

La macchina di Turing

Presentiamo una semplice *macchina astratta* capace di elaborare informazione in formato digitale sulla base di un programma composto da regole: la cosiddetta *macchina di Turing (MdT)*. La MdT prende il nome dal logico e matematico inglese Alan Turing ed è solo una costruzione concettuale. Ma si tratta di una costruzione concettuale di grande fascino: semplicissima (come vedremo, gli 'ingredienti di base' di una MdT sono pochi e di facile comprensione) ed estremamente potente. La definizione di macchina formale di Turing è uno degli strumenti più affascinanti tra quelli che hanno posto le basi per l'evoluzione dell'informatica moderna e dei concetti legati alla struttura di un programma e degli algoritmi ad essi collegati. Questa macchina puramente teorica è nata come sussidio allo studio degli algoritmi e della possibilità di risolvere alcuni particolari problemi di elaborazione. In altre parole si tratta di un "*banco di prova*" teorico per verificare se un problema è risolvibile mediante computer.

L'unità esterna di memorizzazione della MdT è un nastro di lunghezza potenzialmente infinita (la premessa di Turing era che la macchina fosse *concettualmente* semplice, non che fosse anche facile da realizzare!), esso è in grado di memorizzare tutti i dati relativi ad ogni particolare elaborazione indipendentemente dalla loro quantità. Il nastro è sezionabile in singole celle e in ciascuna di esse può essere scritto un simbolo. L'alfabeto dei simboli che possono essere scritti nelle caselle del nastro sarà finito; per semplificare, visto che è sempre possibile 'tradurre' in formato binario alfabeti finiti più complessi (come le dieci cifre della notazione decimale o le lettere che usiamo per scrivere), assumiamo che su ogni casella del nastro si possano scrivere solo i simboli '0' o '1'.

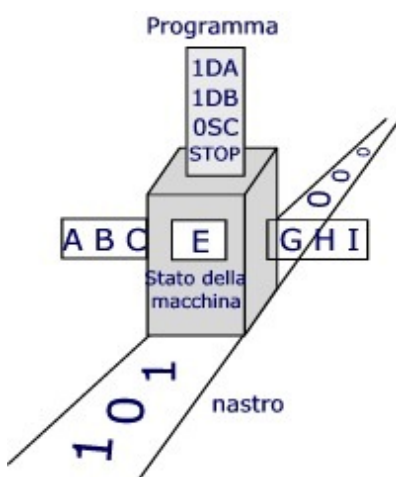
Oltre al nastro con le sue cellette, la nostra MdT comprenderà una testina di lettura/scrittura (pensiamo a qualcosa di simile alla testina di un registratore) che sia in grado di *leggere* il simbolo che si trova nella celletta sopra la quale è posizionata, e di *scrivere* su tale celletta in modo da potere - volendo - modificare il simbolo che ha letto. Il dispositivo di lettura e scrittura, al pari del supporto di memorizzazione, deve essere in grado di trasferire i simboli desiderati sul nastro stesso e, al contempo, deve possedere la capacità di deciderne la direzione di movimento mediante una unità di controllo. Quest'ultima inoltre contiene naturalmente il programma da eseguire. Infatti la MdT potrà assumere una serie finita di *stati* distinti ed è attraverso la distinzione di questi stati che potremo specificare le istruzioni del *programma* da fare eseguire alla macchina.

Questo *programma* farà sì che la testina si sposti sul nastro facendo cambiare stato alla macchina stessa in dipendenza dallo stato in cui la macchina si trovava in precedenza e dal simbolo sul quale la testina si trova posizionata. Ad esempio, potremmo stabilire che, se la macchina si trova nello stato *a* e legge un 1 nella casella su cui è posizionata la testina, la testina si dovrà spostare a destra di una casella e la macchina dovrà passare nello stato *b* (a questo punto, dovremo ulteriormente specificare le istruzioni che la macchina

deve eseguire quando si trova nello stato *b*, in dipendenza da quello che la testina legge sul nastro, e così via). Insomma:

La particolare successione di '0' e di '1' scritta sul nastro prima dell'avvio della macchina rappresenta l'*input* della macchina - i dati sui quali vogliamo farla lavorare. La particolare successione di '0' e di '1' scritta sul nastro nel momento in cui la macchina si ferma (se si ferma - come vedremo potrebbe anche non farlo) rappresenta invece l'*output* della macchina, il risultato del suo processo di elaborazione.

La MdT risulta sostanzialmente "*dedicata*" ad ogni unica applicazione che si intenda simulare: infatti essa non è dotata di un dispositivo di memoria di massa sul quale far risiedere o poter leggere il programma stesso.



La logica di controllo della macchina è composta da *frasi* o *stringhe* composte da cinque campi o istruzioni (dette "*quintette*"). L'esecuzione di ogni *quintetta* è vincolata sostanzialmente sia dal simbolo presente al momento nell'unità di lettura e scrittura che dallo *stato* della macchina. Lo stato della macchina è una condizione essenzialmente arbitraria. In linea di principio potremo definire uno stato arbitrario **Si** di avvio, nel quale la macchina inizia l'esecuzione della procedura di calcolo, fino a raggiungere la speciale condizione **H** (*Halt*) che corrisponde al termine dell'esecuzione. Tra questi due stati possono naturalmente esserci **n** stati differenti che riflettono appunto il risultato dell'elaborazione in corso e stabiliscono quale *quintetta* eseguire di conseguenza.

Ogni quintetta è composta dai seguenti cinque elementi:

1. L'attuale stato della macchina
2. Il simbolo contenuto nella cella correntemente in corso di lettura
3. Il simbolo da scrivere nella medesima cella se non avviene alcuna modifica del dato
4. Lo stato della macchina a seguito delle precedenti operazioni (**2** e **3**)
5. La direzione di scorrimento del nastro (**avanti** o **indietro**)

Ad esempio, la quintetta (**S1**, **7**, **9**, **S2**, **R**) viene eseguita ogni qualvolta la macchina si trova nello stato **S1** e la testina di lettura legge un **7**. A quel punto, il **7** viene sostituito da un **9**, la macchina viene posta nello stato **S2** ed il nastro viene fatto scorrere di una posizione verso destra (**R** = *right* = destra).

Con queste premesse, la progettazione di una MdT per la soluzione di un determinato problema significa definire il formato dei dati immessi mediante il sistema di lettura, quello dei dati prodotti durante l'elaborazione e che poi, di conseguenza, verranno riprodotti sul nastro magnetico quando ad elaborazione terminata si raggiungerà lo stato **H**, e, infine, il numero di *quintette* necessarie per l'implementazione dell'intero algoritmo.

Sembra complicato? Per capire meglio, fate un po' di esperimenti con un [programma di simulazione online](#) (vedi immagine sotto)



Una MdT è in grado di effettuare moltissime operazioni anche molto complesse: in effetti, è in grado di effettuare tutte le operazioni che può compiere il più potente dei computer a noi noti, ovvero, per parlare in forma più tecnica, è in grado di calcolare il valore di qualsiasi funzione per la quale disponiamo di procedure effettive di computazione. Le MdT sono strumenti potenti: non si limitano a trovare la soluzione di semplici problemi di aritmetica, come l'addizione, ma permettono di computare un ambito estremamente ampio di funzioni matematiche: in effetti, per quanto ne sappiamo, *tutte* le funzioni computabili attraverso un metodo definito e un numero finito (anche se magari enormemente grande) di passi. Naturalmente, le MdT capaci di computare funzioni particolarmente complicate lavoreranno a loro volta su lunghi insiemi di regole, e arriveranno alla soluzione svolgendo un gran numero di passi.

Ma veramente le MdT sono in grado di computare tutte le funzioni computabili? E' difficile dare con sicurezza una risposta affermativa a questo interrogativo: può ben darsi che un giorno qualcuno scopra dei metodi effettivi di computazione in grado di computare funzioni che una normale MdT non è in grado di affrontare. Tuttavia, per ora non abbiamo trovato metodi simili: fino ad ora ogni funzione computabile attraverso metodi che da un punto di vista intuitivo possiamo considerare effettivi risulta anche computabile da una opportuna MdT. Questa situazione è rispecchiata da una formulazione della cosiddetta *tesi di Church* (dal nome del matematico e logico Alonzo Church),

secondo la quale l'insieme di funzioni computabili utilizzando una MdT coincide con l'insieme delle funzioni effettivamente computabili.

Non sappiamo se la tesi di Church sia corretta (e probabilmente non lo sapremo mai – anche se un giorno potremmo accorgerci che è sbagliata). In compenso, possiamo dimostrare che esistono alcune funzioni che una MdT *non* è in grado di computare. Tuttavia, si tratta di funzioni che non sappiamo computare neanche in altri modi. Non c'è niente di scandaloso nel fatto che alcune funzioni non siano computabili. Un caso di funzione non computabile è la funzione corrispondente al *problema della fermata*.

Per capire di che si tratta, cominciamo dal ricordare che vi sono *funzioni parziali*, funzioni, cioè, tali che per alcuni argomenti non esiste (non è definito) alcun valore. Così, ad esempio, la funzione "l'attuale presidente della Repubblica del paese 'x'" ha un valore se sostituiamo a 'x' uno stato repubblicano (come l'Italia), ma non ha un valore se sostituiamo a 'x' una monarchia (come l'Inghilterra). Quando abbiamo una MdT che riceve come input valori per i quali la corrispondente funzione non è definita, la MdT non si ferma. Il vero guaio è che, messi davanti a una macchina di Turing (cioè a una particolare configurazione iniziale del nastro e all'insieme di regole che determinano il funzionamento di quella particolare macchina), possiamo non essere in grado di dire se quella macchina si fermerà o no. E non solo non sappiamo farlo noi, ma non lo saprebbe fare neppure una ipotetica MdT 'di secondo livello' alla quale fornissimo come input una rappresentazione adeguata dello stato iniziale e delle regole di funzionamento della MdT 'di primo livello' che stiamo esaminando. In sostanza, il *problema della fermata* (trovare una procedura effettiva che permetta di stabilire, per ogni MdT data, se quella MdT si fermerà o no) non è risolvibile, o quantomeno non lo è attraverso una MdT. In altre parole, una corrispondente funzione che abbia un certo valore (per esempio 1) se la macchina considerata si ferma per l'input in questione e un altro valore (per esempio 0) se invece non si ferma, non è una funzione Turing-computabile.

Concludendo, ricordiamo che, anche se abbiamo detto che la MdT lavora solamente con due simboli, non è affatto detto che si debba limitare a fare operazioni numeriche: è infatti possibile convertire in formato binario anche informazioni testuali, o grafiche, o sonore. Così, ad esempio, possiamo costruire una MdT che riceva come input una tavola delle calorie e il menù di un pasto e restituisca come output il totale delle calorie previste dal pasto, o una MdT che conti le occorrenze delle parole 'cuore' e 'amore' in certe canzoni (che dovranno evidentemente esserle date, anch'esse codificate, come input), o ancora - questo è abbastanza facile - una MdT che trasformi tutte le lettere maiuscole di un testo in minuscole e viceversa. Insomma: dobbiamo sempre ricordare che una computazione è un processo di manipolazione di simboli governato da regole: non è affatto detto che i simboli - a partire dall'"1" e dallo "0" della codifica binaria - debbano rappresentare necessariamente numeri, e non è affatto detto che le regole utilizzate debbano essere sempre regole matematiche.