La Macchina di Touring

Contenuto del corso (in breve)

Il corso è un'introduzione alle tre aree centrali della teoria della computazione:

- Teoria degli Automi (Linguaggi formali e modelli di calcolo)
- Teoria della Calcolabilità /Computabilità
- Teoria della Complessità

Le tre aree sono legate dalla domanda:

Quali sono le capacità e i limiti dei computer?

Coloro che hanno scelto di dedicare alcuni anni della loro formazione allo studio dei computer non possono non essere curiosi circa le loro capacità e limiti.

Contenuto del corso

- MODELLI DI COMPUTAZIONE:

  AUTOMI FINITI DETERMINISTICI E NON DETERMINISTICI.
  ESPRESSIONI REGOLARI. PROPRIETÀ DI CHIUSURA DEI LINGUAGGI REGOLARI. TEOREMA DI KLEENE. PUMPING LEMMA PER I LINGUAGGI REGOLARI.

  MACCHINA DI TURING DETERMINISTICA A NASTRO SINGOLO. IL LINGUAGGIO RICONOSCIUTO DA UNA MACCHINA DI TURING. VARIANTI DI MACCHINE DI TURING E LORO EQUIVALENZA.

- IL CONCETTO DI COMPUTABILITÀ: FUNZIONI CALCOLABILI, LINGUAGGI DECIDIBILI E LINGUAGGI TURING RICONOSCIBILI. LINGUAGGI DECIDIBILI E LINGUAGGI INDECIDIBILI. IL PROBLEMA DELLA FERMATA. RIDUZIONI. TEOREMA DI RICE.

- IL CONCETTO DI COMPLESSITÀ: MISURE DI COMPLESSITÀ: COMPLESSITÀ IN TEMPO DETERMINISTICO E NON DETERMINISTICO. RELAZIONI DI COMPLESSITÀ TRA VARIANTI DI MACCHINE DI TURING. LA CLASSE P. LA CLASSE NP. RIDUCIBILITÀ IN TEMPO POLINOMIALE. DEFINIZIONE DI NP-COMPLETEZZA. RIDUZIONI POLINOMIALI. ESEMPI DI LINGUAGGI NP-COMPLETI.

Calcolabilità

Negli anni ‘30 (del secolo scorso), quando ancora l’informatica e i computer NON esistevano, quindi indipendentemente da essi, alcuni logici si proposero questo progetto:

Formalizzare in modo esatto la nozione intuitiva di problema e di procedura effettiva di calcolo, così da definire quando un problema è risolvibile o no.

Questo progetto fu realizzato indipendentemente e con una diversa risposta da

- Alan Turing (1936) con il suo modello noto come Macchina di Turing
- Alonso Church (1936) con il λ-calcolo.

Tesi di Church - Turing

Fu dimostrato che le due risposte erano equivalenti, cioè definivano la stessa classe di problemi risolubili. Tutto ciò che poteva essere calcolato con una Macchina di Turing poteva essere calcolato (simulando la MdT) col λ-calcolo, e viceversa.

In seguito, ogni altra ‘’ragionevole" formalizzazione della nozione di procedura effettiva ha condotto agli stessi risultati.

Per questo è possibile parlare di un concetto di "calcolabilità", indipendente dal particolare formalismo.

Tesi di Church-Turing:

Tutto ciò che può essere intuitivamente calcolato tramite una procedura effettiva può essere calcolato con una Macchina di Turing.

ALGORITMO = MACCHINA DI TURING

Teoria degli Automi

Avete studiato gli automi finiti come primo semplice modello di computazione.

Gli automi finiti sono dei modelli di calcolo a memoria finita; hanno cioè un certo numero finito di stati di memoria in cui si possono trovare. Processano il dato in ingresso sequenzialmente in base allo stato in cui si trovano, fino ad arrivare ad un certo stato che può essere finale, e accettano, oppure non finale, e rifiutano.

Lo studio di questo semplice modello aiuterà a comprendere l'altro modello più complesso che studieremo, la Macchina di Turing.

Riprendiamo da dove eravamo rimasti…..

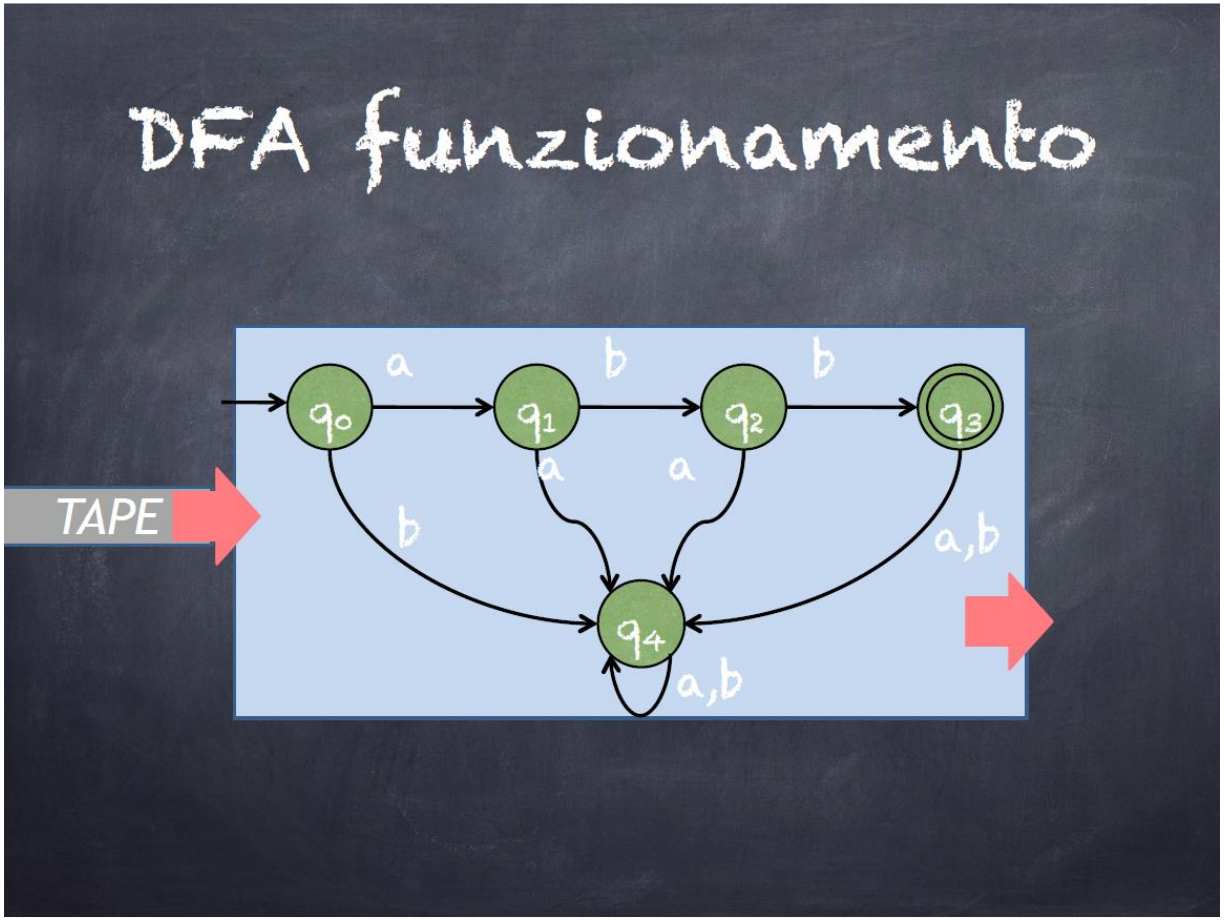
ادامه

- اثبات : دنباله اي است از گـزاره هاي $p_1, p_2, \ldots, p_k$ به گـونه اي كـه p_1 بايد يكـي از اصول و p_2 بايد يكـي از اصول باشد يا با استفاده از p_1 و يكـي از قواعد استنتاج اثبات مي شود و p_3 هم ... كـه به اين مراحل يک اثبات براي p_k گـفته مي شود.

- قضيه (theorem) : گـزاره اي است كـه در منطق، براي آن اثباتـي وجود داشته باشد.

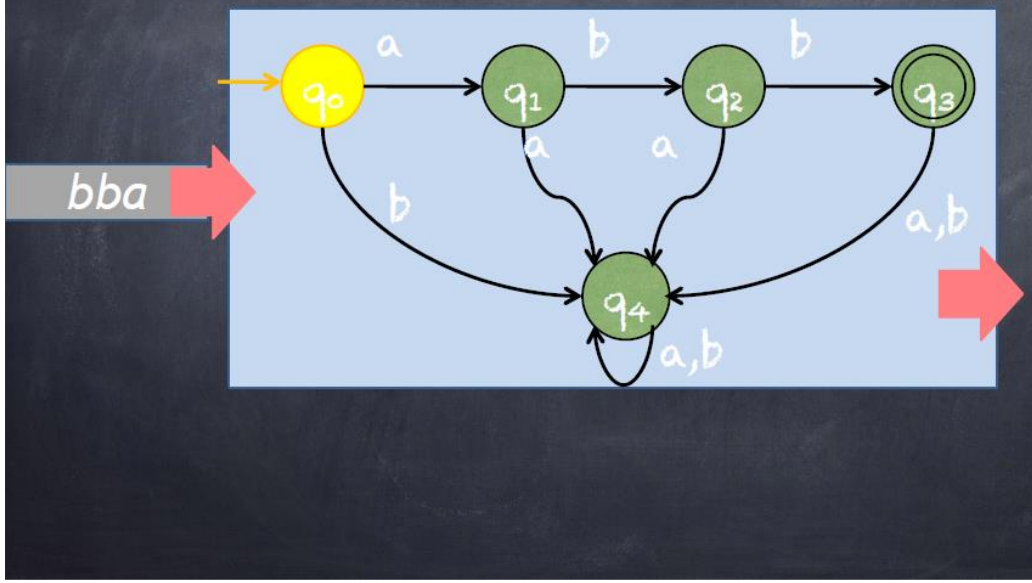
- منطق غير كـامل : فرمول هايي وجود دارد كـه نمي توان آنها را اثبات كـرد.

DFA funzionamento



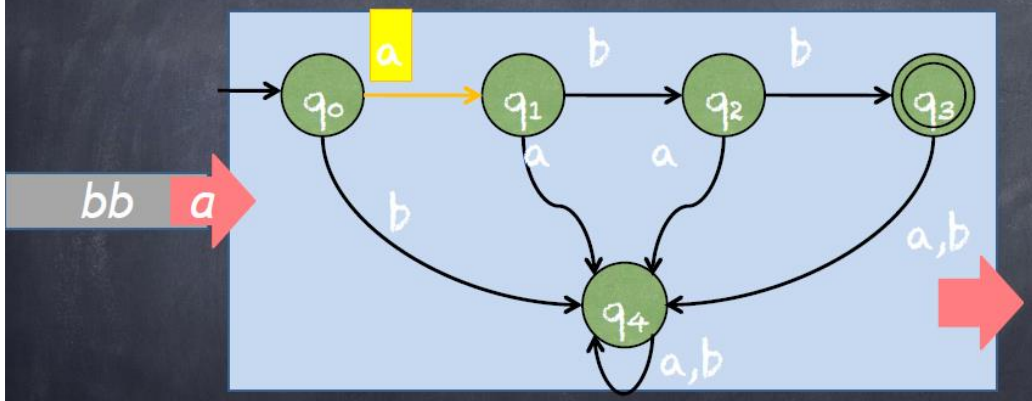
DFA funzionamento

Input string: *abb*



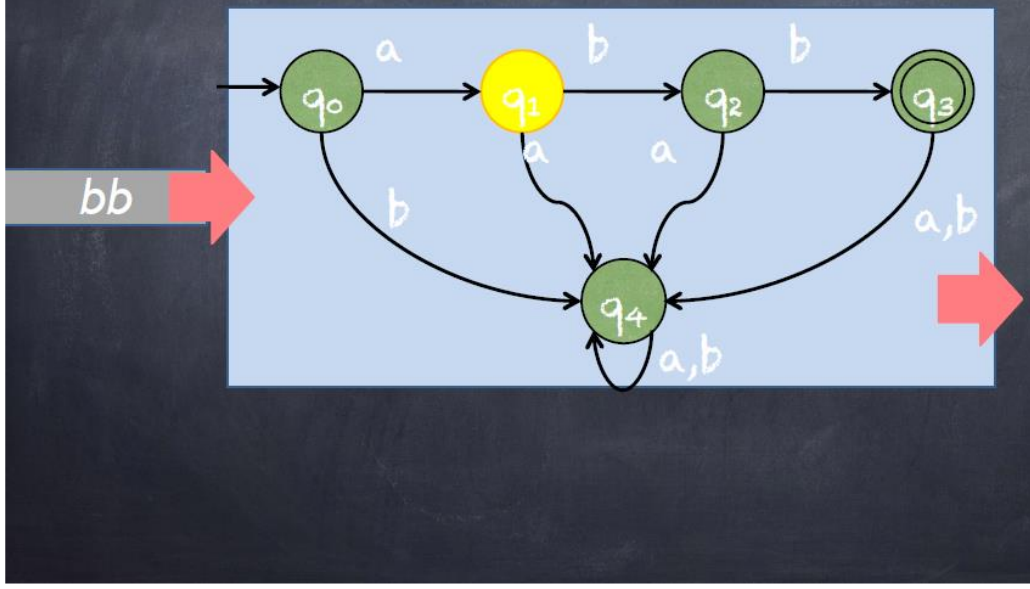
DFA funzionamento

Input string: *abb*



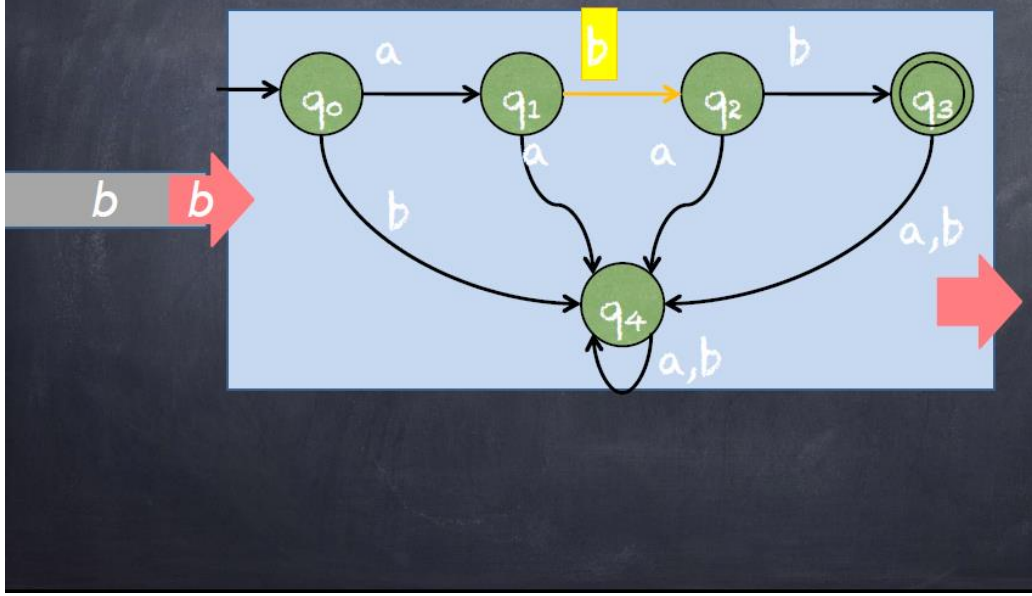
DFA funzionamento

Input string: *abb*



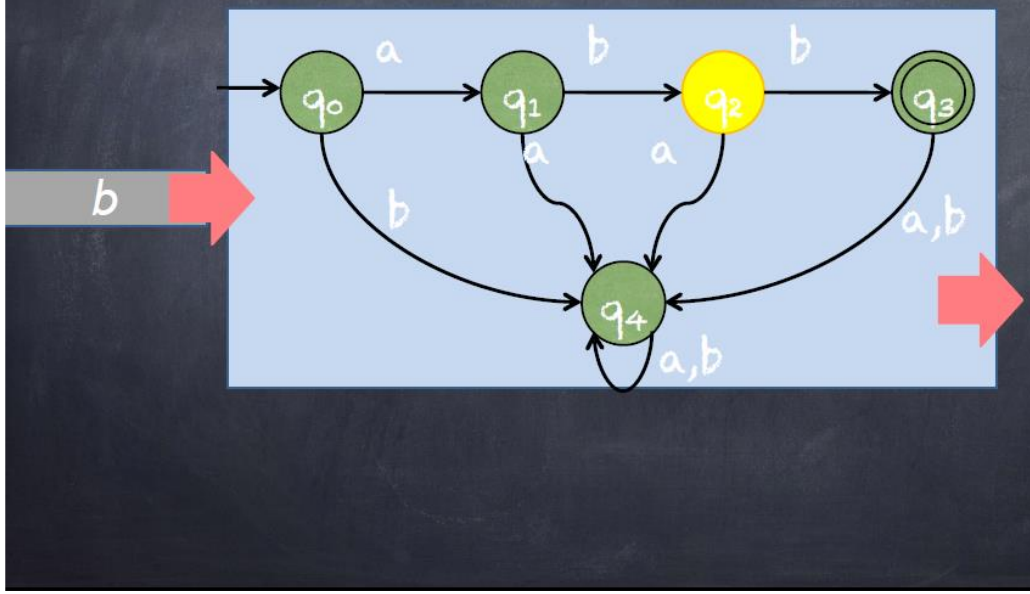
DFA funzionamento

Input string: *abb*



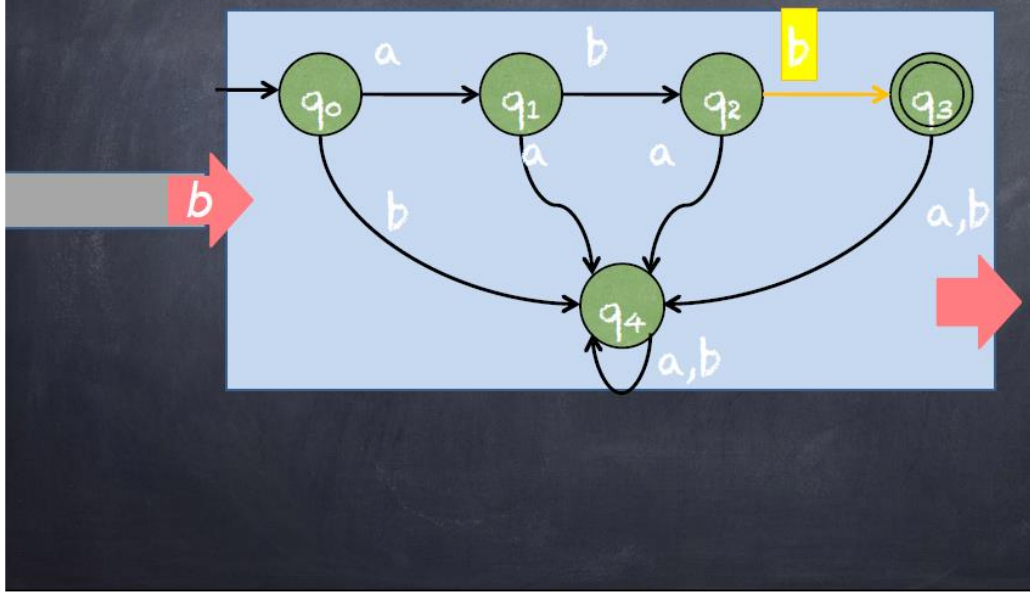
DFA funzionamento

Input string: *abb*



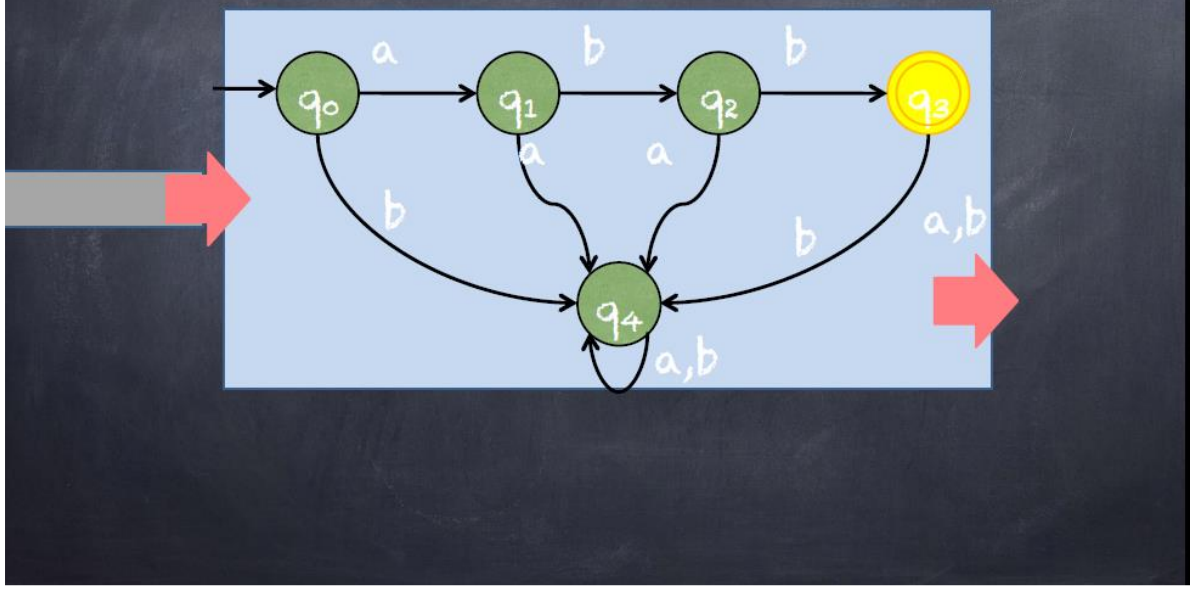
DFA funzionamento

Input string: *abb*



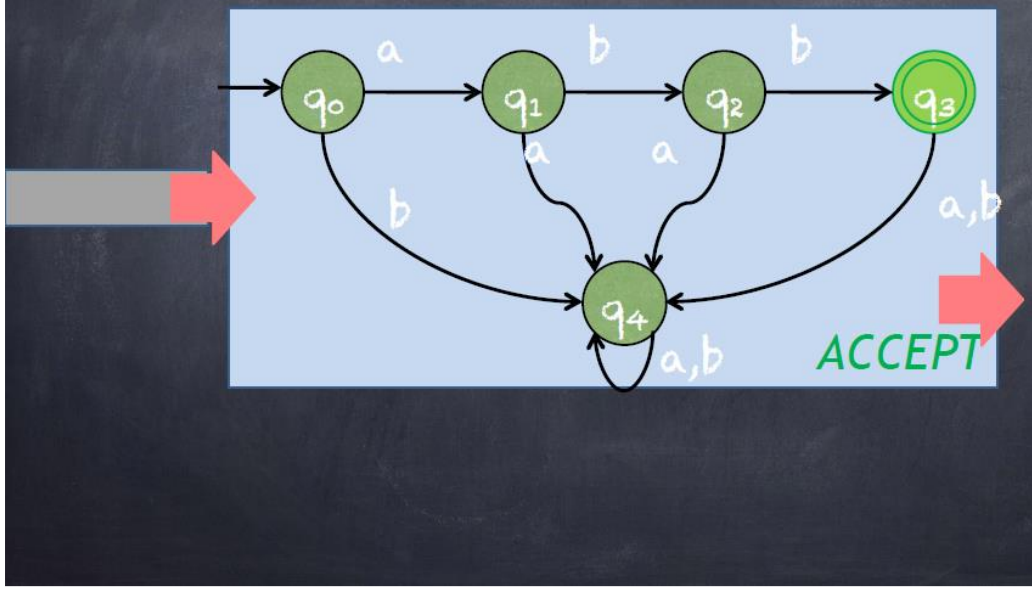
DFA funzionamento

Input string: *abb*

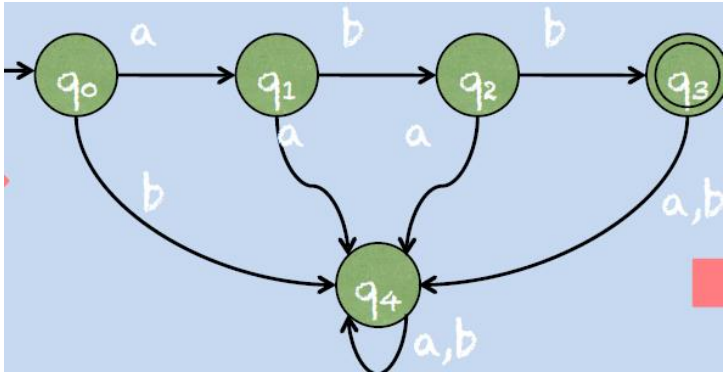


DFA funzionamento

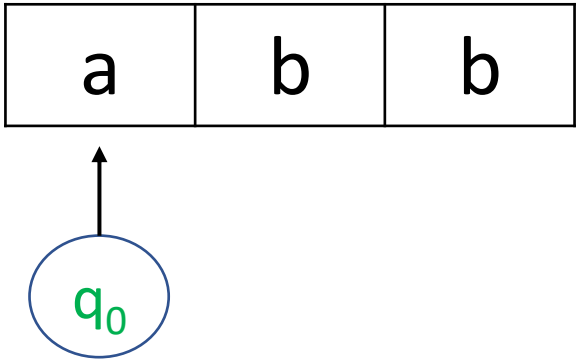
Input string: *abb*



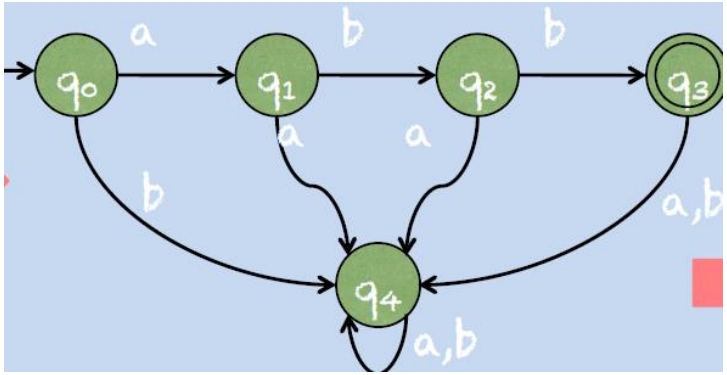
Un altro punto di vista



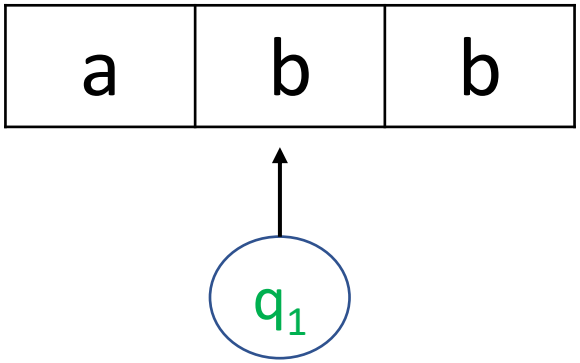
Input string: abb



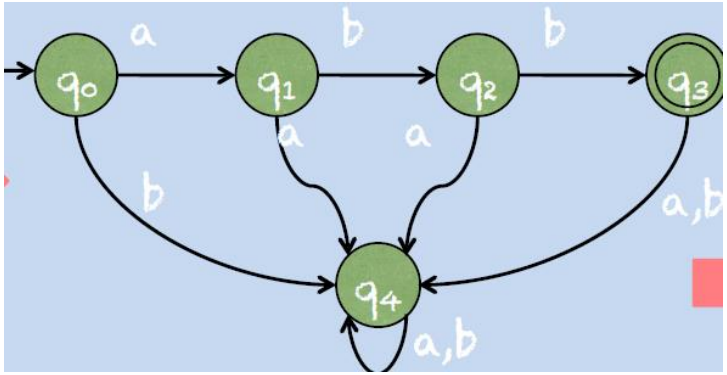
Un altro punto di vista



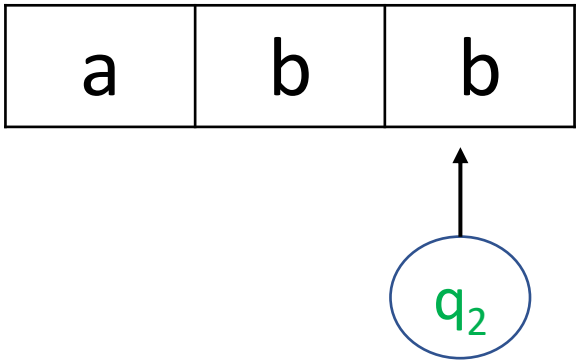
Input string: abb



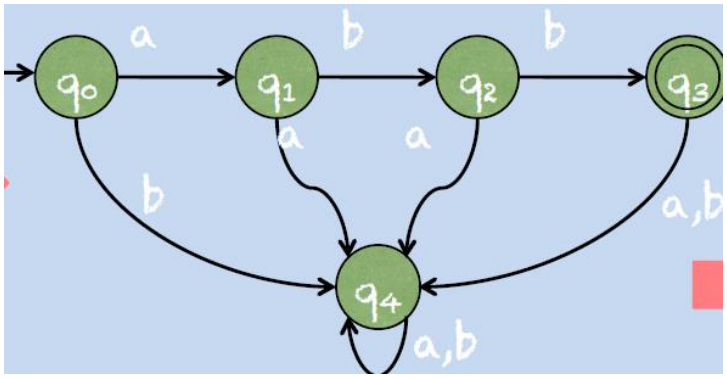
Un altro punto di vista



Input string: abb



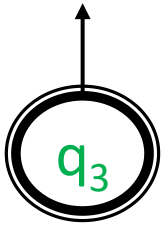
Un altro punto di vista



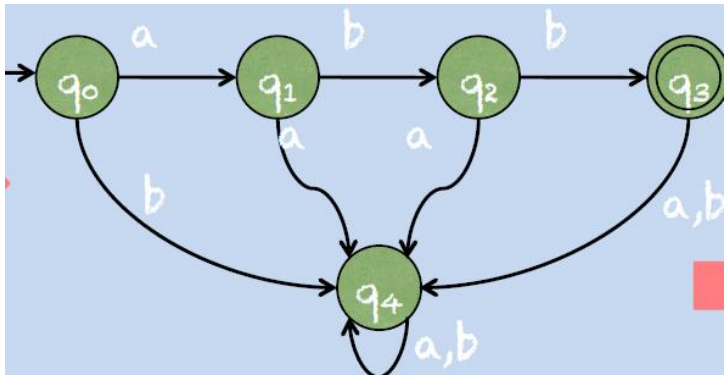
ACCEPT

Input string: abb

a	b	b



DFA o «algoritmo»?



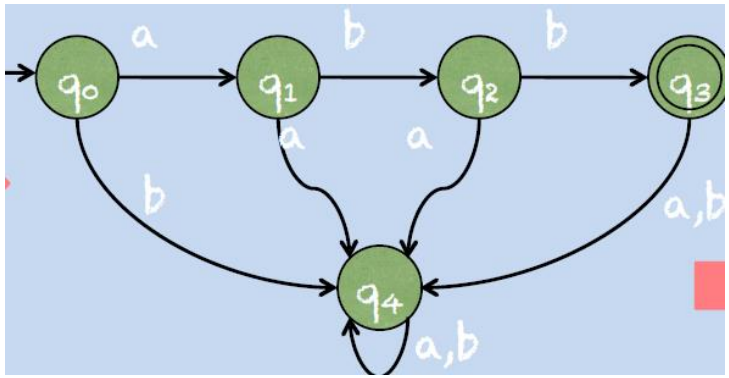
q_0 = finora non ho letto niente (ε)
q_1 = finora ho letto **a**
q_2 = finora ho letto **ab**
q_3 = finora ho letto **abb**
q_4 = ho letto qualcosa che non mi può portare a leggere **abb**

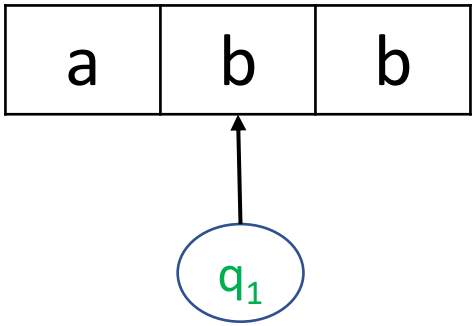
```
Accetto-abb( a1a2 … an )
if a1=b then return FALSE
   else if a2=a then return FALSE
      else if a3=a then return FALSE
         else if n>3 then return FALSE
Return TRUE
```

q_0 q_1 q_2 q_3 q_4 q_4 q_4

Nota: algoritmo senza variabili che permetterebbero di «ricordare» altro

Limiti di un DFA





Memoria finita:

q_0 = finora non ho letto niente (ε)

q_1 = finora ho letto **a**

q_2 = finora ho letto **ab**

q_3 = finora ho letto **abb**

q_4 = ho letto qualcosa che non mi può portare a leggere **abb**

Non posso «ricordare» altro

Non posso «annotare» altro

Mi posso spostare solo verso destra

Non posso oltrepassare i limiti della stringa

Una stringa in input è accettata o è rifiutata dopo averla letta interamente dall'inizio alla fine

Limiti di un DFA

Alcuni calcoli, pur se molto semplici, vanno oltre le capacità di un DFA.

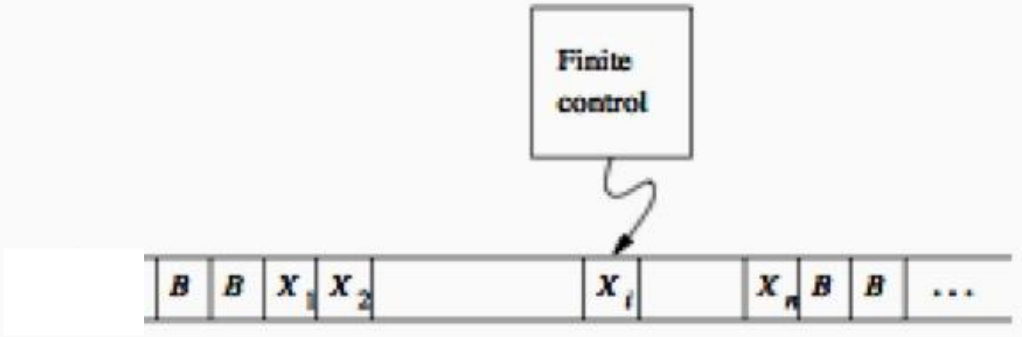
Non può riconoscere linguaggi come $\{ a^n b^n \mid n \geq 0 \}$.

Un DFA «non sa contare», o almeno può contare fino a ….

Esistono modelli con maggiori capacità di un DFA, per esempio automi a pila, automi *linear bounded*, ma il modello con le stesse capacità di un computer è la Macchina di Turing.

Simile ad un DFA (specie col secondo punto di vista), ma con memoria illimitata e senza altre restrizioni.

MdT: in breve

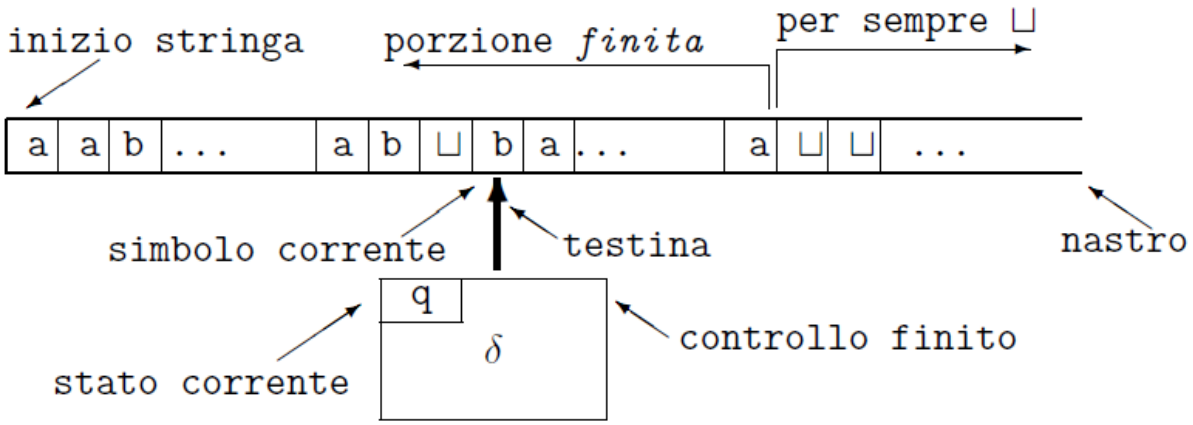


Una TM fa una **mossa** in funzione del suo stato, e del simbolo sotto la testina di lettura del nastro.
In una mossa, una TM

1. cambia stato
2. scrive un simbolo del nastro nella cella sotto la testina
3. muove la testina di una cella verso destra o verso sinistra

MdT:
più precisamente

Schema di una Macchina di Turing.



Descrizione informale di MdT

- Una macchina di Turing utilizza un nastro semi-infinito.
- Inizialmente il nastro contiene solo la stringa di input.
- La macchina di Turing ha una testina di lettura e scrittura, che può muoversi lungo il nastro.
- In funzione dello stato in cui si trova e del simbolo input letto, la macchina cambia stato, scrive un simbolo nella cella su cui era posizionata, sposta la testina di una posizione a destra o a sinistra.
- La macchina continua a computare finché non **accetta** o **rifiuta**.
- Se non raggiunge uno stato corrispondente a uno delle due situazioni precedenti, la computazione **andrà avanti per sempre**.

Come riconoscere $\{ a^n b^n \mid n \geq 0 \}$ con MdT

Cancella ripetutamente: prima occorrenza di (a) e ultima di (b) se la stringa era del tipo $a^n b^n$, non rimangono simboli

Come riconoscere $\{ a^n b^n \mid n \geq 0 \}$ con MdT

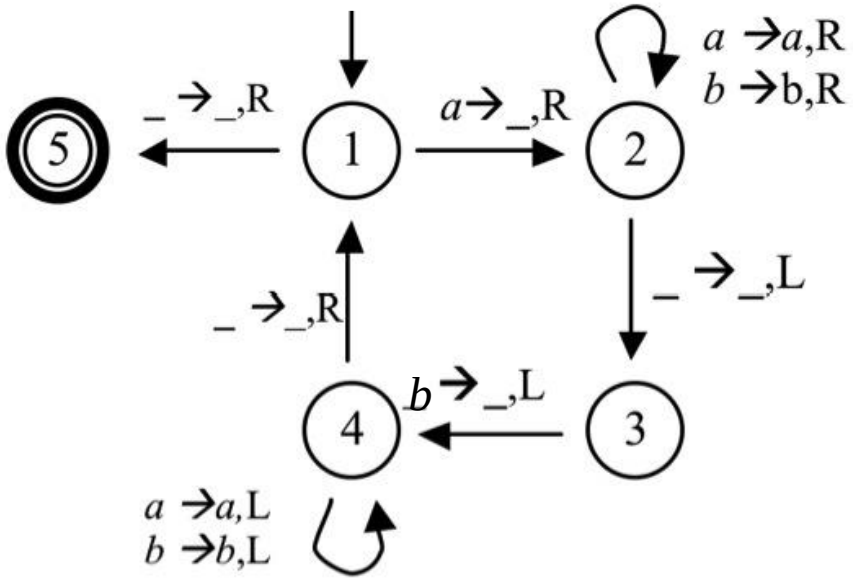
Cancella ripetutamente: prima occorrenza di (a) e ultima di (b) se la stringa era del tipo $a^n b^n$, non rimangono simboli
Cinque passi

1. Se leggi _, vai a 5. Se leggi a, scrivi _ e vai a 2.
2. Spostati a destra (R) di tutti a e b. Al primo _, muovi a sinistra (L) e vai a 3
3. se leggi b, scrivi _ e vai a 4.
4. Spostati a sinistra (L) di tutti a e b. Leggendo _, muovi R e vai a 1.
5. Accept.

Nei casi non indicati, Reject (per esempio se in 1. leggi b).

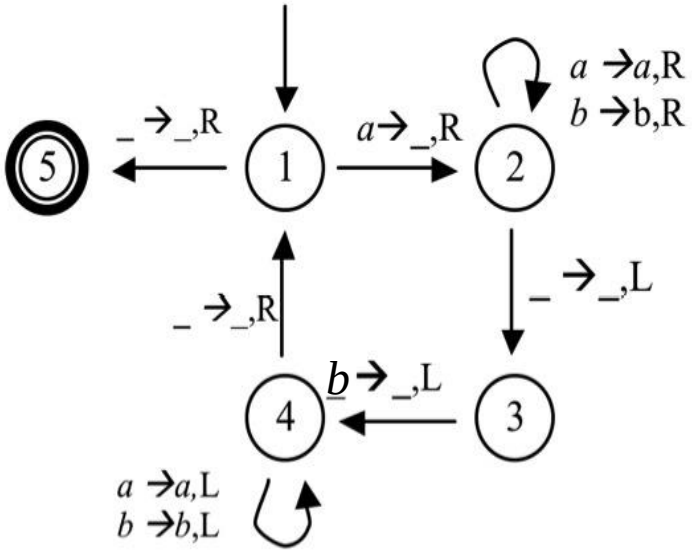
Cancella ripetutamente: prima occorrenza di (a) e ultima di (b)
se la stringa era del tipo $a^n b^n$, non rimangono simboli
Cinque passi

1. Se leggi _, vai a 5. Se leggi a, scrivi _ e vai a 2.
2. Spostati a destra (R) di tutti a e b. Al primo _, muovi a sinistra (L) e vai a 3
3. se leggi b, scrivi _ e vai a 4.
4. Spostati a sinistra (L) di tutti a e b. Leggendo _, muovi R e vai a 1.
5. Accept.



Le transizioni non indicate portano in uno stato di Reject (per esempio se in 1. leggi b).

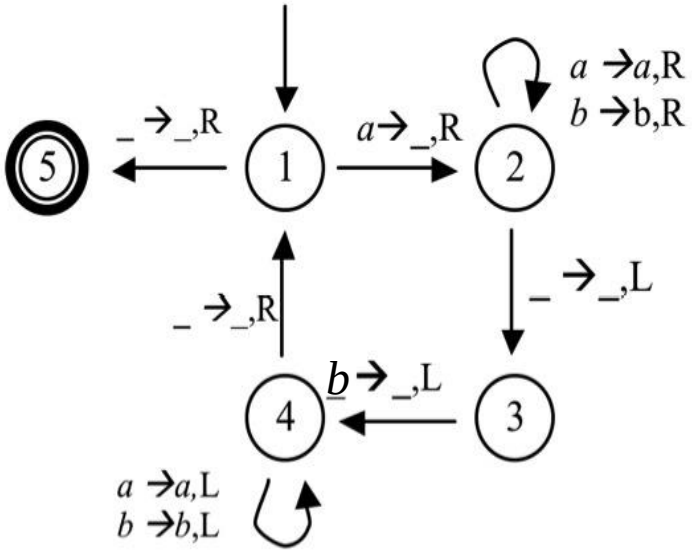
MdT all'opera su *aaaabbbb*

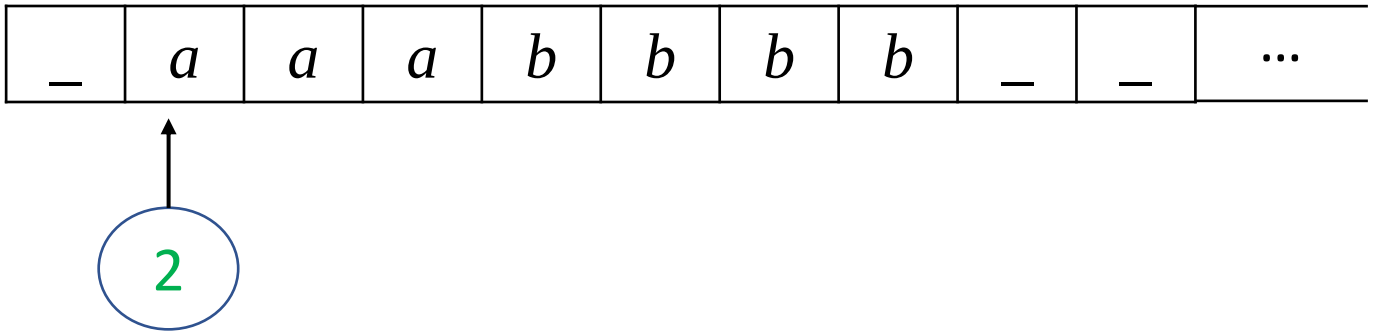


a	a	a	a	b	b	b	b	_	_	...

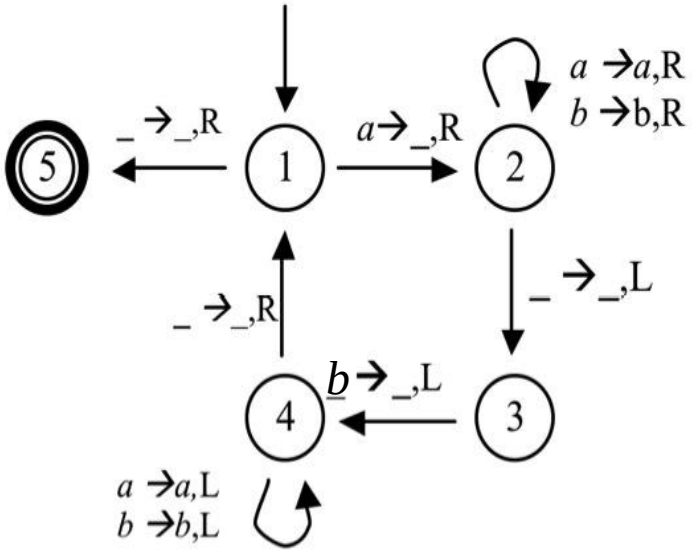
1

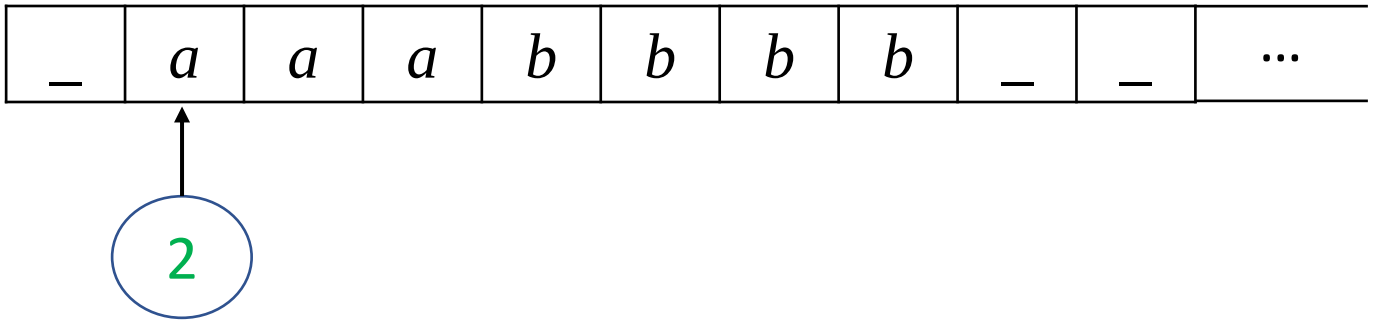
MdT all'opera su *aaaabbbb*



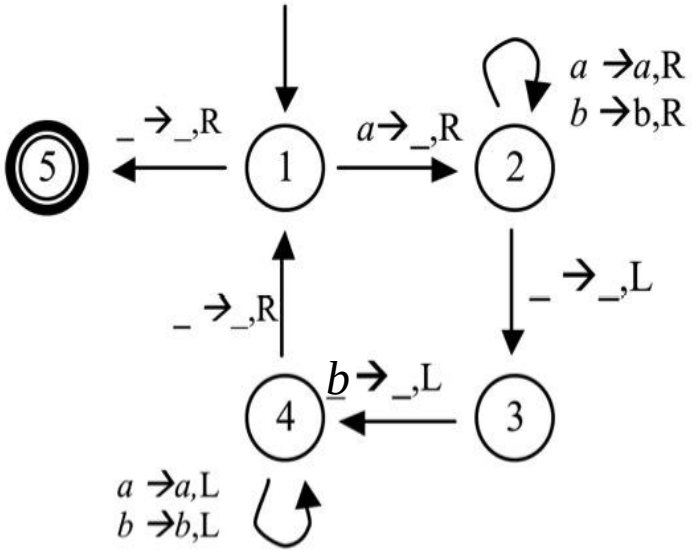


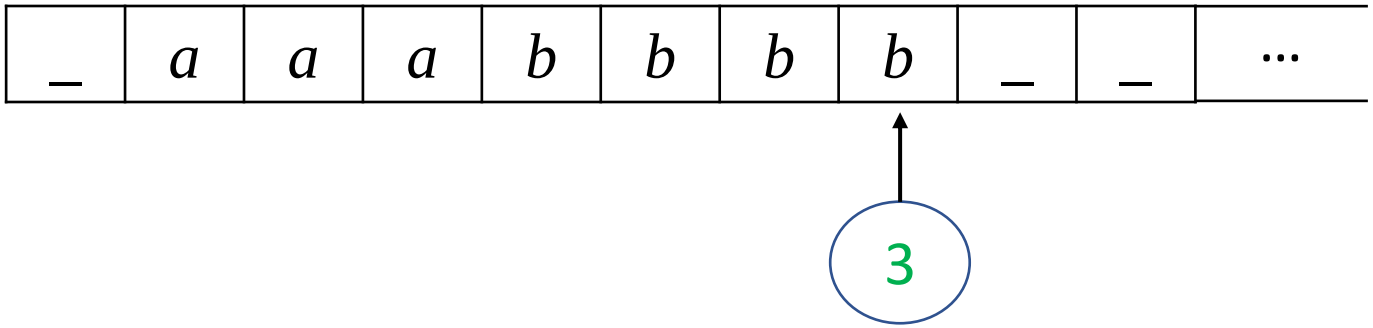
MdT all'opera su *aaaabbbb*



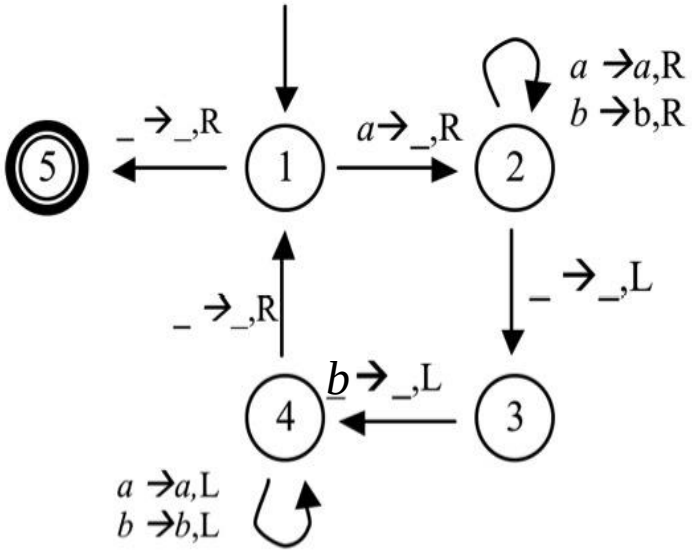


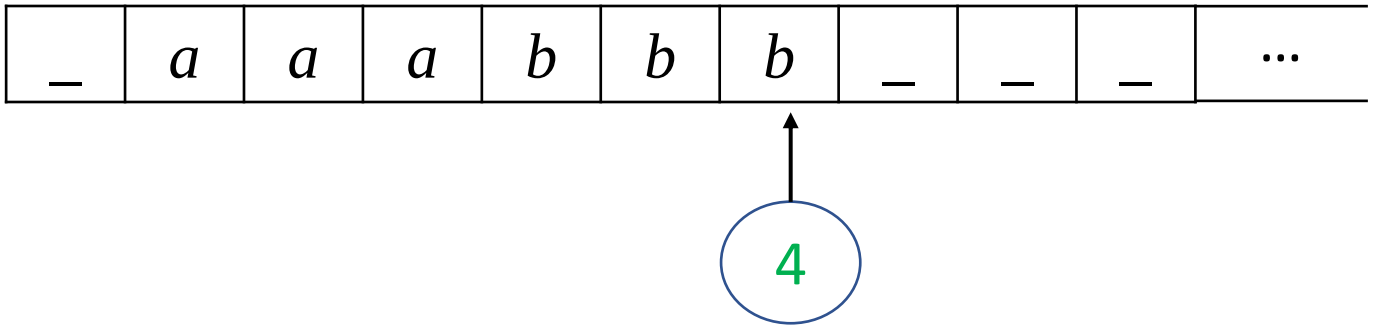
MdT all'opera su *aaaabbbb*



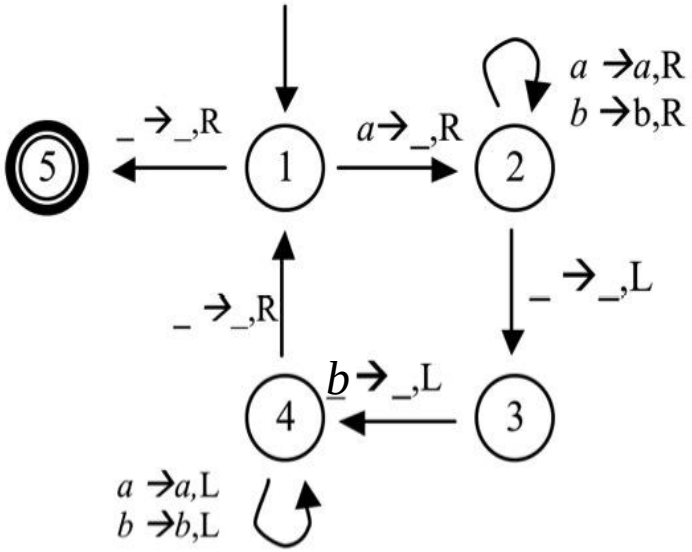


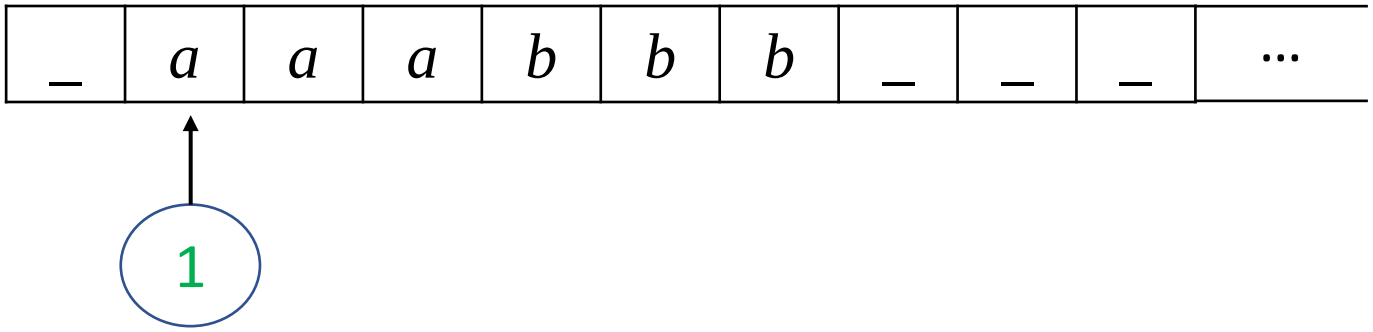
MdT all'opera su *aaaabbbb*



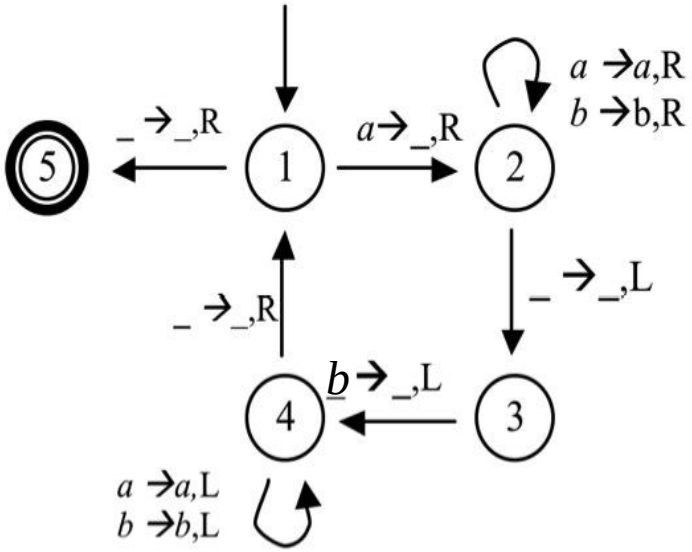


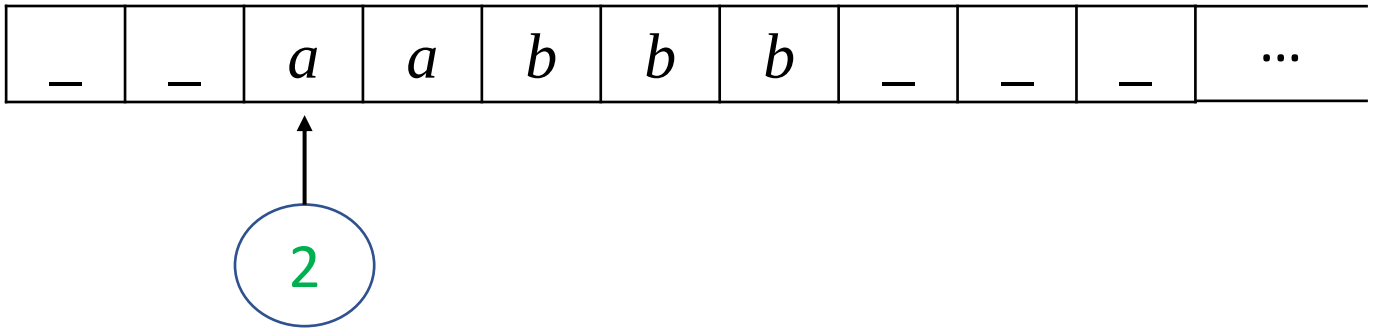
MdT all'opera su *aaaabbbb*



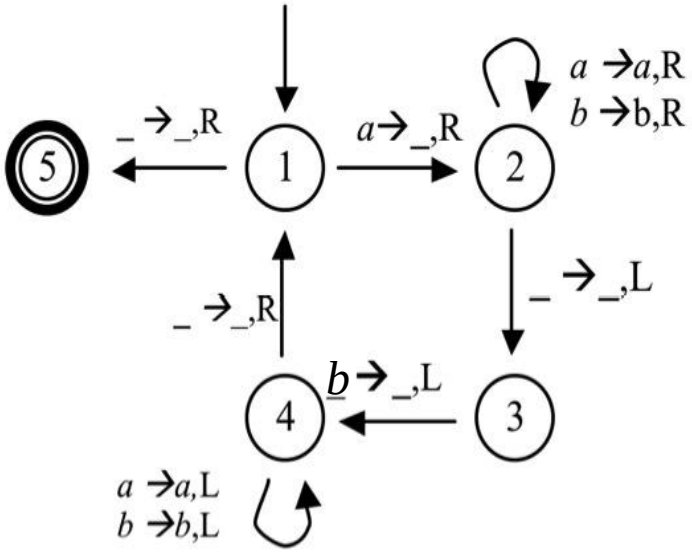


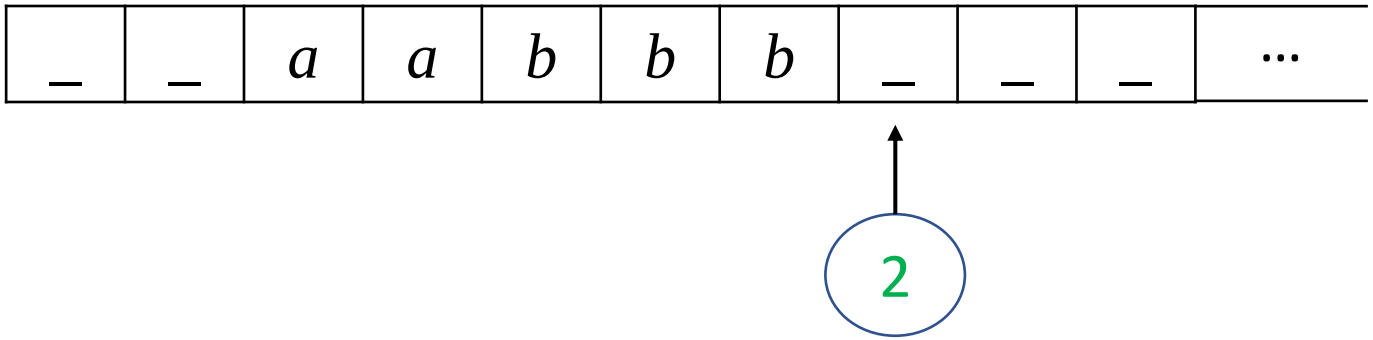
MdT all'opera su *aaaabbbb*



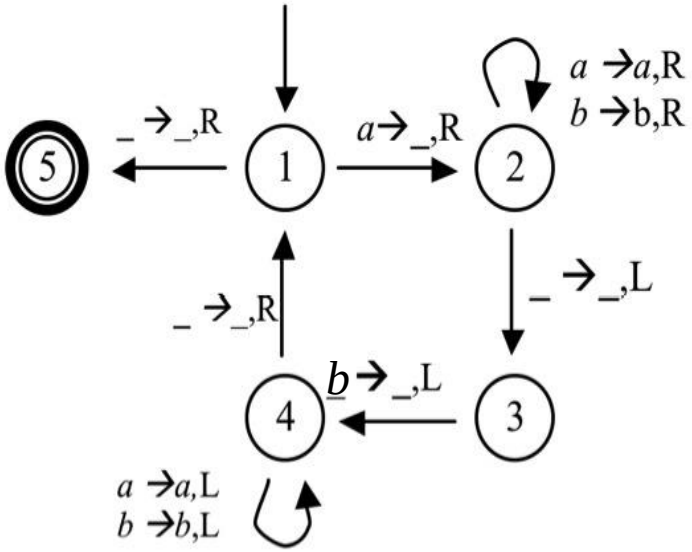


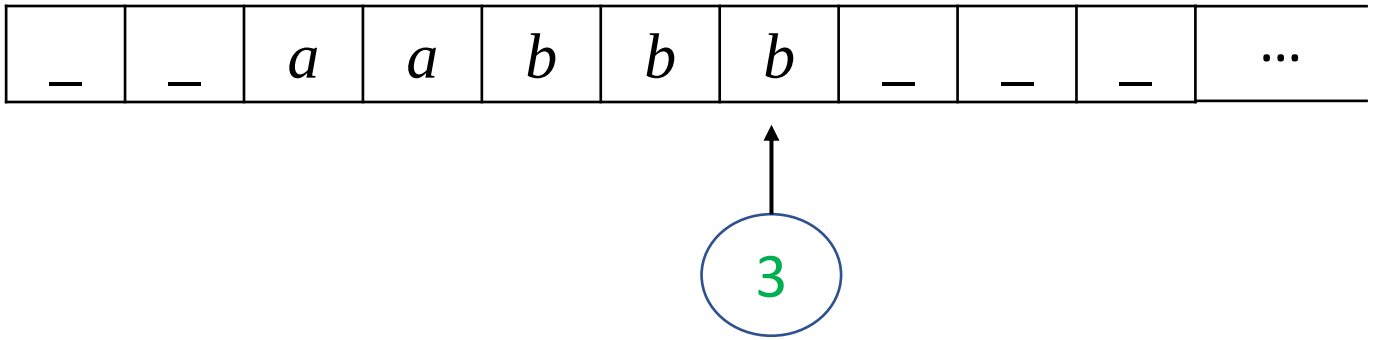
MdT all'opera su *aaaabbbb*



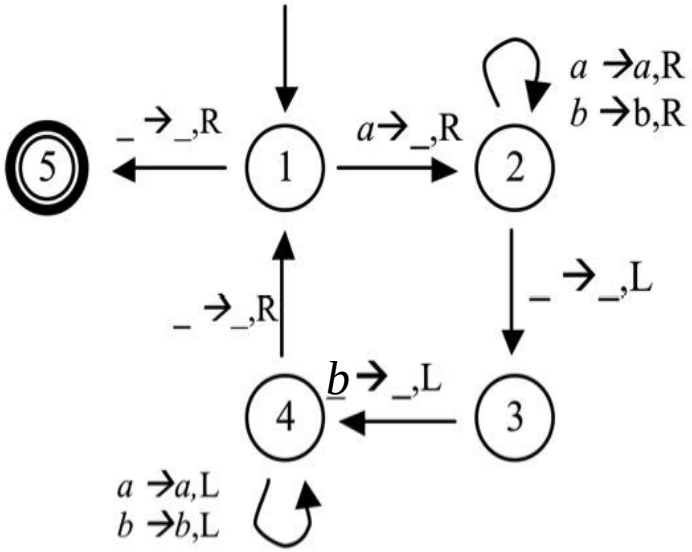


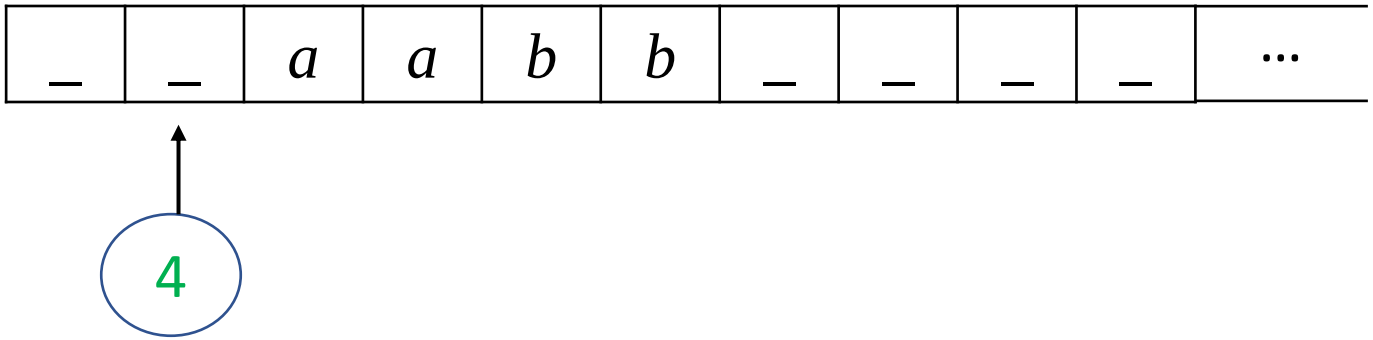
MdT all'opera su *aaaabbbb*



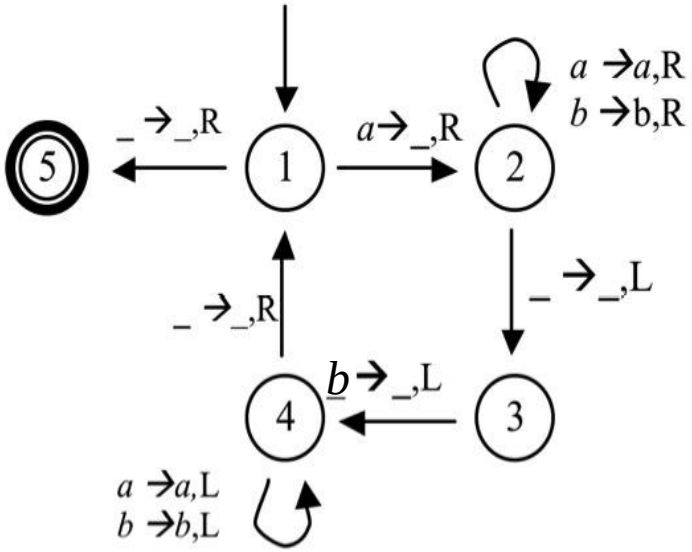


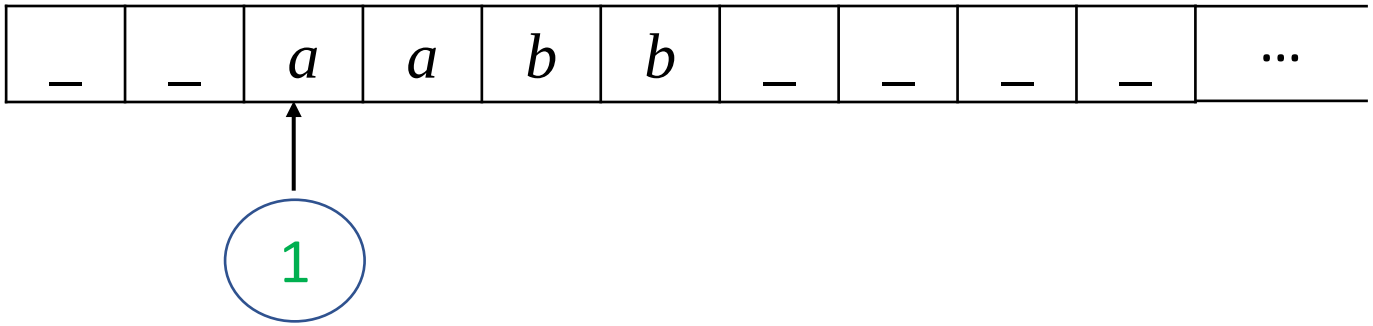
MdT all'opera su *aaaabbbb*



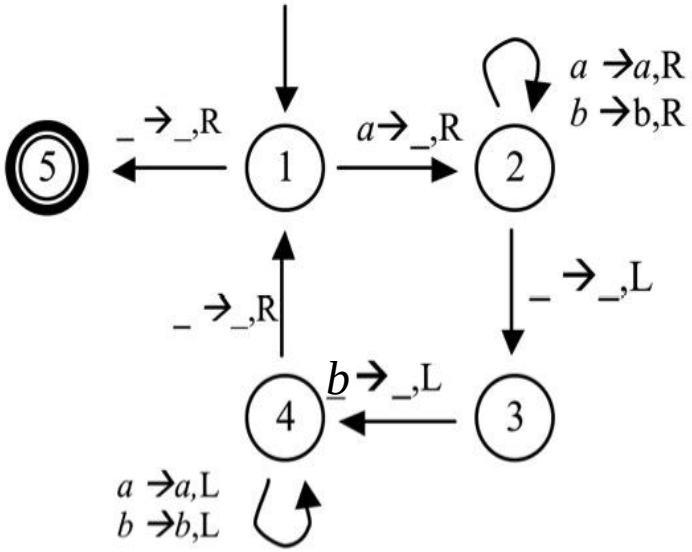


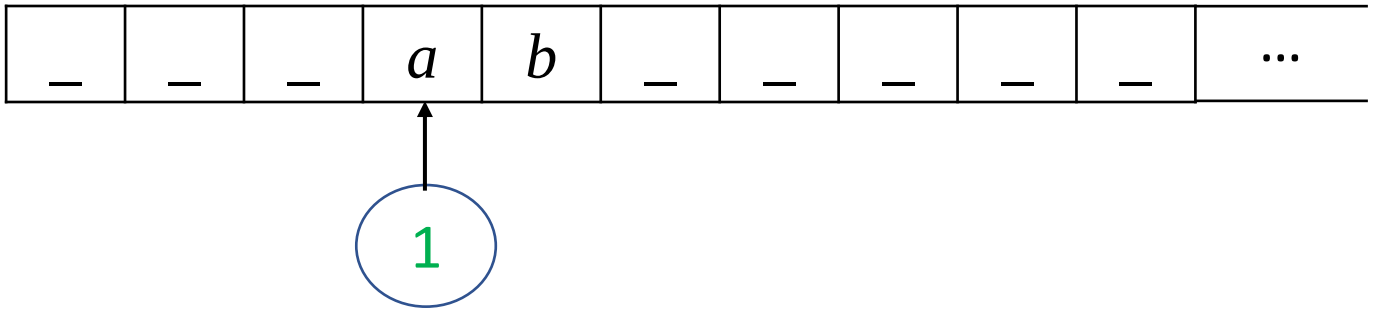
MdT all'opera su *aaaabbbb*



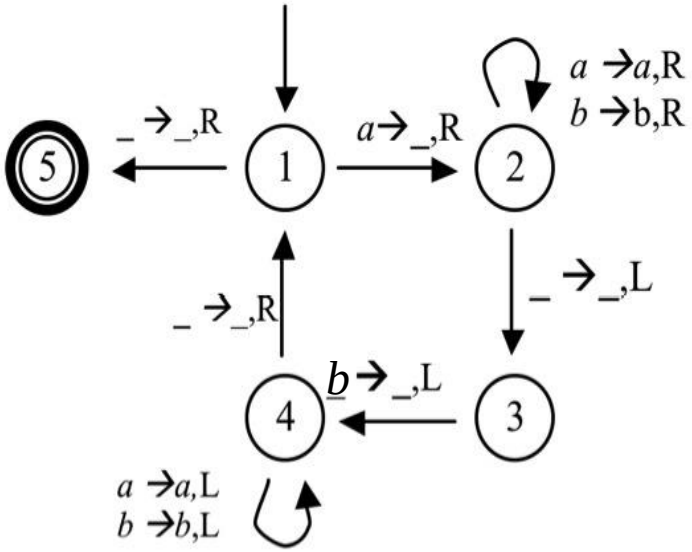


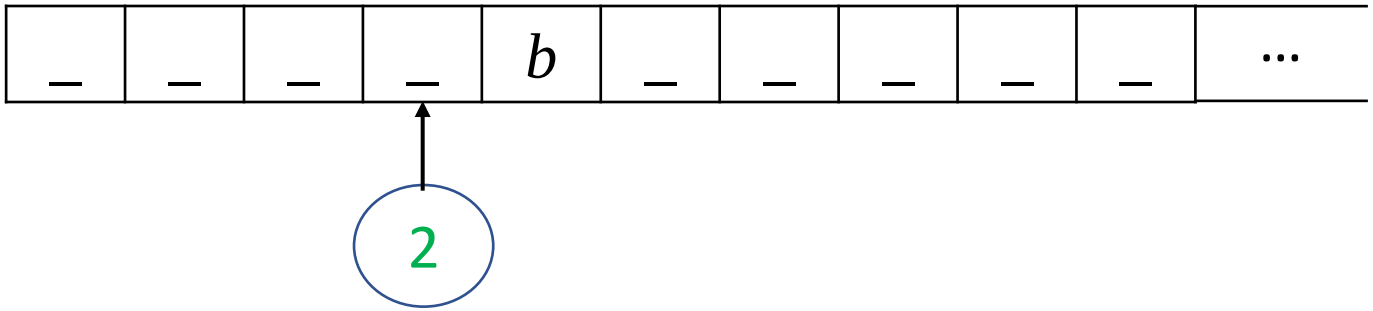
MdT all'opera su *aaaabbbb*



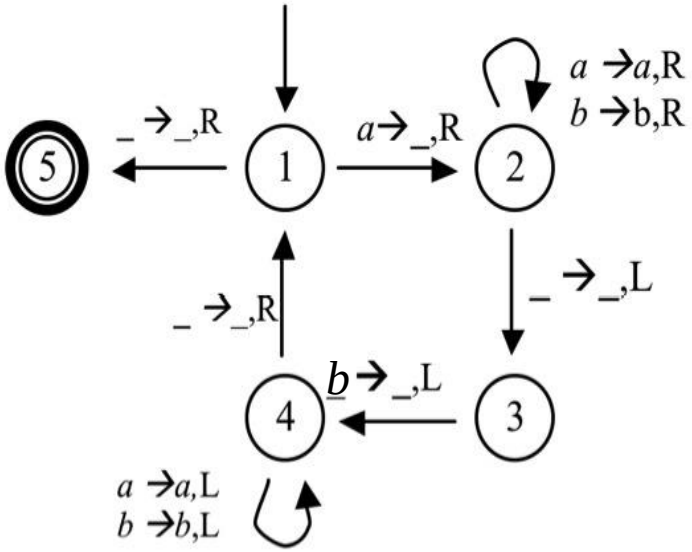


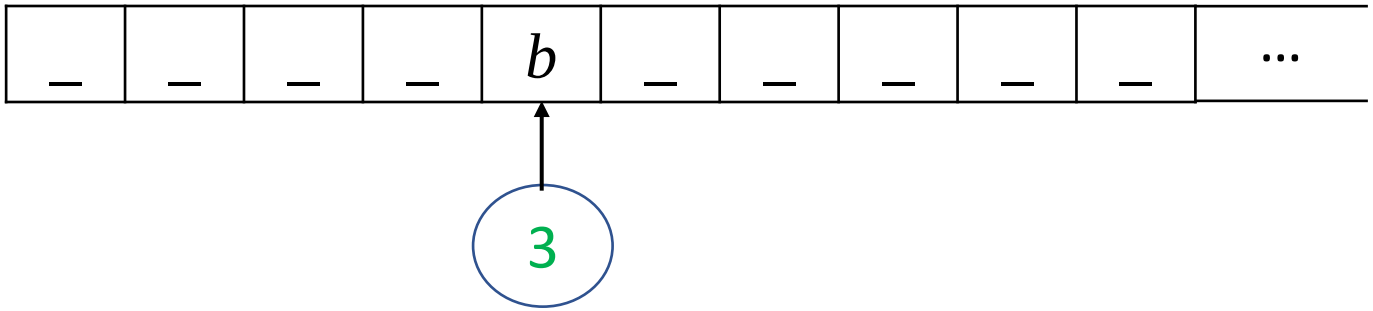
MdT all'opera su *aaaabbbb*



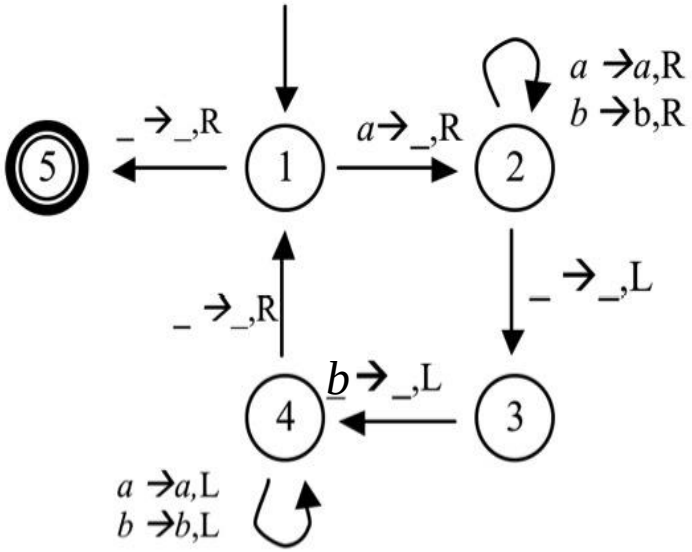


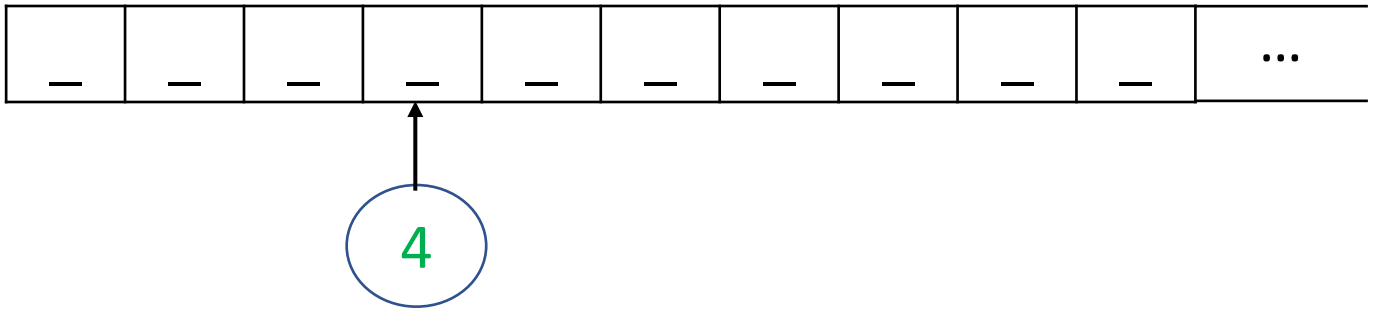
MdT all'opera su *aaaabbbb*



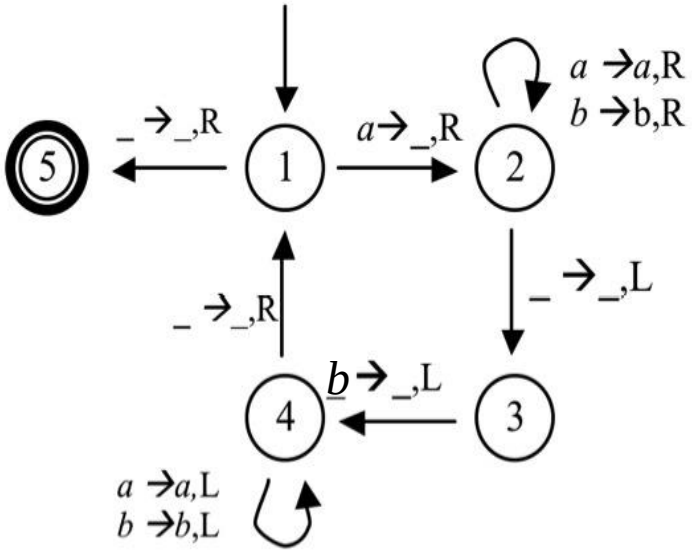


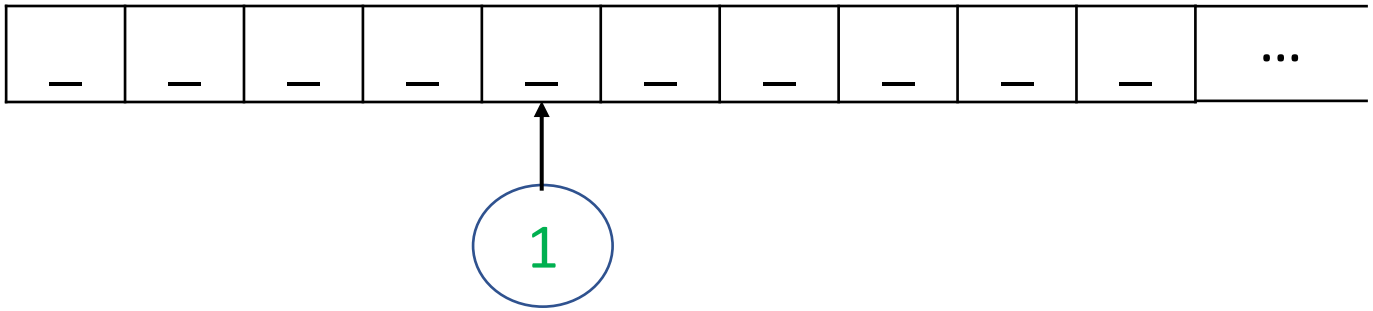
MdT all'opera su *aaaabbbb*



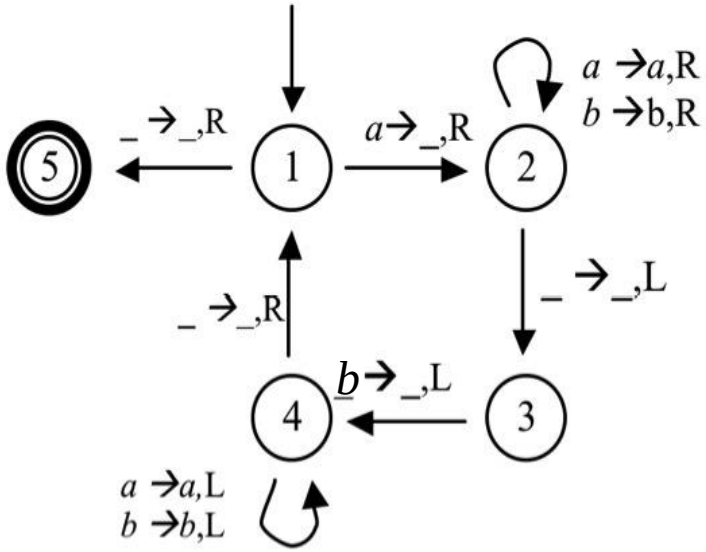


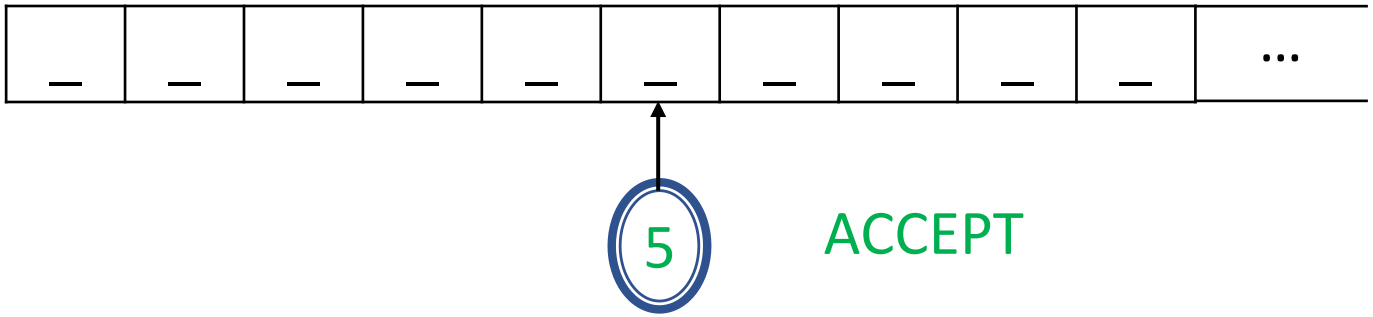
MdT all'opera su *aaaabbbb*



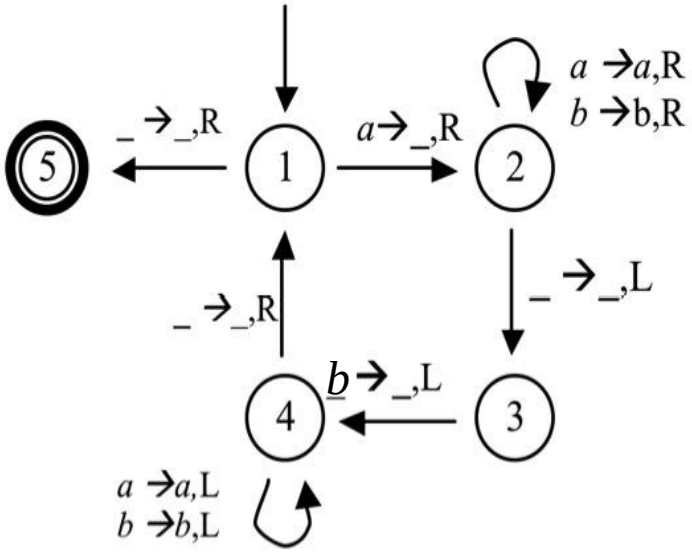


MdT all'opera su *aaaabbbb*





MdT all'opera su *aaababbb*



a	*a*	*a*	*b*	*a*	*b*	*b*	*b*	_	_	···

Altra strategia MdT per $\{ 0^n1^n \mid n \geq 0 \}$.

Una macchina di Turing che accetta $L = \{0^n1^n \mid n > 0\}$.
(Descrizione ad alto livello)

(passo 1) Se legge 0 lo sostituisce con X, se legge 1 rifiuta, se legge Y va al passo 4.

(passo 2) Scorre il nastro verso destra, se trova 1 lo sostituisce con Y, altrimenti rifiuta.

(passo 3) Scorre il nastro a sinistra fino a incontrare X, si sposta a destra e ripete dal passo 1.

(passo 4) Scorre il nastro a destra. Se legge solo delle Y e poi il carattere $\sqcup$, accetta. Altrimenti rifiuta.

$000111 \to X00111\ldots \to X00Y11\ldots \to XX0Y11\ldots \to XXXYYY$

Descrizione formale MdT

Una Macchina di Turing è una settupla

$$(Q, \Sigma, \Gamma, \delta, q_o, q_{accept}, q_{reject})$$

- **Insieme Stati** Q
- **Alfabeto di lavoro** Σ $(\_ \notin \Sigma)$
- Γ: **Alfabeto del nastro** $(\_ \in \Gamma,\ \Sigma \subset \Gamma)$
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$: funzione transizione
- q_0: stato **iniziale**
- q_{accept}: stato **accept**
- q_{reject}: stato **reject**

Descrizione ancora più formale Mdt

Definizione

Una Macchina di Turing **deterministica** *è una settupla*

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$$

dove:

- Q: **Insieme finito degli stati**
- Σ: **Alfabeto dei simboli input** $(\sqcup \notin \Sigma)$
- Γ: **Alfabeto (finito) dei simboli di nastro** $(\sqcup \in \Gamma,\ \Sigma \subset \Gamma,\ L, R \notin \Gamma)$
- $\delta : (Q \setminus \{q_{accept}, q_{reject}\}) \times \Gamma \to Q \times \Gamma \times \{L, R\}$: **funzione transizione**
- $q_0 \in Q$: *stato* **iniziale**
- $q_{accept} \in Q$: *stato di* **accettazione**
- $q_{reject} \in Q$: *stato di* **rifiuto**, $q_{accept} \neq q_{reject}$

Funzione di transizione

Sia $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ una MdT deterministica.
Il suo comportamento è descritto dalla funzione di transizione

$$\delta : (Q \setminus \{q_{accept}, q_{reject}\}) \times \Gamma \to Q \times \Gamma \times \{L, R\}.$$

Se $\delta(q, \gamma) = (q', \gamma', d)$, sappiamo che $q, q' \in Q$, $\gamma, \gamma' \in \Gamma$, $d \in \{L, R\}$.

Se $\delta(q, \gamma) = (q', \gamma', d)$ e se M si trova nello stato q con la testina posizionata su una cella contenente γ, alla fine della transizione:

- M si troverà nello stato q',
- $\gamma' \in \Gamma$ sarà il simbolo scritto sulla cella del nastro su cui la testina si trovava ALL'INIZIO della transizione (contenente γ),
- la testina si sarà spostata sulla cella di sinistra se (tale cella esiste e se) $d = L$, sulla cella di destra se $d = R$.

NOTA: Se $\delta(q, \gamma) = (q', \gamma', L)$, se M si trova nello stato q con la testina posizionata su una cella contenente γ e se tale cella è quella più a sinistra del nastro, la testina non si sposta, resta sulla cella del nastro su cui si trovava all'inizio della transizione.

Diagramma di stato

Come nel caso degli automi, spesso preferiamo utilizzare il diagramma di stato di una specifica Macchina di Turing piuttosto che la descrizione formale della settupla.

Il diagramma di stato di una Macchina di Turing è un grafo i cui nodi hanno come nomi gli stati della macchina.

Le etichette degli archi hanno la forma

$$\text{“simbolo} \rightarrow \text{simbolo, } d\text{”}$$

dove “simbolo” è un simbolo di Γ e $d \in \{L, R\}$. Il simbolo a sinistra della freccia rappresenta il simbolo letto, quello dopo la freccia il simbolo (eventualmente lo stesso) che sostituisce il simbolo letto per effetto della transizione, d lo spostamento per effetto della transizione.

Diagramma di stato

Ad esempio se sull'arco dal nodo q_1 al nodo q_2 compare l'etichetta

$$0 \to \sqcup, R$$

questo corrisponde a

$$\delta(q_1, 0) = (q_2, \sqcup, R)$$

Alle volte si usa l'abbreviazione di non indicare il carattere dopo la freccia se esso non è modificato dalla transizione.

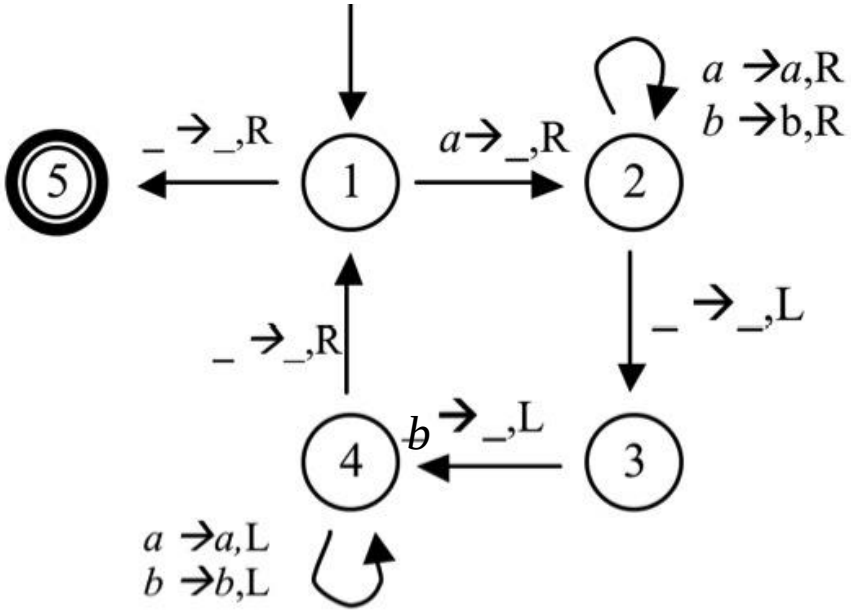
Ad esempio se sull'arco dal nodo q_3 al nodo q_4 troviamo l'etichetta

$$0 \to R$$

questo vuol dire che

$$\delta(q_3, 0) = (q_4, 0, R)$$

Esempio diagramma di stato



Una Macchina di Turing è una settupla

$$(Q, \Sigma, \Gamma, \delta, q_o, q_{accept}, q_{reject})$$

- **Insieme Stati** Q
- **Alfabeto di lavoro** Σ $(\_ \notin \Sigma)$
- Γ: **Alfabeto del nastro** $(\_ \in \Gamma,\ \Sigma \subset \Gamma)$
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$: funzione transizione
- q_0: stato **iniziale**
- q_{accept}: stato **accept**
- q_{reject}: stato **reject**

Un Automa Finito (DFA) è una quintupla

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

- **Insieme Stati** Q
- **Alfabeto** $\sum$
- δ : Q x $\sum$ -> Q : funzione di transizione
- $q_0 \in Q$: stato **iniziale**
- $F \subseteq Q$: stati **finali**

Caccia alle differenze!

Differenze fra MdT e DFA

La definizione di Macchina di Turing (in seguito abbreviata con MdT o con TM) che daremo, metterà in evidenza le seguenti differenze con il modello automa finito:

- Una MdT ha un nastro semi-infinito (infinito a destra), diviso in celle, ciascuna contenente un carattere.
- Una MdT ha una testina che può muoversi a sinistra o a destra (ma non può oltrepassare il limite a sinistra del nastro).
- La testina può leggere o scrivere caratteri (testina di lettura-scrittura).
- Una MdT ha due stati speciali q_{accept}, q_{reject} rispettivamente corrispondenti all'accettazione e al rifiuto della stringa input. Se la MdT si trova in uno di tali stati non può effettuare ulteriori passi di computazione. (La frase sul testo di Sipser è: **Gli stati speciali di accettazione e rifiuto hanno effetto immediato.**)
- Non c'è limite ai passi di computazione sulla stringa input (in particolare il numero di tali passi non è limitato dalla lunghezza dell'input).

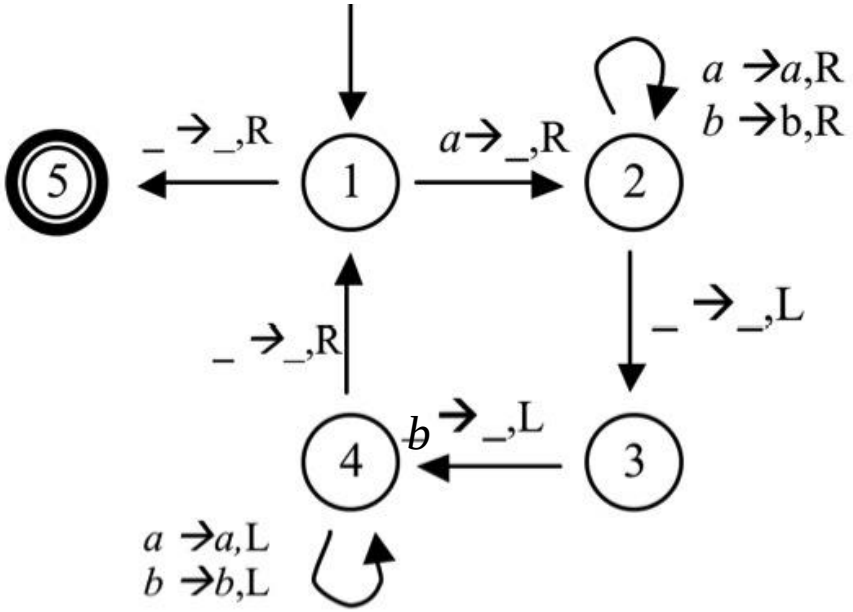
Differenze fra DFA e MdT

The following list summarizes the differences between finite automata and Turing machines.

1. A Turing machine can both write on the tape and read from it.
2. The read–write head can move both to the left and to the right.
3. The tape is infinite.
4. The special states for rejecting and accepting take effect immediately.

Se la MdT si trova in uno di tali stati non può effettuare ulteriori passi di computazione.

Dal diagramma di stato alla settupla



M = ….

MdT o «algoritmo»?

Cancella ripetutamente: prima occorrenza di (a) e ultima di (b)
se la stringa era del tipo $a^n b^n$, non rimangono simboli
Cinque passi

1. Se leggi _, vai a 5. Se leggi a, scrivi _ e vai a 2.
2. Spostati a destra (R) di tutti a e b. Al primo _, muovi a sinistra (L) e vai a 3
3. se leggi b, scrivi _ e vai a 4.
4. Spostati a sinistra (L) di tutti a e b. Leggendo _, muovi R e vai a 1.
5. Accept.

Sapreste scrivere lo pseudo-codice di questo algoritmo?

`Accetto-anbn( a`$_1$`a`$_2$ … `a`$_m$ `)`

……

Linguaggi di stringhe o problemi?

Come posso usare una MdT per risolvere un problema?

Per esempio, se volessi trovare un Minimum Spanning Tree di un grafo o decidere se un grafo è bipartito?