**NFA TO DFA**

Code:

import pandas as pd

# Taking NFA input from User

nfa = {}

n = int(input("No. of states : "))  # Enter total no. of states

t = int(input("No. of input symbols : "))  # Enter total no. of transitions/paths eg: a,b so input 2 for a,b,c input 3

for i in range(n):

    state = input("state name : ")  # Enter state name eg: A, B, C, q1, q2 ..etc

    nfa[state] = {}  # Creating a nested dictionary

    for j in range(t):

        path = input("symbol : ")  # Enter path eg : a or b in {a,b} 0 or 1 in {0,1}

        print("Enter end state from state {} for the input symbol {} : ".format(state, path))

        reaching\_state = [x for x in input().split()]  # Enter all the end states that

        nfa[state][path] = reaching\_state  # Assigning the end states to the paths in dictionary

print("\nNFA :- \n")

print(nfa)  # Printing NFA

print("\nPrinting NFA table :- ")

nfa\_table = pd.DataFrame(nfa)

print(nfa\_table.transpose())

print("Enter final state of NFA : ")

nfa\_final\_state = [x for x in input().split()]  # Enter final state/states of NFA

###################################################

new\_states\_list = []  # holds all the new states created in dfa

dfa = {}  # dfa dictionary/table or the output structure we needed

keys\_list = list(

    list(nfa.keys())[0])  # conatins all the states in nfa plus the states created in dfa are also appended further

path\_list = list(nfa[keys\_list[0]].keys())  # list of all the paths eg: [a,b] or [0,1]

###################################################

# Computing first row of DFA transition table

dfa[keys\_list[0]] = {}  # creating a nested dictionary in dfa

for y in range(t):

    var = "".join(nfa[keys\_list[0]][

                      path\_list[y]])  # creating a single string from all the elements of the list which is a new state

    dfa[keys\_list[0]][path\_list[y]] = var  # assigning the state in DFA table

    if var not in keys\_list:  # if the state is newly created

        new\_states\_list.append(var)  # then append it to the new\_states\_list

        keys\_list.append(var)  # as well as to the keys\_list which contains all the states

###################################################

# Computing the other rows of DFA transition table

x = len(new\_states\_list)

while x != 0:  # consition is true only if the new\_states\_list is not empty

    dfa[new\_states\_list[0]] = {}  # taking the first element of the new\_states\_list and examining it

    for \_ in range(len(new\_states\_list[0])):

        for i in range(len(path\_list)):

            temp = []  # creating a temporay list

            for j in range(len(new\_states\_list[0])):

                temp += nfa[new\_states\_list[0][j]][path\_list[i]]  # taking the union of the states

            s = ""

            s = s.join(temp)  # creating a single string(new state) from all the elements of the list

            if s not in keys\_list:  # if the state is newly created

                new\_states\_list.append(s)  # then append it to the new\_states\_list

                keys\_list.append(s)  # as well as to the keys\_list which contains all the states

            dfa[new\_states\_list[0]][path\_list[i]] = s  # assigning the new state in the DFA table

    new\_states\_list.remove(new\_states\_list[0])  # Removing the first element in the new\_states\_list

    x = x-1

print("\nDFA :- \n")

print(dfa)  # Printing the DFA created

print("\nPrinting DFA table :- ")

dfa\_table = pd.DataFrame(dfa)

print(dfa\_table.transpose())

dfa\_states\_list = list(dfa.keys())

dfa\_final\_states = []

for x in dfa\_states\_list:

    for i in x:

        if i in nfa\_final\_state:

            dfa\_final\_states.append(x)

            break

print("\nFinal states of the DFA are : ", dfa\_final\_states)  # Printing Final states of DFA

Output:



