

A directed graph is a DAG iff it has a topological ordering of its vertices

Compute shortest paths in DAG via topological sorting.

$$G = (V, E)$$

$$|V| = m$$

$$(u, v) \in E$$

L = topological sort of vertices

$$L[1], L[2], \dots, L[m]$$

Shortest path algorithm (linear time):

d = array of length m

$d[i]$ = shortest path cost from vertex 1 to i .

$$d[1] = 0$$

$$d[u] = \infty$$

$$2 \leq k \leq m$$

P : array of pointers (backtracking pointers)

$$P[v] = -\infty$$

$$1 \leq v \leq m$$

$P[v]$ = the predecessor of vertex v in the shortest path from 1 to v .

FOR $i = 1$ to m , DO $u = L[i]$

FOR every v such that $(u, v) \in E$:

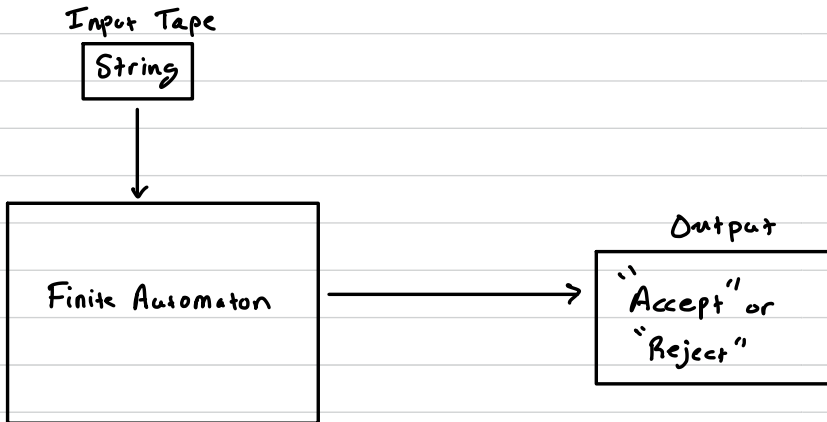
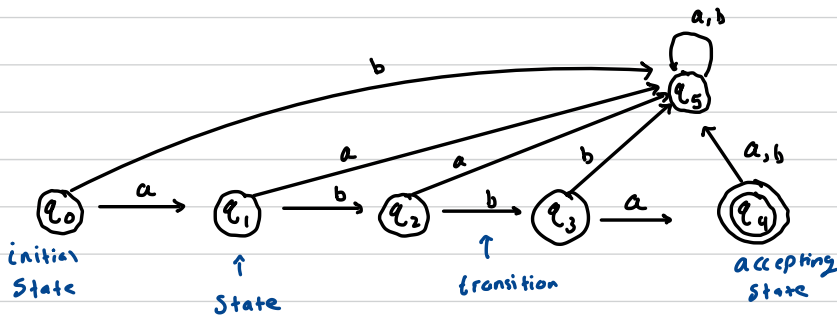
Let $w = \text{weight}(u, v)$

IF $d[v] > d[u] + w$

$$d[v] = d[u] + w$$

$$P[v] = u$$

Deterministic Finite Automata



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

For every state, there is
a transition for every symbol
in the alphabet

To accept a string:

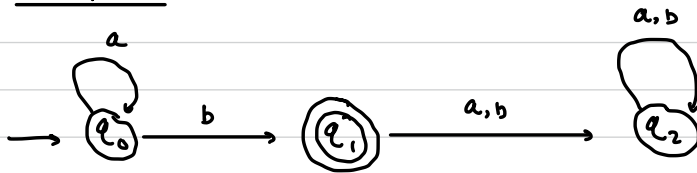
all the input string is scanned
and the last state is accepting

To reject a string:

all the input string is scanned
and the last state is non-accepting

Note that we can have an empty string (λ)

Example:



Language Accepted: $L = \{a^n b : n \geq 0\}$

Formal Definition:

$M = (Q, \Sigma, \delta, q_0, F)$, where

Q : set of states

Σ : input alphabet, $\lambda \notin \Sigma$

δ : transition function

q_0 : initial state

F : set of accepting sets

Transition Function:

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

$$\delta(q, x) = q'$$



Describes the result of a transition from state q with symbol x .

Extended Transition Function :

$$\delta^* : Q \times \Sigma^* \rightarrow Q, \quad \delta^*(q, \lambda) = q$$

$$\delta^*(q, w) = q'$$

Describes the resulting state after scanning string w from state q

$\delta^*(q, w) = q'$ implies that there is a walk of transitions $w = \sigma_1 \sigma_2 \dots \sigma_k$



States may be repeated

Language Accepted by DFA:

Language of DFA M :

it is denoted as $L(M)$ and contains all the strings accepted by M

We say that a language L' is accepted (or recognized) by DFA M if $L(M) = L'$

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$

Language Accepted by M :

$$L(M) = \{ w \in \Sigma^* : \delta^*(q_0, w) \in F \}$$

Language rejected by M :

$$\overline{L(M)} = \{ w \in \Sigma^* : \delta^*(q_0, w) \notin F \}$$

Regular Languages:

A language L is regular if there is a DFA M that accepts it ($L(M) = L$)

- The languages accepted by all DFAs form the family of regular languages.

Regular language examples:

$$\{ abba \}, \{ \lambda, ab, abba \}$$

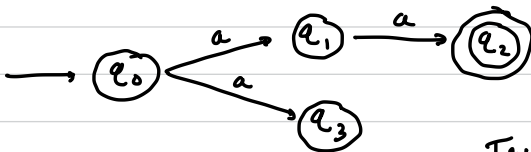
- There is no DFA that accepts non-regular languages

Non-regular language example:

$$L = \{ a^n b^n : n \geq 0 \}$$

Non-Deterministic Finite Automata :

Alphabet = $\{a\}$



Two choices

An NFA accepts a string :

if there is a computation of the NFA
that accepts the string

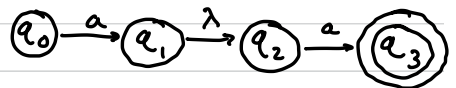
i.e. all the input string is processed and the
automaton is in an accepting state

In the above example, $L = \{a, aa\}$. $\{a^3\}$ and $\{aaaa\}$ rejected

Lambda Transitions: in this case, you jump to the new state

Transition Function δ :

$$\delta(q, x) = \{q_1, q_2, \dots, q_k\}$$



Resulting states following one transition
with symbol x .

Extended Transition Function δ^* . Same with δ but applied
to strings