

# Elementi di Calcolo e Complessità

Federico Matteoni

A.A. 2019/20



# Indice

<b>1</b>	<b>Calcolabilità</b>	<b>5</b>
1.1	Teoria della Calcolabilità . . . . .	5
1.2	Algoritmo . . . . .	5
1.3	Macchina di Turing . . . . .	6
1.3.1	$\Sigma$ . . . . .	6
1.3.2	Transizioni . . . . .	6
1.3.3	Computazione . . . . .	6

## Introduzione

Prof. Pierpaolo Degano [pierpaolo.degano@unipi.it](mailto:pierpaolo.degano@unipi.it)  
Con Giulio Masetti [giulio.masetti@isti.snr.it](mailto:giulio.masetti@isti.snr.it)  
Esame: compitini/scritto + orale

# Capitolo 1

## Calcolabilità

### 1.1 Teoria della Calcolabilità

Illustra **cosa può essere calcolato da un computer** senza limitazioni di risorse come spazio, tempo ed energia. Vale a dire:

- Quali sono i **problemi *solubili*** mediante una **procedura effettiva** (qualunque linguaggio su qualunque macchina)?
- Esistono **problemi *insolubili***? Sono interessanti, realistici, oppure puramente artificiali?
- Possiamo raggruppare i problemi in **classi**?
- Quali sono le **proprietà** delle classi dei problemi solubili?
- Quali sono le relazioni tra le classe dei problemi insolubili?

**Astrazione** Utilizzeremo **termini astratti per descrivere la possibilità di eseguire un programma ed avere un risultato**. Questa astrazione è un **modello** che non tiene conto di dettagli al momento irrilevanti.

Un po' come l'equazione per dire quanto ci mette il gesso a cadere che non tiene conto delle forze di attrito dell'aria.

**Problema della Decisione** Un problema è risolto se si conosce una **procedura** che permette di decidere con un numero **finito** di operazioni di decedere se una proposizione logica è vera o falsa.

### 1.2 Algoritmo

Un algoritmo è un insieme **finito** di istruzioni.

**Istruzioni** Elementi da un insieme di **cardinalità finita** ed ognuna ha **effetto limitato** (localmente e "poco") sui dati (che devono essere **discreti**). Un'istruzione deve richiedere tempo finito per essere elaborata.

**Computazione** Successione di istruzioni finite in cui ogni passo dipende solo dai precedenti. Verificando una porzione finita dei dati (**deterministico**). Non c'è limite alla memoria necessaria al calcolo (è finita ma illimitata). Neanche il tempo è limitato (necessario al calcolo). Tanto tempo e tanta memoria quante ce ne servono.

Un'eccezione a questa definizione di algoritmo è costituita dalle macchine concorrenti/interattive, dove gli input variano nel tempo. Inoltre vi sono formalismi che tengono conto di algoritmi probabilistici e stocastici. Altre eccezioni sono gli algoritmi non deterministici, ma per ognuno di essi esiste un algoritmo deterministico equivalente (Teorema 3.3.6)

### 1.3 Macchina di Turing

Introdotta da **Alan Turing** nel 1936, confuta la speranza "*non ignorabimus*" di poter risolvere qualsiasi cosa con un programma.

Turing originariamente la presenta supponendo di aver un impiegato precisissimo ma stupido, con una pila di fogli di carta ed una penna, ed un foglio di carta con le istruzioni che esegue con estrema diligenza. Non capisce quello che fa, e si chiama "**computer**".

**Struttura matematica** Una Macchina di Turing (MdT) è una quadrupla:

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

$Q = \{q_i\}$  è l'insieme finito degli **stati** in cui si può trovare la macchina.

Indicheremo con lo stato speciale  $h$  la fine corretta della computazione,  $h \notin Q$ .

$\Sigma = \{\sigma, \sigma' \dots\}$  è l'insieme finito di **simboli**. Ci sono elementi che devono per forza esistere:

# carattere **bianco**, vuoto

▷ carattere di inizio della memoria, chiamato **respingente**, che funziona come un inizio file

$\delta \subseteq (Q \times \Sigma) \rightarrow (Q' \cup \{h\}) \times \Sigma' \times \{L, R, -\}$  è **funzione di transizione**.

Mantiene determinismo perché funzione, ad un elemento associa un solo elemento (la transizione è univoca).

Transizioni finite perché prodotto cartesiano di insiemi finiti.

$\delta(q, \triangleright) = (q', \triangleright, R)$ , cioè se sono a inizio file possono solo andare a destra.

Può essere vista come una relazione di transizione,  $\delta \subseteq (Q \times \Sigma) \times (Q' \cup \{h\}) \times \Sigma' \times \{L, R, -\}$

$q_0 \in Q$  lo **stato iniziale**

Mappatura a coda di rondine, bigezione tra  $(m, n) \rightarrow k$ , cioè  $N^2 \rightarrow N$ .

Costruire un modello per il calcolo dopo aver posto delle condizioni affinché qualcosa si possa chiamare algoritmo.

#### 1.3.1 $\Sigma$

$\Sigma^0 = \{\epsilon\}$ , con  $\epsilon$  = parola vuota, che non contiene caratteri

$\Sigma^{i+1} = \Sigma \cdot \Sigma^i = \{\sigma \cdot u \mid \sigma \in \Sigma \wedge u \in \Sigma^i\}$

$\Sigma^* = \bigcup_{i \in \mathbb{N}} \Sigma^i$ , insieme di tutte le possibili combinazioni di simboli

$\Sigma^f = \Sigma^* \cdot (\Sigma - \{\#\} \cup \{\epsilon\})$ , cioè l'insieme di tutte le stringhe che terminano con un carattere non bianco ma può terminare con la stringa vuota

**Esempio**  $\Sigma_B = \{0, 1\} \longrightarrow \Sigma_B^* = \{\epsilon, 0, 1, 01, 10, 010, 110010, \dots\}$  tutti i numeri binari

#### 1.3.2 Transizioni

La **situazione corrente** di una macchina di Turing può essere scritto come  $(q, u, \sigma, v)$  dove:

$q$  è lo **stato attuale**,  $q \in Q$

$u$  è la **stringa a sinistra** del carattere corrente,  $u \in \Sigma^*$

$\sigma$  è il **carattere corrente**,  $\sigma \in \Sigma$

$v$  è il **resto della stringa** che termina con un carattere non nullo,  $v \in \Sigma^f$

Può anche essere più comodamente espressa come  $(q, u \sqcup v)$

#### 1.3.3 Computazione

Una computazione è una transizione  $(q, x) \longrightarrow (q', \omega)$ . Una macchina di Turing parte **sempre** da  $(q_0, \sqcup x)$ .

Ogni computazione può esprimere il numero di passi necessari, ad esempio  $\gamma \xrightarrow{n} \gamma'$ .

∀ computazione  $\gamma \Rightarrow \gamma \xrightarrow{0} \gamma$ . Inoltre se  $\gamma \longrightarrow \gamma' \wedge \gamma' \xrightarrow{n} \gamma''$  allora  $\gamma \xrightarrow{n+1} \gamma''$

**Esempio** Macchina di Turing che esegue la semplice somma di due semplici numeri romani.

$q$	$\sigma$	$\delta(q, \sigma)$
$q_0$	$\triangleright$	$(q_0, \triangleright, R)$
$q_0$	I	$(q_0, I, R)$
$q_0$	+	$(q_1, I, R)$
$q_1$	I	$(q_1, I, R)$
$q_1$	#	$(q_2, \#, L)$
$q_2$	I	$(h, \#, -)$

$$\begin{aligned}
 (q_0, \triangleright II + III) &\longrightarrow (q_0, \triangleright \underline{II} + III) \longrightarrow (q_0, \triangleright II + \underline{III}) \longrightarrow \\
 (q_0, \triangleright II \pm III) &\longrightarrow (q_1, \triangleright III \underline{III}) \longrightarrow (q_1, \triangleright III \underline{III} \underline{I}) \longrightarrow \\
 (q_1, \triangleright III \underline{III} \underline{I}) &\longrightarrow (q_1, \triangleright III \underline{III} \underline{I} \underline{\#}) \longrightarrow (q_2, \triangleright III \underline{III} \underline{I}) \longrightarrow \\
 (h, \triangleright III \underline{III} \underline{I}) &
 \end{aligned}$$

**Esempio** Macchina di Turing che verifica se una stringa di lettere  $a, b$  è palindroma o no.

$q$	$\sigma$	$\delta(q, \sigma)$
$q_0$	$\triangleright$	$(q_0, \triangleright, R)$
$q_0$	$a$	$(q_A, \triangleright, R)$
$q_0$	$b$	$(q_B, \triangleright, R)$
$q_0$	#	$(h, \#, -)$
$q_A$	$a/b$	$(q_A, a/b, R)$
$q_A$	#	$(q_{A'}, \#, L)$
$q_{A'}$	$a$	$(q_R, \#, L)$
$q_B$	$a/b$	$(q_B, a/b, R)$
$q_B$	#	$(q_{B'}, \#, L)$
$q_{B'}$	$a$	$(q_R, \#, L)$
$q_R$	$a/b$	$(q_R, a/b, R)$
$q_R$	$\triangleright$	$(q_0, \triangleright, R)$

$$\begin{aligned}
 (q_0, \triangleright abba) &\longrightarrow (q_0, \triangleright \underline{a} bba) \longrightarrow (q_A, \triangleright \triangleright \underline{b} ba) \longrightarrow \\
 (q_A, \triangleright \triangleright \underline{b} \underline{b} a) &\longrightarrow (q_A, \triangleright \triangleright \underline{b} \underline{b} \underline{a}) \longrightarrow (q_A, \triangleright \triangleright \underline{b} \underline{b} \underline{a} \underline{\#}) \longrightarrow \\
 (q_{A'}, \triangleright \triangleright \underline{b} \underline{b} \underline{a}) &\longrightarrow (q_R, \triangleright \triangleright \underline{b} \underline{b}) \longrightarrow (q_R, \triangleright \triangleright \underline{b} \underline{b}) \longrightarrow (q_R, \triangleright \triangleright \underline{b} \underline{b}) \\
 \longrightarrow (q_0, \triangleright \triangleright \underline{b} \underline{b}) &\longrightarrow (q_B, \triangleright \triangleright \triangleright \underline{b}) \longrightarrow (q_B, \triangleright \triangleright \triangleright \underline{b} \underline{\#}) \longrightarrow \\
 (q_{B'}, \triangleright \triangleright \triangleright \underline{b}) &\longrightarrow (q_R, \triangleright \triangleright \triangleright) \longrightarrow (h, \triangleright \triangleright \triangleright)
 \end{aligned}$$