Theory of Computation - Assignment 1 - Due: 17-Feb-2025, 11 PM

1. The 'finite' in FSA refers to
   (a) Finite number of states
   (b) Length of the string is finite
   (c) Both (A) and (B)
   (d) None
2. The language accepted by FSA can be
   (a) Finite or Infinite
   (b) Must be Finite (always)
   (c) If it is infinite, then it must have infinite number of states
   (d) Must be Infinite (always)
3. How many different FSAs are possible for the language $L = \Sigma^*, \Sigma = \{0, 1\}$
   (a) Finitely many
   (b) Infinitely many
   (c) No FSA exists
4. Are there FSAs in which every state is a final state.
   (a) YES
   (b) NO
5. Let $\Sigma = \{0, 1\}$. Let L be the language defined over Σ such that the number of 0's is a multiple of 3 and the number of 1's is a multiple of 4. The number of states is
   (a) 3
   (b) 4
   (c) 7
   (d) 12
6. How many DFAs are possible for the language $\{x \mid x$ begins with 0 and ends with 1 $\}$.
   (a) It is unique and exactly one
   (b) NO DFA exists
   (c) at least 3 FAs
   (d) None of the above
7. Tick all that are true. Let $\Sigma = \{1\}$. For the regular expression $(11111 + 111)^*$,
   A. the language $L = \{\epsilon, 111, 11111, 1^8, 1^9, 1^{10}, 1^{11}, ...\}$.
   B. there exists a minimal DFA with 9 states
   C. there exists a DFA with 8 states
   D. there is no NFA with 7 states.

8. Consider a DFA with 4 states such that all states are final states. Then, which of the following are true
   A. $L = \Sigma^*$.
   B. Any minimal DFA has exactly one state.
   C. $L \neq \Sigma^*$.
   D. there exists an equivalent epsilon NFA with 4 states.

9. How many different two state DFAs are possible for $\Sigma = \{a, b, c\}$. A. 256
   B. 512
   C. 128
   D. None of the above

10. Let L be a language over $\{a, b\}$ with the property that all strings in L are of odd length. Which of the following is (are) the regular expressions for L.
   A. $(a + b)(aa + ab + ba + bb)^*(b + a)$
   B. $(a + b)(aa + ab + ba + bb)^*$
   C. $(a + b)(aa + ab + ba + bb)^* + (aa + ab + ba + bb)^*(b + a)$
   D. $(aa + ab + ba + bb)^*(b + a)$

11. Tick all that are true. The regular expression for the set of strings over $\{0, 1\}$ not containing 11 as a substring.
   A. $(00 + 01 + 10)^*$
   B. $0^* + 0^*1 + (0^*10^*)^*$
   C. $\epsilon + 1 + 0^* + (0^*10^*)^*$
   D. $(10 + 0)^* + (10 + 0)^*1$.

Bhadresh's Solutions:

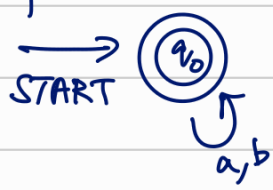
1) FSA - Finite State Automaton. The purpose of a finite state automaton is to parse any string over a given set of alphabets and check if the string agrees with the condition of the automaton, using a finite number of states. Hence, "Finite" in FSA refers to the finite number of states. ∴, my answer is option (a). The FSA has an infinite input tape, meaning it can potentially read over infinite strings too - the result provided by the FSA is independent of the size of the string, hence (b) is wrong. Since this is the case, (b) and (d) are implicitly incorrect.

2) Language accepted by the machine, i.e. $L(M)$, refers to the set of all strings that are accepted by the machine / FSA.

let $M = \{ Q = \{q_0, q_1, \cdots, q_f\}, \Sigma = \{a, b\}, \delta, q_0, F \}$.
$\delta \hookrightarrow Q \times \Sigma \to Q$

$$\therefore L(M) = \{ x \in \Sigma^* \mid \hat{\delta}(q_0, x) = q_f \in F \}.$$

in the definition of the set, there is no restriction of finiteness / infiniteness of the string. in fact, the machine can take even infinite length strings!
∴, the answer is (a). Hence, (b) & (d) are wrong. (c) is false, as I could have an FSA accepting an infinite string, with a single state - (e.g).

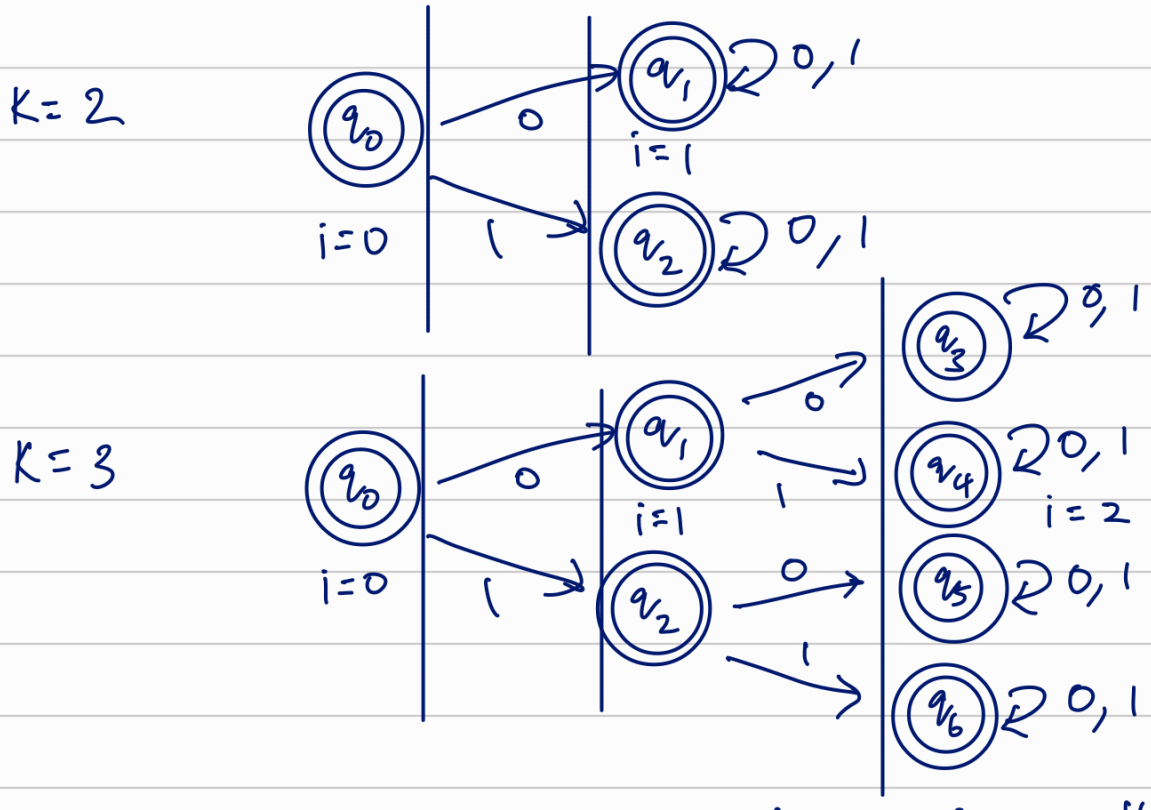


→ accepts any string (even infinite) over $\{a, b\}$. ∴, (c) is incorrect. (contradiction)

3) $L = \Sigma^*$, $\Sigma = \{0, 1\}$.

(c) is wrong, by proof by contradiction.

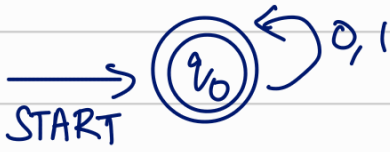
trying to check if (b) is right – look at a single state FSA as above. it works, by observation. lets construct a K-levelled automaton, in which the ith level has 2^i states. ($i \leq K-1$). for e.g.



and so on··· by observation, all of these automatons accept any string over $\{0, 1\}$. ∴ the claim holds good $\forall K \in \mathbb{N}$. Since infinitely many automatons are possible in this model, infinitely many FSAs exist for the language $L = \Sigma^*$, $\Sigma = \{0, 1\}$. ∴ (b) is correct, and so, (a) is wrong.

(the branches in the FSA just brings in more order, final state gives info about the initial alphabets. So, it must work); (also, cardinality of $\mathbb{N}$ is countably ∞, supports my claim)

4) There are indeed FSAs in which EVERY state is a final state - looking into the previous question, the K-levelled FSA model have all of their states marked final. $\therefore$, answer is (a). for example,



$M = \{Q = \{q_0\}, \Sigma = \{0,1\}, \delta, q_0, F = \{q_0\}\}$

q_0 is the only state, and its final. $\therefore$, $\exists$ FSAs s.t. all of it's states are final.

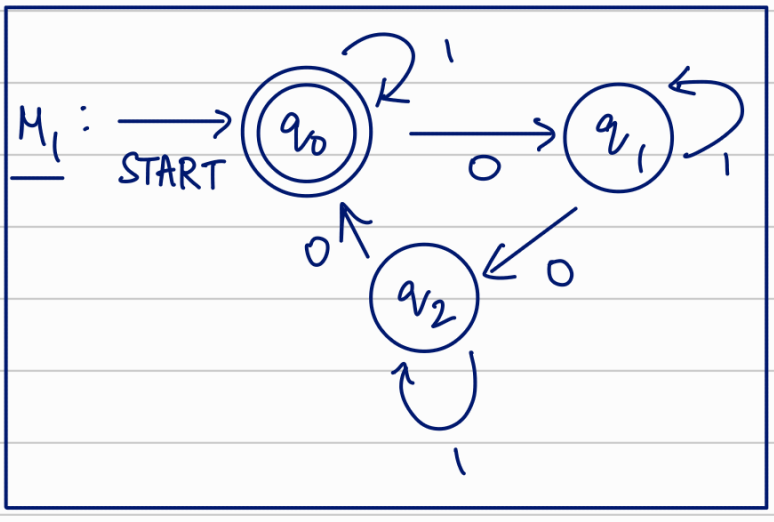
5) $\Sigma = \{0,1\}$

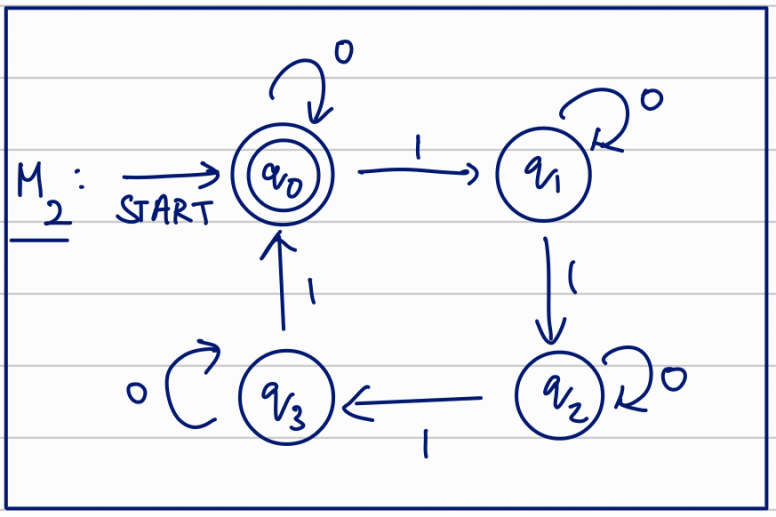
$M = \{Q, \Sigma, \delta, q_0, F\}$ $\quad L(M) = \left\{ x \in \Sigma^* \;\middle|\; \begin{matrix} \#_0(x) \,\%\, 3 = 0 \\ \#_1(x) \,\%\, 4 = 0 \end{matrix} \right\}$

let M_1 be s.t. $L(M_1) = \{ x \in \Sigma^* \mid \#_0(x) \,\%\, 3 = 0 \}$

and

let M_2 be s.t. $L(M_2) = \{ x \in \Sigma^* \mid \#_1(x) \,\%\, 4 = 0 \}$





(w.k.t. $M = M_1 \wedge M_2$.)

P.T.O.

$$M = M_1 \wedge M_2 = (Q_1 \times Q_2, \Sigma, \delta, (q_0, q_0'), F_1 \times F_2)$$

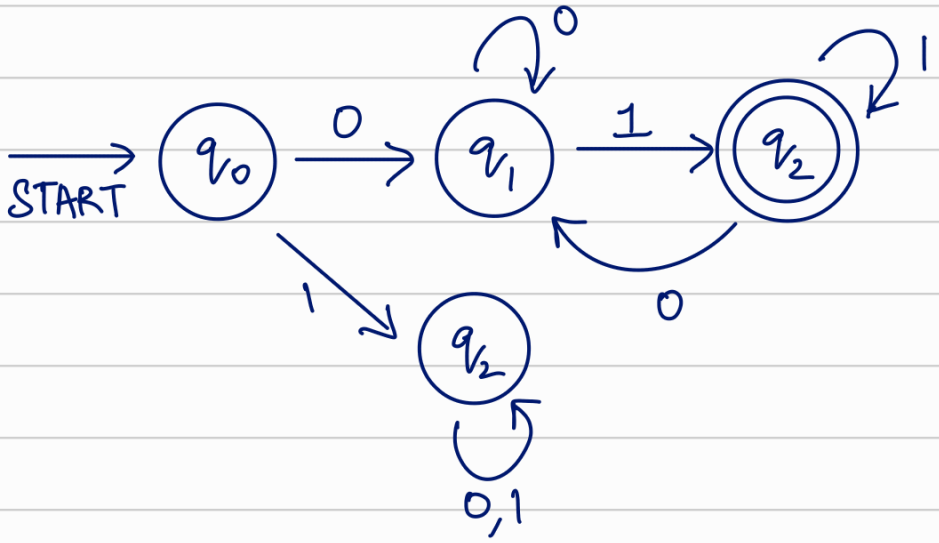
$$L(M) = \left\{ x \in \Sigma^* \;\middle|\; \hat{\delta}((q_0, q_0'), x) = (q_f, q_f') \in F_1 \times F_2 \right\}$$

for M to be a well-defined machine, δ must be well-defined. Here, $\delta : (Q_1 \times Q_2) \times \Sigma \longrightarrow (Q_1 \times Q_2)$, meaning the fn. has to be defined for $|Q_1 \times Q_2|$ no. of values (∃ value in codomain for each in domain). ∴, $|Q_1 \times Q_2| = 3 \times 4 = 12$ states are to be defined in the product automaton. ∴ answer is (d). Others are wrong as the function won't be well-defined otherwise, hence the machine itself too. the representation is as follows:

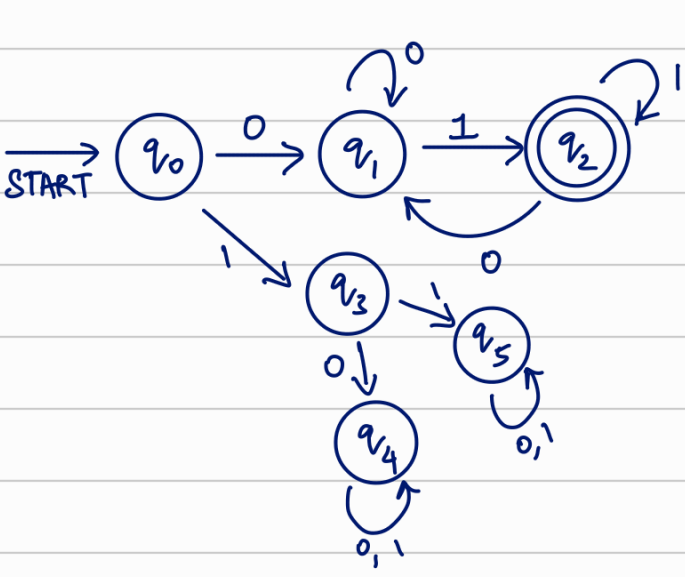
State	0	1
q_0, q_0'	q_1, q_0'	q_0, q_1'
q_0, q_1'	q_1, q_1'	q_0, q_2'
q_0, q_2'	q_1, q_2'	q_0, q_3'
q_0, q_3'	q_1, q_3'	q_0, q_0'
q_1, q_0'	q_2, q_0'	q_1, q_1'
q_1, q_1'	q_2, q_1'	q_1, q_2'
q_1, q_2'	q_2, q_2'	q_1, q_3'
q_1, q_3'	q_2, q_3'	q_1, q_0'
q_2, q_0'	q_0, q_0'	q_2, q_1'
q_2, q_1'	q_0, q_1'	q_2, q_2'
q_2, q_2'	q_0, q_2'	q_2, q_3'
q_2, q_3'	q_0, q_3'	q_2, q_0'

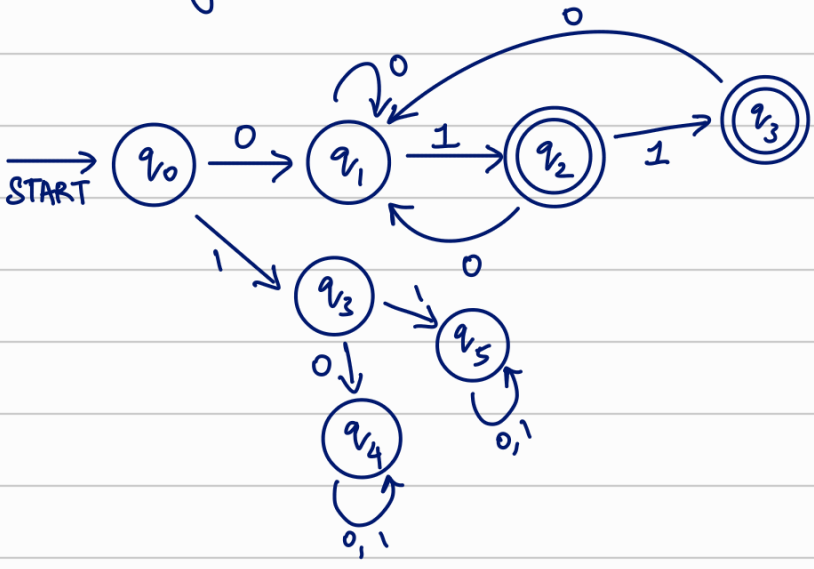
Hence, the pdt. automaton is well-defined with 12 states. Answer is (d).

6) $L(M) = \{x \in \{0,1\}^* \mid x \text{ begins with } 0 \text{ \& ends with } 1\}$
to find no. of DFAs possible for this



the above DFA accepts the language proposed. It can be modified in 2 other ways as shown below:





Since this is the case, answer must be (c). ∴, none of the others can be possible answers.

7)

7. Tick all that are true. Let $\Sigma = \{1\}$. For the regular expression $(11111 + 111)^*$,
   A. the language $L = \{\epsilon, 111, 11111, 1^8, 1^9, 1^{10}, 1^{11}, ...\}$.
   B. there exists a minimal DFA with 9 states
   C. there exists a DFA with 8 states
   D. there is no NFA with 7 states.

given : $\Sigma = \{1\}$ and $RE = (11111 + 111)^*$

ordering the count set wrt no. of 1's - 0, 3, 5, 6, 8, 9, 10, 11

$$L(M) = \left\{ x \in \{0,1\}^* \;\middle|\; \#_1(x) = 3n_1 + 5n_2 \right\}$$

on hindsight it seems like a rather non-trivial problem, but if we recall the coin-exchange problem where using only 2 denominations we have to represent any change greater than a threshold, we realise that both the problems are similar. the base case is 8. we can prove by M.I. that $\forall$ sizes ≥ 8, strings are accepted.

$11 \rightarrow 3+3+5$
$12 \rightarrow 3+3+3+3$
$13 \rightarrow 3+5+5$
$14 \rightarrow 3+5+3+3$
$15 \rightarrow 5+5+5$
$16 \rightarrow 5+5+3+3$

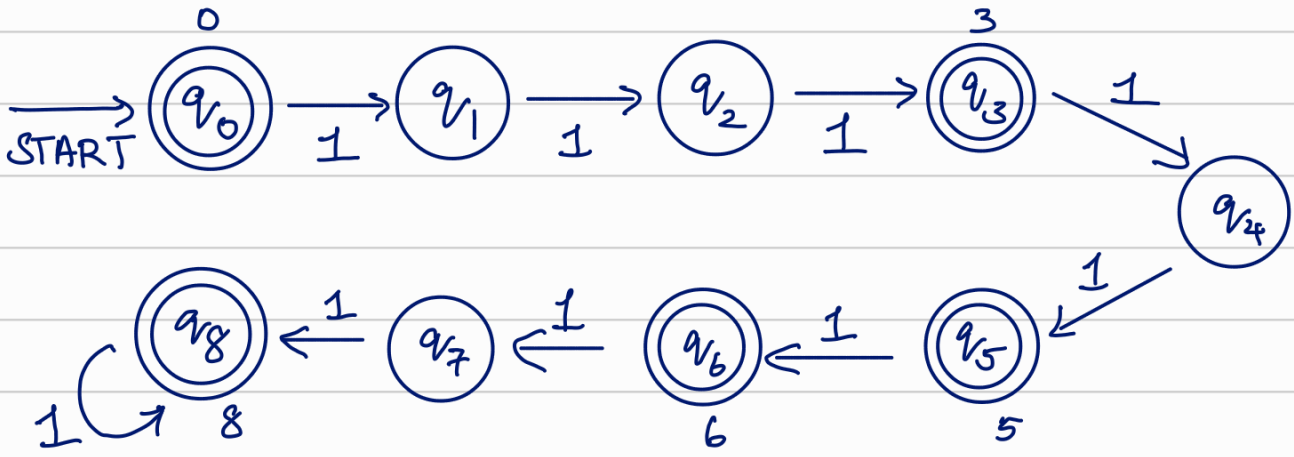
↗ and so on until $K = 3x_1 + 5x_2$ for $K+1$,

$K+1 = K + (3 \times 2) - 5$ (or)
$K + (5 \times 2) - (3 \times 3)$
dep. on availability of multiples.

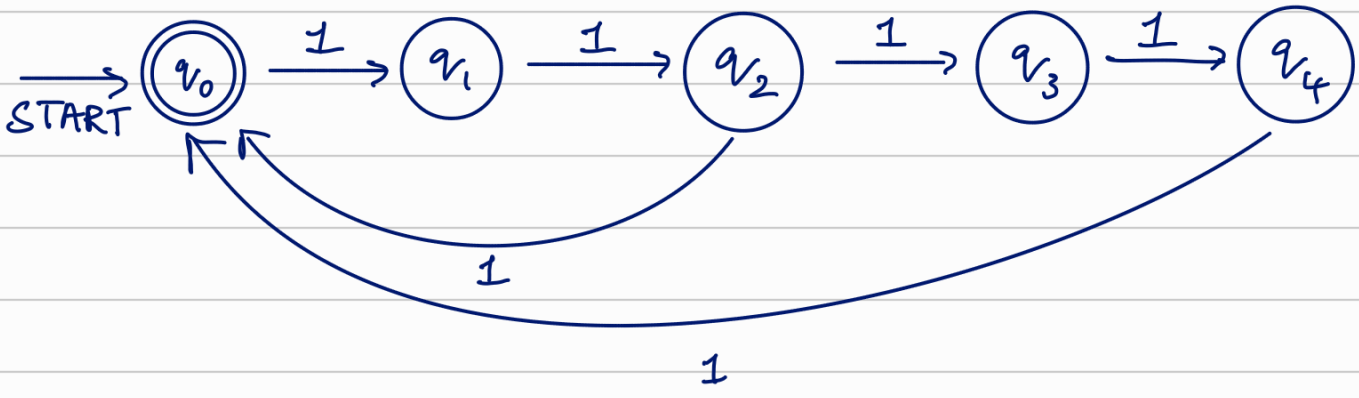
∴, $K+1$ can also be represented.

So, a FSA checking base conditions $\{0, 3, 5, 6, 8\}$ is enough. greater numbers shall be accepted, numbers in range of set but not in set shall be rejected.

P.T.O.

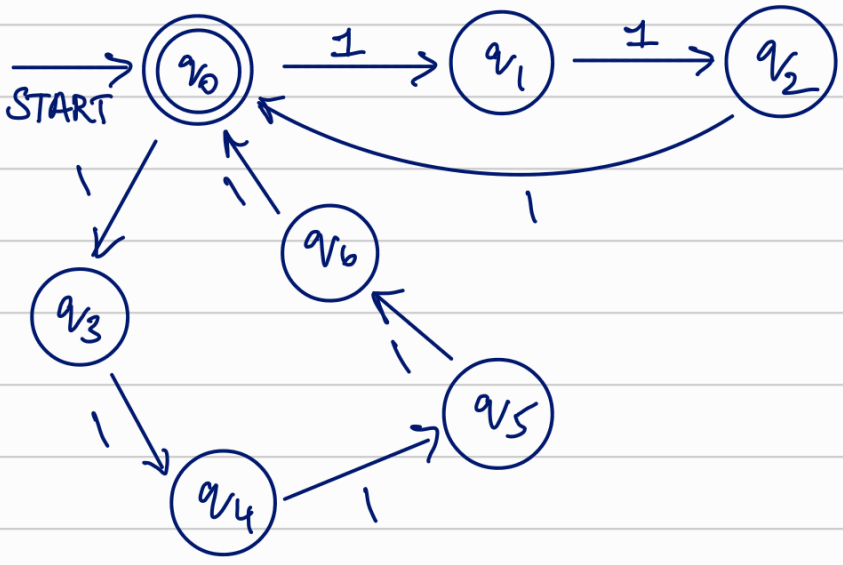


my claim is that this is the minimal DFA, as the base cases need to be checked at the least. ∴, 9-states are reqd to reject 1, 2, 4, 7 & accept 3, 5, 6, 8. & others.... Hence, (b) is correct, meaning (c) is WRONG! assuming that the order in (a) is uniform wrt size, (a) is also incorrect as 1^6 is missing. looking into NFA,



proof that this NFA works lies in the proof of M.I. if K is accepted, K+1 will also be accepted, but the sequence will change. if $K = 3x_1 + 5x_2$, K+1 coued be $K + (3\times2) - 5$ or $K + (5\times2) - (3\times3)$ dep. on avail. of multiplicants. ∴, $K+1 = 3(x_1 - 3) + 5(x_2 + 2)$ (or) $K+1 = 3(x_1 + 2) + 5(x_2 - 1)$ → meaning the no. of 5-state loops & 3-state loops changes. but, K+1 is still in the $3x+5y$ form, hence accepted. Hence, NFA with 5-states exists. this is the minimal NFA,

as at least 5 states are reqd. to check occurrence of multiples of 5. A 7-state NFA can be constructed in the following way:



this works under the same expn. given for M.I.
∴, (d) is false!

8)

8. Consider a DFA with 4 states such that all states are final states. Then, which of the following are true
A. $L = \Sigma^*$.
B. Any minimal DFA has exactly one state.
C. $L \neq \Sigma^*$.
D. there exists an equivalent epsilon NFA with 4 states.

by intuition (a) must be true, as whatever $a \in \Sigma$ you see at any state, you are always going to stay at / move to another final state. since all states are final, 1-state must be enough to accept all strings. ∴, (b) must also be correct. Hence, (c) is wrong. since every state is final state even if we choose to traverse to other states by choosing not to read anything, eventually we will end up at a final state only. ∴, ϵ-NFA exists. Hence, (d) is correct.

9)

9. How many different two state DFAs are possible for $\Sigma = \{a, b, c\}$. A. 256
   B. 512
   C. 128
   D. None of the above

transitions wrt q_0 : q_0, a } each 2 choices (stay at q_0 or move to q_1)
q_0, b
q_0, c }

q_1 : q_1, a } each 2 choices (stay at q_1 or move to q_0)
q_1, b
q_1, c }

other than this, q_0 & q_1 can either be final / not.
∴, each 2 choices. AND, for the total selection, either q_0 (or) q_1 must be start, so 2 choices there. Hence,

$$^2C_1 \times \left(\left({}^2C_1 \times \left(\underbrace{{}^2C_1 \times {}^2C_1 \times {}^2C_1}_{q_0,\ (a,b,c)\ \text{choices.}} \right) \right) \times \left({}^2C_1 \times \left(\underbrace{{}^2C_1 \times {}^2C_1 \times {}^2C_1}_{q_1,\ (a,b,c)\ \text{choices.}} \right) \right) \right)$$

$^2C_1$: q_0 / q_1 start

$^2C_1$: q_0 f/nf

q_1 f/nf

$$= 2 \times 2^4 \times 2^4 = 2 \times 2^8 = 2^9 = 512 \text{ DFAs possible.}$$

∴, (b) is correct.

10)

10. Let L be a language over $\{a, b\}$ with the property that all strings in L are of odd length. Which of the following is (are) the regular expressions for L.
A. $(a+b)(aa+ab+ba+bb)^*(b+a)$
B. $(a+b)(aa+ab+ba+bb)^*$
C. $(a+b)(aa+ab+ba+bb)^* + (aa+ab+ba+bb)^*(b+a)$
D. $(aa+ab+ba+bb)^*(b+a)$

consider (a): $(a+b)(aa+ab+ba+bb)^*(b+a)$
① ② ③

① gives one of a or b, hence odd.
② gives an even count of symbols, hence even.
③ gives one of b or a, hence odd.
adding the counts of symbols by ①, ② and ③ we would get an even no. (odd + even + odd = even).
∴, (a) is incorrect.

(b): $(a+b)(aa+ab+ba+bb)^*$
① ②

① gives one of a or b, hence odd.
② gives an even count of symbols, hence even.
sum of symbols by ① + ② would return an odd value always! (odd + even = odd).
∴, (b) is correct.

(c): its similar to (b) – since we choose any one of $(a+b)(aa+ab+ba+bb)^*$ or $(aa+ab+ba+bb)^*(b+a)$, and either of them have odd symbols only, (c) is also correct!

(d): is the same as (b). ∴, (d) is correct.

11)

11. Tick all that are true. The regular expression for the set of strings over $\{0,1\}$ not containing 11 as a substring.
A. $(00+01+10)^*$
B. $0^*+0^*1+(0^*10^*)^*$
C. $\epsilon+1+0^*+(0^*10^*)^*$
D. $(10+0)^*+(10+0)^*1$.

consider (a): $(00+01+10)^*$

pump it 2 times → $(00+01+10)(00+01+10)$

↓ choose ↓ choose

0110 is accepted by the RE, but it contains 11 as substring.

(b): $0^*+0^*1+(0^*10^*)^*$

choose & pump twice.

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

$\epsilon 1 \epsilon \quad \epsilon 1 \epsilon \rightarrow \epsilon 1 \epsilon\epsilon 1 \epsilon = 11$

∴, (b) is wrong.

(c): $\epsilon+1+(0^*10^*)^*$ → by same logic as (b), (c) is also incorrect.

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

(i) choose & pump $(10+0)^*$ twice

$(10+0)(10+0)$ → can't get 11 in any combo.

(ii) choose $(10+0)^*1$ & pump * once

$(10+0)1$ → can't get 11 in any combo

∴, (d) is correct.