

# **LA MACCHINA DI TURING**

**Monica Castiglioni - matricola 612164**

## **1) NASCITA DELLA MACCHINA DI TURING**

Negli anni Trenta i matematici svilupparono diverse correnti di pensiero volte ad elaborare una teoria di computazione.

Nel 1936, ancor prima dell'avvento dell'"*Era dei Computer*", un decennio prima della costruzione dell'ENIAC a Philadelphia, **Alan Turing** propose un modello che può essere visto come la base di tutti i calcolatori moderni: la **Macchina di Turing** (MdT o Turing Machine – TM), capace di eseguire ogni tipo di calcolo su numeri e simboli.

Il primo problema che Turing si trovò ad affrontare nella definizione della sua macchina riguardava le *operazioni elementari* che essa doveva saper eseguire.

Con "operazioni elementari" nel linguaggio comune possiamo intendere le più semplici operazioni matematiche come l'addizione e la sottrazione, che si imparano a fare sin dal primo anno di scuola elementare.

Ma come "insegnare" a una macchina queste operazioni?

Era necessario trovare un **algoritmo** di risoluzione.

«"Programmi" per risolvere manualmente problemi numerici sono noti fin dal 1800 a.C., quando i matematici babilonesi del tempo di Hammurabi precisarono le regole per risolvere alcuni tipi di equazioni. Le regole consistevano in procedimenti dettagliati passo dopo passo applicati dettagliatamente a particolari esempi numerici. In particolare, il termine "algoritmo" si rintraccia dall'ultima parte del nome del matematico persiano Abu Ja'far Mohammed ibn Mûsâ *al-Khowârizmî*, il cui testo di aritmetica esercitò una notevole influenza per molti secoli.»<sup>1</sup>

L'algoritmo di risoluzione della somma di due numeri naturali può essere così esplicitato in "pseudolinguaggio":

Somma (input: 2 numeri; output: 1 numero)

begin

risultato = 1° numero;

contatore = 2° numero;

while contatore > 0 do

{risultato = risultato + 1;

contatore = contatore - 1;}

restituisce risultato;

end

---

<sup>1</sup> "La macchina di Turing" - <http://digilander.libero.it/ollecram/turing.htm> - Giugno 2003

Questo procedimento, che appare banale in quanto siamo abituati ad applicarlo ogni giorno da anni, è un *algoritmo*, cioè ha un numero finito di passi. Nell'anno scolastico 2001-2002 ho insegnato in una prima elementare e ho utilizzato questo algoritmo (in un primo momento su un grande "gioco dell'oca", quindi su linee dei numeri) per introdurre il concetto di addizione - non certo immediato né scontato per bambini di sei anni - prima partendo dal "+1", vista anche come strutturazione dell'insieme dei numeri naturali: il metodo d'insegnamento assunto si è basato, infatti, sull'Assioma di Peano per la costruzione dell'insieme  $\mathbb{N}$ .

Una volta trovati algoritmi di risoluzione delle operazioni base della matematica è necessario dimostrarne la correttezza. Infatti non esiste algoritmo che possa verificare la correttezza di un altro algoritmo e questa verifica va eseguita senza l'aiuto di macchine.

Per quanto riguarda il precedente algoritmo di soluzione dell'addizione, è facile verificare che si arriva a un termine, poiché il *ciclo while* è basato su una fondamentale proprietà dei numeri naturali (quella cioè di avere una "base", lo 0) e quindi il contatore deve arrivare in un numero finito di passi a verificare la proprietà di conclusione del ciclo. I numeri di passi eseguiti "in avanti" nella linea dei numeri naturali è esattamente quello del numero da sommare, poiché il contatore è esattamente questo numero e viene decrementato di uno a ogni "passo" fatto.

E' importante specificare che, come per la matematica non è importante quale lingua si parli per fare conti (in Inglese o in Italiano la matematica è sempre la stessa), così «nel caso dei computer l'importante non sono tanto i programmi, che ad esempio possono essere scritti in diversi linguaggi (Fortran, C, Java ecc.), bensì

gli algoritmi che questi rappresentano. Gli algoritmi sono le entità astratte al cuore del funzionamento di ogni computer.»<sup>2</sup>

Occorre quindi precisare cosa serve a una macchina per la computazione di questi algoritmi (così come i bambini hanno carta, penna e, soprattutto, dita).

## **2) STRUTTURA DI UNA MACCHINA DI TURING**

Una Macchina di Turing è definita da un **insieme di regole** che determinano il comportamento della macchina su un **nastro di input-output** (lettura e scrittura). Questo può essere immaginato come un nastro di carta di lunghezza infinita (anche se la porzione di dati è sempre finita), diviso in quadrati dette **celle**. Ogni cella contiene un simbolo oppure è vuota (che si indica con il simbolo "b" blank). Una Macchina di Turing ha una testina che si sposta lungo il nastro leggendo, scrivendo oppure cancellando simboli nelle celle del nastro. La macchina analizza il nastro, una cella per volta, iniziando da quella che contiene il simbolo più a sinistra.

---

<sup>2</sup> "La riforma della scuola italiana" <http://www.dima.unige.it/~bartocci/testi/comp.html> - maggio 2003

La Macchina di Turing può quindi essere vista come un vettore:

$$MdT = \langle \text{Sigma}, \text{Gamma}, Q, F, q_0, b, \text{delta}, \{Dx, Sx\} \rangle$$

Sigma = l'**alfabeto di input**, cioè quello che la Macchina di Turing può leggere (se vengono usati k bit con un codice a lunghezza fissa potranno essere rappresentate  $2^k$  lettere);

Gamma = l'**alfabeto di lavoro**, in quanto in memoria può essere utile avere più simboli. Naturalmente  $\text{Sigma} \subseteq \text{Gamma}$ , poiché con l'alfabeto di lavoro si devono poter scrivere tutti i simboli dell'input.

Q = insieme degli **stati possibili** per la Macchina di Turing

F = sottoinsieme di stati chiamati "**stati accettanti**" o "**finali**". Sono quelli in cui termina la Macchina di Turing se la computazione è andata a buon fine.

$q_0$  = **stato iniziale**: è lo stato da cui parte la computazione della Macchina di Turing.

b = **simbolo di blank**, che indica una cella vuota, mai stata scritta.

delta = **funzione di passaggio** tra gli stati.

$\{Dx, Sx\}$  = possibili **spostamenti** della testina di lettura e scrittura (destra, sinistra).

La computazione della Macchina di Turing avviene attraverso la definizione della funzione  $\delta$  sui vari stati della macchina:

$$\text{delta} : \text{Gamma} \times Q \rightarrow \text{Gamma} \times Q \times \{Dx, Sx\}$$

Cioè, la Macchina di Turing legge una coppia  $(a, q_i)$  e la funzione determina una tripla  $(b, q_j, \text{spostamento})$ .

$$\text{delta} : (a, q_1) \rightarrow (b, q_2, Dx)$$

In base al simbolo letto sul nastro, ci possono essere diverse azioni che la Macchina di Turing può eseguire, a seconda della definizione della funzione. La computazione continua fino a quando ci sono istruzioni, quindi la Macchina di Turing si ferma. L'output è dato dal nastro di input che è stato nel frattempo scritto dalla macchina.

Un esempio di linee di programmazione di Macchina di Turing:

$q_0$ :       $\text{delta}(q_0, 0) = (q_1, 0, Dx)$   
               $\text{delta}(q_0, 1) = (q_1, 1, Sx)$

$q_1$ :       $\text{delta}(q_1, 0) = (q_0, 0, Dx)$   
               $\text{delta}(q_1, 1) = (q_2, 1, Sx)$

Tradotte in pseudolinguaggio:

```
begin

read x;
 $q_0$ :
  if  $x = 0$  {scrivi 0, spostati a destra, passa nello stato  $q_1$ };
  if  $x = 1$  {scrivi 1, spostati a sinistra, passa nello stato  $q_1$ };

read x;
 $q_1$ :
  if  $x = 0$  {scrivi 0, spostati a destra, passa nello stato  $q_0$ };
  if  $x = 1$  {scrivi 1, spostati a sinistra, passa nello stato  $q_2$ };

end
```

Non esiste una sequenza predeterminata di istruzioni da eseguire e le istruzioni sono formate principalmente da salti condizionati (*goto*) e le "etichette"  $q_0, q_1, \dots$ , ecc., sono necessarie proprio per questi passaggi.

Si può avere una "istantanea" della Macchina di Turing con un vettore (tripla):

$$MdT = \langle y, q, k \rangle$$

$y$  = dati sul nastro

$q$  = stato corrente

$k$  = puntatore ai dati

Oppure ponendo la lettera relativa allo stato corrente nell'elenco dei dati:

abcdefghijklm

La Macchina di Turing si ferma quando non è specificata la successiva istruzione. Nonostante gli alfabeti debbano essere finiti, la computazione di una Macchina di Turing può essere infinita ("problema della fermata").

Ecco due esempi di programmi su Macchina di Turing con computazione infinita:

$$q_0: \text{ per ogni } a \quad \delta(q_0, a) = (q_0, 0, Dx)$$

In questo programma, la Macchina di Turing, qualsiasi carattere legge, scrive zero e si sposta a destra (può eseguire operazioni anche se nella cella letta trova solo il simbolo di blank), torna sullo stato iniziale  $q_0$ , utilizzando infinite celle del

nastro a destra (si può pensare che la macchina termini la sua computazione - che comunque sarebbe infinita - quando finisce il nastro, andando in "*running out of memory*").

Un altro esempio di computazione infinita:

$$q_0: \quad \text{delta}(q_0, 0) = (q_1, 0, Dx)$$

$$\text{delta}(q_0, 1) = (q_1, 0, Dx)$$

$$q_1: \quad \text{delta}(q_1, 0) = (q_0, 0, Sx)$$

$$\text{delta}(q_1, 1) = (q_0, 0, Sx)$$

In questo programma, la Macchina di Turing, che legga 1 o 0, scrive zero e si sposta a destra, quindi scrive di nuovo zero e si sposta a sinistra in una sequenza infinita. Si noti che gli stati  $q_0$  e  $q_1$  che trovano un 1 servono eventualmente solo al primo ciclo, in quanto ai successivi la macchina leggerà sempre e solo gli zeri che ha appena scritto.

Per convenzione, la testina della Macchina di Turing non può rimanere ferma, poiché una sequenza di azioni svolte a testina ferma può essere sostituita dall'ultima azione svolta senza che l'output cambi.

In generale l'elenco delle definizioni della funzione  $\delta$  è mantenuto in memoria sotto forma di una tabella, che viene "consultata" durante l'elaborazione:

stato	input	output
$q_0$	0	$q_1, 0, Dx$
$q_0$	1	$q_1, 1, Sx$
$q_1$	0	$q_0, 0, Dx$
$q_1$	1	$q_0, 1, Sx$

### **3) CHE COSA PUÒ E NON PUÒ CALCOLARE UNA MACCHINA DI TURING**

Secondo la tesi di Church, una Macchina di Turing può eseguire qualsiasi programma in funzione oggi sui calcolatori (ogni calcolatore moderno può essere visto come una Macchina di Turing): «Le macchine di Turing sono in grado di risolvere tutti i problemi algoritmici effettivamente risolvibili.»<sup>3</sup>

Problemi insolubili su computer moderni, lo sono anche sulla Macchina di Turing e viceversa. Un esempio è il problema di decisione se la computazione di un dato programma terminerà o no su una Macchina di Turing: il "problema della fermata". («La computazione non è computabile»<sup>4</sup>).

---

<sup>3</sup> "La riforma della scuola italiana" <http://www.dima.unige.it/~bartocci/testi/comp.html> - maggio 2003

<sup>4</sup> Ibidem.

I problemi risolubili da una Macchina di Turing sono tutti quelli che possono essere risolti da qualsiasi altro calcolatore. Si dividono principalmente in due gruppi:

1) **problemi di decisione**: richiedono una risposta "sì/no" e consistono spesso nel cercare se una soluzione a un dato problema "*esiste o non esiste*". Un esempio è quello della soddisfacibilità di formule logiche espresse in forma normale congiunta;

2) **problemi di ricerca**: richiedono di *trovare una soluzione*. Tra questi, il precedente semplice algoritmo di addizione.

Questi problemi possono essere risolti da qualsiasi Macchina di Turing.

#### **4) UN PROGRAMMA PER MACCHINA DI TURING: LA DIVISIONE PER DUE**

In questo programma la Macchina di Turing legge in ingresso una sequenza di 1 che possiamo interpretare come un numero naturale: per cui 11111 sarà 5, 1111111 7 e così via. Alla fine della sequenza si troveranno 0 o blank.

Una Macchina di Turing che risolve la divisione per due deve fare i seguenti passi:

1) legge la sequenza di ingresso, partendo dall'1 più a sinistra;

2) se la sequenza di ingresso contiene un solo 1, lo cancella e si ferma;

3) se la sequenza di ingresso contiene almeno due 1, li cancella e va ad aggiungere un 1 alla sequenza "di uscita". Quindi torna sulla cella più a sinistra di ciò che rimane della sequenza di ingresso e riparte da 1).

stato	input	stato successivo	output	spostamento
q <sub>0</sub>	1	1	0	Dx
q <sub>1</sub>	1	2	0	Dx
q <sub>1</sub>	0 / b	fine	-	-
q <sub>2</sub>	1	2	1	Dx
q <sub>2</sub>	0 / b	3	0	Dx
q <sub>3</sub>	0 / b	4	1	Sx
q <sub>3</sub>	1	3	1	Dx
q <sub>4</sub>	1	4	1	Sx
q <sub>4</sub>	0 / b	5	0	Sx
q <sub>5</sub>	1	5	1	Sx
q <sub>5</sub>	0 / b	0	0	Dx
q <sub>6</sub>	0 / b	fine	-	-

## 5) IL WEB COME UNA MACCHINA DI TURING

Molti oggetti che usiamo quotidianamente possono essere visti come una Macchina di Turing: una calcolatrice, un cellulare, un registratore, un televisore...

Anche Internet può essere visto come una Macchina di Turing?

«Sì e no. La maggior parte del web è statico -- le pagine non rispondono agli input. Ma altre parti della rete sono dinamiche e queste parti coincidono con il modello di Turing, e perciò è come qualsiasi altro computer.

In quest'area, è un miscuglio di mondi di Turing incompatibili, debolmente supportati e per lo più isolati e nonostante siano macchine molto potenti non possono lavorare assieme.

L'obiettivo è quello di creare un modo standard di comunicare per le Macchine di Turing, con Internet che fa da connessione.

Ciò include Java e ogni altro linguaggio di programmazione che lavora su qualsiasi sistema operativo che può connettersi in rete, il che vuol dire, ovunque al giorno d'oggi ci sia vita.»<sup>5</sup>

---

<sup>5</sup> "The web as a Turing Machine" - <http://static.userland.com/userLandDiscussArchive/msg010950.html> - giugno 2003



## BIBLIOGRAFIA

- 📖 Corso di "Algoritmi e Strutture Dati II" - prof.ssa N.Sabadini - Università dell'Insubria, Sede di Como - a.a. 2002-2003
- 📖 "Progetto e Analisi di Algoritmi" - Proff. A.Bertoni e M.Goldwurm - Università degli Studi di Milano - Ottobre 2002
- 📖 "Dal +1 alla scoperta dell'addizione: il percorso di strutturazione del concetto di quantità nell'acquisizione dell'operatività aritmetica." - M.Castiglioni, Scuola Elementare di Prestino-Como - giugno 2002
- 🖥️ "La macchina di Turing" - <http://digilander.libero.it/ollecram/turing.htm> - Giugno 2003
- 🖥️ "La Macchina di Turing" - <http://digilander.libero.it/ollecram/turing.htm> - Aprile 2003
- 🖥️ "La riforma della scuola italiana"  
<http://www.dima.unige.it/~bartocci/testi/comp.html> - maggio 2003
- 🖥️ "La tesi di Church" - <http://www.vialattea.net/esperti/inform/church.htm>
- 🖥️ "The web as a Turing Machine" -  
<http://static.userland.com/userLandDiscussArchive/msg010950.html> -  
Giugno 2003
- 🖥️ "Turing Machine" - <http://www.ams.org/new-in-math/cover/turing.html>  
- Giugno 2003
- 🖥️ "What's a Turing Machine?" - <http://www.igs.net/~tril/tm/basics.html> -  
Giugno 2003