Lexical Analysis:

Input ? Lexemes

Output ? Tokens

↓

Lexical Analyzer / Scanner / Lexer

→ Basics — token / lexemes ✓

FSA / FSM

↓

Types

Tokens

✓ " Hi! " ✓

literal string

Literal character 'A'

'B' 'H'

String "Hey! what's up" ✓

Byte b 'H'

Byte string b "Hello"

Number literals — decimal integer

↓ ↓

19.20

_ underscore 19_20

Hex 0x4F

.
.
.
.
.
.
.

etc

⇒ Write a lexer

Token $\xrightarrow{\text{are Represented}}$ Regular Expressions

are Represented by ↓

FSM / FSA

(finite state Machine)
("   "   Automate)

* **FSA** :

→ Recognize patterns within input

→ Accept | Reject

```
1.  Cin >> char
2   while ( char != "m") cin >> char
3.  If ( cin >> char != "a" ) Go to step1
4.       "              = "i")       "
5.       "              = 'n")       "

    Done.
```

main

→ 1. Initialization

→ 2. 'm' looking

→ 3. Recognized 'm' so look 'a'

→ 4. " "ma" " 'i'

→ 5. " "mai" " 'n'

→ main → done

Accept

Eg: Oven

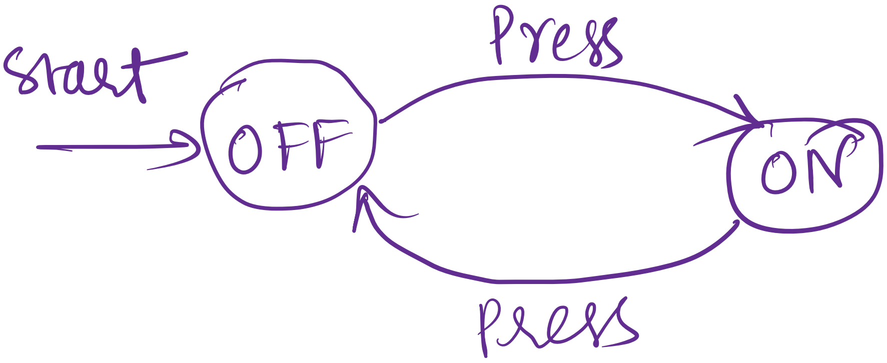
↪

ON	OFF

FSA — Oven (2 States)

ON , OFF

2 states.

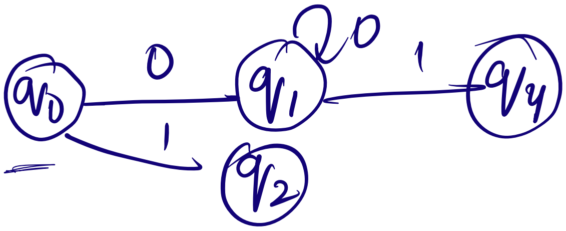


FSA (finite State Automata) | FSM

DFA | DFSMachine
State

Deterministic
finite Automate

↳ 1 state — 1 input

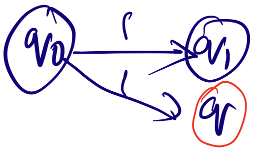


No state has more than
one outgoing edge
with same input

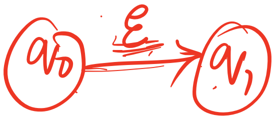
NFA | NFSM

Non-Deterministic

one state → same
input → diff
states



ε-NFA



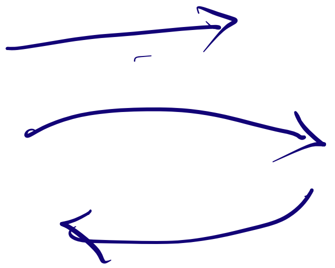
Each step in program —— different recognition process

↓

capture $\xrightarrow{\text{behaviour}}$ graph

⇒ Each node — step / input state

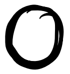
⇒



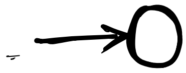
movement of one → another (transition)

⇒ lables on arcs — inputs

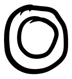
FSA viewed as graphs



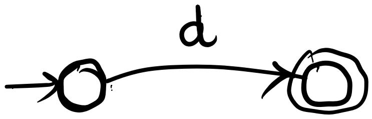
STATE



The start state



Accepting state | final state



A Transition
'd' input

DFSA : 5 Tuples

δ

$(\Sigma, Q, q_0, F, \delta)$

sigma   delta

Σ = finite set of input .

Q = finite set of states (Together all states)

q_0 = Initial State

F = final State

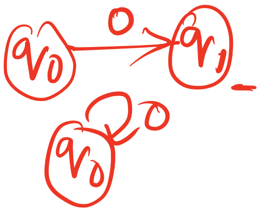
δ = state transition function

$$\delta : \boxed{(Q \times \Sigma) \rightarrow Q}$$

$$Q = \{q_0, q_1, q_2\}$$

$$\Sigma = \{0, 1\}$$

$$q_0 \times 0 \longrightarrow q_1$$



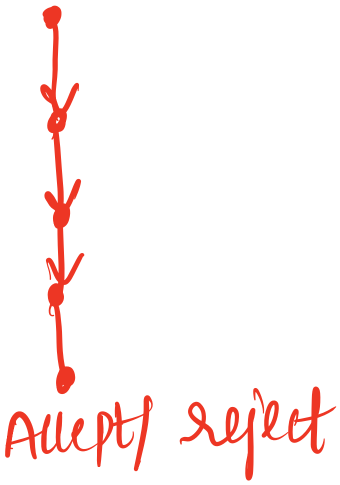
$$\boxed{\text{Every DFA also an NFA}}$$

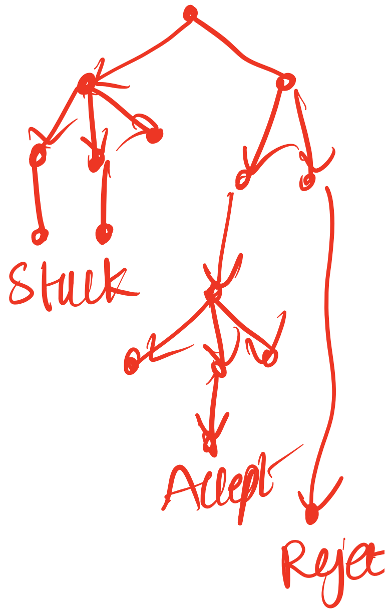
$$\boxed{\text{Every NFA} \not\Rightarrow \text{DFA}}$$

Tree of computations :

Deterministic Computation
(DC)

NDC





NDC ? → Lack of Information

Randomness

↓

✗ · probability ·

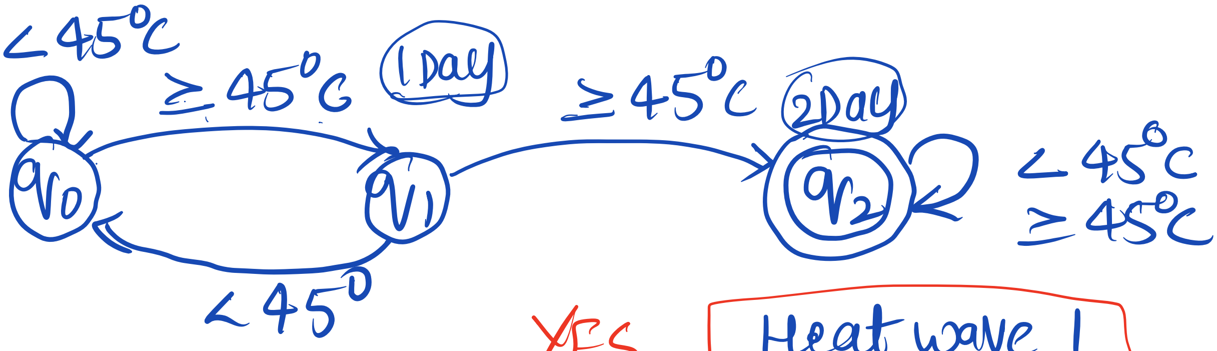
Q) weather data — input (45°)

Heat wave ? (Yes / NO)

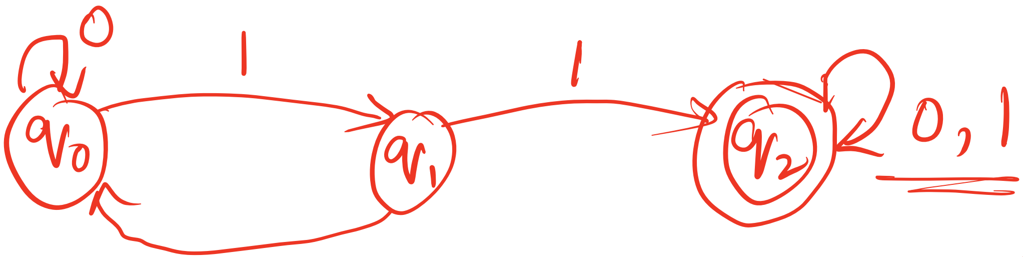
Heat wave : ≥ 45°C (113°F) for 2 consecutive days)

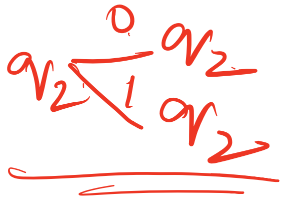
Solution ≥ 45°C = 1 ✓

< 45°C = 0 ✓



YES Heat wave !





DFA .

① Transition Diagram

② Transition Table

$\Sigma \downarrow$ (inputs)

Q →

	0	1
→ q_0	$\{q_0\}$	$\{q_1\}$
q_1	$\{q_0\}$	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$

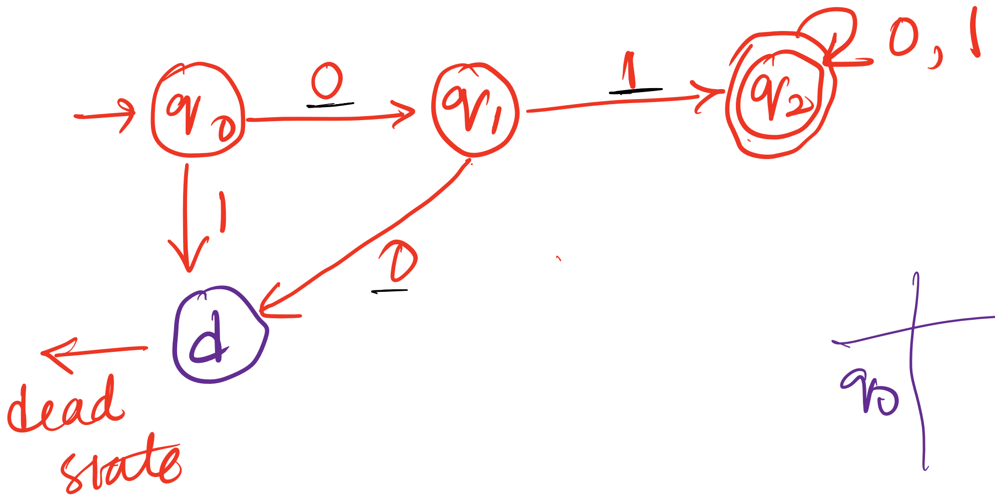
③ Transition function :

$$\delta : (Q \times \Sigma) \rightarrow Q$$

$$q_0 \times 0 \longrightarrow q_0$$

Q) Draw a DFA, which accepts all strings with 01 $\Sigma = \{0, 1\}$

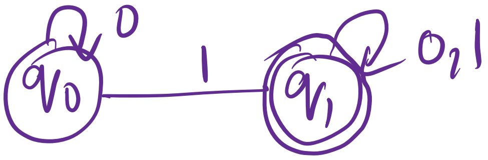
$2 + 1 = $ ③ states



	1
q_0	d

$q_0 \longrightarrow q_1 \longrightarrow q_2$ ⇒ just mention the path

with ② states

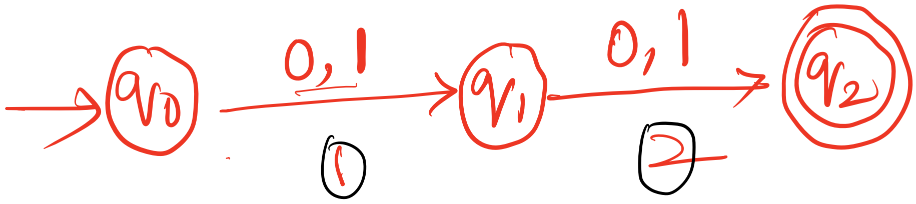


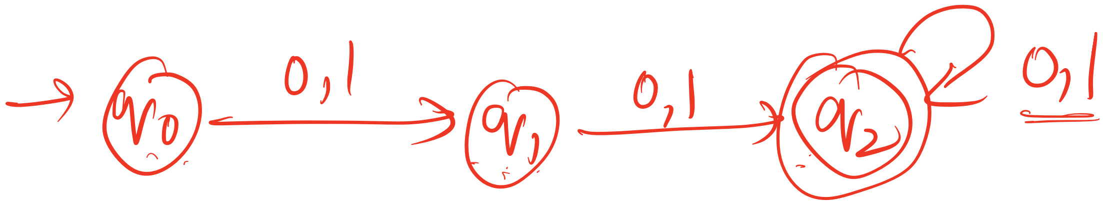
q_0 — q_1

q_0 — q_0 — q_1 } path

Eg: Draw a DFA, accepts all strings of length atleast 2
{0,1}

0,1 & 0,1
1+1 = 2

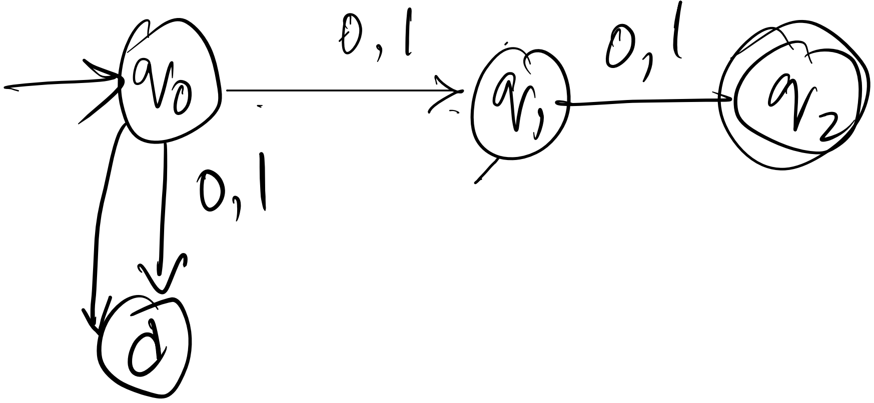




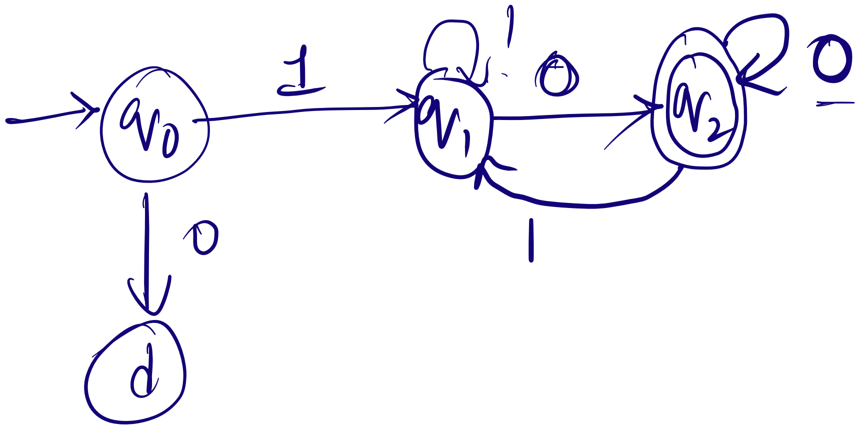
length 2 & more

$q_0 - q_1 - q_2$ — length2

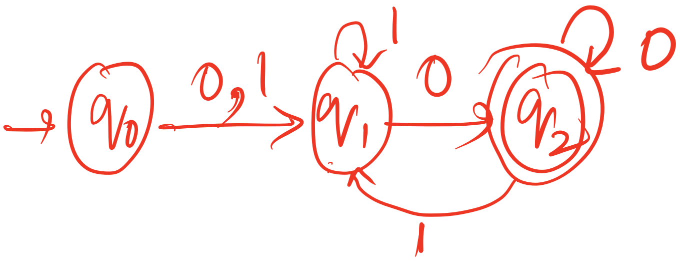
can also be



Q) DFA, starts with 1, and ends with 0 $\Sigma = \{0, 1\}$



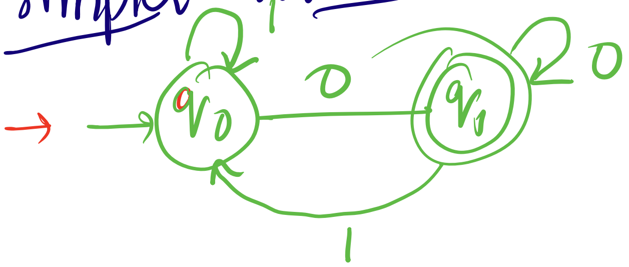
⇒ Ending with 0



So, this can be either

00
10
110
1111010
0011110
1011100
101100 ....
etc.

Simpler solution



Practise Problems

1. Construct a DFA, that goes over $\Sigma = \{0, 1\}$ and accepts string with 3 consecutive 0's

2. Accepts the string containing substring 101

3. Accept the string that has 2nd symbol / place from right end as "0".

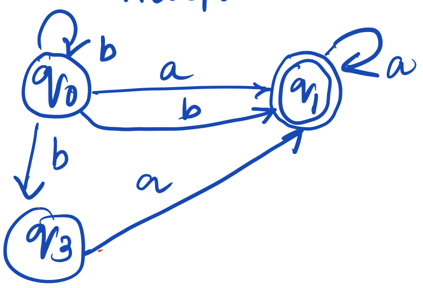
NFA: Exists many paths for one input from current state to next state.

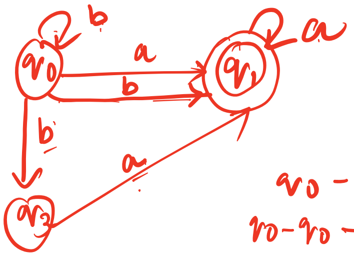
Every NFA $\neq$ DFA
(but, each NFA can translated to DFA)

⇒ 5 tuples
$\{Q, \Sigma, \delta, q_0, F\}$

$$\delta : Q \times \Sigma \rightarrow \underset{\displaystyle 2^{Q}}{\underset{\downarrow}{P(Q)}}$$

Eg: Accept the string $\boxed{ba}$

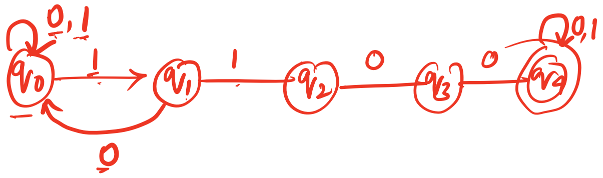




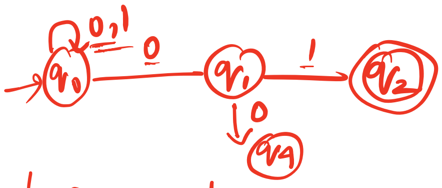
$q_0 - q_1 - q_1$
$q_0 - q_0 - q_1$
$q_0 - q_3 - q_1$
$q_3 - q_1 - q_1$

NFA that accepts
Q) double 1 followed by 00

1110



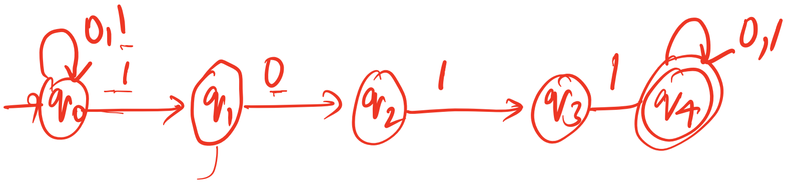
Q) NFA with $\Sigma = \{0,1\}$
accepts string ending with 01



	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_4\}$	$\{q_2\}$
q_2	$\{\ \}$	$\{\ \}$

Q) construct an NFA, that contains

1011


0,1
1
0
1
1
0,1
q_0
q_1
q_2
q_3
q_4