# Logica Matematica
# 3.1 – Le Diverse Forme del Principio di Induzione

Docenti: Alessandro Andretta e Luca Motto Ros

Dipartimento di Matematica
Università di Torino

# Il principio di induzione

In matematica (ed in informatica) è spesso necessario dimostrare che una certa proprietà è vera per tutti i numeri naturali. Vediamo alcuni esempi.

## Esempio 1

Dimostrare che per ogni $n \in \mathbb{N}$,

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}.$$

## Esempio 2

Se $E(n)$ è un'espressione aritmetica che contiene la variabile $n$, l'equazione

$$f(n) = E(n)$$

stabilisce che la funzione $f$ per l'argomento $n$ ha lo stesso valore dell'espressione $E(n)$. Se immaginiamo che la funzione $f$ sia definita ricorsivamente, si può dimostrare per induzione che $f(n) = E(n)$ per ogni valore naturale di $n$, stabilendo così la correttezza della definizione ricorsiva della funzione il cui valore per $n$ è dato da $E(n)$.

## Esempio 3

Consideriamo un frammento di codice della forma:

```
while (b)
S;
```

Se $P$ è una proposizione che esprime una relazione tra i valori delle variabili che compaiono nell'istruzione `S`, allora si può definire un'altra proprietà dei numeri naturali $Q$ ponendo

$Q(n) \leftrightarrow P$ *è vera dopo* $n$ *iterazioni del ciclo while.*

e si può poi cercare di dimostrare che $Q(n)$ è vera per ogni $n \in \mathbb{N}$.

Proprietà di questo tipo sono utilizzate per stabilire che $P$ è una proprietà invariante del ciclo in questione, ovvero un **invariante di ciclo**.

La formulazione più nota del principio di induzione è la seguente.

## Principio di induzione (semplice)

Data una proprietà $P$ dei numeri naturali, se

$$\textit{vale } P(0) \textit{ e per ogni } n \in \mathbb{N} \textit{ vale } P(n) \to P(n+1),$$

allora

$$\textit{per ogni } k \in \mathbb{N} \textit{ vale } P(k),$$

ovvero la proprietà $P$ vale per tutti i numeri naturali.

(Una proprietà dei numeri naturali è una proprietà per la quale abbia senso chiedersi se è vera o falsa per un numero naturale.)

La base dell'induzione è la dimostrazione di $P(0)$, mentre il passo induttivo è la dimostrazione dell'implicazione $P(n) \to P(n+1)$ *per un generico* $n \in \mathbb{N}$, che normalmente si articola nel modo seguente: si assume che $P(n)$ sia vera (questa è detta **ipotesi induttiva**), e si dimostra che allora vale anche $P(n+1)$.

Vediamo un primo esempio di applicazione del principio di induzione.

## Proposizione

$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$ *per ogni* $n \in \mathbb{N}$.

Qui la proprietà $P$ (che vogliamo dimostrare essere vera per tutti i numeri naturali $n$) è l'uguaglianza $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$.

## Dimostrazione.

Dimostriamo la proposizione **per induzione su** $n \in \mathbb{N}$.

La base consiste nel verificare $P(0)$, ovvero che $\sum_{i=0}^{0} i = \frac{0\cdot(0+1)}{2}$.
Svolgendo i calcoli si ha che in effetti

$$\sum_{i=0}^{0} i = 0 = \frac{0 \cdot (0+1)}{2}.$$

(continua...)

## Proposizione

$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$ *per ogni* $n \in \mathbb{N}$.

## Dimostrazione (continua).

Il passo induttivo consiste nel dimostrare che $\forall n\,(P(n) \to P(n+1))$.
Quindi consideriamo un generico $n \in \mathbb{N}$, assumiamo che valga

Ipotesi induttiva $P(n)$: $\quad \sum_{i=0}^{n} i = \frac{n(n+1)}{2}$

e dimostriamo che allora vale anche

Tesi (induttiva) $P(n+1)$: $\quad \sum_{i=0}^{n+1} i = \frac{(n+1)(n+2)}{2}$.

(continua...)

## Proposizione

$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$ *per ogni* $n \in \mathbb{N}$.

## Dimostrazione (continua).

Ipotesi induttiva $P(n)$ $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$

Tesi (induttiva) $P(n+1)$ $\sum_{i=0}^{n+1} i = \frac{(n+1)(n+2)}{2}$

Svolgendo i calcoli si ha

$$\begin{aligned}
\sum_{i=0}^{n+1} i &= 0 + 1 + \ldots + n + (n+1) \\
&= \left(\sum_{i=0}^{n} i\right) + (n+1) \\
&= \frac{n(n+1)}{2} + (n+1) && \text{per l'ipotesi induttiva} \\
&= \frac{n(n+1) + 2(n+1)}{2} \\
&= \frac{(n+1)(n+2)}{2}.
\end{aligned}$$

□

Per dimostrare un'uguaglianza tra due termini è sufficiente mostrare che entrambi sono uguali ad una stessa quantità (che di solito si ottiene "svolgendo i calcoli"). Ad esempio, per dimostrare l'uguaglianza

$$\sum_{i=0}^{n+1} i = \frac{(n+1)(n+2)}{2}$$

si può procedere (sempre usando l'ip. ind. $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$ ) come segue:

$$\sum_{i=0}^{n+1} i = \left(\sum_{i=0}^{n} i\right) + (n+1) = \frac{n(n+1)}{2} + (n+1)$$

$$= \frac{n(n+1)+2(n+1)}{2} = \frac{n^2+n+2n+2}{2} = \frac{n^2+3n+2}{2}.$$

D'altra parte

$$\frac{(n+1)(n+2)}{2} = \frac{n^2+2n+n+2}{2} = \frac{n^2+3n+2}{2}.$$

Poiché sia $\sum_{i=0}^{n+1} i$ che $\frac{(n+1)(n+2)}{2}$ sono uguali a $\frac{n^2+3n+2}{2}$, l'uguaglianza è verificata.

Qualche volta, si deve dimostrare che una certa proprietà $P$ vale non per tutti i numeri naturali, ma solo da un certo $k \in \mathbb{N}$ in poi, ovvero si deve dimostrare che

$$P(n) \textit{ vale per ogni } n \geq k.$$

**Esempio**

Dimostrare che per ogni $n \geq 3$

$$2n + 1 < 2^n.$$

Come si procede in questi casi?

Dal punto di vista teorico, non è necessario introdurre un nuovo principio di induzione per trattare questi casi: infatti

$$P(n) \text{ vale per ogni } n \geq k$$

se e solo se

$$Q(m) \text{ vale per ogni } m \in \mathbb{N},$$

dove $Q(m)$ è l'affermazione $P(m+k)$.

Una dimostrazione per induzione del secondo fatto porta quindi alla dimostrazione del primo.

Dal punto di vista pratico, per dimostrare per induzione (semplice) che

$$P(n) \textit{ vale per ogni } n \geq k$$

si procede come segue:

- si verifica innanzitutto che la proprietà $P$ valga per il primo caso da considerare, ovvero per $k$: in altre parole, il **caso base** in questo caso è $n = k$ e non $n = 0$;
- il **passo induttivo** resta sostanzialmente invariato: si assume che $P$ valga per un generico $n$ (ma con $n \geq k$) e si dimostra che allora $P$ deve valere anche per $n + 1$.

# Esempio

Dimostrare che per ogni $n \geq 3$

$$2n + 1 < 2^n.$$

La proprietà $P(n)$ è $2n + 1 < 2^n$. Procediamo per induzione su $n \geq 3$.

**Caso base** ($n = 3$). Si ha $2 \cdot 3 + 1 = 7$ e $2^3 = 8$, per cui vale $P(3)$.

**Passo induttivo**. Assumiamo che $P(n)$ (ovvero $2n + 1 < 2^n$) valga per un generico $n \geq 3$ e dimostriamo che allora vale anche $P(n + 1)$ (ovvero $2 \cdot (n + 1) + 1 < 2^{n+1}$):

$$\begin{aligned} 2 \cdot (n+1) + 1 &= 2n + 1 + 2 \\ &< 2^n + 2 && \text{(per ipotesi induttiva)} \\ &< 2^n + 2^n && \text{(perché } n \geq 3\text{, da cui } 2^n > 2\text{)} \\ &= 2 \cdot 2^n = 2^{n+1}. \end{aligned}$$

# Definizioni ricorsive di funzioni

Immaginiamo di volere definire una funzione $f : \mathbb{N} \to A$, dove $\mathbb{N}$ è l'insieme dei numeri naturali ed $A$ un insieme qualsiasi. Si può allora utilizzare il seguente schema di ricorsione:

$$\begin{cases} f(0) = a \\ f(n+1) = E(f(n)), \end{cases}$$

dove $a$ è un elemento di $A$ e $E(\dots)$ è una qualche espressione che permette di calcolare il valore di $f(n+1)$ a partire dal valore di $f(n)$.

Una giustificazione intuitiva di questo schema si può ottenere considerando la struttura dei numeri naturali. La funzione $f$ è definita per $0$ perché la prima clausola dello schema ne fornisce il valore $a$. Supponiamo invece che $k$ sia un numero positivo, e che quindi $k = n + 1$ per qualche numero naturale $n$: si può allora immaginare di avere già calcolato il valore di $f(n)$ (la funzione $f$ viene calcolata "dal basso", partendo dall'argomento $0$), e quindi si può calcolare $E(f(n))$ per ottenere il valore di $f(n+1)$, che è proprio $f(k)$.

# Esempio: la funzione quadrato

Si può definire ricorsivamente il quadrato di un numero naturale mediante le clausole:

$$\begin{cases} q(0) = 0 \\ q(n+1) = q(n) + 2 \cdot n + 1 \end{cases}$$

Le clausole della definizione ricorsiva della funzione $q(n)$ consentono di calcolare il valore di questa funzione per un valore arbitrario dell'argomento. Ad esempio

$$\begin{aligned} q(0) &= 0 \\ q(1) &= q(0) + 2 \cdot 0 + 1 = q(0) + 1 = 1 \\ q(2) &= q(1) + 2 \cdot 1 + 1 = 1 + 2 + 1 = 4 \\ q(3) &= q(2) + 2 \cdot 2 + 1 = 4 + 4 + 1 = 9 \\ q(4) &= q(3) + 2 \cdot 3 + 1 = 9 + 6 + 1 = 16 \\ q(5) &= \ldots \end{aligned}$$

$$\begin{cases} q(0) = 0 \\ q(n+1) = q(n) + 2 \cdot n + 1 \end{cases}$$

Vediamo ora che le clausole precedenti definiscono effettivamente la funzione desiderata, dimostrando per induzione che la proprietà $q(n) = n^2$ è vera per ogni valore di $n$.

## Caso base ($n = 0$)

$$\begin{aligned} q(0) &= 0 && \text{(per definizione)} \\ &= 0^2 \end{aligned}$$

*Ipotesi induttiva:* $q(n) = n^2$. *Tesi induttiva:* $q(n+1) = (n+1)^2$.

## Passo indutttivo ($n \to n+1$)

$$\begin{aligned} q(n+1) &= q(n) + 2 \cdot n + 1 && \text{(per definizione)} \\ &= n^2 + 2 \cdot n + 1 && \text{(per ipotesi induttiva)} \\ &= (n+1)^2 && \text{(per proprietà algebriche)} \end{aligned}$$

# Correttezza di programmi (contenenti un ciclo)

Consideriamo il problema di calcolare il quoziente $q$ ed il resto $r$ della divisione tra due numeri interi $X \geq 0$ e $D > 0$.

Per definizione di quoziente e resto, si ha $X = q \cdot D + r$ con $r < D$, da cui $X - q \cdot D = r$. Quindi per determinare $q$ e $d$ bisogna calcolare quante volte si può sottrarre $D$ da $X$ (questo ci dà $q$), e ciò che rimane dopo l'ultima sottrazione è il resto $r < D$.

Questa osservazione ci fornisce un algoritmo per il calcolo della divisione implementabile su un computer: tale algoritmo consiste nel sottrarre ripetutamente $D$ a $X$, aumentando ogni volta di $1$ il valore di $q$ che inizialmente ha valore $0$. Schematicamente, l'algoritmo è il seguente:

1. inizializza le variabili ausiliarie $r$ e $q$ ponendo $r = X$ e $q = 0$;
2. fino a quando $r \geq D$ esegui le seguenti azioni: sottrai $D$ a $r$; aumenta $q$ di $1$;
3. quando $r < D$, dai come output i valori correnti di $q$ (quoziente) e $r$ (resto).

Un programma Java che realizza questo algoritmo è il seguente:

```
class divisione {
public static void main (String[] args) {
   int X, D, q, r;
   X = ? ; // inizializzazione
   D = ? ; // inizializzazione
   q = 0;
   r = X;
   while (r >= D) {
      r = r - D;
      q = q + 1;
   }
   System.out.println ("Il quoziente \‘e: " + q);
   System.out.println ("Il resto \‘e: " + r);
   }
}
```

Come si può dimostrare che il programma precedente è corretto?

Prima di tutto:

La condizione di ingresso del programma, cioè la proprietà che i dati in ingresso $X$ e $D$ devono soddisfare, è che $X \geq 0$ e $D > 0$.

(La seconda proprietà serve ad evitare casi di divisione per $0$.) La condizione di uscita del programma, cioè la proprietà che i dati in uscita $q$ ed $r$ devono soddisfare, è che

$X = q * D + r$, con $r < D$.

Questa proprietà dice proprio che $q$ ed $r$ sono, rispettivamente, il quoziente ed il resto della divisione intera di $X$ per $D$.

La correttezza del programma (qualche volta si parla di questa condizione come di correttezza parziale) asserisce che:

Per ogni dato in ingresso che soddisfa la condizione di ingresso, se il programma termina, allora i dati in uscita soddisfano la condizione di uscita.

Una condizione più esigente di correttezza è quella che si chiama correttezza totale:

Per ogni dato in ingresso che soddisfa la condizione di ingresso, certamente il programma termina e i dati in uscita soddisfano la condizione di uscita.

Per stabilire che un programma soddisfa la specifica vi sono vari modi, ma la tecnica più conveniente consiste nel trovare quello che si chiama un invariante (di ciclo):

Un invariante (di un ciclo) è una proprietà che lega (tutte o alcune del)le variabili coinvolte nel ciclo, e che è vera dopo un numero arbitrario di iterazioni del ciclo. In particolare, è vera anche all'ingresso nel ciclo (cioè dopo $0$ iterazioni).

Ci sono molte proprietà invarianti del ciclo

```
while (r >= D) {
   r = r - D;
   q = q + 1;
}
```

per esempio la proprietà $q \geq 0$. Tuttavia tra tutte le possibili proprietà ce ne sono alcune che sono più interessanti di altre, nel nostro caso:

$$X = q * D + r.$$

In effetti, se il ciclo termina abbiamo che $r < D$ (condizione di uscita del ciclo): quindi se $X = q * D + r$ è effettivamente un invariante, avremo che tale condizione vale anche all'uscita del ciclo (indipendentemente dal numero di cicli eseguiti), e quindi le condizioni di uscita saranno soddisfatte. Avremmo così dimostrato la correttezza parziale del programma che stiamo analizzando.

L'uso di un opportuno invariante di ciclo ci permette quindi di dimostrare che il programma è parzialmente corretto: bisogna però ancora dimostrare che la proprietà

$$X = q * D + r$$

è proprio invariante di ciclo! Questo si può fare per **induzione** sul numero $n$ di iterazioni del ciclo. Verificheremo quindi il caso base ($= 0$ *iterazioni*) e il passo induttivo, ovvero dimostreremo che: *se la proprietà vale dopo* $n$ *iterazioni, allora continua a valere anche dopo* $n + 1$ *iterazioni* (cioè dopo che il ciclo è stato nuovamente eseguito).

### Caso base: $0$ iterazioni

Se il ciclo non è ancora eseguito si ha che $q = 0$ (perché $q$ non viene incrementato) e $r = X$. Allora $X = q * D + r$ perché questo si riduce a dire che $X = 0 * D + X$, che è ovviamente vero.

Supponiamo che valga

**Ipotesi induttiva**

La proprietà $X = q * D + r$ è vera dopo che il ciclo è stato eseguito $n$ volte.

Vogliamo dimostrare che la proprietà resta vera anche dopo la $(n+1)$-esima iterazione (ovvero dopo una nuova esecuzione del ciclo).

Durante questa nuova iterazione vengono modificati i valori di $q$ e di $r$, ottenendo valori

$$q' = q + 1$$
$$r' = r - D$$

dove $q'$ ed $r'$ sono i valori delle variabili $q$ ed $r$ dopo l'esecuzione delle istruzioni

```
r = r - D;
q = q + 1;
```

Vogliamo dimostrare che $X = q' * D + r'$ (che è la nostra *tesi induttiva*).

Eseguendo i calcoli si ottiene

$$\begin{aligned} q' * D + r' &= (q + 1) * D + (r - D) \\ &= q * D + D + r - D \\ &= q * D + r \\ &= X \end{aligned}$$

dove l'ultimo passaggio sfrutta l'ipotesi induttiva.

Per il principio di induzione, si conclude allora che la proprietà

$$X = q * D + r$$

è vera per qualsiasi numero di iterazioni del ciclo, quindi è veramente un invariante di ciclo.

Se vogliamo dimostrare la **correttezza totale** del programma precedente, dobbiamo ora mostrare che se le condizioni di ingresso $X \geq 0$ e $D > 0$ sono soddisfatte, allora il programma termina certamente. Questo vuol dire che il programma non va in loop: il ciclo non viene eseguito all'infinito, ma dopo un numero finito di iterazioni la condizione di uscita dal ciclo $r < D$ viene soddisfatta e il programma termina eseguendo l'ultima istruzione.

Per dimostrare ciò, basta osservare che ad ogni iterazione a $r$ viene sottratto il valore $D$ che, per la condizione di ingresso, è un numero $> 0$: quindi prima o poi deve accadere che $r < D$, che è proprio la condizione di uscita dal ciclo.

## Osservazione

Il fatto che sottraendo a $r$ una quantità non nulla $D$ si debba prima o poi avere $r < D$ segue dal fatto che non ci sono successioni discendenti infinite di numeri naturali, ovvero successioni infinite della forma $n_0 > n_1 > n_1 > n_2 > \ldots$. Vedremo più avanti che questo fatto è equivalente al cosiddetto **principio del minimo**, che a sua volta è equivalente al principio di induzione stesso.

Vediamo un altro esempio della tecnica appena usata per dimostrare la correttezza del programma per la divisione intera, utilizzandola questa volta per sintetizzare un programma per calcolare il quadrato di un numero naturale $N$.

La condizione di ingresso sarà $N \geq 0$, mentre le condizioni di uscita saranno $Y = X * X$ e $X = N$, dove $Y$ è il dato in uscita ed $X$ una variabile ausiliaria utilizzata come contatore.

L'invariante appropriato in questo caso sarà

$$Y = X * X$$

Inizialmente avremo $X = 0$ e $Y = 0$: possiamo quindi già osservare che la proprietà considerata è ovviamente vera in questo caso, ovvero possiamo già verificare il passo base della **dimostrazione per induzione** che la proprietà $Y = X * X$ è un invariante.

Un programma Java che realizza l'algoritmo per il quadrato, basato sul fatto che $(n+1)^2 = n^2 + 2 * n + 1$, è il seguente:

```
class quadrato {
   public static void main (String[] args) {
      int N, X, Y;
      N = ? ; // inizializzazione
      X = 0;
      Y = 0;
      while (X < N) {
         Y = Y + 2 * X + 1;
         X = X + 1;
      }
      System.out.println ("Quadrato = " + Y);
   }
}
```

Vogliamo dimostrare (per **induzione** sul numero di iterazioni del ciclo) che $Y = X * X$ è un invariante per il ciclo del programma precedente.

Abbiamo già verificato il passo base, che si riduce a osservare che è vero che $0 = 0 * 0$.

Per quanto riguarda il passo induttivo, l'*ipotesi induttiva* è

$Y = X * X$ *è vero dopo l'*$n$*-esima iterazione.*

Bisogna dimostrare la *tesi induttiva*, ovvero che

$$Y = X * X$$

resta vera dopo la $(n + 1)$-esima iterazione.

Se $Y'$ è il valore di $Y$ dopo l'esecuzione dell'istruzione

```
Y = Y + 2 * X + 1
```

mentre $X'$ è il valore di $X$ dopo l'esecuzione dell'istruzione

```
X = X + 1
```

possiamo calcolare

$$\begin{aligned} Y' &= Y + 2 * X + 1 \\ &= (X * X) + 2 * X + 1 && \text{(per ipotesi induttiva)} \\ &= (X + 1) * (X + 1) \\ &= X' * X' \end{aligned}$$

Quindi anche il passo induttivo è verificato. Per il principio di induzione, si conclude che la proprietà $Y = X * X$ è proprio un invariante di ciclo.

Come nell'esempio precedente, l'invariante di ciclo garantisce una delle due condizioni di uscita del programma, mentre l'altra è garantita dalla condizione di uscita del ciclo stesso. Infatti all'inizio si ha $X = 0 \leq N$ e il valore di $X$ viene incrementato di $1$ ad ogni ciclo: quindi la prima volta che la condizione $X < N$ del "while" non è soddisfatta, ovvero all'uscita del ciclo, si deve necessariamente avere $X = N$, che è proprio l'altra condizione di uscita del programma. Questo dimostra la correttezza parziale del programma.

Per dimostrare la correttezza totale del programma bisogna solo più verificare che ogni volta che la condizione di ingresso $N \geq 0$ del programma è verificata il programma effettivamente termina (ovvero si esce dal ciclo). Basta allora osservare che poiché il valore di $N - X$ decresce strettamente ad ogni iterazione, il ciclo deve terminare perché non ci può essere una sequenza infinita decrescente di numeri naturali $k_0 > k_1 > k_2 > \ldots$. Questo conclude la dimostrazione della correttezza totale del programma.

# Il principio del minimo

C'è un altro principio fondamentale per ragionare sui numeri naturali:

## Principio del minimo

Se la proprietà $P$ è vera per qualche numero naturale, allora c'è un minimo numero naturale $n$ per il quale vale la proprietà $P$.

Dire che $n$ è il *minimo* per il quale la proprietà $P$ vuole dire che $P(n)$ ma per ogni $k < n$ si ha $\neg P(k)$.

Il principio del minimo si può riformulare equivalentemente come

Ogni insieme $A \subseteq \mathbb{N}$ *non vuoto* ha un elemento minimo rispetto a $\leq$.

## Importante!

Si dimostra che il principio di induzione e il principio del minimo sono **equivalenti**: da ciascuno dei due principi si può derivare l'altro, quindi in un certo senso sono due formulazioni diverse dello stesso principio.

Vediamo ora un'esempio di applicazione del principio del minimo.

## Proposizione

Ogni numero naturale $\geq 2$ ha una scomposizione in fattori primi.

## Dimostrazione.

Per assurdo, sia $n \geq 2$ tale da non avere una scomposizione in fattori primi. Per il principio del minimo, possiamo supporre che $n$ sia il minimo numero con questa proprietà. Ci sono due casi:

1. $n$ è primo: allora $n$ ha una scomposizione in fattori primi, assurdo.
2. $n$ è composto: sia $n = m \cdot l$, dove $2 \leq m, l < n$. I numeri $m$ e $l$ devono avere una scomposizione in fattori primi, perché $n$ è il minimo che non ce l'ha: quindi anche $n$ ha una scomposizione, che si ottiene componendo in modo opportuno le scomposizioni di $m$ e $l$, assurdo.

In entrambi i casi abbiamo contraddetto l'ipotesi che ci sia un numero naturale che non ha scomposizione in fattori primi, quindi abbiamo dimostrato la proposizione. □

Una conseguenza fondamentale del principio del minimo è la seguente.

La relazione $<$ su $\mathbb{N}$ è ben fondata, ovvero in $\mathbb{N}$ non esiste alcuna successione discendente *infinita* della forma

$$n_0 > n_1 > n_2 > \dots$$

La ben fondatezza di $<$ è in realtà equivalente al principio del minimo.

Infatti, se esistesse una successione discendente $n_0 > n_1 > n_2 > \dots$ l'insieme $A = \{n_0, n_1, n_2, \dots\}$ non avrebbe un minimo elemento, contraddicendo il principio del minimo.
Viceversa, supponiamo che $<$ sia ben fondata ma che per assurdo fallisca il principio del minimo, ovvero che esista $A \subseteq \mathbb{N}$ non vuoto e senza un elemento minimo rispetto a $\leq$. Definiamo per ricorsione la successione $n_0, n_1, n_2, \dots$ di elementi di $A$ come segue. Sia $n_0$ un qualunque elemento di $A$ (c'è almeno una scelta per $n_0$ perché $A \neq \emptyset$). Definiamo $n_{k+1}$ come il più grande elemento nell'insieme $\{n \in A \mid n < n_k\}$ (tale insieme è non vuoto perché altrimenti $n_k$ sarebbe il minimo di $A$). Per costruzione si ha che $n_0 > n_1 > n_2 > \dots$, contraddicendo il fatto che $<$ sia ben fondata.

L'uso del principio del minimo in matematica è frequente. Ad esempio, viene implicitamente utilizzato ogni volta che in qualche dimostrazione si usano espressioni del tipo

*Sia* $n \in \mathbb{N}$ *il minimo tale che...*

Un esempio di ciò è la dimostrazione vista in precedenza del fatto che ogni numero naturale maggiore o uguale a $2$ ammette una scomposizione in fattori primi.

In informatica, l'importanza del principio del minimo risiede nel fatto che la ben fondatezza della relazione $<$ sui numeri naturali può essere utilizzata per dimostrare la terminazione di programmi. Implicitamente questa proprietà era già stata utilizzata, per esempio, nella dimostrazione della correttezza totale del programma per la divisione intera o del programma per il calcolo del quadrato di un numero naturale.

Per dimostrare che un programma contenente un ciclo termina (ovvero che dopo un certo numero di iterazioni del ciclo, che dipenderà dall'input iniziale, si esce dal ciclo) si può procedere come segue.

Si assegna ad ogni configurazione di valori $c$ assunti dalle corrispondenti variabili del programma un numero naturale $T(c)$. Tale scelta va fatta in modo che se eseguendo le istruzioni del ciclo il programma passa da una configurazione $c$ ad una configurazione $c'$, allora $T(c) > T(c')$. In altre parole, deve accadere che il numero $T(c)$ diminuisca ogni volta che viene eseguita una nuova iterazione del ciclo.

Fatto questo, si può immediatamente concludere che il ciclo deve terminare: se così non fosse, si otterrebbe una successione discendente infinita

$$T(c) > T(c') > T(c'') > T(c''') > \dots$$

contraddicendo la ben fondatezza di $<$ su $\mathbb{N}$ (ovvero il principio del minimo).

Consideriamo nuovamente il ciclo

```
while (r >= D) {
   r = r - D;
   q = q + 1;
}
```

Una configurazione $c$ è la sequenza dei valori assunti dalle variabili del programma in un dato istante, ovvero $c = \langle X, D, q, r \rangle$ (si ricordi che, anche se non compare esplicitamente nel ciclo, nel programma c'era anche la variabile `X`). Durante una iterazione del ciclo, si passa da una configurazione $c = \langle X, D, q, r \rangle$ ad una configurazione $c' = \langle X', D', q', r' \rangle$ dove $X' = X$, $D' = D$, $q' = q + 1$ e $r' = r - D$. Poniamo allora $T(c) = r$: in questo modo, nel passare da $c$ a $c'$ si ha

$$T(c) = r > r' = T(c')$$

poiché $D > 0$ (condizione di ingresso del programma). Per il principio del minimo, il fatto che $T(c)$ decresca ad ogni iterazione del ciclo garantisce che per qualunque input iniziale che soddisfi le condizioni di ingresso del programma si esca dal ciclo dopo un numero finito di iterazioni.

Consideriamo ora il ciclo

```
while (X < N) {
    Y = Y + 2 * X + 1;
    X = X + 1;
}
```

Una configurazione del programma è in questo caso una sequenza del tipo $c = \langle N, X, Y \rangle$. Eseguendo le istruzioni del ciclo si passa ad una nuova configurazione $c' = \langle N', X', Y' \rangle$ dove $N' = N$, $X' = X + 1$ e $Y' = Y + 2X + 1$. Poniamo $T(c) = N - X$. Questa scelta garantisce che

$$T(c) = N - X > N - X - 1 = N - (X + 1) = N' - X' = T(c').$$

Il fatto che la quantità $T(c)$ decresca ad ogni iterazione del ciclo garantisce nuovamente, grazie al principio del minimo, che il programma corrispondente termini.

# Il principio di induzione forte

Vi sono altre formulazioni del principio di induzione. Una proprietà $P$ dei numeri naturali è progressiva se per ogni $n \in \mathbb{N}$ si ha che

*se la proprietà $P$ vale per tutti gli $m < n$, allora vale anche per $n$.*

Scriviamo $\mathrm{Prog}(P)$ per indicare che $P$ è una proprietà progressiva.

**Principio di induzione forte**

Se $\mathrm{Prog}(P)$, allora per ogni $k \in \mathbb{N}$ vale $P(k)$.

**Importante!**

Si dimostra che anche il principio di induzione forte è **equivalente** al principio di induzione semplice (e quindi anche al principio del minimo).

Utilizziamo il principio di induzione forte per dare una nuova dimostrazione della seguente

## Proposizione

Ogni numero naturale $\geq 2$ ha una scomposizione in fattori primi.

## Dimostrazione.

È sufficiente dimostrare che la prorietà $P(n)$ data da

$$\text{se } n \geq 2 \text{ allora } n \text{ ha una scomposizione in fattori primi}$$

è progressiva. Sia $n$ un generico numero naturale e assumiamo che $P(m)$ valga per ogni $m < n$. Se $n$ è un numero primo non c'è nulla da dimostrare, la sua scomposizione in fattori primi è $n$ stesso. Se invece $n$ è composto, allora $n = m \cdot l$ con $2 \leq m, l < n$. Poiché $m, l < n$, sia $m$ che $l$ sono scomponibili in fattori primi, quindi anche $n$ lo è (la sua scomposizione è il prodotto delle scomposizioni di $m$ ed $l$). □

# Il principio di induzione strutturale semplice

Il principio di induzione può essere generalizzato come segue:

## Principio di induzione strutturale semplice

Sia $A$ un insieme con una funzione $h\colon A \to \mathbb{N}$ *suriettiva*. Data una proprietà $P$, assumiamo che:

($\star$) $P(a)$ vale per tutti gli $a \in A$ con $h(a) = 0$.

($\star\star$) Per ogni $n \in \mathbb{N}$ si ha che:

> *Se $P(a)$ vale per ogni $a$ con $h(a) = n$, allora $P(a)$ vale per ogni $a$ con $h(a) = n + 1$.*

Allora $P(a)$ vale per ogni $a \in A$.

## Importante!

Si dimostra che il principio di induzione strutturale semplice è **equivalente** al principio di induzione.

Vediamo un'applicazione del principio di induzione semplice.

**Lemma**

Per ogni sequenza di numeri naturali $s = \langle s_1, \ldots, s_n \rangle$ di lunghezza $n \geq 2$, se $s_i < n - 1$ per ogni $1 \leq i \leq n$ allora esistono $i \neq j$ tali che $s_i = s_j$.

Applichiamo il principio induzione strutturale semplice con

- $A$ l'insieme delle sequenze $s = \langle s_1, \ldots, s_n \rangle$ di lunghezza $n \geq 2$ tali che $s_i < n - 1$ per ogni $1 \leq i \leq n$,
- $h \colon A \to \mathbb{N}$ definita da $h(s) = \mathrm{lh}(s) - 2$, e
- $P$ la proprietà

$$P(s) \quad \text{se e solo se} \quad s_i = s_j \text{ per qualche } 1 \leq i < j \leq \mathrm{lh}(s).$$

Vogliamo dimostrare che la proprietà $P$ vale per ogni $s \in A$. Per il principio di induzione strutturale semplice, è sufficiente dimostrare che valgono le condizioni $(\star)$ e $(\star\star)$.

Dimostrariamo $(\star)$. Se $s \in A$ e $h(s) = 0$, allora $s = \langle s_1, s_2 \rangle$ e $s_1, s_2 < \mathrm{lh}(s) - 1 = 2 - 1 = 1$: quindi $s_1 = s_2 = 0$ e perciò vale $P(s)$.

Dimostriamo ora $(\star\star)$. Consideriamo un generico $n \in \mathbb{N}$, sia $s \in A$ con $h(s) = n + 1$ (ovvero $\mathrm{lh}(s) = n + 3$) e supponiamo che $P(s')$ valga per ogni $s' \in A$ con $h(s') = n$. Sia $k$ tale che $s_i \leq s_k$ per ogni $1 \leq i \leq n + 3$.

- Se esiste $l \neq k$ tale che $s_l = s_k$, ponendo $i = l$ e $j = k$ siamo a posto, ovvero vale $P(s)$.
- Altrimenti, $s_i < s_k$ per ogni $i \neq k$: siccome $s_k < \mathrm{lh}(s) - 1 = n + 2$, si ha che $s_i < n + 1$ per ogni $i \neq k$. Sia $s'$ la sequenza ottenuta da $s$ rimuovendo $s_k$, ovvero:

$$\begin{cases} s' = \langle s_2, \ldots, s_{n+3} \rangle & \text{se } k = 1 \\ s' = \langle s_1, \ldots, s_{n+2} \rangle & \text{se } k = n + 3 \\ s' = \langle s_1, \ldots, s_{k-1}, s_{k+1}, \ldots, s_{n_3} \rangle & \text{se } 1 < k < n + 3. \end{cases}$$

Allora $\mathrm{lh}(s') = \mathrm{lh}(s) - 1 = n + 2 \geq 2$ e ogni elemento di $s'$ è minore di $n + 1 = \mathrm{lh}(s') - 1$. Perciò $s' \in A$ e $h(s') = n$. Per la nostra ipotesi vale $P(s')$ e quindi esistono $i, j \neq k$ distinti tali che $s_i = s_j$. Questo mostra che vale anche $P(s)$, come desiderato. □

# Il principio dei cassetti

Abbiamo dimostrato che per ogni sequenza di numeri naturali $s = \langle s_1, \ldots, s_n \rangle$ di lunghezza $n \geq 2$, se $s_i < n - 1$ per ogni $1 \leq i \leq n$ allora esistono $i \neq j$ tali che $s_i = s_j$. Da questo segue immediatamente il

## Principio dei cassetti

Se disponiamo $n$ oggetti in $k$ cassetti e $1 \leq k < n$, allora ci saranno almeno due oggetti che finiranno nello stesso cassetto.

## Dimostrazione.

Numeriamo i cassetti con i numeri da $0$ a $k - 1$, e consideriamo la sequenza $\langle s_1, \ldots, s_n \rangle$ di lunghezza $n$, dove per ogni $1 \leq i \leq n$ definiamo

$$s_i = \text{il numero del cassetto in cui viene posto l'}i\text{-esimo oggetto.}$$

Poiché $1 \leq k < n$ si ha $n \geq 2$. Inoltre, $s_i < k \leq n - 1$. Quindi esistono $i \neq j$ tali che $s_i = s_j$, ovvero l'$i$-esimo oggetto e il $j$-esimo oggetto vengono riposti nello stesso cassetto. □

# Il principio di induzione strutturale forte

Similmente, anche il principio di induzione forte può essere generalizzato.

## Principio di induzione strutturale forte

Sia $A$ un insieme con una funzione $h\colon A \to \mathbb{N}$ *suriettiva*. Data una proprietà $P$, assumiamo che

(†) Per ogni $n \in \mathbb{N}$ si ha che:

> *Se $P(a)$ vale per ogni $a$ con $h(a) < n$, allora $P(a)$ vale per ogni $a$ con $h(a) = n$.*

Allora $P(a)$ vale per ogni $a \in A$.

Vedremo un esempio di applicazione del principio di induzione strutturale forte nella Sezione 4.1 (*Sintassi della logica proposizionale*).

## Importante!

Si dimostra che il principio di induzione strutturale forte è **equivalente** al principio di induzione forte (e quindi anche al principio di induzione).

Quindi questi cinque principi sono tra loro equivalenti (Approfondimenti):

- Principio di induzione (semplice) — Definizione
- Principio del minimo — Definizione
- Principio di induzione forte — Definizione
- Principio di induzione strutturale semplice — Definizione
- Principio di induzione strutturale forte — Definizione

# Approfondimenti

## Il principio di induzione

Se $Q(0)$ e $\forall n \in \mathbb{N}\,(Q(n) \to Q(n+1))$, allora $\forall k \in \mathbb{N}\, Q(k)$.

implica il principio del minimo

Se $\exists m \in \mathbb{N}\, P(m)$, allora $\exists n \in \mathbb{N}\,[P(n) \wedge \forall k < n\, \neg P(k)]$.

### Dimostrazione.

Dimostriamo il contrappositivo del principio del minimo, ovvero

$$\text{Se} \quad \forall n \in \mathbb{N}\,[P(n) \to \exists k < n\, P(k)] \quad \text{allora} \quad \neg\exists m \in \mathbb{N}\, P(m).$$

(Se non esiste un minimo $n$ per cui valga $P$, allora $\neg P(m)$ per ogni $m \in \mathbb{N}$.)

Assumiamo perciò $\forall n \in \mathbb{N}\,[P(n) \to \exists k < n\, P(k)]$, e dimostriamo per induzione che $Q(i)$ è vera per ogni $i \in \mathbb{N}$, dove

$$Q(i) \quad \text{se e solo se} \quad \forall j < i\, \neg P(j)\,:$$

questo concluderà la dimostrazione, perché dato qualunque $m \in \mathbb{N}$, poiché $Q(m+1)$ è vera si avrà, in particolare, $\neg P(m)$.

(continua)

## Dimostrazione (continua).

Assumendo $\forall n \in \mathbb{N}\ [P(n) \to \exists k < n\, P(k)]$, dimostriamo per induzione che $\forall i \in \mathbb{N}\, Q(i)$, dove $Q(i)$ se e solo se $\forall j < i\, \neg P(j)$.

Base: $Q(0)$ è vera perché non c'è nessun numero naturale minore di $0$.

Passo induttivo: Assumiamo l'ipotesi induttiva $Q(i)$, cioè $\forall j < i\, \neg P(j)$, e dimostriamo $Q(i+1)$, cioè $\forall j < i+1\, \neg P(j)$, ovvero che $\neg P(0), \ldots, \neg P(i-1), \neg P(i)$. Sappiamo già che $\neg P(0), \ldots, \neg P(i-1)$ per l'ipotesi induttiva: quindi resta solo da dimostrare che $\neg P(i)$. Se per assurdo $P(i)$ fosse vera, allora $i$ sarebbe il minimo numero naturale che soddisfa $P$: ma questo contraddice la nostra assunzione. Quindi $P(i)$ non può essere vera, ovvero $\neg P(i)$: si ha perciò $\forall j < i+1\, \neg P(j)$, cioè $Q(i+1)$.

Per il principio di induzione, $Q(i)$ è vera per ogni $i \in \mathbb{N}$ e la dimostrazione è completa. □

Viceversa, dimostriamo ora che il principio del minimo

Se $\exists m \in \mathbb{N}\, Q(m)$, allora $\exists n \in \mathbb{N}\, [Q(n) \wedge \forall k < n\, \neg Q(k)]$.

implica il principio di induzione

Se $P(0)$ e $\forall n \in \mathbb{N}\, (P(n) \to P(n+1))$, allora $\forall k \in \mathbb{N}\, P(k)$.

(ovvero che i due principi sono equivalenti).

**Dimostrazione.**

Supponiamo per assurdo che $P(0)$ e $P(n) \to P(n+1)$ per ogni $n \in \mathbb{N}$, ma che esista $n \in \mathbb{N}$ per cui $\neg P(n)$. Definiamo $Q$ ponendo $Q(m)$ se e solo se $\neg P(m)$. Per il principio del minimo, esiste allora un più piccolo $n \in \mathbb{N}$ tale che $Q(n)$, ovvero tale che $P(k)$ per tutti i $k < n$ ma $\neg P(n)$. Poiché abbiamo assunto $P(0)$, si deve avere $n > 0$. Ma allora $n - 1 \geq 0$ sarebbe tale che $P(n-1)$, e poiché $P(n-1) \to P(n)$ per ipotesi, si ottiene una contraddizione. □

## Il principio di induzione

Se $Q(0)$ e $\forall n \in \mathbb{N}\,(Q(n) \to Q(n+1))$, allora $\forall k \in \mathbb{N}\, Q(k)$.

implica il principio di induzione forte

Se $\mathrm{Prog}(P)$ (ovvero se $(\forall m < n\, P(m)) \to P(n)$), allora $\forall k \in \mathbb{N}\, P(k)$.

### Dimostrazione.

Assumiamo $\mathrm{Prog}(P)$ e dimostriamo per induzione che $\forall n \in \mathbb{N}\, Q(n)$, dove

$$Q(n) \text{ se e solo se } \forall m < n\, P(m) :$$

questo concluderà la dimostrazione perché dato qualunque $k \in \mathbb{N}$, poiché $Q(k+1)$ è vera si avrà, in particolare, $P(k)$.

(continua)

## Dimostrazione.

Assumiamo $\mathrm{Prog}(P)$ (ovvero che $(\forall m < n\, P(m)) \to P(n)$) e dimostriamo per induzione che $\forall n \in \mathbb{N}\, Q(n)$, dove $Q(n)$ se e solo se $\forall m < n\, P(m)$.

Caso base: $Q(0)$ è vera per lo stesso ragionamento di prima.

Passo induttivo: Supponiamo ora (*ipotesi induttiva*) che valga $Q(n)$ per un generico $n \in \mathbb{N}$, e dimostriamo che anche $Q(n+1)$ è vera. Se $Q(n)$ è vera, allora $\forall m < n\, P(m)$. Poiché $\mathrm{Prog}(P)$, anche $P(n)$ è vera, quindi $\forall m < n+1\, P(m)$, cioè vale $Q(n+1)$.

Per induzione si ha quindi che $\forall n \in \mathbb{N}\, Q(n)$ e la dimostrazione è completa. □

Dimostriamo ora che il principio di induzione forte

Se $\mathrm{Prog}(P)$ (ovvero se $(\forall m < n\, P(m)) \to P(n)$), allora $\forall k \in \mathbb{N}\, P(k)$.

implica il principio di induzione

Se $P(0)$ e $\forall n \in \mathbb{N}\, (P(n) \to P(n+1))$, allora $\forall k \in \mathbb{N}\, P(k)$.

(ovvero che i due principi sono equivalenti).

**Dimostrazione.**

Assumiamo $P(0)$ e $\forall n \in \mathbb{N}\, (P(n) \to P(n+1))$. Per il principio di induzione forte, basta dimostrare che $\mathrm{Prog}(P)$, ovvero che, dato un generico $n \in \mathbb{N}$, si ha $(\forall m < n\, P(m)) \to P(n)$. Distinguiamo due casi. Se $n = 0$, si ha $P(0)$ per ipotesi e quindi l'implicazione è vera. Se invece $n > 0$, dalla premessa $\forall m < n\, P(m)$ si ha in particolare $P(n-1)$, e siccome per ipotesi $P(n-1) \to P(n)$ si ottiene $P(n)$: dunque l'implicazione è verificata anche in questo caso. □

Dimostriamo che il principio di induzione

Se $Q(0)$ e $\forall n \in \mathbb{N}\,(Q(n) \to Q(n+1))$, allora $\forall k \in \mathbb{N}\, Q(k)$.

implica il principio di induzione strutturale semplice

Sia $A$ un insieme con una funzione $h\colon A \to \mathbb{N}$ *suriettiva*. Data una proprietà $P$, assumiamo che:

($\star$) $P(a)$ vale per tutti gli $a \in A$ con $h(a) = 0$.

($\star\star$) Per ogni $n \in \mathbb{N}$ si ha che:

*Se $P(a)$ vale per ogni $a$ con $h(a) = n$, allora $P(a)$ vale per ogni $a$ con $h(a) = n + 1$.*

Allora $P(a)$ vale per ogni $a \in A$.

## Dimostrazione.

Definiamo, per un generico $n \in \mathbb{N}$:

$$Q(n) \quad \text{se e solo se} \quad \forall a \in A \, (h(a) = n \to P(a)) \, .$$

Abbiamo che vale $Q(0)$ per la l'ipotesi $(\star)$ su $P$. Inoltre, per ogni $n \in \mathbb{N}$ si ha che $Q(n) \to Q(n+1)$ per l'ipotesi $(\star\star)$ su $P$. Quindi per induzione $Q(k)$ per ogni $k \in \mathbb{N}$. Sia ora $a \in A$ qualsiasi, e sia $n = h(a)$: poiché vale $Q(n)$, si ha $P(a)$. Quindi $P(a)$ per ogni $a \in A$. □

Viceversa il principio di induzione strutturale semplice

Sia $A$ un insieme con una funzione $h\colon A \to \mathbb{N}$ *suriettiva*. Data una proprietà $P$, assumiamo che:

($\star$) $P(a)$ vale per tutti gli $a \in A$ con $h(a) = 0$.

($\star\star$) Per ogni $n \in \mathbb{N}$ si ha che:

*Se $P(a)$ vale per ogni $a$ con $h(a) = n$, allora $P(a)$ vale per ogni $a$ con $h(a) = n + 1$.*

Allora $P(a)$ vale per ogni $a \in A$.

implica il principio di induzione

Se $Q(0)$ e $\forall n \in \mathbb{N}\,(Q(n) \to Q(n+1))$, allora $\forall k \in \mathbb{N}\, Q(k)$.

Infatti, ponendo $A = \mathbb{N}$ e $h(n) = n$ per ogni $n \in \mathbb{N}$, si vede immediatamente che il principio di induzione è un caso particolare del principio di induzione strutturale semplice: quindi questi due principi sono in realtà tra loro equivalenti (ed equivalenti al principio del minimo e al principio di induzione forte).

Dimostriamo che il principio di induzione forte

Se $\mathrm{Prog}(Q)$ (ovvero se $(\forall m < n\, Q(m)) \to Q(n)$), allora $\forall n \in \mathbb{N}\, Q(n)$.

implica il principio di induzione strutturale forte

Sia $A$ un insieme con una funzione $h\colon A \to \mathbb{N}$ *suriettiva*. Data una proprietà $P$, assumiamo che

($\dagger$) Per ogni $n \in \mathbb{N}$ si ha che:

*Se $P(a)$ vale per ogni $a$ con $h(a) < n$, allora $P(a)$ vale per ogni $a$ con $h(a) = n$.*

Allora $P(a)$ vale per ogni $a \in A$.

## Dimostrazione.

Definiamo, per un generico $n \in \mathbb{N}$:

$$Q(n) \quad \text{se e solo se} \quad \forall a \in A \, (h(a) = n \to P(a)) \, .$$

Abbiamo $\mathrm{Prog}(Q)$, perché $Q(m)$ per ogni $m < n$ implica che $Q(n)$ è vera, per l'ipotesi (†) su $P$. Quindi per induzione forte $Q(k)$ per ogni $k \in \mathbb{N}$. Sia ora $a \in A$ qualsiasi, e sia $n = h(a)$: poiché vale $Q(n)$, si ha $P(a)$. Quindi $P(a)$ per ogni $a \in A$. □

Viceversa il principio di induzione strutturale forte

Sia $A$ un insieme con una funzione $h\colon A \to \mathbb{N}$ *suriettiva*. Data una proprietà $P$, assumiamo che

(†) Per ogni $n \in \mathbb{N}$ si ha che:

*Se $P(a)$ vale per ogni $a$ con $h(a) < n$, allora $P(a)$ vale per ogni $a$ con $h(a) = n$.*

Allora $P(a)$ vale per ogni $a \in A$.

implica il principio di induzione forte

Se $\mathrm{Prog}(P)$ (ovvero se $(\forall m < n\, P(m)) \to P(n)$), allora $\forall n \in \mathbb{N}\, P(n)$.

Infatti, ponendo $A = \mathbb{N}$ e $h(n) = n$ per ogni $n \in \mathbb{N}$, si vede immediatamente che il principio di induzione forte è un caso particolare del principio di induzione strutturale forte: quindi questi due principi sono in realtà tra loro equivalenti (ed equivalenti al principio di induzione semplice, al principio del minimo e al principio di induzione strutturale semplice).

Back