Exp 3: CONVERSION OF NFA TO DFA

Name-Husna Qasim

Reg no.-RA1911031010138

Class-CSE-IT (L2)

AIM: To write a program for converting NFA to DFA.

ALGORITHM:

1. Start
2. Get the input from the user
3. Set the only state in SDFA to “unmarked”.
4. while SDFA contains an unmarked state do:

a. Let T be that unmarked state

b. for each a in % do S = e-Closure(MoveNFA(T,a))

c. if S is not in SDFA already then, add S to SDFA (as an “unmarked” state)

d. Set MoveDFA(T,a) to S

5. For each S in SDFA if any s & S is a final state in the NFA then, mark S an a final state in the DFA
6. Print the result.
7. Stop the program

PROGRAM:

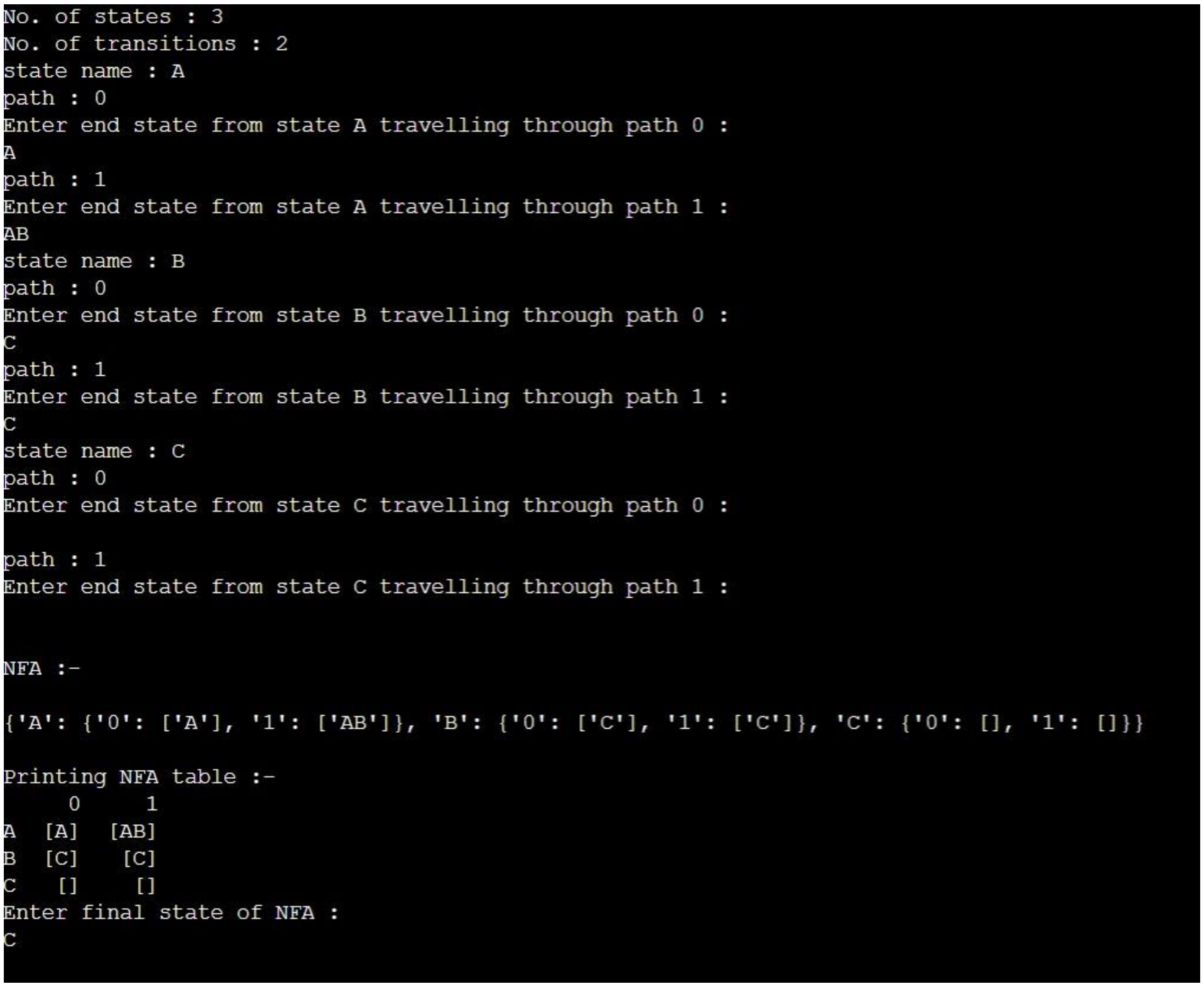
```
import pandas as pd
nfa = {}
```

```
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)
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)
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:
```

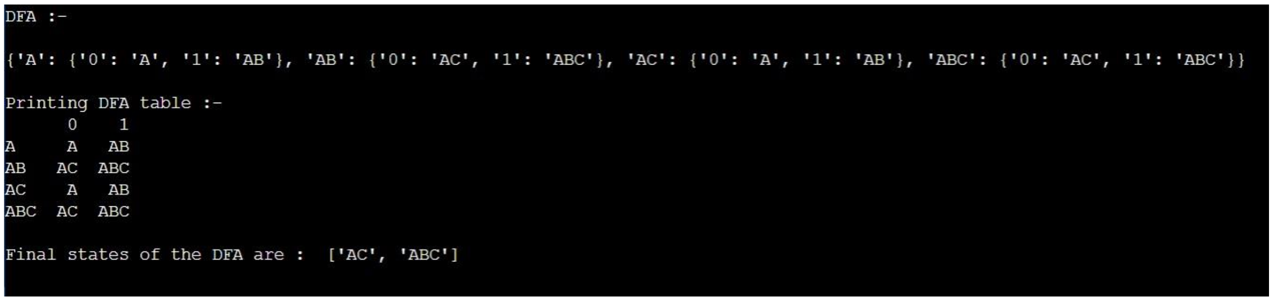
```
for i in x:         if i in
nfa_final_state:
            dfa_final_states.append(x)
break
print("\nFinal states of the DFA are : ", dfa_final_states)
```

INPUT :



```
No. of states : 3
No. of transitions : 2
state name : A
path : 0
Enter end state from state A travelling through path 0 :
A
path : 1
Enter end state from state A travelling through path 1 :
AB
state name : B
path : 0
Enter end state from state B travelling through path 0 :
C
path : 1
Enter end state from state B travelling through path 1 :
C
state name : C
path : 0
Enter end state from state C travelling through path 0 :

path : 1
Enter end state from state C travelling through path 1 :


NFA :-

{'A': {'0': ['A'], '1': ['AB']}, 'B': {'0': ['C'], '1': ['C']}, 'C': {'0': [], '1': []}}

Printing NFA table :-
     0     1
A  [A]  [AB]
B  [C]   [C]
C   []    []
Enter final state of NFA :
C
```

OUTPUT :



```
DFA :-

{'A': {'0': 'A', '1': 'AB'}, 'AB': {'0': 'AC', '1': 'ABC'}, 'AC': {'0': 'A', '1': 'AB'}, 'ABC': {'0': 'AC', '1': 'ABC'}}

Printing DFA table :-
      0    1
A     A   AB
AB   AC  ABC
AC    A   AB
ABC  AC  ABC

Final states of the DFA are :  ['AC', 'ABC']
```

RESULT :

The given NFA was converted to a DFA using python successfully.