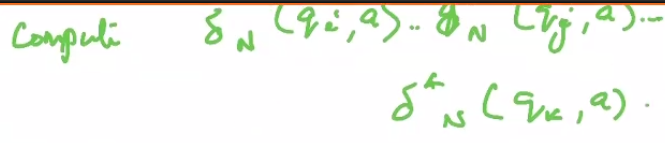
**Conversion from NFA to DFA**

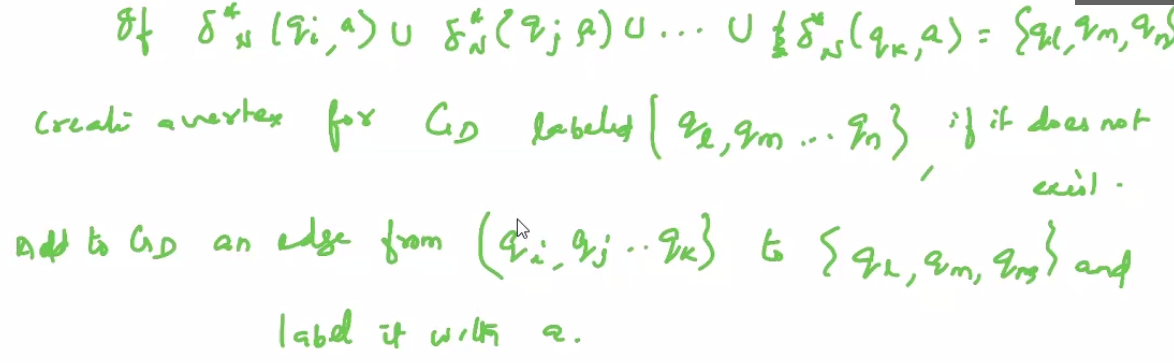
Every NFA has an equivalent DFA structure

Algorithm:

1. Create a graph G with vertex q0 as initial vertex
2. Repeat the following until no more edges remain in the NFA:

Take any vertex {qi,qj...qk} of G so that it has no outgoing edge for some input symbol a belonging to character set Sigma.

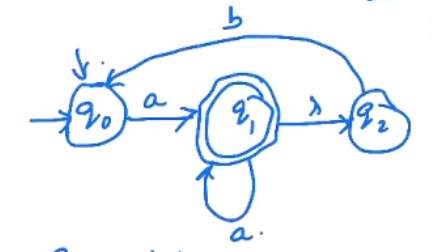




1. Every state q of G whose label contains qf (subset of F, i.e. qf is a final state) is labeled as the final state vertex of G
2. If the NFA accepts Lambda, the vertex {q0} in G is also taken as a final vertex.

Eg

1)



If Initial symbol

Q0 = a => {q1, q2}

Q0 = b => {}

Now for {q1,q2}, there is no outgoing edge

So we will compute delta(q1,a) and delta(q2,a) and take their union

Delta(q1, a) = {q1, q2}

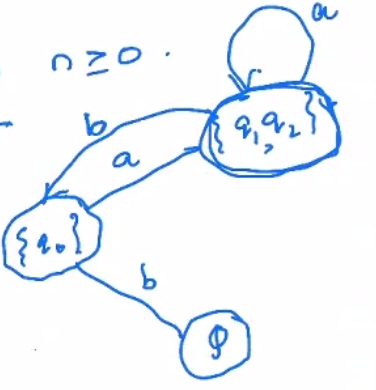
Delta(q2, a) = {}

Thus, Delta({q1,q2}, a) = {q1,q2}

Similarly for symbol b

Delta({q1,q2}, b) = {q0}

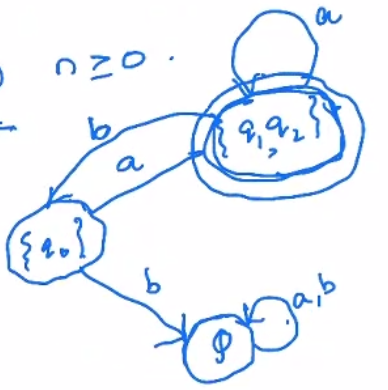
Now the graph G is



Now select Phi {} as the state

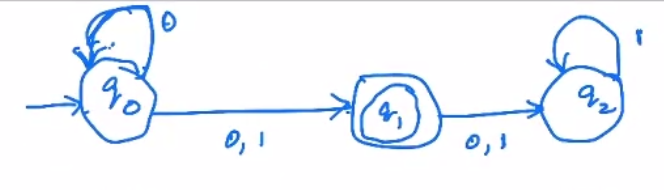
Delta({}, b) = {}

Final graph G:



Total no of possible states in NFA is 2^q. DFA will be just some possible combination of states of the NFA, so it is always possible to construct an equivalent DFA for every NFA.

Eg 2)



The corresponding DFA

