightarrow aPb \mid \varepsilon$

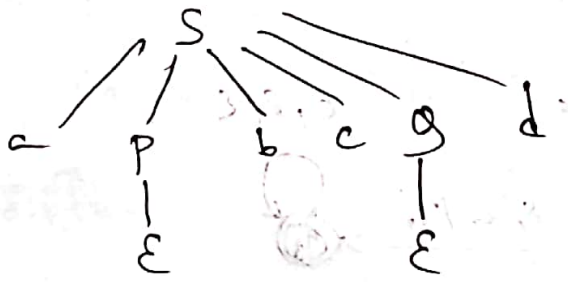
$Q \rightarrow cQd \mid \varepsilon$

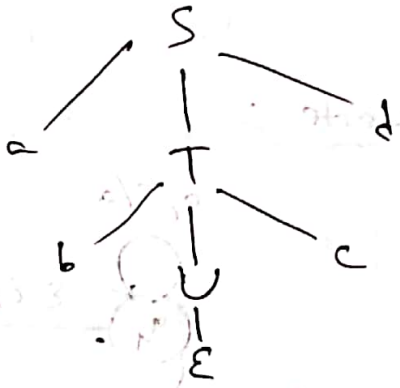
$T \rightarrow aTd \mid bUc$

$U \rightarrow bUc \mid \varepsilon$

A string = abcd

Parse Trees:



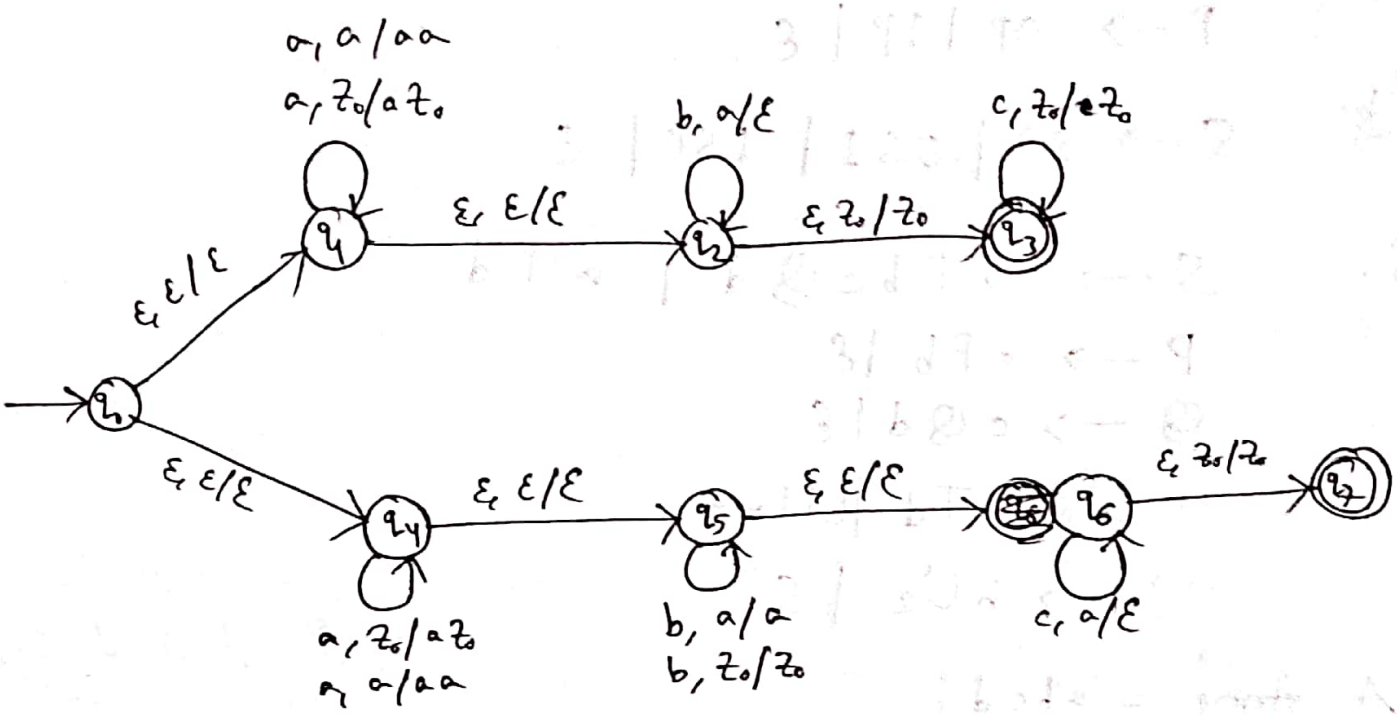


7.

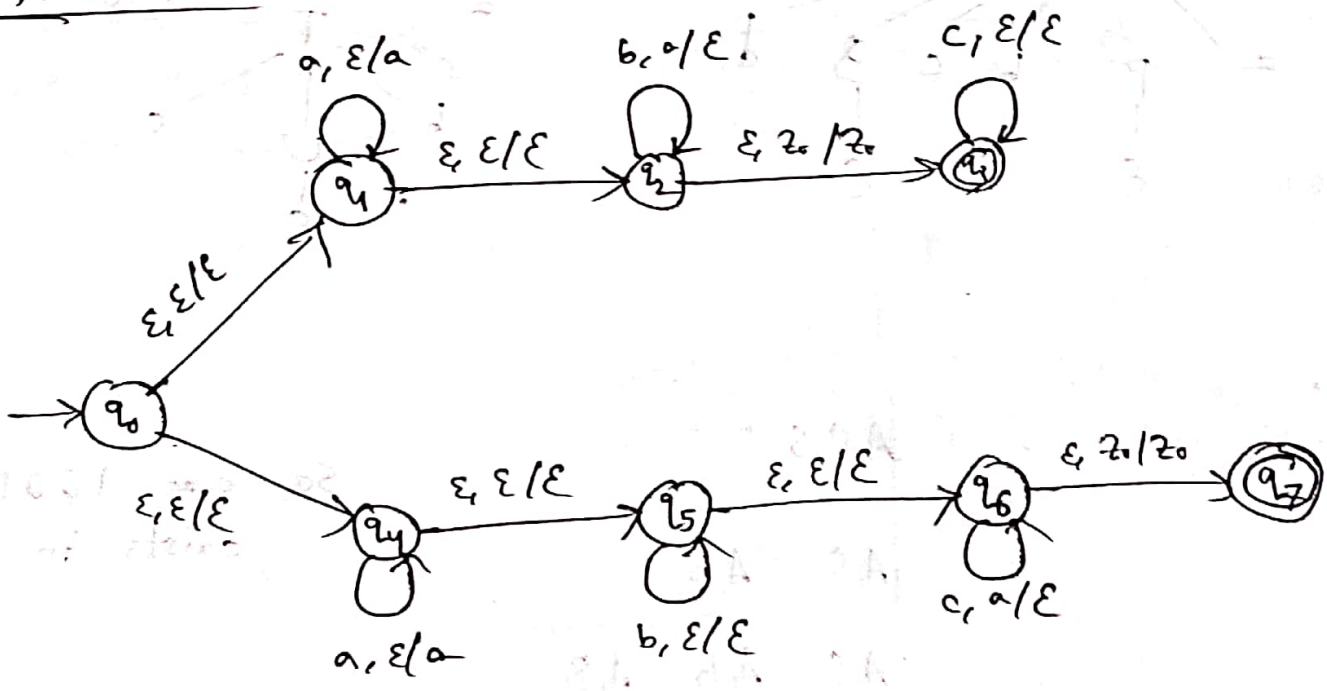
A, C, S				
A, C	A, B			
A, C	A, B	A, B		
A, C	A, B	A, B	A, S, C	
A, Y	A, Z	A, Z	A, Y	A, Y
1	0	0	1	1

So, ~~can~~ 10011 exists in L(G)

8.

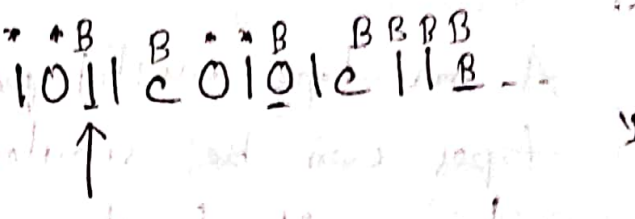


9. Alternate :



9. ~~I assume the number has LSB on left most bit~~

Given the numbers have equal length.



Rules:

$\delta([q_0, B, B], [B, a]) = ([q_4, a, B], [*, a], R)$

$\delta([q_1, a, B], [B, 0/1]) = ([q_1, a, B], [B, 0/1], R)$

$\delta([q_4, a, B], [B, c]) = ([q_2, a, B], [B, c], R)$

$\delta([q_2, a, B], [*, 0/1]) = ([q_2, a, B], [*, 0/1], R)$

$\delta([q_2, a, B], [B, b]) = ([q_3, a, b], [*, b], R)$

$\delta([q_3, a, b], [B, 0/1/c]) = ([q_3, a, b], [B, 0/1/c], R)$

$\delta([q_3, a, b], [B, B]) = ([q_4, B, B], [B, a \text{ XOR } b], L)$

$\delta([q_4, B, B], [B, 0/1]) = ([q_4, B, B], [B, 0/1], L)$

$\delta([q_4, B, B], [*, 0/1]) = ([q_5, B, B], [*, 0/1], L)$

$\delta([q_5, B, B], [*, 0/1]) = ([q_5, B, B], [*, 0/1], L)$

$\delta([q_5, B, B], [B, c]) = ([q_6, B, B], [B, c], L)$

$\delta([q_6, B, B], [B, 0/1]) = ([q_6, B, B], [B, 0/1], L)$

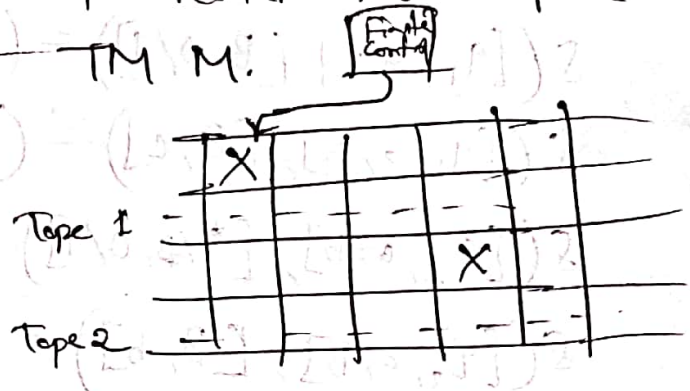
$\delta([q_6, B, B], [*, 0/1]) = ([q_0, B, B], [*, 0/1], R)$

Since there is no other carry, no other rule is needed.

10. a) i) True.

A ~~one tape~~ multitape TM M consisting of k tapes can be simulated by a single tape TM having 2k tracks and storage in states.

Here, for example, track 3 and 4 hold the tape content of tape 1 and 2 of TM M.



This TM can simulate one move of M in O(n) time where n is the input size.

So, what M can do in O(n) moves, this can do the same in $O(n^2)$ moves.

ii) True.

A multitape TM can simulate n steps of ~~a~~ conventional computer in $O(n^3)$ steps. Because there can be at most $O(n^2)$ addresses to be looked up for each instruction and n of them will take $O(n^3)$ include shifting and other details.

If it is single tape TM, it can do in $O(n^6)$ steps.

n^3, n^6 are both polynomials of n

11. (a) Recursively Enumerable: If a language can be recognized by a Turing Machine, it is called RE language. Example: $\{0^n 1^n 2^n \mid n \geq 0\}$, L_u

(b) Recursive : A language L is recursive if L=L(M) for some Turing machine M such that,

- if $w \in L$, M accepts
- if $w \notin L$, M eventually halts without accepting

Example: $\{0^n 1^n 2^n \mid n \geq 0\}$

12. b) i) not RE - L_d = Diagonalization language

$L_d = \{ w_i \mid M_i \text{ does not accept } w_i \}$

ii) RE but not recursive - L_u = Universal language, Halting problem

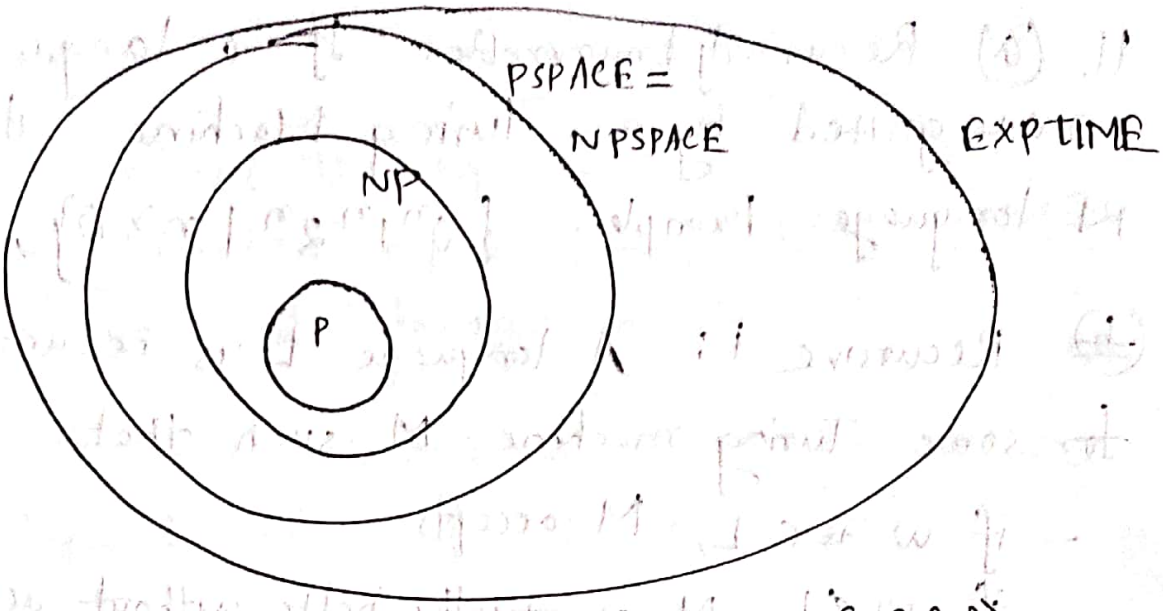
12. a) Cook's Theorem: SAT is NP-complete.

If problem Y is NP-complete, for every $X \in NP$, $X \leq_p Y$. Now, if Y is solvable in polytime, X can also be solved in polytime by means of the reduction.

So, every $X \in NP$, belongs to P

thus, finding polytime solution for an NP-complete problem implies P = NP.

b)



Here, we know, $NSPACE(f(n)) \subseteq PSPACE(f^2(n))$

So, PSPACE = NPSPACE

It is also more belived that $P \subset NP$ as there exists some NP-complete problems. And it can be proved that $NP \subseteq PSPACE$ since, a machine making polynomial moves can occupy at most polynomial cells. And EXPTIME means problems solvable in exponential time, which is a superset of all. Since the total number of moves a TM makes is at most exponential, given that it halts.

2017-10

1. (a) Boolean Satisfiability Problem: Given a boolean formula ϕ consisting of boolean variables and operations (AND, OR), the problem of finding whether valid assignment to the variables exists such that ϕ evaluates true, is the Boolean SAT problem.

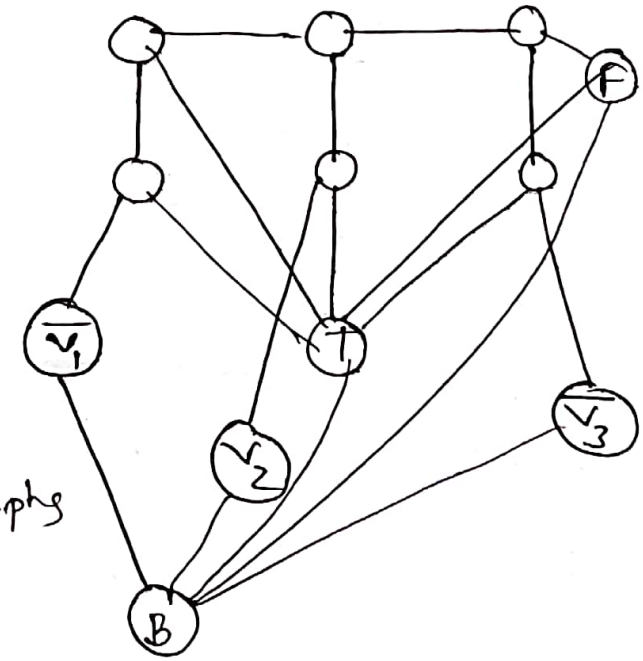
(b) We know 3-SAT is NP-complete.

Now, Graph 3-coloring is NP.

Proof: Given a coloring instance, we can check all edges in $O(|E|)$ time to check whether any pair of vertices on same edge have same color. This is a polytime certifier.

& Now, we reduce 3-SAT to Graph 3-Coloring

Reduction: For every clause in the CNF form, we form the graph shown here, it is easy to check that, it has a 3-coloring iff not all variables $\overline{v_1}$, v_2, $\overline{v_3}$ are false.



Thus, it corresponds to

$$\overline{x_1} \vee x_2 \vee \overline{x_3}$$

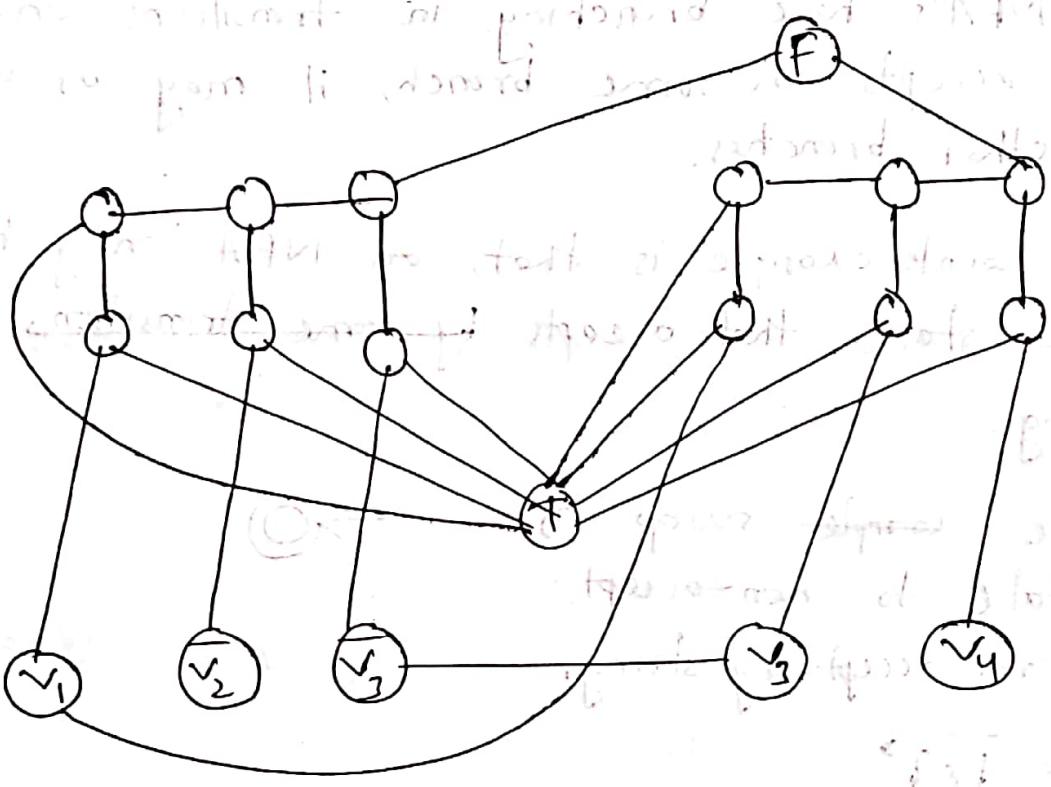
Similarly, for all clauses, graphs are formed and thus the reduction is complete

(c) Given formula

$$(x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_1 \vee x_3 \vee x_4)$$

The graph is as follows,

Here, $v_1, v_2, \ldots$ etc and T, F are all connected to B by default



2. (a) Here, $\Sigma^* - L = \overline{L}^*$

Now, for achieving the complement of a language from original DFA, simply swapping the accepting and non-accepting states works.

No, this will ~~not~~ work for every DFA.

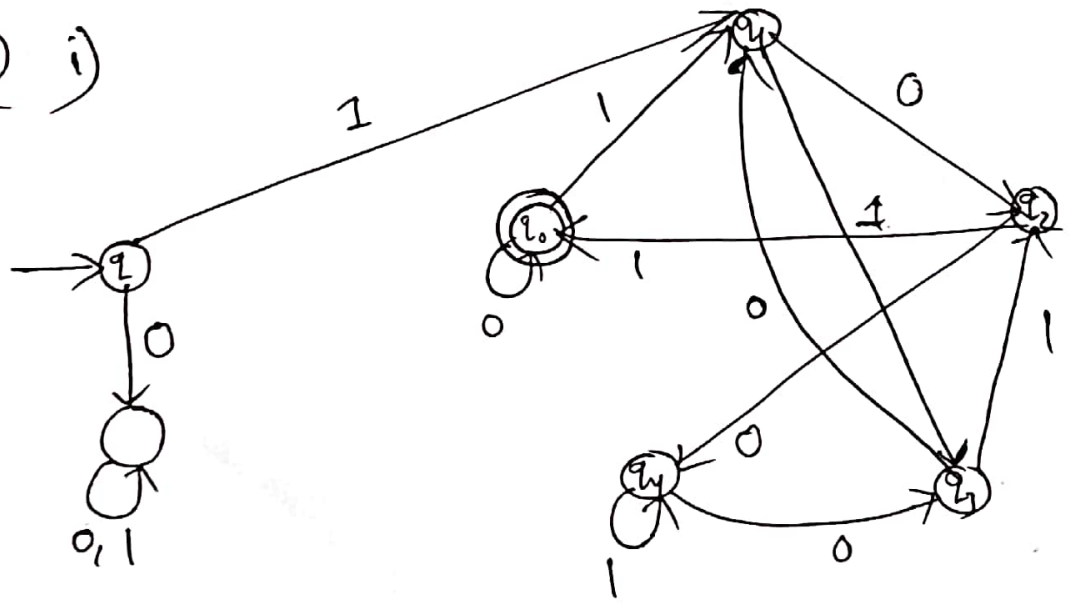
— No, this method does not work for NFA's. Because, NFA's have branching in transitions and when it accepts in some branch, it may as well reject in other branches.

A simple counterexample is that, an NFA may have only one state that accepts ~~by some transitions~~ only empty string.

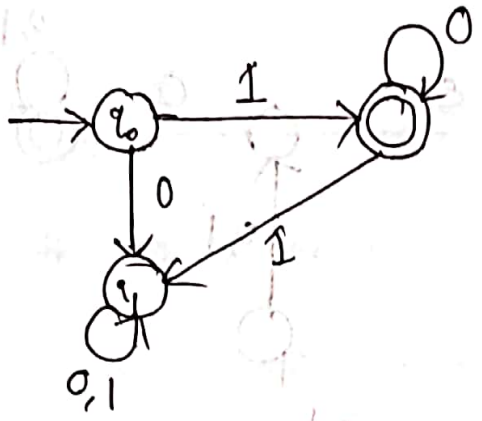
But if we ~~complete~~ swap its accept state to non-accept, it does not accept any string.

$$\emptyset \neq \overline{\{\varepsilon\}}^*$$

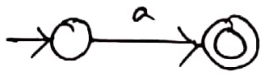
(b) i)



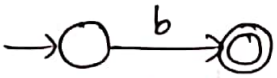
(ii) Π is $= 10^*$



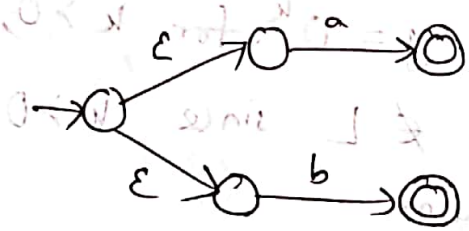
(c) a



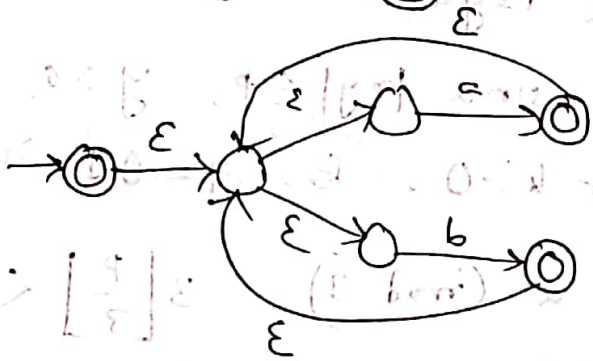
b



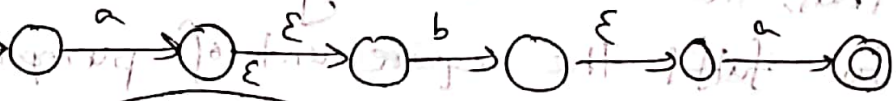
$a \cup b$



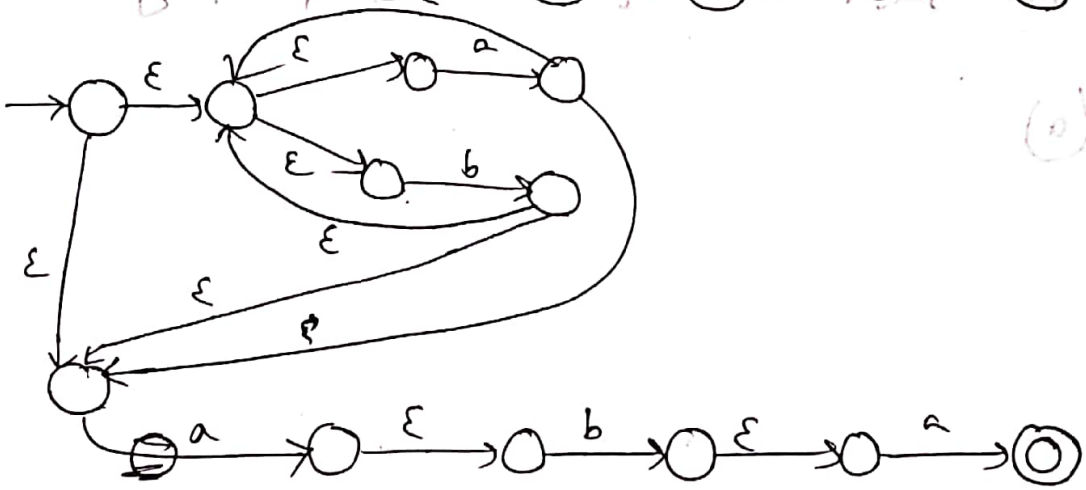
$(a \cup b)^*$



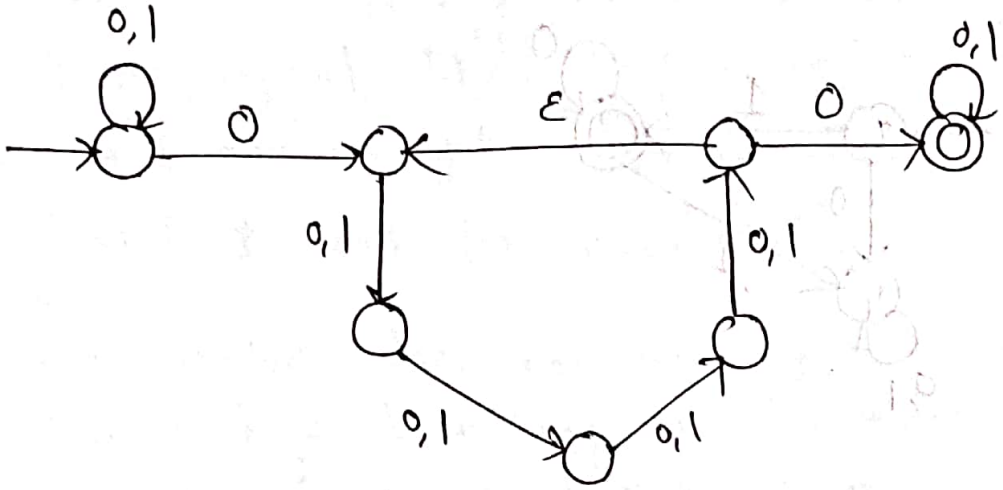
~~$(a \cup b)^*$~~ aba



$(a \cup b)^* aba$



3. (a)



(b) (i) Wrong string chosen.

if $w = 0^p 1 0^p 1$, $y = 0^k$ for $k > 0$,

so, $xy^0z = 0^{p-k}10^p1 \notin L$ since $k > 0$.

So, L is non regular

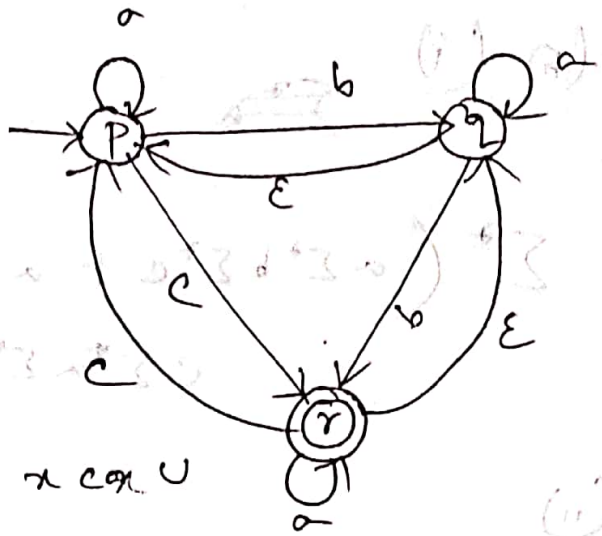
ii) if $w = 0^{2p}1^p$, since $|xy| \leq p$, $y > 0$. y must be $y = 0^k$ for $k > 0$. So, $y = 01$ is wrong

iii) if $p \equiv 1$ or $2 \pmod 3$ $\quad 3\left\lfloor \frac{p}{3} \right\rfloor < p$

Thus, the chosen string is not long enough to fulfil the requirement of pumping lemma.

4. (a)

(i)



Interpretation:

$\Sigma^* b \Sigma^* b \Sigma^* \cup \Sigma^* c \Sigma^*$

for length three or less,

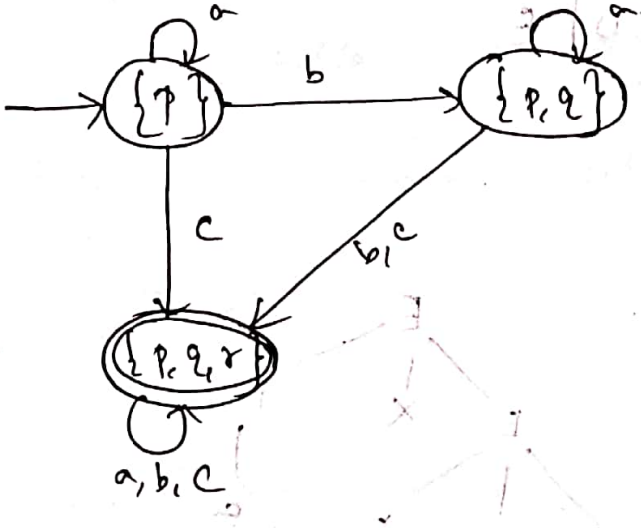
$bbx \cup xbb \cup bxb \cup xxc \cup xcx \cup cxx \cup cxcx$

where $x \in \{a, b, c\}$

(ii) $E(p) = \{p\}$

$E(q) = \{p, q\}$

$E(r) = \{p, q, r\}$



(b) (i) 

$$\Sigma^*(a\Sigma^*b\Sigma^*c + a\Sigma^*c\Sigma^*b + b\Sigma^*a\Sigma^*c + b\Sigma^*c\Sigma^*a + c\Sigma^*a\Sigma^*b + c\Sigma^*b\Sigma^*a)\Sigma^*$$

(ii) $aaaa^*(bb)^*b$

(iii) $(aa)^*(bb)^*b \cup a(aa)^*(bb)^*$

(iv) $aa\Sigma^*aa + ab\Sigma^*ab + ba\Sigma^*ba + bb\Sigma^*bb$

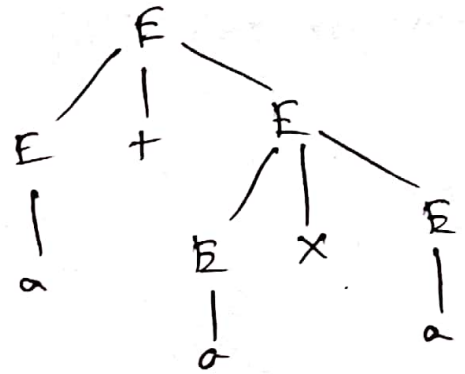
(v) $(b^*ab^*ab^*ab^*)^*$

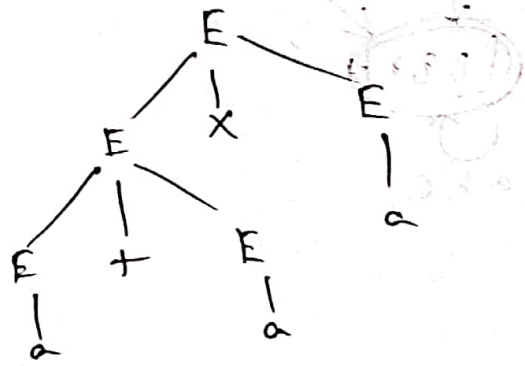
5. (a) $S \rightarrow SS \mid 0S1 \mid 1S0 \mid \varepsilon$

(b) $S \rightarrow P1P1P1P$

$P \rightarrow 0P \mid 1P \mid \varepsilon$

6. (a) if $w = a+a \times a$

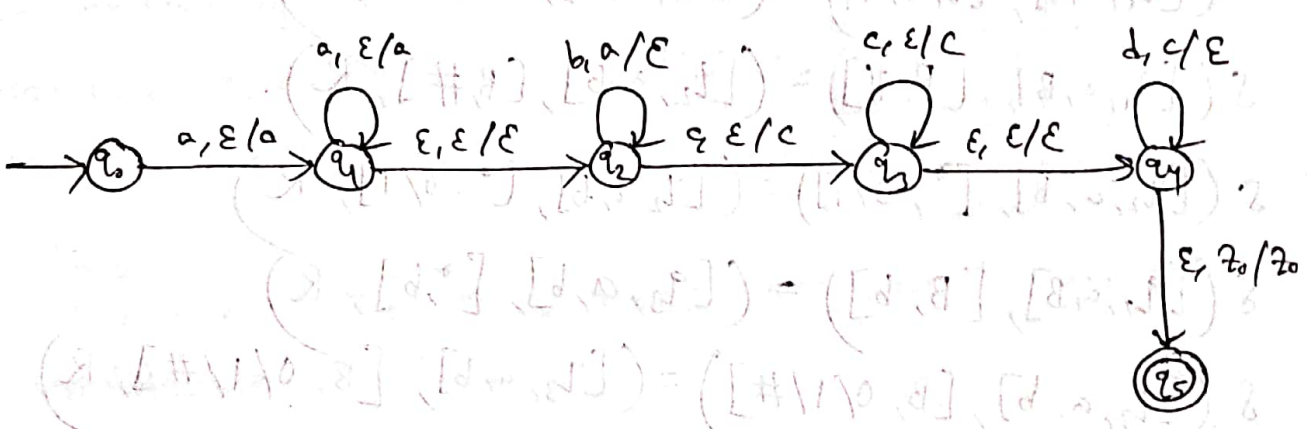




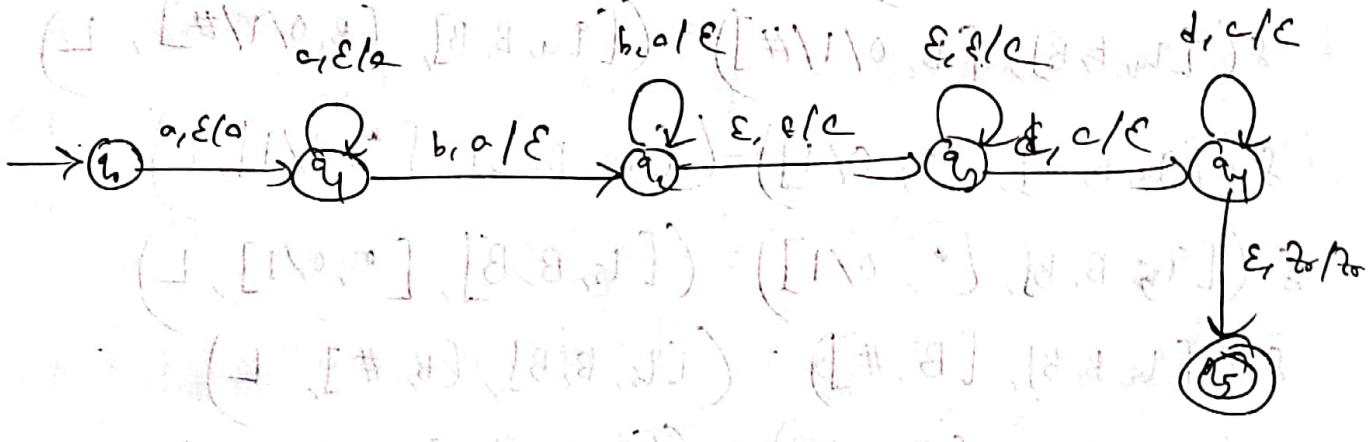
(b)

$$E \rightarrow T \mid E+T$$
$$T \rightarrow F \mid T*F$$
$$F \rightarrow a \mid (E)$$

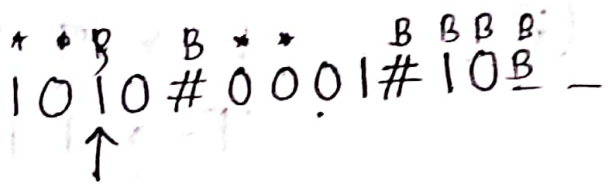
Ans 8. (a)



Ans:



9. TM for logical OR



Rules:

$\delta([q_0, B, B], [B, a]) = ([q_1, a, B], [*, a], R)$

$\delta([q_1, a, B], [B, 0/1]) = ([q_1, a, B], [B, 0/1], R)$

$\delta([q_1, a, B], [B, \#]) = ([q_2, a, B], [B, \#], R)$

$\delta([q_2, a, B], [*, 0/1]) = ([q_2, a, B], [*, 0/1], R)$

$\delta([q_2, a, B], [B, b]) = ([q_3, a, b], [*, b], R)$

$\delta([q_3, a, b], [B, 0/1/\#]) = ([q_3, a, b], [B, 0/1/\#], R)$

$\delta([q_3, a, b], [B, B]) = ([q_4, B, B], [*B, a \text{ OR } b], L)$

$\delta([q_4, B, B], [B, 0/1/\#]) = ([q_4, B, B], [B, 0/1/\#], L)$

$\delta([q_4, B, B], [*, 0/1]) = ([q_5, B, B], [*, 0/1], L)$

$\delta([q_5, B, B], [*, 0/1]) = ([q_5, B, B], [*, 0/1], L)$

$\delta([q_5, B, B], [B, \#]) = ([q_6, B, B], [B, \#], L)$

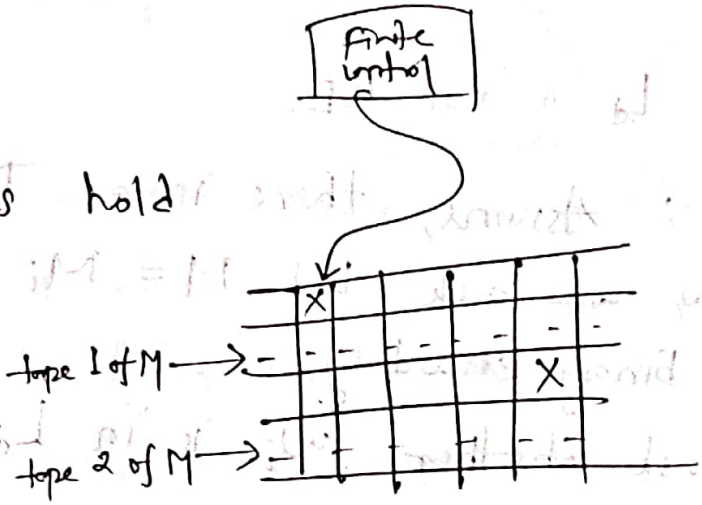
$\delta([q_6, B, B], [B, 0/1]) = ([q_6, B, B], [B, 0/1], L)$

$\delta([q_6, B, B], [*, 0/1]) = ([q_0, B, B], [*, 0/1], R)$

No other rule is needed as there is no carry.

10. (i) True.

To simulate a multitape TM M with k tapes, a single tape TM with 2k tracks and state storage can be used.



Here, half of the tracks hold tape content of M and others hold head marker positions.

(ii) False.
No, a DTM may need exponential time to simulate a NDTM.

(iii) True.

A DTM needs $O(n^3)$ moves to simulate n moves of conventional computer if multi-tape is allowed. For single tape multitrack TM, it needs $O(n^6)$ moves. Both are polynomials $P(n)$ of n.

11. i) L_d: The set of all strings w_i such that M_i, the Turing machine whose binary encoding is the same as w_i, does not accept w_i.

ii) L_u: The set of pairs (M, w) where the machine coded by M accepts the string w.

iii) Halting Problem: The set of pairs (M, w) ~~for~~ so that the Turing machine M either accepts w or halts on w without accepting.

Here, L_d is not RE.

Proof: Assume, there is a TM M so that $L(M) = L_d$

Now, ~~we ask~~ let $M = M_i$ ~~or the~~ as per the binary encoding.

We ask whether w_i is in L_d.

If $w_i \in L_d$, M_i ~~cannot~~ accepts w_i, but that contradicts by definition of L_d.

if $w_i \notin L_d$, M_i does not accept w_i but that contradicts by definition of L_d

So, no such TM exists

So, L_d is ~~not~~ RE.

12. We know ~~that~~ ~~max~~-clique, is NP-complete.

Because, ~~Max-Clique $\leq_p$ Independent set $\leq_p$ SAT~~

~~Now, we show, subgraph Isomorphism $\leq_p$ Max-Cli.~~

SAT $\leq_p$ Independent-set $\leq_p$ Max-Clique

Now, we show, Max-Clique $\leq_p$ Subgraph Isomorphism

Say we can solve, subgraph isomorphism in polytime

Now, for a non-clique graph instance, and a value k,

Let $G_1 = G$, $G_2 = K_k$ where, K_k is the complete graph with k nodes.

If G_2 is a subgraph of G_1, G has a non-clique. And the inverse is also true.

So, Non-Clique $\leq_p$ Subgraph Isomorphism

So, it is NP-complete.

13. (a) i) P: ~~The~~ problems that can be solved by a deterministic Turing machine in polynomial moves $p(n)$ where $n = |w|$ = size of input.

ii) NP: ~~The problems that~~ A language L is in NP if there is NDTM ~~M~~ M and a polynomial time complexity $\leq p(n)$ such that, when M is given a input of length n, there are no sequences of more than $T(n)$ moves of M. These are the problems checkable in polytime.

iii) PSPACE: A language L is in PSPACE if L is a language decided by a polynomial space DTM.

iv) NPSPACE: A language L is in NPSPACE if L

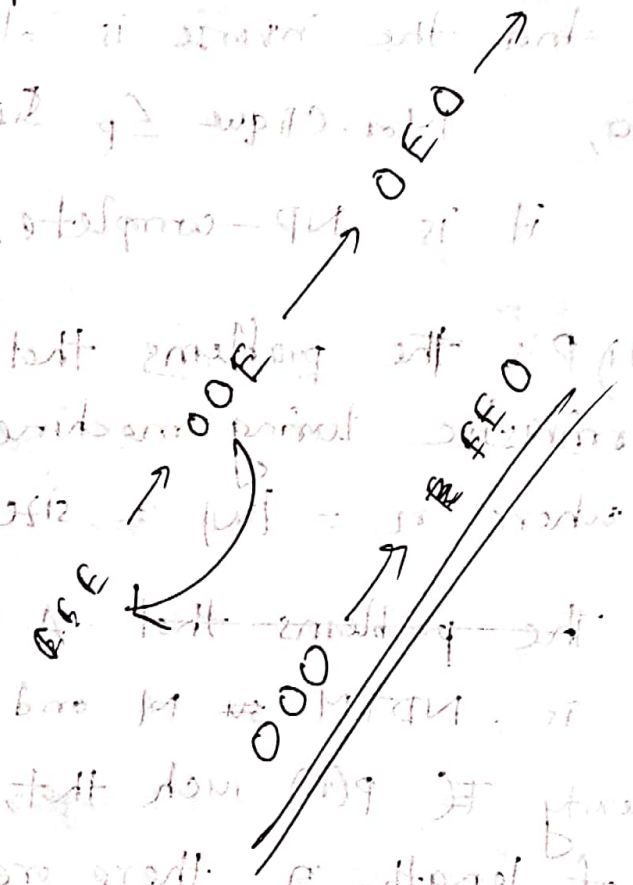
is a language decided by a polynomial space NDTM.

(b) i) Widely believed → (and proven)

i) $P \subseteq NP$

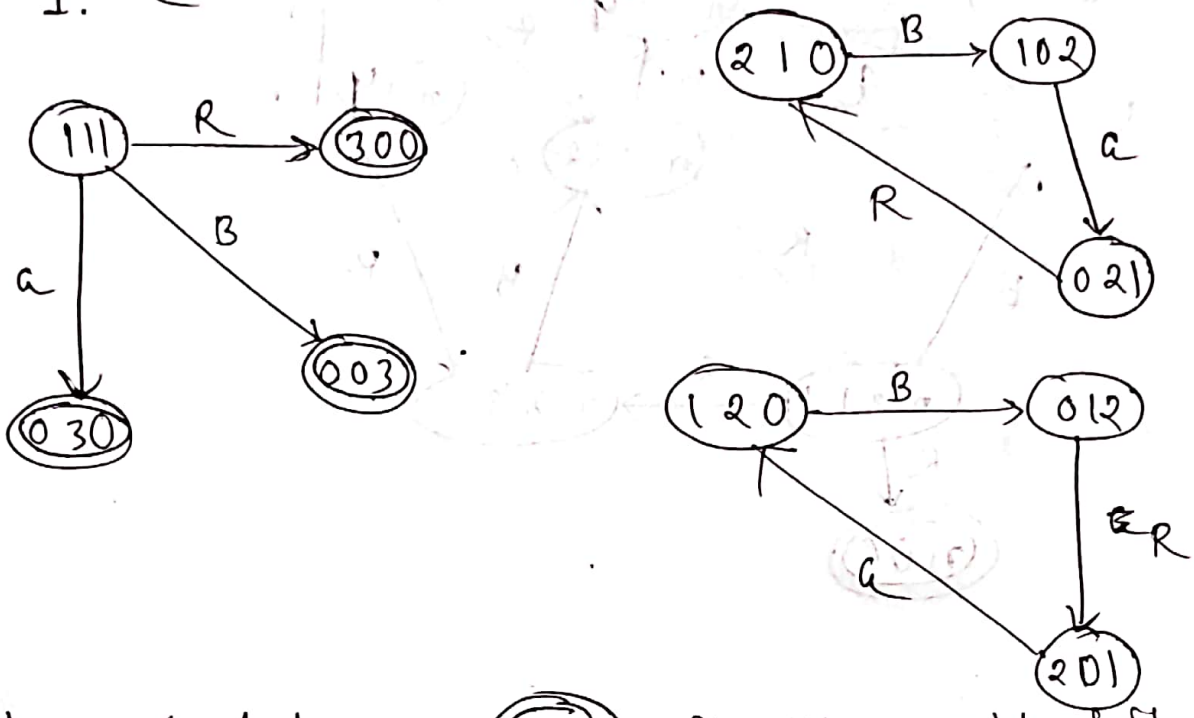
ii) $PSPACE = NPSPACE$

iii) $NP \subseteq PSPACE$



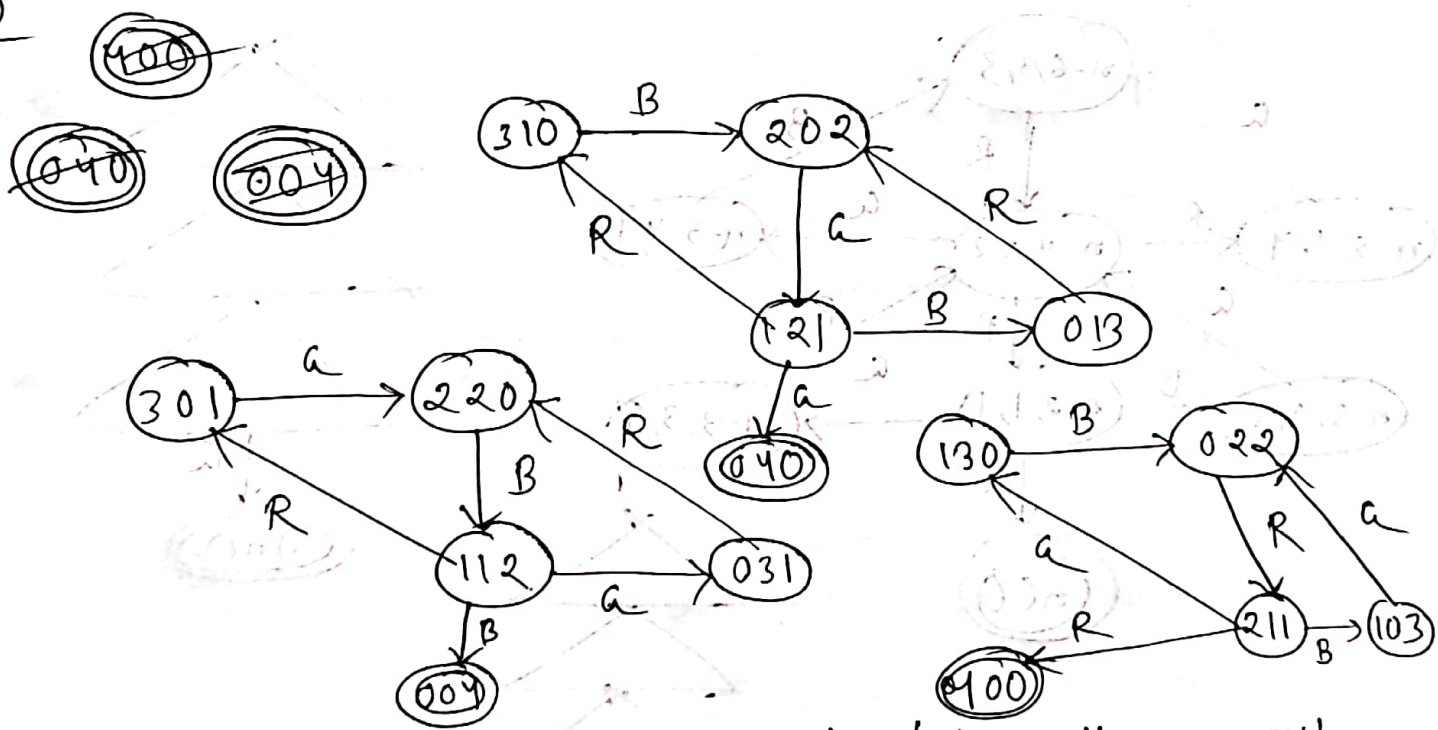
2016-17

1. (a)



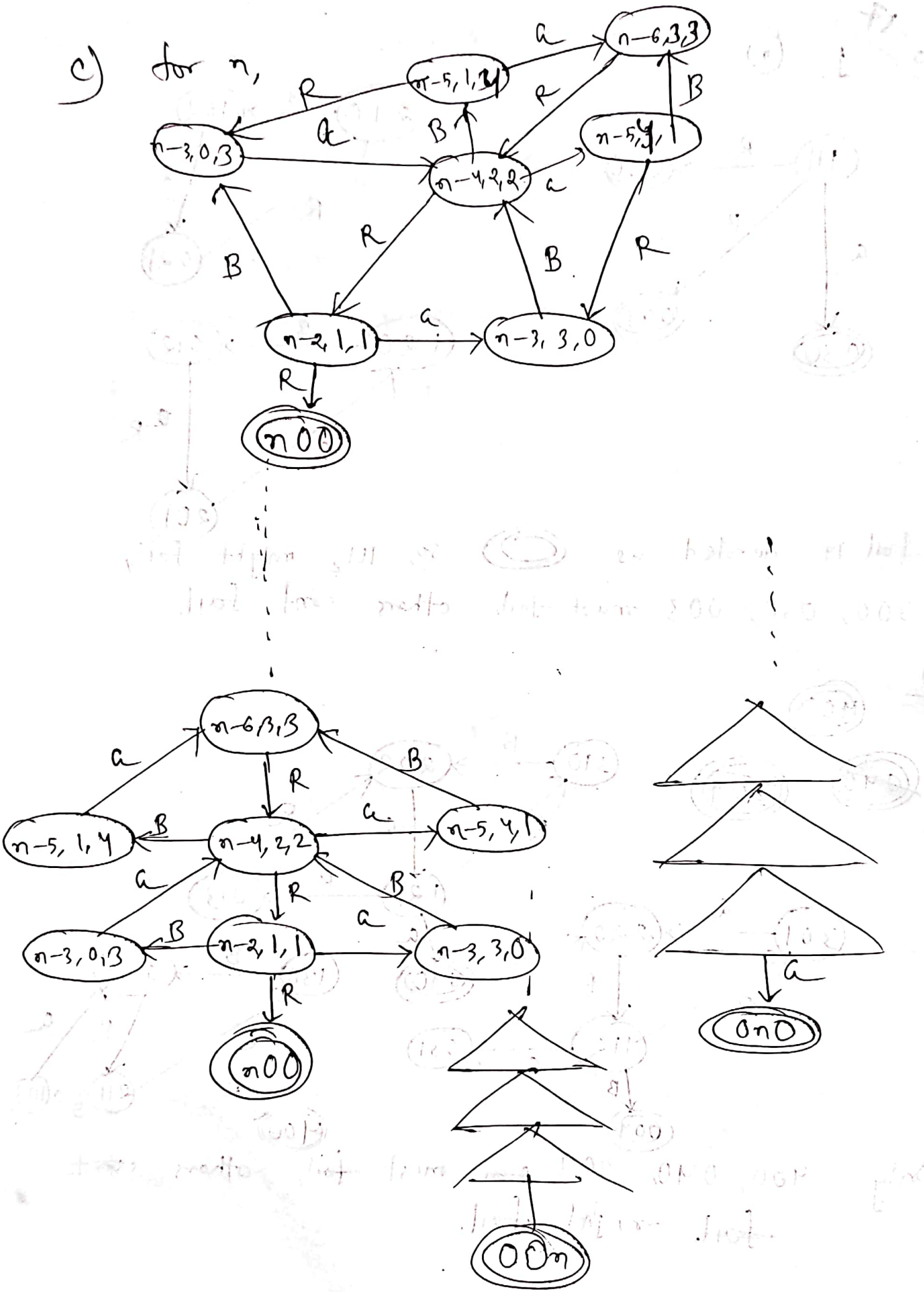
Fail is denoted as ◯ so, 111 might fail, 300, 030, 003 must fail, others cant fail.

(b)



Only 400, 040, 004 ~~can~~ must fail, others ~~cant fail.~~ might fail.

c) for n,

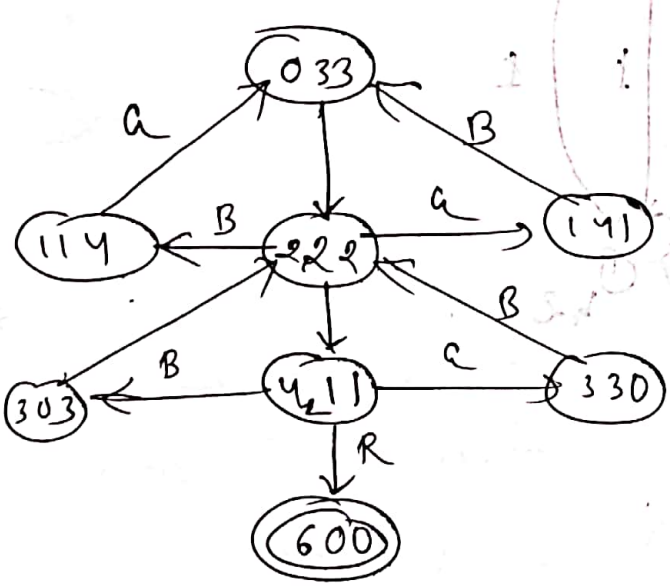


It seems, the state drograms is always a combination of three beautiful Christmas trees ( ), this is cool. and cute.

But there must be clearer invariants that prove whether or not all states belong ~~to~~ to these in cases other than $n = 3$.

Furthermore, it is clear that, all states that are in Christmas trees can reach fail states. So, they all might fail. Others can't fail.

(d)



But when $n = 6$,
this node (222)
corresponds to all trees.

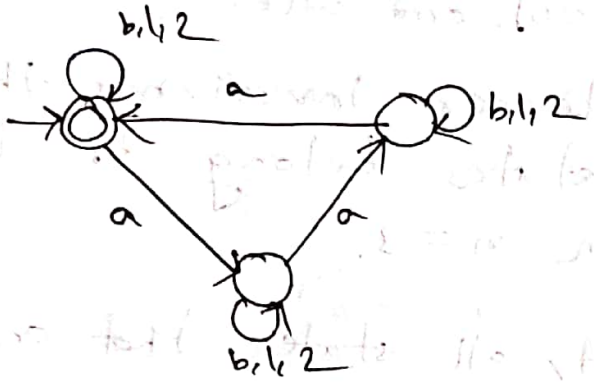
So, all of these trees are in turn connected.

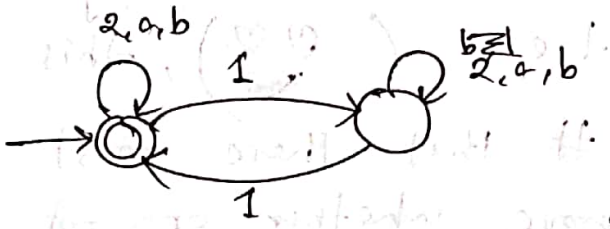
There aren't any other states.

So, all might fail.

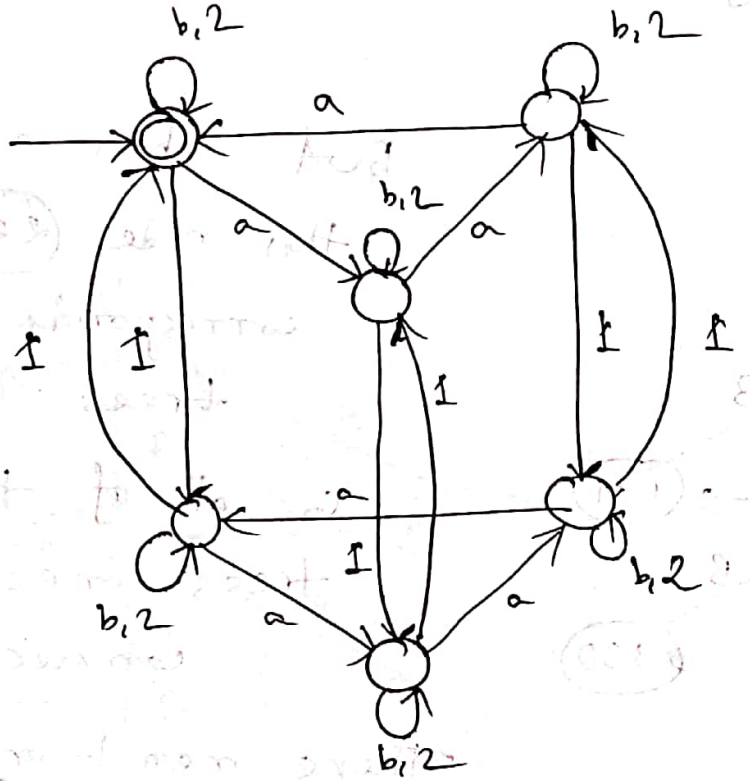
600, 060, 006 must fail.

2 (a) Two individual DFAs,

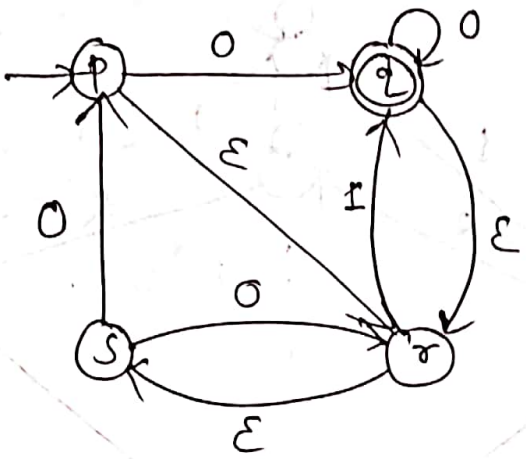




Taking intersection,

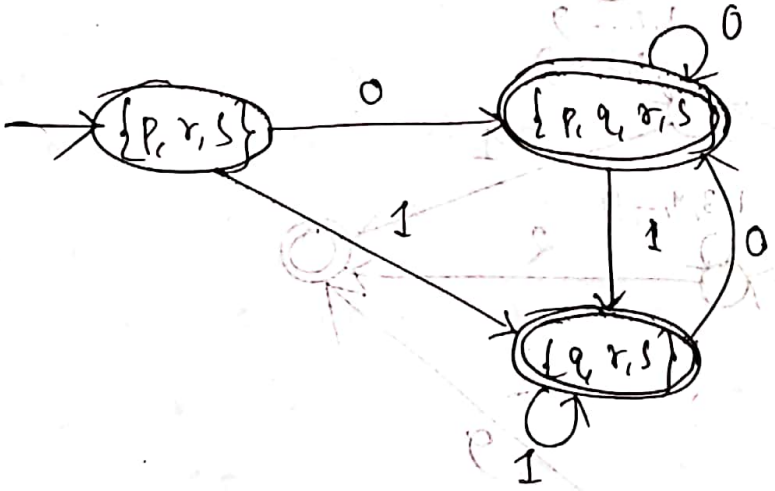


(b)



$E(p) = \{p, r, s\}$ $E(q) = \{q, r, s\}$ $E(r) = \{r, s\}$

$E(s) = \{s\}$



(c) $\emptyset^* = \{\varepsilon\}$

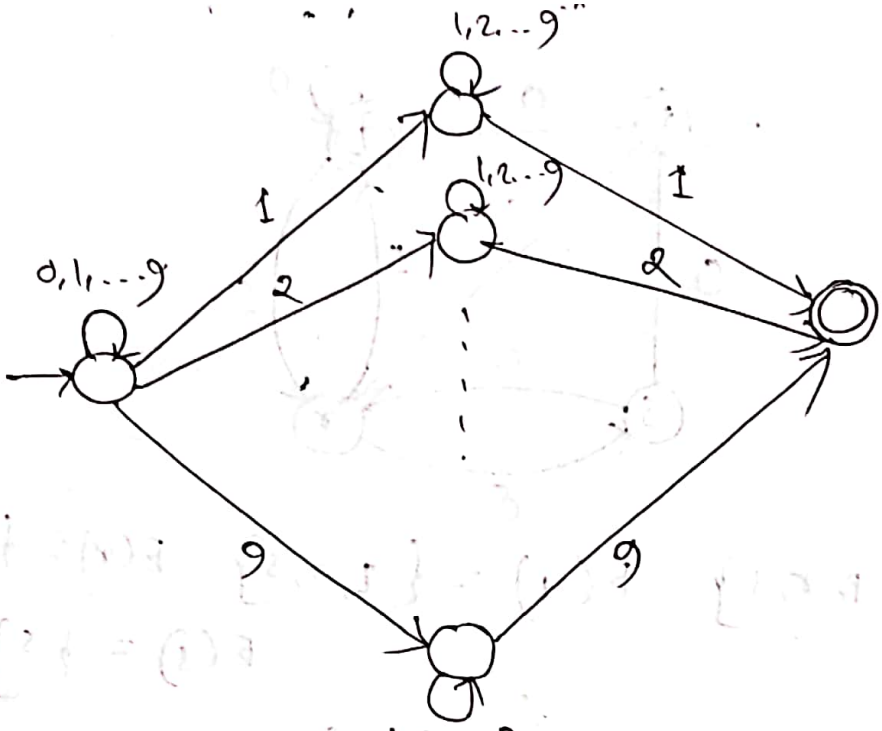
Taking Kleene Star means concatenating ~~too~~ 0 or more strings from a set.

Now, when I take 0 strings from a null set and concatenate, I effective get an empty string.

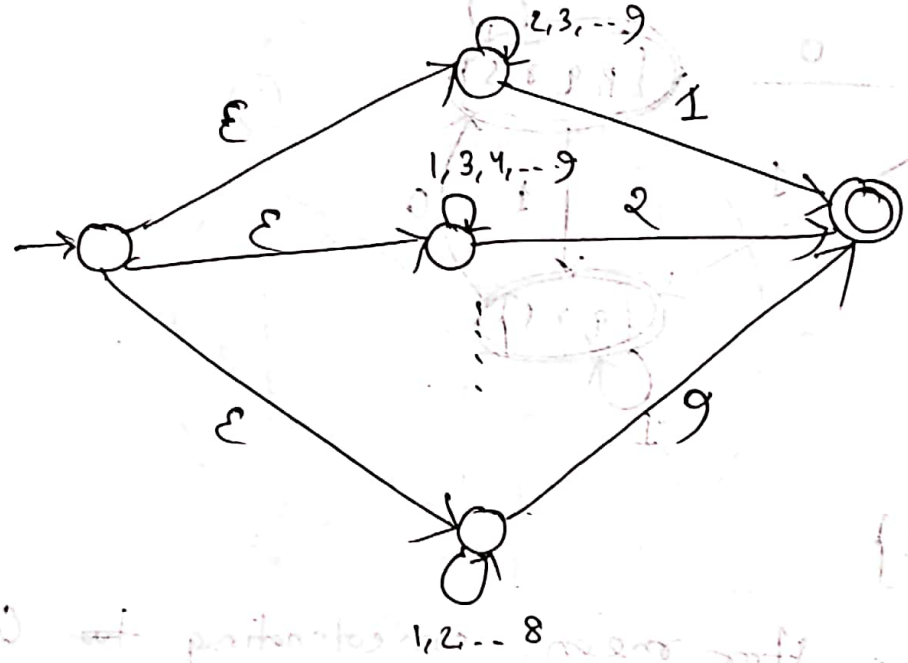
And no more than 0 strings can be taken from null set.

$\therefore \emptyset^* = \{\varepsilon\}$

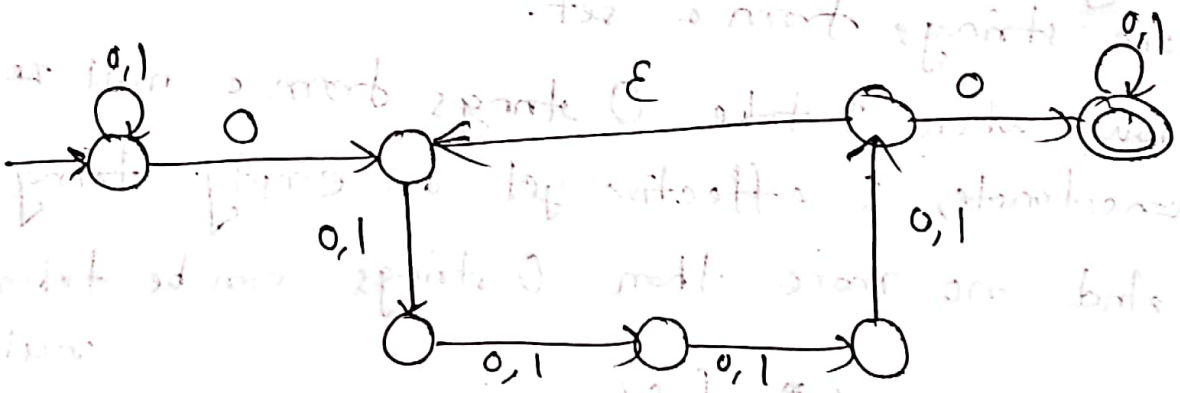
9. (a) (i)



(ii)



(iii)



(b) i) $(\varepsilon+1)(01)^*(\varepsilon+0)$

ii) $\Sigma^*(a\Sigma^*b + b\Sigma^*a)\Sigma^*$ where, $\Sigma = a+b+c$

iii) $(\varepsilon+1)(00^*1)^*0^*(11^*0)^*1^*$

4. (a) The idea will not work when, the start state may have ~~a~~ other functions in the ~~D~~ NFA. Say, while operating, the NFA reaches the original ~~start~~ state for some reason but it's not ment to be on accept state.

However, by Hasty's design, these unwanted strings will also be accepted.

(b) Let N_1 be an NFA for a regular language L_1.

Let $N_1 = (Q_1, \Sigma_1, \delta_1, q_1, f_1)$, N_2 be ~~a~~ $L_2 = L(N_2)$,

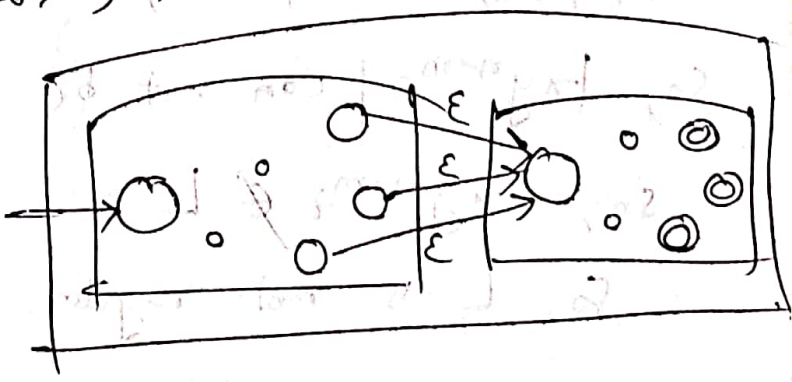
$N_2 = (Q_2, \Sigma_2, \delta_2, q_2, f_2)$

Now, let $N = (Q, \Sigma, \delta, q_0, F)$,

$Q = Q_1 \cup Q_2$

$\Sigma = \Sigma_1 = \Sigma_2$

$q_0 = q_1$, $F = f_2$



Now, let's define δ,

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \notin F_1 \text{ and } q \in Q_1 \\ \delta_1(q,a) & \text{if } q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_2\} & \text{if } q \in Q_1 \text{ and } a = \varepsilon \\ \delta_2(q,a) & \text{if } q \in Q_2 \end{cases}$$

(c) $L = \{ 1^n \mid n = \text{prime number} \}$

~~Let p be~~ Assume L is regular.

Let p be the pumping length

Let $n > p+2$ be a prime.

Now, $w = 1^n$, n = prime, $n > p+2$

Now, $|xy| \leq p$, $|y| > 0$ ~~Let $|y| = m$~~

$$\therefore |xy^{n-m}z| = |xz| + |y^{n-m}| \geq |xz| + (n-m)|y|$$

$$= n - m + (n-m)m$$

$$= (m+1)(n-m)$$

Now, $m > 0$ so, $m+1 > 1$,

$n > p+2$, $m \leq p$ so, ~~$n > m$~~ $n - m > 1$

So, $|xy^{n-m}z|$ can not be prime.

So, $xy^{n-m}z \notin L$

So, L is not regular.

5. (b) i) $S \rightarrow PSP \mid 0 \mid 1$

$P \rightarrow 0 \mid 1$

ii) $S \rightarrow P1P1P1P$

$P \rightarrow 0P \mid 1P \mid \varepsilon$

iii) $S \rightarrow PSP \mid 0$

$P \rightarrow 0 \mid 1$

b)

10011 B B

↑

$\delta(q_0, \#/1/0) = (q_0, \#/1/0, R)$

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

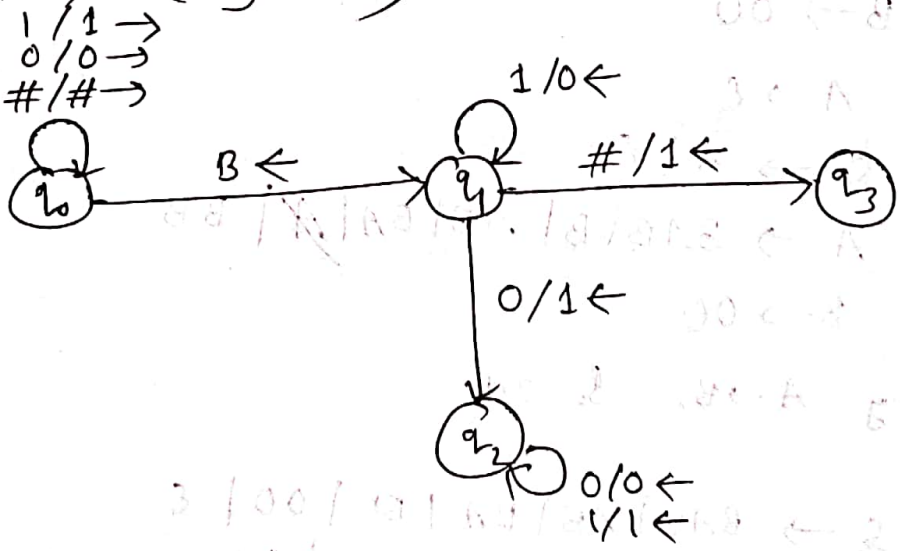
$\delta(q_1, 1) = (q_1, 0, L)$

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

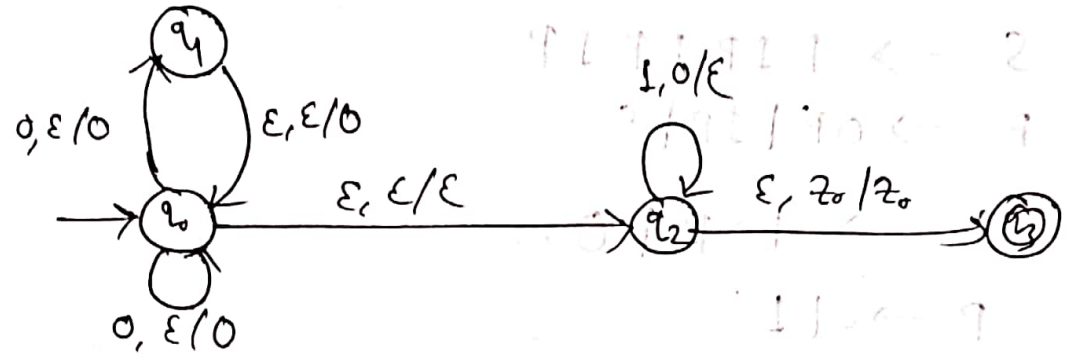
$\delta(q_2, a) = (q_2, a, L)$; $a = 0/1$

$\delta(q_1, \#) = (q_3, 1, L)$

Diagram:



6(a) $L = \{ 0^n 1^m \mid n \le m \le 2n \}$



Accepts 00111, Rejects 0011111

(b) A appears on RHS

S_0, $S_0 \to A$

$A \to BAB \mid B \mid \varepsilon$

$B \to 00 \mid \varepsilon$

— No useless variable

— Removing $B \to \varepsilon$

$S_0 \to A$

$A \to BAB \mid B \mid \varepsilon \mid AB \mid BA \mid A$

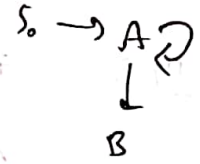
$B \to 00$

— Removing $A \to \varepsilon$

$S_0 \to A \mid \varepsilon$

$A \to BAB \mid B \mid AB \mid BA \mid A \mid BB$

$B \to 00$



— Removing $A \to B$, $S_0 \to A$

$S_0 \to BAB \mid AB \mid BA \mid BB \mid 00 \mid \varepsilon$

$A \to BAB \mid AB \mid BA \mid BB \mid 00$

Now, fixing others

$S_0 \rightarrow$ ~~BT~~ $BV \mid AB \mid BA \mid BB \mid UU \mid \varepsilon$

$A \rightarrow BV \mid AB \mid BA \mid BB \mid UU$

$V \rightarrow AB$

$U \rightarrow 0$

this can't be removed

(c)

B B
B 0 0 1 0 1 0 1 1 B

$\delta([q_0, B], [*, a]) = ([q_0, B], [*, a], R)$

$\delta([q_0, B], [B, a]) = ([q_1, a], [*, a], R)$

$\delta([q_1, a], [*, 0/1]) = ([q_1, a], [*, 0/1], R)$

$\delta([q_1, a], [B, a]) = ([q_1, a], [B, a], R)$

$\delta([q_1, a], [B, \bar{a}]) = ([q_2, B], [*, \bar{a}], L)$

$\delta([q_2, B], [B/*, 0/1]) = ([q_2, B], [B/*, 0/1], L)$

$\delta([q_2, B], [B, B]) = ([q_0, B], [B, B], R)$

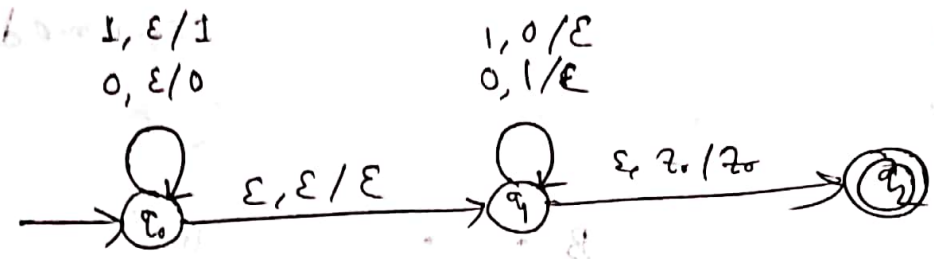
~~$\delta([q_2, B], [$~~

$\delta([q_0, B], [B, B]) = ([q_3, B], [B, B], L)$

↓ Accepting

7. (a) Possible. We can just simulate BPS on a TM.

(b)



(c) let pumping length = p

let $w = a^p b^p c^p$

$\underbrace{a\,a\,\ldots\,a}_{p}\,\underbrace{b\,b\,\ldots\,b}_{p}\,\underbrace{c\,c\,\ldots\,c}_{p}$

$w = uvwxy$, now, $|vwx| \leq p$. So, it can't contain all of a, b, c. It can at most contain two of characters.

Thus, in uv^iwx^iy, for $i \neq 1$, the number of characters will not match.

So, By pumping lemma for CFL,

B is not context-free.

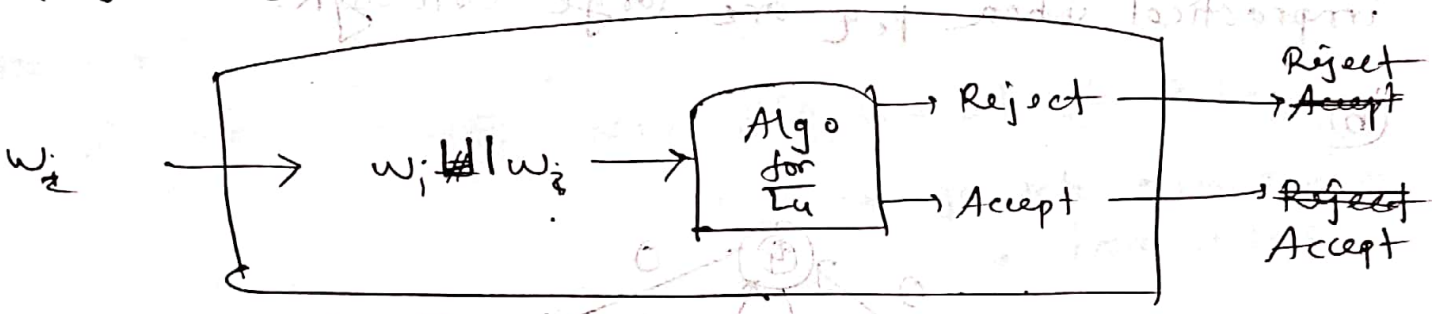
8. (a) $A_{TM} = L_u$

Here, undecidable means not recursive.

Proof : Say if, we assume L_u is recursive.

Then $\overline{L_u}$ is also recursive.

Now, we can use the TM for $\overline{L_u}$ to design a TM for L_d.



Reduction of L_d to $\overline{L_u}$

(b) $S \rightarrow 00S00 \mid 01S10 \mid 10S01 \mid 11S11 \mid \varepsilon$

(c) EXPR $\rightarrow$ TERM | EXPR + TERM

TERM $\rightarrow$ FACTOR | TERM * FACTOR

FACTOR $\rightarrow$ (EXPR) | a