Ricorsione Informatica e Principio di Induzione Matematica

Prof. Ing. Loris Penserini, PhD

https://orcid.org/0009-0008-6157-0396

Ricorsione Informatica e
Principio di Induzione Matematica

Principio di Induzione Matematica

Per il **Principio d'Induzione Matematica** (**PIM**) l'obiettivo è **dimostrare logicamente** una proprietà **per tutti i numeri naturali**, tramite due passi puramente **deduttivi**:

1. Base: $P(0)$ è vero.
2. Passo: da $P(n) \Rightarrow P(n+1)$ **deduci** $\forall n\, P(n)$.

- Non c'è nessuna probabilità o generalizzazione empirica: se le premesse sono vere, la conclusione è **necessariamente** vera.
- **Quindi, dal punto di vista logico, il PIM è una regola di inferenza deduttiva all'interno della logica dei predicati.**

Induzione Empirica

Ho osservato 1000 cigni bianchi → quindi probabilmente tutti i cigni sono bianchi

Qui si parte da casi particolari osservati e si generalizza per analogia o probabilità a un caso universale.

È una **inferenza ampliativa** (la conclusione va oltre le premesse) e non garantisce certezza logica.

Da non confondersi: il PIM è inferenza deduttiva

Il motivo per cui il nome «**induzione**» è rimasto è che l'immagine intuitiva è la stessa: nell'induzione empirica "salgo" da molti casi osservati a una regola generale; nel **PIM** "salgo" da $P(0)$ a $P(1)$, poi $P(2)$, e così via, fino a tutti.

Ma la differenza sostanziale è che **nel PIM questo "salire" non è inferenza ampliativa**, **bensì una catena deduttiva** infinita compendiata in modo finito.

Ricorsione vs PMI

Differenza pratica

- In **matematica**, il **PIM** è un **metodo dimostrativo**: garantisce che una proprietà valga sempre.
- In **informatica**, la **ricorsione** è un **metodo computativo**: serve a *calcolare* il valore richiesto.

Tuttavia, dietro le quinte, la logica che rende possibile la ricorsione (cioè sapere che la funzione "prima o poi" arriva al caso base e quindi produce un risultato) è giustificabile proprio con un ragionamento induttivo.

Ricorsione vs Induzione

L'idea chiave è che sui numeri naturali (e, più in generale, su strutture "induttive" come array, liste, alberi, ecc.) **ricorsione** e **induzione** sono due facce della stessa medaglia:

- Ricorsione: è il principio di definizione (come si costruiscono funzioni),
- Induzione: è il principio di dimostrazione (come si dimostrano proprietà) lungo la stessa struttura.

Formalizzazione moderna

Quando la logica dei predicati venne formalizzata da **Frege** e **Peano**, si chiarì che:

$$[P(0) \wedge \forall n (P(n) \rightarrow P(n+1))] \Rightarrow \forall n P(n)$$

è un teorema deduttivo nella logica del primo ordine. In parole semplici, se una proprietà è vera per 0 e se da "vera per n" segue "vera per n+1", allora è vera per tutti i numeri naturali.

Non c'è nessun elemento empirico, statistico o ampliativo: la conclusione è una conseguenza necessaria delle premesse, data la struttura dei naturali definita dagli assiomi di Peano.

Equivalenza concettuale

L'**induzione matematica** dimostra che una proprietà è vera per *tutti i numeri naturali* costruendo una catena di validità.

La **ricorsione** calcola un risultato per un input generico riducendolo al caso base, percorrendo una catena di chiamate fino al risultato finale.

In pratica:

- Il **caso base dell'induzione** corrisponde al **caso base della ricorsione**.
- Il **passo induttivo** corrisponde alla **chiamata ricorsiva** (che riduce il problema a uno più semplice).

Principio dell'Induzione (formale)

Il principio d'induzione asserisce che se **P** è una proprietà (o predicato) sui numeri naturali ($\mathbb{N}$),

- se $\boldsymbol{P}(\boldsymbol{n_0})$ è vera per $\boldsymbol{n_0} \in \mathbb{N}$
- *Ipotesi induttiva*: si assume che $\boldsymbol{P}(\boldsymbol{n})$ è vera con $\boldsymbol{n} \in \mathbb{N}, \boldsymbol{n} \geq \boldsymbol{n_0}$ e se si dimostra che

$$\boldsymbol{P}(\boldsymbol{n}+\mathbf{1}) \text{ è vera}$$

allora

$$\boldsymbol{P}(\boldsymbol{n}) \text{ è } \boldsymbol{vera}\ \forall \boldsymbol{n} \in \mathbb{N}$$

PW-1_induz

<u>*Problema:*</u>

Si dimostri che la proprietà

$P(n) \equiv 1 + 3 + 5 + \cdots + (2n - 1) = n^2, \ \forall \boldsymbol{n} \in \mathbb{N}.$

Es. $P(0) = 0 = 0^2, \ P(1) = 1 = 1^2, \ P(2) = 1 + 3 = 2^2,$
$P(3) = 1 + 3 + 5 = 3^2, P(4) = 1 + 3 + 5 + 7 = 4^2, \ldots$

PW-1_induz : Soluzione

Problema:

Si dimostri che la proprietà

$P(n) \equiv 1 + 3 + 5 + \cdots + (2n - 1) = n^2, \ \forall \boldsymbol{n} \in \mathbb{N}.$

Procedimento (dimostrazione per induzione):

$$P(0) \text{ è } vera$$

$$Consideriamo\ un\ generico\ n \geq 1\ con$$

$$\boldsymbol{n} \in \mathbb{N}\ e\ supponiamo\ \boldsymbol{P(n)}\ vera\ ,$$

$$dimostriamo\ P(n+1)\ sia\ vera$$

PW-1_induz : Soluzione

<u>Problema:</u>

Si dimostri che la proprietà

$$P(n) \equiv 1 + 3 + 5 + \cdots + (2n - 1) = n^2,\ \forall \boldsymbol{n} \in \mathbb{N}.$$

<u>Soluzione, dimostrazione per induzione:</u>

$$P(n + 1) = 1 + 3 + 5 + \cdots + (2n - 1) + (2(n + 1) - 1)$$

$$ma\ per\ ipotesi: 1 + 3 + 5 + \cdots + (2n - 1) = n^2$$

$$Per\ cui: P(n + 1) = n^2 + (2(n + 1) - 1) =$$

$$= n^2 + 2n + 1 = (n + 1)^2$$

Cioè *P(n+1)* è vera

PW-2_induz

Problema:

Si dimostri che la proprietà $P(n) \equiv n \cdot (n+3), \forall \boldsymbol{n} \in \mathbb{N}$, è pari.

PW-2_induz : Soluzione

<u>Problema:</u>

Si dimostri che la proprietà $P(n) \equiv n \cdot (n+3), \forall \boldsymbol{n} \in \mathbb{N}$, è pari.

<u>Soluzione, dimostrazione per induzione:</u>

$$P(n+1) = (n+1)^2 + 3 \cdot (n+1) = n^2 + 2n + 1 + 3n + 3$$

$$= n \cdot (n+3) + 2 \cdot (n+2) = P(n) + 2 \cdot (n+2)$$

Per cui vera poiché composta da due numeri pari.

PW-3_induz

Problema:

Si dimostri che la proprietà $P(n) \equiv (n^3 + 2n), \forall \boldsymbol{n} \in \mathbb{N}$, è divisibile per 3.

PW-3_induz: Soluzione

Problema:

Si dimostri che la proprietà $P(n) \equiv (n^3 + 2n), \forall \boldsymbol{n} \in \mathbb{N}$, è divisibile per 3.

Soluzione, dimostrazione per induzione:

$P(0)$ è vera

Per il principio dell'induzione se è vera per $P(n+1)$ allora è vera $\forall n$, per cui:

$$P(n+1) = (n+1)^3 + 2(n+1) = n^3 + 3n^2 + 3n + 1 + 2n + 2$$

E' facile verificare che ogni singolo fattore è divisibile per 3, per cui anche $P(n+1)$ risulta divisibile per 3.

Ricorsione Informatica

La ricorsione è uno strumento molto potente per realizzare alcune tipologie di algoritmi, ma è anche fonte di molti errori di difficile diagnosi.

La ricorsione è connessa al principio di induzione matematica.

La particolarità della ricorsione è nel fatto che il metodo (o funzione) richiama se stesso e questo comporta la sospensione del **metodo chiamante** in attesa del risultato del **metodo chiamato**.

Fasi della Ricorsione

Quando un metodo ricorsivo invoca se stesso, la macchina virtuale Java (o l'Interprete Python) esegue le stesse azioni che vengono eseguite quando viene invocato un metodo qualsiasi:

- sospende l'esecuzione del metodo invocante
- esegue il metodo invocato fino alla sua terminazione
- riprende l'esecuzione del metodo invocante dal punto in cui era stato sospeso

I principali tipi di ricorsione: **semplice**, **in coda** e **multipla**.

Ricorsione semplice (lineare)

È la forma più comune: la funzione richiama sé stessa una sola volta per volta.

Esempi: somma dei primi n numeri, il fattoriale, ecc.

Ricorsione in Coda

Si parla di ricorsione in coda (o tail recursion) quando la chiamata ricorsiva è l'ultima operazione eseguita dalla funzione.

In altre parole, dopo la chiamata ricorsiva non ci sono più calcoli da fare.

Ricorsione Multipla

Qui la funzione richiama sé stessa più di una volta in ciascun passo.

È tipica di problemi che si ramificano (alberi, combinazioni, Fibonacci, ecc.).

PW-4: Somma dei primi n numeri

Induzione matematica

Problema:

Si dimostri che la proprietà

$$P(n) = 1 + 2 + 3 + \cdots + n = \frac{n \cdot (n+1)}{2}, \ \forall \boldsymbol{n} \in \mathbb{N}.$$

PW-4: Somma dei primi n numeri

Induzione matematica

Problema:

Si dimostri che la proprietà

$P(n) = 1 + 2 + 3 + \cdots + n = \frac{n \cdot (n+1)}{2}, \ \forall \boldsymbol{n} \in \mathbb{N}.$

Soluzione, dimostrazione per induzione:

Caso base: $P(1) =1$

Passo induttivo: supponiamo vale per P(n)

Dimostriamo che vale anche per:

$$P(n+1) = 1 + 2 + \cdots + n + (n+1) = \frac{n \cdot (n+1)}{2} + (n+1) = \frac{n \cdot (n+1) + 2 \cdot (n+1)}{2} =$$

$$\boldsymbol{\frac{(n+1)(n+2)}{2}}$$

Come volevasi dimostrare.

PW-4: Soluzione informatica ricorsiva

Ricorsione Informatica:

Caso base: **somma(1) = 1**

Caso «n»:

somma(n) = somma(n-1) + n

python **Ricorsione Semplice**

```python
def somma(n):
    if n == 1:  # caso base
        return 1
    else:       # passo ricorsivo
        return somma(n-1) + n
```

Es. n=3 →

somma(3) | return somma(3-1) + 3

somma (2) | return somma(2-1) + 2

somma (1) | return 1

Chiamate alla funzione...

Ricorsione...

PW-4: Soluzione informatica ricorsiva

Ricorsione in Coda

```
def somma_tail(n, acc=0):
    if n == 0:
        return acc  # caso base: restituisce direttamente l'accumulatore
    return somma_tail(n - 1, acc + n)  # chiamata ricorsiva in coda

print("Inserisci un numero intero: ", end='', flush=True)
n=int(input())
print("",n) # serve solamente a visualizzare il numero inserito
            # (poiché non c'è un terminale in Jupyter)
print("La somma dei primi ",n," numeri è: ",somma_tail(n))
```

[4] ✓ 3.3s

```
Inserisci un numero intero:  5
La somma dei primi  5  numeri è:  15
```

Esempio di Ricorsione

Utilizzare la **ricorsione semplice** per risolvere in modo algoritmico il classico problema del **Fattoriale** di un numero naturale (può essere facilmente dimostrato con il *principio di induzione matematica*):

$$n! \begin{cases} 1 & n = 0 \\ n \cdot (n-1)! & \forall n > 0, n \in \mathbb{N} \end{cases}$$

PW-5: Fattoriale

<u>*Problema:*</u>

$$P(n) = n! \begin{cases} 1 & n = 0 \\ n \cdot (n-1)! & \forall n > 0, n \in \mathbb{N} \end{cases}$$

PW-5: Soluzione con PIM

Problema:

$$P(n) = n! \begin{cases} 1 & n = 0 \\ n \cdot (n-1)! & \forall n > 0, n \in \mathbb{N} \end{cases}$$

Soluzione, dimostrazione per induzione :

$P(0) = 1$ è vera

$$P(n) = n! = 1 \cdot 2 \cdot \ \dots \cdot (n-1) \cdot n$$

Per il principio dell'induzione se è vera per $P(n+1)$ allora è vera $\forall n$, per cui:

$$P(n+1) = 1 \cdot 2 \cdot \ \dots \cdot ((n+1)-1) \cdot (n+1)$$
$$= 1 \cdot 2 \cdot \ \dots \cdot n \cdot (n+1) = P(n) \cdot (n+1) = (n+1)!$$

Fattoriale: induzione e ricorsione

Definizione matematica per induzione:

- Caso base: $0! = 1$
- Caso «n»: $n! = n \cdot (n-1)!$

In codice ricorsivo:

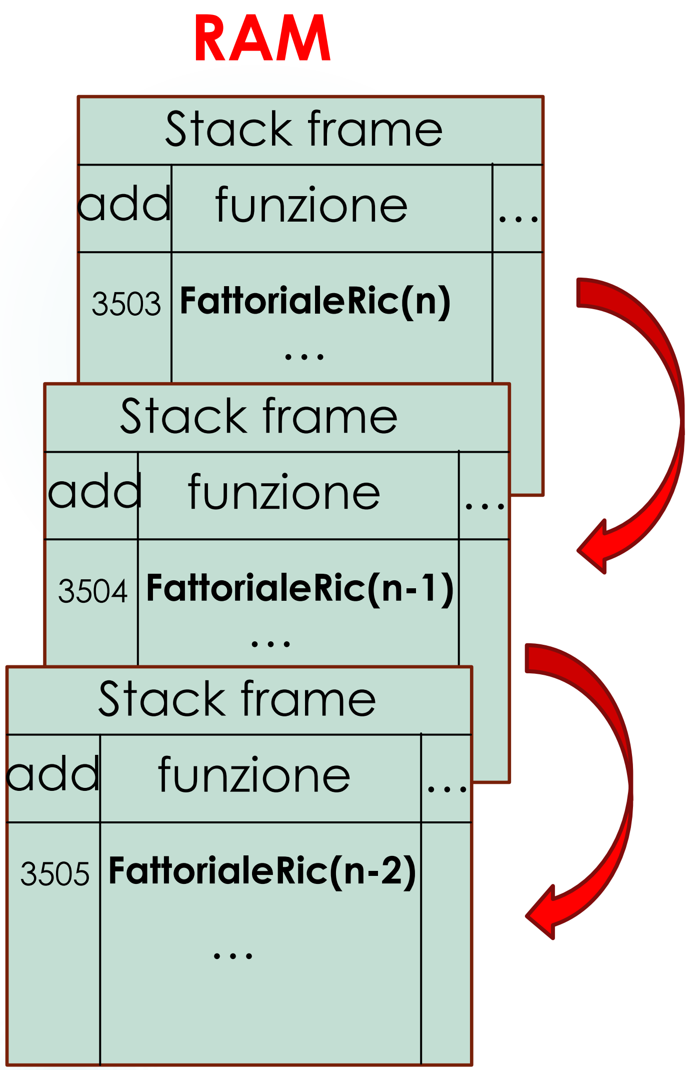
Ricorsione > fatt_semplice.py > ...

```python
# Fattoriale con ricorsione semplice
def fattoriale_ric(n):
    if n == 0:
        return 1
    else:
        return n * fattoriale_ric(n - 1)

#Main
print("è:",fattoriale_ric(int(input("Fattoriale di: "))))
```

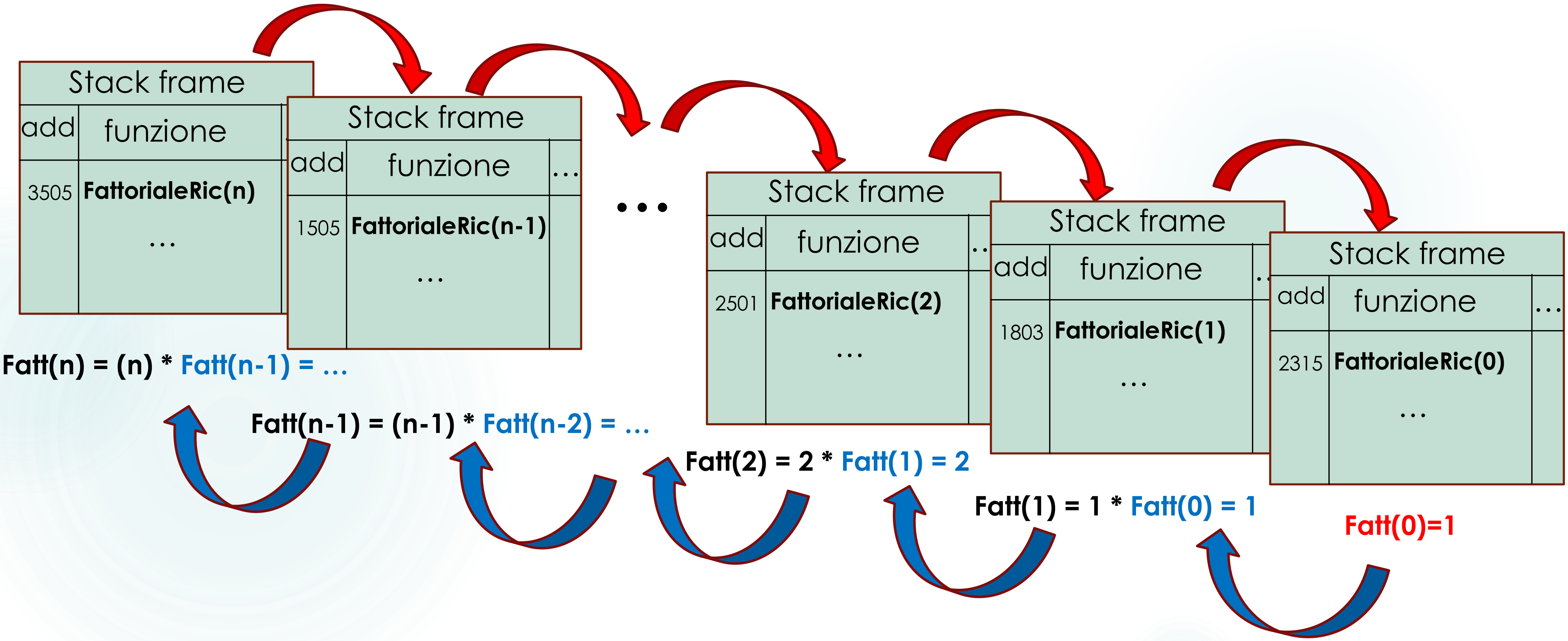
Esempio di Algoritmo Ricorsivo

JAVA

```java
public class FattRic {

    public FattRic() throws IOException{
        int n;

        System.out.print("Calcola il FATTORIALE di: ");
        BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
        String line = input.readLine();
        n = Integer.parseInt(line);
        int result = FattorialeRic(n);
        System.out.print("Il FATTORIALE e': "+result);
    }

    public int FattorialeRic(int x) {
        int fattoriale;

        if (x==0)
          fattoriale = 1;
        else
          fattoriale = x * FattorialeRic(x-1);

        return fattoriale;
    }

    public static void main(String[] args) throws IOException{
        new FattRic();
    }
}
```



Esempio di Ricorsione in Memoria



Fattoriale: ricorsione in coda

Può essere facilmente ottimizzabile dai compilatori poiché riconducibile alla forma iterativa. **Non con l'interprete Python.**

Ricorsione > fattoriale_tail.py > ...

```python
# Fattoriale con ricorsione in coda
def fattoriale_tail(n, acc=1):
    if n == 0:
        return acc
    return fattoriale_tail(n - 1, acc * n)

#Main
print("è:",fattoriale_tail(int(input("Fattoriale di: "))))
```

Fattoriale con iterazione

Versione iterativa:

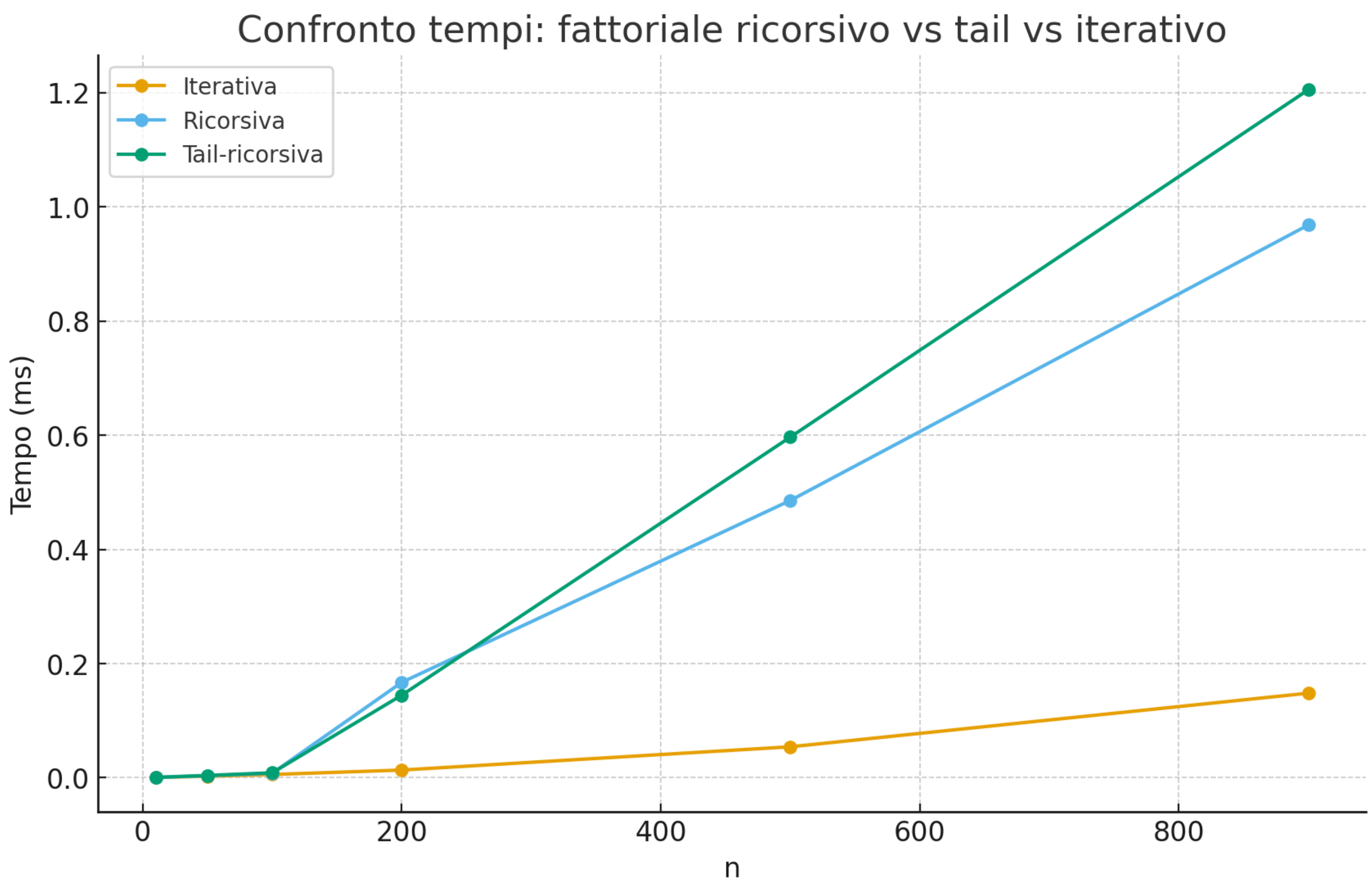
Ricorsione > fatt_iter.py > ...

```python
# Fattoriale con approccio iterativo
def fattoriale_iter(n):
    risultato = 1
    for i in range(1, n + 1):
        #risultato *= i   # questo è equivalente alla riga 5
        risultato = risultato * i
    return risultato

# MAIN
print("è:",fattoriale_iter(int(input("Fattoriale di: "))))
```

Fattoriale: ricorsivo vs iterativo

Aspetto	Ricorsiva	Iterativa
Stile di scrittura	più elegante e vicina alla definizione matematica ($n! = n * (n-1)!$)	più "procedurale", esplicita il ciclo
Uso della memoria	richiede una nuova area di stack per ogni chiamata	usa una sola variabile (risultato)
Prestazioni	più lenta (overhead delle chiamate di funzione)	più veloce (nessuna chiamata ricorsiva)
Rischio di errore	può generare **RecursionError** se n è grande (stack overflow)	nessun rischio di overflow dello stack
Chiarezza logica	esprime meglio il concetto matematico di ricorsione	più chiara per chi ragiona in termini di loop
Facilità di debug	più complessa da seguire (stack di chiamate annidate)	più semplice da tracciare passo per passo

Fattoriale: ricorsivo vs iterativo



Fattoriale: «test»

Lettura dei risultati del grafico

- I **tempi crescono linearmente** con n per tutte e tre (come atteso).
- L'**iterativa** è la più veloce: niente overhead di chiamate di funzione.
- La **ricorsiva in coda** in Python **non è ottimizzata**: costa quanto (o più di) quella classica.
- Le versioni ricorsive sono limitate dal **recursion limit** (≈ 997 stack-frame): oltre quel valore → **RecursionError**.
- **L'iterativa non ha il problema del RecursionError**

Fattoriale: «RecursionError»

Può essere facilmente ottimizzabile dai compilatori poiché riconducibile alla forma iterativa. **Non con l'interprete Python.**