

Lezioni di Matematica Discreta
26 Ottobre 2009 ore 15

Antonino Salibra

NOTAZIONI:

1. $\mathbb{N} = \{1, 2, 3, \dots\}$ è l'insieme dei numeri naturali più grandi di 0.
2. $\mathbb{N}_0 = \{0, 1, 2, 3, \dots\}$ è l'insieme dei numeri naturali.
3. Se n è un numero naturale, allora $[1, n] = \{1, 2, \dots, n\}$.
4. Se X è un insieme, allora $\mathcal{P}(X)$ è l'insieme delle parti di X ovvero l'insieme dei suoi sottoinsiemi.
5. Y^X denota l'insieme delle funzioni da X in Y .
6. Se X è un insieme, allora $|X|$ denota la sua cardinalità.
7. Se A è un alfabeto finito, A^* denota l'insieme delle stringhe su A .
8. Se A è un alfabeto finito, A_n^* denota l'insieme delle stringhe su A di lunghezza n .
9. Se a è un carattere, a^n denota la stringa $aa\dots a$ (n volte).
10. Se R è una relazione di equivalenza su X e $a \in X$, allora $[a]_R$ denota la classe di equivalenza di a .
11. $\binom{n}{k}$ denota il coefficiente binomiale n su k .

1. Introduzione

1.1. Dati semplici

Un tipo di dato è un insieme di elementi o dati con delle operazioni elementari su di essi. Nel seguito distinguiamo tra dati semplici e composti (o strutturati).

- *Algoritmi su numeri naturali e interi.* Esempi di problemi: Calcolare la somma di x e y . Determinare se il naturale x è un numero primo. Scomporre x in fattori primi. I numeri si rappresentano come stringhe sull'alfabeto delle cifre.
- *Algoritmi su stringhe (= sequenze di caratteri).* Le operazioni sulle stringhe sono diverse da quelle sulle sequenze di interi, per esempio. Esempio di algoritmo su stringhe: Determinare se una stringa è palindroma, etc.

1.2. Dati composti

- *Algoritmi su sequenze di dati semplici.* Si rappresentano di solito nei programmi con i vettori (se le sequenze hanno una lunghezza massima). Per esempio, sequenze di stringhe o numeri. Algoritmi tipici sulle sequenze sono quelli di ordinamento e di ricerca.
- *Algoritmi su grafi.*

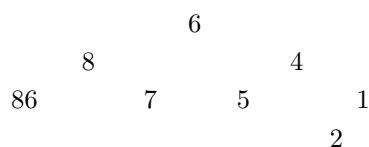
1. *L'orario degli esami* [4, pagina 230]. Consideriamo un corso di laurea con n studenti e r corsi. Immaginiamo che gli esami siano scritti e che tutti gli studenti di un corso facciamo gli esami contemporaneamente. Due esami si possono svolgere lo stesso giorno se non hanno studenti in comune. Qual'è il numero minimo di giorni per svolgere tutti gli esami? Si costruisce un grafo i cui nodi sono etichettati dai nomi dei corsi; un arco connette il corso x al corso y se i due corsi

hanno alcuni studenti in comune (e quindi gli esami non possono svolgersi lo stesso giorno). Il numero cromatico del grafo risultante corrisponde al numero minimo di giorni per svolgere gli esami. (Spiegazione de numero cromatico: coloriamo i nodi del grafo. Due nodi adiacenti, cioè connessi da un arco, devono avere colori diversi. Il numero cromatico e' il numero minimo di colori necessari).

2. Problema matematico: è possibile colorare una carta geografica planare con 4 colori in maniera tale che stati confinanti hanno colori diversi? Problema informatico: Scrivere un programma che prende in input una carta geografica e restituisce la carta colorata con quattro colori [4, pagina 230-1].

• *Algoritmi su alberi.*

1. Ordinamento di sequenze di naturali: leggendo la sequenza da sinistra a destra pongo gli elementi su un albero binario ordinato. Esempio: data la sequenza 6, 4, 8, 1, 5, 86, 7, 2, mi costruisco un albero binario, che ha il primo numero 6 come radice. Gli altri numeri vengono inseriti scendendo nell'albero a destra (a sinistra) se il numero è più piccolo (grande).



2. Il problema del minimo albero ricoprente di un grafo [4, Cap. 10, Sezione 5, pag.262]. Esempio: città connesse con cavi telefonici con costo della posa dei cavi.

In conclusione, i dati semplici o composti di un programma sono sempre *strutture finite*. La matematica discreta studia le proprietà matematiche di queste strutture finite che molto spesso sono essenziali alla comprensione del problema informatico. Si cerca di rispondere alla domanda: È vero che i grafi verificano la seguente proprietà? Per esempio, è vero che possiamo colorare un grafo planare con 4 colori?

Esempio di proprietà matematiche utili per scrivere un algoritmo efficiente. Se r, k, n sono numeri interi, allora con la notazione

$$r \equiv k \pmod{n}$$

si intende n divide $r - k$. Questa notazione è stata introdotta da Gauss nel diciannovesimo secolo. Un calcolo in aritmetica modulare si esegue considerando i numeri modulo n . Per esempio se $n = 5$, allora i numeri importanti sono 0, 1, 2, 3, 4. Così, 27 è "uguale" a 2 modulo 5.

Example 1.1. $42 \equiv 22 \pmod{10}$ perché $42 - 22 = 20$ è divisibile per 10. Nel caso $n = 10$, $r \equiv k \pmod{10}$ sse le cifre meno significative di r ed n nella loro rappresentazione posizionale in base 10 coincidono.

L'algoritmo di Miller-Rabin sui numeri naturali probabilmente primi si basa sui seguenti due teoremi, il primo dei quali è dovuto a Fermat.

Theorem 1.1. (*Piccolo Teorema di Fermat*) Se N è primo, allora $r^{N-1} \equiv 1 \pmod{N}$ per ogni intero r primo con N (cioè, il massimo comun divisore $\gcd(N, r)$ di N ed r è uguale ad 1).

Example 1.2.

(i) Verifichiamo che $7^{5-1} = 7^4 \equiv 1 \pmod{5}$. Si ha $7^4 \equiv 2^4 = 16 \equiv 1 \pmod{5}$. Tutti i numeri possono essere considerati a meno di multipli di 5. Quindi 7 e 2 sono uguali modulo 5.

(ii) Calcoliamo $2^{37} \pmod{13}$. $2^{37} = (2^4)^9 \times 2 = 16^9 \times 2 \equiv 3^9 \times 2 = (3^3)^3 \times 2 = 27^3 \times 2 \equiv 1^3 \times 2 = 2 \pmod{13}$, perché $16 \equiv 3 \pmod{13}$ e $27 \equiv 1 \pmod{13}$.

(iii) Applicando il Piccolo Teorema di Fermat al numero primo $N = 13$ ed ad $r = 2$, possiamo calcolare velocemente: $2^{37} = (2^{12})^3 \times 2 = (2^{13-1})^3 \times 2 \equiv 1^3 \times 2 = 2 \pmod{13}$.

Il Piccolo Teorema di Fermat è verificato purtroppo anche da numeri N che non sono primi. Per esempio, $N = 561 = 3 \times 11 \times 17$.

Theorem 1.2. Se N è primo, allora le uniche soluzioni dell'equazione

$$x^2 \equiv 1 \pmod{N}$$

sono $x \equiv \pm 1 \pmod{N}$.

Infatti, se N è primo, allora N divide $x^2 - 1 = (x-1)(x+1)$ allora N divide uno dei due fattori $x-1$ oppure $x+1$. Se N non è primo, la proprietà non vale. Per esempio, se $N = 8$ allora $x = +1, -1, +3, -3$ verificano la congruenza $x^2 \equiv 1 \pmod{8}$.

Algoritmo di Miller-Rabin:

1. Sia dato un input $N > 2$ dispari. Allora $N - 1$ è pari e possiamo scrivere $N = 2^s \cdot d$ con d dispari. (Per esempio, se $N = 13$, $N - 1 = 12 = 2^2 \cdot 3$)
2. Si scelga casualmente un numero $1 < r < N$.
3. Si calcoli il $\gcd(r, N)$ (massimo comun divisore di r ed N).
4. Se $\gcd(r, N) \neq 1$ oppure $r^{N-1} \not\equiv 1 \pmod{N}$

allora N è composto. Fine.

altrimenti [Commento: in questo punto dell'algoritmo $r^{N-1} \equiv 1 \pmod{N}$ cioè

$$r^{2^s \cdot d} = (r^d)^{2^s} \equiv 1 \pmod{N}.$$

Quindi, possiamo calcolare

$$(r^d)^{2^s}, (r^d)^{2^{s-1}}, (r^d)^{2^{s-2}}, \dots, r^d \pmod{N}$$

Per il Teorema 1.2, se N è primo, tutti questi valori sono uguali ad 1 modulo N ,]

for $k = s$ **to** 0 **do**

begin

Se $r^{2^k \cdot d} \not\equiv \pm 1 \pmod{N}$ **allora** N è composto. Fine.

Se $r^{2^k \cdot d} \equiv -1 \pmod{N}$ **allora** N è primo. Fine.

end-begin

N è primo. Fine.

Se l'output dell'algoritmo è " N è composto", allora la risposta è corretta. Se l'output dell'algoritmo è " N è primo" allora la probabilità di errore con una sola esecuzione dell'algoritmo è pari ad $1/4$. Possiamo abbassare la probabilità di errore ripetendo l'algoritmo più volte, scegliendo casualmente numeri r diversi. 561 è il numero più piccolo che viene testato probabilmente primo, ma non lo è in realtà.

Example 1.3.

(i) Eseguiamo l'algoritmo con $N = 13$ ed $r = 2$. Si ha $\gcd(2, 13) = 1$ e $r^{N-1} = 2^{12} = (2^4)^3 \equiv 3^3 = 27 \equiv 1 \pmod{13}$, perché $2^4 = 16 \equiv 3 \pmod{13}$.

$N - 1 = 12 = 2^2 \times 3$. Si calcoli iterativamente

$$2^3, (2^3)^2 = 2^6, (2^6)^2 = 2^{12}$$

modulo 13.

(ii) Eseguiamo l'algoritmo con $N = 19$ ed $r = 2$. Si ha $\gcd(2, 19) = 1$ e $r^{N-1} = 2^{18} = (2^4)^4 \times 2^2 \equiv (-3)^4 \times 2^2 \equiv 3^4 \times 2^2 = (3^2 \times 2) \times (3^2 \times 2) \equiv -1 \times -1 = 1 \pmod{19}$.

Allora il costrutto "for" calcola iterativamente $2^{12}, 2^6, 2^3$.

2. Insiemi

La collezione dei dati in input (output) di un algoritmo è un *insieme*, che in generale è infinito, mentre l'insieme dei dati in input (output) di un programma in un linguaggio di programmazione è finito (la memoria del computer è finita!). L'insieme dei programmi sintatticamente corretti in un linguaggio di programmazione è un insieme infinito.

1. *Definizione di un insieme finito tramite enumerazione dei suoi elementi* [2, Cap.2]. Un insieme è una collezione di elementi. Gli elementi di un insieme finito vengono racchiusi tra parentesi graffe.

Example 2.1. *Ogni numero naturale modulo 5 è congruo ad un elemento dell'insieme $\{0, 1, 2, 3, 4\}$.*

2. *Definizione di un insieme tramite una proprietà* [2, Cap.2]. Esempi di proprietà: essere alto, essere un numero naturale, essere residente a Iesolo, etc. Data una fissata proprietà P ed un elemento x , si hanno esattamente due possibilità: x verifica la proprietà P oppure x non verifica la proprietà P . In matematica non esistono altre possibilità. Per esempio, Salibra non è alto e quindi non verifica la proprietà "essere alto", mentre 5 è un numero naturale, ossia 5 verifica la proprietà "essere un numero naturale". Data una proprietà possiamo definire l'insieme degli elementi che la verificano.

Example 2.2. *Alcuni insiemi definiti da proprietà:*

- $\{n \mid n \text{ è un multiplo di } 11\}$.

Al posto del simbolo " \mid " che significa "tali che" si può usare in modo equivalente il simbolo ":". Per esempio, $\{n : n \text{ è un multiplo di } 11\}$.

Per risparmiare spazio e tempo, è preferibile dare nomi brevi agli insiemi:

- $X_1 = \{n \mid n \text{ è un multiplo di } 11\}$.
- $X_2 = \{n \mid n \text{ è un uomo alto}\}$.
- $X_3 = \{n \mid n \text{ è dispari}\}$.

Una proprietà si riferisce sempre ad una prefissata collezione di elementi. Per esempio, nella definizione dell'insieme precedente la proprietà "essere dispari" divide l'insieme dei numeri naturali in due parti: i numeri naturali che verificano la proprietà: $1, 3, 5, \dots$, e quelli che non la verificano: $2, 4, 6, \dots$.

A volte si utilizzano proprietà che determinano coppie di elementi (oppure triple, etc). Una coppia si scrive racchiudendo i due elementi tra parentesi tonde: $(1, 2)$ costituisce la coppia il cui primo elemento è 1 ed il secondo elemento è 2. In una coppia l'ordine degli elementi è importante. Le due coppie $(1, 2)$ e $(2, 1)$ sono diverse: $(1, 2) \neq (2, 1)$. Ecco alcuni insiemi di coppie ed insiemi di triple:

- $X_4 = \{(x, y) \mid x, y \text{ sono numeri naturali e } x < y\}$.
- $X_5 = \{(x, y) \mid x \text{ è padre di } y\}$.
- $X_6 = \{(x, y, z) \mid x, y, z \text{ sono numeri naturali e } x + y = z\}$.

Se X ed Y sono insiemi, l'insieme delle coppie con primo elemento in X e secondo elemento in Y si indica con $X \times Y$.

Exercise 2.1. *Sia $\{5, 15\} \subseteq \mathbb{N}_0$ un sottoinsieme finito dei numeri naturali. Definire una proprietà P tale che $\{5, 15\} = \{x : x \text{ soddisfa } P\}$.*

Soluzione:

$$P(x) \equiv (x = 5) \text{ oppure } (x = 15)$$

Un'altra possibilità è

$$P(x) \equiv (x \leq 15) \text{ e } (5 \text{ divide } x) \text{ ed inoltre } (x \neq 0, 10).$$

La relazione di appartenenza. Per esprimere l'appartenenza o meno di un elemento x ad un insieme A si usa il simbolo \in . Si scrive $x \in A$ se x è un elemento dell'insieme (appartiene all'insieme) A , mentre si scrive $x \notin A$ se x non è un elemento dell'insieme A . Quindi l'appartenenza è una relazione tra elementi ed insiemi.

Example 2.3.

- $3 \in \{0, 1, 2, 3, 4\}$ mentre $5 \notin \{0, 1, 2, 3, 4\}$.

Si considerino gli insiemi X_1, X_2, \dots, X_6 dell' Esempio 2.2. Allora si ha:

- $121 \in X_1$ mentre $5 \notin X_1$.

- Antonino Salibra $\notin X_2$.

- $1 \in X_3$, ma $2 \notin X_3$.

- $(2, 5) \in X_4$, ma $(4, 4) \notin X_4$.

- $(Luigi Salibra, Antonino Salibra) \in X_5$.

- $(2, 3, 5) \in X_6$, ma $(4, 4, 7) \notin X_6$.

La relazione di uguaglianza. Due insiemi sono uguali se hanno gli stessi elementi. Nota che l'ordine degli elementi in un insieme non è importante. Per esempio,

$$\{0, 1, 2, 3, 4\} = \{3, 1, 4, 2, 0\}.$$

La relazione di contenuto. Per esprimere se gli elementi di un insieme sono anche elementi di un altro insieme si utilizza il simbolo \subseteq . Si scrive $A \subseteq B$ se l'insieme A è contenuto nell'insieme B mentre si scrive $A \not\subseteq B$ se A non è contenuto in B . Quindi la relazione di *contenuto* è una relazione tra coppie di insiemi. Esempio: $\{2, 4\} \subseteq \{0, 1, 2, 3, 4\}$ mentre $\{2, 7\} \not\subseteq \{0, 1, 2, 3, 4\}$.

3. La relazione d'ordine \subseteq di contenuto o uguale. L'algebra Booleana dei sottoinsiemi di un insieme X : unione \cup , intersezione \cap , complementazione $X - A$ di un sottoinsieme $A \subseteq X$, insieme vuoto \emptyset e insieme massimo X . L'insieme $\mathcal{P}(X)$ delle parti di X . Le proprietà delle operazioni Booleane [2, Cap.2]:

- Idempotenza: $A \cap A = A$; $A \cup A = A$
- Proprietà associativa: $(A \cap B) \cap C = A \cap (B \cap C)$; $(A \cup B) \cup C = A \cup (B \cup C)$
- Proprietà commutativa: $A \cap B = B \cap A$; $A \cup B = B \cup A$
- Assorbimento: $A \cup (A \cap B) = A$; $A \cap (A \cup B) = A$
- Distributività: $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$; $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
- Complementazione: $A \cup (X - A) = X$; $A \cap (X - A) = \emptyset$

Come provare che l'enunciato "Per tutti i sottoinsiemi A, B, C di X , $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ " è vero?

Example 2.4. (Prima dimostrazione di uguaglianza di due espressioni insiemistiche) Sia X un insieme. Dimostrare che per tutti i sottoinsiemi A, B di X vale la seguente proprietà: $A \cup (A \cap B) = A$.

Soluzione: L'uguaglianza delle due espressioni insiemistiche si dimostra provando che $A \cup (A \cap B) \subseteq A$ e $A \subseteq A \cup (A \cap B)$. Cominciamo con la prima disuguaglianza:

$A \cup (A \cap B) \subseteq A$: Se $x \in A \cup (A \cap B)$ allora $x \in A$ oppure $x \in A \cap B$, per definizione dell'operatore insiemistico di unione \cup . In entrambi i casi $x \in A$.

$A \subseteq A \cup (A \cap B)$: Se $x \in A$ allora x appartiene ad ogni sovrainsieme di A , in particolare all'insieme $A \cup (A \cap B)$.

Example 2.5. (Seconda dimostrazione di uguaglianza di due espressioni insiemistiche) Sia X un insieme. Dimostrare che per tutti i sottoinsiemi A, B, C di X vale la proprietà distributiva: $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

Soluzione: Proviamo prima che $A \cup (B \cap C) \subseteq (A \cup B) \cap (A \cup C)$. Sia $x \in A \cup (B \cap C)$. Allora abbiamo due casi esaustivi: $x \in A$ oppure $x \in B \cap C$.

- Se $x \in A$ allora x appartiene ad ogni sovrainsieme di A , in particolare $x \in A \cup B$ e $x \in A \cup C$, da cui $x \in (A \cup B) \cap (A \cup C)$.
- Se $x \notin A$ allora $x \in B \cap C$ (stante l'ipotesi $x \in A \cup (B \cap C)$), che implica $x \in B$ e $x \in C$. Da $x \in B$ segue che $x \in A \cup B$, mentre da $x \in C$ segue che $x \in A \cup C$. In conclusione, $x \in (A \cup B) \cap (A \cup C)$.

Proviamo ora il viceversa: $(A \cup B) \cap (A \cup C) \subseteq A \cup (B \cap C)$. Sia $x \in (A \cup B) \cap (A \cup C)$. Allora abbiamo $x \in A \cup B$ e $x \in A \cup C$. Analizziamo due casi.

- Se $x \in A$ allora $x \in A \cup (B \cap C)$ che è un sovrainsieme di A .
- Se $x \notin A$ allora da $x \in A \cup B$ segue $x \in B$, mentre da $x \in A \cup C$ segue $x \in C$. In conclusione, $x \in B \cap C$, e quindi $x \in A \cup (B \cap C)$.

3. Logica: enunciati e loro dimostrazione

1. Enunciati semplici. Esempi: [2, Ex.1.2.1]; $\{3, 5\} \subseteq \{x \in \mathbb{N}_0 : x \text{ è dispari}\}$; [2, Ex.1.2.5,6].

2. Enunciati composti [2, Sezione 1.3]. Connettivi logici:

$$\wedge \text{ (AND); } \vee \text{ (OR); } \neg \text{ (NOT); } \rightarrow \text{ (IF THEN); } \leftrightarrow \text{ (SE e SOLO SE)}$$

3. Enunciati esistenziali e universali e predicati (o proprietà) [2, Sezione 1.4 e 1.5].

4. Tecniche elementari di dimostrazione [2, Sezione 1.6].

- Prova di enunciato esistenziale $\exists xP(x)$: trovare un esempio.
- Controprova di enunciato universale $\forall xP(x)$: trovare un controesempio [2, Ex.1.6.3].
- Prova di enunciato universale $\forall xP(x)$: si utilizzano le proprietà generali delle variabili quantificate; esempio: [2, Ex. 1.6.2].
- Controprova di enunciato esistenziale $\exists xP(x)$: si prova che $P(x)$ è falso utilizzando le proprietà generali della variabile x .
- Altre tecniche di dimostrazione: dimostrazione per contraddizione [2, Sezione 1.6, Theorem]; dimostrazione per contropositiva [2, Sezione 3.5, Example].
- Prove di uguaglianza di espressioni numeriche e di espressioni insiemistiche.

5. Tavole di verità: and, or, not [2, Sezione 3.1]; equivalenza logica ed il connettivo if-and-only-if [2, Sezione 3.2]; if-then [2, Sezione 3.2]. Quantificatori esistenziali ed universali [2, Sezione 3.6].

Precedenze tra operatori logici:

Precedenze: To save parenthesis, we define the following priorities between the logical symbols:

1 : \neg, \forall, \exists

2 : \wedge, \vee

3 : \rightarrow

Then $\forall xA \wedge B \rightarrow C \vee \neg D$ corresponds to the formula $((\forall xA) \wedge B) \rightarrow (C \vee (\neg D))$, where A, B, C, D are arbitrary formulas.

3.1. Esercizi di Formalizzazione

In questa sezione presentiamo alcuni esercizi di formalizzazione.

3.1.1. Primo Esempio

Supponiamo di avere il predicato binario Axy (x ammira y), il predicato unario Px (x è un professore) ed una costante m (che interpretiamo come Miriam). Nota che Axy e Px sono abbreviazioni per risparmiare parentesi. E' un modo compatto di scrivere $A(x, y)$ e $P(x)$.

1. *Miriam ammira ogni professore:*

$$\forall x(Px \rightarrow Amx)$$

La traduzione letterale in linguaggio naturale del precedente enunciato formale è:

Per ogni x , se x è un professore allora Miriam ammira x .

In generale, "Ogni P..." "Tutti i P..." si traducono così:

$$\forall x(Px \rightarrow \dots)$$

2. *Alcuni professori ammirano Miriam:*

$$\exists x(Px \wedge Amx)$$

La traduzione letterale in linguaggio naturale del precedente enunciato formale è:

Esiste un x tale che x è un professore e x ammira Miriam.

3. *Miriam ammira solo i professori:*

$$\forall x(Px \rightarrow Amx) \wedge \forall x(\neg Px \rightarrow \neg Amx)$$

Si può portare il quantificatore universale fuori

$$\forall x((Px \rightarrow Amx) \wedge (\neg Px \rightarrow \neg Amx))$$

perché vale la seguente regola logica:

$$\forall x(Qx \wedge Rx) \leftrightarrow (\forall xQx \wedge \forall xRx)$$

4. *Un solo professore ammira Miriam:*

$$\exists x(Px \wedge Amx \wedge \forall z(Pz \wedge Azm \rightarrow z = x))$$

5. *Almeno due professori ammirano Miriam:*

$$\exists xy(Px \wedge Py \wedge (x \neq y) \wedge Amx \wedge Aym)$$

Aggiungiamo altri predicati: $B(x, y)$ (x ama y), $C(x)$ (x è un corso), $S(x)$ (x è uno studente).

7. *Nessuno studente ama ogni corso:*

$$\neg \exists x(Sx \wedge \forall y(Cy \rightarrow Bxy))$$

8. *Nessun corso è amato da tutti gli studenti:*

$$\neg \exists x(Cx \wedge \forall y(Sy \rightarrow Byx))$$

Example 3.1. (*Esempio di formalizzazione*) Formalizziamo il predicato "x è un numero primo". Ricordiamo che un numero naturale è primo se è diverso da 0, 1 e divisibile solo per 1 e se stesso.

$$Px \equiv_{def} x \neq 0 \wedge x \neq 1 \wedge \forall z(\exists k(x = z \times k) \rightarrow z = 1 \vee z = x)$$

3.1.2. Terzo Esempio

Formalizziamo l'enunciato “*Esistono infiniti elementi*” in un linguaggio che ha il simbolo di uguaglianza “ $=$ ” come unico predicato binario.

È importante osservare le seguenti proprietà degli enunciati universali ed esistenziali:

- Un enunciato esistenziale $\exists xRx$ può essere *verificato* se si trova un elemento a nell'universo tale che la formula Ra è vera.
- Un enunciato universale $\forall xRx$ può essere *falsificato* se si trova un elemento a nell'universo tale che la formula Ra è falsa.

Quindi, un enunciato esistenziale è sempre falso in un universo vuoto (cioè con zero elementi), mentre un enunciato universale è sempre vero in un universo vuoto.

- *Esiste almeno un elemento*: $\exists x(x = x)$.
- *Esistono almeno due elementi*: $\exists xy(x \neq y)$
- *Esistono almeno tre elementi*: $\exists xyz(x \neq y \wedge x \neq z \wedge y \neq z)$
- *Esistono almeno n elementi*: $\exists x_1 \dots x_n(x_1 \neq x_2 \wedge \dots \wedge x_i \neq x_j \wedge \dots \wedge x_{n-1} \neq x_n)$ (per $i \neq j$).
- *Esiste al più un elemento* (che è equivalente a *Non esistono almeno due elementi*). Uno degli enunciati (equivalenti) seguenti:

- $\neg \exists xy(x \neq y)$
- $\forall xy(\neg x \neq y)$
- $\forall xy(x = y)$

- *Esistono al più due elementi*. Uno degli enunciati (equivalenti) seguenti:

- $\neg \exists xyz(x \neq y \wedge x \neq z \wedge y \neq z)$
- $\forall xyz \neg(x \neq y \wedge x \neq z \wedge y \neq z)$
- $\forall xyz(\neg x \neq y \vee \neg x \neq z \vee \neg y \neq z)$
- $\forall xyz(x = y \vee x = z \vee y = z)$

- *L'universo è costituito da zero elementi*: $\phi_0 \equiv_{def} \neg \exists x(x = x)$
- *L'universo è costituito da un elemento*: $\phi_1 \equiv_{def} \exists x(x = x) \wedge \forall xy(x = y)$
- *L'universo è costituito da due elementi*: $\phi_2 \equiv_{def} \exists xy(x \neq y) \wedge \forall xyz(x = y \vee x = z \vee y = z)$
- *L'universo è costituito da tre elementi*:
 $\phi_3 \equiv_{def} \exists xyz(x \neq y \wedge x \neq z \wedge y \neq z) \wedge \forall xyz u(x = y \vee x = z \vee x = u \vee y = z \vee y = u \vee z = u)$

Infine, l'enunciato “*Esistono infiniti elementi*” è equivalente alla soddisfacibilità dell'insieme infinito di enunciati $\neg \phi_0, \neg \phi_1, \neg \phi_2, \dots, \neg \phi_n, \dots$

Se vogliamo esprimere “*Esistono infiniti elementi*” con un solo enunciato abbiamo bisogno di avere un linguaggio più ricco. Introduciamo oltre all'uguaglianza un predicato binario B . Successivamente formalizziamo l'enunciato “*B è una funzione iniettiva ma non surgettiva*”.

- *B è una funzione*: $\forall xyz(Bxy \wedge Bxz \rightarrow y = z)$
- *B è una funzione totale*: $\forall xyz(Bxy \wedge Bxz \rightarrow y = z) \wedge \forall x \exists y Bxy$
- *B è una funzione totale iniettiva*: $\forall xyz(Bxy \wedge Bxz \rightarrow y = z) \wedge \forall x \exists y Bxy \wedge \forall xyz(Bxz \wedge Byz \rightarrow x = y)$
- *B(x, y) è una funzione totale iniettiva ma non surgettiva*: $\forall xyz(Bxy \wedge Bxz \rightarrow y = z) \wedge \forall x \exists y Bxy \wedge \forall xyz(Bxz \wedge Byz \rightarrow x = y) \wedge \exists y \forall x \neg Bxy$

4. Numeri naturali

Il tipo di dato più semplice è l'insieme dei numeri naturali con le usuali operazioni aritmetiche di addizione e moltiplicazione. Attenzione: Nel libro [2] l'insieme dei numeri naturali è definito come $\mathbb{N} = \{1, 2, 3, \dots\}$, mentre noi seguiremo l'usuale convenzione che 0 è un naturale. Quindi studieremo le leggi algebriche dell'insieme

$$\mathbb{N}_0 = \{0, 1, 2, \dots\}.$$

4.1. Le leggi dell'algebra [2, Sezione 4.1]

1. . Come provare che l'enunciato " $\forall xy(x + y = y + x)$ " è vero? Si potrebbe verificare la proprietà tramite la tabellina dell'addizione per le cifre da 0 a 9 e poi controllare che l'algoritmo dell'addizione che abbiamo imparato alla scuola elementare verifica la proprietà. Comunque la prova dipende dalla rappresentazione che abbiamo scelto per i numeri naturali. Cosa succede se scegliamo di rappresentare i numeri in un alfabeto binario oppure con la rappresentazione romana? Il concetto di numero esiste a prescindere dalla sua rappresentazione concreta!
2. Leggi dell'algebra come assiomi. $(\mathbb{N}_0, +, \times, 0, 1)$ è un quasi anello integrale:

(a) $(\mathbb{N}_0, +, 0)$ e $(\mathbb{N}_0, \times, 1)$ sono monoidi commutativi, cioè verificano le seguenti leggi:

- *Proprietà associativa della somma*: $\forall xyz\{x + (y + z) = (x + y) + z\}$;
Per non mettere troppe parentesi tonde abbiamo scritto $\forall xyz\{\dots\}$ al posto di $\forall xyz(\dots)$.
Lo faremo spesso.
Inoltre spesso non scriveremo i quantificatori universali \forall ; scriveremo quindi: vale la proprietà associativa della somma, $x + (y + z) = (x + y) + z$, sottintendendo che vale per ogni x, y, z .
- *Proprietà associativa del prodotto*: $x \times (y \times z) = (x \times y) \times z$
- *Proprietà commutativa della somma*: $x + y = y + x$;
- *Proprietà commutativa del prodotto*: $x \times y = y \times x$
- *Proprietà dell'elemento neutro*: $x + 0 = 0 + x = x$; $x \times 1 = 1 \times x = x$.

(b) $x \times 0 = 0 \times x = 0$.

(c) *Distributività della moltiplicazione rispetto alla somma*: $x \times (y + z) = (x \times y) + (x \times z)$,

(d) *Proprietà di cancellazione della somma*: $x + z = y + z$ implica $x = y$, che formalmente si scrive:

$$\forall xyz(x + z = y + z \rightarrow x = y).$$

(e) *Proprietà di cancellazione del prodotto* (questa legge è chiamata anche legge dell'integralità):

$$\forall xyz(z \neq 0 \wedge z \times x = z \times y \rightarrow x = y).$$

Come abbreviazione scriveremo spesso

$$(\forall z \neq 0)(\dots)$$

al posto di

$$\forall z(z \neq 0 \rightarrow \dots)$$

Notazione: Di solito scriveremo xy al posto di $x \times y$ e supporremo che la moltiplicazione lega di più dell'addizione, cioè $xy + z$ sta per $(x \times y) + z$.

3. Le proprietà che definiscono un *ordinamento parziale* \leq sugli elementi di un insieme X :

- (a) *Proprietà riflessiva*: $\forall x(x \leq x)$
- (b) *Proprietà transitiva*: $\forall xyz(x \leq y \wedge y \leq z \rightarrow x \leq z)$
- (c) *Proprietà antisimmetrica*: $\forall xy(x \leq y \wedge y \leq x \rightarrow x = y)$

Abbiamo già visto un esempio di ordine parziale: la relazione di contenuto uguale \subseteq sui sottoinsiemi di un insieme X .

Un *ordinamento parziale stretto* $<$ sugli elementi di un insieme X è definito dalle proprietà transitiva e antisimmetrica e dalla seguente proprietà:

4. *Proprietà irreflessiva*: $\forall x \neg(x < x)$

Possiamo sempre definire un ordinamento parziale stretto a partire da un ordinamento parziale \leq :

$$x < y \text{ sse } x \leq y \wedge x \neq y.$$

Viceversa, possiamo definire un ordinamento parziale da uno stretto:

$$x \leq y \text{ sse } x < y \vee x = y.$$

Un ordine parziale \leq è *totale* se è verificata la seguente ulteriore proprietà:

5. $\forall xy(x \leq y \vee y \leq x)$.

4. *L'ordinamento sui numeri naturali* si definisce come segue:

$$x \leq y \text{ sse } \exists z(x + z = y).$$

L'ordinamento stretto sui numeri naturali si definisce come segue:

$$x < y \text{ sse } x \leq y \wedge x \neq y \text{ sse } \exists z \neq 0(x + z = y).$$

Supporremo che valga la seguente legge:

(*Legge di tricotomia*) Dati i numeri naturali x ed y , soltanto una delle seguenti tre condizioni vale: $x < y$ oppure $y < x$ oppure $x = y$.

Verifichiamo per \leq le proprietà che definiscono un ordinamento parziale:

- Proprietà riflessiva: Da $x + 0 = x$ segue $x \leq x$.
- Proprietà transitiva: sia $x \leq y$ e $y \leq z$. Dalla definizione di \leq segue che esistono k, r tali che $x + k = y$ e $y + r = z$. Allora si ha:

$$x + (k + r) = (x + k) + r = y + r = z,$$

da cui segue $x \leq z$.

- Prima prova della proprietà antisimmetrica: Sia $x \leq y$ e $y \leq x$; allora, esistono k, r tali che $x + k = y$ e $y + r = x$. Con semplici calcoli deriviamo:

$$x + (k + r) = (x + k) + r = y + r = x.$$

Dalla legge di cancellazione della somma e da $x + (k + r) = x + 0$ segue che $k + r = 0$, e quindi $k = r = 0$. In conclusione, $y = x + k = x + 0 = x$.

- Seconda prova della proprietà antisimmetrica: Sia $x \leq y$ e $y \leq x$. Supponiamo per assurdo che $x \neq y$. Allora, dalla definizione di ordinamento stretto segue che $x < y$ e $y < x$. Ma ciò contraddice l'assioma di tricotomia. Quindi supporre $x \neq y$ ha portato ad un assurdo; ne segue che $x = y$ vale.
- L'ordine è totale dalla legge di tricotomia.
- 0 è il minimo, cioè $0 \leq x$ per ogni numero naturale x .

4.2. Il principio di induzione

4.2.1. Il principio di induzione con base 0

Il *Principio di Induzione* [2, Sezioni 4.3, 4.4, 4.5, 4.6, 4.8] nella sua forma più semplice: sia $P(x)$ una proprietà sui numeri naturali.

$$P(0) \wedge \forall x(P(x) \rightarrow P(x+1)) \rightarrow \forall xP(x).$$

In altri termini, per provare $\forall xP(x)$ bisogna:

1. *Base dell'induzione* $x = 0$: Provare che P vale per il numero 0;
2. *Ipotesi di induzione*: Supporre che P valga per x .
3. Utilizzare l'ipotesi d'induzione per provare che P vale anche per $x + 1$.

Notazione: Una sommatoria si scrive con il simbolo Σ . Per esempio,

$$1 + 2 + \dots + n = \sum_{1 \leq k \leq n} k$$

Per esempio, se $n = 3$, allora

$$\sum_{1 \leq k \leq 3} k = 1 + 2 + 3.$$

Example 4.1. *Provare per induzione che*

$$\sum_{0 \leq k \leq x} (2k + 1) = (x + 1)^2.$$

1. Base dell'induzione $x = 0$: *In questo caso* $2 \times 0 + 1 = 1 = (0 + 1)^2$.
2. *Ipotesi di induzione*: *Supponiamo che*

$$\sum_{0 \leq k \leq x} (2k + 1) = (x + 1)^2.$$

3. *Utilizziamo l'ipotesi d'induzione per provare che*

$$\sum_{0 \leq k \leq x+1} (2k + 1) = ((x + 1) + 1)^2.$$

Infatti,

$$\sum_{0 \leq k \leq x+1} (2k + 1) = \left(\sum_{0 \leq k \leq x} 2k + 1 \right) + 2(x + 1) + 1 = (x + 1)^2 + 2(x + 1) + 1 = ((x + 1) + 1)^2.$$

Ne segue dal principio di induzione che l'uguaglianza vale per ogni numero naturale x .

4.2.2. Il principio di induzione con base n_0

Il *Principio di Induzione* con base di induzione il numero naturale n_0 :

$$P(n_0) \wedge \forall x \geq n_0(P(x) \rightarrow P(x+1)) \rightarrow \forall x \geq n_0P(x).$$

In altri termini, per provare $\forall x \geq n_0P(x)$ bisogna:

1. *Base dell'induzione* $x = n_0$: Provare che P vale per il numero n_0 ;
2. *Ipotesi di induzione*: Supporre che P valga per $x \geq n_0$.
3. Utilizzare l'ipotesi d'induzione per provare che P vale anche per $x + 1$.

Example 4.2. *Provare che per ogni numero naturale $x \geq 8$, esistono due numeri interi y, z (y o z possono essere negativi) tali che $x = 3y + 5z$.*

Prova:

1. Base dell'induzione ($x = 8$): $8 = 3 \times 1 + 5 \times 1$;
2. Ipotesi di induzione: *Supponiamo che $x \geq 8$ e che $x = 3y + 5z$.*
3. Allora $x + 1 = 3y + 5z + 1$. Ma $1 = 5 \times 2 - 3 \times 3$, da cui $x + 1 = 3y + 5z + 1 = 3y + 5z + 5 \times 2 - 3 \times 3 = 3(y - 3) + 5(z + 2)$.

Ne segue dal principio di induzione che l'uguaglianza vale per ogni numero naturale x .

Notazione: Con $[x, y]$ intendiamo l'insieme di tutti gli elementi z tali che $x \leq z \leq y$.

4.2.3. Il principio di induzione completa

Il Principio di Induzione Completa:

$$P(n_0) \wedge \forall x \geq n_0 (\forall k \in [n_0, x] P(k) \rightarrow P(x + 1)) \rightarrow \forall x \geq n_0 P(x).$$

1. *Base dell'induzione $x = n_0$:* Provare che P vale per il numero n_0 ;
2. *Ipotesi di induzione:* Supporre che P valga per tutti i numeri naturali nell'intervallo $[n_0, x]$.
3. Utilizzare l'ipotesi d'induzione per provare che P vale anche per $x + 1$.

Example 4.3. *Provare che ogni numero naturale $x \geq 2$ è scomponibile in fattori primi.*

Prova:

1. Base dell'induzione ($x = 2$): 2 è un numero primo;
2. Ipotesi di induzione: *Supponiamo che tutti i numeri nell'intervallo $[2, x]$ siano scomponibili in fattori primi.*
3. *Consideriamo $x + 1$. Si hanno due possibilità: $x + 1$ è primo oppure no. Nel primo caso una scomposizione in fattori primi di $x + 1$ è data dal numero stesso. Nel secondo caso, $x + 1$ non è primo; quindi esistono due altri numeri y, z tali che $x + 1 = yz$ e $y, z \neq 1, x + 1$. Da queste due ultime relazioni segue che y e z appartengono all'intervallo $[2, x]$. Quindi, per ipotesi d'induzione sia y che z sono scomponibili in fattori primi: $y = p_1 \dots p_r$ e $z = q_1 \dots q_s$, con p_i, q_j primi. In conclusione, $x + 1 = yz = p_1 \dots p_r q_1 \dots q_s$ ammette una scomposizione in fattori primi.*

Ne segue dal principio di induzione che ogni numero naturale $x \geq 2$ è scomponibile in fattori primi.

4.3. Dati induttivi

I numeri naturali sono definiti "induttivamente" a partire dal numero 0 applicando l'operazione di successore (+1). In altre parole,

1. 0 è un numero naturale;
2. Se x è un numero naturale allora $x + 1$ è un numero naturale;
3. Nient'altro è numero naturale.

Il principio di induzione trova la sua giustificazione nella precedente definizione.

Altri tipi di dato sono definiti induttivamente a partire da alcuni dati di base utilizzando delle operazioni che "costruiscono" dati più complessi. Di seguito troverete degli esempi.

Example 4.4. (Stringhe sull'alfabeto $\{a, b\}$) *Indichiamo con ϵ la stringa vuota, cioè la stringa senza caratteri. L'insieme $\{a, b\}^*$ delle stringhe di alfabeto $\{a, b\}$ è definito induttivamente come segue:*

1. " ϵ " è una stringa;
2. Se α è una stringa allora la concatenazione αa della stringa α e del carattere a è una stringa;

3. Se α è una stringa allora la concatenazione αb della stringa α e del carattere b è una stringa;
4. Nient'altro è stringa.

Per esempio, la concatenazione di $ababa$ e bbb è la stringa $abababbb$, mentre la concatenazione di $ababa$ e la stringa vuota ϵ è uguale alla stringa $ababa$.

Sia P una proprietà sulle stringhe. Il principio di induzione per il tipo di dato stringa (le variabili α e β variano sull'insieme delle stringhe):

$$P(\epsilon) \wedge \forall \alpha \{P(\alpha) \rightarrow P(\alpha a) \wedge P(\alpha b)\} \rightarrow \forall \alpha P(\alpha).$$

Exercise 4.1. Provare per induzione che, per ogni stringa γ , la lunghezza della stringa $\gamma\gamma$ è un numero pari.

Indichiamo con $l(\gamma)$ la lunghezza di una stringa γ . La lunghezza $l(\gamma a)$ (rispettivamente $l(\gamma b)$) della concatenazione della stringa γ e del carattere a (b) è uguale a $l(\gamma) + 1$.

1. Base dell'induzione: Abbiamo il caso seguente. $\epsilon\epsilon = \epsilon$ ha lunghezza 0.
2. Ipotesi di induzione: Consideriamo una stringa γ , e supponiamo per ipotesi d'induzione che $l(\gamma\gamma)$ sia pari.
3. Consideriamo la stringa γa (Un analogo argomento vale per la stringa γb). Se permutiamo i caratteri presenti in una stringa la sua lunghezza non cambia. Quindi

$$l(\gamma a \gamma a) = l(\gamma \gamma a a) = l(\gamma \gamma) + 2.$$

Per ipotesi d'induzione $l(\gamma\gamma)$ è pari e la somma di due numeri pari è ancora pari.

Ne segue dal principio di induzione che ogni stringa $\gamma\gamma$ ha lunghezza pari.

Example 4.5. (Espressioni aritmetiche) Siano $a, 0, 1, [,], +, *$ dei caratteri.

L'insieme EXP delle espressioni aritmetiche è definito induttivamente come segue:

1. $a, 0$ e 1 sono espressioni;
2. Se E_1 e E_2 sono espressioni allora $[E_1 + E_2]$ e $[E_1 * E_2]$ sono espressioni.
3. Nient'altro è espressione.

Esempio: Le stringhe $[a + 1]$ e $[0 + [a * 0]]$ sono espressioni, mentre le stringhe $[a + 1[$ e $[$ non sono espressioni.

Sia P una proprietà che può essere soddisfatta o meno dalle espressioni. Il principio di induzione per il tipo di dato espressione (le variabili x e y variano sull'insieme delle espressioni):

$$P(a) \wedge P(0) \wedge P(1) \wedge \forall xy \{P(x) \wedge P(y) \rightarrow P([x + y]) \wedge P([x * y])\} \rightarrow \forall x P(x).$$

Exercise 4.2. Provare per induzione che ogni espressione ha un numero pari di parentesi.

Indichiamo con $p(E)$ il numero di parentesi dell'espressione E .

1. Base dell'induzione: L'espressione a ha 0 parentesi (che è pari), così come le espressioni 0 e 1 .
2. Ipotesi di induzione: Supponiamo che le espressioni E_1 e E_2 abbiano un numero pari di parentesi, cioè $p(E_1)$ e $p(E_2)$ sono numeri pari.
3. Consideriamo $[E_1 + E_2]$. Allora si ha: $p([E_1 + E_2]) = p(E_1) + p(E_2) + 2$, che è un numero pari, perché per ipotesi d'induzione $p(E_1)$ e $p(E_2)$ sono numeri pari. Un ragionamento simile funziona per l'espressione $[E_1 * E_2]$.

Example 4.6. (Comandi) Dopo aver definito le espressioni, definiamo induttivamente il tipo dei comandi. Supponiamo che a, b sono le sole variabili a disposizione per eseguire comandi di assegnamento.

L'insieme COM dei comandi è definito induttivamente come segue:

1. Se E è una espressione allora le stringhe “ $a := E$ ” e “ $b := E$ ” sono comandi;
2. Se C_1, C_2 sono comandi ed E_1, E_2 sono espressioni, allora “ $C_1; C_2$ ” e “**while** $E_1 = E_2$ **do** C_1 **od**” sono comandi.
3. Nient’altro è comando.

Ecco un comando sintatticamente corretto: **while** $[a * 0] = 1$ **do** $a := [a * [a * 1]]$; $b := [1 + 1]$ **od**.

Sia P una proprietà che può essere soddisfatta o meno dai comandi. Il principio di induzione per il tipo di dato comando lo scriviamo informalmente (c, c_1, c_2 sono variabili che prendono valori in COM, mentre $E, E_1 E_2$ sono variabili che prendono valori in EXP).

$$\begin{array}{ll}
\text{Da} & \forall E \{P(a := E) \wedge P(b := E)\} \\
e & \forall c_1 c_2 \{P(c_1) \wedge P(c_2) \rightarrow P(c_1; c_2)\} \\
e & \forall E_1 E_2 \forall c \{P(c) \rightarrow P(\mathbf{while} E_1 = E_2 \mathbf{do} c \mathbf{od})\} \\
\text{segue} & \forall c P(c)
\end{array}$$

Se volessimo provare per induzione che una proprietà P vale per tutti i comandi, dovremmo partire dalla base dell’induzione che in questo caso comprende infiniti casi. Infatti possiamo scrivere un numero infinito di comandi di assegnamento $a := E$ al variare di E nell’insieme infinito EXP.

4.4. Funzioni

Un programma definisce una “regola” di trasformazione dei dati in input nei dati in output. Il concetto di funzione in matematica (si veda [2, Sezione 5.1]) formalizza e generalizza questa intuizione.

Definition 4.1. Siano A e B insiemi. Una funzione $f : A \rightarrow B$ (leggi: una funzione f da A in B) è una regola che ad ogni elemento x di A assegna uno ed un sol elemento y di B . L’insieme A è il dominio della funzione f , mentre l’insieme B è il suo codominio.

L’elemento y , che è il risultato dell’applicazione della funzione f ad x , si scrive $y = f(x)$. A volte si scrive f_x al posto di $f(x)$.

Una funzione, oltre che dal suo dominio A e codominio B , è caratterizzata dal suo comportamento “input-output”, che può essere rappresentato dall’insieme seguente:

$$\{(x, y) : (x \in A) \wedge (y \in B) \wedge (y = f(x))\}$$

B^A denota l’insieme delle funzioni da A in B .

Example 4.7. La funzione di elevamento a quadrato ha l’insieme \mathbb{N}_0 come dominio e codominio, ed è definita come segue: $f(x) = x^2$. Se applichiamo f all’elemento $2 \in \mathbb{N}_0$ otteniamo $f(2) = 4$ come risultato.

Example 4.8. (Composizione di funzioni; si veda anche [2, Sezione 5.3]). Consideriamo il seguente comando, dove a è una variabile di tipo nat:

$$\mathbf{while} a = [1 + 1] \mathbf{do} a := [a * a] \mathbf{od}$$

Non è difficile verificare che il comportamento input-output del programma

$$\mathbf{input} a; \mathbf{while} a = [1 + 1] \mathbf{do} a := [a * a] \mathbf{od}; \mathbf{output} a$$

definisce la seguente funzione f da \mathbb{N}_0 in \mathbb{N}_0 :

$$f(x) = \begin{cases} x & \text{if } x \neq 2; \\ 4, & \text{if } x = 2. \end{cases}$$

Si consideri ora il programma:

$$\mathbf{input} a; a := [a * a]; \mathbf{output} a$$

il cui comportamento input-output definisce la seguente funzione g da \mathbb{N}_0 in \mathbb{N}_0 :

$$g(x) = x^2.$$

Qual'è il comportamento input-output del programma

input a ; **while** $a = [1 + 1]$ **do** $a := [a * a]$ **od**; $a := [a * a]$; **output** a

ottenuto componendo il primo comando **while** $a = [1 + 1]$ **do** $a := [a * a]$ **od** ed il secondo comando $a := [a * a]$? La composizione delle due funzioni f e g . La composizione di f e g si scrive $f;g$ con notazione informatica oppure $g \circ f$ con notazione matematica. Essa è definita così:

$$(g \circ f)(x) = g(f(x)).$$

In altre parole, per calcolare $(g \circ f)(x)$ prima applichiamo la funzione f ad x e successivamente applichiamo la funzione g a $f(x)$. Nel nostro esempio,

$$(g \circ f)(x) = \begin{cases} x^2 & \text{if } x \neq 2; \\ 16, & \text{if } x = 2. \end{cases}$$

4.5. Definizione ricorsiva di funzioni

Se un insieme A è definito per induzione, allora possiamo definire “ricorsivamente” funzioni di dominio A . Nel seguito prima di presentare lo schema generale di ricorsione, consideriamo degli esempi.

Example 4.9. Ricordiamo che i numeri naturali sono definiti induttivamente a partire dal numero 0 e dalla funzione successore “+1” (si veda Sezione 4.3). Allora definiamo la funzione 2^x per ricorsione a partire dalla funzione successore e dalla funzione di moltiplicazione:

$$(i) \ 2^0 = 1;$$

$$(ii) \ 2^{x+1} = 2 \times 2^x.$$

Proviamo per induzione che la proprietà “la funzione 2^x è ben definita su x ” vale per ogni numero naturale x .

Base dell'induzione	2^0 è ben definita perché $2^0 = 1$
Ipotesi d'induzione	Supponiamo di aver ben definito 2^x
Allora	2^{x+1} è ben definita perché $2^{x+1} = 2 \times 2^x$
Conclusione	Per il principio d'induzione 2^x è definita per ogni x

Come si calcola ricorsivamente:

$$2^2 = 2^{1+1} \stackrel{(ii)}{=} 2 \times 2^1 = 2 \times 2^{0+1} \stackrel{(ii)}{=} 2 \times (2 \times 2^0) \stackrel{(i)}{=} 2 \times (2 \times 1) = 2 \times 2 = 4$$

Example 4.10. Definiamo il fattoriale di x per ricorsione come segue:

$$(i) \ 0! = 1;$$

$$(ii) \ (x+1)! = (x+1) \times x!.$$

Proviamo per induzione che la proprietà “la funzione fattoriale è ben definita su x ” vale per ogni numero naturale x .

Base dell'induzione	$!$ è ben definita su 0 perché $0! = 1$
Ipotesi d'induzione	Supponiamo di aver ben definito $x!$
Allora	$(x+1)!$ è ben definita perché $(x+1)! = (x+1) \times x!$
Conclusione	Per il principio d'induzione $x!$ è definita per ogni x

Come si calcola ricorsivamente:

$$3! \stackrel{(ii)}{=} 3 \times 2! \stackrel{(ii)}{=} 3 \times (2 \times 1!) \stackrel{(ii)}{=} 3 \times (2 \times (1 \times 0!)) \stackrel{(i)}{=} 3 \times (2 \times (1 \times 1)) = 3 \times (2 \times 1) = 3 \times 2 = 6.$$

Nei prossimi due esempi definiamo ricorsivamente la somma ed il prodotto. Il dominio della funzione somma (oppure del prodotto) è l'insieme delle coppie di numeri naturali.

Example 4.11. Definiamo la somma di x e y per ricorsione a partire dalla funzione successore “+1”. L'induzione è sul secondo argomento y :

$$(i) \quad x + 0 = x;$$

$$(ii) \quad x + (y + 1) = (x + y) + 1.$$

Fissiamo $x \in \mathbb{N}_0$. Allora la proprietà “ $P(y) \equiv$ la funzione $x + y$ è ben definita su y ” dipende soltanto da y . Proviamo per induzione che vale $\forall y P(y)$. Dall'arbitrarietà della scelta di x , seguirà che la funzione $x + y$ sarà ben definita per ogni x, y .

Base dell'induzione	$x + 0$ è ben definita perché uguale a x
Ipotesi d'induzione	Supponiamo di aver ben definito $x + y$
Allora	$x + (y + 1)$ è ben definita perché uguale a $(x + y) + 1$
Conclusione	Per il principio d'induzione $x + y$ è ben definita per ogni y

Come si calcola ricorsivamente:

$$\underline{5 + 3} = (\underline{5 + 2}) + 1 = ((\underline{5 + 1}) + 1) + 1 = (((\underline{5 + 0}) + 1) + 1) + 1 = ((5 + 1) + 1) + 1 = (6 + 1) + 1 = 7 + 1 = 8.$$

Example 4.12. Definiamo il prodotto di x e y per ricorsione a partire dalla funzione successore “+1” e dalla somma. L'induzione è sul secondo argomento y :

$$(i) \quad x \times 0 = 0;$$

$$(ii) \quad x \times (y + 1) = (x \times y) + x.$$

Fissiamo $x \in \mathbb{N}_0$. Proviamo per induzione che la proprietà “ $x \times y$ è ben definita su y ” vale per ogni y .

Base dell'induzione	$x \times 0$ è ben definita perché uguale a 0
Ipotesi d'induzione	Supponiamo di aver ben definito $x \times y$
Allora	$x \times (y + 1)$ è ben definita perché uguale a $(x \times y) + x$
Conclusione	Per il principio d'induzione $x \times y$ è ben definita per ogni y

Come si calcola ricorsivamente:

$$\underline{5 \times 3} = (\underline{5 \times 2}) + 5 = ((\underline{5 \times 1}) + 5) + 5 = (((\underline{5 \times 0}) + 5) + 5) + 5 = ((0 + 5) + 5) + 5 = (5 + 5) + 5 = 10 + 5 = 15.$$

Example 4.13. Definiamo la lunghezza di una stringa per ricorsione. Essa è una funzione dall'insieme $\{a, b\}^*$ nell'insieme \mathbb{N}_0 .

$$(i) \quad \text{len}(\epsilon) = 0; \text{len}(a) = 1; \text{len}(b) = 1$$

$$(ii) \quad \text{len}(\alpha\beta) = \text{len}(\alpha) + \text{len}(\beta).$$

Base dell'induzione	$\text{len}(\epsilon)$ è ben definita perché uguale a 0
Base dell'induzione	$\text{len}(a)$ e $\text{len}(b)$ sono ben definiti perché uguali a 1
Ipotesi d'induzione	Supponiamo di aver ben definito $\text{len}(\alpha)$ e $\text{len}(\beta)$
Allora	$\text{len}(\alpha\beta)$ è ben definita perché uguale a $\text{len}(\alpha) + \text{len}(\beta)$
Conclusione	Per il principio d'induzione $\text{len}(\alpha)$ è ben definita per ogni α

Veniamo ora alla definizione generale dello schema di definizione ricorsiva. Per semplicità ci limiteremo all'insieme dei numeri naturali.

Definition 4.2. Una funzione $g : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ è definita ricorsivamente a partire da una costante c e da una funzione $h : \mathbb{N}_0 \times \mathbb{N}_0 \rightarrow \mathbb{N}_0$ se

$$g(0) = c; \quad g(x + 1) = h(x, g(x)).$$

Per esempio, nel caso della funzione 2^x abbiamo $c = 1$ e $h(x, y) = 2 \times y$. Allora abbiamo:

- (i) $2^0 = c = 1$
- (ii) $2^{x+1} = h(x, 2^x) = 2 \times 2^x$.

Nel caso del fattoriale $c = 1$ e $h(x, y) = (x + 1) \times y$. Allora abbiamo:

- (i) $0! = c = 1$
- (ii) $(x + 1)! = h(x, x!) = (x + 1) \times x!$.

Generalizziamo lo schema di ricorsione con dei parametri in più.

Definition 4.3. Una funzione $f : \mathbb{N}_0^{n+1} \rightarrow \mathbb{N}_0$ è definita ricorsivamente a partire da una funzione $g : \mathbb{N}_0^n \rightarrow \mathbb{N}_0$ e da una funzione $h : \mathbb{N}_0^{n+2} \rightarrow \mathbb{N}_0$ se

$$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n); \quad f(x_1, \dots, x_n, y + 1) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y)).$$

Per esempio, nel caso della funzione $x + y$ abbiamo un parametro x , $g(x) = x$ e $h(x, y, z) = z + 1$. Allora abbiamo:

- (i) $x + 0 = g(x) = x$
- (ii) $x + (y + 1) = h(x, y, x + y) = (x + y) + 1$.

Nel caso del prodotto $x \times y$ abbiamo un parametro x , $g(x) = 0$ e $h(x, y, z) = z + x$. Allora abbiamo:

- (i) $x \times 0 = g(x) = 0$
- (ii) $x \times (y + 1) = h(x, y, x \times y) = (x \times y) + x$.

4.6. Il principio del buon ordinamento

Minimo e Massimo. Il principio del buon ordinamento: ogni sottoinsieme di \mathbb{N}_0 ha minimo elemento [2, Sezione 4.7]

4.7. Il principio d'induzione e prove di correttezza

Il principio di induzione è collegato ai cicli iterativi della programmazione. Spieghiamo il perché con un esempio.

Consideriamo il problema di calcolare il quoziente ed il resto della divisione di $x \geq 0$ per $d > 0$. Essi sono definiti come gli unici numeri q ed r che soddisfano la seguente relazione:

$$x = (q \times d) + r, \quad 0 \leq r < d.$$

L'algoritmo usuale consiste nel sottrarre ripetutamente d a x , aumentando di volta in volta di 1 il valore di q (eventuale quoto) e decrementando di x il valore di r (eventuale resto).

$$q := 0; r := x; \text{while } (r \geq d) \text{ do } q := q + 1; r := r - d \text{ od} \tag{1}$$

L'algoritmo è corretto se proviamo per induzione sul numero di volte che eseguiamo il corpo del "while" che la relazione $x = (q \times d) + r$ è un "invariante", cioè se vale prima dell'esecuzione del "while" allora vale anche dopo l'esecuzione del "while".

Definiamo per induzione le due funzioni $q : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ e $r : \mathbb{N}_0 \rightarrow \mathbb{N}_0 \cup \{\text{errore}\}$:

- (i) $q_0 = 0$;
- (ii) $q_{n+1} = q_n + 1$.
- (i) $r_0 = x$;
- (ii)

$$r_{n+1} = \begin{cases} \text{errore} & \text{if } r_n < d \text{ oppure } r_n = \text{errore}; \\ r_n - d, & \text{if } r_n \geq d. \end{cases}$$

Proviamo per induzione su n che $\forall n (r_n \neq \text{errore} \rightarrow x = q_n d + r_n)$. Ricordiamo che per provare una implicazione

Base dell'induzione: $q_0 d + r_0 = 0 \times d + x = x$, perchè per definizione $r_0 \neq \text{errore}$
 Ipotesi d'induzione: Supponiamo che sia vera $r_n \neq \text{errore} \rightarrow x = q_n d + r_n$
 Allora Proviamo che $r_{n+1} \neq \text{errore} \rightarrow x = q_{n+1} d + r_{n+1}$. Supponiamo che $r_{n+1} \neq \text{errore}$. Allora dalla definizione di r segue che $r_{n+1} = r_n - d$. Quindi si ha:
 $q_{n+1} d + r_{n+1} = (q_n + 1)d + (r_n - d) = q_n d + d + r_n - d = q_n d + r_n = x$
 Quest'ultima uguaglianza si ha per ipotesi d'induzione perchè da $r_{n+1} \neq \text{errore}$ si ha necessariamente $r_n \neq \text{errore}$.
 Conclusione Per il principio d'induzione $\forall n (r_n \neq \text{errore} \rightarrow x = q_n d + r_n)$.

Ritorniamo ora al comando in (1). Dopo aver eseguito i comandi di assegnamento $q := 0; r := x, q_0$ e r_0 sono i valori delle variabili q ed r . Non è difficile provare per induzione su n che q_n e r_n con $r_n < d$ sono i valori delle variabili q ed r dopo aver eseguito il corpo del while per n volte. Quindi, dal fatto che vale $\forall n (r_n \neq \text{errore} \rightarrow x = q_n d + r_n)$, dopo l'esecuzione del "while" avremo che $x = q_n d + r_n$ per un opportuno n , ed inoltre $0 \leq r_n < d$. Ne segue che il comando (1) è corretto.

5. Sequenze e Relazioni

1. Composizione di funzioni (si veda [2, Sezione 5.3]).
2. Funzioni iniettive, surgettive e bigettive. Inversa di una funzione bigettiva [2, Sezione 5.2, 5.4]. Se $f : A \rightarrow B$ è una funzione bigettiva, denotiamo con $f^{-1} : B \rightarrow A$ la sua inversa.
3. Siano A, B insiemi. Una *funzione parziale* da A in B è una funzione $f : C \rightarrow B$ da un sottoinsieme C di A in B . Denoteremo con $f : A \rightarrow B$ una funzione parziale da A in B , e con $\text{dom}(f)$ il dominio di definizione di f . Per esempio, la funzione $f : \text{Pari} \rightarrow \mathbb{N}_0$, dove Pari è l'insieme dei numeri pari, definita da $f(2x) = x$ è una funzione parziale $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ con $\text{dom}(f) = \text{Pari}$.

5.1. Cardinalità

1. La cardinalità (= numero di elementi) di un insieme finito ed infinito [2, Cap. 6]. Se A è un insieme $|A|$ denota la sua cardinalità.
2. Insiemi finiti. Notazione: if n is a natural number, then

$$[1, n] = \{1, 2, \dots, n-1, n\}.$$

- Un insieme A è costituito da un numero finito di elementi sse esiste un numero naturale n ed una corrispondenza bigettiva da A nell'insieme $[1, n]$.
 - If $n > k$, $|A| = n$ e $|B| = k$, allora non esiste alcuna funzione iniettiva da A in B , mentre il viceversa è vero, cioè esistono funzioni iniettive da B in A .
 - If $n > k$, $|A| = n$ e $|B| = k$, allora esistono funzioni surgettive da A in B , mentre il viceversa è non è vero, cioè non esistono funzioni surgettive da B in A .
3. Insiemi infiniti. I seguenti insiemi sono tutti infiniti: \mathbb{N} (insieme dei numeri naturali), \mathbb{Z} (insieme dei numeri interi), \mathbb{Q} (insieme dei numeri razionali), \mathbb{R} (insieme dei numeri reali), \mathbb{P} (insieme dei programmi nel linguaggio di programmazione Pascal).
 - Le seguenti condizioni sono equivalenti per un insieme A :
 - A è infinito;
 - Esiste una funzione bigettiva da A in un suo sottoinsieme proprio.
 - Esiste una funzione iniettiva da A in un suo sottoinsieme proprio.
 - Esiste una funzione surgettiva da un sottoinsieme proprio di A nello stesso A .
 - Due insiemi infiniti A e B hanno la stessa cardinalità se esiste una funzione bigettiva da A in B . È molto più utile la seguente caratterizzazione: Due insiemi infiniti A e B hanno la stessa cardinalità se esiste una funzione iniettiva da A in B ed una funzione iniettiva da B in A .

- Esistono differenti ordini di infinito. Per esempio, la cardinalità dell'insieme dei numeri reali è strettamente più grande di quella dell'insieme dei numeri naturali. A livello intuitivo, l'insieme dei numeri reali ha due ordini di infinito: oltre al fatto che l'insieme dei numeri reali è infinito, il singolo numero reale ha in generale una rappresentazione “infinita” con un numero infinito di cifre dopo la virgola.
 - Esiste una funzione iniettiva da \mathbb{N} in \mathbb{R} perché ogni numero naturale è anche un numero reale. Questo risultato comporta che $|\mathbb{N}| \leq |\mathbb{R}|$.
 - Non esiste una funzione bigettiva da \mathbb{N} in \mathbb{R} (il che comporta $|\mathbb{N}| < |\mathbb{R}|$). Per semplicità consideriamo l'intervallo $[0, 1)$ di tutti i numeri reali tra 0 e 1 con quest'ultimo numero escluso. Supponiamo per assurdo che esista una corrispondenza bigettiva r tra \mathbb{N} e $[0, 1)$. allora possiamo enumerare tutti i reali tra 0 e 1:

$$r_0, r_1, r_2, \dots, r_n, \dots$$

Senza perdita di generalità, possiamo supporre che ciascun numero reale r_n abbia un numero infinito di cifre dopo la virgola

$$r_n = 0, c_0^n c_1^n \dots c_k^n \dots$$

con eventualmente $c_k^n = 0$ per tutti i k sufficientemente grandi. Con un argomento diagonale costruiamo un numero reale q tra 0 e 1 che non è presente nella lista:

$$q = 0, d_0 d_1 \dots d_n \dots,$$

dove l' n -sima cifra d_n di q è uguale a $c_n^n + 1$ modulo 10 (dove c_n^n è l' n -sima cifra del numero r_n). Allora, q differisce da r_0 per la prima cifra dopo la virgola, e in generale differisce da r_n per l' n -sima cifra dopo la virgola. In conclusione, q non è nella lista. Questo contraddice la bigettività della corrispondenza r tra \mathbb{N} e \mathbb{R} .

- *Esistono problemi che non ammettono soluzione algoritmica?* Sarebbe interessante, per esempio, trovare una soluzione algoritmica al problema della fermata ed al problema della correttezza, che descriviamo brevemente qui di seguito.

Sia \mathbb{P}_0 l'insieme dei sottoprogrammi Pascal del tipo

```
Function Pippo(x : N) : N; begin ... Pippo:= ... end;
```

che calcolano funzioni parziali da \mathbb{N}_0 in \mathbb{N}_0 , cioè prendono in input un dato di tipo “nat” e restituiscono in output, nel caso in cui terminino la computazione, un dato di tipo “nat”.

Una funzione parziale $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ è *calcolabile* se esiste un sottoprogramma $q \in \mathbb{P}_0$ il cui comportamento input-output coincide con quello della funzione f : ($x \in \text{dom}(f)$ e $f(x) = y$) se e solo se la chiamata $q(x)$ del sottoprogramma q con parametro attuale x termina la computazione con output y . In tal caso, diremo che il sottoprogramma q calcola la funzione parziale f .

- (Il Problema della Fermata) È possibile determinare in modo effettivo se un “while” in un programma va in ciclo? In modo più formale, esiste un programma che prende in input un sottoprogramma $p \in \mathbb{P}_0$ ed un naturale $y \in \mathbb{N}_0$ e restituisce:
 - * 1 se la chiamata $p(y)$ del sottoprogramma p con parametro attuale y termina la computazione;
 - * 0 se la chiamata $p(y)$ del sottoprogramma p con parametro attuale y NON termina la computazione?

La risposta è NO.

- (Il Problema della Correttezza) Sia $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ una funzione calcolabile. È possibile decidere se un programma $q \in \mathbb{P}_0$ calcola f ? La risposta è NO.

Supponiamo per assurdo che il problema della fermata ammetta una soluzione algoritmica che scriveremo come una “function” in Pascal:

```
Function Fermata(q : P_0, x : N): N; ...; Fermata := ...
```

$Fermata(q, x) = 1$ se q termina la computazione sull'input x ; $Fermata(q, x) = 0$ altrimenti. Consideriamo una enumerazione effettiva dei sottoprogrammi di \mathbb{P}_0 tramite una funzione $p : \mathbb{N}_0 \rightarrow \mathbb{P}_0$ bigettiva e calcolabile:

$$p_0, p_1, \dots, p_n, \dots \quad (2)$$

Consideriamo il seguente sottoprogramma Assurdo.

Function *Assurdo*($x : \mathbb{N}$): \mathbb{N} ;

begin

if $Fermata(p_x, x) = 0$ then *Assurdo*:= 1 else begin while true do $x:=x$ end;

end;

Il sottoprogramma *Assurdo* è uno dei sottoprogrammi p_n , per un certo n . Ma allora otteniamo una contraddizione:

$$Fermata(Assurdo, n) = 1 \text{ sse } Fermata(p_n, n) = 0 \text{ sse } Fermata(Assurdo, n) = 0$$

$$Fermata(Assurdo, n) = 0 \text{ sse } Fermata(p_n, n) = 1 \text{ sse } Fermata(Assurdo, n) = 1$$

In conclusione, il problema della fermata non ammette soluzione algoritmica.

5.2. Stringhe (o parole) e sequenze finite

1. Il prodotto Cartesiano $X \times Y$ di due insiemi X e Y è l'insieme delle coppie (a, b) con $a \in X$ e $b \in Y$. $X \times Y \times Z$ è l'insieme delle triple (a, b, c) con $a \in X$, $b \in Y$ e $c \in Z$, etc. Le triple $(a, b, c) = (a, (b, c))$ come coppie, le coppie come insiemi: $(a, b) = \{a, \{a, b\}\}$. Notazione $X \times X \times \dots \times X$ n -volte si abbrevia X^n (vedi l'inizio di [2, Sezione 10.2]).
2. Sequenze finite su $X = \bigcup_{n \in \mathbb{N}_0} X^n$. L'insieme X^0 ha come unico elemento la sequenza vuota.
3. Stringhe su un alfabeto A come sequenze finite di caratteri. In questo caso (a, b, a, b) si scrive in maniera più compatta come $abab$. Le stringhe vengono di solito indicate con le lettere greche α, β, γ . La stringa vuota è denotata con ϵ . La giustapposizione di due stringhe α e β viene indicata con $\alpha\beta$. Per esempio, la giustapposizione di "casa" e "matta" è la stringa "casamatta".
4. L'insieme delle stringhe su un alfabeto finito A ha la stessa cardinalità dell'insieme dei numeri naturali. Si può dimostrare definendo una funzione iniettiva da \mathbb{N} in A^* ed un'altra sempre iniettiva da A^* in \mathbb{N} . Per esempio, sia $A = \{a, b, c, \dots, z\}$ l'alfabeto finito dei caratteri della lingua italiana (21 caratteri). Definiamo una corrispondenza biunivoca $\rho : A \rightarrow \{0, 1, \dots, 20\}$: $\rho(a) = 0$, $\rho(b) = 1, \dots, \rho(z) = 20$. Definiamo una prima funzione iniettiva $f : A^* \rightarrow \mathbb{N}$:

$$f(a_1 \dots a_k) = \rho(a_1) + \rho(a_2) \times 21 + \rho(a_3) \times 21^2 + \dots + \rho(a_k) \times 21^{k-1}.$$

ed una seconda funzione iniettiva $g : \mathbb{N} \rightarrow A^*$:

$$f(n) = a^n,$$

dove a^n denota la stringa $aa \dots a$ (n volte).

5. Stringhe di lunghezza k su un alfabeto $A =$ funzioni da k in A .
6. Notazione $X \times X \times \dots \times X$ k -volte si abbrevia X^k perché sequenze di lunghezza k su $X =$ funzioni da k in X [2, Sezione 10.4].

5.3. Relazioni

1. Relazione binaria come sottoinsieme del prodotto cartesiano [2, Sezione 10.1]. Funzioni come relazioni binarie.
2. Relazioni d'ordine parziale e totale:
3. Sia (A, \leq) un insieme parzialmente ordinato.

- Relazione di comparabilità: x e y sono *comparabili* se $x \leq y$ o $y \leq x$; x e y sono *incomparabili* se nè $x \leq y$ nè $y \leq x$;
 - Catene: Un sottoinsieme X di A è una *catena* se gli elementi di X sono comparabili a due a due.
 - Anti-catene: Un sottoinsieme X di A è una *anti-catena* se gli elementi di X sono incomparabili a due a due.
4. Ordini ben fondati: un ordinamento parziale è *ben fondato* se non ammette catene discendenti infinite: $a_0 > a_1 > a_2 > \dots > a_n > \dots$. L'ordinamento su \mathbb{N}_0 è ben fondato, mentre sull'insieme \mathbb{Z} degli interi non lo è: $0 > -1 > -2 > \dots$.
5. Esempi:

- *Ordine naturale su \mathbb{N}_0* : $x \leq y$ se esiste $k \in \mathbb{N}_0$ tale che $y = x + k$.
- *Ordine di divisibilità su \mathbb{N}_0* : $x|y$ se x divide y . Questo ordinamento ammette 1 come minimo elemento e 0 come massimo. È un ordinamento ben fondato perché, per ogni x , esiste soltanto un numero finito di divisori di x .
- *Ordine prefisso sulle stringhe*: $\alpha \leq_{pre} \beta$ se esiste una stringa γ tale che $\beta = \alpha\gamma$. L'ordine prefisso è ben fondato.
- *Ordine lessicografico sulle stringhe*: Sia $A = \{a_1, \dots, a_n\}$ un alfabeto finito di caratteri con $a_1 < a_2 < \dots < a_n$. Allora $\alpha \leq_{lex} \beta$ se $\alpha \leq_{pre} \beta$ oppure esistono stringhe γ, δ, η e caratteri a_i e a_j tali che $\alpha = \gamma a_i \delta$ e $\beta = \gamma a_j \eta$ con $a_i < a_j$. Un metodo per enumerare tutte le stringhe su un alfabeto finito, per esempio $A = \{a, b, c\}$ con $a < b < c$, è quello di enumerare prima le stringhe di lunghezza 0 (la sola stringa vuota ϵ), poi quelle di lunghezza 1, poi quelle di lunghezza 2 e così via. Le stringhe di lunghezza n , che sono in numero finito, sono ordinate in ordine lessicografico:

$$\epsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab, aac, aba, abb, abc, \dots$$

- *Ordine "contenuto o uguale" sull'insieme $\mathcal{P}(X)$ delle parti di un insieme X* : Se A e B sono sottoinsiemi di X , $X \subseteq Y$ se ogni elemento di X è anche elemento di Y .
6. Relazioni di equivalenza [2, Sezione 7.2, 7.3, 12.2]. Partizioni (vedi l'inizio di [2, Sezione 12.1]).

Sia R una relazione di equivalenza su A .

- $[a]_R = \{b : aRb\}$ denota la classe di equivalenza di $a \in A$. In alcuni libri la classe di equivalenza $[a]_R$ è denotata con a/R .
- La famiglia di sottoinsiemi $([a]_R : a \in A)$ determina una partizione di A .
- L'insieme $\{[a]_R : a \in A\}$ è chiamato insieme quoziente ed è denotato da A/R .

Sia $P = (X_i : i \in I)$ una partizione di A . Allora la relazione

$$aRb \text{ sse } \exists i \in I. a, b \in X_i$$

è una relazione di equivalenza su A .

6. Combinatoria: i principi del contare

Il tema principale di questo capitolo è lo studio della cardinalità degli insiemi finiti i cui elementi possono essere dati strutturati e non.

6.1. Il principio dell'addizione [2, Sezione 10.1]

Cardinalità dell'unione di insiemi disgiunti. Il principio del cassetto (pigeonhole principle).

6.2. Il principio della moltiplicazione [4, Sezione 1.2]

1. Cardinalità del prodotto cartesiano [2, Sezione 10.2]:

$$|X \times Y| = |X| \times |Y|$$

In particolare,

$$|X \times X| = |X|^2$$

2. Cardinalità delle funzioni da X in Y [2, Sezione 10.4]:

$$|Y^X| = |Y|^{|X|}$$

. Ecco alcune applicazioni di questo risultato:

- $|X \times X| = |X^{\{1,2\}}| = |X|^2$ perché a ciascuna coppia (a, b) con $a, b \in X$ corrisponde in maniera univoca la funzione $f : \{1, 2\} \rightarrow X$ tale che $f(1) = a$ e $f(2) = b$.
- Le stringhe di lunghezza n su un alfabeto finito A corrispondono alle funzioni da $[1, n]$ in A . Se indichiamo con A_n^* l'insieme finito delle stringhe di lunghezza n , abbiamo

$$|A_n^*| = |A^{[1, n]}| = |A|^n$$

- Cardinalità dell'insieme delle parti $\mathcal{P}(X)$ di un insieme X [2, Sezione 10.4][4, Sezione 1.2, pag. 5-6]. L'insieme $\mathcal{P}(X)$ può essere messo in corrispondenza biunivoca con l'insieme delle funzioni da X in $\{0, 1\}$:

$$A \subseteq X \mapsto f_A : X \rightarrow \{0, 1\}, \text{ con } f_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{else} \end{cases}$$

Allora la cardinalità dell'insieme $\mathcal{P}(X)$ è:

$$|\mathcal{P}(X)| = |\{0, 1\}^X| = 2^{|X|}.$$

3. La cardinalità delle funzioni iniettive da X in Y [2, Sezione 10.5] [4, Sezione 1.3: Le iniezioni da D in C] (con $|X| = k$ e $|Y| = n$) è uguale a

$$(n)_k = n(n-1)(n-2)\dots(n-k+1).$$

Ecco un'applicazione di questo risultato:

- Le k -stringhe "iniettive" (cioè senza ripetizioni) su un alfabeto A di n caratteri corrispondono alle funzioni iniettive da \hat{k} in A . In totale abbiamo quindi un numero $(n)_k$ di k -stringhe iniettive su un alfabeto di n caratteri. Per esempio, se $A = \{a, b, c, d\}$ allora abd corrisponde alla funzione iniettiva $f : \{1, 2, 3\} \rightarrow A$ tale che $f(1) = a$, $f(2) = b$ e $f(3) = d$.
4. L'insieme delle funzioni bigettive da X in X [2, Sezione 10.6] [4, Sezione 1.3: Fattoriali] è anche uguale all'insieme delle funzioni iniettive da X in X . Se $|X| = n$ allora abbiamo $n!$ funzioni bigettive. Le funzioni bigettive di un insieme finito vengono chiamate *permutazioni*. Si veda [2, Sezione 10.6] per la rappresentazione delle permutazioni tramite cicli.

6.3. Il principio del pastore o della divisione [4, Sezione 1.4]

La regola del pastore dice: *se vuoi sapere quante pecore ci sono in un gregge conta le zampe e poi dividi per quattro*. Se A, B sono insiemi, abbiamo:

$$|A| = |A \times B| / |B|$$

Nel seguito vedremo come, applicando la regola del pastore, la cardinalità di insiemi finiti ordinati possa dare informazione sulla cardinalità di insiemi finiti non ordinati. L'ordine fa crescere il numero di possibilità ma semplifica i conti potendo utilizzare la moltiplicazione. Quindi, lo slogan è:

Cercare l'ordine per applicare successivamente il principio del pastore!

6.3.1. Le contrapposizioni

Ordine-Disordine:

1. *Primo esempio:*

k -stringhe *iniettive* su un alfabeto di n caratteri (ordine)

versus

k -sottoinsiemi di un insieme di n elementi (disordine)

Esempio: Se “togliamo” l’ordine alla 4-stringa iniettiva $cead$ sull’alfabeto $A = \{a, b, c, d, e\}$ otteniamo il 4-sottoinsieme $\{a, c, d, e\}$ di A .

Ricordiamo che l’insieme delle k -stringhe iniettive può anche essere visto come l’insieme delle funzioni iniettive da \hat{k} in un insieme di n elementi.

La dicotomia “ordine-disordine” corrisponde a quella in informatica “programma sequenziale-programma parallelo” e a quella in fisica “esperimento sequenziale-esperimento parallelo”.

- *Programma Sequenziale (ordine) versus Programma Parallelo (disordine):* sia “Out” un comando di output di un dispositivo che può inviare caratteri in sequenza tramite l’operatore “;” oppure in parallelo tramite l’operatore “||”. Allora il primo comando qui di seguito corrisponde ad inviare in output la stringa $cead$, mentre il secondo ad inviare in output il multinsieme $\{a, c, d, e\}$.
 - Out(c); Out(e); Out(a); Out(d)
 - Out(a)||Out(c)||Out(d)||Out(e).
- *Esperimento Sequenziale (ordine)-Esperimento in Parallelo (disordine)*
 - k estrazioni *successive* di caratteri da una scatola che contiene inizialmente tutti i caratteri.
 - k estrazioni *in parallelo* di caratteri da una scatola che contiene tutti i caratteri.

Come si calcola la cardinalità dei k -sottoinsiemi di un insieme di n elementi a partire dalla cardinalità conosciuta delle k -stringhe iniettive su un alfabeto di n caratteri (= funzioni iniettive da \hat{k} in un insieme di n elementi)? Con il principio del pastore!

Spieghiamo il procedimento con un esempio. Quante sono le 4-stringhe iniettive sull’alfabeto $\{a, b, c, d, e\}$ (per esempio, $acde$, $cead$) che corrispondono al 4-sottoinsieme $\{a, c, d, e\}$? Sono pari al numero delle permutazioni di 4 elementi. Quindi, per ciascun 4-sottoinsieme abbiamo $4!$ 4-stringhe iniettive corrispondenti. In totale abbiamo $(5)_4/4! = 5$ 4-sottoinsiemi.

In generale, abbiamo

$$|\text{Funzioni iniettive da } \hat{k} \text{ in } [1, n]| / |\text{Permutazioni di } \hat{k}| = n(n-1) \dots (n-k+1)/k! = (n)_k/k!$$

Il precedente numero si chiama *coefficiente binomiale* e si indica con $\binom{n}{k}$. La ragione di un tale nome e l’importanza dei coefficienti binomiali verranno spiegati in seguito.

2. *Secondo esempio:*

k -stringhe su un alfabeto di n caratteri (ordine)

versus

k -multinsiemi ad elementi in un alfabeto di n caratteri (disordine)

Un multinsieme è per definizione una collezione di oggetti ciascuno con una propria molteplicità. Per esempio, nel multinsieme $\{a, a, a, b, b, c, c, c\}$ gli elementi a e c occorrono con molteplicità 3 mentre l’elemento b con molteplicità 2. Se “togliamo” l’ordine alla stringa $abaabccc$ otteniamo il multinsieme $\{a, a, a, b, b, c, c, c\}$.

- *Programma Sequenziale (ordine) versus Programma Parallelo (disordine):*
 - Out(a); Out(b); Out(a); Out(a); Out(b); Out(c); Out(c); Out(c)

- $\text{Out}(a) \parallel \text{Out}(a) \parallel \text{Out}(a) \parallel \text{Out}(b) \parallel \text{Out}(b) \parallel \text{Out}(c) \parallel \text{Out}(c) \parallel \text{Out}(c)$
- *Esperimento Sequenziale (ordine)-Esperimento in Parallelo (disordine):*
 - k estrazioni *successive* da una scatola che contiene sempre tutti i caratteri di A (equivalentemente lancio in *successione* di k palline distinguibili in urne ciascuna delle quali è etichettata da un carattere diverso di A).
 - k estrazioni *in parallelo* di caratteri da k scatole che contengono tutti i caratteri (equivalentemente lancio in *parallelo* di k palline indistinguibili in urne ciascuna etichettata da un diverso carattere dell'alfabeto).

Purtroppo in questo esempio il principio del pastore non è applicabile.

3. **Terzo esempio:** dato un numero naturale n , fissiamo k numeri r_1, r_2, \dots, r_k tali che $r_1 + \dots + r_k = n$.

Permutazioni di un insieme A con n elementi (ordine)

versus

Partizioni $(X_i : 1 \leq i \leq k)$ di A con $|A| = n$ e $|X_i| = r_i$ ($i = 1, \dots, k$) (disordine)

- *Esperimento Sequenziale (ordine)-Esperimento in Parallelo (disordine)*
 - k estrazioni successive in un'urna con n elementi distinti, dove la prima estrazione pesca in *sequenza* r_1 elementi, etc.
 - k estrazioni successive in un'urna con n elementi distinti, dove la prima estrazione pesca in *parallelo* r_1 elementi, etc.

Possiamo applicare il principio del pastore. Spieghiamo il procedimento con un esempio. Fissiamo $n = 7$, $r_1 = 2$, $r_2 = 3$ e $r_3 = 2$, da cui segue $r_1 + r_2 + r_3 = 7$. Consideriamo la partizione $\{1, 5\}, \{2, 3, 6\}, \{4, 7\}$ di $[1, n]$ le cui classi di equivalenza hanno rispettivamente $r_1 = 2$, $r_2 = 3$ e $r_3 = 2$ elementi. Quante sono le permutazioni dell'insieme $\{1, 2, 3, 4, 5, 6, 7\}$ che corrispondono alla partizione $\{1, 5\}, \{2, 3, 6\}, \{4, 7\}$? Sono pari al numero delle permutazioni dell'insieme $\{1, 5\}$ moltiplicato il numero di permutazioni di $\{2, 3, 6\}$ moltiplicato il numero di permutazioni di $\{4, 7\}$: in totale $2!3!2! = 24$ permutazioni. Per esempio, entrambe le permutazioni 5267134 e 1627534 corrispondono alla data partizione. In generale, abbiamo

$$|\text{Permutazioni di } [1, n]| / |\text{Permutazioni di } \hat{r}_1| \times \dots \times |\text{Permutazioni di } \hat{r}_k| = n! / r_1! \dots r_k!$$

Il precedente numero si chiama *coefficiente multinomiale*.

4. **Quarto esempio (facoltativo):** Abbiamo visto prima la contrapposizione: funzioni da \hat{k} in $[1, n]$ versus multinsiemi. Però non era applicabile il principio del pastore. Si può modificare la contrapposizione nel modo seguente [2, Sezione 11.2].

Consideriamo delle urne speciali che possono contenere stringhe iniettive di palline distinguibili invece che insiemi di palline distinguibili. Se lancio una nuova pallina a_0 dentro un'urna che contiene la stringa $a_1 a_2 \dots a_r$, ho $r + 1$ possibilità:

$$a_0 a_1 a_2 \dots a_r; \quad a_1 a_0 a_2 \dots a_r; \quad \dots \quad a_1 a_2 \dots a_r a_0.$$

Lancio in successione di k -palline distinguibili in n urne speciali. In totale ho $n(n+1)(n+2) \dots (n+k-1)$ possibilità.

versus

Lancio in parallelo di k -palline indistinguibili in n urne speciali oppure no. In totale ho $(n+k-1)_k / k! = \binom{n+k-1}{k}$ possibilità, che è quindi il numero di k -multinsiemi.

Esempio di 4 urne e 8 palline: 362|58|1|47 versus $\{1, 1, 1, 2, 2, 3, 4, 4\}$. La prima urna contiene 362, la seconda 58, la terza 1 e l'ultima 47.

5. **Quinto esempio (facoltativo):** Permutazioni di un $k+n-1$ -insieme *versus* k -multinsiemi di un n -insieme.

Esempio: sia $k=8$ e $n=4$. Consideriamo l'alfabeto $\{1, \dots, 8, a, b, c\}$.

$$362a58c1b47 \text{ versus } \{1, 1, 1, 2, 2, 3, 4, 4\}$$

$$473b61a2c58 \text{ versus } \{1, 1, 1, 2, 2, 3, 4, 4\}$$

Devo quindi dividere il numero di permutazioni $(k+n-1)!$ per il numero di permutazioni dell'insieme \hat{k} ($\hat{8}$ nell'esempio) e per il numero di permutazioni dell'insieme $\widehat{n-1}$ ($\{a, b, c\}$ nell'esempio).

6.4. Sottoinsiemi e coefficienti binomiali

In questa sezione cominciamo a studiare le proprietà notevoli dei coefficienti binomiali $\binom{n}{k}$ al variare di n e k , ricordando che $\binom{n}{k}$ è il numero di k -sottoinsiemi di un n -insieme.

1. **Nuovo calcolo del valore del coefficiente binomiale** $\binom{n}{k}$. Un k -sottoinsieme $\{a_1, \dots, a_k\}$ di un n -insieme A genera un numero k di $(k-1)$ -sottoinsiemi eliminando di volta in volta un elemento:

$$(\{a_1, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_k\}, a_i) \mapsto \{a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_k\}$$

In totale, al variare dei k -sottoinsiemi, ho $k\binom{n}{k}$ possibilità.

Viceversa, un $(k-1)$ -sottoinsieme $\{b_1, \dots, b_{k-1}\}$ di un n -insieme A genera un numero pari a $(n-k+1)$ di k -sottoinsiemi aggiungendo di volta in volta un elemento $c \in A - \{b_1, \dots, b_{k-1}\}$ tra gli $(n-k+1)$ elementi rimasti:

$$(\{b_1, \dots, b_{k-1}\}, c) \mapsto \{b_1, \dots, b_{k-1}, c\}$$

In totale, al variare dei $(k-1)$ -sottoinsiemi, ho $(n-k+1)\binom{n}{k-1}$ possibilità.

Le due corrispondenze che abbiamo descritto sono l'una inversa dell'altra, per cui abbiamo descritto una corrispondenza bigettiva tra l'insieme delle coppie (k -sottoinsieme, elemento del k -sottoinsieme) e l'insieme delle coppie ($(k-1)$ -sottoinsieme, elemento non appartenente al $(k-1)$ -sottoinsieme).

In conclusione, abbiamo la seguente uguaglianza notevole:

$$k\binom{n}{k} = (n-k+1)\binom{n}{k-1}$$

Iterando il ragionamento otteniamo

$$\binom{n}{k} = [(n-k+1)/k]\binom{n}{k-1} = [(n-k+1)/k][(n-k+2)/k-1]\binom{n}{k-2} = \dots = (n)_k/k!$$

2. **La formula di Stifel:** $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ ed il triangolo aritmetico di Pascal (o di Tartaglia?). [2, Sezione 11.1]

La formula di Stifel è importante perché dimostra una relazione tra i sottoinsiemi di un n -insieme ed i sottoinsiemi di un $(n-1)$ -insieme. Dimostriamola.

Sia A un n -insieme e $c \in A$ un fissato elemento. Allora i k -sottoinsiemi di A si dividono in due classi: quelli che contengono c

$$\{a_1, \dots, a_{k-1}, c\} \mapsto \{a_1, \dots, a_{k-1}\}$$

(che corrispondono ai $(k-1)$ -sottoinsiemi di $A - \{c\}$) e quelli che non contengono c

$$\{a_1, \dots, a_k\} \text{ con } c \neq a_i \text{ per ogni } i$$

(che corrispondono ai k -sottoinsiemi di $A - \{c\}$). Quindi abbiamo la formula di Stifel.

La formula di Stifel può essere utilizzata per calcolare ricorsivamente tutti i coefficienti binomiali. Possono essere disposti in un triangolo aritmetico, detto di Pascal dai francesi e di Tartaglia dagli italiani [2, Sezione 11.1, pag. 106]. Vi sono tantissime relazioni interessanti sui coefficienti binomiali. Qui di seguito ne riportiamo alcune:

- $\binom{n}{k} = \binom{n}{n-k}$ [4, Sezione 1.4]. Esiste una corrispondenza bigettiva tra i k -sottoinsiemi di un n -insieme A ed i suoi $(n-k)$ -sottoinsiemi:

$$\{a_1, \dots, a_k\} \mapsto A - \{a_1, \dots, a_k\}.$$

- Cardinalità delle parti di un n -insieme [2, Sezione 11.1]:

$$\sum_{k=0}^n \binom{n}{k} = 2^n$$

Ogni sottoinsieme di un n -insieme ha come cardinalità un numero compreso tra 0 ed n .

- I numeri $\binom{n}{k}$, fissato n , crescono con k che varia da 1 fino ad $(1/2) \times (n+1)$. Segue dalla relazione $\binom{n}{k} = [(n-k+1)/k] \binom{n}{k-1}$ che abbiamo dimostrato prima in questa sezione: $(n-k+1)/k \geq 1$ sse $k \leq n-k+1$ sse $k \leq (1/2) \times (n+1)$.

3. *Il teorema binomiale* [4, Sezione 1.5, pag. 11], [2, Sezione 11.3].

6.5. Numeri di Fibonacci [4, Sezione 1.6] [3, Cap. 13]

1. Il problema delle coppie di colombi [4, Sezione 1.6]: supponiamo che ogni coppia di colombi impieghi un mese per diventare adulta ed un secondo mese per procreare un'altra coppia. Se all'inizio della generazione abbiamo una sola coppia non adulta e nessun animale muore, quante coppie avremo dopo n mesi? Calcoliamo per doppia induzione il numero A_n di coppie adulte ed il numero N_n di coppie non adulte dopo n mesi.

$$N_0 = 1; \quad A_0 = 0; \quad N_{n+1} = A_n; \quad A_{n+1} = A_n + N_n = A_n + A_{n-1}$$

Quindi

$$A_0 = 0; \quad A_1 = 1; \quad A_{n+1} = A_n + A_{n-1}$$

Il numero totale di coppie F_n dopo n mesi:

$$F_n = A_n + N_n = A_n + A_{n-1} = A_{n+1}.$$

Quindi,

$$F_0 = 1; \quad F_1 = 1; \quad F_{n+1} = F_n + F_{n-1}$$

2. *Sottoinsiemi di* [1, n] *che non contengono due numeri consecutivi* [4, Sezione 1.6]

Proviamo a contare il numero G_n dei sottoinsiemi di un n -insieme che non contengono due numeri consecutivi. Li possiamo dividere in due classi, la classe dei sottoinsiemi che contengono n ma non contengono due numeri consecutivi (così non possono contenere neanche $n-1$) e la classe dei sottoinsiemi che non contengono nè n nè due numeri consecutivi. La prima classe ha G_{n-2} elementi, mentre la seconda classe ne ha G_{n-1} . In totale, abbiamo la seguente relazione di ricorrenza:

$$G_0 = 1; \quad G_1 = 2; \quad G_n = G_{n-1} + G_{n-2}$$

Calcoliamo ora il valore di G_n , partendo dal numero $G_{n,k}$ dei k -sottoinsiemi di un n -insieme che non contengono due numeri consecutivi. Quest'ultimo numero è pari al numero di n -stringhe di 0,1 in cui occorre k volte 1 (e $n-k$ volte 0) e non ho due 1 consecutivi. Si calcola così: prima consideriamo $n-k$ cifre 0 scritte di seguito poi interponiamo k cifre 1 in modo tale che due non siano consecutive.

$$G_n = \sum_{k \geq 0} G_{n,k} = \sum_{k \geq 0} \binom{n-k+1}{k}$$

I numeri di Fibonacci sono

$$F_0 = 1; \quad F_1 = 1; \quad (n \geq 2) \quad F_n = G_{n-1} = \sum_{k \geq 0} \binom{n-k-1}{k}$$

3. La somma delle diagonali di angolo 45 gradi del triangolo aritmetico sono i numeri di Fibonacci [4, Sezione 1.6]. Segue dalla relazione $F_n = \sum_{k \geq 0} \binom{n-k-1}{k}$.

I primi numeri di Fibonacci sono:

1 1 2 3 5 8 13 21 34 55 89....

che corrispondono alla somma delle diagonali di angolo 45 gradi del triangolo aritmetico:

```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 25 25 21 7 1
    
```

Diagonale 0: 1

Diagonale 1: 1

Diagonale 2: $1 + 1 = 2$

Diagonale 3: $1 + 2 = 3$

Diagonale 4: $1 + 3 + 1 = 5$

Diagonale 5: $1 + 4 + 3 = 8$

Diagonale 6: $1 + 5 + 6 + 1 = 13$

Diagonale 7: $1 + 6 + 10 + 4 = 21$

4. Proprietà dei numeri di Fibonacci (facoltativo):

- $gcd(F_n, F_{n+1}) = 1$ [3, Teorema 13.1]
- $F_{m+n} = F_{m-1}F_n + F_mF_{n+1}$ (Facile per induzione su n)
- $gcd(F_n, F_m) = F_{gcd(n,m)}$ [3, Teorema 13.3] + Lemma che lo precede.
- (Formula di Binet [4, Sezione 1.6 pag. 14]) Esiste una progressione geometrica $1, x, x^2, \dots, x^n, \dots$ che verifica la relazione di ricorrenza della successione di Fibonacci? Se si, allora

$$x^n = x^{n-1} + x^{n-2}$$

cioè

$$x^2 = x + 1$$

cioè x deve essere radice della precedente equazione. Essa si può riscrivere come segue:

$$x^2 = x + 1 = xF_1 + F_0$$

e più in generale

$$x^{n+1} = xF_n + F_{n-1} \quad (\text{per induzione})$$

Le radici dellequazione $x^2 - x - 1$ sono

$$\alpha = (1 - \sqrt{5})/2; \quad \beta = (1 + \sqrt{5})/2$$

Quindi valgono le identità

$$F_n = (\beta^{n+1} - \alpha^{n+1})/(\beta - \alpha)$$

cio'è

$$F_n = (1/\sqrt{5})[((1 + \sqrt{5})/2)^{n+1} - ((1 - \sqrt{5})/2)^{n+1}]$$

6.5.1. Zeckendorfs Theorem (1939) and data compression

Edouard Zeckendorf, a Belgian amateur mathematician, discovered the following theorem in 1939. In 1960, David E. Daykin proved that the Fibonacci sequence is the only one which satisfies the theorem; in other words, the theorem asserts a defining property of the Fibonacci sequence!

Theorem 6.1. [3, Teorema 13.4] *Ogni numero intero positivo si scrive in maniera univoca come somma finita di numeri di Fibonacci distinti e non consecutivi. Più precisamente, per ogni intero positivo n , esiste un'unica stringa finita binaria $b_1b_2 \dots b_t$ ($t \geq 1$), senza uni consecutivi tale che $n = \sum_{i \geq 1} b_i F_i$.*

Quindi i numeri di Fibonacci possono essere utilizzati come sistema numerico (per esempio, $17 = 101001 = F_1 + F_3 + F_6 = 1 + 3 + 13$) e non solo.

Data Compression: Letters, in everyday use, occur with very pronounced frequencies: E most commonly, then T, then A, etc. Sending digital messages (using binary digits) is quicker if the letters are encoded with fewer bits for the more frequent letters. In Fibonacci coding, invented by Alberto Apostolico and Aviezri Fraenkel in 1985, the string 101001 can encode G, the 17th most common letter: an extra 1 is added at the end to signal letter boundaries. THEOREM is encoded in standard ASCII as seven 8-bit bytes: 56 bits. The Fibonacci code makes a 46% saving with only 30 bits: 011000011111011100011111000011. Per completezza riportiamo la codifica delle lettere

ORDER LETTER CODE

1 E 11
 2 T 011
 3 A 0011
 4 O 1011
 5 I 00011
 6 N 10011
 7 S 01011
 8 H 000011
 9 R 100011
 10 D 010011
 11 L 001011
 12 C 101011
 13 U 0000011
 14 M 1000011
 15 W 0100011
 16 F 0010011
 17 G 1010011
 18 Y 0001011
 19 P 1001011
 20 B 0101011
 21 V 00000011
 22 K 10000011
 23 J 01000011
 24 X 00100011
 25 Q 10100011
 26 Z 00010011

6.6. Some applications of Fibonacci numbers

The Fibonacci numbers are important in the run-time analysis of Euclid's algorithm to determine the greatest common divisor of two integers: the worst case input for this algorithm is a pair of consecutive Fibonacci numbers.

The Fibonacci numbers and principle is also used in the financial markets. It is used in trading algorithms, applications and strategies. Some typical forms include: the Fibonacci fan, Fibonacci Arc, Fibonacci Retracement and the Fibonacci Time Extension.

Fibonacci numbers are used by some pseudorandom number generators.

Fibonacci numbers are used in a polyphase version of the merge sort algorithm in which an unsorted list is divided into two lists whose lengths correspond to sequential Fibonacci numbers - by dividing the

list so that the two parts have lengths in the approximate proportion.

Fibonacci numbers arise in the analysis of the Fibonacci heap data structure.

A one-dimensional optimization method, called the Fibonacci search technique, uses Fibonacci numbers.

In music, Fibonacci numbers are sometimes used to determine tunings, and, as in visual art, to determine the length or size of content or form (music)—formal elements. It is commonly thought that the first movement of Bela Bartok's "Music for Strings, Percussion, and Celesta" was structured using Fibonacci numbers.

6.7. Il principio del crivello o di inclusione-esclusione [2, Sezione 11.4] [4, Sezione 3.1, 3.2, 3.3]

Si comincia con [2, Teorema 11.4, pag. 113], che è la formula di Da Silva, e poi si considera la formula di Sylvester [4, Sezione 3.3, pag. 73]; si finisce con degli esempi.

1. Permutazioni senza punti fissi [4, Sezione 3.3, pag. 74]
2. La funzione ϕ di Eulero [4, Sezione 3.4] e [2, Sezione 11.5]
3. Il problema dei ménages [4, Sezione 3.5]

6.8. Partizioni [2, Cap 12] [4, Sezione 1.9]

REFERENCES

1. F. Bellissima, F. Montagna. Matematica per l'informatica: aritmetica e logica, probabilità, grafi. Carocci Editore, 2006.
2. Norman L. Biggs: Discrete Mathematics, Oxford University Press, 2002.
3. David M. Burton. Elementary Number Theory. Allyn and Bacon, 1980.
4. M. Cesaroli, F. Eugeni, M. Protasi. Elementi di Matematica Discreta. Zanichelli, 1988.