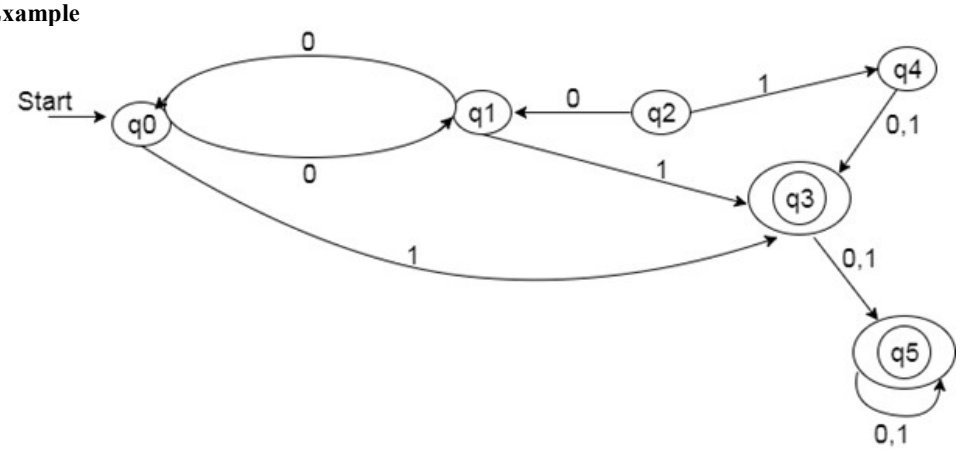
Reduction of the number of states in a finite automaton (Minimization of DFA).
Minimization of DFA means reducing the number of states from given DFA.

Step 1	Remove all the states that are unreachable from the initial state via any set of the transition of DFA.
Step 2	Draw the transition table for all pair of states.
Step 3	Now split the transition table into two tables T1 and T2. T1 contains all final states, and T2 contains non-final states.
Step 4	Find similar rows from T1 such that: 1. δ (q, a) = p 2. δ (r, a) = p
Step 5	Repeat step 3 until we find no similar rows available in the transition table T1.
Step 6	Repeat step 3 and step 4 for table T2 also.
Step 7	Now combine the reduced T1 and T2 tables. The combined transition table is the transition table of minimized DFA.

Example



Solution:

Step 1: In the given DFA, q2 and q4 are the unreachable states so remove them.
Step 2: Draw the transition table for the rest of the states.

State	0	1
→q0	q1	q3
q1	q0	q3
*q3	q5	q5
*q5	q5	q5

Step 3: Now divide rows of transition table into two sets as:

1. One set contains those rows, which start from non-final states:

State	0	1
→q0	q1	q3
q1	q0	q3

2. Another set contains those rows, which starts from final states.

State	0	1
*q3	q5	q5
*q5	q5	q5

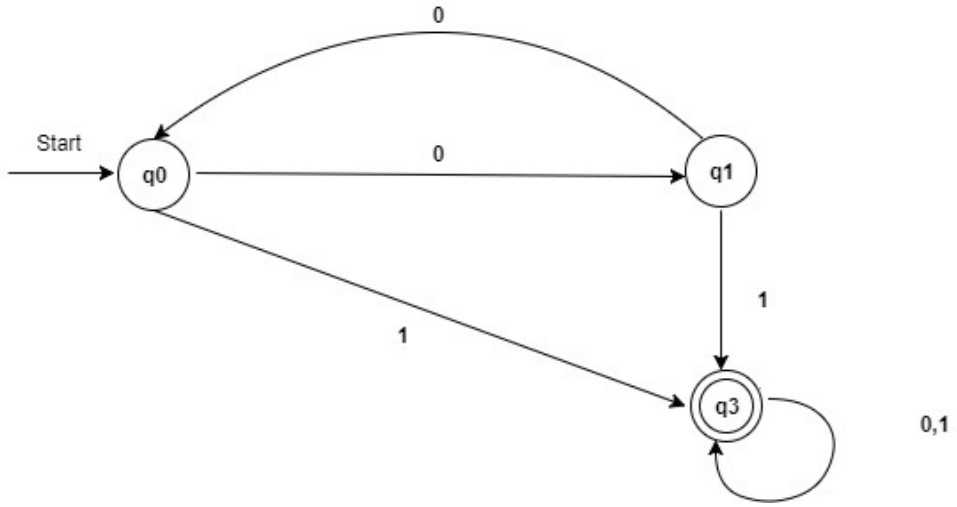
Step 4: Set 1 has no similar rows so set 1 will be the same.

Step 5: In set 2, row 1 and row 2 are similar since q3 and q5 transit to the same state on 0 and 1. So skip q5 and then replace q5 by q3 in the rest.

State	0	1
*q3	q3	q3

Step 6: Now combine set 1 and set 2 as:

State	0	1
→q0	q1	q3
q1	q0	q3
*q3	q3	q3



Regular Languages and Regular Grammar
Concept of languages and grammar. Regular expressions. Connection between regular expressions and regular languages.
Regular grammars
Regular grammar generates regular language. They have a single non-terminal on the left-hand side and a right-hand side consisting of a single terminal or single terminal followed by a non-terminal.

The productions must be in the form:

A ⇢ xB
A ⇢ x
A ⇢ Bx
where A, B ∈ Variable(V) and x ∈ T* i.e. string of terminals.

Types of regular grammar (type-3 grammar):

Left Linear grammar(LLG)
In LLG, the productions are in the form if all the productions are of the form
A ⇢ Bx
A ⇢ x
where A, B ∈ V and x ∈ T*

Right linear grammar(RLG)
In RLG, the productions are in the form if all the productions are of the form
A ⇢ xB
A ⇢ x
where A, B ∈ V and x ∈ T*

The language generated by regular grammar (type-3 grammar) is a regular language, for which a FA can be designed.