Cosa succede?

```
1   n = 1231
2
3   d = 2
4   while d*d <= n and n%d != 0:
5       d += 1
6
7   if n%d == 0:
8       print(False)
9   else:
10      print(True)
```

Problema della Fermata

- Dato un programma P ed un input x, è possibile stabilire se $P(x)$ converge o diverge?

- Esiste un programma $\mathrm{Termina}$ che con input P e x restituisce True se $P(x)$ termina e False altrimenti?

Problema della Fermata

Indecidibilità

```python
def Paradosso(P):
    while Termina(P, P) == True:
        print('...siamo nel while...')
    print('Fatto!')

Paradosso(Paradosso)
```

Paradosso(Paradosso) ↓

⇓

Termina(Paradosso, Paradosso) == False

⇓

Paradosso(Type equation here.Paradosso)↑

Paradosso(Paradosso) ↑

⇓

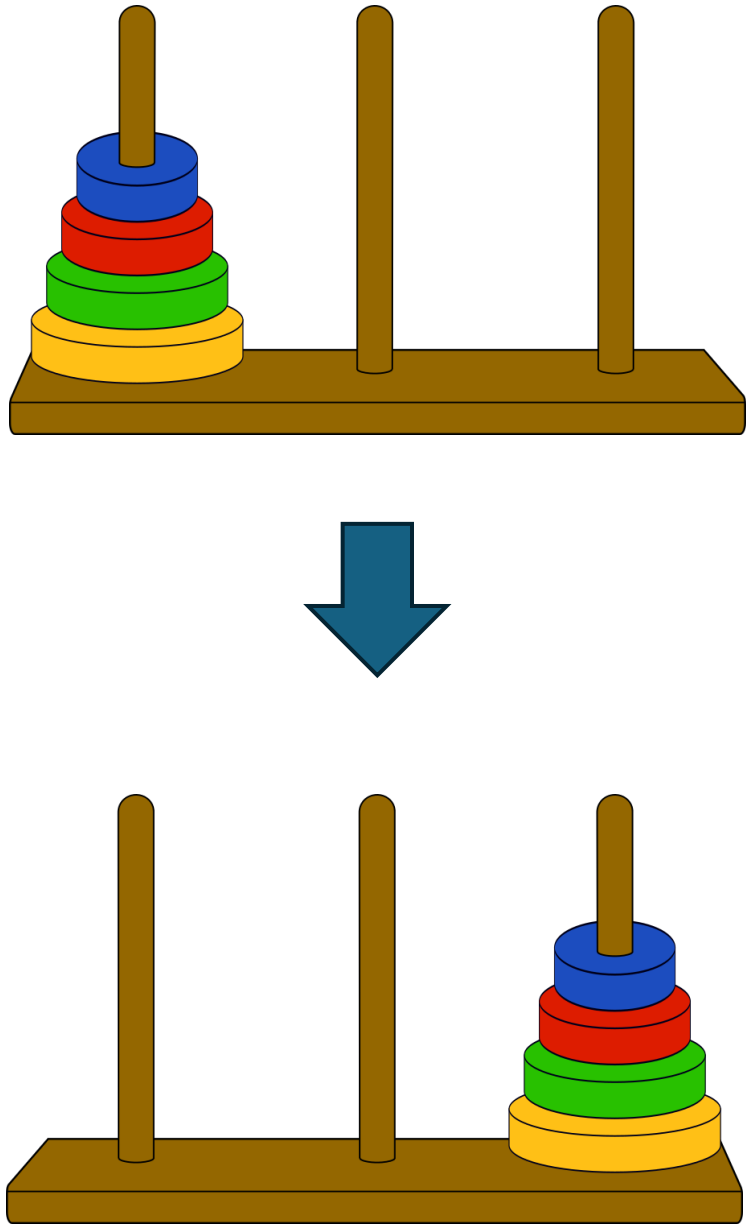
Termina(Paradosso, Paradosso) == True

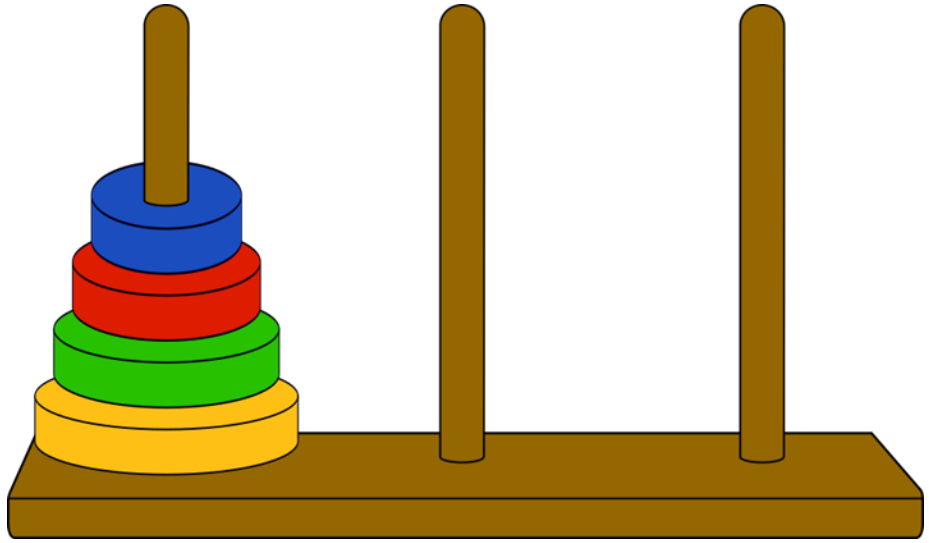
⇓

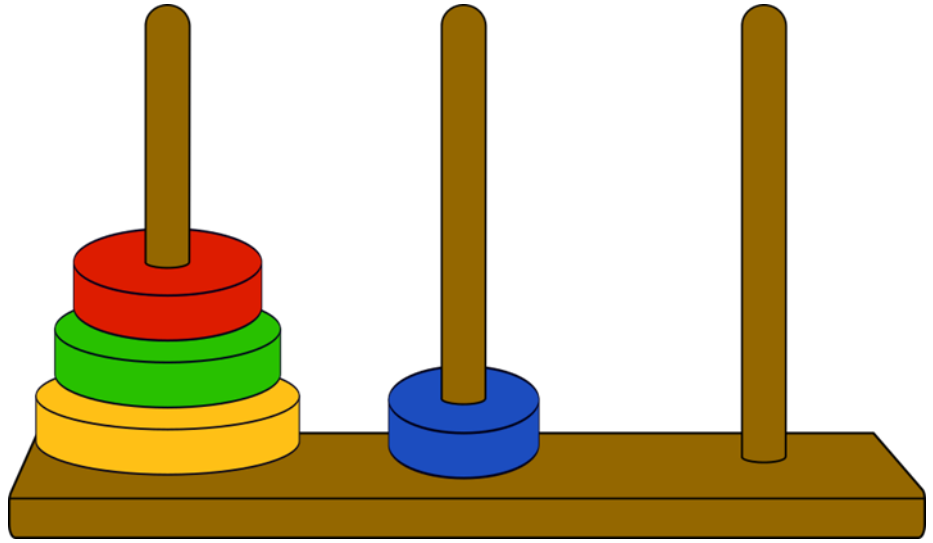
Paradosso(Paradosso)↓

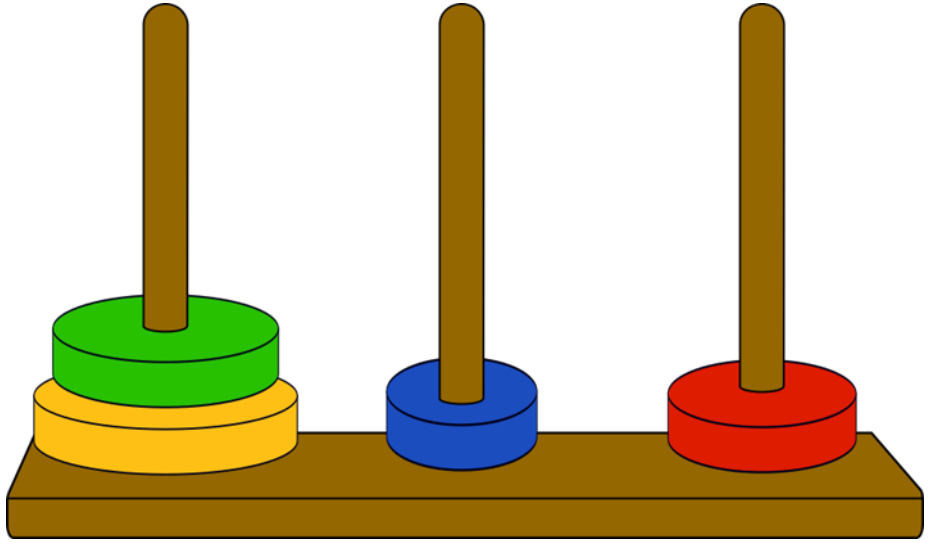
Intrattabilità

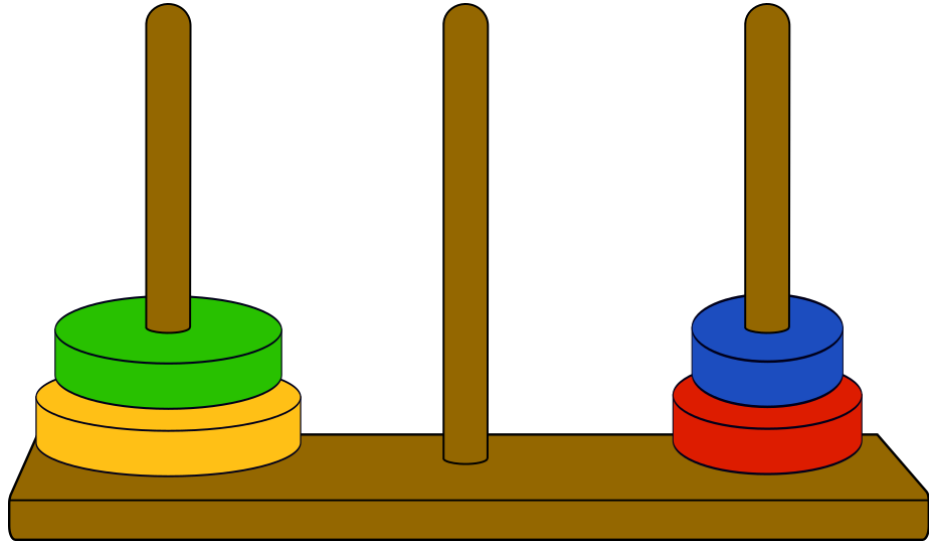
Torri di Hanoi











Quante mosse sono **necessarie**?

$\mathbf{T(n)} =$ numero di mosse necessario per $\mathbf{n}$ dischi

Spostare il disco più grande sul piolo finale richiede

1. spostare tutti gli altri $\mathbf{n - 1}$ dischi sull'altro piolo libero, rispettando le regole del gioco; questo richiede $\mathbf{T(n - 1)}$ mosse
2. spostare il disco più grande, ora libero, sul piolo finale.

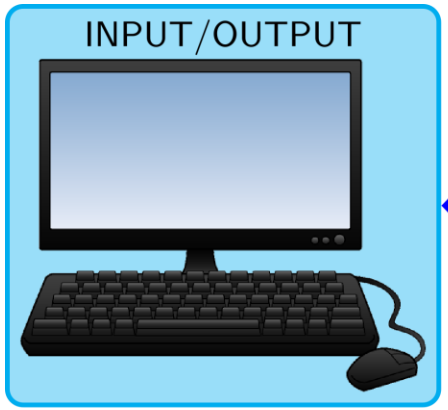
$$\mathbf{T(n) = 2T(n - 1) + 1}$$

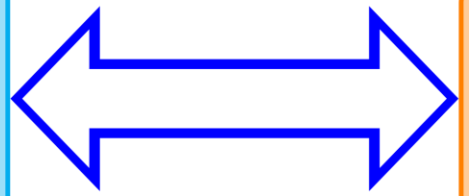
Quante mosse sono **necessarie**?

$$
\begin{aligned}
\mathbf{T(n)} &= \mathbf{2T(n-1)+1} \\
&= \mathbf{2(2T(n-2)+1)+1} \\
&= \mathbf{4T(n-2)+3} \\
&= \mathbf{8T(n-3)+7} \\
&\dots \\
&= \mathbf{2^k T(n-k)+2^k-1} \\
&\dots
\end{aligned}
$$

$$\mathbf{T(n) = 2^{n-1}T(1) + 2^{n-1} = 2^n - 1}$$

Macchina di Von Neumann





CPU
Unità di elaborazione

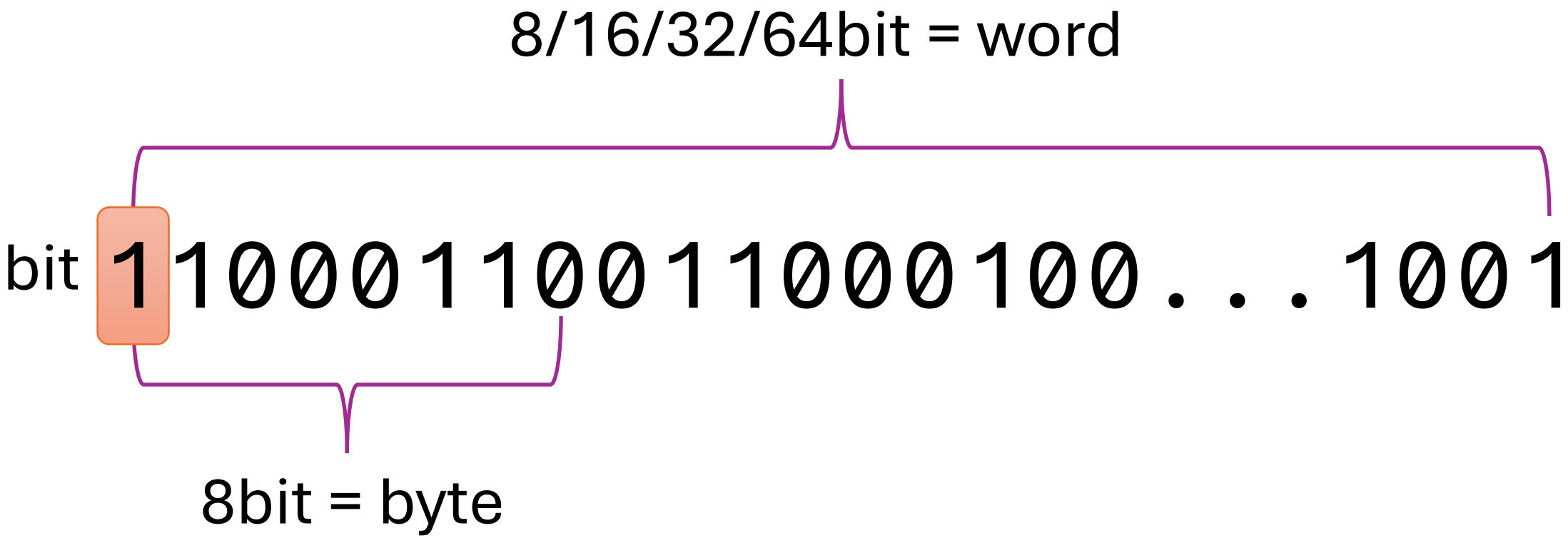
Lettura/Scrittura

MEMORIA

Indirizzo	Dato
0x0001	10110011
0x0002	11001100
0x0003	11110000
⋮	⋮

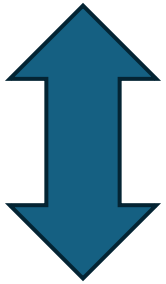
dati + programmi

Codice binario



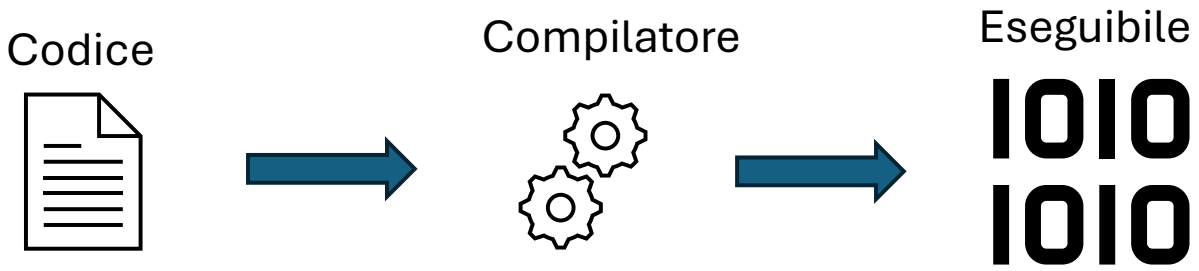
Programmi

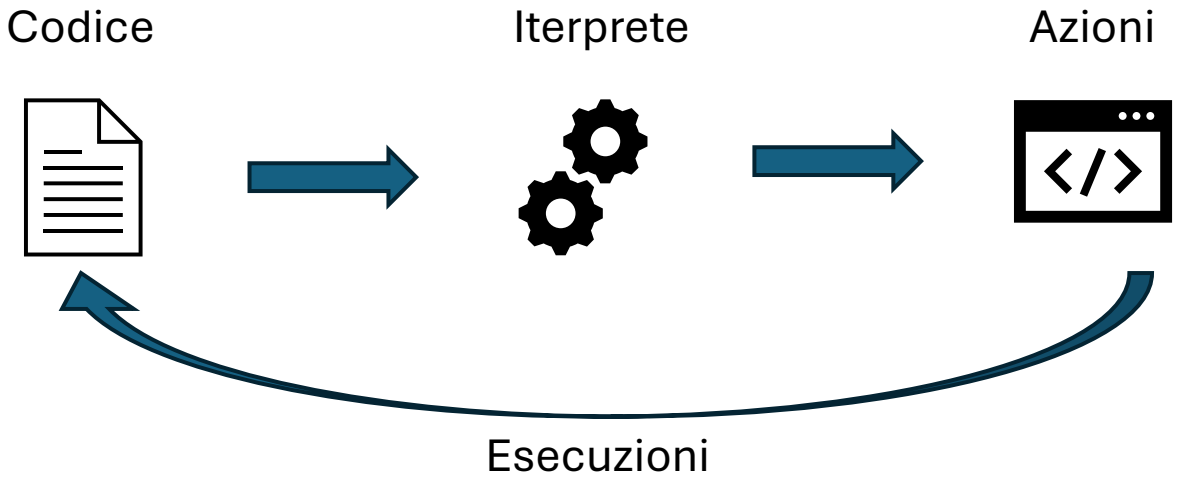
```
if n == 0:
    print('Zero')
else:
    print('Non zero')
```



```
01111100 01010110   # carica il valore di n
01100100 10010001   # carica il valore 0
01101011 00000010   # confronta i precedenti valori
01110010 01010110   # se uguali salta a...
...
```

Compilatore vs Interprete





Rappresentazione dei dati

Interi

$$
\begin{aligned}
11110110100 &= 1\times 2^{10}+1\times 2^{9}+1\times 2^{8}+1\times 2^{7}+0\times 2^{6}+1\times 2^{5}+\\
&\quad 1\times 2^{4}+0\times 2^{3}+1\times 2^{2}+0\times 2^{1}+0\times 2^{0}\\
&= 1024+512+256+128+32+16+4=1972.
\end{aligned}
$$

$$\lfloor N/2\rfloor \Longleftrightarrow b'_k b'_{k-1}\ldots b'_0$$

$$
\begin{aligned}
N &= 2\lfloor N/2\rfloor + b_0\\
&= 2\left(b'_k\times 2^{k}+\ldots+b'_0\times 2^{0}\right)+b_0\\
&= b'_k\times 2^{k+1}+\ldots+b'_0\times 2^{1}+b_0\times 2^{0}
\end{aligned}
$$

Con n bit 2^n interi. Anche negativi, p.e. usare un bit per il segno

Rappresentazione dei dati

Razionali

$$x = \pm . m \times 2^{\pm e}$$

Rappresentazione in virgola mobile. IEEE 754

Rappresentazione dei dati

Caratteri e testo