Modulo 3
ALGORITMI E DECIDIBILITÀ
(CON APPROFONDIMENTO MACCHINE DI TURING)

- **Algoritmi**
- **Problemi di decidibilità**
- **Macchine di Turing**
- **Macchina di Turing universale**
- **Tesi di Turing-Church**
- **Macchine di Turing e problemi di decidibilità**

Bibliografia:

- R. Penrose, *La mente nuova dell'imperatore*, Rizzoli, Milano, 1992 (cap. 2).
- B.J. Copeland, 'The Church-Turing thesis', *Stanford Encyclopedia of Philosophy*
  (disponibile in rete all'URL:
  http://plato.stanford.edu/entries/church-turing/).
- A. Turing, 'Computing machinery and intelligence', *Mind*, 1950, pp. 433-460; trad. it. in V. Somenzi, R. Cordeschi, *La filosofia degli automi*, Boringhieri, Torino, 1986, capitolo 7.

♦ ALGORITMI

Esempio:

Qual è il **massimo comune divisore** (**MCD**) di 3654 e 1365?

MCD(3654, 1365)=21

Problema: Come si fa a verificare che 21 è **effettivamente** il massimo comune divisore di 3654 e1365?

Occorre un **algoritmo**, cioè una **procedura sistematica**, **meccanica**, **effettiva** per verificarlo.

Algoritmo di Euclide

3654	:	1365	dà resto 924
1365	:	924	dà resto 441
924	:	441	dà resto 42
441	:	42	dà resto 21
42	:	21	dà resto 0

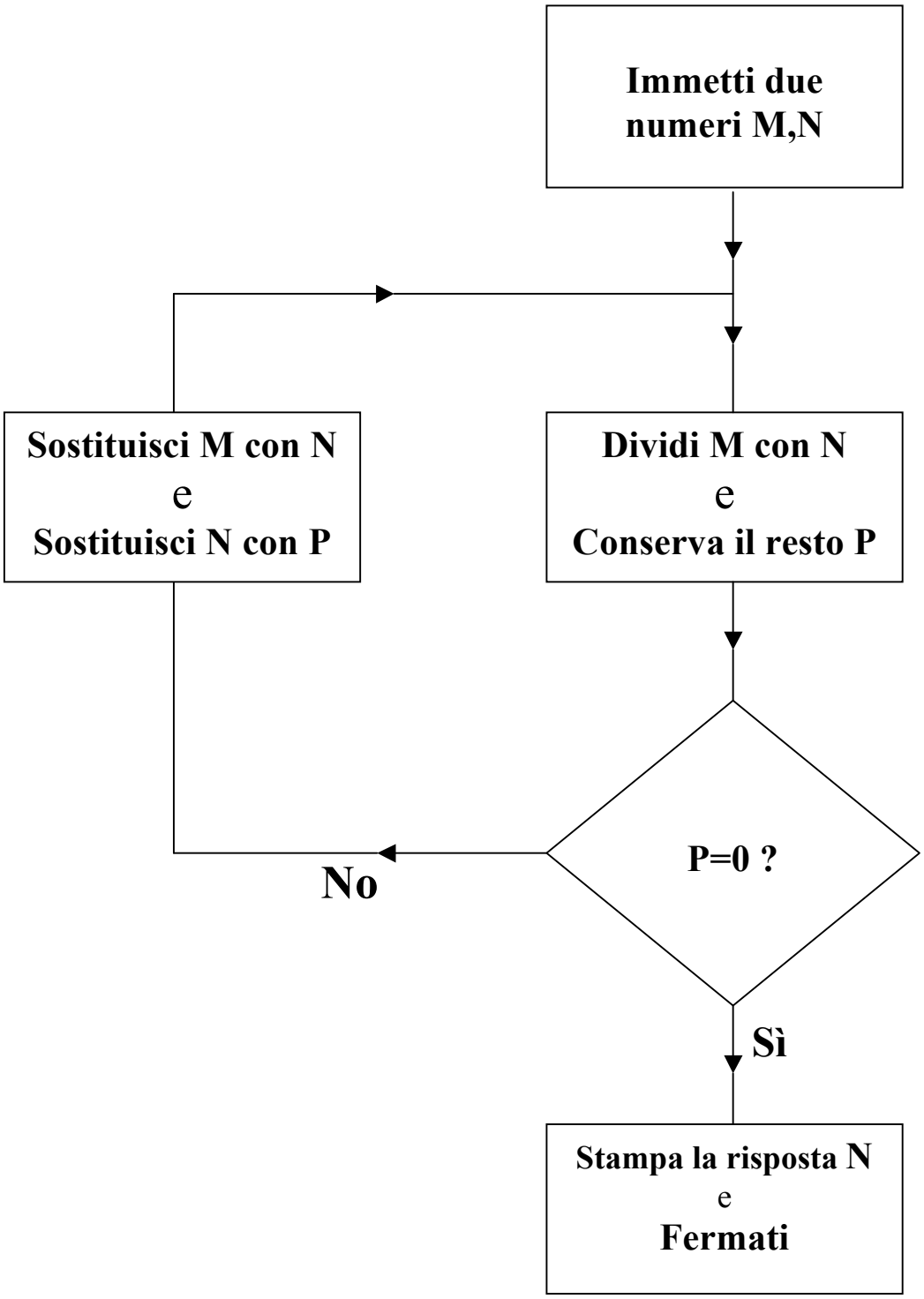
⇓

MCD(3654, 1365)=21

- L’applicazione della procedura può richiedere un tempo lunghissimo, ma in ogni caso **terminerà** e si otterrà una risposta **in un numero finito di passi**.

- Questa procedura può essere applicata a un’infinità di coppie di numeri. Comunque, può essere descritta in **termini finiti**, per esempio con un **diagramma di flusso.**

DIAGRAMMA DI FLUSSO (PROVVISORIO) PER L'ALGORITMO DI EUCLIDE



Nel diagramma di flusso dell'algoritmo di Euclide abbiamo assunto di conoscere le operazioni per ottenere il **resto**. Anche queste operazioni possono essere descritte algoritmicamente.

Sia M=7 •••••••
Sia N=3 •••

•••••••
•••
• **(Resto di 7:3)**

Sottrarre 3 da 7 tante volte fino a quando la differenza risulta strettamente minore di 3.

1. ••••••• (7-3) [1ª applicazione della sottrazione]

2. 7-3>3, quindi procedi sottraendo 3 al risultato di 1.

3. •••• (4-3) [2ª applicazione della sottrazione]

4. 4-3 $\not>$ 3, quindi fermati e rispondi:
"il resto di 7:3 è 4-3".

DIAGRAMMA DI FLUSSO PER L'ALGORITMO "RESTO"

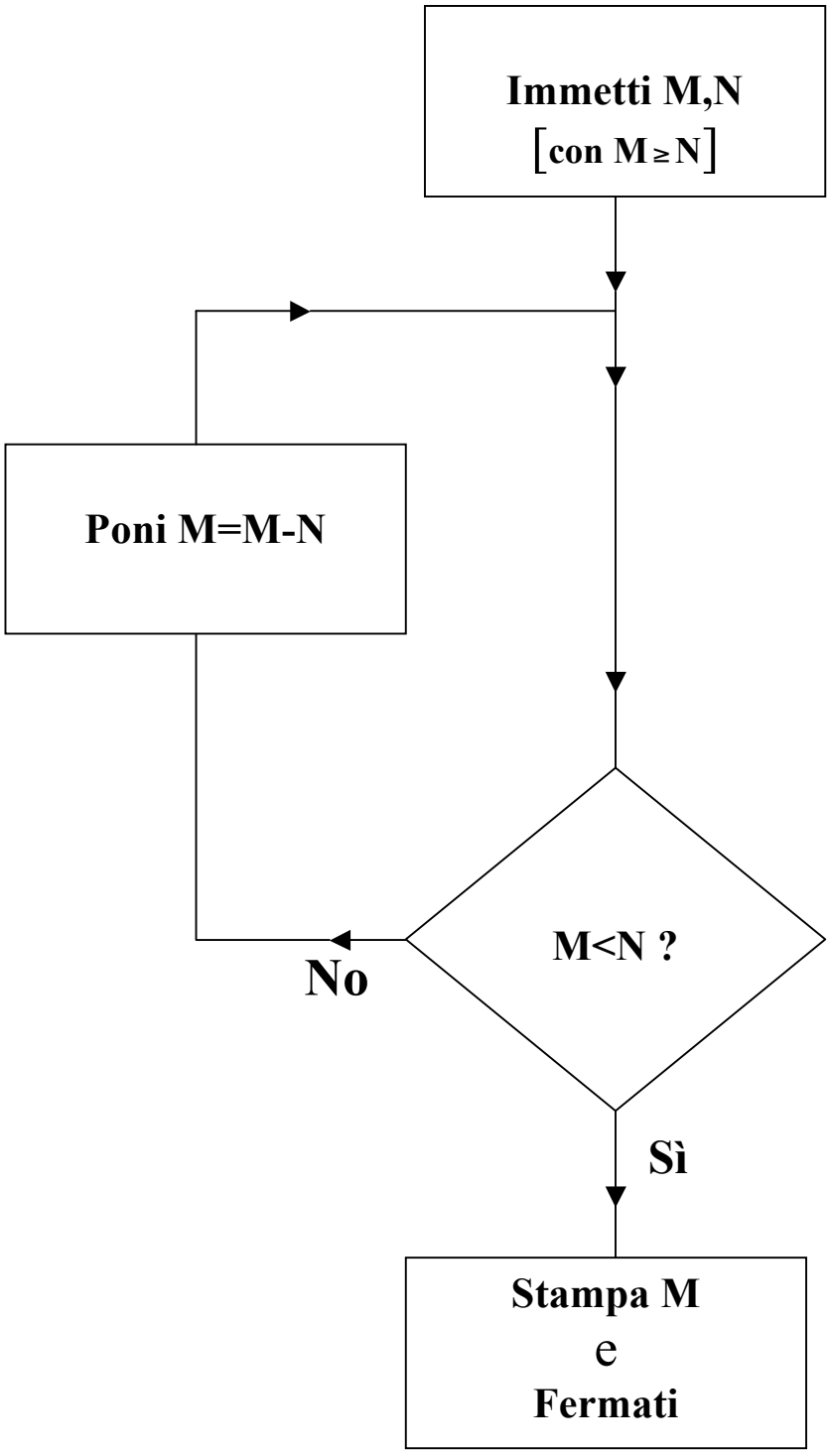
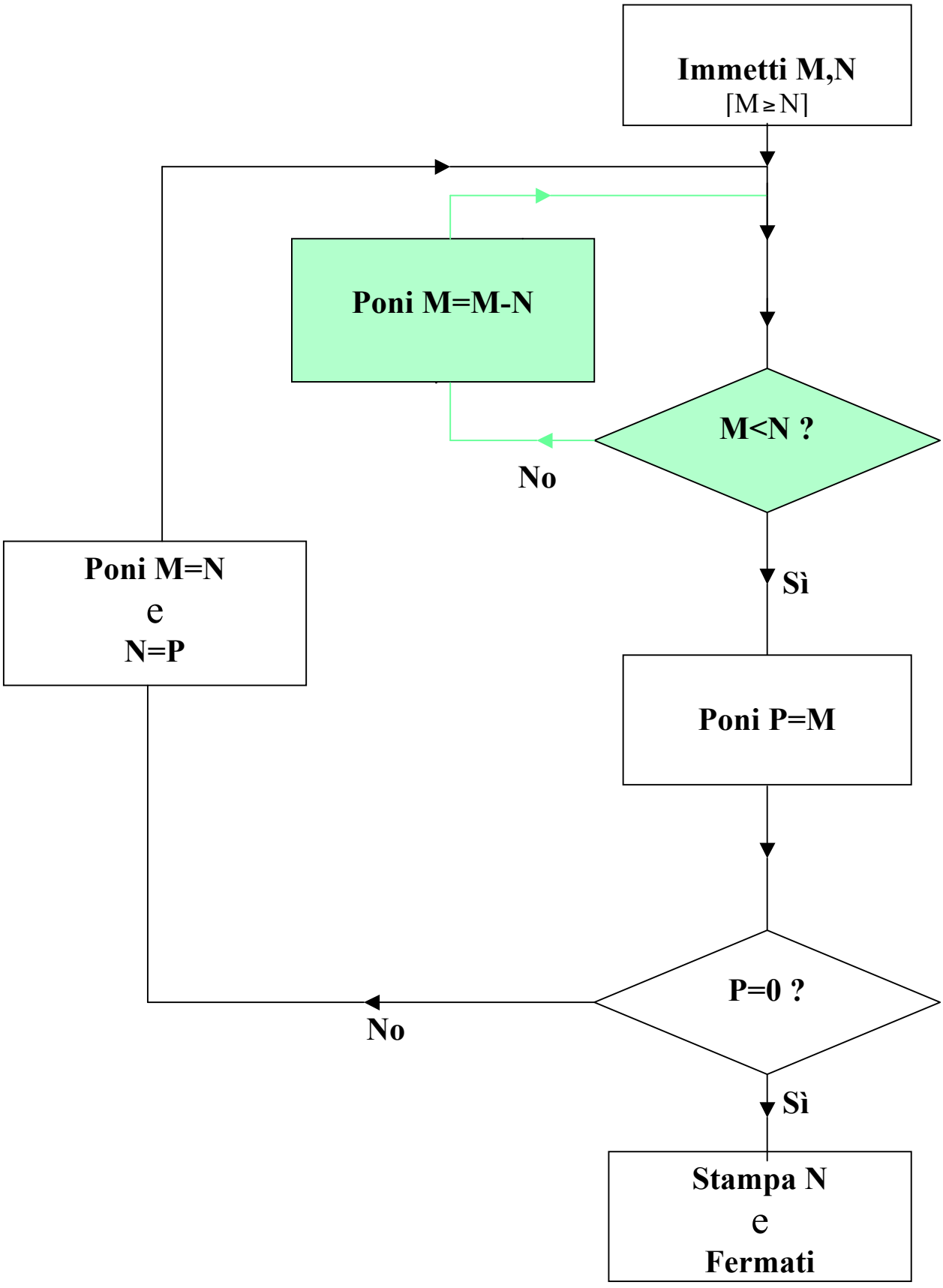


DIAGRAMMA DI FLUSSO PER L'ALGORITMO DI EUCLIDE



PROGRAMMA *BASIC* PER ALGORITMO RESTO

```
1    PRINT "Immetti due numeri M,N, con M>=N":
     INPUT M,N
2    IF M<N THEN 4
3    LET M=M-N: GOTO 2
4    PRINT M
```

PROGRAMMA *BASIC* PER ALGORITMO EUCLIDE

```
1    PRINT "Immetti due numeri M,N, con M>=N":
     INPUT M,N
2    IF M<N THEN 4                    } SUBROUTINE
3    LET M=M-N: GOTO 2                }
4    LET P=M
5    IF P=0 THEN 7
6    LET M=N: LET N=P: GOTO 2
7    PRINT N
```

♦ PROBLEMI DI DECIDIBILITÀ

Quando un metodo o una procedura si dicono effettivi o meccanici?

Definizione (*intuitiva*)
Un metodo o una procedura *P* (per raggiungere un certo risultato) si dicono **effettivi** o **meccanici** sse

1. *P* è formulato nei termini di un numero finito di istruzioni esatte, dove ogni istruzione è espressa per mezzo di un numero finito di simboli;
2. *P* produrrà sempre il risultato desiderato in un numero finito di passi, se eseguito correttamente;
3. *P* può essere eseguito (in pratica o in principio) da un essere umano, non aiutato da una macchina, con carta e matita;
4. *P* non richiede, per essere eseguito da un essere umano, nessuna intuizione o ingegno.

Algoritmo := metodo effettivo

Esempio di procedure effettiva:
Test, con le tavole di verità, per stabilire se una proposizione del calcolo proposizionale classico è o non è una verità logica (tautologia).

Le asserzioni che riguardano l'esistenza di un metodo effettivo per raggiungere un certo risultato sono di solito espresse dicendo che c'è un metodo effettivo per ottenere i valori di una determinata funzione.
Per esempio: l'asserzione che c'è un metodo effettivo per determinare se una data proposizione del calcolo proposizionale classico è o non è una verità logica (cioè il metodo delle tavole di verità), è espressa dicendo che c'è un metodo effettivo per ottenere i valori di una funzione, diciamo *t*, il cui dominio è l'insieme delle proposizioni del calcolo proposizionale classico e il cui valore per una data proposizione α, scritto $t(\alpha)$ è 1, se α è una verità logica; è 0, se α non lo è.

$$t(\alpha) = \begin{cases} 1, \text{ se } \alpha \text{ è una tautologia;} \\ 0, \text{ se } \alpha \text{ non è una tautologia} \end{cases}$$

Problema: Esistono "problemi matematici non computazionali", per la soluzione dei quali cioè non esiste nessuna **procedura algoritmica generale**?

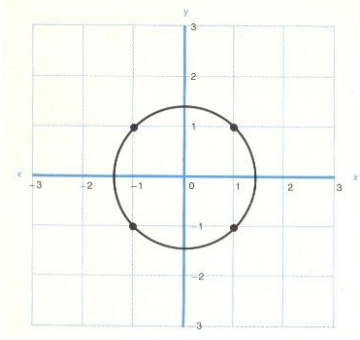
D. Hilbert:
(Parigi 1900, Bologna 1928) [*Entscheidungsproblem*]

[**10° Problema di Hilbert**]: Descrivere una procedura meccanica grazie alla quale sia possibile decidere se una *equazione diofantea* ammetta o non ammetta **soluzioni intere**.

Definizione Un'*equazione diofantea* è un'equazione polinomiale i cui **coefficienti** sono numeri **interi** (positivi o negativi) e gli **esponenti** sono numeri **interi positivi**.

Esempi di equazioni diofantee:

1. $x^2 + y^2 - 2 = 0$
2. $x^2 + y^2 - 3 = 0$
3. $x^2 - 3xy = 5$
4. $x^n + y^n - z^n = 0$
   se $n=2$ Teorema di Pitagora
   se $n>2$ Equazione di **Fermat**

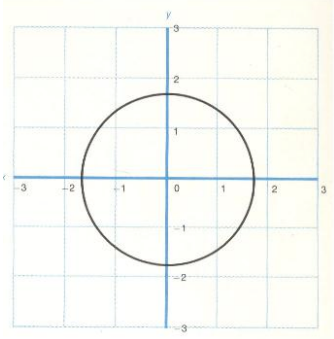


$x^2 + y^2 - 2 = 0$

Sol(Int)=$\{(1,1), (1,-1),(-1,1),(-1,-1)\} \subset$ Sol

$(\sqrt{2},0) \in$Sol, ma $(\sqrt{2},0) \notin$Sol(Int).

Quindi: questa equazione diofantea ammette soluzioni intere



$x^2 + y^2 - 3 = 0$

Sol(Int)=$\varnothing$

Sol$\neq \varnothing$; per esempio $(\sqrt{3},0) \in$Sol

Quindi: questa equazione diofantea non ammette soluzioni intere

Nel 1970 Matyasevich fornì una risposta negativa al 10^o problema di Hilbert.

La nozione di metodo effettivo che abbiamo dato è **informale!**

Il lavoro di Matyasevich però presuppone una definizione precisa di procedura effettiva, meccanica.

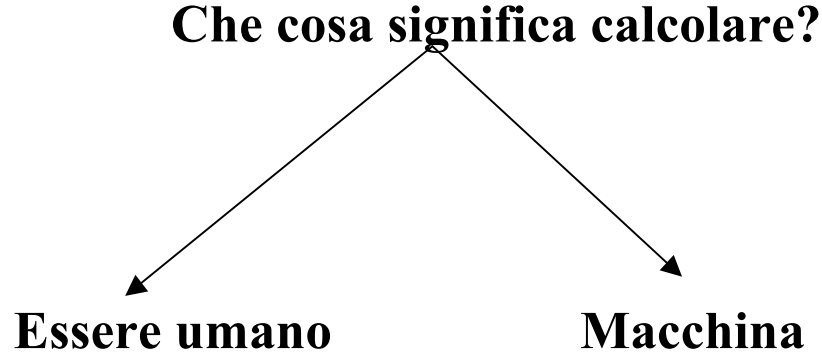
Matyasevich si è avvalso anche dei lavori di Church e soprattutto di **Alan Turing** (1912-1954).

Uno degli scopi del lavoro di **Turing** [1936]
(A. Turing, "On computable numbers, with an application to the Entscheidungsproblem", *Proceedings of the London Mathematical Society*, **42**, pp. 230-265) fu quello di

dare una definizione **formale** della nozione di metodo effettivo

♦ Macchine di Turing

Turing cercò di fornire una risposta *matematica* al problema



Per calcolare ci si serve:

1) di un certo numero finito di simboli che possono essere sistemati in un certo modo, per esempio, su un foglio di carta. Sul foglio di carta si può **scrivere** o **cancellare** i simboli dell'alfabeto un **numero finito di volte**.

Si può inoltre:

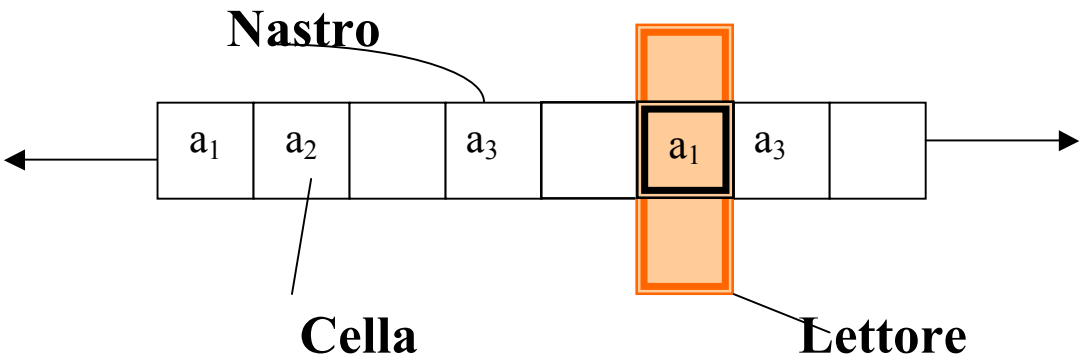
2) modificare il proprio campo visivo un **numero finito di volte**;
3) ricordando un **numero finito di istruzioni** e di atti già compiuti, si passa da una combinazione iniziale di simboli (che rappresentano gli argomenti iniziali della funzione da calcolare) a una combinazione finale di simboli (che rappresenta il risultato della funzione).

Intuitivamente: Una **MACCHINA DI TURING** (MT) è un modello meccanico dei processi (1)-(3); questo modello è ottenuto mediante opportune astrazioni, semplificazioni e idealizzazioni che però si possono considerare inessenziali.

Una MT è una macchina matematicamente idealizzata; è un **oggetto matematico**, non fisico.

Intuitivamente: Una MT dispone di:

1) **alfabeto finito**, cioè un insieme finito di simboli;
2) **nastro** potenzialmente infinito nei due sensi, diviso in **celle**; ciascuna cella può contenere al massimo un simbolo;
3) **lettore** ("occhio") capace di osservare solo una cella alla volta;
4) **memoria** capace solo di un certo numero finito di **stati**, detti **stati interni**.

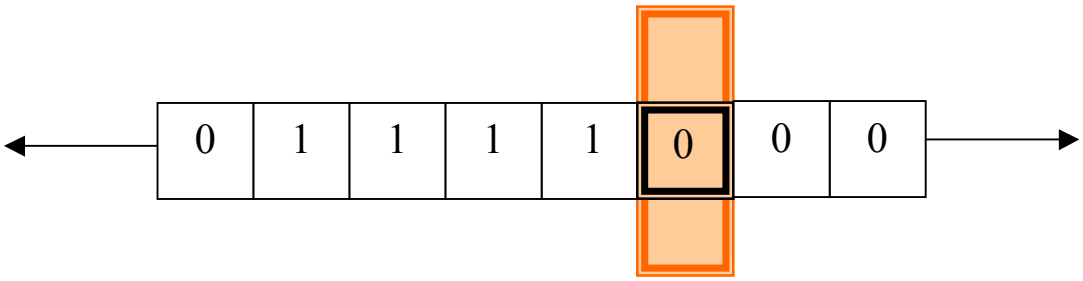


Si assume che la macchina funzioni per **passi elementari successivi**. Più precisamente, in un passo, la macchina può:

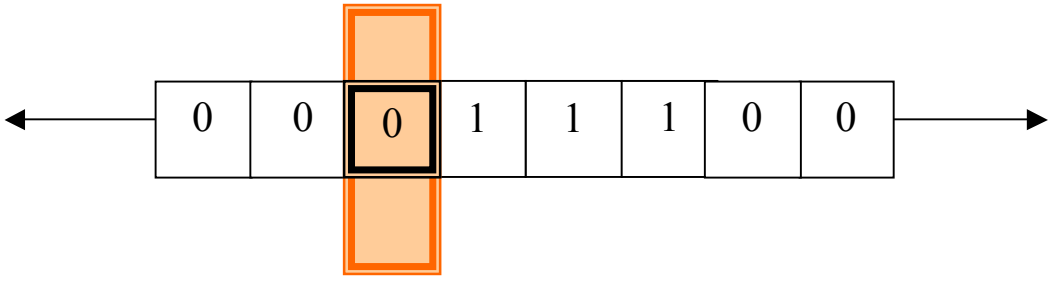
a) stampare o cancellare un solo simbolo, e passare eventualmente da un certo stato interno a un altro stato interno;
b) spostarsi di un passo a sinistra o a destra, e passare eventualmente da un certo stato interno a un altro stato interno;
c) fermarsi.

Ulteriori semplificazioni e convenzioni:

a) L'alfabeto contiene due soli simboli: 0 e 1;
b) Il lettore rimane fermo, mentre il nastro si muove;
c) Una cella si considera vuota quando contiene il simbolo 0;
d) Il nastro contiene sempre un numero finito di simboli 1;
e) **Prima dell'avvio**, il lettore della macchina si trova sempre su una cella vuota che alla sua destra ha solo celle vuote, e immediatamente alla sua sinistra c'è una cella che contiene l'"ultimo" simbolo 1 dell'**input**.



d) Alla fine (supposto che la macchina si fermi!), il lettore si trova sulla cella vuota immediatamente a sinistra del primo simbolo 1 dell'**output**.



Caratteristiche di una MT:

a) Il numero degli stati (interni) è finito;
b) il suo comportamento è completamente determinato dal suo stato interno e dall'input. Dato il suo **stato iniziale** e l'**input**, la macchina opera in modo completamente **deterministico.**

Possiamo descrivere il comportamento di una MT attraverso una **tavola** (**di transizione degli stati**). Identificheremo la TM proprio con la tavola che descrive il suo comportamento.

La tavola è costituita da **5 colonne**.

- Nella **prima colonna** (indicata con **S#**) indicheremo lo stato "attuale" della macchina;
- Nella **seconda colonna** (indicata con **R**) indicheremo ciò che la macchina **legge** (cioè 1 o 0);
- Nella **terza colonna** (indicata con **W**) indicheremo ciò che la macchina **scrive**;
- Nella **quarta colonna** (indicata con **M**) indicheremo il movimento che la macchina compie (cioè L, se si sposta a sinistra (di una cella); R, se si sposta a destra; H se si ferma);
- Nella **quinta colonna** (indicata con **N#**) lo stato in cui passa la macchina.

Convenzione: Indicheremo gli stati con numeri naturali > 0.

Quindi, una riga della tavola di una MT avrà, per esempio, la forma seguente:

S#	R	W	M	N#
2	1	0	R	3

Si legge:
Se ti trovi nello stato 2 e leggi il simbolo 1, allora stampa il simbolo 0 (cioè cancella), vai a destra [n.b.: è il nastro che si muove] e portati nello stato 3.

Esempio 1: **MT0**
Macchina che, qualsiasi cosa osservi sulla cella in cui è posizionata, cancella quello che trova, cioè stampa (o ristampa se c'è già) 0 e si ferma.

S#	R	W	M	N#
1	0	0	H	1
1	1	0	H	1

Esempio 2: **MT1**
Macchina che, qualsiasi cosa osservi sulla cella in cui è posizionata, stampa 1(o ristampa se c'è già) e si ferma.

S#	R	W	M	N#
1	0	1	H	1
1	1	1	H	1

Esempio 3: **MTR**
Macchina che, qualsiasi cosa osservi sulla cella in cui è posizionata, si sposta a destra di una cella e si ferma

S#	R	W	M	N#
1	0	0	R	2
1	1	1	R	2
2	0	0	H	2
2	1	1	H	2

Esempio 4: **MTL**

Macchina che, qualsiasi cosa osservi sulla cella in cui è posizionata, (il nastro) si sposta a sinistra di una cella e si ferma.

S#	R	W	M	N#
1	0	0	L	2
1	1	1	L	2
2	0	0	H	2
2	1	1	H	2

Esempio 5: **MTR0**

Macchina che si ferma sulla prima cella vuota a sinistra rispetto a quella dalla quale è partita.

S#	R	W	M	N#
1	0	0	R	2
1	1	1	R	2
2	0	0	H	2
2	1	1	R	2

Esempio 6: **MTL0**
Macchina che si ferma sulla prima cella vuota a destra rispetto a quella dalla quale è partita.

S#	R	W	M	N#
1	0	0	L	2
1	1	1	L	2
2	0	0	H	2
2	1	1	L	2

MTR0 e MTL0 si fermano sempre perché il nastro contiene sempre un numero finito di celle non vuote.

Esempio 7: **MTR1**
Macchina che ricerca, per fermarsi, la prima cella a sinistra che contenga 1.

S#	R	W	M	N#
1	0	0	R	2
1	1	1	R	2
2	0	0	R	2
2	1	1	H	2

Esempio 8: **MTL1**

Macchina che ricerca, per fermarsi, la prima cella a destra che contenga 1.

S#	R	W	M	N#
1	0	0	L	2
1	1	1	L	2
2	0	0	L	2
2	1	1	H	2

MTR1 e MTL1 possono non fermarsi mai.

Vediamo alcune MT scritte nel linguaggio di "Turing Machine Simulator".

Esempio 9: **MTCOPIA**

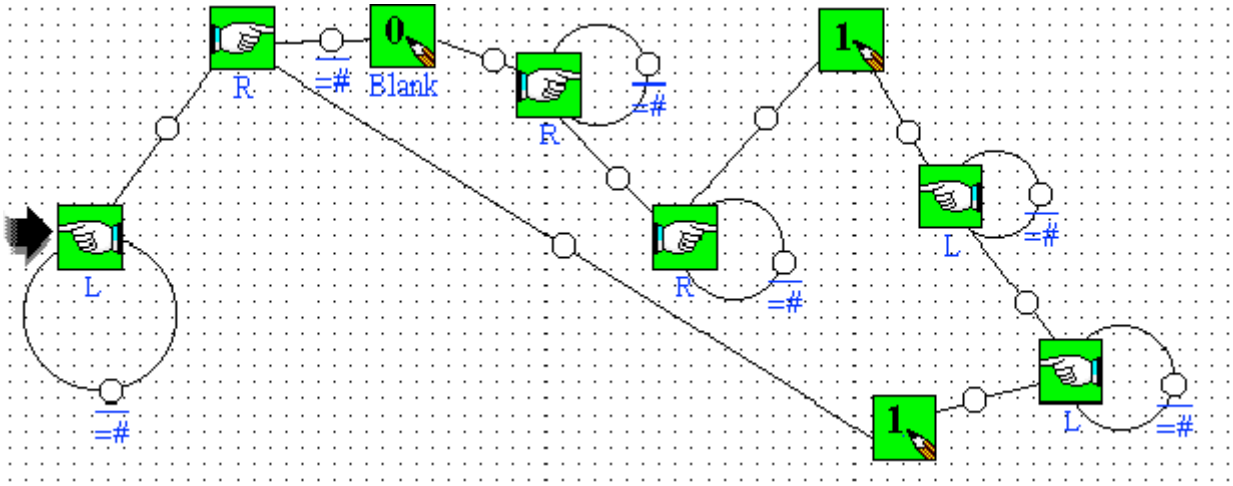
```
{copia una stringa di simboli 1 a destra della cella di partenza}

{S#=State  R=Read  W=Write  M=Move  N#=New State }


{ S# R  :  W  M  N# }

  1  0  :  0  R  2
  1  1  :  1  R  2
  2  0  :  0  L  3
  2  1  :  1  R  2
  3  1  :  0  L  4
  3  0  :  0  H  3
  4  0  :  0  L  5
  4  1  :  1  L  4
  5  0  :  1  R  6
  5  1  :  1  L  5
  6  0  :  0  R  7
  6  1  :  1  R  6
  7  0  :  1  L  3
  7  1  :  1  R  7
```

Visualizzazione di MT Copia con “Visual Turing”



Come rappresentiamo i **numeri naturali** con le macchine di Turing?

Il numero n viene rappresentato con n+1 simboli 1

Così:

0 ⟶ 1
1 ⟶ 11
2 ⟶ 111

Dato $n \in \mathbb{N}$, indicheremo la rappresentazione di n con $\overline{n}$

Esempio 10: **MTSUCC**
Calcola il **successore** di un numero.

```
{ S# R   :   W   M   N# }

  1   0   :   0   R   1
  1   1   :   1   R   2
  2   0   :   1   R   3
  2   1   :   1   R   2
  3   0   :   0   H   3
  3   1   :   0   H   3
```

Esercizio: Costruire la MT “predecessore”

Nel caso di funzioni binarie, l’input viene rappresentato da due stringhe di simboli 1, separate da 0.

Nel caso di funzione n-arie, , l’input viene rappresentato da *n* stringhe di simboli 1, separate da 0.

Esempio 11a: **MTEUmana**

Implementazione per il calcolo del **MCD** di due numeri.

```
{ S#  R   :   W   M   N#}

  1   0   :   0   R   1
  1   1   :   0   R   2
  2   0   :   0   R   12
  2   1   :   1   R   3
  3   0   :   0   R   4
  3   1   :   1   R   3
  4   0   :   0   L   5
  4   1   :   1   R   4
  5   0   :   1   L   9
  5   1   :   0   L   6
  6   0   :   0   L   7
  6   1   :   1   L   6
  7   0   :   1   R   8
  7   1   :   1   L   7
  8   0   :   1   L   10
  8   1   :   0   R   3
  9   0   :   1   L   10
  9   1   :   1   L   9
 10   0   :   0   H   10
 10   1   :   0   L   11
 11   0   :   1   L   1
 11   1   :   1   L   11
 12   0   :   0   H   12
 12   1   :   1   R   12
```

Esempio 11b: **MTEu-HCF**

Altra implementazione per il calcolo del **MCD** di due numeri.

```
{ S# R   :   W   M   N#}

  1   0   :   0   R   1
  1   1   :   0   R   2
  2   0   :   0   R   13
  2   1   :   1   R   3
  3   0   :   0   R   4
  3   1   :   1   R   3
  4   0   :   0   L   5
  4   1   :   1   R   4
  5   0   :   0   L   6
  5   1   :   0   L   6
  6   0   :   1   L   11
  6   1   :   1   L   7
  7   0   :   0   L   8
  7   1   :   1   L   7
  8   0   :   0   R   1
  8   1   :   1   L   9
  9   0   :   1   L   7
  9   1   :   1   L   10
 10   0   :   1   R   1
 10   1   :   1   L   10
 11   0   :   0   L   12
 11   1   :   1   L   11
 12   0   :   1   L   1
 12   1   :   1   L   12
 13   0   :   0   H   13
 13   1   :   1   R   13
```

Esempio 12: **MTsum**

Calcola la somma di due numeri (naturali)

```
{ S# R   :   W   M   N# }

  1   0   :   0   R   1
  1   1   :   0   R   2
  2   0   :   1   R   3
  2   1   :   1   R   2
  3   0   :   0   L   4
  3   1   :   1   R   3
  4   0   :   0   H   4
  4   1   :   0   H   4
```

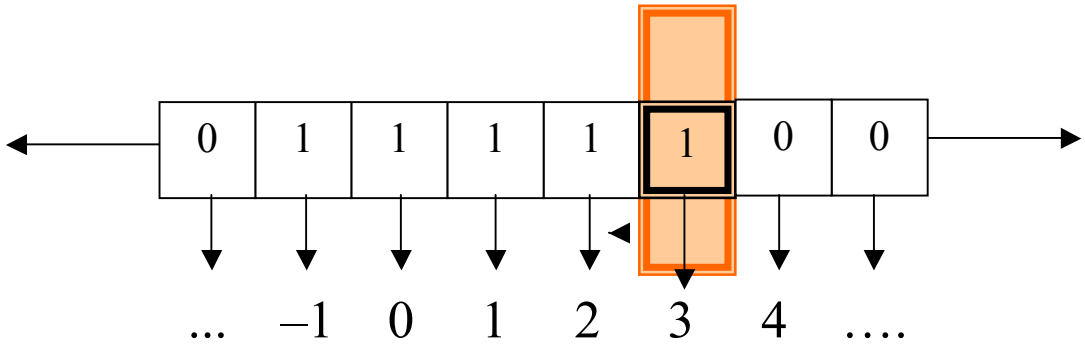
Esempio 13: **MTnsum**

Calcola la somma di n numeri (naturali)

```
{ S#  R   :   W   M   N# }

  1   0   :   0   R   1
  1   1   :   0   R   2
  2   0   :   1   R   3
  2   1   :   1   R   2
  3   0   :   0   R   4
  3   1   :   1   R   3
  4   0   :   0   L   5
  4   1   :   1   L   6
  5   0   :   0   L   5
  5   1   :   0   H   5
  6   0   :   0   L   6
  6   1   :   1   L   7
  7   0   :   0   R   8
  7   1   :   1   L   7
  8   0   :   0   R   1
  8   1   :   0   R   1
```

Ogni cella è individuata da un **numero intero**:



Da che cos'è determinata la **configurazione** di una MT in un certo tempo *t*?

La **configurazione** al tempo *t* è determinata dalla:

- **posizione** del lettore della macchina al tempo *t*;
- **condizione** del nastro al tempo *t* (cioè dal sistema delle iscrizioni che si trovano sul nastro in quell'istante);
- **stato** (interno) in cui si trova la macchina in *t*.

In virtù della definizione di MT (tavola), la configurazione K' in cui la macchina si viene a trovare dopo aver compiuto un passo resta *univocamente determinato* dalla configurazione K in cui la macchina si trovava prima del passo.

Matematicamente:

posizione ⟶ *numero intero*

condizione ⟶ *funzione c*: $\mathbb{Z} \xrightarrow{\text{interi}} \{0,1\}$ t.c. $c(x) = 1$ solo per un insieme finito di x.

Stato ⟶ *numero naturale*

Quindi: una **configurazione** è una tripla

$$K = (p, c, s)$$

posizione condizione stato

Data la configurazione K= (p, c, s), quale sarà la configurazione K' immediatamente successiva?

K'= (p', c', s'), dove:

a) p' = $\begin{cases} p+1, & \text{se la tavola della macchina che si trova nello stato } s \text{ e nella condizione } c(p) \text{ contiene } \mathbf{L} \text{ nella stessa riga;} \\ p-1, & \text{se la tavola della macchina che si trova nello stato } s \text{ e nella condizione } c(p) \text{ contiene } \mathbf{R} \text{ nella stessa riga;} \\ p, & \text{altrimenti.} \end{cases}$

b) $c'(n)$ = $\begin{cases} 0, & \text{se } n=p \text{ e la tavola della macchina che si trova nello stato } s \text{ e nella condizione } c(p) \text{ contiene 0 nella stessa riga;} \\ 1, & \text{se } n=p \text{ e la tavola della macchina che si trova nello stato } s \text{ e nella condizione } c(p) \text{ contiene 1 nella stessa riga;} \\ c(n), & \text{altrimenti.} \end{cases}$

c) s' è l'ultimo numero (dello stato) della riga che inizia con s e che contiene $c(p)$.

Intuitivamente: l'esecuzione di un computo da parte di una MT è una successione di configurazioni a partire dalla **configurazione iniziale** (data dalla posizione in cui la macchina si trova all'inizio, l'iscrizione che si trova sul nastro nel momento in cui la macchina parte, e uno stato iniziale) fino a una
configurazione finale (data dalla posizione in cui si ferma la macchina, dall'iscrizione del nastro in quel momento e dallo stato finale).

Definizione Una **computazione della MT *M*** a partire dalla posizione p_1 nella condizione iniziale c_1 e nello stato iniziale s_1 è una successione finita di configurazioni $(K_1, K_2, \ldots, K_n)$ di *M* t.c.

1) $K_1 = (p_1, c_1, s_1)$;
2) $K_{i+1} = K_i'$;
3) K_n è una configurazione finale di *M*; esiste cioè nella tavola di *M* una riga che contiene uno stato finale, una condizione finale c_n e H.

Possiamo dare la definizione generale di funzione *Turing-computabile*.

Definizione Una funzione numerica k-aria f^k: $N^k \rightarrow N$ si dice **Turing-computabile** se e solo se esiste una macchina di Turing *M* tale che per qualsiasi k-pla di numeri naturali $(n_1,...,n_k)$ esiste una computazione di *M* che soddisfa le seguenti condizioni:

a) la sua **condizione iniziale** è data dalla successione di cifre $\overline{n_1},...,\overline{n_k}$, intervallate dal simbolo vuoto (0) [input].

b) La sua **condizione finale** è data da $\overline{f^k(n_1,...,n_k)}$ [output].

Intuitivamente: Una funzione numerica *f* è Turing-computabile quando esiste una macchina di Turing che calcola il valore di *f* per ogni scelta possibile di argomenti.

♦ MACCHINA DI TURING UNIVERSALE

La caratteristica fondamentale delle MT è che ne esiste una che può "simulare'' il comportamento di tutte le altre [Turing 1936].

TEOREMA:(*esistenza della TM universale*)[Turing 1936]
Esiste una macchina di Turing *U* (detta *macchina di Turing universale*) tale che per ogni macchina di Turing *M*, la MT *U* può simulare il computo di *M* con input *n*, una volta che il nastro contenga come input una successione codificata (nell'alfabeto {0,1}) di istruzioni di *M* e di *n*. L'output di *U* è identico a quello di *M*.

Per dimostrare questo teorema occorre innanzitutto:

1. trovare un modo sistematico di *enumerare* le MT;
2. *codificare* la tavola di ciascuna MT in una successione di simboli 0,1.

1. Enumerazione delle MT (secondo Penrose)

$M_0 =$	1	0	0	L	1
	1	1	0	R	1
$M_1 =$	1	0	0	L	1
	1	1	1	R	1
$M_2 =$	1	0	0	L	1
	1	1	1	R	1
$M_3 =$	1	0	0	L	1
	1	1	0	H	1
$M_4 =$	1	0	0	L	1
	1	1	0	L	2
$M_5 =$	1	0	0	L	1
	1	1	1	R	1
$M_6 =$	1	0	0	L	1
	1	1	0	L	1
	2	0	0	L	1
. .	.	.	.	.	.
. .	.	.	.	.	.
. .	.	.	.	.	.

Definizione Il numero che identifica la TM, si dice **indice** della TM.

Quindi, l'indice di M_6 è 6.

Ovviamente questa enumerazione deve essere sistematica (si veda Penrose, cap. 2)

2. Codifica delle istruzioni (secondo Penrose)

Le istruzioni della n-esima Macchina di Turing M_n possono essere codificate, attraverso opportuni accorgimenti, come una **successione** di simboli 0 e 1; questo richiede, per esempio, che a ciascuno dei simboli L, R. H, venga attribuita una diversa successione di simboli 0,1.

La codifica è fatta in modo tale che all'n-esima macchina di Turing M_n venga attribuita una successione di simboli 0,1 che rappresenta (nel **sistema binario**) il numero *n*.

Per esempio, nel sistema di Penrose, la Macchina di Turing M_6 è codificata con 110, che rappresenta il numero binario 110 e cioè il numero 6 nel sistema decimale.

Ogni MT viene rappresentata da un numero *binario*.

Occorrerà anche una successione di simboli 0,1 per separare la codifica della MT che la MT universale simula e l'input vero e proprio.

- Scriviamo $M_n(m)=p$ per indicare che quando l'n-esima MT agisce sul numero m dà come risultato il numero p.

- Sia M_n l'n-esima MT, sia $m \in \mathbb{N}$ e U la MT universale.

  Scriviamo $U(n, m) := M_n(m)$

  (Quando sul nastro viene scritto il numero m e la (codifica della) MT M_n, allora la Macchina di Turing Universale U applicata a m dà come risultato esattamente lo stesso risultato che avrebbe dato M_n applicata a m.)

Poiché U è una MT, anch'essa sarà individuata da un numero binario.

Qual è il numero binario che rappresenta U ?

Codifica binaria di *U* ⟶ **Decodifica** ⟶ **Tavola di *U***

CODIFICA BINARIA DELLA MACCHINA DI TURING UNIVERSALE (secondo PENROSE)

1000000001011101001101000100101010110100011010001010000011010100110100010101001011010000110100010100101011
1001001110100101001001011101010001110101010010010101110101010011010001010001010110100000110100100000101011
1000100111010010100001010111010010001110100101010000101110100101001101000010000111010100001110101000010010
0
1110100010101011010100101011010000011010101001011010010010001101000000001101000000111010100101010111010000
0100111010010101010101010111010000101010111010000101000101110100010100110100100001010011010010100100110100
1
0001011010100010111010010010101110100101000111010100101001001110101010100000110100101010111010100100010110
1010000101101010001001101010101010001011010010101001001011010100100101110101010010101110101001010011010101
0
0001110100010010010101110101010010101110101010000011101010010000011010101010010111010100101011010001001000
1
1101000000011101001010010101010111010010100100101011101000001010111010000100011101000001010100111010000010
1
0111010000010001011101000100001110100001001010011101000100001011010001010010111010001010010111010010000010
1
0100010101001001110100010101010111010010000011101001001010101011101010101001101001000101011010010010011010
0
0000001011010000010001101000001001011010000000001101001010001011101001010100011010010100101011010000010011
1
0100101010010111010010011101010000001010111010100000011010101000101010110100101010110101000010101110101001
0
1010111010100010010111010100100001011101000000111010100100010111010100101001101010100010111010100101001011
1
0101000001011101010100000101110100000011101010100001010111010010101011010101000010111010100010101011101010
1
0010010111010101010000111010100000000111010010010000110100100100010111010101010100111010000000010110100100
0
1010101010100101110100100001101001000101010111010000100011101000100001110100001101000000010110100000100101
1
1010101001010101101000100010010111010000010011101010100110100000101010110100001000011101001000010001110101
0
1010101001110100001001001110100010010000111010000101001011010000101000011101010101010101110100010010011010
0
0100100111010100101001011101000100010101110100000001110100010010010111010011010010010000101101010101001101
0
0101000101110100001101010000100010111010100110101010010100101110101010011010010010111010011010010000010110
1
0001010101000111010010000101011010000001001101001000100101110100100001101010000010010111010010010100110100
1
0010101011010011010010010100101110100010001010111010000001110100010010010111010011010010000101101010101001
0
1110100100111010010101010010111010010001110101000001010101110011010100000101101001001110101000000101110100
1
0111010010101010111010100010010111010010011101001010100010111010001001110101000010110100100111010010101010
1
1110100100011101001010101001011101001000111010100000101010111001101010000010111010010011101010000001011101
0
1011010100000101011010010100101110101000010010111010000110101000100001011010100110101000100010111010101010
0
1011101010000010101110100101001011101010000100101110100001101010001000010111010100110101000100010111010101
0
1010001110101001001001011101010001110101001010001011101010001011101010000100101111010010001110100010100010
1
1010010100101110101001010100101110100101010101010110101000010101010110100001001110100001010101010111010101
0
0010101110101010001010111010000001110101010001001011101000000111010101001000101110101000000110101000010110
1
0000001110100100000010111010100011101010010001010111010100110101010100010101101000001101010101001010110100
0
0010100101011010100100111010100110101010100100101110101001101000100100111010000011010101010010101110100010
1
0010100010100101011101000100011101001001010101010110100101001010001101001000000101111010000011010101001010
1
0101010101110100101101001001000101011001101001001001010101110100110100100100101101001011010010010010011011
0
1010010001010101011101000100011101001001010101010110100101001010001101001000000101111010000011010101001010
1
1011010010101011010010001000101110100010101011010100000101011010001000001101001000101011010000100111010100
1
0101010101110100101101001001000101011001101001001001010101110100110100100100101101001011010010010010011011
0
0100101101001001010001011001101001001010010101110100010101110100100101110011010010010101001011100110100101
0
0010101011101000100011101000010100101110100101000101110100101000101011010001001110100101000100101110100010
0
1101001010010001011100110100100010001110100010011101001010010101011100110100001010010011001101010000011010
0
0000001110100101010010101011101001000111010010101001010111001101000010100100110011010101010101011010
0
0010101010010101110011010100010000110100000000111010001001010101011101000100011101010101010101010110100001
0
0
1110100100010010101110100101010001001101010000000101101001001110101000010101110100100001101010000000010110
1
0100011101010010010111010000110101000010101011010100010111010100001010010111010100010111010100010101011110
0
0110101000101011010000110101000100101

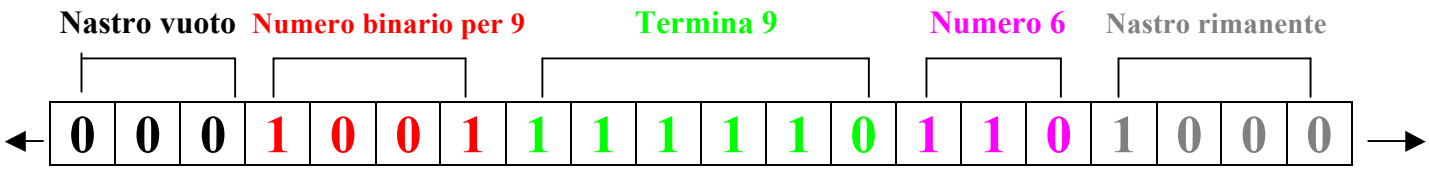
Esempio: Vogliamo che la MT U simuli la MT M_9 definita (in Penrose) nel modo seguente:

$M_9 =$	1	0	0	L	1
	1	1	0	R	1

Il numero binario che codifica M_9 è 1001.
Vogliamo che U simuli il comportamento di M_9 applicata al numero 6; vogliamo quindi

$$U(9,6).$$

Nella rappresentazione di Penrose, il nastro, prima che la macchina parta, si presenterà così:



CONSEGUENZE DELL'ESISTENZA DELLA MTU

Dire che una MT Universale (**MTU**) è una MT che può simulare (incorporare) ogni MT è come dire che:

- Dati *m* diagrammi di flusso, scritti con un certo simbolismo, che descrivono l'esecuzione di *m* compiti specifici, c'è un **diagramma di flusso universale** *n* che può riprodurre ciascuno degli *m* diagrammi specifici.

- UMT è uno strumento **programmabile** di computazione "general purpose", che fornisce le basi logiche per la costruzione dei moderni computer. La sua universalità è garantita dalla distinzione fra le operazioni elementari, eseguite dall'**hardware**, e le istruzioni specificate da un dato programma, contenuto nel **software**.

A. Una MTU può essere **fisicamente implementata** su diversi tipi di hardware.

B. Ogni computer convenzionale è **logicamente** (non fisicamente) **equivalente** a una MTU.

Problema: B. implica la tesi (M) ?

Tesi (**M)**:

Se qualcosa è calcolabile da un computer allora è calcolabile da una MTU

Risposta: **NO**! [contrariamente a quanto sostenuto da gran parte della letteratura: si veda Copeland, "The Church-Tuting thesis", *Stanford Encyclopedia of Philosophy*, pp. 4-5]

Turing non ha dimostrato che una MTU può calcolare una qualsiasi funzione calcolabile da un computer con architettura. Turing ha dimostrato che

Una MTU può calcolare una qualsiasi funzione che una MT può calcolare.

La tesi (M) è proprio una tesi, che può essere accettata o meno; essa postula l'equivalenza fra le nozioni di essere

- **calcolabile da una macchina di Turing**

e

- **calcolabile da una macchina che si conforma alle leggi fisiche del mondo reale**.

La tesi (M), contrariamente a quanto spesso si sostiene, non è la tesi di Turing-Church!

La **tesi di Turing-Church** (**TTC**) concerne le nozioni di

metodo **effettivo** e metodo **meccanico**

♦ TESI DI TURING

Le macchine di Turing ["macchine logiche di calcolo" nell'originale articolo di Turing del 1936] possono fare tutto ciò che può essere descritto come "puramente meccanico".

In altri termini: se esiste una procedura effettiva per ottenere il valore di una funzione matematica, allora questa funzione può essere calcolata da una MT.

Le MT furono introdotte da Turing per analizzare l'*Entscheidungsproblem*. Esse quindi vogliono simulare un **essere umano impegnato in un calcolo**.

La tesi di Turing è plausibile?

Nel 1936 A. **Church** propose un'altra nozione formale (molto diversa rispetto a quella di Turing) di metodo effettivo. In seguito altre nozioni formali sono state proposte.

Risultato:

Una funzione è Turing-computabile sse è Church-computabile sse…..

⇓

Le nozioni di Turing-computabilità e di Church-computabilità individuano la stessa classe di funzioni.

Possiamo parlare allora di **TESI DI TURING-CHURCH** (TTC)

Se accettiamo la validità di TTC, allora il problema dell'esistenza e della non-esistenza di metodi effettivi si riduce (in matematica e logica) al problema dell'esistenza e della non-esistenza di Macchine di Turing.

Teorema (Turing-Church) Se TTC vale, allora non c'è nessun metodo effettivo (meccanico) per il calcolo dei predicati. Formalmente: non esiste nessuna MT che possa determinare (decidere), in un numero finito di passi, se ogni proposizione del calcolo dei predicati è o non è un teorema del calcolo.

♦ MACCHINE DI TURING E PROBLEMI DI DECIDIBILITÀ

I problemi di decidibilità richiedo risposte effettive (meccaniche) di tipo sì/no (1/0).

Se accettiamo TTC, possiamo riformulare i problemi di decisione nei termini di MT.

Esempi:
1) Sia P il problema "*n* è un numero pari"?

Questo problema è decidibile perché esiste una MT che produce come output

1, se *n* è pari;
0, se *n* è dispari.

2) Esercizio: Costruire la MT che decida il problema "n è un numero pari".

Diciamo che un problema è **decidibile** quando esiste una macchina di Turing che fornisce risposte 1/0 in modo adeguato (in modo simile all'esempio precedente).

Se accettiamo TTC, possiamo riformulare l'*Entscheidungsproblem*) di Hilbert nei termini del cosiddetto

Problema della fermata (*Halting problem*)

Esiste una macchina di Turing che sia in grado di stabilire (per ogni $n,m \in \mathbf{N}$) se la n-esima macchina di Turing M_n che agisce sul numero m si fermi oppure no ?

Intuitivamente:
Le macchine di Turing hanno la capacità di *autodescriversi.* In particolare, nel linguaggio di una macchina di Turing M sarà possibile esprimere il problema: la macchina M si ferma o non si ferma quando le viene chiesto di calcolare il valore di una funzione per certi argomenti?
Potremo allora porre alla stessa macchina la seguente domanda:

“Ti fermerai qualora ti venga chiesto se non ti fermerai?”

Risolvere positivamente il problema della fermata implicherebbe stabilire che ogni problema matematico è decidibile, perché esisterebbe un algoritmo che fornisce una risposta 1/0 (positiva/negativa) a un qualsiasi problema matematico.

Teorema di Turing (irrisolvibilità del problema della fermata)

Non esiste nessuna macchina di Turing che sia in grado di decidere se una macchina di Turing si fermerà.
In altri termini, non esiste nessuna macchina di Turing che, dato l'input (*n*,*m*), produca l'output 1 se la macchina di Turing di indice *n* e di input *m* si ferma; produca l'output 0 altrimenti.

Se accettiamo TTC, il Teorema di Turing si può riformulare così:

Non esiste nessun algoritmo universale per decidere i problemi matematici.

Dimostrazione (informale) del Teorema di Turing.

L'idea chiave della dimostrazione è la seguente:
Se esistesse una MT in grado di rispondere alla domanda "Ti fermerai qualora ti venga chiesto se non ti fermerai?", allora si determinerebbe una contraddizione: la macchina si ferma se e solo se la macchina non si ferma.

Supponiamo che esista una MT di indice i (indicata con H_i) tale che, presa una qualsiasi TM di indice n, per esempio T_n e preso un qualsiasi input m,

H_i produce l'output

$$\begin{cases} 1, \text{ se } T_n \text{ si ferma quando applicata a } m \text{ (cioè } \exists T_n(m)); \\ 0, \text{ se } T_n \textbf{ non} \text{ si ferma quando applicata a } m. \end{cases}$$

Quindi:

$$H_i(n,m) = \begin{cases} 1, \text{ se } \exists T_n(m); \\ \\ 0, \text{ se } T_n \textbf{ non} \text{ si ferma quando applicata a } m. \end{cases}$$

Immaginiamo ora una matrice infinita che elenchi tutti gli output di H_i. Questa matrice sarà del tipo seguente:

		Input m ↓						
		0	1	2	3	4	5	6……
	0	***0***	0	0	0	0	0	0……
	1	1	***1***	1	1	1	1	1……
	2	1	1	***0***	0	1	1	1……
Macchine di Turing	3	1	1	1	***1***	1	1	1……
$n \rightarrow$	4	0	0	0	0	***0***	0	0……
	5	1	1	1	0	0	***1***	1……
	6	0	1	1	1	1	1	***1***……
	•	•	•	•	•	•	•	•……
	•	•	•	•	•	•	•	•……

Cambiamo leggermente la MT H_i in modo tale da ottenere una nuova macchina H_j tale che

- Applicata all'input n, H_j produce l'output 1, se $H_i(n,n)=0$, cioè, sse l'n-esima macchina di Turing applicata all'input n **non si ferma**.

- Applicata all'input n, H_j **non si ferma** (cioè non produce nessun output) sse $H_i(n,n)=1$, cioè, sse l'n-esima macchina di Turing applicata all'input n produce un output ($\exists T_n(n)$).

Si può dimostrare che H_j è ottenuta da H_i:
a) sostituendo tutti gli output 0 della diagonale con 1;
b) facendo sì che H_i si muova all'infinito a sinistra, quando nella diagonale compare l'output 1.

Otteniamo allora:

➢ $H_i(j\ ,j)=1$	sse	la j-esima macchina di Turing (cioè H_j) applicata a j si ferma
	sse	$H_j(j)=1$.
➢ $H_j(j)=1$	sse	$H_i(j\ ,j)=0$
	sse	la j-esima macchina di Turing (cioè H_j) applicata a j **non si ferma**.

Quindi:

La macchina di Turing H_j si ferma sse e solo non si ferma! Contraddizione.

□

⇓

Turing ha dimostrato che l'*Entscheidungsproblem* di Hilbert non ha soluzione.

Per dimostrare che un problema matematico è *indecidibile* è sufficiente dimostrare che la sua soluzione è equivalente alla soluzione del problema della fermata.

Il Teorema di Turing ovviamente non dimostra l'impossibilità di decidere *tutti* i problemi matematici!

Il Teorema di Turing **non** dimostra che esistono alcuni problemi matematici indecidibili (a esclusione del problema della fermata!). Il Teorema di Turing non dice niente sulla (ir)risolvibilità di *specifici problemi matematici.*