1.

$\Sigma = \{a, b\}$

a. Regular expr:

$(a a^* (b + \cancel{bb})) + (a a^*)$

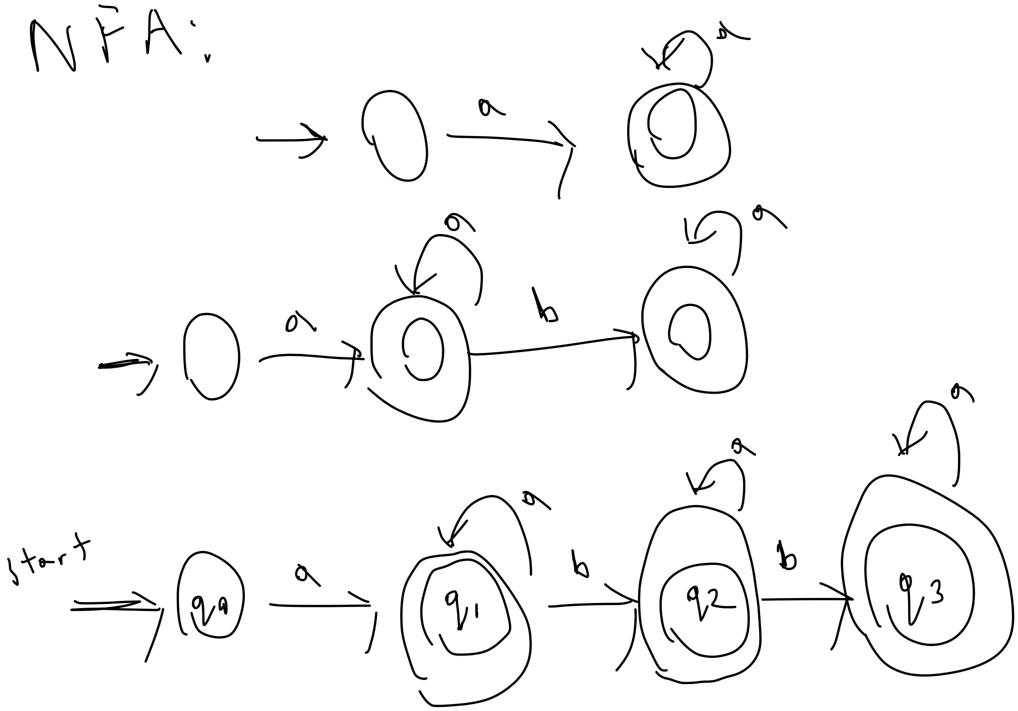
$b a^* b$

$(a a^* (b + \cancel{bb'}))^+$

$b a^+ b$

$a a^* + a a^* b a^* + a a^* b a^* b a^*$

NFA:



DFA construction:

Queue = $[[q_0]]$

$q_0 = s =$ $\xrightarrow{a} q_1$

$\xrightarrow{b} [\,]$

Queue = $[[q_1], [\,]]$

$q_1 = s =$ $\xrightarrow{a} q_1$

$\xrightarrow{b} q_2$

Queue = $[[\,], [q_2]]$

$[\,] = s =$ $\xrightarrow{a} [\,]$

$\xrightarrow{b} [\,]$

Queue = $[[q_2]]$

$q_2 = s =$ $\xrightarrow{a} q_2$

$\xrightarrow{b} q_3$

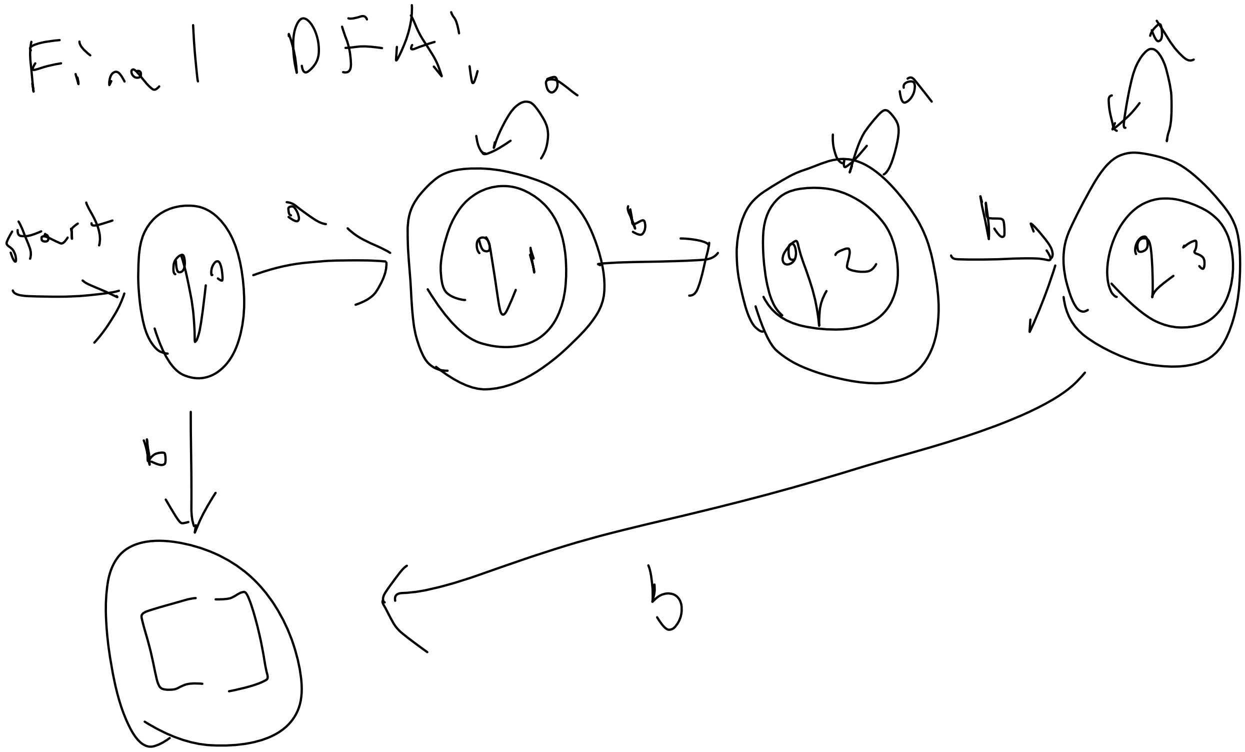
Queue = $[[q_3]]$

$q_3 = s =$ $\xrightarrow{a} q_3$

$\xrightarrow{b} [\,]$

1. a. cont.

Final DFA:

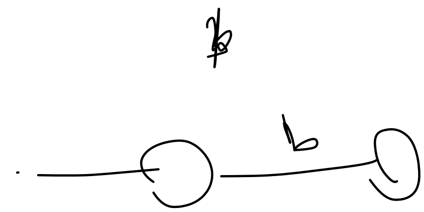


1.

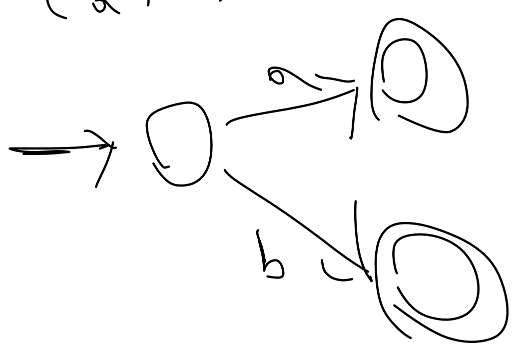
b. Regular exp:

$$((a+b)(a+b))^* \, b$$

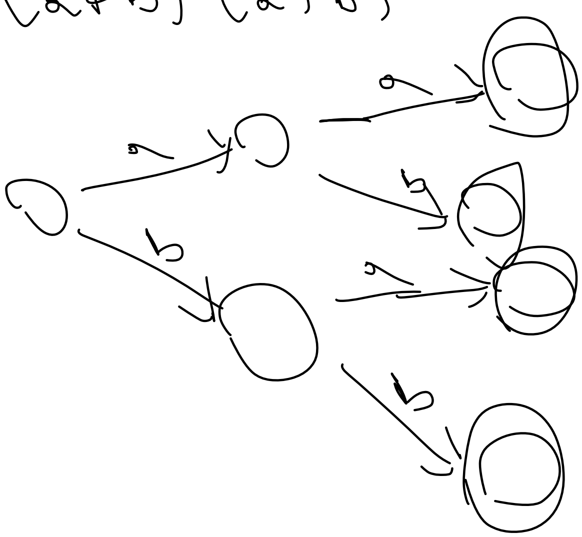
NFA:



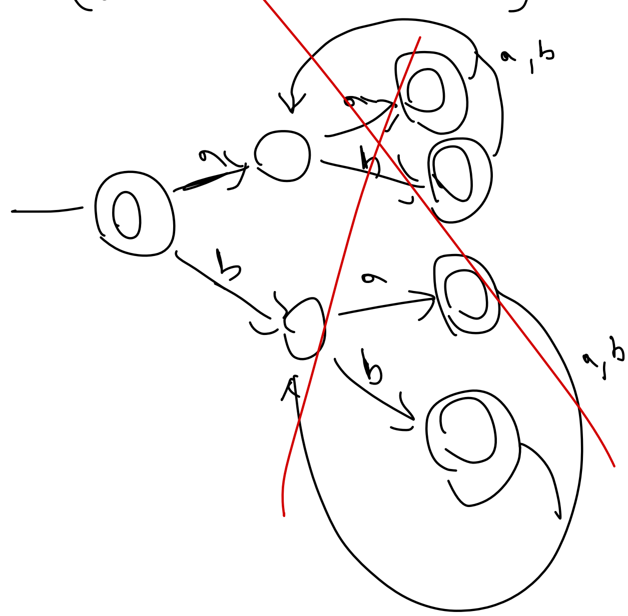
$(a+b)$



$(a+b)(a+b)$

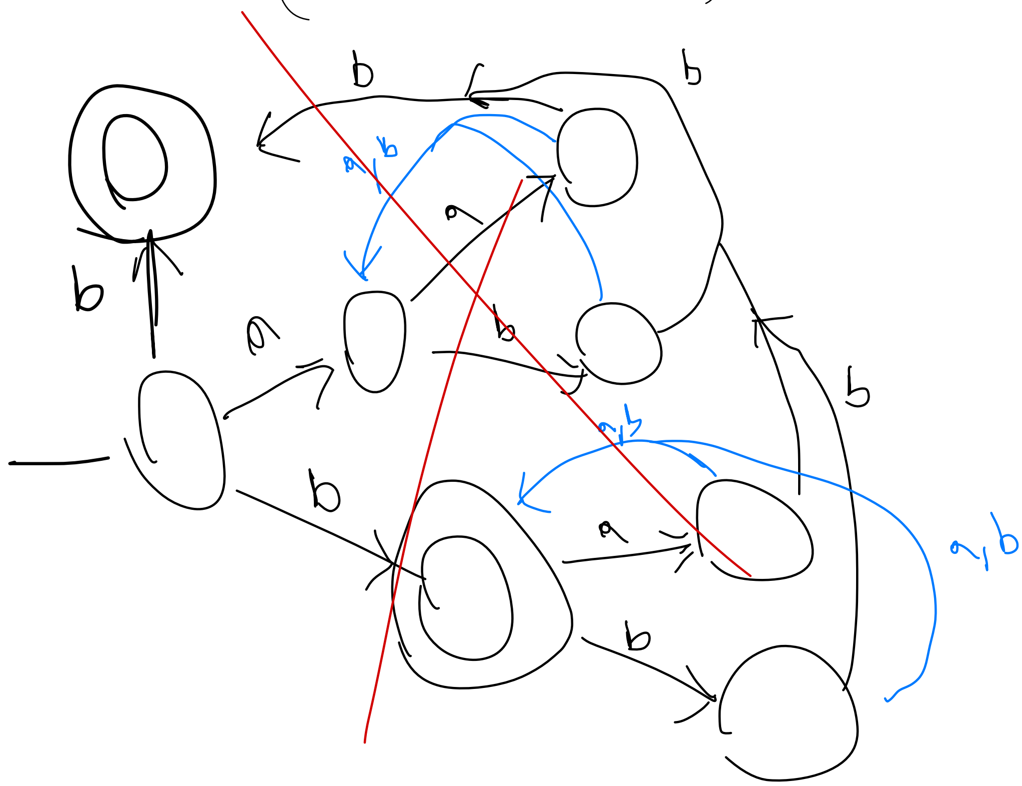


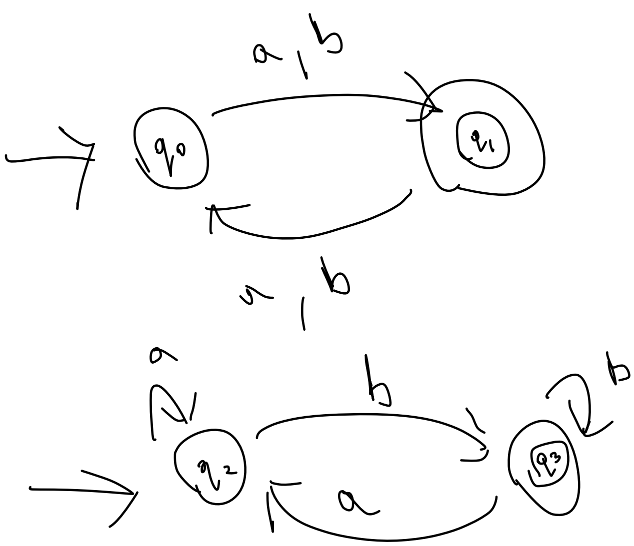
$((a+b)(a+b))^*$



1. b. cont.

NFA: $((a+b)(a+b))^* b$



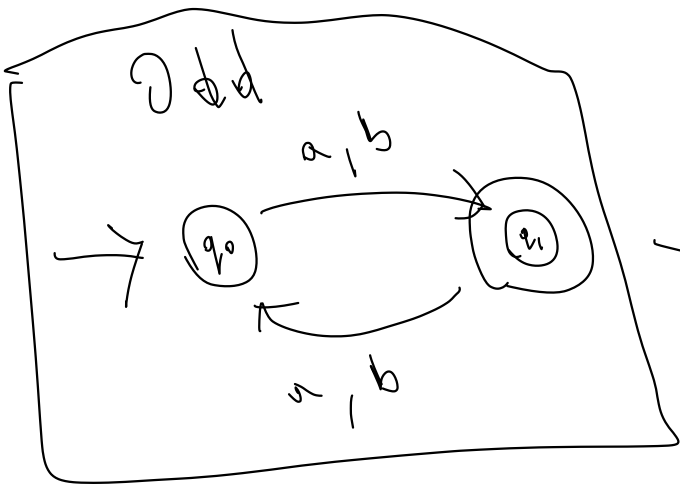


odd test

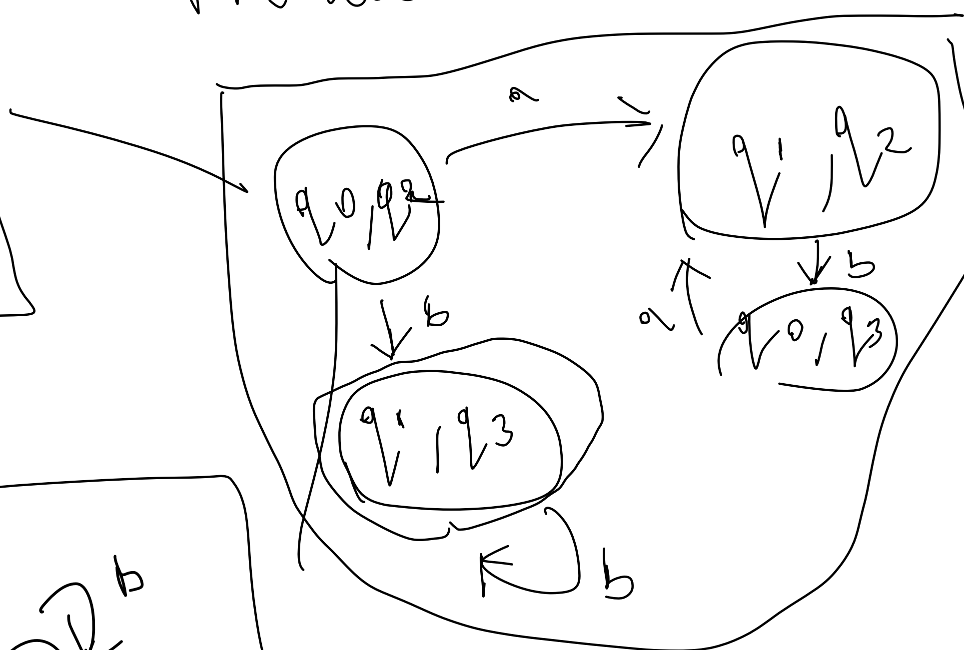
and

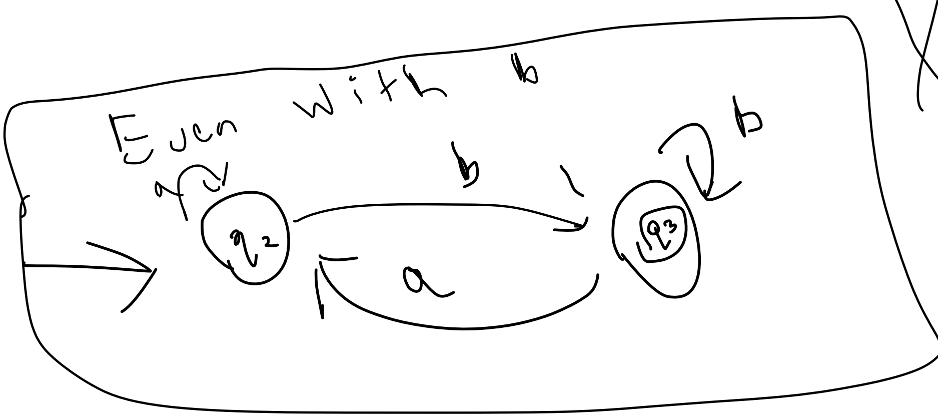
even with b

DFA Intersection



Product DFA





1. c.

Regular expr:

$$(a+b)^* \; (a(a+b)^* a) \; (a+b)^{\cancel{*}}$$

$$(b(a+b)^* b)$$

$$(ab(a+b)^* ba)$$

bbaa

baba

abab

aabb

abba

baab

ab ba

aa bb

Final Regex:

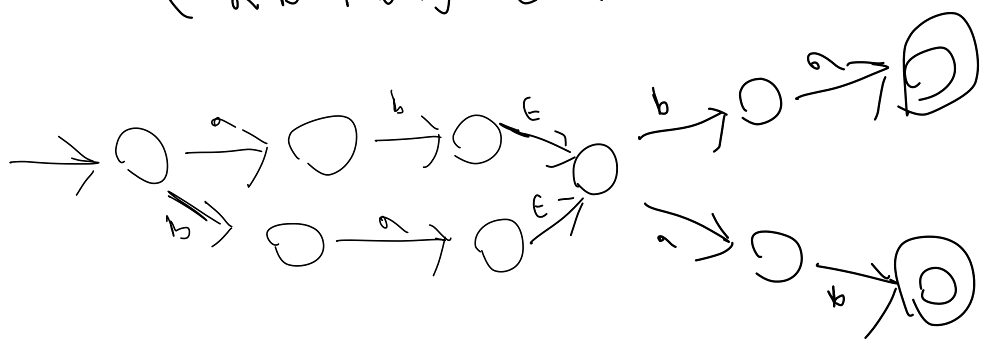
$$(a+b)^* \; (((ab+ba)(ba+ab))) + (aabb+bbaa)))(a+b)^*$$

1. c.

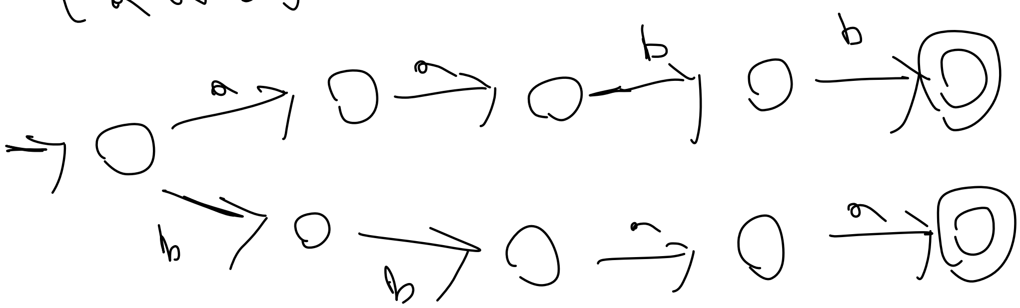
NFA

$(a+b)^*$

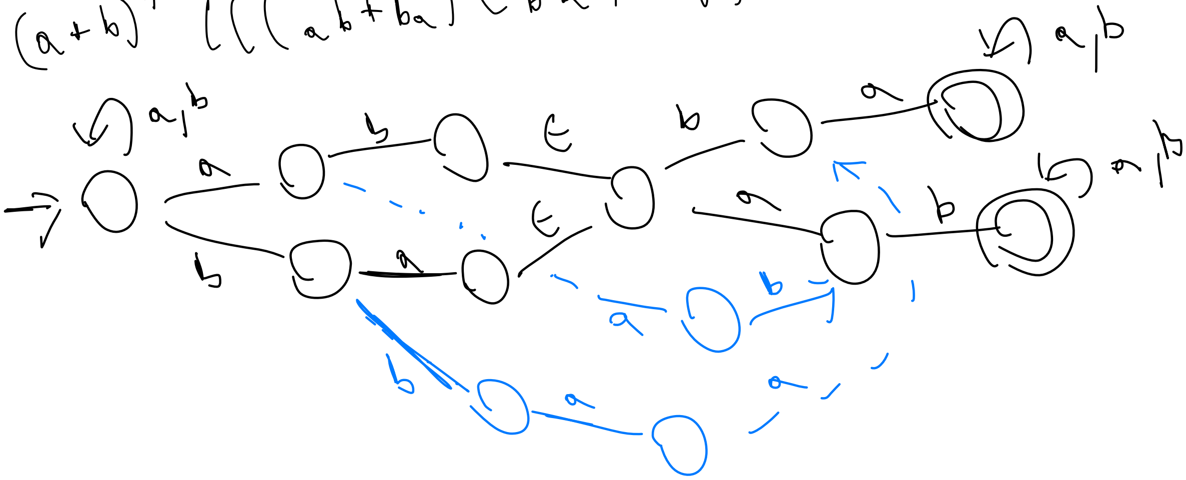
$(ab+ba)(ba+ab)$



$(aabb+bbaa)$



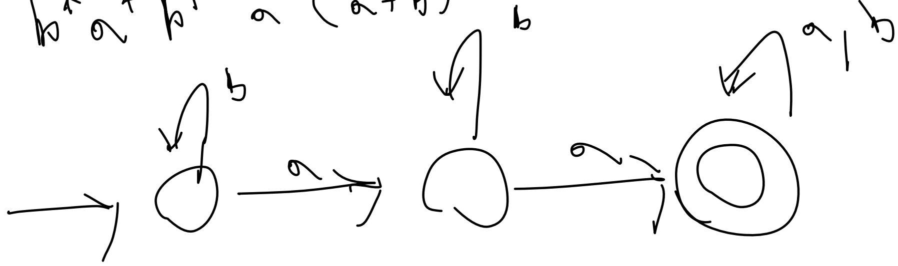
$(a+b)^* (((ab+ba)(ba+ab)) + (aabb+bbaa))(a+b)^*$



1. c.

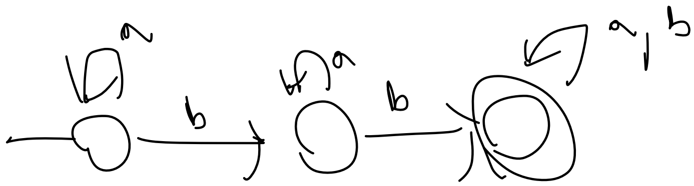
Must have at least two a's

$b^* a^+ b^* a^+ (a+b)^*$

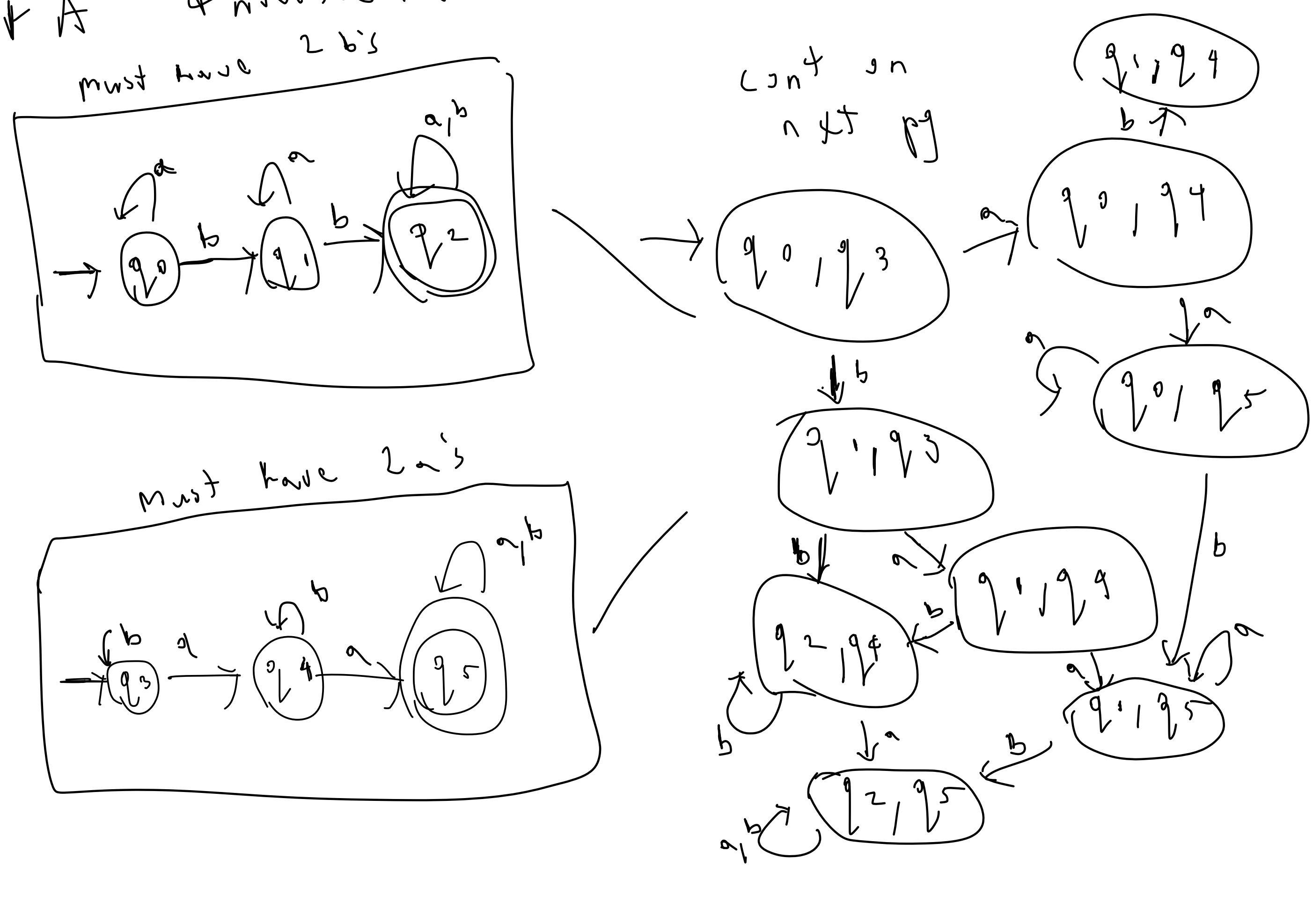


Must have at least two b's

$a^* b^+ a^* b^+ a^* b^*$

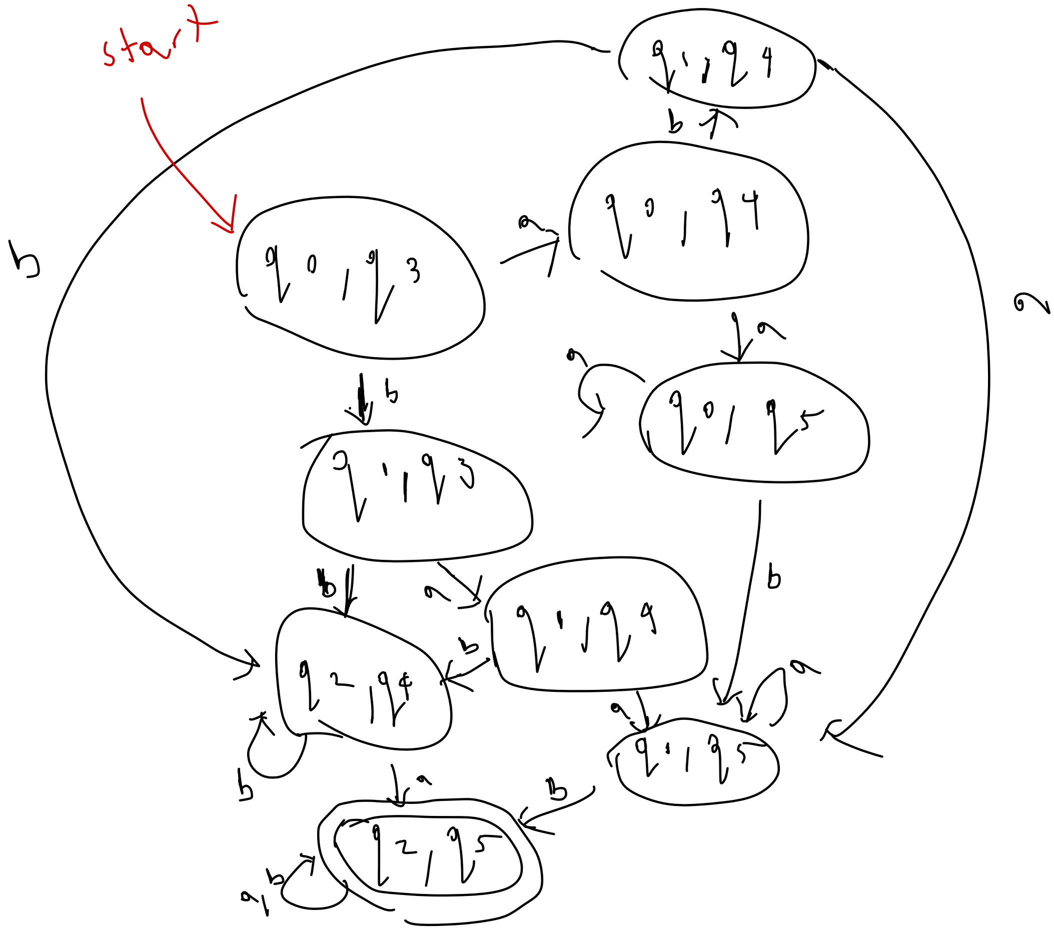


DFA Intersection



1. c.

Final DFA



2. Claim:

$\Sigma = \{+, -, 0\text{-}9, \}$

$L_{3,5} = \{ i \mid i \% 3 \text{ and } i \% 5 \}$

Let $L_{3,5}$ be the set of integers that are divisible by both 3 and 5. Let i be any integer of $L_{3,5}$.

Proof(s):

$L_{3,5}$ can be considered be a smaller language of L_3 (integers divisible by 3) & L_5 (integers divisible by 5) through intersection. Simply put, both rules of languages must be applied.

2. Proof. (cont.)

$L_5 = (\epsilon + `-')(1-9)(0-9)^*(0+5)$ Lowest capture is 10

what digits equals divisible by 3

$9=9$ $1+2=3$ $\cancel{3+3+3=9}$

$8+1=9$ $1+1+1=3$ $\cancel{6+3=9}$

$\cancel{3+0=3}$ $\cancel{3+3=6}$

$1+5=6$ $2+4=6$

$\cancel{L_3 = (\epsilon + `-')((12+21)+(111)+(3)+(6)+(24+42)+\ldots}$

$\cancel{\ldots(61+18)+(15+51)+(9)+(9))^*}$

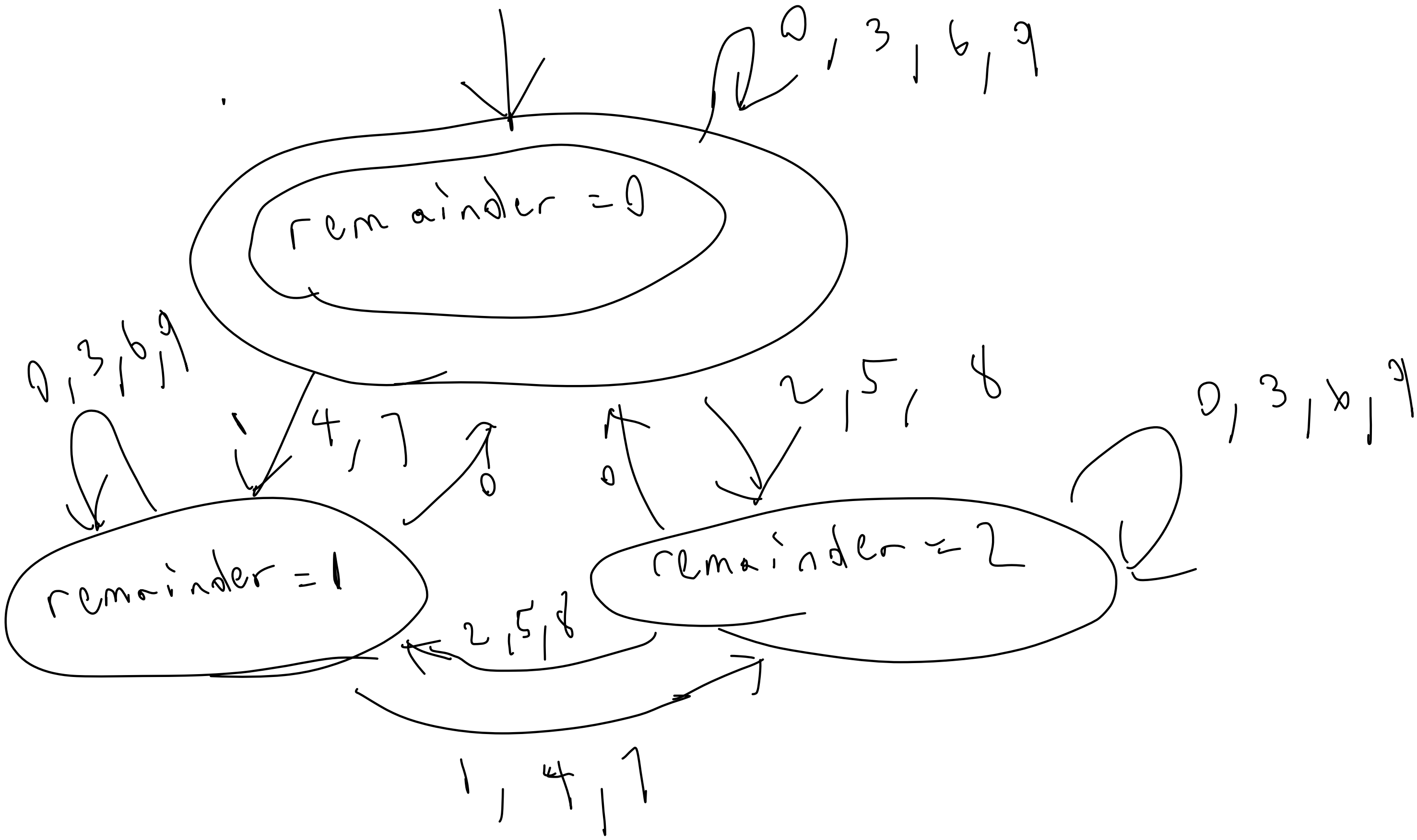
Ignore

See L_3 proof next pg

NFAs:

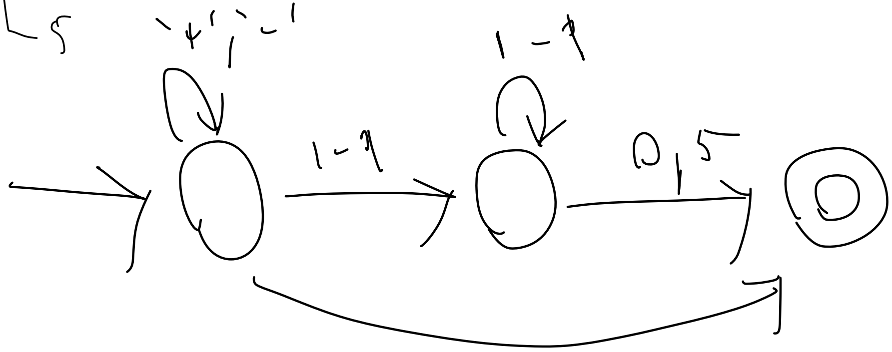
Proof of L3 using modulo

start recursive modulo 3



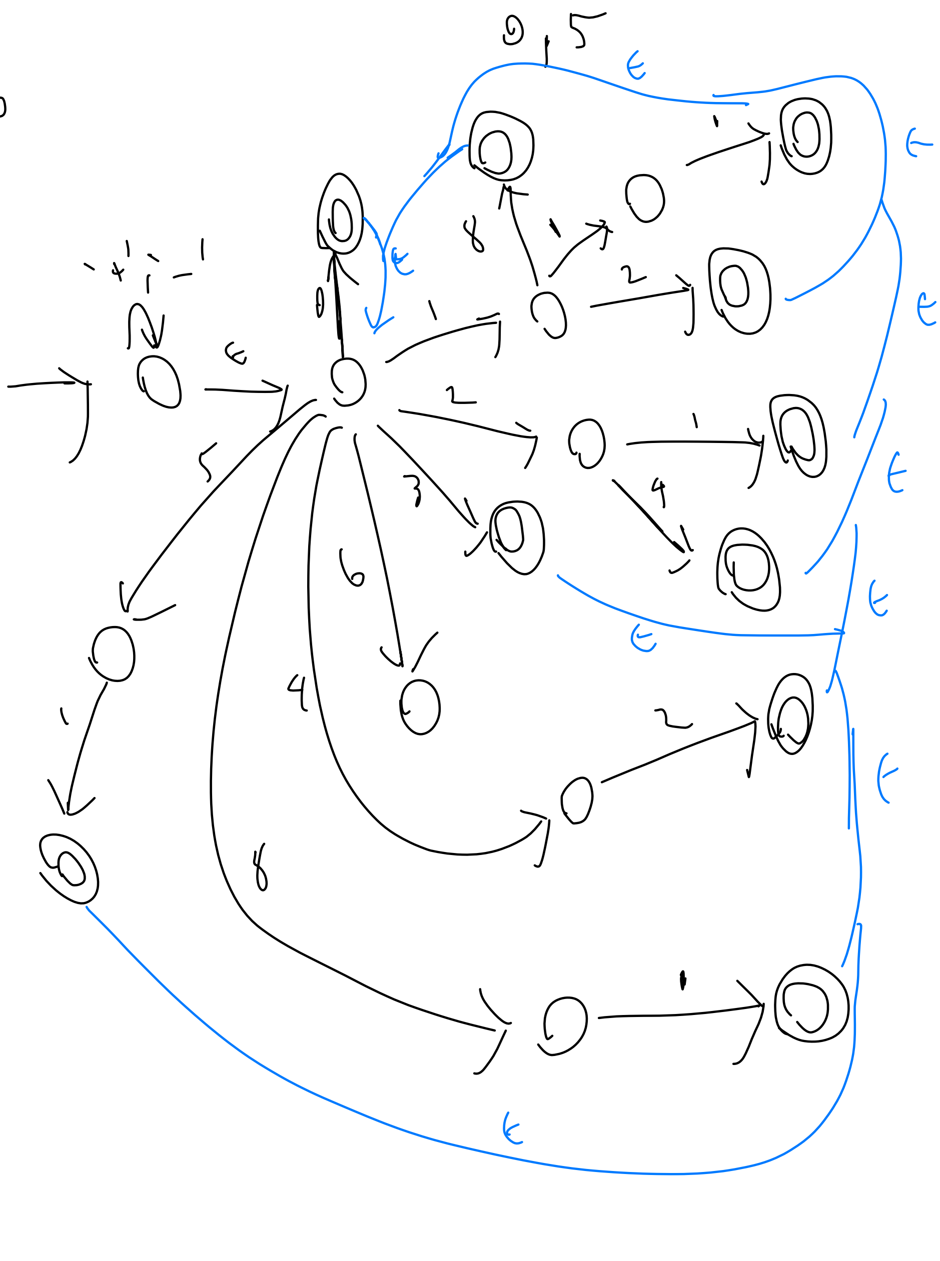
NFAs:

L_5



scrapped but feel free to judge

L_3



3. Program Design

0. Read file to into string

1. Using lexer / parser
Obtain temporary data structure
(Python list, AST)
Ex: ('q0', ['q1', 'q3'], [('q0', 'a', 'q1'),
('q0', 'a', 'q2'),
⋮
('q3', 'a', 'q2')])

2. Use result and
construct NFA obj, and

* Extract sigma & q from
transitions

3. cont.

3. d = D1.convert_to_dfa()

$\underbrace{\text{convert\_to\_dfa()}}_{\text{NTD algorithm}}$

4. print(D).

Pseudo code NTD algorithm

```
queue = [self.start]
create empty dfa, d.
while queue != [];
    q = queue.deque   (queue[0])
    queue = queue[1:]
    process q -> update d.
return d.
```

```
def add_transition(f, c, t):

    f - from node
    c - symbol on node (a or b
    t - to node


with open(sys.argv[1]) as f:
```