CS410 PROJECT 1 PHASE 2

Berke Bilensoy S021407

Part 1 (this part is same with first submission check part 2)

In this project, we are expected to implement the algorithm which converts given NFAs to DFAs. NFA inputs are given to us as txt files. This algorithm is taking inputs from txt files and creates transition table. For these tables, this algorithm uses matrixes as data structure.

I will use the first example in the homework to give an example.

M=(Q, Σ, δ, START, FINAL) → M=(Q, Σ, δ, {A},{C})

This is how we represent automata. For visualize it, we need to read every line.

ALPHABET
0 → Σ = {0,1}
1

STATES
A
B → Q = {A, B, C}
C

START
A

FINAL
C

TRANSITIONS

A 0 A
A 1 A
A 1 B
B 1 C

→ δ →

This is the transition table algorithm creates

	0	1
A	{A}	{A,B}
B	Ø	{C}
C	Ø	Ø

After that, algorithm takes this table to create the DFA. I showed in under how algorithm checks the transition table and creates the new one.

Steps:

1.

	0	1
A	{A}	{A,B}

2.

	0	1
A	{A}	{A,B}
{A,B}	{A}	{A,B,C}

3.

	0	1
A	**{A}**	**{A,B}**
{A,B}	**{A}**	**{A,B,C}**
{A,B,C}	**{A}**	**{A,B,C}**

After all of this, algorithm creates txt file as a output in the same format with the information of this table on above.

Part 2

I wrote the code by using java. Firstly, I used scanner for read the file and put it into a ArrayList. When I put all items into array, I started to separate them to their own arraylists such as:

```
ArrayList<String> list = new ArrayList<String>();
ArrayList<String> alphabet = new ArrayList<String>();
ArrayList<String> states = new ArrayList<String>();
ArrayList<String> startState = new ArrayList<String>();
ArrayList<String> finalState = new ArrayList<String>();
ArrayList<String> transitions = new ArrayList<String>();
```

And everything been stored in these arrays. These are the outputs of the arrays I mentioned above:

```
[ALPHABET, 0, 1, STATES, A, B, C, START, A, FINAL, C, TRANSITIONS, A, 0, A, A, 1, A, A, 1, B, B, 1, C, END]
[0, 1]
[A, B, C]
[A]
[C]
[A, 0, A, A, 1, A, A, 1, B, B, 1, C]
```

Then I needed to create transmission table so I used 3 dimensional ArrayList:

```
  int alphabetc = alphabet.size();
      int statec = states.size();

      trans = new ArrayList[statec][alphabetc];
      for (int i = 0; i < statec; i++) {
         for (int j = 0; j < alphabetc; j++) {
            trans[i][j] = new ArrayList<>();
         }
      }
```

Then, I had to fill the transition table from transition ArrayList.
Transition ArrayList:

[A, 0, A, A, 1, A, A, 1, B, B, 1, C]

I created for and if statement for check and implement transition table.

```
for (int i = 0; i< transitions.size(); i++){
            if(i % 3 == 0){
```

Every three index is 1 transition, so I implemented an algorithm with mod of 3 (i % 3).
Such as:
[A, 0, A]
Mod of 3:
[A, 0, A]
 0 1 2

If mod of i is 0 it means, we found which row we need to put. If mod of i is 1 that means, we found the column and if mod of i is 2 it means we need to put that state to transition table.

Then I printed transition table to see it clearly with a print method.

This is the output for NFA:

```
   0   1
A  A |A B |
B     | C |
C     |   |
```

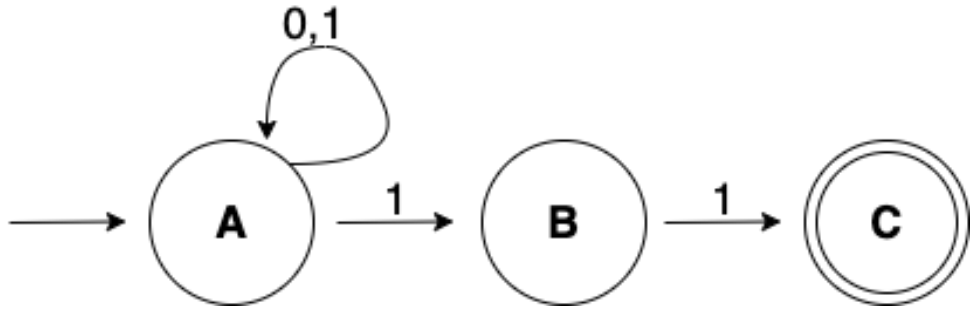
For transition of DFA, I thought to take these steps:

1. Create new 3-dimensional ArrayList named DFA.
2. Then, take the first row from the NFA transition table.
3. Create new ArrayList for the states which are created in that step.

4. Add new rows for those states. (with .add() method).
5. Check transition tables for new states and implement union algorithm (if there is duplicate or more from one state such as: {B, C, C} because of union do → {B, C}
6. Do this since there is no new state.
7. Print the DFA matrix

This is how my algorithm works.

NFA



DFA