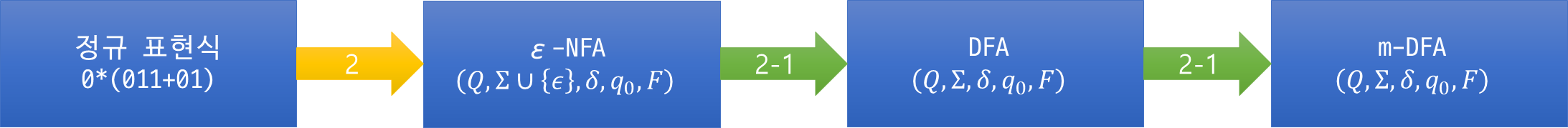
CS322 프로젝트 2-1. ε-NFA to m-DFA

# **개요**

정규 표현식(regular expression)에 대응하는 m-DFA를 생성하는 프로젝트 2의 보조 과제이다.

프로젝트 2에서 아래와 같은 순서로 정규 표현식을 최소 상태 DFA로 변환하게 될 터이다. 이 중 epsilon-NFA를 m-DFA로 변환하는 과정을 미리 작성하는 것이 보조 프로젝트의 목표이다.



# **프로그램 설명**

project 2-1:

testcase/e-NFA.txt  
testcase/m-DFA.txt  
DFA.py  
eNFA.py  
mDFA\_convert.py  
main.py

각 python 코드의 import 구조는 오른쪽 그림과 같다.

## 개발 환경

개발 언어: python 3.5(IDLE 사용)  
개발 운영체제: Windows 10

## **DFA.py**

프로젝트 1-1에서 작성한 레거시 코드이다. 과제 1-1을 하던 당시 변수 이름을 마구잡이로 지었는데, 이번 프로젝트를 진행하는 데 방해되어 모두 수정하였다. DFA는 5개의 변수(states, vocabulary, transition\_function, initial\_state, final\_states)로 구성된다.

## eNFA.py

epsilon-NFA 클래스를 정의하고 있다. 전체적인 형태는 DFA와 크게 다를 것 없으나, e\_closure(어떤 상태 집합에서 epsilon으로 전이되는 상태들의 집합)와 transition\_function의 사전 값이 상태들의 tuple로 나오는 등의 수정이 있다. dfa 함수는 e-NFA와 동등한(equivalent) DFA를 만들어 리턴한다. 다만 상태의 수가 기하급수적으로 늘어나기 때문에 성능이 좋지 못하다. 따라 도달할 수 없는 상태들(unreachable states), 구분되지 않는 상태들(undistinguishable states)를 줄일 필요가 있다.

## mDFA\_convert.py

주어진 DFA를 받아 상태를 최소화하는 몇 가지 함수들을 작성한 파일이다.

* **reachable(dfa):** DFA의 initial state로부터 도달 가능한 모든 상태 집합을 리턴한다. DFA에서 도달할 수 없는 상태를 알고자 할 경우 dfa.states.difference(reachable(dfa))를 입력하면 된다.
* **reduce\_unreachable(dfa):** DFA에서 도달할 수 없는 상태들을 제거하고 states와 transition\_function, final\_states를 알맞게 수정한 결과를 리턴한다.
* **minimize\_dfa(dfa):** DFA를 입력받아 구분 불가능한(undistinguishable) 상태를 찾아내고, 이들을 묶어 새로운 DFA를 리턴하는 함수이다.
* rename(dfa): 위의 과정을 거쳐 dfa를 최소화하면 상태의 이름이 튜플 형태로 나타난다. 프로그램 상에서는 문제될 게 없지만, m-DFA로 기록할 경우 튜플 괄호 안의 반점(,)이 애매하게 파싱될 위험이 있다. 따라 모든 상태 이름을 {q0, q1, q2, …} 꼴로 바꾸는 함수를 작성하였다.

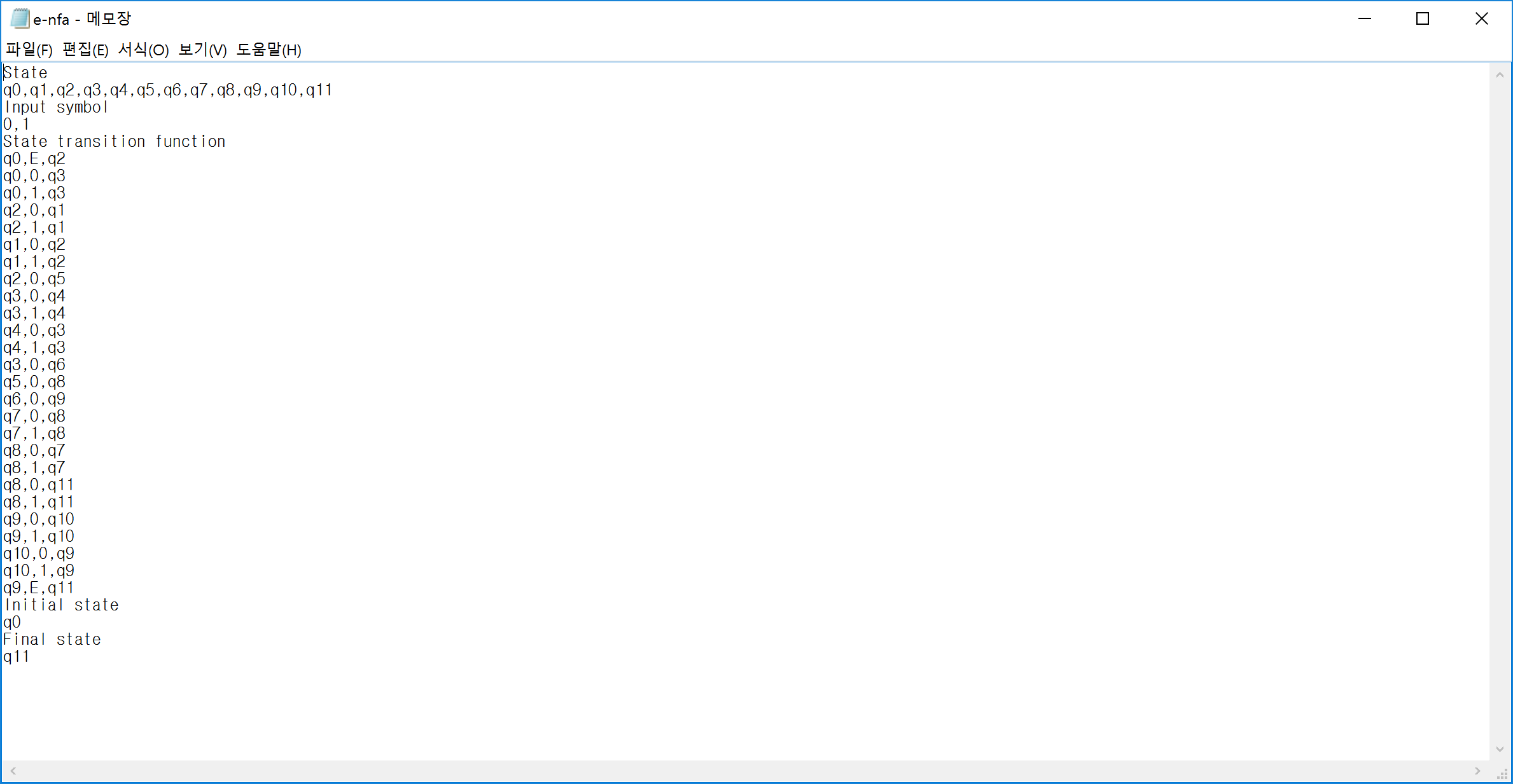
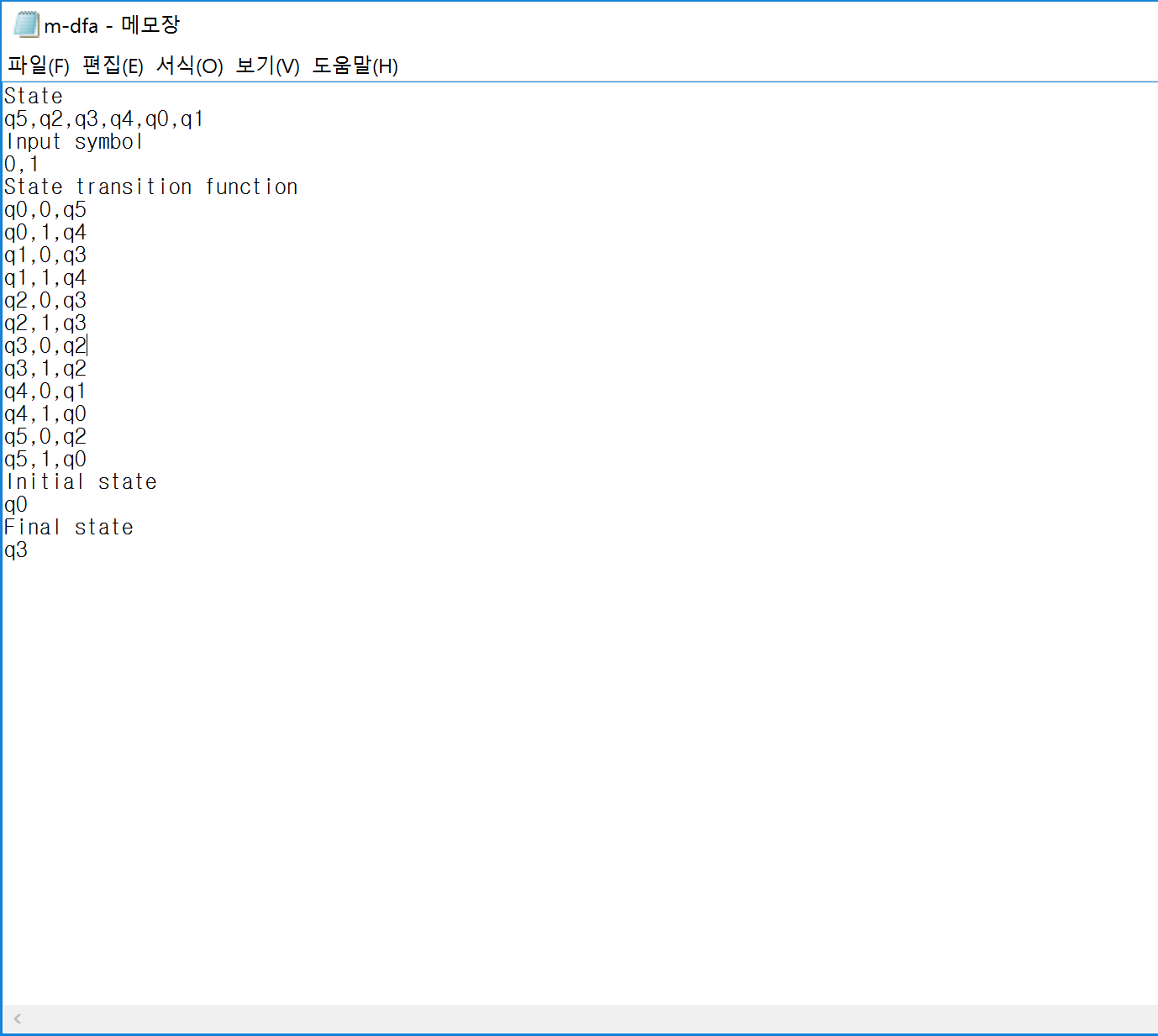
## main.py

testcase/e-nfa.txt를 읽어 프로그램 위에 e-NFA를 만든 뒤, 이를 m-DFA로 변환하여 testcase/m-dfa.txt로 저장하는 코드이다. 프로그램 실행 시에는 이 파일만 고려하면 충분하다.

# **실행 예시**

## **윈도우 예시(cmd)**

\PA2-1> notepad testcase/e-nfa.txt  
\PA2-1> python main.py  
\PA2-1> notepad testcase/m-dfa.txt

* 시작 상태(initial state)가 반드시 q0는 아님을 유의 바람.
* 위 e-NFA와 m-DFA는 “00을 substring으로 가지는 홀수 길이의 string”을 의미한다.  
  위 m-DFA를 그림으로 옮기면 아래와 같다.