Bài 1: Đơn định hóa Otomat
Bài 2: Tối thiểu hóa Otomat
Link code bằng python: https://github.com/hoang6k/coding_automata_python

Nhóm 1:
Bùi Thị Thu Huệ
Lê Thị Duyên
Nguyễn Chí Thảo
Đào Minh Hoàng

Bài 1: Đơn định hóa Otomat

Thuật toán:

Input: Cho NFA $M = (Q, \Sigma, \delta, q_0, F)$:

Q'=∅, δ'=∅, *S*=queue{ [q_0] } (*S* là hàng đợi các trạng thái chuẩn bị thăm)

while *S*≠∅:

p=*S*.pop() #Lấy ra đỉnh để thăm tiếp theo

Q'.push(p) #Cho đỉnh đang thăm vào tập trạng thái Otomat mới

delta_set là tập các hàm chuyển mà có trạng thái khởi đầu là $q \in p$

foreach *c* in Σ:

q_set là tập các trạng thái mà từ *delta_set* đến được thông qua *c*

if *q_set*=∅ then continue

Loại bỏ các trạng thái trùng trong *q_set*

δ'.push(δ(*p*, *c*)=*q_set*)

if *q_set* not in *Q'* then *S*.push(*q_set*)

F' = {*q*| *q*∈*Q*, *q*∩*F*≠∅}

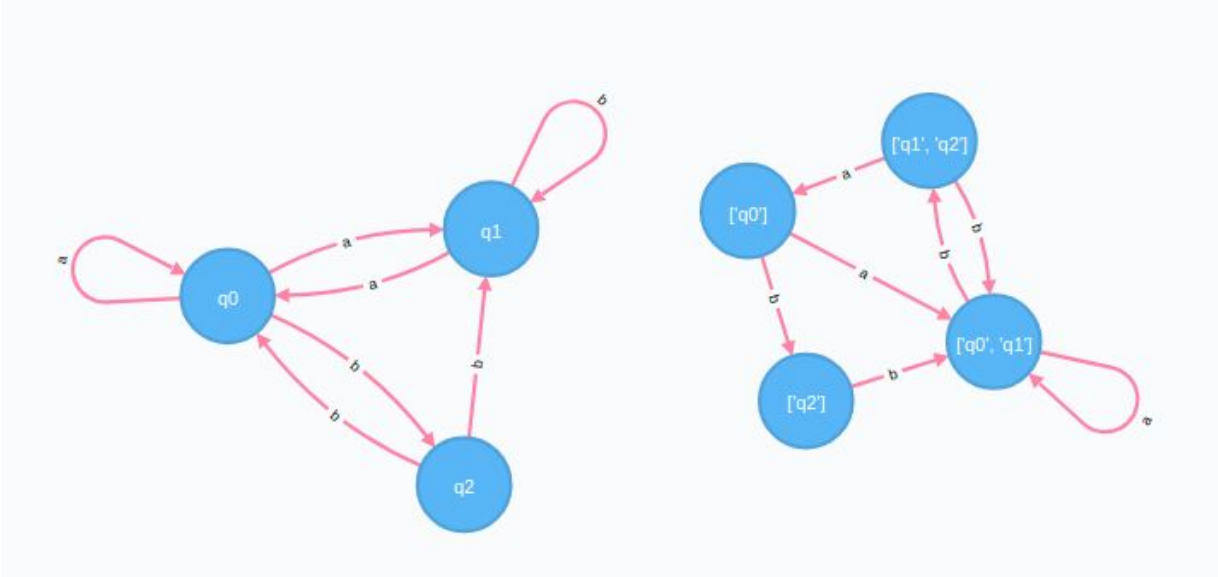
Output: DFA $M' = (Q', \Sigma, \delta', [q_0], F')$

Code:

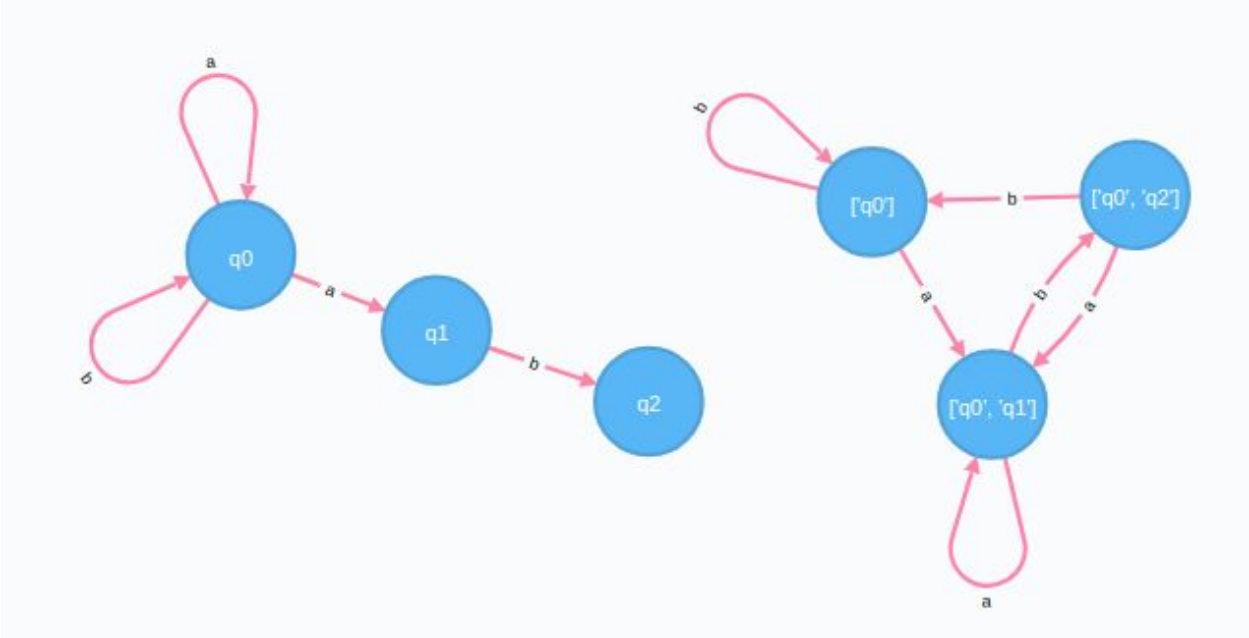
- Ngôn ngữ Python, link https://github.com/hoang6k/coding_automata_python
- file NFA_2_DFA.py là file chạy chương trình chuyển NFA về DFA, input NFA lấy từ file NFA_1 hoặc NFA_2
- file create_FA.py để tạo NFA từ file NFA_1 và NFA_2
- file Delta.py để tạo class hàm chuyển với các tham số q, c, set: $\delta(q, c)=set$
- file Automata.py để tạo class Otomat và chứa phương thức chuyển NFA về DFA
- file create_graph.py để tạo đồ thị bằng công cụ Neo4j

Ví dụ: với q_2 là trạng thái kết thúc của các NFA bên trái, chuyển sang DFA ở bên phải

```
NFA
    Q: q0 q1 q2
    A: a b
    Delta:
        delta(q0, a) = ['q0', 'q1']
        delta(q0, b) = ['q2']
        delta(q1, a) = ['q0']
        delta(q1, b) = ['q1']
        delta(q2, b) = ['q0', 'q1']
    q0: q0
    F: q2
DFA
    Q: ['q0'] ['q2'] ['q0', 'q1'] ['q1', 'q2']
    A: a b
    Delta:
        delta(['q0'], a) = ['q0', 'q1']
        delta(['q0'], b) = ['q2']
        delta(['q2'], b) = ['q0', 'q1']
        delta(['q0', 'q1'], a) = ['q0', 'q1']
        delta(['q0', 'q1'], b) = ['q1', 'q2']
        delta(['q1', 'q2'], a) = ['q0']
        delta(['q1', 'q2'], b) = ['q0', 'q1']
    q0: ['q0']
    F: ['q2'] ['q1', 'q2']
```



```
NFA
    Q: q0 q1 q2
    A: a b
    Delta:
        delta(q0, a) = ['q0', 'q1']
        delta(q0, b) = ['q0']
        delta(q1, b) = ['q2']
    q0: q0
    F: q2
DFA
    Q: ['q0'] ['q0', 'q1'] ['q0', 'q2']
    A: a b
    Delta:
        delta(['q0'], a) = ['q0', 'q1']
        delta(['q0'], b) = ['q0']
        delta(['q0', 'q1'], a) = ['q0', 'q1']
        delta(['q0', 'q1'], b) = ['q0', 'q2']
        delta(['q0', 'q2'], a) = ['q0', 'q1']
        delta(['q0', 'q2'], b) = ['q0']
    q0: ['q0']
    F: ['q0', 'q2']
```



Bài 2: Tối thiểu hóa Otomat

Thuật toán: (Hopcroft's algorithm - https://en.wikipedia.org/wiki/DFA_minimization **)**

***Input*:** Cho DFA $M = (Q, \Sigma, \delta, q_0, F)$:

- Xóa bỏ các trạng thái không thể chạm tới:

```
let reachable_states := {q0};
let new_states := {q0};
do {
   temp := the empty set;
   for each q in new_states do
      for each c in Σ do
         temp := temp ∪ {p such that p = δ(q,c)};
      end;
   end;
   new_states := temp \ reachable_states;
   reachable_states := reachable_states ∪ new_states;
} while (new_states ≠ the empty set);
unreachable_states := Q \ reachable_states;
```

- Tạo ra các nhóm chứa các trạng thái tương đương với nhau:

```
P := {F, Q \ F};
W := {F};
while (W is not empty) do
    choose and remove a set A from W
    for each c in Σ do
        let X be the set of states for which a transition on c leads to a state in A
        for each set Y in P for which X ∩ Y is nonempty and Y \ X is nonempty do
            replace Y in P by the two sets X ∩ Y and Y \ X
            if Y is in W
                replace Y in W by the same two sets
            else
                if |X ∩ Y| <= |Y \ X|
                    add X ∩ Y to W
                else
                    add Y \ X to W
        end;
    end;
end;
```

Q' = P

F' = {q| q∈Q, q∩F≠∅}

δ' được xây dựng nhờ gộp các hàm chuyển của các trạng thái cùng nhóm

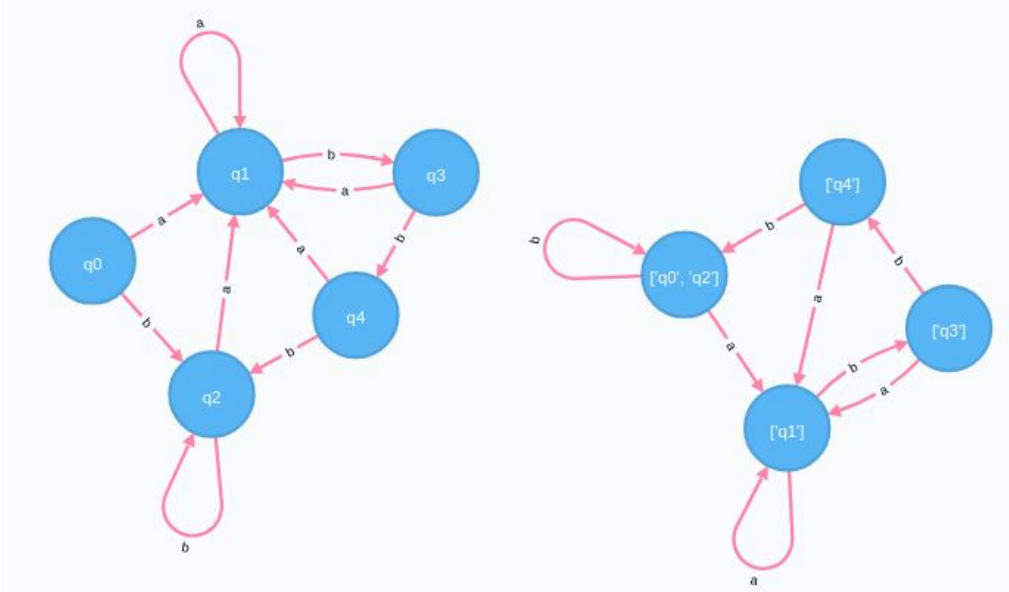
***Output*:** DFA $M' = (Q', \Sigma, \delta', P_0, F')$ (với P_0 là nhóm có chứa q_0)

Code:

- file Minimize_DFA.py là file chạy chương trình tối thiểu hóa DFA, create_FA.py tạo FA với input DFA lấy từ file DFA_1, DFA_2 hoặc DFA_3
- file Automata.py để tạo class Otomat và chứa phương thức tối thiểu DF

Ví dụ: Với q_3 là trạng thái kết thúc của DFA bên trái, DFA được tối thiểu ở bên phải

```
DFA
    Q: q0 q1 q2 q3 q4
    A: a b
    Delta:
        delta(q0, a) = q1
        delta(q0, b) = q2
        delta(q1, a) = q1
        delta(q1, b) = q3
        delta(q2, a) = q1
        delta(q2, b) = q2
        delta(q3, a) = q1
        delta(q3, b) = q4
        delta(q4, a) = q1
        delta(q4, b) = q2
    q0: q0
    F: q4
DFA
    Q: ['q0', 'q2'] ['q1'] ['q3'] ['q4']
    A: a b
    Delta:
        delta(['q0', 'q2'], a) = ['q1']
        delta(['q0', 'q2'], b) = ['q0', 'q2']
        delta(['q1'], a) = ['q1']
        delta(['q1'], b) = ['q3']
        delta(['q3'], a) = ['q1']
        delta(['q3'], b) = ['q4']
        delta(['q4'], a) = ['q1']
        delta(['q4'], b) = ['q0', 'q2']
    q0: ['q0', 'q2']
    F: ['q4']
```



```
DFA
    Q: q0 q1 q2 q3 q4 q5 q6 q7
    A: a b
    Delta:
        delta(q0, a) = q1
        delta(q0, b) = q0
        delta(q1, a) = q0
        delta(q1, b) = q2
        delta(q2, a) = q3
        delta(q2, b) = q1
        delta(q3, a) = q3
        delta(q3, b) = q0
        delta(q4, a) = q3
        delta(q4, b) = q5
        delta(q5, a) = q6
        delta(q5, b) = q4
        delta(q6, a) = q5
        delta(q6, b) = q6
        delta(q7, a) = q6
        delta(q7, b) = q3
    q0: q0
    F: q3
DFA
    Q: ['q0'] ['q1'] ['q2'] ['q3']
    A: a b
    Delta:
        delta(['q0'], a) = ['q1']
        delta(['q0'], b) = ['q0']
        delta(['q1'], a) = ['q0']
        delta(['q1'], b) = ['q2']
        delta(['q2'], a) = ['q3']
        delta(['q2'], b) = ['q1']
        delta(['q3'], a) = ['q3']
        delta(['q3'], b) = ['q0']
    q0: ['q0']
    F: ['q3']
```