EXP:4: CONVERTING NFA TO DFA

AIM: To Design a code to convert NFA to DFA.

LANGUAGE USED: Python 3

ALGORITHM/PROCEDURE: -

Suppose there is an NFA N < Q, ∑, q0, δ, F > which recognizes a language L. Then the DFA D < Q', ∑, q0, δ', F' > can be constructed for language L as:

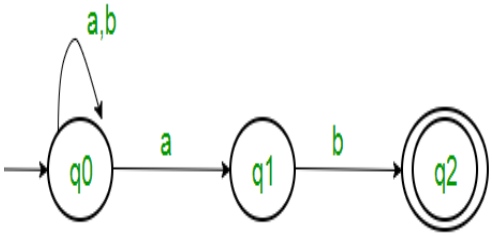
Step 1: Initially Q' = ϕ.

Step 2: Add q0 to Q'.

Step 3: For each state in Q', find the possible set of states for each input symbol using transition function of NFA. If this set of states is not in Q', add it to Q'.

Step 4: Final state of DFA will be all states with contain F (final states of NFA)

SPACE TREE DIAGRAM:



State	a	b
q0	q0,q1	q0
q1		q2
q2		

State	a	b
q0	{q0,q1}	q0

SOURCE CODE: -

```
import pandas as pd
nfa = {}
n = int(input("No. of states : "))
t = int(input("No. of transitions : "))
for i in range(n):
    state = input("state name : ")
    nfa[state] = {}
    for j in range(t):
        path = input("path : ")
        print("Enter end state from state {} travelling through path {} : ".format(state,path))
        reaching_state = [x for x in input().split()]
        nfa[state][path] = reaching_state
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()]
new_states_list = []
dfa = {}
keys_list = list(list(nfa.keys())[0])
path_list = list(nfa[keys_list[0]].keys())
dfa[keys_list[0]] = {}
for y in range(t):
    var = "".join(nfa[keys_list[0]][path_list[y]])
    dfa[keys_list[0]][path_list[y]] = var
    if var not in keys_list:
        new_states_list.append(var)
        keys_list.append(var)

while len(new_states_list) != 0:
    dfa[new_states_list[0]] = {}
    for _ in range(len(new_states_list[0])):
```

```
        for i in range(len(path_list)):
            temp = []
            for j in range(len(new_states_list[0])):
                temp += nfa[new_states_list[0][j]][path_list[i]]
            s = ""
            s = s.join(temp)
            if s not in keys_list:
                new_states_list.append(s)
                keys_list.append(s)
            dfa[new_states_list[0]][path_list[i]] = s
    new_states_list.remove(new_states_list[0])
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)
```

INPUT/OUTPUT:

```
No. of states : 4
No. of transitions : 2
state name : A
path : a
```

Enter end state from state A travelling through path a :

A B

path : b

Enter end state from state A travelling through path b :

A

state name : B

path : a

Enter end state from state B travelling through path a :

C

path : b

Enter end state from state B travelling through path b :

C

state name : C

path : a

Enter end state from state C travelling through path a :

D

path : b

Enter end state from state C travelling through path b :

D

state name : D

path : a

Enter end state from state D travelling through path a :

path : b

Enter end state from state D travelling through path b :

NFA :-

{'A': {'a': ['A', 'B'], 'b': ['A']}, 'B': {'a': ['C'], 'b': ['C']}, 'C': {'a': ['D'], 'b': ['D']}, 'D': {'a': [], 'b': []}}

Printing NFA table :-

	a	b
A	[A, B]	[A]
B	[C]	[C]
C	[D]	[D]

D [] []

Enter final state of NFA :

D

DFA :-

{'A': {'a': 'AB', 'b': 'A'}, 'AB': {'a': 'ABC', 'b': 'AC'}, 'ABC': {'a': 'ABCD', 'b': 'ACD'}, 'AC': {'a': 'ABD', 'b': 'AD'}, 'ABCD': {'a': 'ABCD', 'b': 'ACD'}, 'ACD': {'a': 'ABD', 'b': 'AD'}, 'ABD': {'a': 'ABC', 'b': 'AC'}, 'AD': {'a': 'AB', 'b': 'A'}}

Printing DFA table :-

a b

A AB A

AB ABC AC

ABC ABCD ACD

AC ABD AD

ABCD ABCD ACD

ACD ABD AD

ABD ABC AC

AD AB A

Final states of the DFA are : ['ABCD', 'ACD', 'ABD', 'AD']

RESULT: Thus, we have successfully implemented a code for converting NFA to DFA.