MINIMIZATION OF DFA

Theory of Computation Assignment

-Kapeel Suryavanshi (BT16CSE084)

Algorithm:

1. I removed unreachable states from the DFA.
2. I made two sets, one of non-terminal states and the other one of final or terminal states.
3. Then, I pick 1 set and proceed by taking pair of transitions of set from Transition Table into consideration and start combining the states that are indistinguishable. (Two states 'A' and 'B' are indistinguishable when δ(A,X) and δ(B,X) where X is an input symbol, both lie in the same set, and are distinguishable if they do not lie in the same set).
4. If a state from the set in consideration is distinguishable, I split the set.
5. I repeat the above process for each set over each input symbol. I stop when I see that no new sets are formed for each of the input symbol.

Implementation:

- **Variables and Data Type used:**

Data Type	Variable Name	Description
map<string,vector<string> >	table	A map with key value as the state and vector as it's transitions. This is my *Transition Table.*
vector<string>	final_states	Vector storing the final(terminal) states of the DFA.
vector<string>	nonterm_states	Vector storing the non-terminal states of DFA.
vector<string>	input_symbols	Vector storing the input symbols.
map<string,int>	m	Map keeping track of the set (set number-int) in which the corresponding state(key-string) lies (used in function Minimize_DFA).
vector<vector<string> >	set	2D vector of strings storing the sets for minimizing DFA.

- **Functions and their description:**
  - ❖ **int main() :**
    Driver Function. It takes input of DFA from a file, stored them and calls appropriate functions for minimizing DFA
  - ❖ **void Minimize_DFA(***map<string,vector<string> > &table, vector<string> &nonterm_states, vector<string> &final_states,int inp_sym***)** :
    This is the actual function that performs the minimization. It first calls remove_unreachable_states() that removes all unreachable states. Then, it also does the job of making sets and checks whether the sets in a set are distinguishable or not. Here, for each input symbol, rather then considering pairs of transition of states, whole set as a whole is considered. A map is maintained which tells in which set a set lies. Next, I iterate through a set, check in which set the states (of current set) lies. If a state is distinguishable, I update the map and then call update_set(). Lastly, once I have the final sets, I start combining the states that lie in same set, so as to make one state and then call update_table_n_sets().
  - ❖ **void remove_unreachable_states(***map<string,vector<string> > &table,vector<string> &nonterm_states, vector<string> &final_states***) :**

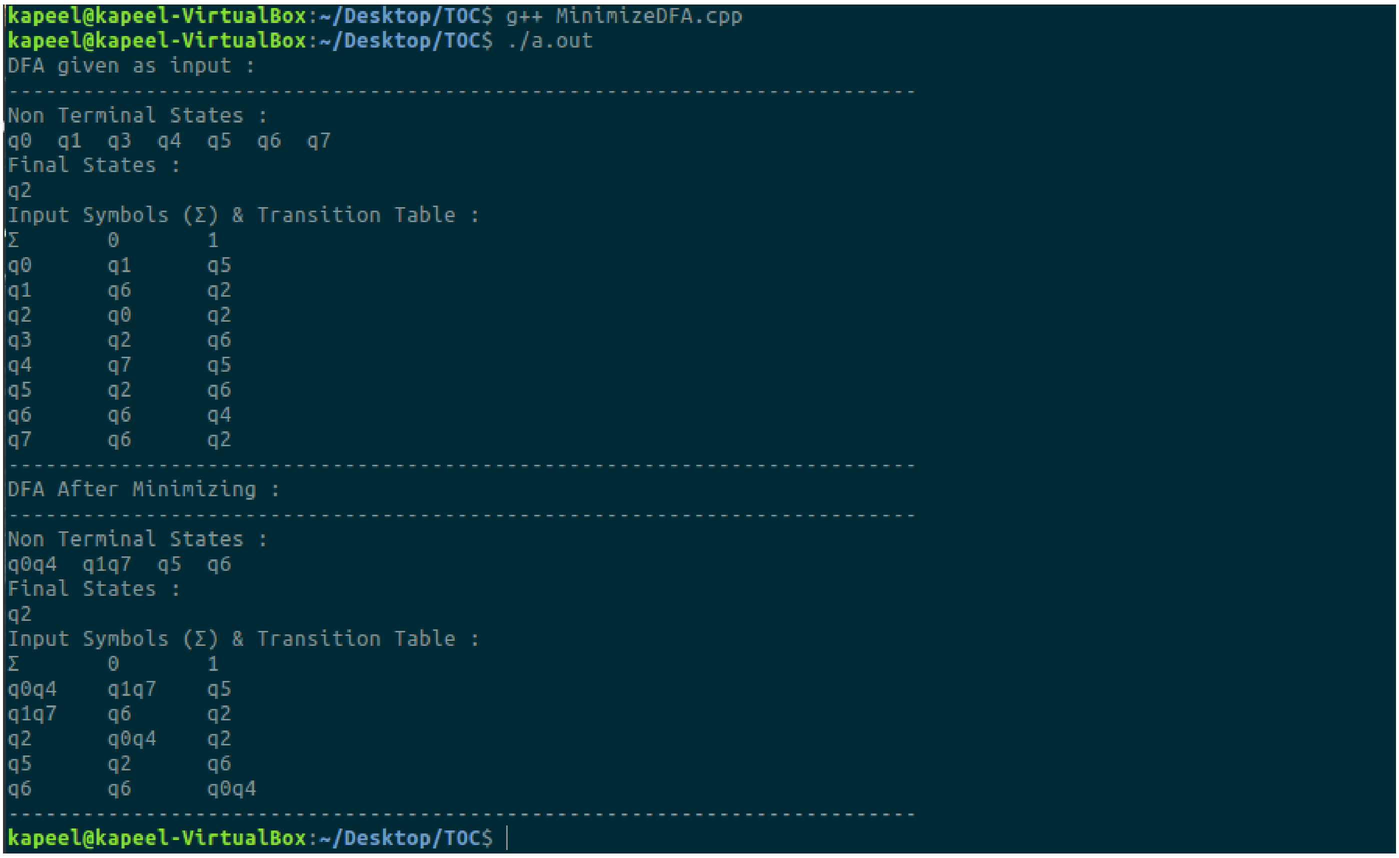
This function removes the states that are not reachable at all by using BFS traversal.

- **void update_set(***vector<vector<string> > &set, map<string,int> &m, int max_num_set,int &curr_num_set***) :**
  This function updates the sets of states that are distinguishable in their current state. Creates new set for such states.
- **void update_table_n_states(***vector<string> &comb_set, map<string,int> &m, map<string,vector<string> > &table, vector<string> &nonterm_states, vector<string> &final_states***):**
  This function updates the transition table and the vectors of non-terminal and terminal states. It takes a vector of string consisting of the combined sets of states and updates the table and vectors accordingly.
- **void print(***map<string,vector<string> > &table,vector<string> &nonterm_states, vector<string> &final_states,vector<string> &input_symbols***):**
  Utility function to print the parameters of DFA.

- **<u>Input:</u>**
  - Input is taken from file.
  - First line of input, starts with NT, depicting non-terminal states.
  - Second line of input starts with F, depicting final or terminal states.
  - Third line consists of input symbols.
  - Rest of the lines comprise of Transition Table.
  - Example :

    ```
    NT = q0 q1 q3 q4 q5 q6 q7
    F = q2
    Σ 0 1
    q0 q1 q5
    q1 q6 q2
    q2 q0 q2
    q3 q2 q6
    q4 q7 q5
    q5 q2 q6
    q6 q6 q4
    q7 q6 q2
    ```

- **Output:**
    - First, the DFA that is given as input is printed on the terminal.
    - Then, the DFA obtained after minimization is printed on terminal.
    - Output of my program when above input was given is :



```
kapeel@kapeel-VirtualBox:~/Desktop/TOC$ g++ MinimizeDFA.cpp
kapeel@kapeel-VirtualBox:~/Desktop/TOC$ ./a.out
DFA given as input :
-------------------------------------------------------------------------
Non Terminal States :
q0  q1  q3  q4  q5  q6  q7
Final States :
q2
Input Symbols (Σ) & Transition Table :
Σ       0       1
q0      q1      q5
q1      q6      q2
q2      q0      q2
q3      q2      q6
q4      q7      q5
q5      q2      q6
q6      q6      q4
q7      q6      q2
-------------------------------------------------------------------------
DFA After Minimizing :
-------------------------------------------------------------------------
Non Terminal States :
q0q4  q1q7  q5  q6
Final States :
q2
Input Symbols (Σ) & Transition Table :
Σ       0       1
q0q4    q1q7    q5
q1q7    q6      q2
q2      q0q4    q2
q5      q2      q6
q6      q6      q0q4
-------------------------------------------------------------------------
kapeel@kapeel-VirtualBox:~/Desktop/TOC$
```