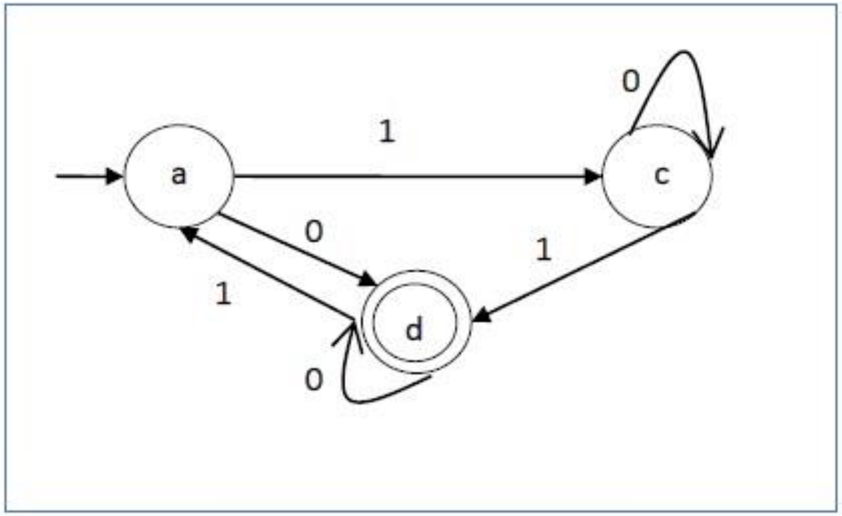
CS202 – System Software

Dr. Manish Khare

Lecture 7

- Let us consider the DFA shown in Figure. From the DFA, the acceptable strings can be derived.



- Strings accepted by the above DFA: {0, 00, 11, 010, 101, ...}
- Strings not accepted by the above DFA: {1, 011, 111, ...}

- Consider the finite state machine whose transition function δ is given by following Table in the form of a transition table. Here, $Q = \{q_0,q_1,q_2,q_3\}$, $\sum = \{0,1\}$, $F = \{q_0\}$. Give the entire sequence of states for the input string 110001.

State	Input	
	0	1
$\rightarrow q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

$$
\begin{aligned}
\delta(q_0,\ \overset{\downarrow}{1}10101) &= \delta(q_1, \overset{\downarrow}{1}0101) \\
&\qquad\quad \downarrow \\
&= \delta(q_0, \overset{}{0}101) \\
&\qquad\quad \downarrow \\
&= \delta(q_2, 101) \\
&\qquad\quad \downarrow \\
&= \delta(q_3, 01) \\
&\qquad\quad \downarrow \\
&= \delta(q_1, 1) \\
&= \delta(q_0, \Lambda) \\
&= q_0
\end{aligned}
$$

Hence,

$$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_1 \xrightarrow{1} q_0$$

The symbol $\downarrow$ indicates that the current input symbol is being processed by the machine.

Construction of DFA

- We can construct DFA of two types of problems
  - Construction of DFA for languages consisting of strings ending with a particular substring
  - Construction of DFA for languages consisting of strings starting with a particular substring

Type – 1

Construction of DFA for languages consisting of strings ending with a particular substring

- **<u>Step-01:</u>**
  - Determine the minimum number of states required in the DFA.
  - Draw those states.
- Use the following rule to determine the minimum number of states-
  - Calculate the length of substring.
  - All strings ending with 'n' length substring will always require minimum (n+1) states in the DFA.

- **<u>Step-02:</u>**
  - Decide the strings for which DFA will be constructed.

- **<u>Step-03:</u>**
  - Construct a DFA for the strings decided in Step-02.

- **<u>Step-04:</u>**
  - Send all the left possible combinations to the starting state.
  - Do not send the left possible combinations over the dead state.

Exercise

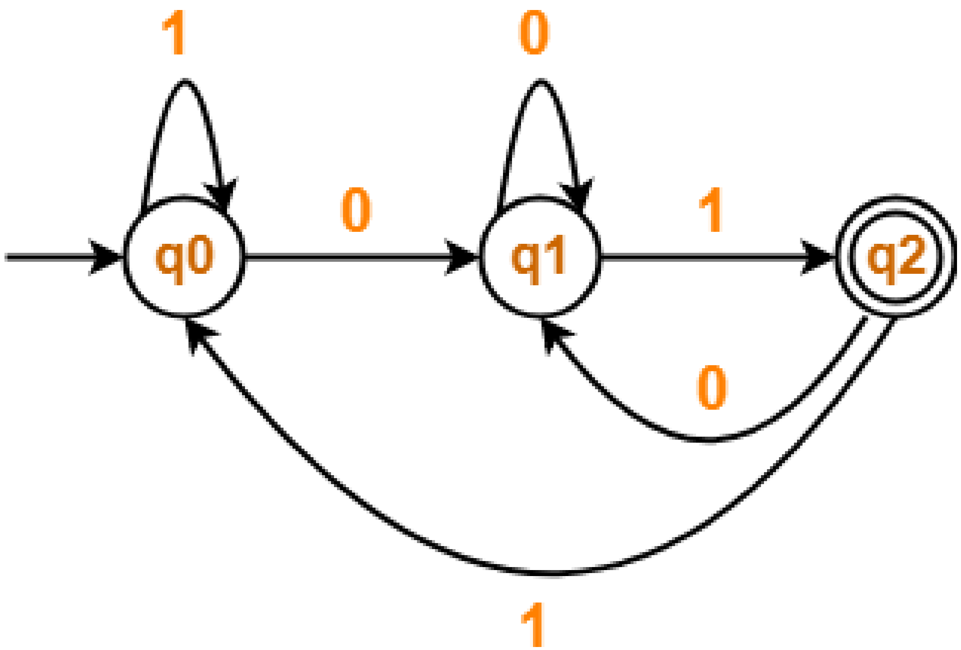
- Draw a DFA for the language accepting strings ending with '01' over input alphabets $\sum = \{0, 1\}$

- Regular expression for the given language = $(0 + 1)^*01$

- **Step-01:**
  - All strings of the language ends with substring “01”.
  - So, length of substring = 2.
- Thus, Minimum number of states required in the DFA = 2 + 1 = 3.
- It suggests that minimized DFA will have 3 states.

- **Step-02:**
- We will construct DFA for the following strings-
  - 01
  - 001
  - 0101

- Step 3 and 4
- The required DFA is-

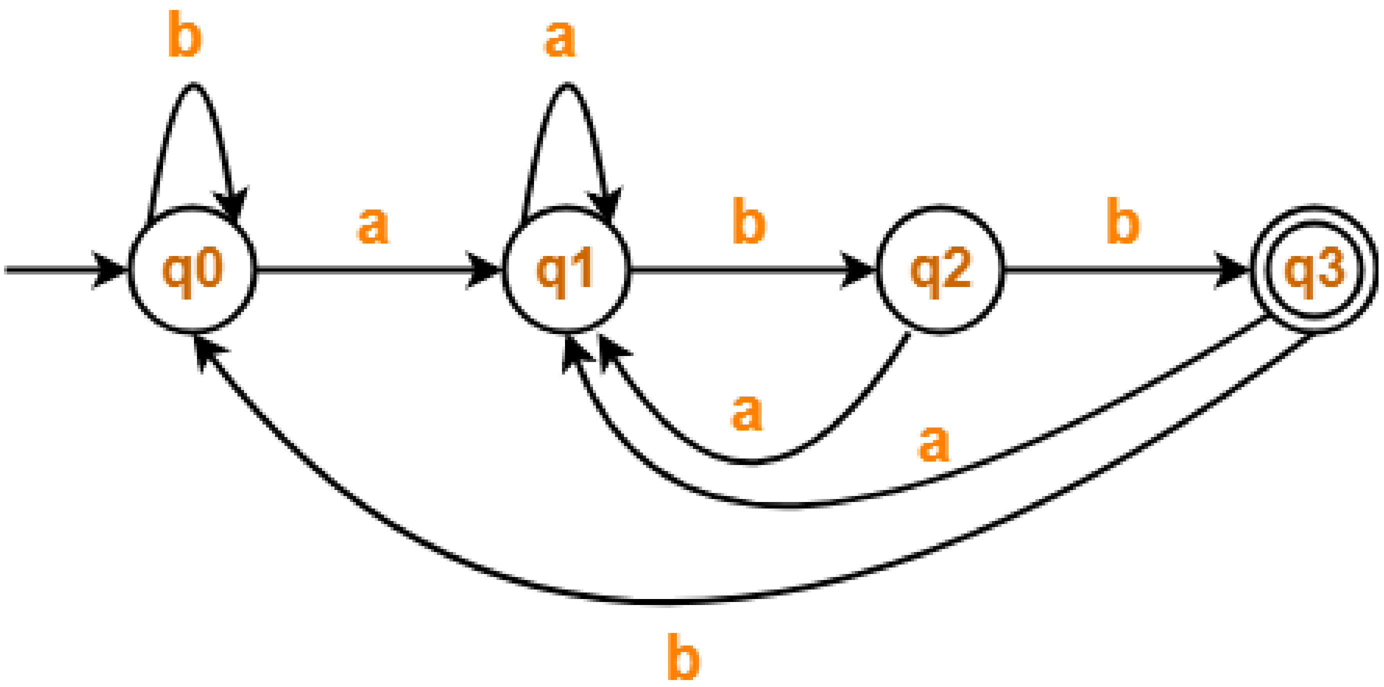


DFA

Exercise

- Draw a DFA for the language accepting strings ending with 'abb' over input alphabets $\sum = \{a, b\}$

- Regular expression for the given language = $(a + b)^*abb$



DFA

Type-2

- **Step-01:**
  - Determine the minimum number of states required in the DFA.
  - Draw those states.

- Use the following rule to determine the minimum number of states-
  - Calculate the length of substring.
  - All strings starting with ‘n’ length substring will always require minimum (n+2) states in the DFA.

- **Step-02:**
  - Decide the strings for which DFA will be constructed.
- **Step-03:**
  - Construct a DFA for the strings decided in Step-02.
- **Step-04:**
  - Send all the left possible combinations to the dead state.
  - Do not send the left possible combinations over the starting state.

Exercise

- Draw a DFA for the language accepting strings starting with 'ab' over input alphabets $\sum = \{a, b\}$

- Regular expression for the given language = $ab(a + b)^*$

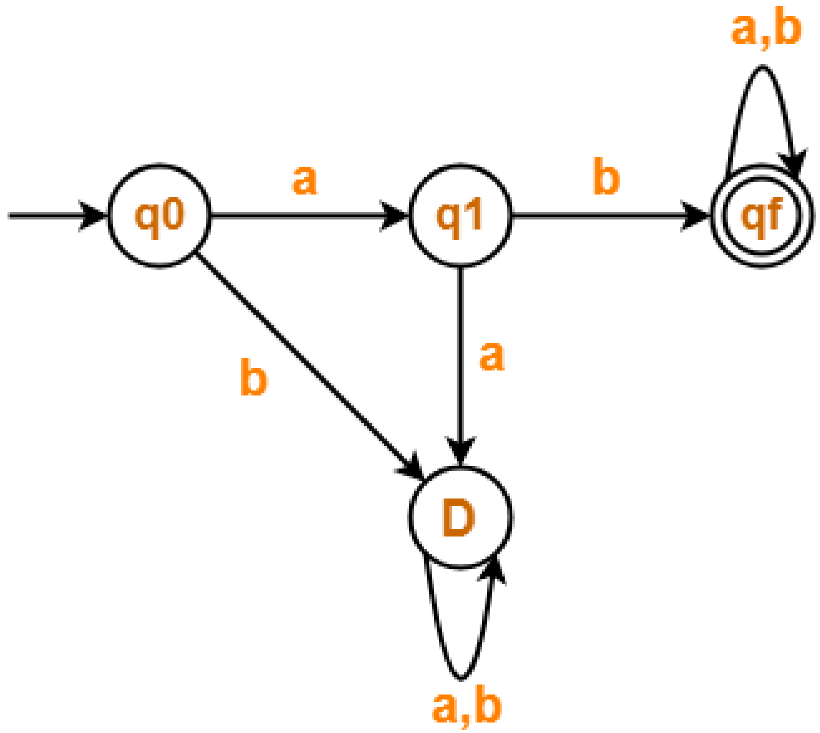
Step-01:

- All strings of the language starts with substring “ab”.
- So, length of substring = 2.
- Thus, Minimum number of states required in the DFA = 2 + 2 = 4.
- It suggests that minimized DFA will have 4 states.

Step-02:

- We will construct DFA for the following strings-
- ab
- aba
- abab

- **<u>Step-03 and 4:</u>**
- The required DFA is-

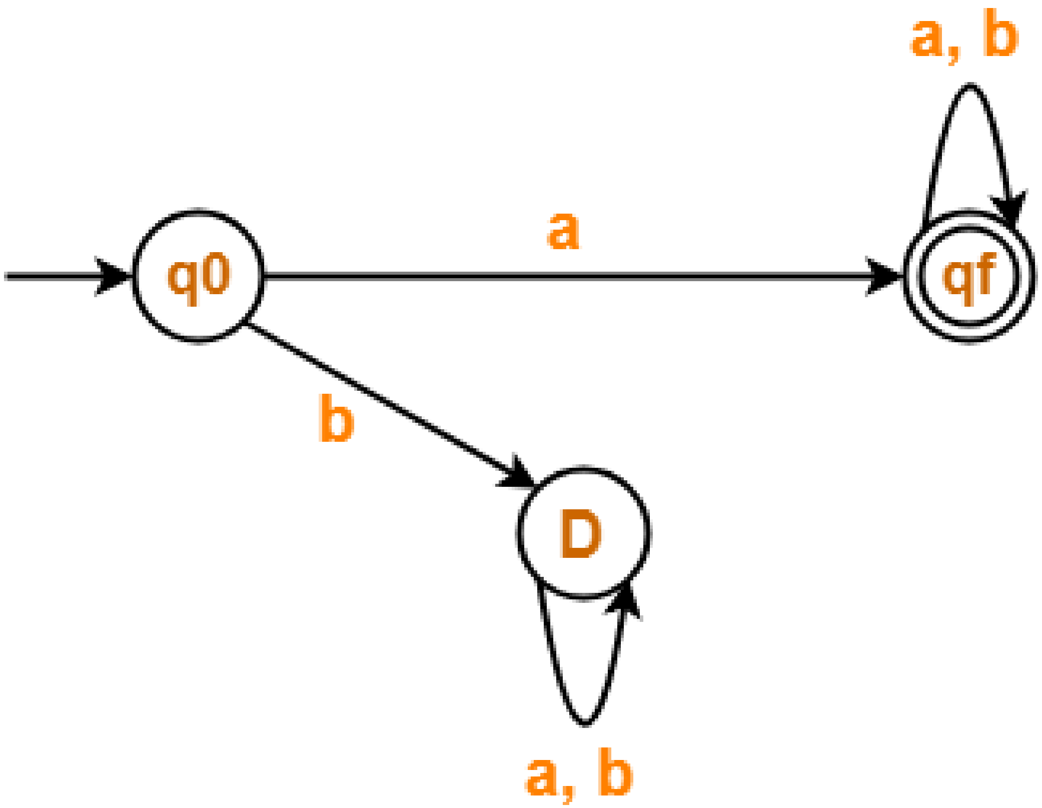


DFA

Exercise

- Draw a DFA for the language accepting strings starting with 'a' over input alphabets $\sum = \{a, b\}$

- Regular expression for the given language = $a(a + b)^*$



DFA

Minimization of DFA

- **Step-01:**
  - Eliminate all the dead states and inaccessible states from the given DFA (if any).

- **Dead State**
  - All those non-final states which transit to itself for all input symbols in ∑ are called as dead states.

- **Inaccessible State**
  - All those states which can never be reached from the initial state are called as inaccessible states.

- **Step-02:**
  - Draw a state transition table for the given DFA.
  - Transition table shows the transition of all states on all input symbols in Σ.
- **Step-03:**
  - Now, start applying equivalence theorem.
    - Take a counter variable k and initialize it with value 0.
    - Divide Q (set of states) into two sets such that one set contains all the non-final states and other set contains all the final states.
    - This partition is called P_0.

Step-04:

- Increment k by 1.
- Find P_k by partitioning the different sets of P_{k-1} .
- In each set of P_{k-1} , consider all the possible pair of states within each set and if the two states are distinguishable, partition the set into different sets in P_k.

- Two states q_1 and q_2 are distinguishable in partition P_k for any input symbol ‘a’,
- if δ (q_1, a) and δ (q_2, a) are in different sets in partition P_{k-1}.

Step-05:

- Repeat step-04 until no change in partition occurs.
- In other words, when you find $P_k = P_{k-1}$, stop.

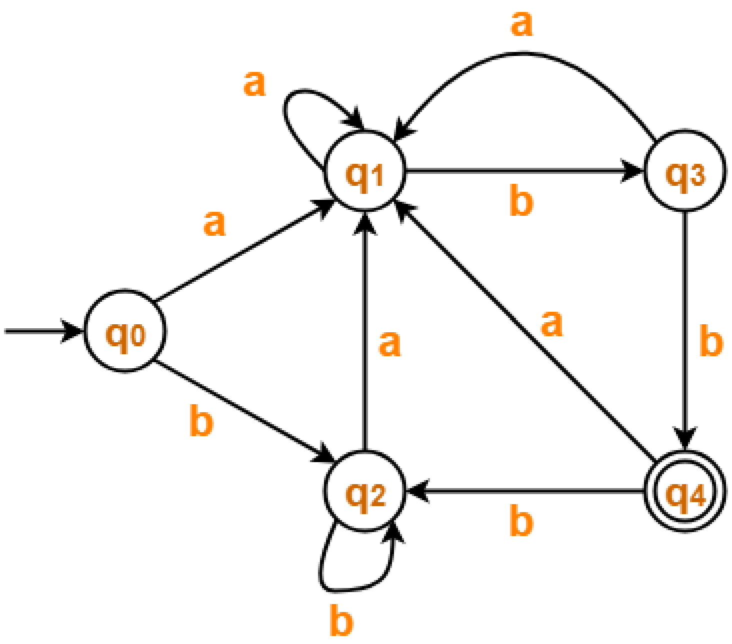
Step-06:

- All those states which belong to the same set are equivalent.
- The equivalent states are merged to form a single state in the minimal DFA.

Number of states in Minimal DFA
= Number of sets in P_k

Exercise

- Minimize the given DFA-



Step-01:

- The given DFA contains no dead states and inaccessible states.

Step-02:

- Draw a state transition table-

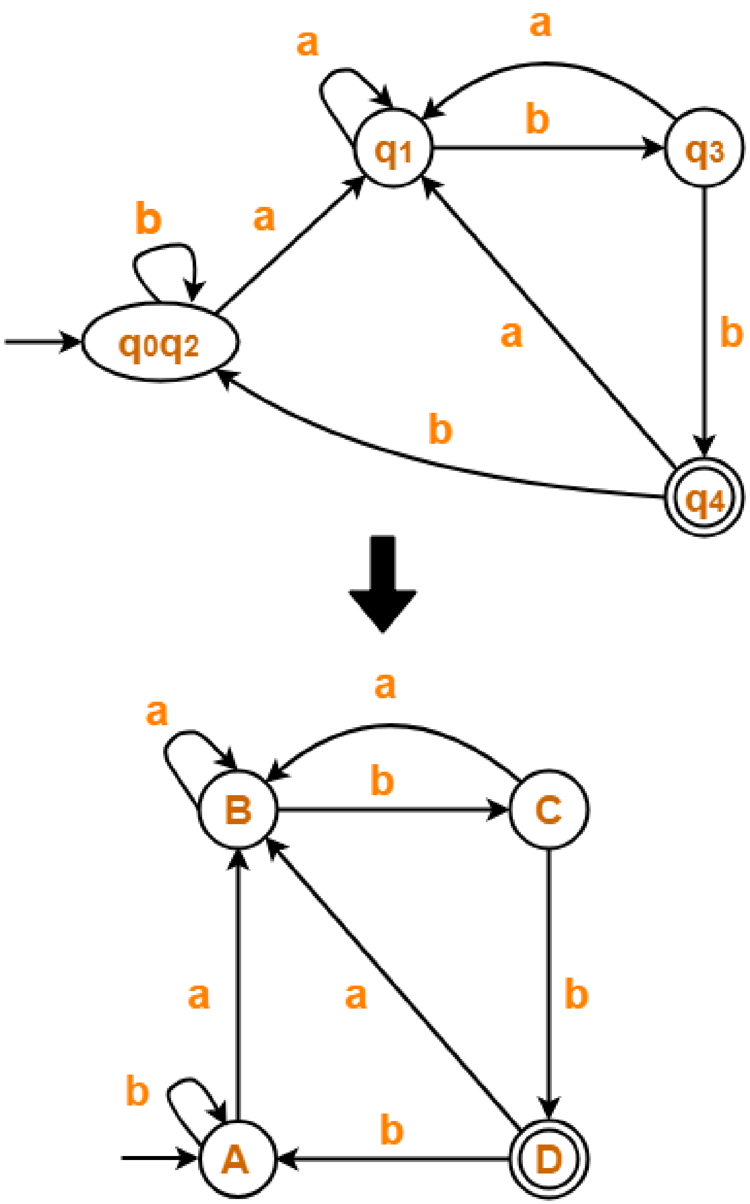
States	**a**	**b**
→q0	q1	q2
q1	q1	q3
q2	q1	q2
q3	q1	*q4
***q4**	q1	q2

Step-03:

- Now using Equivalence Theorem, we have-
  - $P_0 = \{ q_0 , q_1 , q_2 , q_3 \} \{ q_4 \}$
  - $P_1 = \{ q_0 , q_1 , q_2 \} \{ q_3 \} \{ q_4 \}$
  - $P_2 = \{ q_0 , q_2 \} \{ q_1 \} \{ q_3 \} \{ q_4 \}$
  - $P_3 = \{ q_0 , q_2 \} \{ q_1 \} \{ q_3 \} \{ q_4 \}$

- Since $P_3 = P_2$, so we stop.
- From P_3, we infer that states q_0 and q_2 are equivalent and can be merged together.

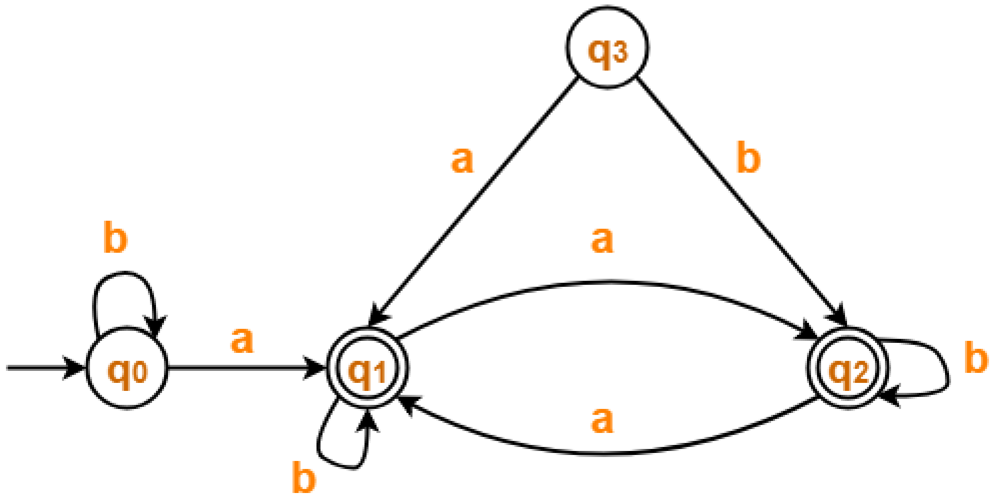
- So, Our minimal DFA is-



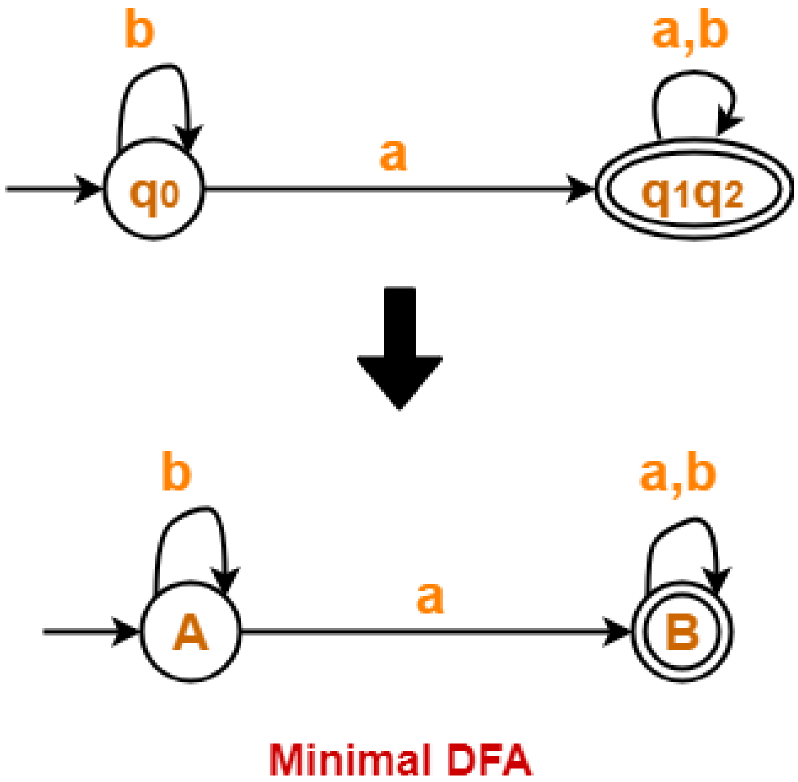
Minimal DFA

Exercise

- Minimize the given DFA-

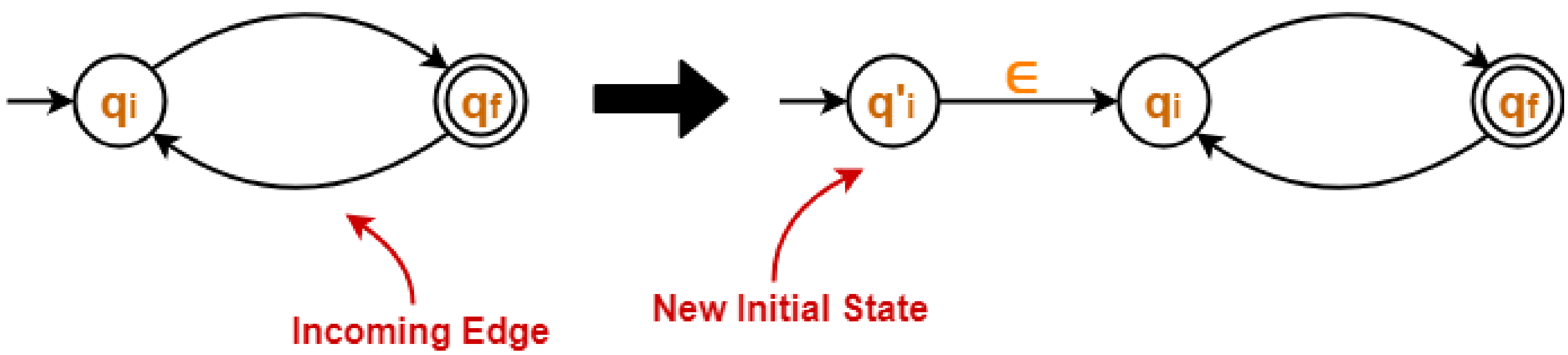


- Our Minimal DFA is:



DFA to Regular Expression

- **Step-01:**
  - For DFA to regular expression, first we need to check that, the initial state of the DFA must not have any incoming edge.
  - If there exists any incoming edge to the initial state, then create a new initial state having no incoming edge to it.
- Example

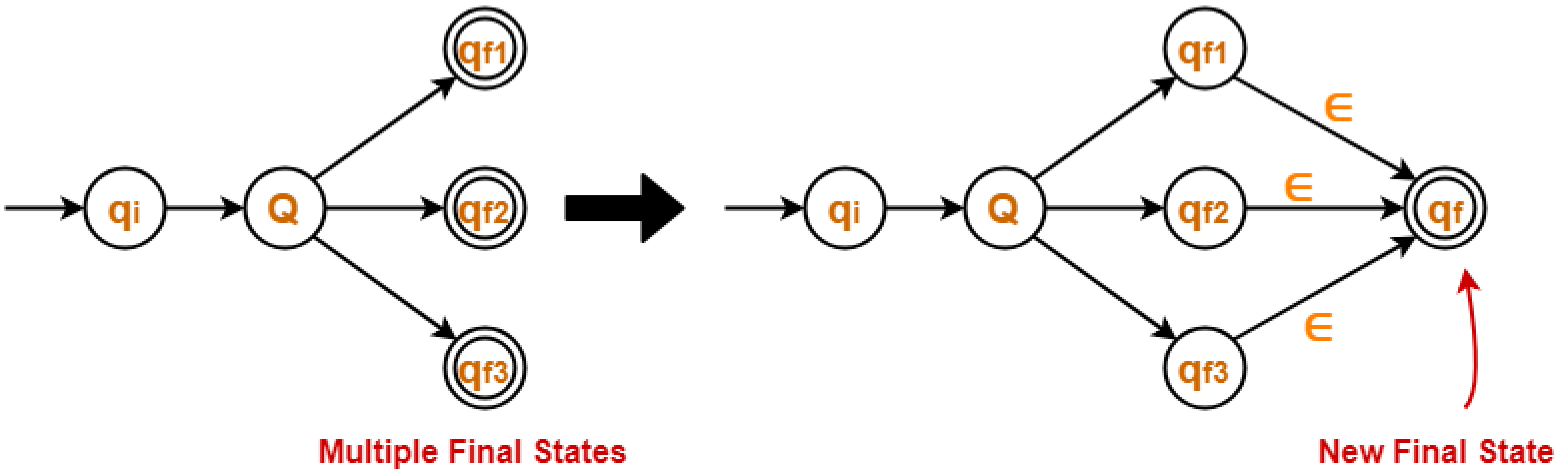


➢ **Step-02:**

- There must be exist only one final state in the DFA.

- If there exists multiple final states in the DFA, then convert all the final states into non-final states and create a new single final state.

➢ Example

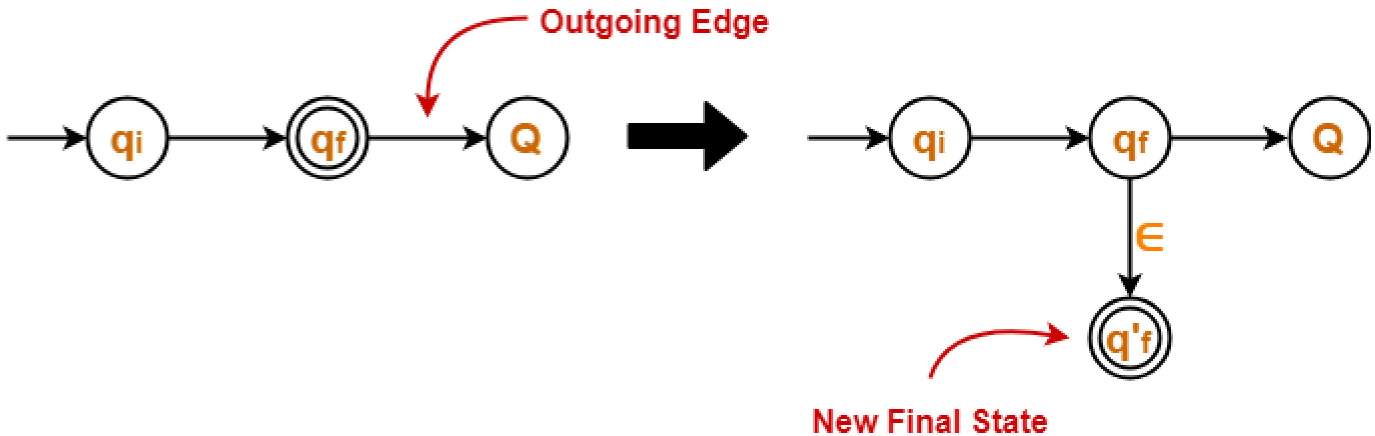


Step-03:

- The final state of the DFA must not have any outgoing edge.

- If there exists any outgoing edge from the final state, then create a new final state having no outgoing edge from it.

- Example



Step-04:

- Eliminate all the intermediate states one by one.
- These states may be eliminated in any order.

In the end,

- Only an initial state going to the final state will be left.
- The cost of this transition is the required regular expression.

NOTE

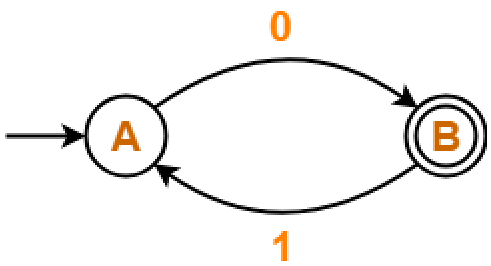
This method can be applied to any finite automata.

(NFA, ∈-NFA, DFA etc)

State Elimination Method

Exercise

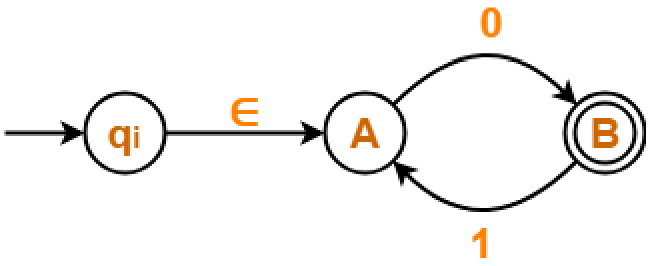
- Find regular expression for the following DFA-



Step-01:

- Initial state A has an incoming edge.
- So, we create a new initial state q_i.

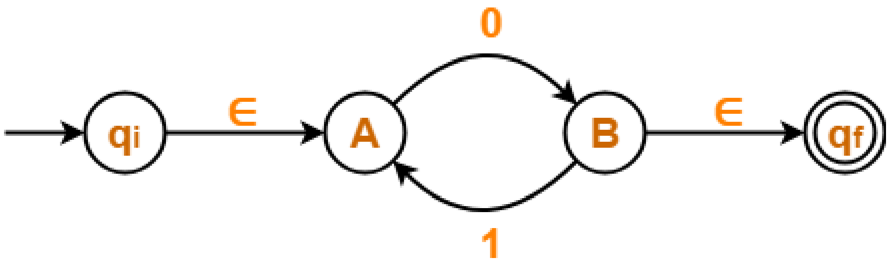
The resulting DFA is-



Step-02:

- Final state B has an outgoing edge.
- So, we create a new final state q_f.

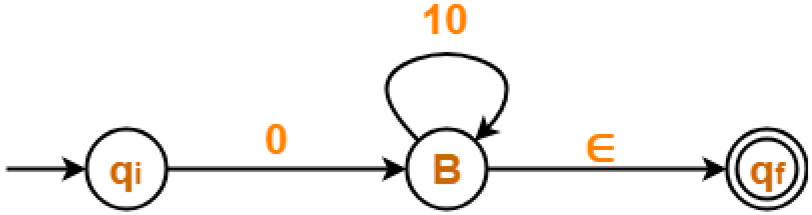
The resulting DFA is-



Step-03:

- Now, we start eliminating the intermediate states.
  - First, let us eliminate state A.
  - There is a path going from state q_i to state B via state A.
  - So, after eliminating state A, we put a direct path from state q_i to state B having cost $\in.0 = 0$
  - There is a loop on state B using state A.
  - So, after eliminating state A, we put a direct loop on state B having cost 1.0 = 10.

Eliminating state A, we get-



Step-04:

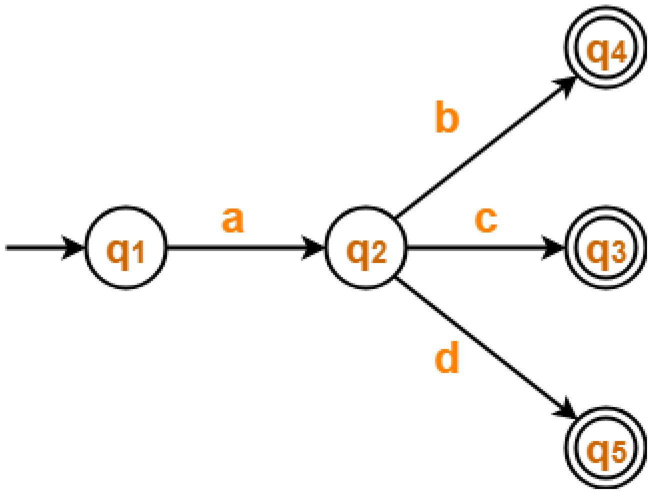
- Now, let us eliminate state B.
  - There is a path going from state q_i to state q_f via state B.
  - So, after eliminating state B, we put a direct path from state q_i to state q_f having cost $0.(10)^*.\in = 0(10)^*$

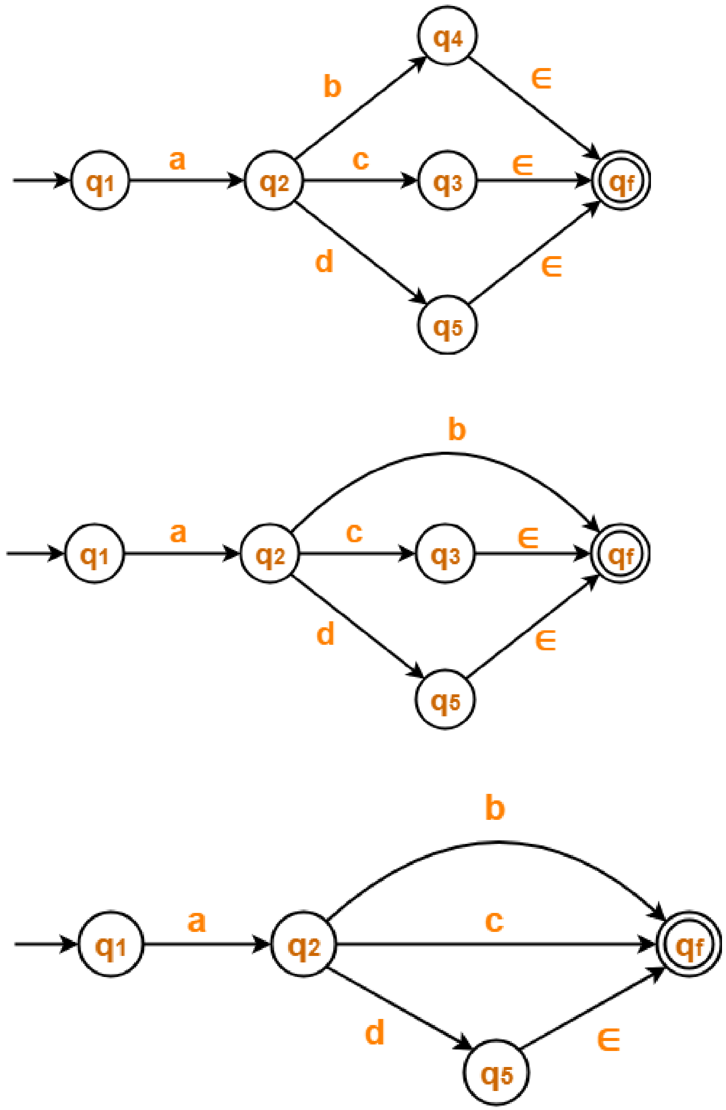
- Eliminating state B, we get-

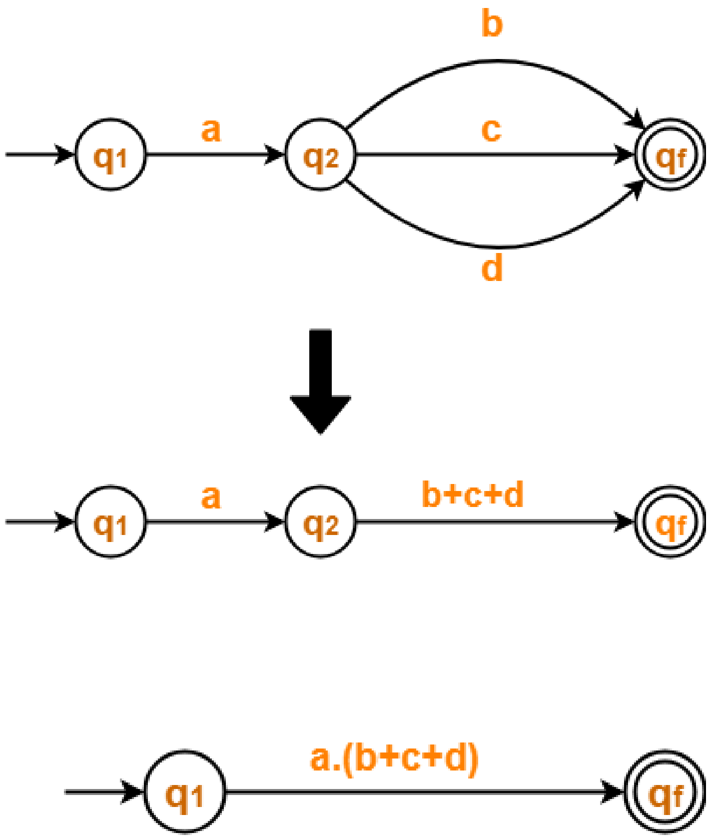
- So, Now from here
  - **Regular Expression = 0(10)***

Exercise

- Find regular expression for the following DFA-







Regular Expression = a(b+c+d)