NIKIT GOKHE
Class – SY Comp D1
Roll No. 224024
GR No. 21810522

Assignment No 1

AIM:

Write a Program to convert Non-deterministic finite automaton (NFA) to Deterministic finite automaton (DFA)

Algorithm

Input − An NDFA Table

Output − An equivalent DFA Table

Step 1 − From the given transition table of NFA, Create a blank state table under possible input alphabets for the equivalent DFA.

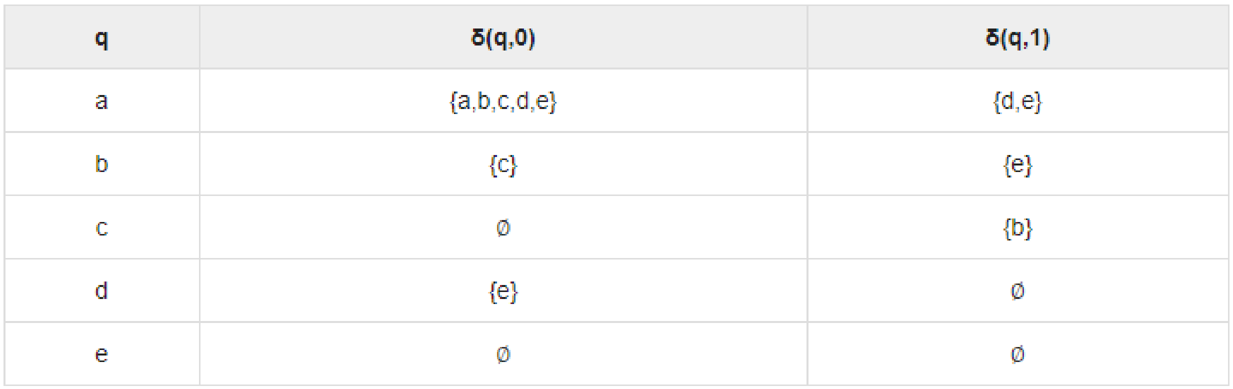
Step 2 − Mark the start state of the DFA by q0 (Same as the NDFA).

Step 3 − Find out the combination of States {Q_0, Q_1,... , Q_n} for each possible input alphabet.

Step 4 − Each time we generate a new DFA state under the input alphabet columns, we have to apply step 4 again, otherwise go to step 6.

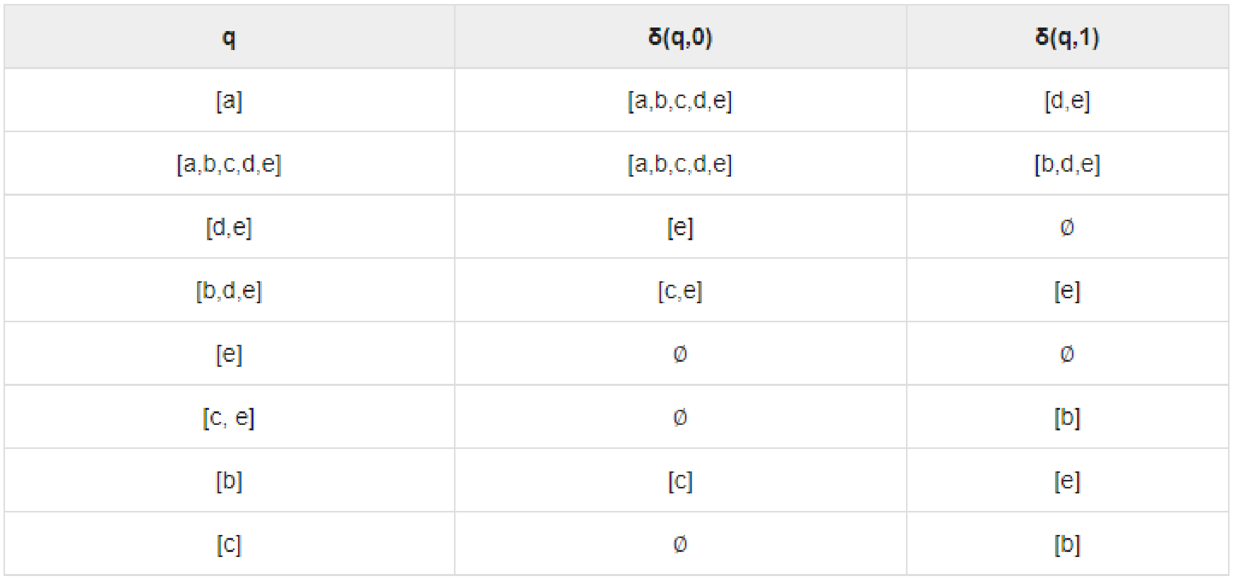
Step 5 − The states which contain any of the final states of the NDFA are the final states of the equivalent DFA.

Given Transition table of NFA



q	δ(q,0)	δ(q,1)
a	{a,b,c,d,e}	{d,e}
b	{c}	{e}
c	∅	{b}
d	{e}	∅
e	∅	∅

Using the above algorithm, we find its equivalent DFA. The state table of the DFA is shown in below.



q	δ(q,0)	δ(q,1)
[a]	[a,b,c,d,e]	[d,e]
[a,b,c,d,e]	[a,b,c,d,e]	[b,d,e]
[d,e]	[e]	∅
[b,d,e]	[c,e]	[e]
[e]	∅	∅
[c, e]	∅	[b]
[b]	[c]	[e]
[c]	∅	[b]

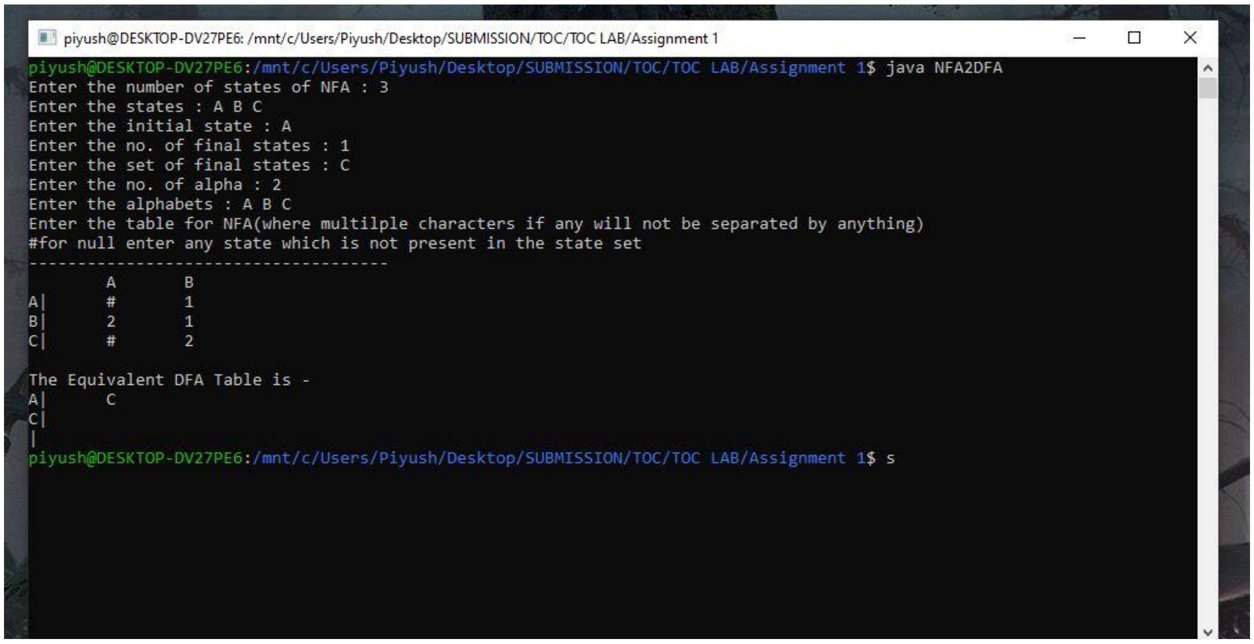
```
import java.util.*;

public class NFA2DFA {

        public static void main(String[] args)

        {

                Scanner sc = new Scanner(System.in);
```

```
System.out.print("Enter the number of states of NFA : ");
int nos = sc.nextInt();
System.out.print("Enter the states : ");
char states_ar[] = new char[nos];
HashMap<Character, Integer> states = new HashMap<Character, Integer>();
for(int i=0;i<nos;i++)
{
        char ch = sc.next().charAt(0);
        states_ar[i] = ch;
        states.put(ch, i);
}
System.out.print("Enter the initial state : ");
char ini_state = sc.next().charAt(0);
System.out.print("Enter the no. of final states : ");
int nof = sc.nextInt();
System.out.print("Enter the set of final states : ");
HashMap<Character, Integer> fin_states = new HashMap<Character, Integer>();
for(int i=0;i<nof;i++)
fin_states.put(sc.next().charAt(0), 1);
System.out.print("Enter the no. of alpha : ");
int noa = sc.nextInt();
System.out.print("Enter the alphabets : ");
char alpha[] = new char[noa];
for(int i=0;i<noa;i++)
        alpha[i] = sc.next().charAt(0);
Queue<String> q = new LinkedList<>();
HashMap<String, Integer> map = new HashMap<String, Integer>();
q.add(ini_state+"");
map.put(ini_state+"", 1);
```

```
                    System.out.println("Enter the table for NFA(where multilple characters if any will not be
separated by anything)");
                    System.out.println("#for null enter any state which is not present in the state set");
                    String table_NFA[][] = new String[nos][noa];
                    System.out.print("------------------------------------\n\t");
                    for(int al=0;al<noa;al++)
                              System.out.print(alpha[al]+"\t");
                    System.out.println();
                    for(int i=0;i<nos;i++)
                    {
                              System.out.print(states_ar[i]+"|\t");
                              for(int j=0;j<noa;j++)
                              {
                                        table_NFA[i][j] = sc.next();
                                        //System.out.print("\t");
                              }
                    }
                    System.out.println("\nThe Equivalent DFA Table is - ");
                    String table_DFA[][] = new String[1000][noa];
                    String final_states_DFA[] = new String[1000];
                    int top_finalstates = 0;
                    HashMap<Character, Integer> stts;
                    for(int i=0;q.size() > 0;i++)


                              String cur_state = q.remove();
                              System.out.print(cur_state+"|\t");
                              for(int j=0;j<noa;j++)
                              {
                                        table_DFA[i][j] = "";
                                        if(cur_state.length() == 1 && !states.containsKey(cur_state.charAt(0)))
                                        {
                                                table_DFA[i][j] = cur_state;
                                                if(!map.containsKey(table_DFA[i][j]))
                                                {
                                                        q.add(table_DFA[i][j]);
                                                        map.put(table_DFA[i][j], 1);
                                                }
                                                //System.out.println("for check"+(table_DFA[i][j].charAt(0)));
                                                if(fin_states.containsKey((table_DFA[i][j].charAt(0))))
                                                        final_states_DFA[top_finalstates++] = table_DFA[i][j];
                                                System.out.print(table_DFA[i][j]+"\t");
                                                continue;
                                        }
                                        int flag_for_final = 0;
                                        stts = new HashMap<Character, Integer>();
                                        for(int k=0;k<cur_state.length();k++)
                                        {


                                                if(!states.containsKey(cur_state.charAt(k)))
                                                        continue;
                                                if(fin_states.containsKey(cur_state.charAt(k)))
                                                        flag_for_final = 1;
                                                for(int
ch=0;ch<table_NFA[states.get(cur_state.charAt(k))][j].length();ch++)
                                                {
        if(!stts.containsKey(table_NFA[states.get(cur_state.charAt(k))][j].charAt(ch)) &&
states.containsKey(table_NFA[states.get(cur_state.charAt(k))][j].charAt(ch)))
                                                        {

table_DFA[i][j] += table_NFA[states.get(cur_state.charAt(k))][j].charAt(ch);//null state is appearing i.e AC
```

```
                    stts.put(table_NFA[states.get(cur_state.charAt(k))][j].charAt(ch), 1);
                                        }
                                   }



                              }
                              if(!map.containsKey(table_DFA[i][j]))
                              {
                                   q.add(table_DFA[i][j]);
                                   map.put(table_DFA[i][j], 1);
                              }
                              if(flag_for_final == 1)
                                   final_states_DFA[top_finalstates++] = table_DFA[i][j];
                              System.out.print(table_DFA[i][j]+"\t");
                         }
                         System.out.println();
                    }
                    /*System.out.print("Initial state : "+ini_state+"\nFinal States : ");
                    for(int fs=0;fs<top_finalstates;fs++)
                         System.out.print(final_states_DFA[fs]);
                    System.out.println(top_finalstates);*/
          }
}
```

OUTPUT:


piyush@DESKTOP-DV27PE6: /mnt/c/Users/Piyush/Desktop/SUBMISSION/TOC/TOC LAB/Assignment 1

```
piyush@DESKTOP-DV27PE6:/mnt/c/Users/Piyush/Desktop/SUBMISSION/TOC/TOC LAB/Assignment 1$ java NFA2DFA
Enter the number of states of NFA : 3
Enter the states : A B C
Enter the initial state : A
Enter the no. of final states : 1
Enter the set of final states : C
Enter the no. of alpha : 2
Enter the alphabets : A B C
Enter the table for NFA(where multilple characters if any will not be separated by anything)
#for null enter any state which is not present in the state set
-------------------------------------
        A       B
A|      #       1
B|      2       1
C|      #       2

The Equivalent DFA Table is -
A|      C
C|
|
piyush@DESKTOP-DV27PE6:/mnt/c/Users/Piyush/Desktop/SUBMISSION/TOC/TOC LAB/Assignment 1$ s
```