EXPERIMENT - VII
MINIMIZATION OF DFA

September 15, 2020

ADITHYA D RAJAGOPAL
ROLL NO : 9
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING TRIVANDRUM

AIM

To write a program to minimize any given DFA.

THEORY

Deterministic Finite Automata

A deterministic finite automata (DFA) N over an alphabet set Σ is a structure of the form:

$$N = (Q, s, \delta, F)$$

where,

- Q is a finite set of states.
- s ∈ Q is the start state.
- Δ : Q X Σ → Q is the transition function.
- F ⊆ Q is the set of final states.

Two states p,q ∈ Q of a DFA M are said to be equivalent, if for every string x ∈ Σ^* the state that M reaches from p given x is accepting if and only if the state that M reaches from q given x is accepting.

Minimizing States in a DFA

The number of states in a given DFA A = (Q, s, δ, F) over an alphabet set Σ can be minimized using the following steps:

- Throw away the states which are not reachable from the start state s.
- Collapse the equivalent states.

The equivalent states can be found out using the state minimization algorithm.

State Minimization Algorithm

INPUT : A = (Q, s, δ, F)
OUTPUT : $\approx$ for A

1. Create a table of size n x n where n is the number of states in the DFA.
2. Initialize entry for each pair in table to "unmarked".
3. Mark (p,q) if p $\in$ F and q $\notin$ F or vice-versa.
4. Scan table entries and repeat till no more marks can be added :
   (i) If there exists unmarked (p,q) with a $\in$ A such that δ(p,a) and δ(q,a) are marked, then mark (p,q).
5. Return $\approx$ as: p $\approx$ q iff (p,q) is left unmarked in the table.

ALGORITHM

Algorithm 1 Algorithm to minimize any DFA

```
Start
Input the DFA (dfa).
Initialize an empty object of type DFA (minDfa).
minDfa.noofalphabets = dfa.noofalphabets
Initialize a matrix m of size dfa.noofstates x dfa.noofstates.
Set every cell of the matrix m to 0.
Initialize a flag variable f to 1.
for all state pairs (x,y) do
    if X is a final state an y is a non-final state or vice-versa then
        Set m[x][y]=1.
    end if
end for
while f ≠ 0 do
    Set f to 0.
    for i=0 to dfa.noofstates do
        for j=i+1 to dfa.noofstates do
            if m[i][j]=0 then
                if for any u ∈ Σ , m[dfa.transitiontable[i][u]][dfa.transitiontable[j][u]]=1 then
                    Set m[i][j]=1 and f=1.
                end if
            end if
        end for
    end for
end while
Represent those pair of states (a,b) which has m[a][b]=0 by a single state a in minDFA.
Stop
```

SOURCE CODE

```
def MinimizeDFA(dfa):
        print()
        print("Generating minimized DFA...")
        print()
        m=[[0 for _ in range(n)] for _ in range(n)]
        eq=[[] for _ in range(n)]
        flag=1
        f=dfa[3]
        tt=dfa[2]
        for i in range(n):
                if i not in f:
                        for j in f:
                                m[i][j]=1
                                m[j][i]=1
        while flag!=0:
                flag=0
                for i in range(n):
                        for j in range(i+1,n):
                                if m[i][j]==0:
                                        for u in range(s):
                                                if m[tt[i][u]][tt[j][u]]==1:
                                                        m[i][j]=1
                                                        m[j][i]=1
                                                        flag=1
        for i in range(n):
                for j in range(n):
                        if m[i][j]==0:
                                eq[i].append(j)
        delta=[]
        for i in eq:
                if i not in delta:
                        delta.append(i)
        eq=delta
```

```
        delta=[]
        for state in eq:
                trans=[]
                for j in range(s):
                        t=tt[state[0]][j]
                        for loc in eq:
                                if t in loc:
                                        trans.append(loc[0])
                delta.append(trans)
        print("Start State of minimized DFA : 0")
        print()
        print("State",end="\t")
        for i in range(s-1):
                print(sigma[i],end="\t")
        print(sigma[s-1])
        for i in range(len(delta)):
                print(i,end="\t")
                for j in range(s-1):
                        print(delta[i][j],end="\t")
                print(delta[i][s-1])
        print()
        f=[]
        for state in eq:
                if state[0] in dfa[3]:
                        f.append(state[0])
        print("Final States of minimized DFA : ",end="")
        for i in range(len(f)-1):
                print(f[i],end=",")
        print(f[len(f)-1])

global s
global n
global sigma
n=int(input("Enter number of states in the DFA:"))
Q=[i for i in range(n)]
```

```
s=int(input("Enter number of input symbols:"))
print("State",end="\t")
x=('a','b','c','d','e','f','g','h','i','j')
sigma=[]
for i in range(s):
        print(x[i],end="\t")
        sigma.append(x[i])
print()
dfa=[]
dfa.append(Q)
delta=[]
for i in range(n):
        print(i,end="\t")
        t=input().split()
        row=[]
        for k in t:
                row.append(int(k))
        delta.append(row)
S=int(input("Enter the start state of the DFA:"))
f=input("Enter the final states of the DFA:").split(",")
F=[]
for i in range(len(f)):
        F.append(int(f[i]))
dfa.append(S)
dfa.append(delta)
dfa.append(F)
MinimizeDFA(dfa)
```

SAMPLE OUTPUT

```
user@adithya-d-rajagopal:~/s7/cd$ python3 p7.py
Enter number of states in the DFA:6
Enter number of input symbols:2
State   a       b
0       1       2
1       1       2
2       3       4
3       5       3
4       1       2
5       5       3
Enter the start state of the DFA:0
Enter the final states of the DFA:1,3,4,5

Generating minimized DFA...

Start State of minimized DFA : 0

State   a       b
0       1       2
1       1       2
2       3       1
3       3       3

Final States of minimized DFA : 1,3
user@adithya-d-rajagopal:~/s7/cd$
```

RESULT

A program to minimize any DFA has been implemented using Python and the outputs have been verified.