Vishakha Agrawal- 2023101040

21-8-24

AT- Theory Assignment I

Ans 1. Given:

An NFA with 5 states, $n = 5$

Size of alphabet, Σ, $p = 3$

Solution:

For each symbol in the alphabet, a state can transition to any of the 5 states including itself.

∴ # transitions for all 5 states for 1 input symbol $= n \times n = 25$

∵ $p = 3$, such transitions are possible for all input symbols.

∴ Total possible # transitions: (excluding epsilon transitions)

$p \times 25 = 3 \times 25 = 75$ transitions.

ϵ transitions dont consume any input symbols. They allow transitions to another (or same) state without reading any symbol from the input tape.

Each state can have epsilon transitions to any of the 5 states.

Thus, there are $5 \times 5 = 25$ epsilon transitions

∴ The maximum # of transitions that the given NFA can have is:

$$75 + 25 = 100$$

2 Given:

A FST defined by $(Q, \Sigma, \Gamma, \delta, S)$

Σ = set of unicode characters

We have to construct a transducer that outputs the part of the program that is not connected.

Solution:

Let c represent all / any unicode characters except %.

We design the transducer is the following way:

Current state	Input	Next State	Output
q_0	%	q_1	ϵ
q_0	c	q_0	c
q_1	%	q_2	ϵ
q_1	c	q_0	%c
q_1	ϵ		%
q_2	%	q_3	ϵ
q_2	c	q_2	ϵ
q_3	%	q_0	ϵ
q_3	c	q_2	ϵ

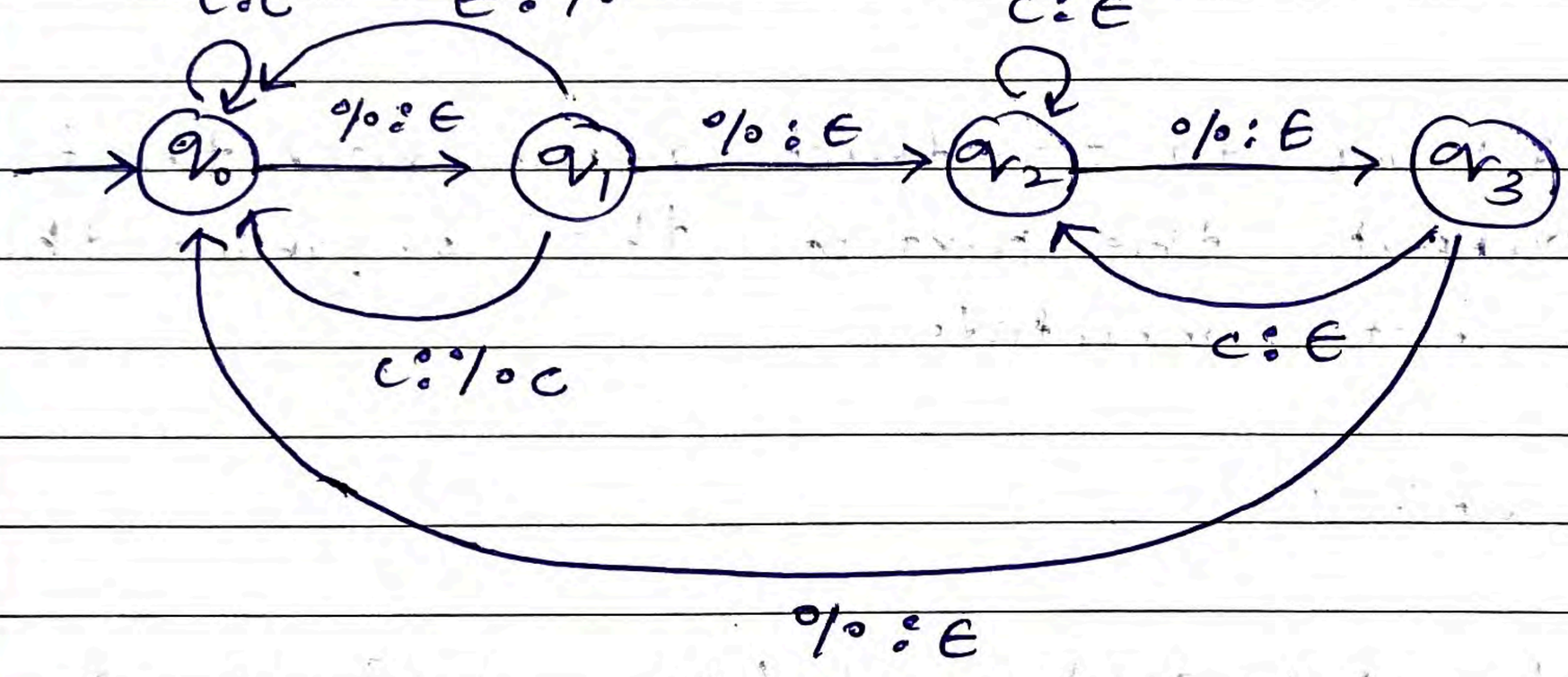
where:

q_0 represents the start state that reads code.

q_1 represents the start of a comment

q_2 represents the state reached on.

determining that a comment has started
q_3 represents the potential end of a comment.



Reasoning:

q_0 starts reading code, if it encounters a % ~~on the~~ in the code, it could be the start of a comment. It transitions to q_1
If it encounters any other character 'c', 'c' is outside a comment, so it outputs 'c'.

If q_1 encounters another % (after q_0), this confirms start of a comment.
q_1 transitions to q_2. If it encounters any other character 'c', this means that there is no comment. Both % & 'c' are outputted. q_1 transitions to q_0 to scan for any potential start of a comment.
If q_1 reads a %, this could be the potential start of end of a comment,

if transitions to q_3. If it encounters any other character, this means that the comment has started, any thing within a comment is not outputted. It stays in q_2.
If q_3 encounters a '%', it signifies end of a comment. q_3 transitions to q_0 to start scanning for the next piece of code.
If q_3 encounters a 'c', this means that the comment has not ended although a potential marker for end of comment was detected. Nothing is outputted. q_3 transitions to q_2 to scan for potential end of comment.

ε : % transition from $q_1 \rightarrow q_0$
after potential start of comment

If a potential end of comment is detected, Λ but the code ends without any further characters, all characters including both % must be outputted as none of the characters are a part of a comment or mark the start or end of a comment.

Example:

```
%%print("cat")%%%Horse%
```

(%% : comment end)

Output:

```
%Horse%
```

After the comment ends, q_3 transitions to q_0 q_0 on reading %, transitions to q_1, q_1 on reading H, transitions to q_0 and outputs %H.

Subsequent iterations on q_0 give the output % Horse

q_0 detects a % and transitions to q_1 - The code ends here, But % must be outputted. Thus, nothing is read. ~~from the in~~ & a % is outputted resulting in

% Horse %

Ans 3. DFA Minimization:

~~Transition table:~~

$Q = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$

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

$q_0 = \{0\}$

$F = \{6\}$

Transition Function:

	0	1
0	1	8
1	2	4
2	8	3
3	2	8
4	5	6
5	8	6
6	7	8
7	7	7
8	8	8

0 Equivalence:

{0, 1, 2, 3, 4, 5, 7, 8} . {6}

1 Equivalence:

{0, 1, 2, 3, 7, 8}

{4, 5} {6}

Explanation: 3 and 4 are not 1-Equivalent. On getting input 1, 3 transitions to 8 while 4 transitions to 6. Similarly 3 and 5 are not 1 Equivalent.
4 and 5 are ~~not~~ 1 equivalent. On getting input 0 they go to q_5 & q_8 (in first set). On getting input 1, they go to q_6 (second set)

2 Equivalence:

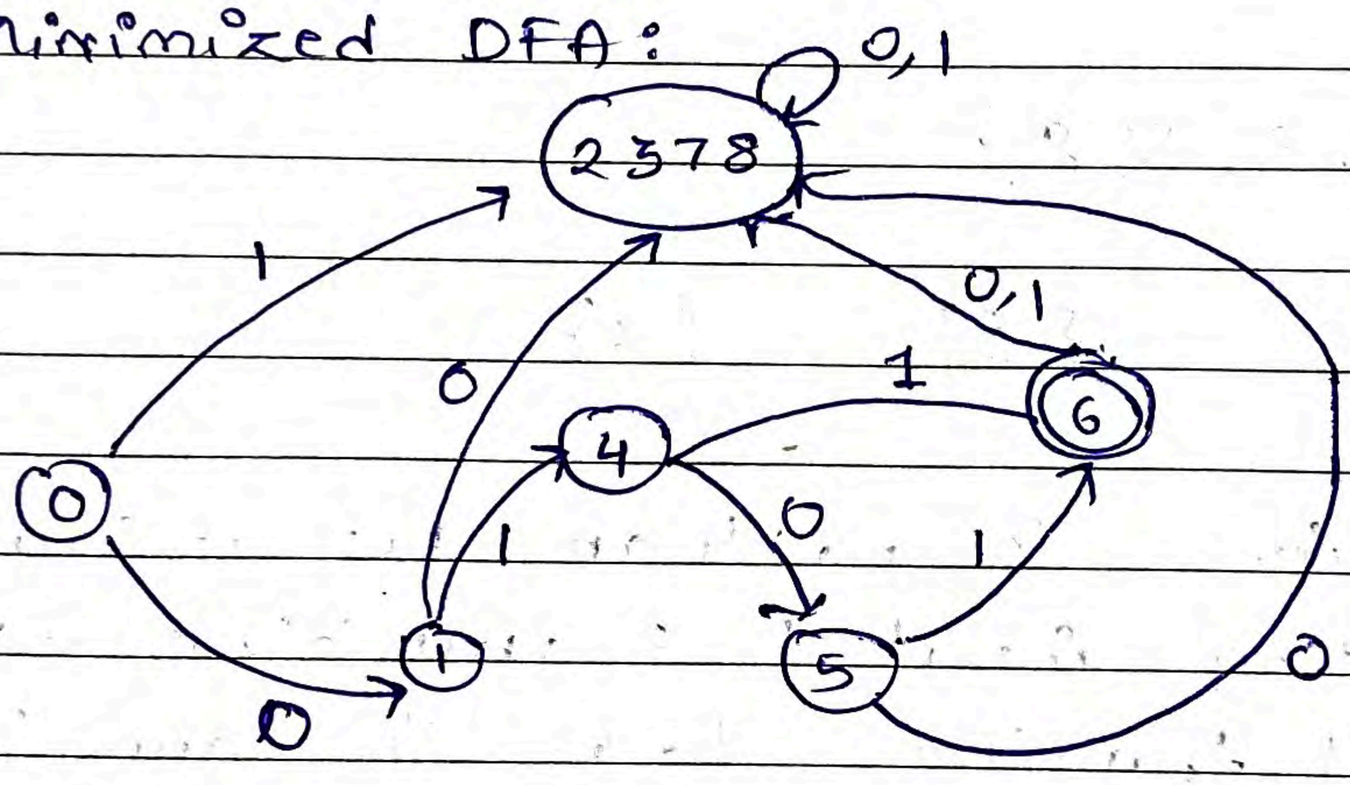
{0} {1} {2, 3, 7, 8}
{4} {5} {6}

Explanation: 0 is not equivalent to 1 On getting input 1, state 1 goes to 4 while state 0 goes to 8. 2 is not equivalent to 1 as on getting 1, 1 & 2 go to 4 and 5 respectively which do not belong to the same set.
4 and 5 are not equivalent. On getting input 0, they go to 5 and 8 respectively which are in different sets.

3 equivalence :

{0} {1} {2, 3, 7, 8} {4} {5} {6}

Minimized DFA :



Transition Function :

	0	1
0	1	2378
1	2378	4
2378	2378	2378
4	5	6
5	2378	6
6	2378	2378

Thus we have the minimized DFA with 6 states.

s4. We have to define an NFA that accepts strings:

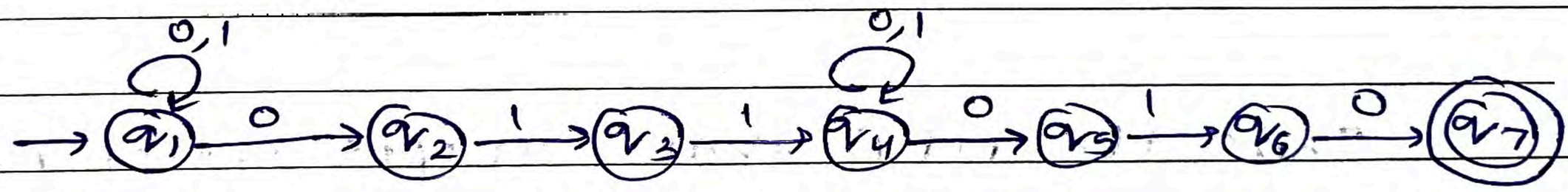
i) That either end in 010, & have 011 somewhere in preceeding or

ii) end in 101 & have 100 somewhere in preceeding.

Let A_1 be the language that accepts strings as described in i)

Let A_2 be the language that accepts/ contains strings as described in ii)

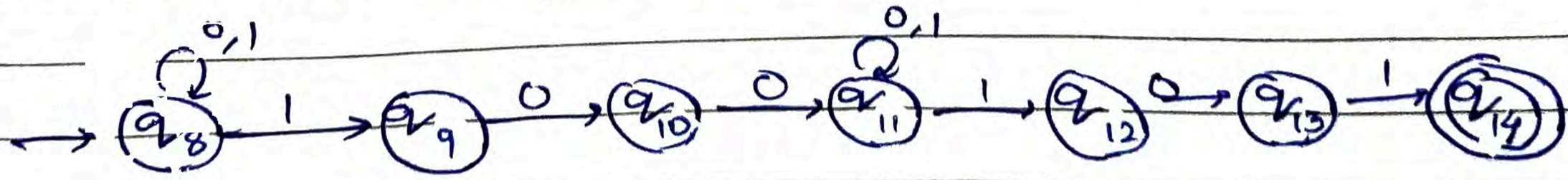
NFA that accepts A_1:



Reasoning:

The string can begin with any combination of symbols and can have any combination of symbols on reaching q_4. However it must have 011 somewhere in preceeding before it ends in 010 on reaching q_7.

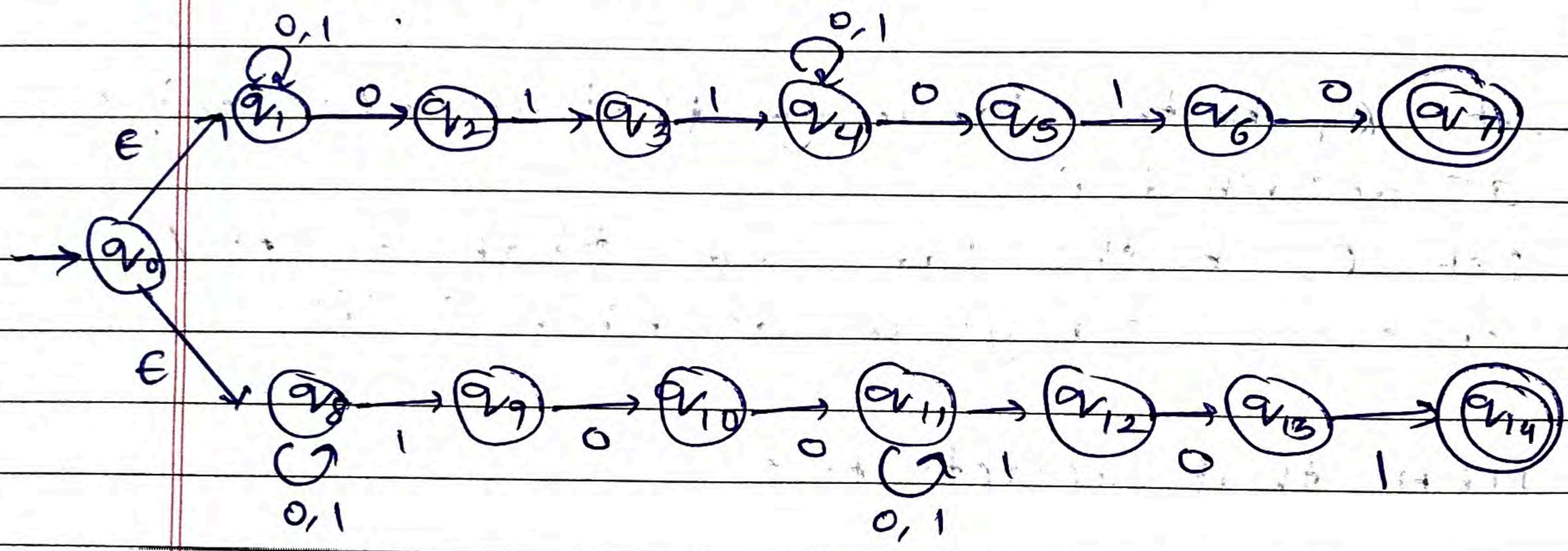
NFA that accepts A_2:



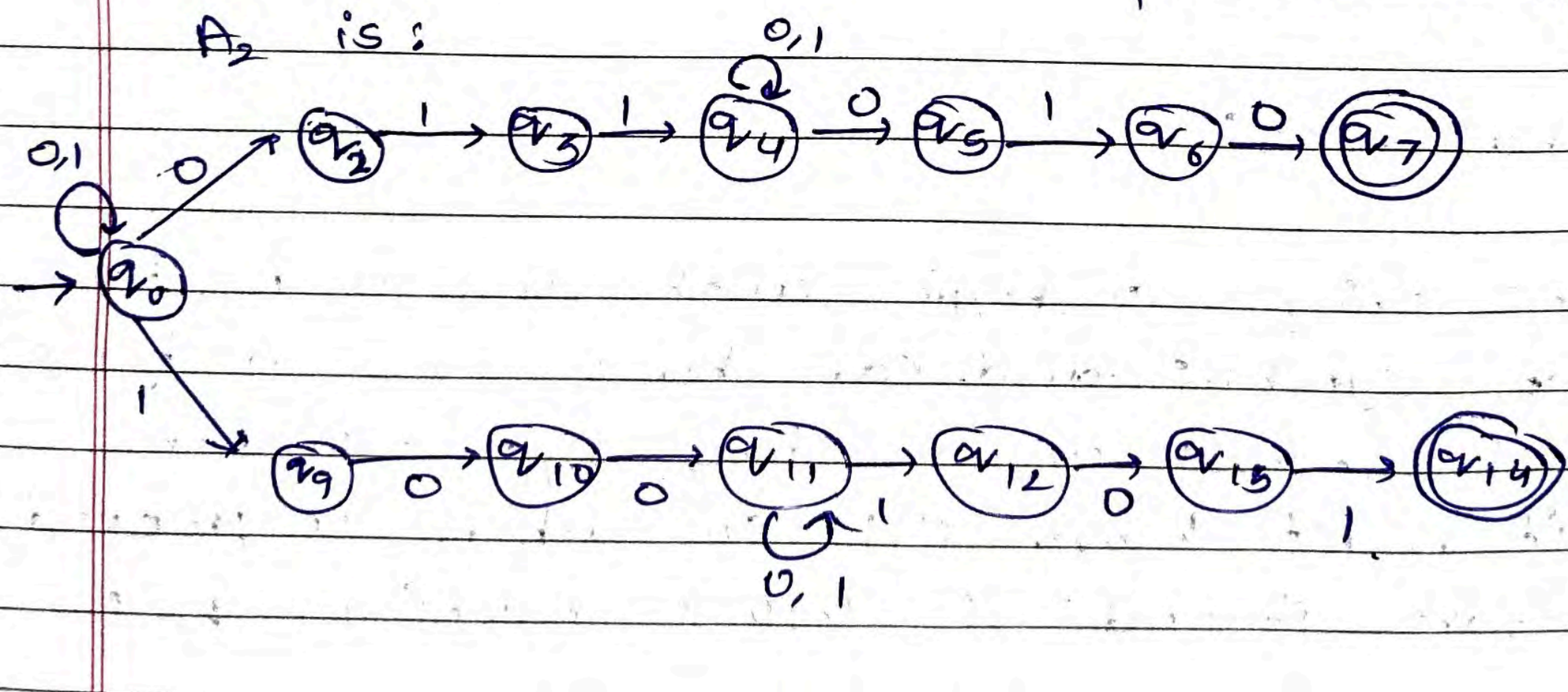
∴ A_1 & A_2 are regular languages, $A_1 \cup A_2$ is also regular.

∴ It is possible to construct a new NFA that accepts either of the two languages A_1 or A_2.

Let q_0 be the new start state of the new NFA N.



Thus, the NFA that accepts either A_1 or A_2 is:



Qs5. (i) The given finite state automata accepts the following language:

$$L = \{ a(a+b)^* ba \}$$

Reasoning:

The automata only accepts strings that:
The string must start with an a
followed by any sequence of a's and b's
which includes:
an optional segment of 'a' that can lead back to 3 and repetitions of b in state 4.
Finally, the string must end with a ba.

Examples:
aba, abba, abbabbaba, abbbbba, abbaba

(ii) If $\Sigma = \{a, b\}$

We have to find a regular expression for language L of all strings in Σ^* with exactly one occurence of aaa.

Solution:
Regular expression for the language L:

$$(b + ab + aab)^* aaa(b + ba + baa)^*$$

Reasoning:

Before aaa:

$(b + ab + aab)^*$

This allows for any combination or sequence of a's and b's that dont contain aaa. They prevent consecutive aaa's occuring, no matter in what order they are taken.

After aaa:

$(b + ba + baa)^*$

This accounts for all possible strings with a & b that ensure that any combination taken any number of times will not lead to aaa.

Ans 6. Given:

A is any language.

$A_{1/3-1/3} = \{x \mid \text{for some } y, |x| = |y| = |z| \; \& \; xyz \in A\}$

RTP:

If A is regular, $A_{1/3-1/3}$ is not necessarily regular.

Proving that $L = \{0^n 2^n \mid n \geq 0\}$ is not regular using pumping lemma.

(By contradiction)

Let L be regular, its pumping length be p.

Let $s = 0^p 2^p \in L$, $|s| = 2p$

Let us try splitting s into xyz st.

$|xy| \leq p$ & $|y| > 0$

$\therefore |xy| \leq p$, y can only contain 0s.

Let $y = 0^k$

st $x = 0^m$ $\quad 0 \leq m < p$

$y = 0^k$ $\quad 1 \leq k \leq p-m$

$z = 0^{p-(m+k)} 2^p$

Let us pump y and see if $xy^i z \in L$ for all values of i.

$i = 0$

The resulting string is $xz = 0^m 0^{p-(m+k)} 2^p$

$= 0^{p-k} 2^p$

This string has fewer 0s than p and p 2s. $\therefore$ It does not belong to L.

$i = 2$

Resulting string is $xy^2z = 0^m 0^{2k} 0^{p-(m+k)} 2^p$

$= 0^{p+k} 2^p$

This string has a greater number of 0's than p & p 2's. $\therefore$ It does not belong to L.

Thus for values of $i \neq 1$, the pumped string does not belong to L ∴ Our assumption that L is regular is false.

∴ L is not regular.

Proving that if A is regular $A_{1/3-1/3}$ is necessarily not:

Proof by contradiction:

Let A be a regular language. For the sake of contradiction, let $A_{1/3-1/3}$ be regular.

Let $L = \{0^n 1^m 2^p \mid n, m, p \geq 0\}$ be a regular language.

Let us take $w = 0^n 1^m 2^n \in A$. If we remove the middle thirds portion of this string, we get all strings of the form $0^n 2^n$.

Let $A^{1/3-1/3}$ be a language that contains all strings obtained by removing the middle thirds of strings in A.

$$A^{1/3-1/3} = \{0^n 2^n \mid n \geq 0\}$$

Now, we proved above that the language $\{0^n 2^n \mid n \geq 0\}$ is not regular.

However, we assumed that $A^{1/3-1/3}$ (the set of all strings with their middle thirds removed) is regular. We showed that $A^{1/3-1/3}$ is not regular.

$= \{0^n 2^n \mid n \geq 0\}$

This contradicts our assumption that $A^{1/3-1/3}$ is regular. $\therefore$ Our assumption is false.

$\therefore$ $A^{1/3 \cdot 1/3}$ is not necessarily regular even if A is regular.

57. Given:

M_1 & M_2 are DFAs with k_1 & k_2 states respectively

$U = L(M_1) \cup L(M_2)$

$U \neq \phi$

RTP:

$\exists\ s \in U$ st $|s| < \max(k_1, k_2)$

Proof: (By contradiction)

Let us assume for the sake of contradiction that every string $s \in U$ is st $|s| \geq \max(k_1, k_2)$.

$\therefore$ $|s| \geq \max(k_1, k_2)$, $|s| \geq k_1$ if M_1 accepts s & $|s| \geq k_2$ if M_2 accepts s

Let w be the shortest string in U.

Without loss of generality, let us ~~assum~~ consider the case where w is accepted by M_1.

$|w| \geq k_1$ (Based on our assumption that $|s| \geq \max(k_1, k_2)$).

If M_1 accepts w, it must have traversed through $w+1$ states.

$\therefore |w| \geq k_1$

$\Rightarrow |w| + 1 > k_1$

By the pigeonhole principle, $\because M_1$ has k_1 states, but $|w|+1 > k_1$, atleast one state out of the k_1 states must be traversed more than once during the execution of w. This repetition of a transition during execution must lead to a cycle.

Let q be the repeated transition.

Let $w = xyz$ where x leads to the first visit of q, y is the part of w that is executed in the cycle, z is the remaining part of w that leads to an accepting state.

$\therefore$ Executing y leads back to q, from q z goes to an accepting state, xz will also lead to an accept state as y can be pumped down or pumped up

and the string will still be accepted.
i.e. A string $v = xz$ where $|v| < |w|$ will also be accepted by M_1.

∴ $xz \in L(M_1) \subseteq U$ and $|xz| < |w|$

However, this contradicts our assumption that w is the shortest string in U. because we found a shorter string $v \in U$.

∴ Our assumption that $|w| \geq \max(k_1, k_2)$ is false.

∴ U contains some string $|s| < \max(k_1, k_2)$.

QED //

ns8. 1. RTP:

$L = \{w \mid w$ has balanced parantheses$\}$ is not regular.

Proof by contradiction using pumping lemma:

Let us assume that L is regular. (For sake of contradiction)

Let us take a string $w \in L$ where $w = (^p)^p$, w contains p opening braces and p closing braces, $|w| = 2p$

Using the pumping lemma, we can partition

$w = xyz$ st $|xy| \leq p$ & $|y| > 0$

$\therefore |xy| \leq p$, y will only contain say k opening parantheses. Say $y = C^k$ where $k > 0$, y has k opening parantheses.

If we pump y for $i = 2$ we get

$$xy^i z = xy^2 z = xC^k C^k z.$$

Thus the resulting string contains $p+k$ opening braces and p closing braces. The number of opening parantheses is greater than the number of closing parantheses.

$\Rightarrow$ After pumping the string is no longer balanced. Our assumption (that L is regular) is false.

$\therefore$ L is not regular.

2. RTP:

$L = \{a^{n!} \mid n > 0\}$ is not regular.

Proof by contradiction using pumping lemma:

For the sake of contradiction, let us assume that L is regular.

Let p be the pumping length. Let us choose $w = a^{p!}$ as a string from the language.

$|w| = p!$

We have to divide $w = xyz$ st $|xy| \leq p$, $|y| > 0$ & $xy^i z \in L \quad \forall i \geq 0$

$\because |xy| \leq p$, y only contains some part of the as from the first p a's in w.

Let $y = a^k$ for $k > 0$ and $x = a^m$

let us pump y for $i = 0$

$xy^i z = xz = a^m a^{p!-m-k} = a^{p!-k}$

(where $xyz = a^m a^k a^{p!-m-k}$

We observe that $|xz| = p! - k$. Since $k > 0$,
($\because |y| > 0$ according to pumping lemma),
$p! - k < p!$ for $k \geq 1$.

Similarly when $i = 2$

$xy^i z = a^m a^k a^k a^{p!-m-k} = a^{p!+k}$

We observe that $|xyz| = p! + k$
$\because k \geq 1$, $p! + k > p!$

In both cases (for $i = 0$ & 2) we see that the ~~pumped~~ resultant string ~~is not of the form~~ does not have a length that is a factorial. Our assumption that L is regular is false.
$\therefore L = \{a^{n!} \mid n \geq 0\}$ is not regular

Q.E.D.

Ans 9. Given grammar:

G: R → ST | UV

T → UV | W

V → XY | Z

X → .YZ | T

To Find:
Terminals, non-terminals & start symbol in G

Solution:

→ In a CFG, non-terminals are symbols that can be expanded further or replaced by other symbols using production rules and occur on the left of the production rule.

In the given grammar, we observe that R, T, V, X are non-terminals.

→ In a CFG, terminals are symbols that cannot be expanded further. They appear only on the right side of production rules.

Here; S, U, W, Y, & Z are terminals.

→ The start symbol is typically a non-terminal on the left of the first production rule.

Thus, the start symbol of the grammar

is R.

∴ G for G:

Non-Terminals : R, T, V, X

Terminals : S, U, W, Y, Z

Start symbol : R

ns 10. To Find:

CFG for $A = \{ a^i b^j c^k \mid i, j, k \geq 0$ either $i = j$ or $j = k \}$

If A is ambiguous.

Solution:

Let $A_1 = \{ a^i b^j c^k \mid i = j \}$

$A_2 = \{ a^i b^j c^k \mid j = k \}$

CFG for A_1:

Variables in A_1 : S_1 (start state) A_1, B_1

Terminals : a, b, c

Grammar A_1:

$S_1 \rightarrow A_1 B_1$ (Concatenation of $a^i b^j$ & c^k where $i = j$)

$A_1 \rightarrow aA_1b \mid \varepsilon$

$B_1 \rightarrow cB_1 \mid \varepsilon$

Formally $G = (V, \Sigma, R, S)$

$G = (\{S_1, A_1, B_1\}, \{a, b, c\}, \{S_1 \to A_1 B_1,\ A_1 \to aA_1 B_1 \mid \epsilon,\ B_1 \to cB_1 \mid \epsilon\}, S_1)$

CFG for A_2:

Variables for A_2 : S_2, A_2, B_2
Terminals : a, b, c
Start state : S_2

Grammar of A_2:

$S_2 \to A_2 B_2$ (Concatenation)
$A_2 \to aA_2 \mid \epsilon$
$B_2 \to bB_2 c \mid \epsilon$

Formally,

$G = (\{S_2, A_2, B_2\}, \{a, b, c\}, \{S_2 \to A_2 B_2,\ A_2 \to aA_2 \mid \epsilon,\ B_2 \to bB_2 c \mid \epsilon\}, S_2)$

Let us construct CFG for A by taking a union of A_1 and A_2.

$A_1 = A_1 \cup A_2$

Let S be the start symbol for A.
Then grammar A:

$S \to S_1 \mid S_2$

$S_1 \to A_1 B_1$

$A_1 \to aA_1b \mid \varepsilon$

$B_1 \to cB_1 \mid \varepsilon$

$S_2 \to A_2 B_2$

$A_2 \to aA_2 \mid \varepsilon$

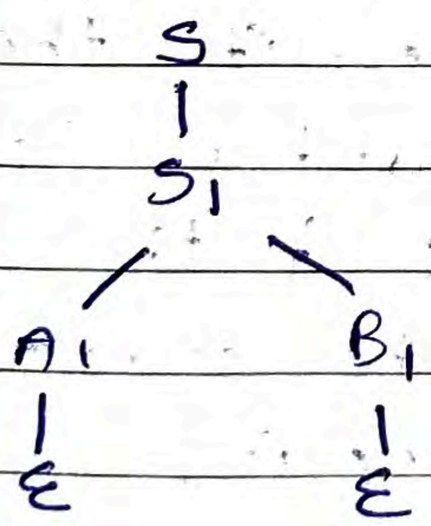
$B_2 \to bB_2c \mid \varepsilon$

Checking for ambiguity:

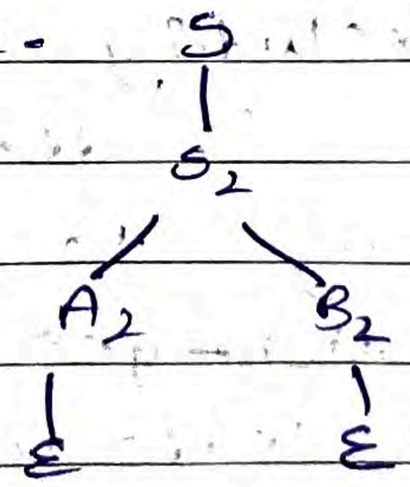
Let us take $w = a^i b^j c^k$, s.t. $i = j = k = 0$.
$w = \varepsilon$ is a string in both A_1 & A_2

Possible parse trees for $w = \varepsilon$ using A:

1.



2.



using production rules to generate two different parse trees to generate ε.
We observe that $\exists w \in A$ s.t it has more than one parse tree that can be used to derive it. Here $w = \varepsilon$. However

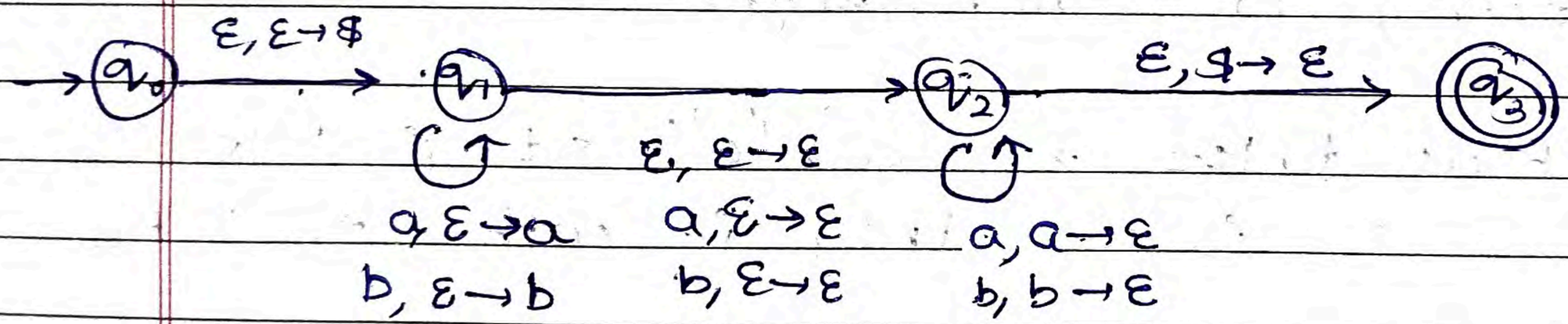
this is also true for any $a^i b^j c^k \in A$ st $i=j=k$. Any such string will have atleast 2 different parse trees or left most derivations.

∴ The CFG A is ambiguous.

Ans. 11. To Construct:

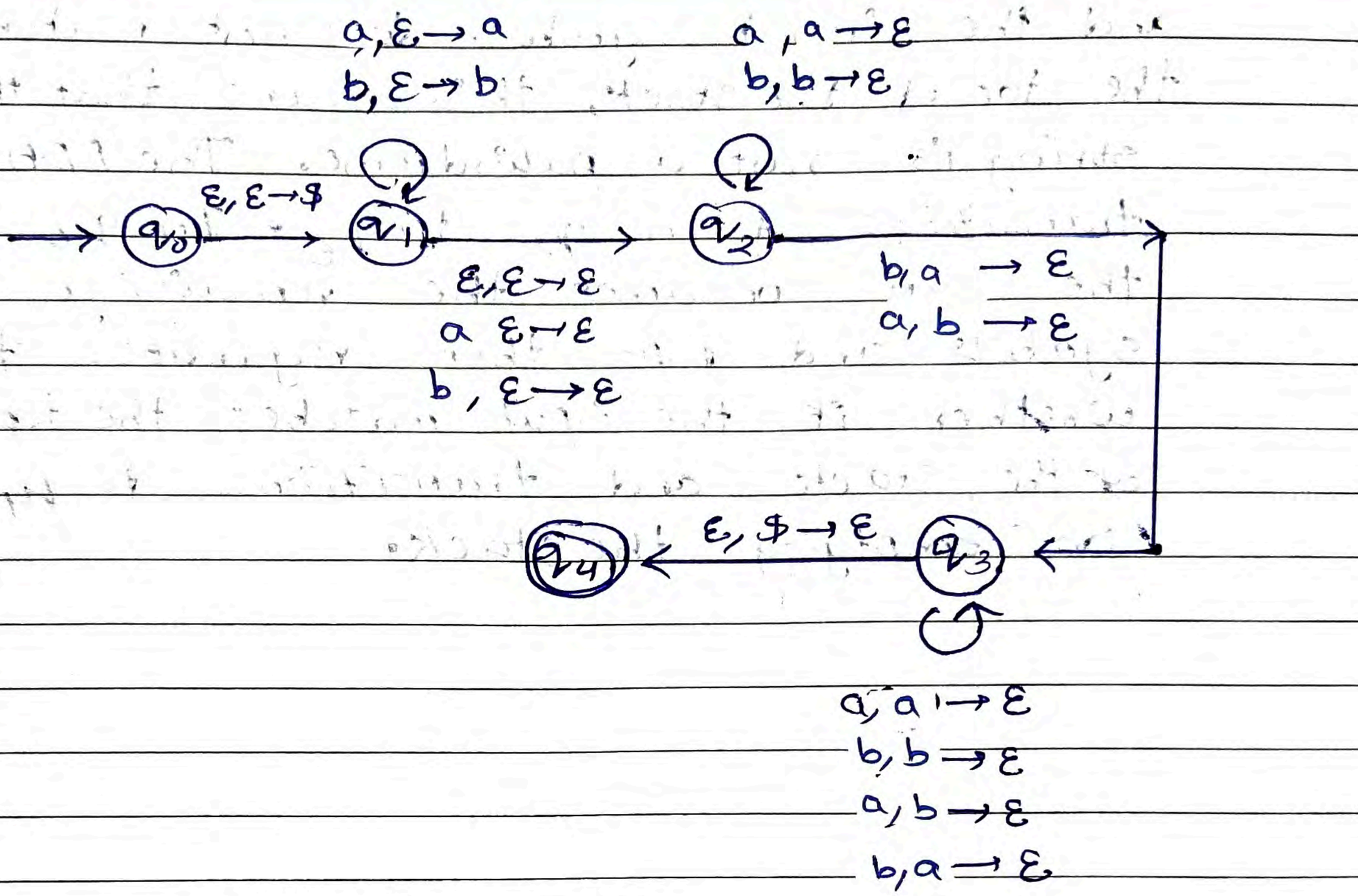
A PDA for the language of all non-palindromes over $\{a, b\}$.

We will start by constructing a PDA for the language of all palindromes over $\{a, b\}$



We push the first half of the string into the stack excluding the middle symbol if the palindrome is odd. After ** non-deterministically transitioning to q_2, we check if the input matchinge's the top of the stack & pop. This leads to the PDA accepting palindromes of all lengths.

Constructing a PDA that accepts the language of all non palindromes:



Reasoning:

Initially, we proceed as in the palindrome case and push the first half of the string into the stack. We skip the middle character if the string has odd length. We non-deterministically 'guess' the middle of the string & transition to q_2.

Our goal is to find atleast one mismatch in the first and second halves of the strings. Even if we find a single inconsistency, the string is not a palindrome.

Once we reach the second half of the string, we try to match the input symbols with the top of the stack and pop them. If at some point there is a mismatch

and the input symbol does not match the top of the stack, this means that the string is not a palindrome. The PDA transitions from q_2 to q_3. In q_3 the PDA consumes the remaining symbols and pops them regardless of whether it the input matches the top of the stack and transitions to q_4 on emptying the stack.

Ans 12: To find:
Language of given PDA & # of strings of length 100 accepted by it.

Solution:

The language $L = \{ a^n b^m \mid n > m, m \geq 0 \}$ is the language of the PDA.

Reasoning:

The PDA reads nothing from the input tape & pushes # onto the stack. It transitions from q_0 to q_1.
In q_1, it pushes A for every a read from the tape & remains in q_1.
It non-deterministically transitions to q_2 after pushing some a's onto the stack.
In q_2, for each b encountered, A is popped from stack
It can only transition from q_2 to q_3 via an epsilon transition if there is still an A remaining on the stack ensuring that # of a's & b's can never be equal.

Now, given length of string : $n+m=100$
We Know $n > m$, so

$$n = 100 - m$$
$$100 - m > m$$
$$100 > 2m$$
$$m < 50$$

∴ $m \geq 0$ & $m < 50$, m can range from 0 to 49 giving rise to strings in L of the form:

$a^{51}b^{49}, a^{50}b^{48}, \ldots a^{99}b, a^{100}$

∴ m ranges 0-49, there will exist 50 such strings where $n+m=100$.

∴ # strings of length 100 that the PDA accepts = 50.